Practica 2 - Circuitos Aritméticos para Números Enteros

Autores: *Cristian David Araujo A., Maverick Sossa Tobón*

Electrónica Digital 2
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Universidad de Antioquia

Resumen

En este informe se da una muestra del diseño, implementación y simulación de un circuito digital en una FPGA destinado a la realización de operaciones aritméticas y lógicas básicas con números enteros ingresados manualmente. La parte central es el diseño de la Unidad Aritmético Lógica (ALU), en donde se usa el lenguaje de descripción de hardware System Verilog con el fin de tener un programa con a lógica del funcionamiento del sistema. Luego de simular con la herramienta ModelSim, se procede a realizar la imprementación con la tarjeta DE10-Lite.

Objetivos y Funcionalidad

El objetivo general de esta práctica es diseñar e implementar un sistema electrónico digital que pueda realizar operaciones aritméticas y lógicas básicas en números enteros. La Unidad Aritmética Lógica (ALU) será el corazón de este sistema y llevará a cabo las operaciones necesarias. La descripción de la ALU, junto con la lógica requerida para ingresar datos y mostrar resultados, se realiza utilizando el lenguaje de descripción de hardware SystemVerilog.

La Unidad Aritmética Lógica (ALU) está diseñada para realizar operaciones de suma, resta, AND y OR en operandos de N bits, pero para efectos practicos se utilizan 5 bits. Los valores de entrada (A y B) se ingresan a través de interruptores en la board DE10-Lite. La selección de operación se controla mediante un pulsador en la board, con resultados mostrados en displays de 7 segmentos y LEDs que indican códigos de condición como negativo, cero, carry y overflow. Además, el reloj y la señal de reinicio (Reset) son componentes clave para la operación en tiempo real de la ALU.

Diseño

En la sección de diseño, se presenta un diagrama de bloques para ofrecer una visión clara del circuito digital planificado. El diagrama de bloques (Figura 1) ilustra un generador de pulsos que recibe una señal llamada operSelect. Esta señal se utiliza para seleccionar la operación actual de la Unidad Aritmética Lógica (ALU) y cambia cada vez que se presiona el pulsador conectado a operSelect. El módulo Selector de Operación se encarga de codificar esta información para que la ALU comprenda la operación actual.

El núcleo del diseño es el módulo principal ALU, que realiza tanto operaciones lógicas como aritméticas. El resultado de estas operaciones se envía a un codificador que convierte la salida binaria en formato BCD (Binary-Coded Decimal). Luego, un decodificador BCD a 7 segmentos se

encarga de representar las unidades y las decenas en dos displays separados.

Además, se incorpora un tercer display conectado a un codificador que recibe una señal de una flag entregada por la ALU llamada "N". Esta señal se utiliza para mostrar el símbolo .en caso de que el número resultante sea negativo. Este diseño permite una representación clara y efectiva de las operaciones y sus resultados en el sistema.

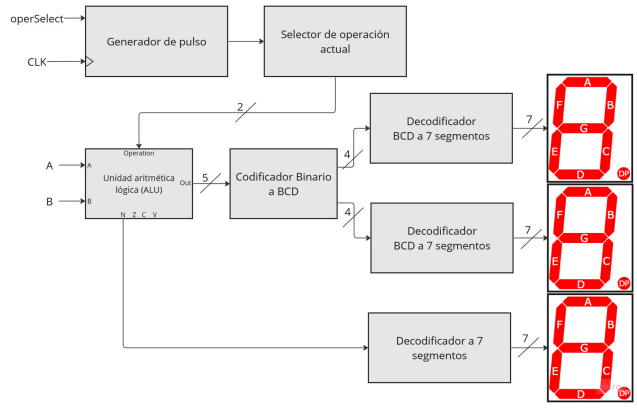

Figura **0-1**: Diagrama de bloques

Simulación

En proceso de simulación se realizó con el test bench, en código, mediante las herramientas de Quartus.

A forma de aclaración, el tb se hizo de tal forma que en 8 ciclos de reloj (clk) se completa un período de la señal del pulso.

La imagen inmediatamente inferior muestra una serie de operaciones.

La primera es una suma entre 15 y 7, la cual genera un overflow ya que 22 no está en el rango [-16, 15], por tanto el resultado obtenido no tendrá sentido.

La siguiente es una resta entre 15 y 1. Por el complemento a dos, la operación queda como si fuera una suma entre 15 y 31, en donde el carry generado es mostrado en las flags.

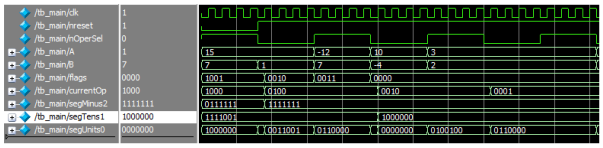

Figura **0-2**: Simulación del test bench

A manera de verificación de uno de los requisitos principales, en el cual se pide que la cantidad de sumatores a nivel de síntensis debe ser sólo uno en el módulo de la ALU, se muestra la siguiente figura.

El motivo por el que se muestran dos sumadores es por algo intrínseco al entorno de desarrollo, en donde para poder sumar el carry se vio en la necesidad de crear otro sumador.

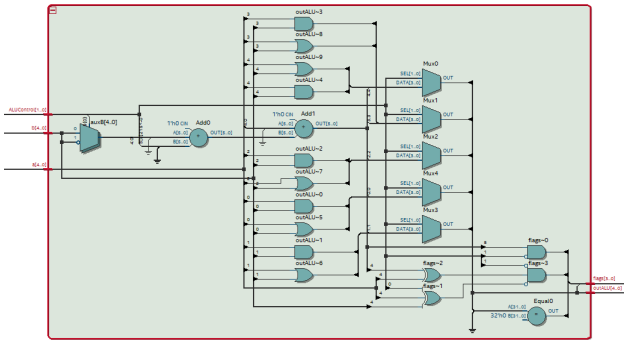

Figura **0-3**: Síntensis de la ALU

Entonces, se muestra una simulación que en función de los pulsos generados por nOperSel y de unas entradas A y B, realiza una operación aritmética o lógica en donde para la completa interpretación de los resultados se tienen las flags. También se tiene la síntesis de la ALU, con la cual se verifica que la cantidad de sumadores corresponden con los solicitados.

Conclusiones

- La creación de diagramas de bloques y una planificación cuidadosa son piedras angulares para el éxito en el diseño de sistemas digitales. En este proyecto, hemos experimentado de primera mano cómo la creación de diagramas de bloques nos ha brindado una visión general clara de cómo se interconectan los componentes y módulos. Esto no solo simplifica el proceso de diseño, sino que también reducen los errores. Al tener una representación gráfica de cómo se relacionan los elementos, podemos identificar posibles problemas y optimizar la estructura del sistema antes de comenzar la implementación.
- Entender las reglas sintácticas y semánticas del lenguaje SystemVerilog es esencial en el diseño de sistemas digitales. No obstante, es crucial tener en cuenta que la verdadera maestría de este lenguaje se adquiere a medida que enfrentamos problemas y desafíos en proyectos reales.