Publicación 01

Lester García, Andrea Pérez, Alejandra Samayoa

Universidad Rafael Landívar

Resumen

Los lenguajes descriptores de hardware, o Hardware Descriptor Language cuyas siglas en inglés son HDL, permitieron la estandarización del diseño e implementación de circuitos lógicos digitales. Además, estos nuevos estándares convergieron en la facilitación de la producción de hardware en masa. Debido a lo anterior, en este documento se presenta la implementación y simulación de dos tipos de HDL, básico y verilog. Así mismo se plantea el concepto de modularidad mostrando como se relacionan la lógica con la aritmética y se propone el siguiente paso hacia la construcción de un sistema digital.

I. Introducción

En la actualidad todos los dispositivos digitales están basados en una serie de chips y compuertas diseñadas tanto para almacenar como para procesar información. George Boole fue un matemático inglés quien definió el álgebra booleana en 1847, este sistema lógico es el que hoy permite la existencia y buen funcionanmiento de los dispositivos eléctricos y electrónicos en la actualidad.

Una función booleana debe ser representada mediante una compuerta lógica, la implementación de las mismas con sus respectivos circuitos son las que hoy en día utilizan los ordenadores y otros dispositivos electrónicos. Con el crecimiento de la tecnología se ha requerido de la implementación de circuitos mucho más complejos lo cual no es del todo eficiente.

Por otro lado, el uso de FPGA's ha sido impulsada por Xilinx como una alternativa a la implemantación de circuitos integrados. Teniendo como características pricipales el rendimiento, el precio, la fidelidad y el mantenimiento a largo plazo. Los FPGA son chips re-programables, se puede borrar y agregar cualquier tipo de circuito sin tener la necesidad de utilizar un protobord o de un cautín entre otros elementos físicos. El simulador de hardware es otra manera más instructiva para aprender el comportamiento de circuitos. Es usado generalmente para construir y ejecutar programas diseñados de una forma lógica.

II. Lógica Booleana

Cuando se habla de álgebra booleana se trata de todo aquello que trabaja con 0's y 1's. Una función booleana es aquella que opera únicamente esos dos valores, con los que recibe una entrada binaria y como resultado se tienen salidas binarias.

Para representar de forma gráfica una función booleana se puede utilizar una tabla de verdad, la cual consiste en enumerar todas las combinaciones posibles de las entradas y con ello todos los diferentes valores que toma la salida de la función. Además, una función booleana se puede representar por medio de compuertas lógicas tales como And, Or, Not, Nand (compuerta primitiva) entre otras.

III. Nand2Tetris

Para el desarrollo e implementación de las compuertas lógicas se utilizó el lenguaje descriptor de hardware "HDL

por sus siglas en inglés, el cual fue proporcionado por http://www.nand2tetris.org. Nand2Tetris contiene una herramienta de simulación de hardware dentro de la carpeta de trabajo proporcionada en clase el cual se utiliza para simular el comportamiento de las diferentes compuertas lógicas y compararlo con un archivo de extensión .cmp y si la compilación era correcta, se generaba un archivo con extensión .out con el mismo nombre del CHIP.

El siguiente código fuente muestra la implementación de una compuerta XOR en el lenguaje HDL:

```
/**
 * Exclusive-or gate:
 * out = not (a == b)
 */
CHIP Xor {
    a, b;
    OUT out;

    PARTS:
    Not(in = a, out = a0);
    Not(in = b, out = b0);

    And(a = a, b = b0, out = out0);
    And(a = a0, b = b, out = out1);
    Or(a = out0, b = out1, out = out);
}
```

Por otro lado, el script que se proporciona a continuación fue utilizado para evaluar el funcionamiento de la compuerta XOR previamente desarrollada.

```
load Xor.hdl,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;

set a 0,
set b 0,
eval,
output;

set a 0,
set b 1,
eval,
output;

set a 1,
set b 0,
eval,
output;

set a 1,
set b 1,
eval,
output;
```

Tras la correcta compilación del código de la compuerta XOR y su respectiva prueba de integridad se obtienen los resultados de la simulación los cuales se pueden observar y analizar seleccionando la pestaña OUTPUT como se muestra a continuación:

- Output

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

Por otro lado, en el apartado COMPARE se tiene la posibilidad de comparar los resultados obtenidos por el script y el chip programado con los resultados esperados predeterminados.

- Compare

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

Si la tabla presentada en el apartado compare es igual a la tabla del output significa que el resultado es el correcto y se logró una simulación exitosa, y en caso contgrario, significa que existió un error lógico en la programación del chip en cuestión ya sea porque no existe o por una conexión incorrecta de los pines utilizados o un error propiamente de la sintaxis de HDL.

Al finalizar la depuración del código y realizar todas las pruebas de simulación deseada, si todo está correcto, el chip de lógica booleana fue implementado exitosamente.

IV. Verilog

Verilog, al igual que Nand2Tetris, es un lenguaje de descriptor de hardware. Para el desarrollo de este lenguaje se utilizará Vivado, software desarrollado por Xilinx, con el cual se podrá utilizar y trabajar con la FPGA designada la cual es la Basys3. Esta placa cuenta básicamente con una amplia colección de dispositivos de entrada y salida que permitirán su correcta configuración para poder realizar diferentes pruebas con circuitos lógicos.

V. Arirmética Booleana

Los circuitos y dispositivos electrónicos actuales son altamente potentes y precisos en la realización de procesos y tareas de la vida diaria, con grandes capacidades de almacenamiento y procesamiento, sin embargo, estos no serían útiles si no se pudieran hacer diferentes operaciones con los registros almacenados dentro. Por tanto, es allí dónde entra en vigor el papel desempeñado por la ALU (Unidad de aritmética lógica) o Arithmetic Logic Unit por sus siglas en inglés, es la parte de un computador encargada de realizar todas las operaciones aritméticas y lógicas con registros de tipo entero, es decir, con 0's y 1's que representen valores enteros en el sistema decimal.

Si bien es posible elaborar una ALU que realice cualquier operación (sin importar la complejidad presentada) hay que tomar en cuenta que según aumente la complejidad de dicha operación, el costo y uso de energía lo harán también.

Asimismo, la ALU tiene la capacidad de activar ciertos indicadores o banderas que evalúan los resultados obtenidos por las operaciones realizadas, por ejemplo, un bit indica si el resultado es 0, menor a 0 entre otros diferentes resultados.

VI. Discusión de Resultados

A partir de los resultados obtenidos se puede observar y comprobar el funcionamiento de la ALU realizada por medio de los resultados de la simulación utilizando las herramientas y archivos proporcionados por Nand2Tetris. A partir de la comparación realizada de los FPGA's se pudo observar la diferencia de potencia entre las mismas dependiendo de su proposito, así también se pudo observar que el código en VHDL por ser ya un lenguaje para desarrollar es más complejo que el HDL que únicamente se utiliza para simular.

sectionConclusiones

- Las FPGA's A7 son de propósitos más académicos por lo cual no tienen tanta potencia como las S7.
- La ALU es un componente vital en el procesador de un computador, esta misma realiza todas las operaciones aritméticas y lógicas.
- HDL es un lenguaje de simulación con fines educativos, mientras que VHDL ya es un lenguaje para desarrollo.

VII. Trabajo Futuro

A partir de lo desarrollado se seguirán construyendo y simulando las partes básicas de un computador para después utilizar el lenguaje Verilog y programar con ayuda de una Basys3.

Referencias

[1] Nisan, N. (2005). The elements of computing systems.Prentice-Hall of India..

[2] Introduction to FPGA Technology: Top 5 Benefits - National Instruments. (2011). Ni.com. Retrieved 21 February 2018, from http://www.ni.com/white-paper/6984/es/

[3] `http://www.nand2tetris.org/`.

[4] https://www.xilinx.com/products/silicon-devices/fpga/xa-spartan-7.html

[5] https://www.xilinx.com/products/silicon-devices/fpga/xa-artix-7.html