PROYECTO FINAL

ALU DE 8 BITS EN VHDL

Mayrenis Mendoza

Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad Latina de Panamá

RESUMEN: El trabajo describe la implementación de una Unidad Aritmética y Lógica - ALU de 8 bits, para realizar las operaciones básicas aritméticas de suma, resta y las operaciones lógicas AND y XOR. Los bloques lógicos del ALU han sido codificados en VHDL (VHSIC-HDL: Very High Speed Integrated Circuit - Hardware Description Language). El software de distribución gratuita, ISE Design Suite versión 14.7 fue utilizado para simular y programar.

PALABRAS CLAVE: ALU, aritméticas, arquitectura, código, entidad, operaciones, testbench, unsigned, VHDL, xilinx.

ABSTRACT: The work describes the implementation of an Arithmetic and Logic Unit - 8-bit ALU, to perform the basic arithmetic operations of addition, subtraction and the logical operations AND and XOR. The logic blocks of the ALU have been encoded in VHDL (VHSIC-HDL: Very High Speed Integrated Circuit - Hardware Description Language). The free distribution software, Vivado 2020.2v was used for simulation and programming.

KEY WORDS: ALU, arithmetic, architecture, code, entity, operations, testbench, unsigned, VHDL, xilinx.

I. INTRODUCCION

La ALU (Arithmetic-Logic Unit o Unidad Aritmético-Lógica) es la encargada de realizar las operaciones aritméticas (suma, resta, multiplicación, etc.) y lógicas (AND, OR, NOT, etc.) que ejecuta un microprocesador. Como se mencionó en la sección anterior, existen registros que la ALU utiliza para guardar información relacionada con el resultado de una operación (el registro de banderas), los cuales le ayudan a definir características del resultado permitiendo una correcta ejecución e interpretación de los datos, como por ejemplo si el resultado de cierta operación fue cero o si hubo un acarreo en el último bit. Existen varios tipos de circuitos integrados (CIs) disponibles que se denominan unidades aritméticas y lógicas, aunque no tienen la capacidad total de una unidad aritmética y lógica de una computadora. Estos CIs son capaces de realizar diversas operaciones aritméticas y lógicas con entrada de datos binarios.

II. DESARROLLO DE CONTENIDO

1. Lenguaje VHDL

VHDL, un lenguaje de descripción de hardware Very High Speed Integrated Circuits Hardware Description Language (VHDL), es el nombre que recibe el lenguaje para el diseño, simulación y síntesis de circuitos digitales más utilizado por la industria hoy en día. A través de VHDL se pueden crear circuitos digitales para probar su funcionalidad y posteriormente "sintetizar" el diseño en una tablilla electrónica; aunque se debe de tomar en cuenta de que no todos los diseños son sintetizables. VHDL describe el comportamiento de un circuito electrónico por medio de sentencias (líneas de código) que representan la funcionalidad del sistema, las cuales posteriormente se puede implementar el circuito (implementado en un dispositivo programable como los FPGA). Existen diferentes plataformas de las cuales nos podemos apoyar para desarrollar diseños con VHDL, que tiene herramientas propias para la esquematización, pruebas y síntesis del diseño en cuestión, pero la mayoría están especializadas para sus productos. Entre las plataformas de desarrollo encontramos VIVADO design suite de la compañía Xilinx, la cual ofrece un entorno de desarrollo muy amigable, una herramienta de pruebas a través de archivos de "TestBench", un analizador sintáctico actualizado con el último estándar VHDL y un método de síntesis especializado para los FPGA que la compañía desarrolla. Con VIVADO es posible realizar el diseño del circuito a través de diferentes métodos: máquinas de estados, código VHDL y diagramas esquemáticos; con la versatilidad convertir un método al otro automáticamente.

1.1. VHDL soporta distintos tipos de descripciones

Estructural. VHDL puede ser usado como un lenguaje de descripción estructural, donde se especifican por un lado los componentes del sistema y por otro sus interconexiones.

Comportamental. VHDL también se puede utilizar para la descripción comportamental o funcional de un circuito. Sin necesidad

de conocer la estructura interna de un circuito, es posible describirlo explicando su funcionalidad. Esto es especialmente útil en simulación ya que permite simular un sistema sin conocer su estructura interna, pero este tipo de descripción se está volviendo cada día más importante porque las actuales herramientas de síntesis permiten la creación automática de circuitos a partir de una descripción de su funcionamiento.

1.2. Fundamentos del lenguaje

Toda descripción de un sistema en VHDL está constituida por, al menos, tres tipos de elementos: Entidades, Arquitecturas, Bibliotecas. Generalmente, se realiza la descripción de un diseño definiendo una entidad que represente el mismo a través de una arquitectura que especifique su funcionamiento o la interconexión de componentes que lo integran utilizando elementos almacenados en bibliotecas.

En esta sección se pretende dar una idea de los elementos que forman una descripción VHDL.

2. Software

Vivado Design Suite es un paquete de software producido por Xilinx para la síntesis y análisis de diseños HDL, reemplazando a Xilinx ISE con características adicionales para el desarrollo de sistemas en un chip y síntesis de alto nivel. Vivado representa una reescritura y un replanteamiento de todo el flujo de diseño (en comparación con ISE).

Al igual que las versiones posteriores de ISE, Vivado incluye el simulador lógico integrado ISIM. Vivado también introduce síntesis de alto nivel, con una cadena de herramientas que convierte el código C en lógica programable.

Vivado Design Suite ofrece un nuevo enfoque para una productividad ultra alta con el diseño basado en IP y C / C ++ de próxima generación. Cuando se combina con la nueva Guía de metodología de diseño de productividad de alto nivel UltraFast ™, los usuarios pueden obtener una ganancia de productividad de 10 a 15 veces en comparación con los enfoques tradicionales. Las ediciones Vivado HLx son:

- Vivado HL Design Edition: incluye Dynamic Function eXchange y Vivado High-Level Synthesis
- Vivado HL System Edition: todas las funciones de Design Edition más System Generator para DSP
- Vivado HL WebPACK ™ Edition: versión gratuita y limitada por dispositivo de Vivado HL Design Edition
- Vivado Lab Edition: tamaño más pequeño, fácil de instalar en entornos de laboratorio, incluidas todas las capacidades de programación y depuración en la Design Edition completa.

3. Descripcion del problema

La ALU, es un circuito combinacional, es decir, un sistema cuya salida en un instante determinado dependen directamente de las entradas que tenga en ese mismo instante. En un esquema de maestro-esclavo, la unidad de control maneja las instrucciones y la ALU las ejecuta. Para que la ALU pueda procesar o ejecutar una

instrucción, es necesario que la CU le proporcione los siguientes datos:

- El código de operación
- La dirección del primer dato o el valor
- La dirección del segundo dato o el valor
- La dirección donde almacenará el resultado

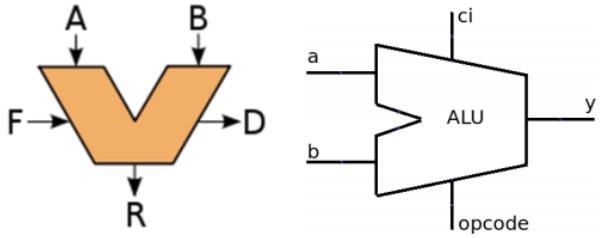

Figura 1. Representación de una ALU

En este trabajo se realiza una ALU de 8 bits en VHDL. Para crear una ALU de 8 bits capaz de hacer las siguientes operaciones:

- Sumar. A + B
- Resta: Not A - B
- Operación NAND: A nand B
- Operación XOR: A xor B
- A
- B
- Not A
- Not B

Estas operaciones son las más sencillas que puede hacer una ALU ya que se pueden realizar en un solo ciclo de reloj. En caso de querer ampliar la ALU para realizar multiplicaciones o divisiones, la ALU tardaría más de un ciclo de reloj en realizar dichas operaciones, lo cual cambiaria totalmente el diseño.

Operación	Resultado
000	A + B
001	Not A - B
010	A nand B
011	A xor B
100	A
101	B
110	Not A
111	Not B

Tabla 1. Funciones

La ALU realiza operaciones sobre dos operandos de 8 bits, denominados A y B. La salida de la ALU se selecciona mediante el bit más significativo de la señal OP, mientras que la operación que realiza se especifica mediante los otros dos bits de esta señal.

III. METODOLOGÍA

La plataforma para el desarrollo encontramos VIVADO DESING de la compañía Xilinx, la cual nos ofrece un entorno de desarrollo muy amigable, una herramienta de pruebas a través de archivos de "TestBench", un analizador sintáctico actualizado con el último estándar VHDL y un método de síntesis especializado para los FPGA que la compañía desarrolla. Con VIVADO es posible realizar el diseño del circuito a través de diferentes métodos: máquinas de estados, código VHDL y diagramas esquemáticos; con la versatilidad convertir un método al otro automáticamente.

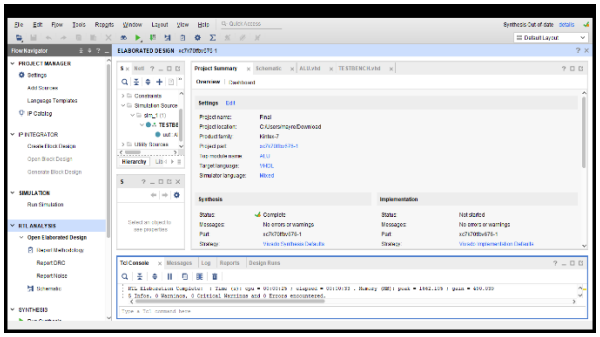

Figura 2. Interfaz del VIVADO 2020.2 de Xilinx

El código del programa utilizado para implementar la ALU.

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.std_logic_arith.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;

use IEEE.numeric_std.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU is

  Port (A: in STD_LOGIC_VECTOR (7 downto  0);

        B: in STD_LOGIC_VECTOR (7 downto 0);

        Op: in STD_LOGIC_VECTOR (2 downto 0);

      Res: out STD_LOGIC_VECTOR (7 downto 0));

end ALU;

architecture Behavioral of ALU is

begin

   with Op select

      Res <= A + B When "000",

         not A - B When "001",

         A nand B When "010",

         A xor B When "011",

         A  When "100",

         B  When "101",

         not A When "110",

         not B when others;

end Behavioral;
```

Primero que todo Como se utilizará operaciones matemáticas, con el programa se debe definir la librería `IEEE.numeric_std.ALL` que es la que permite esta clase de operaciones.

El programa se debe comenzar definiendo la entidad en la cual se declaran las diferentes entradas (A, B y OP) y la salida de la ALU (RES). Además, se debe incluir `unsigned` para permitir el desarrollo de las operaciones aritméticas que están en nuestra ALU.

Luego de implementar el código de nuestra ALU de las operaciones que usaremos, se realiza un archivo de simulación deseñales (TestBench) para observar que la ejecución de las operaciones esté correctamente realizada.

ALU.vhd *

C:/Users/mayre/Download/Final.srcs/sources_1/new/ALU.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
use IEEE.numeric_std.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ALU is
    Port ( A : in STD_LOGIC_VECTOR (7 downto 0);
           B : in STD_LOGIC_VECTOR (7 downto 0);
           Op : in STD_LOGIC_VECTOR (2 downto 0);
           Res : out STD_LOGIC_VECTOR (7 downto 0));
end ALU;

architecture Behavioral of ALU is
begin
   with Op select
       Res <= A + B When "000",
          not A - B When "001",
          A nand B When "010",
          A xor B When "011",
          A  When "100",
          B  When "101",
          not A When "110",
          not B when others;

end Behavioral;
```

Figura 3. Programa con el código de la ALU en VHDL

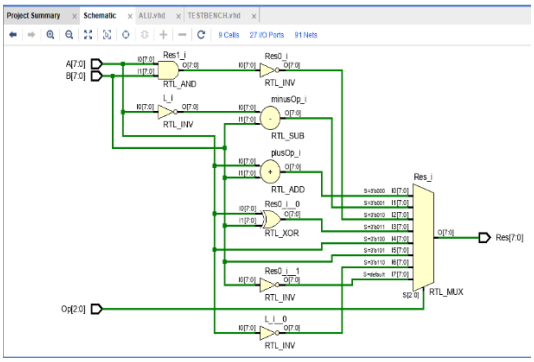

Figura 4. Diagrama esquimatico de la ALU de 8 bits

TestBench

Una vez codificada la ALU, se procede a realizar algunas pruebas de funcionalidad a través del archivo de testbench, en el cual se van a simular algunas señales para verificar cada una de las funciones posibles que realiza la ALU y verificar que los resultados que estas arrojen sean los correctos. Los testbench pueden crearse de 2 maneras diferentes, de manera gráfica o por medio de un archivo codificado en VHDL. La manera en las que se realizan los testbench en este documento es a través de un archivo VHDL ya que permite una mayor manipulación de los datos y no está restringido a los parámetros que el asistente muestra al momento de crearlo gráficamente.

TESTBENCH.vhd *

C:/Users/mayre/Download/Final.srcs/sim_1/new/TESTBENCH.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using...
ENTITY TESTBENCH IS
END TESTBENCH;
ARCHITECTURE behavioral OF TESTBENCH IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ALU
    PORT(
         A : IN  std_logic_vector(7 downto 0);
         B : IN  std_logic_vector(7 downto 0);
         Op : IN  std_logic_vector(2 downto 0);
         Res : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal A : std_logic_vector(7 downto 0) := (others => '0');
   signal B : std_logic_vector(7 downto 0) := (others => '0');
   signal Op : std_logic_vector(2 downto 0) := (others => '0');

    --Outputs
   signal Res : std_logic_vector(7 downto 0);
   -- No clocks detected in port list. Replace <clock> below with ...


BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: ALU PORT MAP (
          A => A,
          B => B,
          Op => Op,
```

Figura 5. Programa de la simulación de la ALU (TestBench)

TESTBENCH.vhd *

C:/Users/mayre/Download/Final.srcs/sim_1/new/TESTBENCH.vhd

```
    A<= "00010100";
    B<= "00100010";
    Op<= "010";
    wait for 100 ns;

    A<= "01100000";
    B<= "00001100";
    Op<= "011";
    wait for 100 ns;

    A<= "00110000";
    B<= "11000000";
    Op<= "100";
    wait for 100 ns;

    A<= "00100100";
    B<= "10000011";
    Op<= "101";
    wait for 100 ns;

    A<= "00001010";
    B<= "10100000";
    Op<= "110";
    wait for 100 ns;

    A<= "00101000";
    B<= "00001111";
    Op<= "111";
    wait for 100 ns;

        wait;
    end process;

END;
```

Figura 5.1. sucesión de pruebas para la salida

IV. RESULTADOS

Basándonos con la implementación de los códigos con el lenguaje VHDL que se realizó para el diseño creado a lo largo del proyecto, se desarrollaron archivos de pruebas para verificar la funcionalidad de cada uno de ellos. El archivo llamado "TestBench" mostrando así, a través de señales simuladas dentro de la plataforma de Xilinx, el comportamiento del componente o módulo ante los diferentes valores ingresados a través de estas mismas señales se verifica que los resultados arrojados por las señales ingresadas al sistema sean los correctos según el comportamiento definido en el código VHDL del módulo.

Obtuvimos los resultados de cada operación escogida para nuestra ALU de 8 bits, podemos observar en a figura 6.

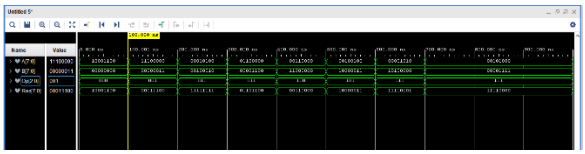

Figura 6. Simulación y grafica de nuestra ALU

Para realizar las pruebas de funcionalidad de la Unidad Aritmética-Lógica se diseñó el archivo de TestBench en el cual se simularon las entradas de los datos de los registros A y B, que son los datos con los que trabaja la ALU, y el Código de Operación que identifica cual operación debe realizar. En este caso se simularon los valores (binario) definidos en la siguiente tabla:

Operación		A	B	Resultado
A + B	000	10001100	00000000	10001100
Not A - B	001	11100000	00000011	00011100
A nand B	010	00010100	00100010	11111111
A xor B	011	01100000	00001100	01101100
A	100	00110000	11000000	00110000
B	101	00100100	10000011	10000011
not A	110	00001010	10100000	11110101
not B	111	00101000	00001111	11110000

Table 2. Resultado de del TestBench de la ALU

V. CONCLUSIÓN

Se ha codificado e implementado con éxito un ALU de 8 bits, para esto la elaboración de la ALU nos permitió realizar las operaciones aritméticas y lógicas simples de una forma confiada, debemos de saber antes que toda la importancia de programar para nosotros para obtener nuestra implementación del código necesario. Además, el diseño de circuitos digitales en un lenguaje de descripción de hardware tal como VHDL, es un proceso sencillo, ya que las herramientas de desarrollo proporcionan una manera fácil e intuitiva para la elaboración de sistemas digitales virtuales que, posteriormente, puedan ser implementados en dispositivos físicos o capaces de ser integradas en circuitos más complejos.

En el desarrollo de este proyecto, se elaboró exitosamente que arrojaron los datos esperados, con el aprendizaje de este lenguaje VHDL que es muy importante e interesante para este desarrollo podemos decir que es un software completo que nos permite hacer muchas funciones.

.

REFERENCIA

(S/f). Edu.ni. Recuperado el 15 de mayo de 2021, de https://repositorio.unan.edu.ni/6162/1/86380.pdf

De Una Unidad Aritmética Lógica Con Compuertas Lógicas Y Multiplexores, D. Y. C. (s/f). UNIDAD ACADÉMICA DE INGENIERÍA CIVIL CARRERA DE INGENIERÍA DE SISTEMAS. Edu.ec. Recuperado el 15 de mayo de 2021, de http://repositorio.utmachala.edu.ec/bitstream/48000/10941/1/TUAIC_2017_IS_CD0005.pdf

Definición de ALU. (s/f). Definicionabc.com. Recuperado el 15 de mayo de 2021, de https://www.definicionabc.com/tecnologia/alu.php

(S/f-b). Uam.mx. Recuperado el 15 de mayo de 2021, de http://kali.azc.uam.mx/erm//Media/Problemarios/Proyecto_1.pdf

De compuertas lógicas y multiplexores, E. E. P. V. C. P. M., De suma entera sin signo, se P. I. las O. A. B. de U. A. E. en E. P. I. la A., De suma, P. I. las O., Resta, Y., & de codificación., M. en V. E. (s/f). 3.8 Construcción de una ALU básica. U-cursos.cl. Recuperado el 15 de mayo de 2021, de https://www.u-cursos.cl/ingenieria/2009/2/EL54B/1/material_docente/bajar%3Fid_material%3D244739