Multiplicador de números enteros de 16-bits con signo

Camilo Álvarez Muñoz, Juan Camilo Arteaga Ibarra

Dpto. de electrónica y telecomunicaciones, Universidad de Antioquia, Medellín, Colombia

1. Abstract

Durante el desarrollo de este laboratorio se diseñó un multiplicador de números enteros de 16-bits con signo mediante el lenguaje de descripción de hardware SystemVerilog y el sistema de desarrollo DE10-Lite. Para la construcción de este multiplicador se prohibió el uso del operador *, por lo cual se debió investigar acerca de los distintos métodos que permiten la multiplicación de números de n-bits con signo y además de esto se implementaron dos bloques principales que permitieran la comunicación de datos, el almacenamiento de los operandos, el resultado, y el funcionamiento de los circuitos combinacionales y secuenciales requeridos para manipular el hardware de la DE10-Lite.

Keywords: Multiplicador, Maquina de estados, registros, FPGA.

2. Objetivos

- Diseñar un sistema capaz de multiplicar números enteros con signo ingresados mediante los periféricos de la FPGA De10-lite sin hacer uso del operador *.

- Familiarizarse con el lenguaje de verificación y descripción de hardware SystemVerilog.

3. Diseño e implementación

Con el fin de realizar un sistema organizado, se realizó la implementación de dos bloques principales: la unidad de control y la unidad de datapath. Estas unidades son las que manipularán el sistema para traer a la vida las múltiples funciones que permitirán la correcta implementación del diseño del multiplicador en el sistema de desarrollo DE10-lite.

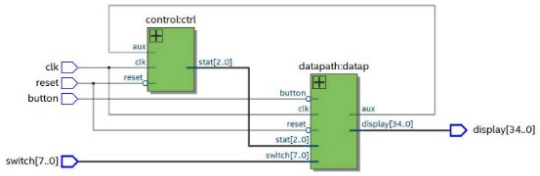

Imagen 1. Diagrama de conexiones del sistema.

Unidad de control: Los operandos A, B y el resultado de la multiplicación S son números de 16 bits; estos operandos solo pueden ingresarse al sistema a través de los 10 switches y 2 botones disponibles en la FPGA. Por lo tanto, es necesario hacer uso de 8 switches que representaran un número binario de 8 bits, este número definirá los 8 bits más significativos o los 8 menos significativos de los operandos A y B; estos números binarios de 8 bits serán ingresados consecutivamente a través de la unidad de datapath hasta completar los operandos de 16 bits A y B para realizar la multiplicación. Para tener control acerca de la parte del número que se esté ingresando, en esta unidad se diseñó una máquina de estados finitos que informa a

la unidad de datapath sobre el estado actual del sistema, para así generar la correcta visualización de los datos y posteriormente la realización de la multiplicación. Este control fué posible con un botón que cambia de estado al ser pulsado y otro que reinicia el sistema (reset). La máquina de estados está conformada por seis estados, siendo el primero un estado en el que se definen todas las variables a 0, el segundo estado pide y recibe los bits menos significativos del primer arreglo de bits, el tercer estado pide los bits más significativos del arreglo anterior, los siguientes dos estados realizan lo mismo para el segundo arreglo y el último estado da el resultado de la operación de multiplicación de los 16 bits, mostrando en el display siete segmentos.

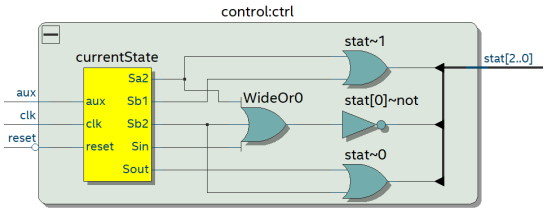

Imagen 2. Diagrama de conexiones unidad de control.

También es posible observar la máquina de estados utilizada:

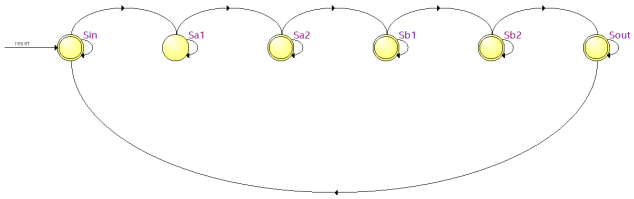

Imagen 3. Máquina de estados de la unidad de control.

Unidad de Datapath: En esta unidad se encuentran todas aquellas operaciones que permiten el funcionamiento aritmético, lógico y de almacenamiento en el sistema; el claro ejemplo de esto es el almacenamiento de los operandos A y B ingresados sucesivamente a través de los 8 switches en la FPGA. Ambos números de 16-bits con signo serán ingresados a un submódulo de multiplicación, este realizará la multiplicación de los números con ayuda del método conocido como "Shift-and-Add Multiplication"; posteriormente el resultado de la multiplicación será retornado al datapath para su visualización.

La unidad de control trasmitirá al datapath una señal que define el estado en el que se encuentra el sistema, esta señal permite la correcta visualización de los datos. Para esto se diseñó un submódulo decodificador que lee los datos de los operandos A y B y el resultado de la multiplicación para traducirlo a un valor que será representado en formato Hexadecimal a través de los displays 7 segmentos disponibles en la DE10-Lite.

Multiplicador números de n-bits con signo: este módulo realiza el proceso de la multiplicación de los dos números enteros binarios de 16 bits, haciendo uso del método mencionado anteriormente. En este método primero se obtiene los valores absolutos de los números a multiplicar, luego se procede a guardar el signo en una variable y se continua con la implementación del método "Shift-and-Add", iterando sobre cada uno de los bits de uno de los arreglos, realizando una suma con desplazamiento que permite la obtención del resultado deseado, para el cual finalmente se le hará una comparación y así saber si se le debe agregar el signo negativo. Este módulo fue realizado completamente de forma combinacional, por lo que en sí no necesita de ningún ciclo de reloj para su funcionamiento, sin embargo, para la toma

de los arreglos de bits usados en la multiplicación si es necesario el uso de ciclos de reloj, por lo cual se diseñó una señal auxiliar que actuase como reloj para la recolección de datos cada vez que se presionase un botón, de manera que cada flanco de subida de esta nueva señal indicase el cambio de estado y por ende la asignación de datos a las diferentes variables utilizadas, las cuales están siendo actualizadas constantemente.

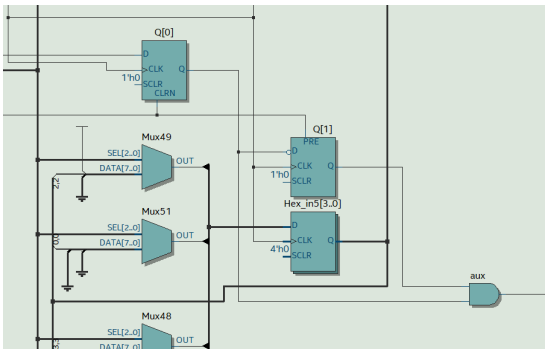

Imagen 4. Flip flops que generan la señal auxiliar al presionar el botón.

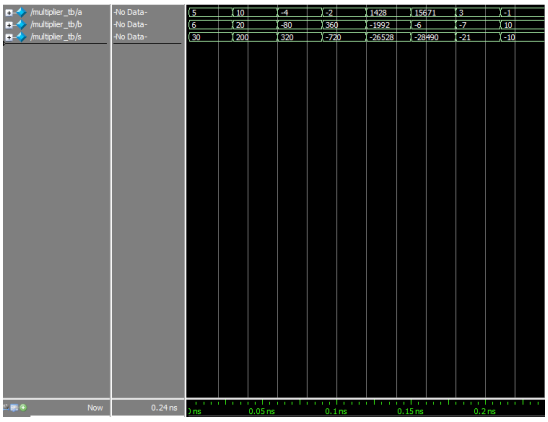

Imagen 5. Simulación del módulo de multiplicación.

Para la simulación se hizo uso de un delay de 30 pico segundos entre cada cambio de valor de los arreglos.

Decodificador 7-segmentos: Este submódulo recibe un número binario de 4 bits, este valor será traducido a un número de 7 bits que representa el respectivo valor en hexadecimal de los datos ingresados. Los operandos A, B y el resultado S son números de 16 bits, por lo cual fue necesario instanciar esta función 4 veces para traducir y mostrar correctamente el valor en hexadecimal correspondiente al número requerido a tiempo real. Para representar correctamente el estado en el que se encuentra el sistema fue necesario instanciar un quinto display 7 segmentos que rota entre los valores: 0(Reset), A, B o S.

en donde el número de ciclos del reloj sea crucial.

4. Conclusiones

- Uno de los grandes retos de la implementación del multiplicador era la importancia en minimizar el números de ciclos de reloj requeridos para realizar la operación; la implementación que fue realizada con el método “shift-and-add” se basaba en un ciclo combinacional, esta operación no puede ser realizada de forma instantánea por la FPGA, por lo tanto para realizar la multiplicación con ayuda del software se crearán múltiples sumadores y circuitos secuenciales virtuales que generarán un retraso en el sistema. A pesar de que este retraso no es grande, es importante tener en consideración situaciones así en futuros casos en donde se realicen sistemas más complejos

- La transición a un nuevo lenguaje como lo es SystemVerilog y el software QuartusPrime ha llevado a el encuentro con múltiples errores y dificultades durante el desarrollo de la programación de la práctica. Para solucionar muchos de estos errores fue necesario consultar en distintos medios como videos, blogs y ejemplos suministrados por el docente en clase.

- Al observar el diagrama generado por la función "Netlist Viewer – RTL Viewer", de la unidad de control, es posible notar que el estado Sa1 no aparece, por lo que se cree que es un error del software Quartus Prime, ya que, al desplegar la máquina de estados, esta es descrita de la manera correcta.

- Para el desarrollo del laboratorio fueron necesarias 8 horas de trabajo.