



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRÁCTICA NO. 1 ALU CON OPERACIONES LÓGICAS, ARITMÉTICAS Y CORRIMIENTO

Arquitectura de Computadoras

Autores:

Hernández Vergara, Eduardo
Rojas Cruz, José Ángel
Alcantara Covarrubias, Erik
Martínez Alquicira, Mariana

Profesor:

Pastrana Fernández Carlos Jesús

15 de Octubre de 2022

Índice

1. Introducción	2
1.1. Objetivo	2
1.2. Introducción teórica	2
1.2.1. Unidad Aritmetica Lógica (ALU)	2
1.2.2. ¿Qué es VHDL?	3
1.2.3. ¿Qué es una FPGA?	3
1.3. Material y Equipo Empleado	3
2. Desarrollo Experimental	4
2.1. Objetivo Específico	4
3. Código	5
4. Conclusiones	8
4.1. Conclusiones Generales	8
4.2. Conclusiones Eduardo Hernández Vergara	8
4.3. Conclusiones José Ángel Rojas Cruz	8
4.4. Conclusiones Erik Alcantara Covarrubias	9
4.5. Conclusiones Mariana Martínez Alquicira	9
5. Referencias	10

1. Introducción

1.1. Objetivo

El alumno realizará una ALU, la que incluirá registros de corrimientos, lógicos y aritméticos la cual podrá realizar las operaciones con datos de 'n' bits especificados, tendrá una terminal que sirva para el control de carga.

1.2. Introducción teórica

1.2.1. Unidad Aritmetica Lógica (ALU)

La ALU es la parte de la computadora que realiza las operaciones lógicas y aritméticas de los datos. Todos los demás elementos de los sistemas de computadoras (Control Unit, Registros, memoria, entrada/salida) son algunos de los datos que se le entregan a la ALU para que los procese y después los devuelva. Entonces, en un sentido, hemos alcanzado la esencia de una computadora cuando consideramos una ALU.

Una ALU y, en realidad todos los componentes electrónicos en una computadora están basados en el uso de simples dispositivos lógicos digitales simples que pueden guardar dígitos binarios y realizar simples operaciones lógicas booleanas.

En la siguiente figura se muestra en términos generales, como está interconectada la ALU con el resto del procesador. Los datos son presentados en los registros de la ALU, y los resultados de la operación se guardan en los registros. Estos registros son almacenados temporalmente mientras el procesador esté conectado por rutas de señal a la ALU. A la ALU también se le pueden configurar banderas como el resultado de una operación. Por ejemplo, la bandera de overflow se configura en 1 si el resultado excede el tamaño del registro en donde va a ser almacenado. Los valores de estas banderas también son almacenados en los registros dentro del procesador. La unidad de control nos provee señales que controlan la operación del ALU y el movimiento de datos de entrada y salida del ALU.

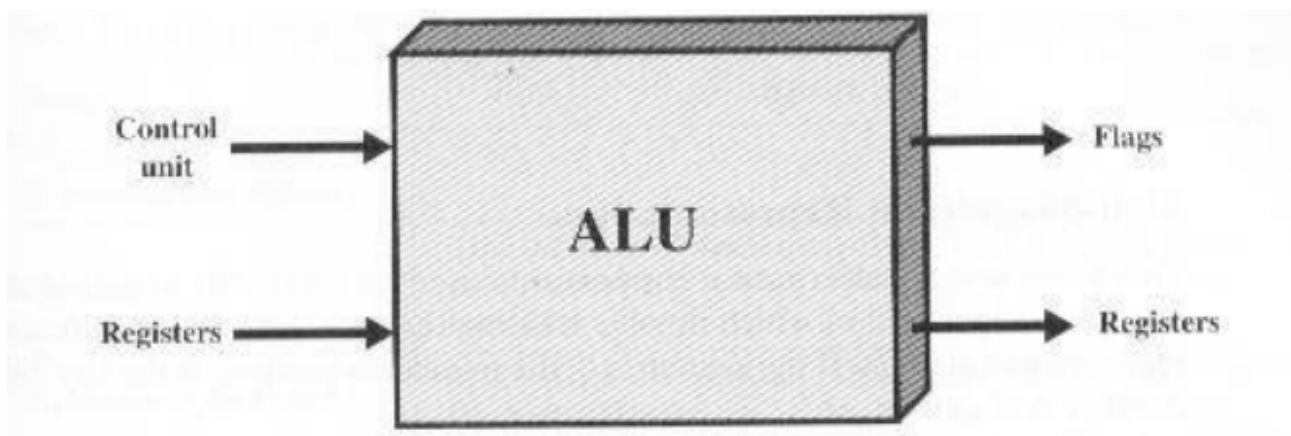


Figura 2: Entradas y Salidas del ALU

1.2.2. ¿Qué es VHDL?

VHDL es un lenguaje de diseño de software destinado a usarse en todas las fases del diseño de sistemas digitales. VHDL viene de VHSIC (Very High Speed Integrated Circuits) lenguaje descriptor de hardware. Su desarrollo empezó en 1983 con un contrato por parte del departamento de defensa de los Estados Unidos de América, y se volvió un estándar de la IEEE en 1987.

1.2.3. ¿Qué es una FPGA?

Una FPGA (Field Programmable Gate Array) es un complejo circuito integrado digital programable compuesto por bloques lógicos configurables (CLB) y puertos de entrada/salida (IOB), cuya interconexión y funcionalidad puede ser programada mediante un lenguaje de descripción especializado.

Su principal característica y ventaja es que pueden reprogramarse para un trabajo específico o cambiar sus requisitos después de haberse fabricado. Esto también implica que en muchos casos se pueden hacer cambios físicos sin hacer modificaciones costosas en la placa que lo soporta.

Básicamente, una FPGA es un conjunto de múltiples circuitos (lógicos y de otros tipos) dispuestos matricialmente, cuyas interconexiones son programables por el usuario para la aplicación requerida. En una FPGA se programa su hardware, a diferencia de los microcontroladores / microprocesadores, en los que solo existe un hardware fijo y se programa su software (firmware).

Históricamente, las FPGA fueron inventadas en el año 1984 por Ross Freeman y Bernard Von Der Schmitt, cofundadores de la empresa Xilinx, fabricante de las mismas. Como resultado de numerosas evoluciones, la compañía produjo la primera familia de dispositivos lógicos programables por el usuario, de propósito general.

Las FPGA, además de contener puertas lógicas AND y OR, tienen memoria RAM, controladores de reloj, etc., por lo que son muy apropiadas para el diseño de sistemas embebidos con microprocesador. La compañía Xilinx ha evolucionado dicha tecnología hasta convertirla en un nuevo concepto a tener en cuenta en ciertos entornos de trabajo.

1.3. Material y Equipo Empleado

- 1 FPGA
- 1 Tarjeta de evaluación de pruebas ó Protoboard microswitches con resistencias 10 KoHms, 20 leds con resistencias 330 oHms, 1 display de 4 dígitos multiplexado de ánodo común.
- 1 eliminador de celular de 5v con cable mini usb.
- Quartus Prime Lite

2. Desarrollo Experimental

2.1. Objetivo Específico

1. Desarrollar en la FPGA un programa en VHDL de una ALU, en cuál se puedan seleccionar tres tipos de operaciones, Lógicas, Aritméticas y Shifters, éstas son:
 - Lógicas (A y B de 10 bits):
 - a) Negación o Complemento a 1 de A.
 - b) Complemento a 2 de A.
 - c) AND entre A y B.
 - d) OR entre A y B.
 - Shifters (A de 10 bits):
 - a) LSL (Logical Shift Left).
 - b) ASR. (Arithmetic Shift Right).
 - Aritméticas (A y B)
 - a) Suma 1 byte c/ carry out.
 - b) Resta 1 byte.
 - c) Multiplicación de 5 bits
2. Con los siguientes especificaciones:
 - Realizar el programa en VHDL que incluya todas las operaciones básicas (Lógicas, Shifters, Aritméticas) del tamaño de dato correcto como se solicita en la descripción.
 - Recuerde que todas las operaciones anteriores deben funcionar como una unidad, para este inciso se deben programar todas las opciones por componentes o multi procesos.
 - Para las opciones de corrimientos de debe incluir un clock para sincronizar los desplazamientos.
 - Todas las entradas deben ser consideradas dependiendo de la operación a realizar, estas deberán ser introducidas por medio de microswitches.
 - La asignación de pines se debe de hacer de acuerdo a la disponibilidad en las terminales de salida en la FPGA.
 - La salida del bloque aritmético se tendrá que ser desplegada en el display de 7 seg. 4 dígitos y a su vez la salida de los shifters y la unidad lógica se mostrara únicamente mediante Leds.
3. Se manejara el mismo programa del inciso anterior pero ahora todas las operaciones tanto Aritméticas, shifters o lógicas se mostraran mediante mensaje de texto pregrabado en memoria ROM indexada en VHDL para el texto identificador de la operación por ejemplo: SunnA, rEstA, nnulti, And, or, not, Co1, Co2, ror, roL, LSL,LSren tipo ventana deslizante display de 4 dígitos de tipo ánodo común de 7 seg. El cual se multiplexará para que se despliegue 5 segundos antes de mostrar el resultado de la operación. Cada una de las letras será almacenada en un solo byte, y se multiplexará similar a los números pero con ventana deslizante que significa que el mensaje se desplaza para mostrar todas las letras durante 5 segundos de forma automática antes de mostrar el resultado.

3. Código

Listing 1: Barrel Shifter (Prototipo)

```
1  LIBRARY IEEE;
2  USE IEEE.std_logic_1164.ALL;
3  USE IEEE.numeric_std.ALL;
4
5  ENTITY barrelShifters IS
6      PORT (
7          a : IN STD_LOGIC_VECTOR(9 DOWNT0 0);
8          cntrl, iniciar : IN STD_LOGIC;
9          salidaLC : OUT STD_LOGIC;
10         salShifters : OUT STD_LOGIC_VECTOR(9 DOWNT0 0)
11     );
12 END ENTITY barrelShifters;
13
14 ARCHITECTURE shifters OF barrelShifters IS
15
16     SIGNAL contador : INTEGER RANGE 0 TO 49999999 := 0;
17     SIGNAL aux : STD_LOGIC_VECTOR(9 DOWNT0 0) := "0000000000";
18     SIGNAL salidMed : STD_LOGIC;
19
20 BEGIN
21     --Mi divisor de frecuencias
22     DivisorFrecuencia : PROCESS (clk, iniciar)
23     BEGIN
24         IF iniciar = '0' THEN
25             salidMed <= '0';
26             contador <= 0;
27         ELSIF rising_edge(clk) THEN
28             IF contador = 49999999 THEN
29                 contador <= 0;
30                 salidMed <= NOT salidMed;
31             ELSE
32                 contador <= contador + 1;
33             END IF;
34         END IF;
35     END PROCESS DivisorFrecuencia;
36
37     salidaLC <= salidMed;
38     --Las opciones del shifter
39     Shifter : PROCESS (salidaMed, iniciar, a, cntrl)
40     BEGIN
41         IF iniciar = '0' THEN
42             aux <= "0000000000";
43         ELSIF rising_edge(salidaMed) THEN
44             IF (cntrl = '0') THEN
45                 aux <= a(8 DOWNT0 0) & '0'; --LSL
46             ELSE
47                 aux <= a(9) & a(9 DOWNT0 1);-- Arithmetic Shifter Right
48             END IF;
49         END IF;
50     END PROCESS Shifter;
51
52     salShifters <= aux;
53     a <= aux;
54 END ARCHITECTURE shifters;
```

Listing 2: Display Operaciones (Prototipo)

```

1  -- PROTOTIPO
2  LIBRARY IEEE;
3  USE IEEE.std_logic_1164.ALL;
4  USE IEEE.numeric_std.ALL;
5  ENTITY displayOperaciones IS
6      PORT (
7          cntrlSeg : IN STD_LOGIC_VECTOR(0 TO 1);
8          cntrlArt  : IN STD_LOGIC_VECTOR(0 TO 1);
9          cntrlShf  : IN STD_LOGIC;
10         cntrlLog  : IN STD_LOGIC_VECTOR(0 TO 1);
11         d0, d1, d2, d3 : OUT STD_LOGIC_VECTOR(0 TO 6)
12     );
13 END ENTITY displayOperaciones;
14 ARCHITECTURE dOperaciones OF displayOperaciones IS
15     SIGNAL ad0, ad1, ad2, ad3 : STD_LOGIC_VECTOR(0 TO 6);
16 BEGIN
17     ProcDisplay : PROCESS (cntrlSeg, cntrlArt, cntrlShf, cntrlLog, ad1,
18         ad2, ad0, ad3)
19     BEGIN
20         IF cntrlSeg = "00" THEN --Arithmetic
21             IF cntrlArt = "00" THEN --Suma
22                 ad0 <= "0100100"; --S Anodo
23                 ad1 <= "1000001"; --U Anodo
24                 ad2 <= "1101010"; --n Andodo
25                 ad3 <= "1101010"; --n Andodo
26             ELSIF cntrlArt = "01" THEN -- Resta
27                 ad0 <= "1111010"; --r Anodo
28                 ad1 <= "0110000"; --E Anodo
29                 ad2 <= "0100100"; --S Anodo
30                 ad3 <= "1110000"; --t Andodo
31             ELSIF cntrlArt = "10" THEN --Mult
32                 ad0 <= "1101010"; --n Anodo
33                 ad1 <= "1101010"; --n Anodo
34                 ad2 <= "1000001"; --U Anodo
35                 ad3 <= "1110001"; --L Andodo
36             END IF;
37         ELSIF cntrlSeg = "01" THEN --Shifter
38             IF cntrlShf = '0' THEN -- LSL
39                 ad0 <= "1110001"; --L Anodo
40                 ad1 <= "0100100"; --S Anodo
41                 ad2 <= "1110001"; --L Andodo
42                 ad3 <= "1111111"; --NULL Andodo
43             ELSE --ASR
44                 ad0 <= "0001000"; --A Anodo
45                 ad1 <= "0100100"; --S Anodo
46                 ad2 <= "1111010"; --r Andodo
47                 ad3 <= "1111111"; --NULL Andodo
48             END IF;
49         ELSIF cntrlSeg = "10" THEN --Logic
50             IF cntrlLog = "00" THEN --NOT
51                 ad0 <= "1101010"; --n Anodo
52                 ad1 <= "1100010"; --o Anodo
53                 ad2 <= "1110000"; --t Andodo
54                 ad3 <= "1111111"; --null Andodo
55             ELSIF cntrlLog = "01" THEN -- COMP2
56                 ad0 <= "0110001"; --C Anodo
57                 ad1 <= "1100010"; --o Anodo
58                 ad2 <= "0011000"; --P Anodo
59                 ad3 <= "0010010"; --2 Andodo

```

```

59         ELSIF cntrlLog = "10" THEN --AND
60             ad0 <= "0001000"; --A Anodo
61             ad1 <= "1101010"; --n Anodo
62             ad2 <= "1000010"; --d Anodo
63             ad3 <= "1111111"; --null Andodo
64         ELSIF cntrlLog = "11" THEN --OR
65             ad0 <= "1111111"; --null Anodo
66             ad1 <= "1100010"; --o Anodo
67             ad2 <= "1111010"; --r Anodo
68             ad3 <= "1111111"; --null Andodo
69         END IF;
70     END IF;
71 END PROCESS ProcDisplay;
72 d0 <= ad0;
73 d1 <= ad1;
74 d2 <= ad2;
75 d3 <= ad3;
76 END ARCHITECTURE dOperaciones;

```

4. Conclusiones

4.1. Conclusiones Generales

El circuito presentado en la practica fue elaborado deacuerdo a los requerimientos solicitados, se trabajo de forma eficiente en la implementacion de las operaciones que se tenian que realizar ademas de ir aplicando diversos conceptos teoricos sobre las partes de un computador; ademas del uso de las matematicas para la elaboracion de un algoritmo que permitiee obtener los resultados esperados a partir de ciertas entradas de datos. Con esto podemos observar la importancia de los ALU como parte fundamental de los procesadores que se utilizan actualmente y por ello como futuros ingenieros debemos conocer su aplicacion tanto teorica como practica dando un enfoque hacia la problematica que buscamos resolver

4.2. Conclusiones Eduardo Hernández Vergara

En el desarrollo de la práctica pudimos aprender/recordar algunos de los elementos más básicos de la programación en VHDL, esto con el objetivo de desarrollar nuestra ALU básica/intermedia, en lo personal hubo cosas complicadas al momento de intentar implementarlas a VHDL y programarlo en la FPGA, como fue el caso de la ventana deslizante, y la unidad aritmetica, la ventana deslizante fue especialmente retadora por los ciclos de reloj, ya que fue complicado sincronizar procesos adecuadamente para que saliera la ventana deslizante, mientras que la unidad Aritmetica simplemente no salia, en cuanto a las opciones de shifters y las operaciones lógicas fueron relativamente sencillas, en el caso de los Shifters el unico problema era hacer que ciclara solo, mientras que las operaciones lógicas no fueron desafio alguno. Sin embargo estos obstaculos fueron superados para finalmente dar solución a la práctica.

4.3. Conclusiones José Ángel Rojas Cruz

En el desarrollo de la práctica, me di cuenta de la importancia que tiene conocer a fondo el funcionamiento de los componentes que forman parte de la unidad aritmética de la CPU, debido a que era importante conocerlo para saber cómo es que se manejarían los distintos componentes de la ALU y las salidas esperadas de cada uno de ellos, para así poder replicarlos en VHDL con la ayuda de la FPGA. Lo anterior se logró comprender al unir todos los elementos de la ALU en uno solo, desde la unidad aritmético-lógica hasta las operaciones de barrel shifters, teniendo en cuenta que en algunos casos tenían distintas formas de presentar las salidas, por lo que se debía de entender cómo lograr hacer este cambio y aplicarlo correctamente. Por último, la implementación de la ventana deslizante represento un reto por lo menos lógico debido a que se tuvieron que usar algunas herramientas que no se habían utilizado antes en programas desarrollados anteriormente, como consultar una memoria ROM y aplicar distintos eventos con solo una frecuencia de reloj. Creo que esta práctica hace que me dé cuenta de que necesito un estudio más profundo de las herramientas que nos brinda el lenguaje VHDL para poder facilitar y en algunos casos solucionar, problemas que se me presenten más adelante.

4.4. Conclusiones Erik Alcantara Covarrubias

Gracias a esta practica nos dimos cuenta de la gran importancia de el uso de los diferentes componentes de una CPU, la unidad de control, los multiplexores, y la ALU fueron muy complicados de realizar, pero gracias a ellos pudimos hacer operaciones matematicas y lograr decisiones en base a datos obtenidos de forma automatica

4.5. Conclusiones Mariana Martínez Alquicira

El desarrollo de esta práctica nos sirvió para recordar algunos conceptos manejados en la anterior materia, al igual que poner en práctica conocimientos que adquirimos en las clases, como el entendimiento de las partes de la computadora y que parte tiene control de qué. Con respecto a la programación, la parte que más se nos complico fue la parte aritmética, ya que nos percatamos que al ponerlo en marcha, obteníamos resultados erróneos cuando las cifras eran iguales. La parte lógica fue más sencilla ya que únicamente para el complemento 1, obtenemos la negación, para el complemento 2 usamos la misma negación del complemento 1 y le sumamos uno, por ultimo las operaciones AND y OR. Y por parte de las operaciones Shifters fue un tanto complicado pero no presentó más inconvenientes y la parte con más trabajo fue la parte del display. Al final el circuito pudo cumplir el objetivo de la práctica, el cual implicaba la elaboración de una ALU en la cual se pueden hacer operaciones aritméticas, lógicas y Shifters, con datos obtenidos de forma aleatoria.

5. Referencias

Referencias

- [1] Stallings, W. (2003). *Computer Organization and Architecture*. Pearson Education International.
- [2] Jasinski, R. (2015). *Effective Coding with VHDL: principles and best practice*. The MIT Press.
- [3] Akka Technologies. (n.d.). *FPGA: qué es y cuáles son las características de este componente. [online]*. Available at: <https://www.akka-technologies.com/fpga/> [Accessed 16 Oct. 2022]..