Laboratorio #3: Lógica Combinacional y Aritmética I

Jorge Agüero Zamora, Arturo Chinchilla Sánchez, Luis Murillo Rios
georgeaz56@gmail.com mchinchilla11@gmail.com luismuelino@gmail.com
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—Este laboratorio trata acerca del diseño y la confección de una Unidad Lógica Aritmética (ALU) de N bits. Durante la etapa de diseño se utiliza la metodología de Diseño Modular y para la creación de la ALU se utiliza el lenguaje de descripción de Hardware SystemVerilog. Además, con la ayuda de una FPGA se implementó una ALU de 3 bits, la cuál cuenta con las funciones de suma, resta, and, or, not, corrimiento lógico a la derecha e izquierda y corrimiento aritmético a la derecha. También con la ayuda de dos registros y la herramienta Quartus, se hace un análisis de ruta crítica para ALUs de 32, 64, 128 y 256 bits, donde se calcula las frecuencias máximas a la que pueden operar.**

***Palabras clave*—Tiempo de propagación, tiempo de contaminación, diseño modular, ALU**

I. Introducción

La Unidad Lógica Aritmética (o ALU por sus siglas en inglés) como su nombre lo dice, es la encargada de realizar las operaciones lógicas y aritméticas dentro de un procesador. Según [1], una ALU combina una variedad de operaciones matemáticas y lógicas, por ejemplo, una ALU podría realizar suma, resta y las operaciones de AND y OR. También hace referencia a que ésta forma el corazón de la mayoría los sistemas computacionales.

La ALU se implementa mediante un circuito combinacional, debido a que éstos generalmente presentan la característica de que son muy rápidos, ya que no necesitan esperar la sincronización del reloj, lo que ayuda a que en sistemas digitales más complejos como un microprocesador las operaciones se realicen a altas velocidades.

Algunas implementaciones de una ALU cuentan con otras salidas llamadas banderas (*flags*) que ofrecen información extra acerca del resultado de una operación hecha en la ALU. Las banderas generalmente están representadas por las letras N (*Negative*), Z (*Zero*), C (*Carry*) y V (*Overflow*), que indican respectivamente, si el resultado en la salida de la ALU tiene signo negativo, es cero, si el sumador generó un "$Carry$" de salida o un "$Overflow$".

Durante el diseño de los circuitos combinacionales, se debe tomar en cuenta temas como:

- Los tiempos de propagación de las señales en cada uno de los dispositivos que integran el circuito. En [1] se define el tiempo de propagación como el tiempo máximo desde que una entrada cambia hasta que la(s) salida(s) alcanzan su valor final.
- El tiempo de contaminación que [1] lo define como el tiempo mínimo desde que una entrada cambia hasta que cualquier salida comienza a cambiar su valor.
- La ruta crítica, que para [1] es el camino más largo y por lo tanto el más lento. Esto es el caso extremo, donde la señal viaja a través de una cantidad de dispositivos, que al sumar su tiempo de propagación generan el resultado más elevado.
- La frecuencia máxima de operación, que se calcula con la ayuda de la ruta crítica y los tiempos de propagación, esta frecuencia se utiliza para asegurarse que las señales estén a tiempo en la salida del circuito combinacional cuando sean necesitadas, y así evitar leer una señal no deseada.

En este laboratorio se aplican los conceptos de lógica combinacional en el diseño e implementación de circuitos digitales lógicos y aritméticos utilizando SystemVerilog como Lenguaje de Descripción de Hardware. Además se tocarán temas relacionados con la lógica combinacional, como tiempos de propagación y contaminación, ruta crítica, frecuencia máxima de operación, etc. Además, está estructurado de la siguiente manera, sección de resultados, sección de análisis de los resultados obtenidos, conclusiones y las referencias utilizadas durante este laboratorio.

II. Resultados

II-A. Problema 1

Para el problema uno se necesita realizar una ALU parametrizable en SystemVerilog utilizando la herramienta de Quartus y presentar el mismo en una FPGA con un display 7 segmentos para la visualización del resultado y LEDs para las banderas.

Siguiendo el diseño estándar de una ALU se deben tener tres valores de entrada, los cuales son los operadores A, B y el bus de control de la ALU. Para el diseño especifico de este laboratorio se utilizaron 4 bits de control ya que la ALU realiza 9 operaciones los cuales no son representables en 3 bits de control. Las diferentes operaciones que debe realizar la ALU son: suma, resta, and, or, not (sobre un solo operando), xor, corrimiento a la izquierda y corrimiento a la derecha (tanto lógicos,como aritméticos). Para este diseño, la respectiva tabla de verdad con los datos correspondientes a los bits de control para la selección de su respectiva operación, para esto se utiliza un MUX que seleccione solo una operación. El cual se encuentra representado en el cuadro I.

Cuadro I
TABLA DE OPERACIONES DE LA ALU

Control	Operación	Descripción
0000	suma	A + B
0001	resta	A - B
0010	and	A & B
0011	negación	∼B
0100	or	A | B
0101	xor	A⊕ B
0110	Corrimiento izq	B<< $A_{[log_2 N-1:0]}$
0111	Corrimiento der	B>>$A_{[log_2 N-1:0]}$
1000	Corrimiento aritm	B>>>$A_{[log_2 N-1:0]}$
...		
default		0

Adicional mente la ALU debe presentar valores en la salida de datos lo cual es representado mediante un resultado de la operación que se desea realizar.Ademas a esto se generan 4 banderas de estado: Negativo (N), Cero (Z), Acarreo (C) y Desbordamiento (V). Para seguir el estándar de la arquitectura ARM.

Para obtener los distintos módulos para cada operación se realizaron sus respectivas tablas de verdad para dos entradas A y B de 1 bit y su respectiva salida. Las mismas se encuentran en la bitácora de trabajo en la sesión dos. No se pusieron en este documento por ser tan extensas.

Además, se creó un TestBench para verificar el correcto funcionamiento de la ALU, para cada operación se implementaron 4 casos de prueba. En la figura II-A se muestra la forma de las ondas generadas.

II-B. Problema 2

El problema dos del laboratorio consta de medir las frecuencias máximas de operación de la ALU diseñada, con diferentes tamaños de buses de entrada, los cuales son 32, 64, 128 y 256 bits. Para esto se utilizó un circuito sencillo que cuenta con dos registros de N bits, uno para alimentar la entradas de la ALU y otro para recibir la salida de esta, conectados a un mismo reloj(clk), aunque negado

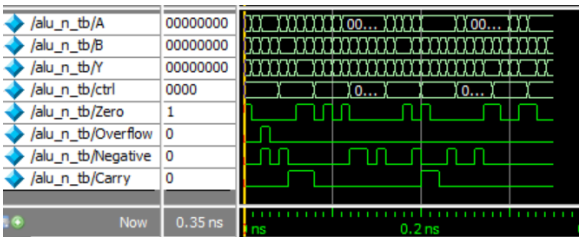

Figura 1. Formas de onda generadas por el testbench

para el segundo registro, para que en un mismo ciclo se alimente el dato a la ALU y y se lea el registro teniendo una diferencia de tiempo entre ellos. La Figura II-B muestra el circuito utilizado para medir la frecuencia.

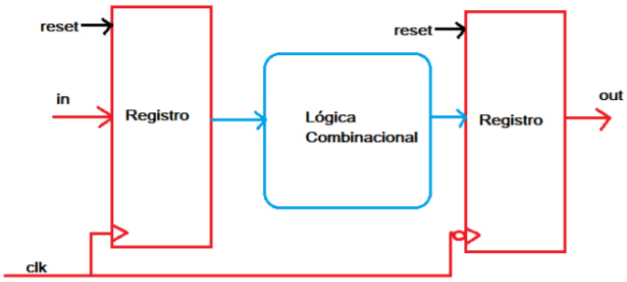

Figura 2. Circuito para medir frecuencias máximas de operación

Se utilizó el software Quartus para modelar un registro parametrizable de N bits por comportamiento. Después se definieron nuevos módulos estructurales que instancian los registros y la ALU con las diferentes cantidades de bits que se quieren probar (i.e 32, 64, 128, 256). Luego de sintetizados estos módulos se utilzó la herramienta TimeAnalyzer del software Quartus para obtener los reportes de frecuencia máxima. Los resultados obtenidos pueden ser vistos en el Cuadro II

Cuadro II
TABLA DE FRECUENCIAS MÁXIMAS PARA ALUS DE DIFERENTE TAMAÑO

Bits	Frecuencia Máxima de Operación (MHz)
32	40.24
64	40.54
128	38.38
256	55.08

III. ANÁLISIS DE RESULTADOS

III-A. Problema 1

Durante la simulación realizada con ayuda del TestBench, 8 de las 9 operaciones funcionaron correctamente, sin embargo, el corrimiento aritmético

a la derecha, en el caso específico donde el número era negativo (bit de signo = 1) falló, esto debido a que no se estaba utilizando las entradas de tipo "signed". Se intentó corregir, pero dicha solución aunque si funcionó para la implementación en la FPGA no funcionó para el TestBench. En las demás operaciones, tanto los resultados como las banderas de salida funcionaron para todas las entradas probadas.

A la hora de querer corregir el error en el corrimiento aritmético se topo con un error a la hora de realizar la simulación, el cambio que se realizo en ese momento fue aplicar "$usigned$" a las entrada y salida del modulo de corrimiento aritmético. Lo cual a la hora de hacer la sintetizar el diseño no presento ningún error. Se cree que a la hora de simular los datos en el testbench es donde ocurre el error, ya que al declarar que la entrada de ese dato va a ser sin signo y no declarar lo mismo en el testbench ocurre un error de tipos.

III-B. Problema 2

Los resultados obtenidos en el problema dos para la ALU de 32, 64 y 256 bits son inesperados. Se esperaba que al aumentar la cantidad de bits en los buses de entrada y de salida, las frecuencias de operación se disminuyeran. Puesto que al haber más bits, se necesitaría de mayor cantidad compuertas lógicas para lograr la lógica deseada, especialmente en el sumador, el cual presenta la ruta crítica del sistema. Esto hace que aumente el tiempo de propagación del circuito combinacional, por ende reduciendo la frecuencia de operación del circuito de prueba.

El comportamiento esperado se puede ver en la transición de la ALU de 64 bits a la de 128, donde la la frecuencia de operación sí se reduce. Se investigaron los reportes de utilización de recursos para las diferentes instancias del circuito de prueba, pero los reportes no mostraban mayor cambio más allá de la cantidad de pines de entrada y salida y los valores de fan-in y fan-out derivados de esta cantidad de pines.

Al intentar compilar el circuito de prueba con la ALU de 256 bits se encontró que el dispositivo no podía ser implementado en la FPGA Cyclone V utilizada para el laboratorio, puesto que la cantidad de pines requerida es mayor que la que puede ser proveída por el dispositivo. Se teoriza que ésta pudo ser una de las fuentes de error en el cálculo de la frecuencia máxima de operación para esta instancia de la ALU.

Los circuitos de prueba para 32 y 64 bits tienen solamente una pequeña diferencia en sus frecuencias de operación, teniendo la instancia de 64 bits una frecuencia un poco mayor que la de 32. Esto puede ser debido a la optimización que el software de síntesis hace a los circuitos para que estos cumplan con los requerimientos de tiempo especificados para el análisis de tiempo. Puesto que la diferencia en la cantidad de bits no es tan grande , y los mismos requerimientos de tiempo fueron utilizados para las mediciones, es posible que se hayan encontrado implementaciones para ambas instancias que pudieran operar a estas frecuencias.

IV. Conclusiones

La utilización de un análisis modular para la resolución del problema facilitó el proceso de diseño de la ALU de gran manera, puesto que los conceptos de análisis modular se traducen naturalmente a las habilidades de modelado estructural que permiten los HDL.

La utilización de un TestBench como herramienta de auto-chequeo para las diferentes operaciones de la ALU utilizando diversos casos fue sencillo de implementar y demostró el funcionamiento de la ALU en 3 bits.

En el problema 1, para la simulación con el TestBench, 8 operaciones de 9 funcionaron correctamente, solamente el corrimiento aritmético falló. Sin embargo para la implementación en la FPGA todas las operaciones funcionaron de la manera esperada.

Es importante tener en cuenta el tiempo de propagación, de contaminación y las rutas críticas de los circuitos que se diseñan, más aún si estos van a ser utilizados como elementos de circuitos secuenciales, puesto que estos van a limitar la posible frecuencia de operación del circuito donde sean integrados.

Referencias

[1] S. Harris and D. Harris, *Digital design and computer architecture*. Amsterdam [i 11 pozostałych]: Elsevier / Morgan Kaufmann Publishers, 2016.