IMPLEMENTACIÓN DE MULTIPLICADOR DE 4 BITS

Juan De La Puente, Joaquín Hernán Costa, Edgardo Ricci

UIDET – CeTAD, Calle 116 y 48 Depto. de Electrotecnia (2do piso)
Facultad de Ingeniería, UNLP. La Plata (1900). Email: josrap@ing.unlp.edu.ar

INTRODUCCIÓN

A través del diseño de circuitos digitales básicos se pretende la realización de un multiplicador digital de 4 bits. Esta propuesta didáctica se enmarca en un proyecto mayor en el que se utilizan diseños de arquitecturas de microprocesadores sencillos en procesos inteligentes de comunicación y decisión en módulos de micro-sistemas.

Tales arquitecturas se diseñan paulatinamente a partir de pequeñas partes, en tareas individuales para luego integrarlos en las partes constitutivas mayores (Unidades Aritmético Lógicas, Unidades de Control, etc.). Como resultado final se efectúa una comparación entre ambas tecnologías de implementación, destacando tanto las ventajas como las dificultades que surgen de cada una de ellas.

PARTE EXPERIMENTAL

Diseño de Componentes Básicos

El multiplicador partió de la realización de las librerías de celdas básicas necesarias para la realización del diseño tales como compuertas NAND, NOR y negadoras. Se generaron los circuitos esquemáticos que luego fueron asociados a símbolos y posteriormente fueron simulados para comprobar su funcionamiento. Por simplicidad aquí se mostrará únicamente el resultado obtenido para una compuerta negadora.

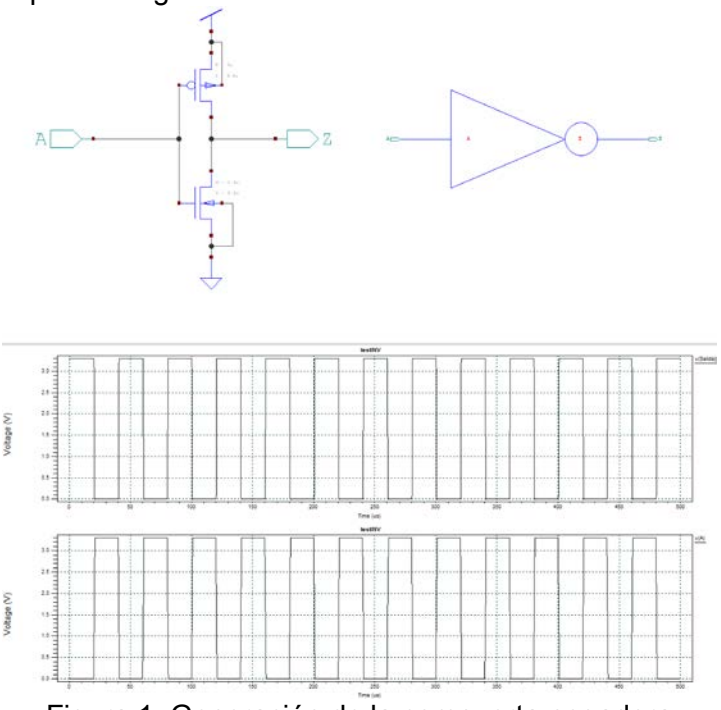

Figura 1: Generación de la compuerta negadora

En la figura 1 se muestra dicha compuerta desde el circuito esquemático hasta su simulación. Se observó que para una entrada en alto (1 lógico) la salida se encuentra en bajo (0 lógico) y viceversa. Dicho comportamiento coincide con el funcionamiento esperado para el negador. Es

importante tener en cuenta que en este caso el tiempo de retardo de la inversión es muy pequeño, del orden de las decenas de *picosegundos*, debido a que no hay ninguna compuerta lógica adicional entre la señal de entrada y salida (sólo se encuentran los 2 transistores propios del negador). En las secciones siguientes se verá que entre la señal de entrada y la señal de salida, habrá alrededor de 100 transistores, lo cual provocará un retardo entre la salida y la entrada que será un limitante dado por la tecnología en la que se trabajó. Es por esto que se deben tener en cuenta estos retardos en la implementación y compensarlos al máximo para lograr un correcto funcionamiento [1], [2]

Este procedimiento se repitió para el resto de compuertas lógicas básicas de manera de crear una librería propia para efectuar la implementación del multiplicador.

Multiplicación de números binarios

La multiplicación de números binarios se realiza de la misma manera que para números decimales con la diferencia de que en binario al tener representados los números unos y ceros, los productos parciales son más sencillos. Un factor importante será que, cuando se desee implementar un producto de números de 4 bits habrá cuatro productos parciales que deberán ser resueltos por el circuito a implementar [3].

Elección del diseño a implementar

Línea de Selección	Salida
00	A:GND
01	B:X
10	C:2X
11	D:3X

Tabla 1: Líneas de entrada de los multiplexores.

Para la implementación se propone un circuito que utiliza dos multiplexores 4:1, en el cual el multiplicando (en adelante X) se conecta en las líneas de entrada de los multiplexores y los dos bits menos significativos del multiplicador (en adelante Y) se conectan en las líneas de selección de un multiplexor (que llamaremos mux LSB) y los más significativos en las líneas de selección del multiplexor restante (que llamaremos mux MSB). De esta manera se realizan dos productos parciales por multiplexor, que luego, con los correspondientes corrimientos se sumarán dando el resultado del producto entre X e Y.

En la tabla 1 se muestra cómo se implementaron las líneas de entrada a los multiplexores, la línea A es cero siempre por lo que se encuentra conectada a tierra, la línea B corresponde al multiplicando X, la línea C es X desplazado 1 bit a izquierda y la D es la suma de las líneas B y C, debiendo utilizarse un sumador de 5 bits dado que se consideró el desplazamiento de la línea C.

Es importante ver que dadas las características de las líneas C y D se deben adicionar dos bits conectados a tierra en las líneas A y B para contemplar tanto el desplazamiento de la tercer entrada como el acarreo que arroja el sumador de la cuarta entrada; esto resulta en que los multiplexores deben tener un bus de 6 bits por entrada.

Del mismo modo resultó necesario que a la salida de los multiplexores se agreguen dos líneas a tierra (las más altas para mux LSB y las más bajas para el mux MSB) de manera de entrar al sumador de salida con 8 bits por operando. Los sumadores utilizados en esta implementación son del tipo *Ripple Carry*, lo cual genera un retardo de propagación que influye en las líneas de entrada de los multiplexores y debe ser compensado con un retardo equivalente en las otras líneas de entrada para asegurar que los datos lleguen al mismo tiempo y el resultado sea correcto.

Por otra parte el sumador de salida debe ser de 8 bits ya que se contempla el caso en el que X e Y sean 1111. En las figuras 2 y 3 se observa el circuito esquemático resultante del diseño a

implementar, se agregó además una línea para habilitar o deshabilitar el circuito dado que al no haberse implementado un árbol de retardos en las entradas y además por simplicidad del circuito se decidió no utilizar *Latches*, surgió la necesidad de contar con un control asincrónico de la etapa de multiplexores para asegurar la estabilidad de los datos de entrada y salida.

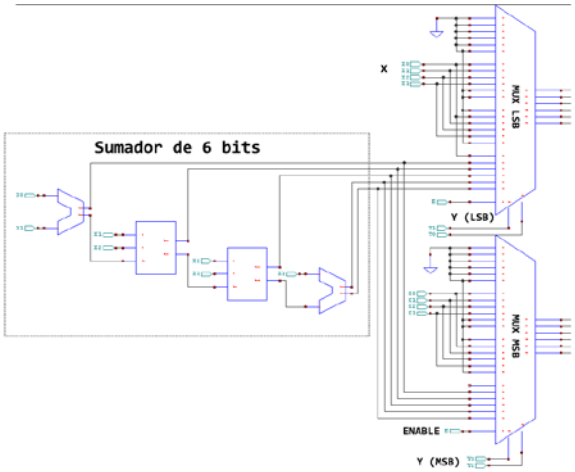

Figura 2: Circuito esquemático de la etapa de entrada.

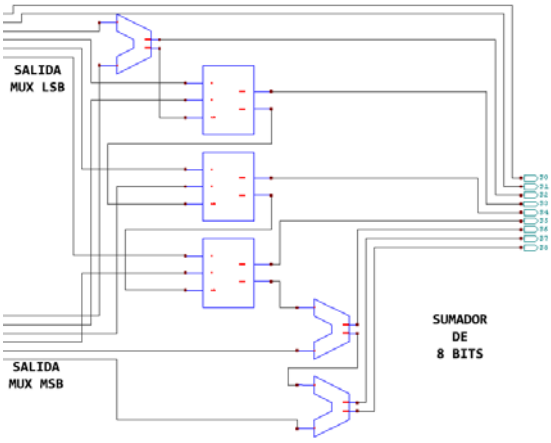

Figura 3 : Circuito de la etapa de salida.

Etapa de Simulación

Ya realizado el esquemático del multiplicador se realizó la simulación de todas las combinaciones posibles de multiplicaciones. Por simplicidad en la tabla 2 se muestran sólo cuatro casos de los 256 posibles. Durante la simulación, se prestó especial atención al mayor producto posible (1111x1111), cuidando que el desborde del bit más significativo no se pierda. Se observó además a la línea de enable y se comprobó que cuando la línea está en 0 a la salida vale cero independientemente de los valores de X e Y como era esperado.

X_2	X_1	X_2	X_0	Enable	Y_3	Y_2	Y_1	Y_0	Salida
0	0	0	0	1	0	0	0	0	00000000
1	1	1	1	1	1	1	1	1	11100001
0	1	1	0	1	1	0	1	1	01000010
1	1	0	0	0	0	1	0	1	00000000

Tabla 2: Algunos resultados de la simulación

Los datos ingresados al multiplicador fueron configurados de manera tal que duren 20 µ*s*, es decir que cada 20 µ*s* se realiza un producto diferente. Cabe aclarar que la salida está simulada por bits, siendo el bit más significativo y el bit el menos significativo. Como se puede ver en la figura 4, la salida vale cero a partir de los 60 µ*s* debido a que el *Enable* está en cero.

Una vez comprobada la funcionalidad del dispositivo implementado, se medirá el peor tiempo de retardo del multiplicador, lo que ocurre cuando se realiza el producto entre 1111 y 1111, ya que es en este caso que se da la mayor utilización de compuertas en el dispositivo. Las señales de entrada fueron configuradas de manera tal que el tiempo que tardan en pasar de 0 a 1 o viceversa es de 1 µ*s*. Se tomó esta decisión para evitar los posibles *glitches* que se generan en el simulador por utilizar tiempos más rápidos.

Para llevar a cabo la medición, se comienza haciendo el producto entre 0000 y 0000 y luego se conmutan tanto los bits X como los Y a 1111 para poder medir el tiempo de la transición, ya que al multiplicar por cero no se utiliza ninguna de las compuertas del dispositivo. El bit a analizar es el , ya que es el que más tarda en alcanzar un valor estable, por lo tanto éste es el que genera el peor caso.

Para determinar el tiempo de retardo, se debe hacer la diferencia entre el tiempo en que la señal de entrada alcanza un 10% de su valor máximo y la de salida alcanza un 90% de su máximo valor Dichos valores corresponden a: 330 mV para el 10% del valor máximo, y 2,97 V para el 90% del valor máximo. Como se puede ver en la figura 5, los cursores verticales fueron posicionados en los valores mencionados anteriormente, y se hizo coincidir los cursores horizontales con los verticales para tomar los tiempos en que se alcanzaban dichos valores. Se determinó que la señal de entrada alcanza un 10% del valor máximo a los 80,1 µ*s* y la señal de salida alcanza un 90% de su máximo valor a los 80,61 µ*s*. Haciendo la diferencia entre estos valores se obtuvo que el tiempo de retardo es de 0,51 µ*s*.

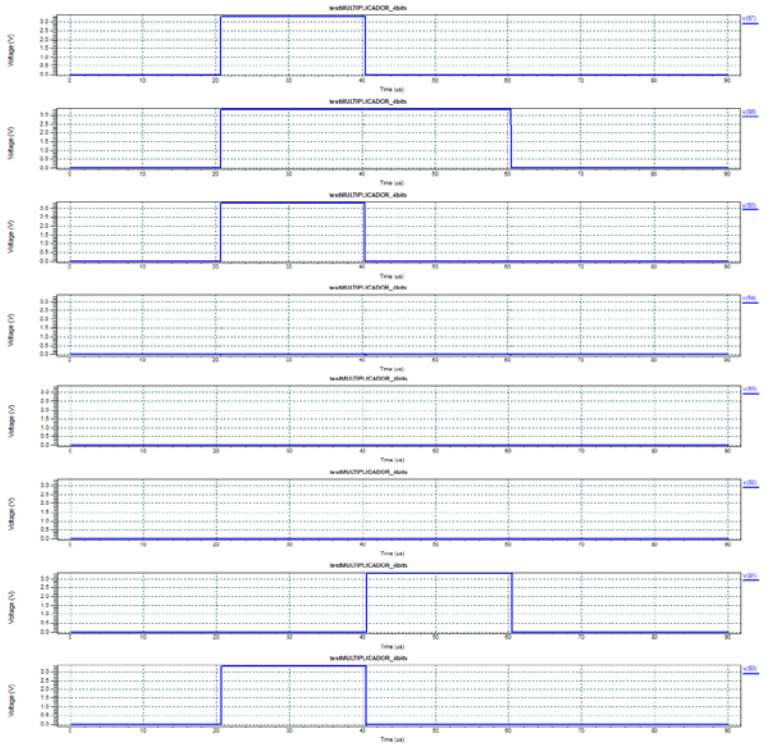

Figura 4: Simulación del multiplicador.

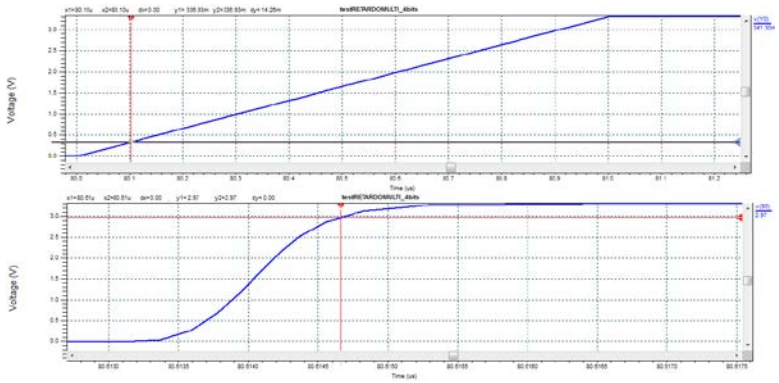

Figura 5: Medición de retardo del multiplicador.

Diseño de *Layouts*

Sabiendo que el dispositivo implementado funciona correctamente, se pasa a desarrollar el *Layout* con el software de diseño utilizando una tecnología de 500 nm. Para ello, se desarrolló una

librería que contiene todas las celdas lógicas digitales básicas necesarias para hacer los componentes que forman parte del multiplicador.

Una vez elaborada la librería, se llamó de ésta a las distintas celdas, y conectándolas entre sí, se consiguieron los dos tipos de sumadores requeridos (*Half Adder* y *Full Adder*) y el multiplexor 4:1. Con estos elementos incorporados en la librería y llevando a cabo el mismo procedimiento anterior, se desarrolló el multiplicador. Por motivos de espacio, sólo se mostrará el *Layout* del multiplexor 4:1 en la figura 6.

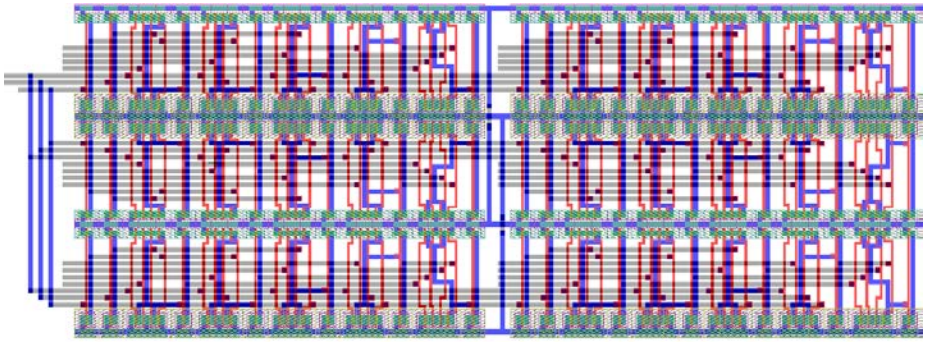

Figura 6: *Layout* de un multiplexor 4:1 de 6 bits.

Implementación en Verilog

Dado que Verilog es un lenguaje que permite programar de manera modular se partió del diseño elegido y se programaron por separado los multiplexores y los sumadores que luego se instanciaron en el código del multiplicador, el que describe las conexiones entre éstos elementos constitutivos. Como la implementación en lenguajes de descripción de hardware requiere que exista una estructura jerárquica entre los módulos se aseguró que el código del multiplicador esté en el tope de esta jerarquía, de manera de evitar errores tanto en la compilación como en la conexión entre módulos. En la figura 7 se observa que el diagrama en bloques resultante coincide con el circuito esquemático obtenido anteriormente.

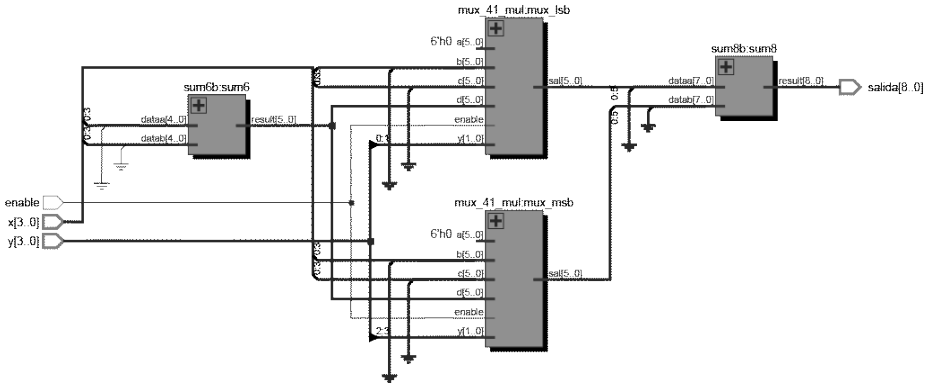

Figura 7: Diagrama en bloques del multiplicador implementado en Verilog.

Luego se verificó si los módulos internos coinciden con lo esperado. En el caso de los multiplexores (figura 9) se implementó una línea de *Enable* que permite inhibir la etapa de entrada como ya se expresó anteriormente. Por el lado de los sumadores (figura 10) el diseño de los mismos es idéntico, un *Half Adder* en el LSB y en el MSB y en los bits intermedios *Full Adders*, dichos sumadores se encuentran conectados a través de la línea de acarreo salvo en el *Half Adder* del MSB en el cual el carry out del sumador anterior es uno de los operandos.

Una vez comprobada la correspondencia entre el diseño original y lo realizado en el lenguaje de descripción de hardware se desea probar su funcionamiento. Para tal fin se simuló el circuito

digital y para ello se generó un nuevo código de Verilog del banco de pruebas de la implementación [4] y [5], en el cual se especifican los valores de las entradas del circuito así como el tiempo con el cual irán cambiando dichas entradas (cada 20 μs). Esto es así ya que se pretende verificar el funcionamiento del multiplicador en distintas circunstancias de interés ya expuestas en la tabla de valores ingresados en la simulación realizada en la sección anterior.

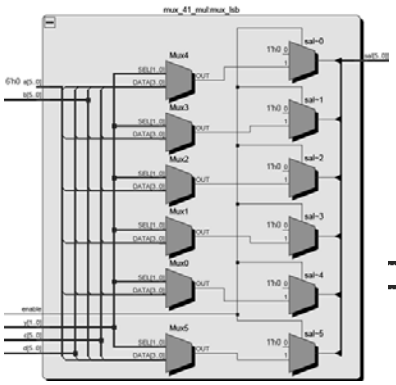

Figura 9: Esquemático del multiplexor

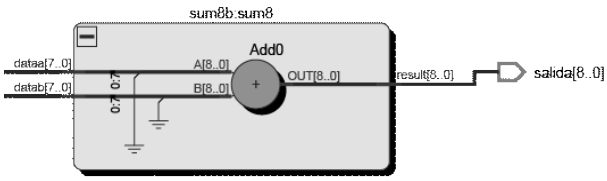

Figura 10: Esquemático del sumador.

En la figura 8 se muestran los resultados obtenidos y se comprueba que la implementación del multiplicador funciona como se esperaba, sin *Overflow* en el caso de la multiplicación más grande, resultado igual a cero en el caso de que alguno de los factores sea cero y las líneas de *Enable* funcionan correctamente.

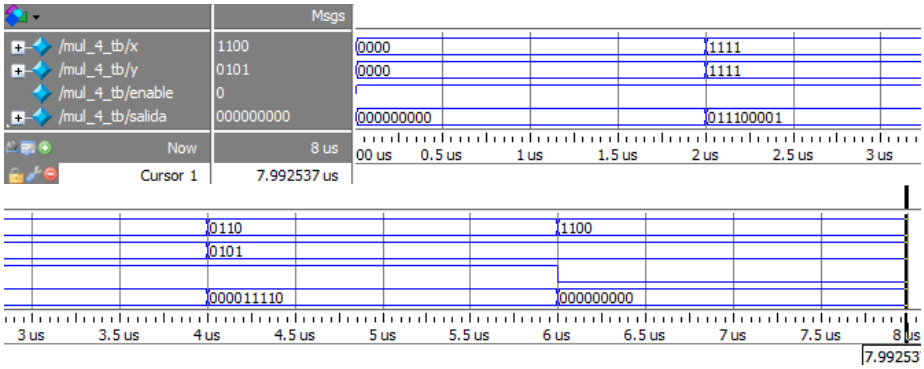

Figura 8: Simulación de la implementación en Verilog.

CONCLUSIONES

Con los resultados obtenidos en el presente trabajo se lograron cumplir los objetivos planteados, adquiriéndose conocimiento sobre el uso de herramientas así como el conocimiento de las etapas de implementación y los criterios a utilizar en función de las especificaciones deseadas tanto como las consideraciones a tener en cuenta frente a las dificultades que se presentan.

Como se mencionó en la introducción este proyecto formará parte de un micro-sistema más complejo; más específicamente será uno de los módulos constitutivos de una unidad aritmético lógica.

BIBLIOGRAFÍA

[1] P. Julian, *Circuitos Integrados Digitales CMOS. Análisis y Diseño*, ed. Alfaomega. 2015.
[2] R. Baker, *CMOS Circuit Design, Layout and Simulation*, 3rd. edition, Wiley – IEEE Press. 2010.

[3] R. Tocci, *Sistemas Digitales: Principios y aplicaciones*, ed. Prentice Hall. 1996.
[4] S. Planitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, 2nd edition, Prentice Hall PTR, 2001. ISBN : 0-13-044911-3
[5] Altera Corporation, *ModelSim-Altera Software Simulation User Guide*, January 2013.