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
Mariana Alquisira (revisar)

Profesor:
Pastrana Fernández Carlos Jesús

15 de Octubre de 2022

Índice

1. Introdución

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 aritmeticas 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 electronicos en una computadora estan basados en el uso de simples dispositivos lógicos digitales simples que pueden guardar digitos binarios y realizar simples operaciones lógicas boleanas.

En la siguiente figura se muestra en términos generales, como esta 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 guarda en los registros. Estos registros son almacenados temporalmente mientras el procesador este conectado por rutas de señal al ALU. Al 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 almacenadas en los registros dentro del procesador. La unidad de control nos provee señalesque controlan la operación del ALU y el movimiento de datos de entrada y salida del ALU.

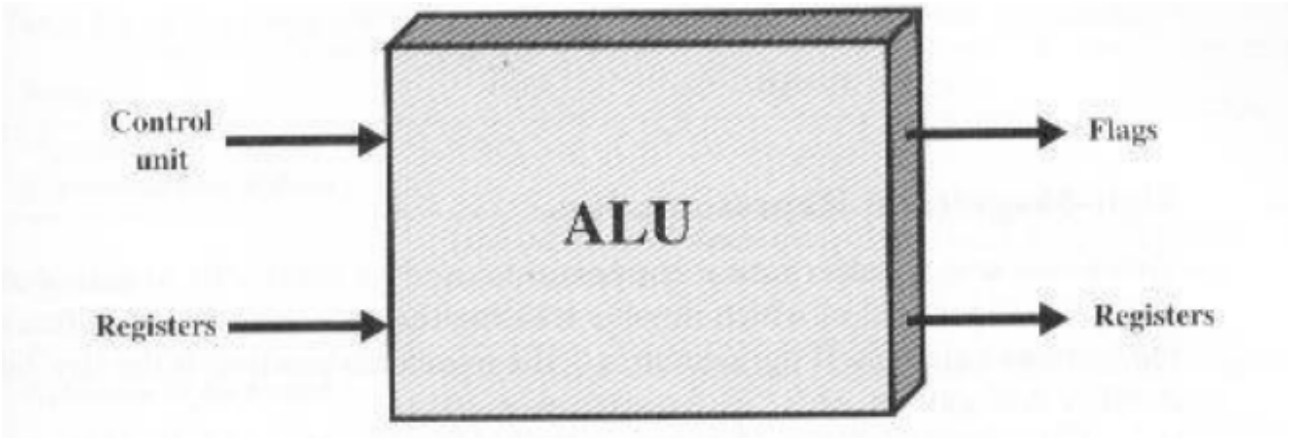

Figura 2: Entradas y Salidas del ALU

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 defenza de los Estados Unidos de Ámerica, 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 digitos 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, estás 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 digitos 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)

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY barrelShifters IS
    PORT (
        a : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
        cntrl, clk, iniciar : IN STD_LOGIC;
        salidaLC : OUT STD_LOGIC;
        salShifters : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)
    );
END ENTITY barrelShifters;

ARCHITECTURE shifters OF barrelShifters IS

    SIGNAL contador : INTEGER RANGE 0 TO 49999999 := 0;
    SIGNAL aux : STD_LOGIC_VECTOR(9 DOWNTO 0) := "0000000000";
    SIGNAL salidMed : STD_LOGIC;

BEGIN
    --Mi divisor de frecuencias
    DivisorFrecuencia : PROCESS (clk, iniciar)
    BEGIN
        IF iniciar = '0' THEN
            salidMed <= '0';
            contador <= 0;
        ELSIF rising_edge(clk) THEN
            IF contador = 49999999 THEN
                contador <= 0;
                salidMed <= NOT salidMed;
            ELSE
                contador <= contador + 1;
            END IF;
        END IF;
    END PROCESS DivisorFrecuencia;

    salidaLC <= salidaMed;
    --Las opciones del shifter
    Shifter : PROCESS (salidaMed, iniciar, a, cntrl)
    BEGIN
        IF iniciar = '0' THEN
            aux <= "0000000000";
        ELSIF rising_edge(salidaMed) THEN
            IF (cntrl = '0') THEN
                aux <= a(8 DOWNTO 0) & '0'; --LSL
            ELSE
                aux <= a(9) & a(9 DOWNTO 1);-- Arithmetic Shifter Right
            END IF;
        END IF;
    END PROCESS Shifter;

    salShifters <= aux;
    a <= aux;
END ARCHITECTURE shifters;
```

Listing 2: Display Operaciones (Prototipo)

```
-- PROTOTIPO
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
ENTITY displayOperaciones IS
    PORT (
        cntrlSeg : IN STD_LOGIC_VECTOR(0 TO 1);
        cntrlArt : IN STD_LOGIC_VECTOR(0 TO 1);
        cntrlShf : IN STD_LOGIC;
        cntrlLog : IN STD_LOGIC_VECTOR(0 TO 1);
        d0, d1, d2, d3 : OUT STD_LOGIC_VECTOR(0 TO 6)
    );
END ENTITY displayOperaciones;
ARCHITECTURE dOperaciones OF displayOperaciones IS
    SIGNAL ad0, ad1, ad2, ad3 : STD_LOGIC_VECTOR(0 TO 6);
BEGIN
    ProcDisplay : PROCESS (cntrlSeg, cntrlArt, cntrlShf, cntrlLog, ad1,
       ad2, ad0, ad3)
    BEGIN
        IF cntrlSeg = "00" THEN --Artimethic
            IF cntrlArt = "00" THEN --Suma
                ad0 <= "0100100"; --S Anodo
                ad1 <= "1000001"; --U Anodo
                ad2 <= "1101010"; --n Andodo
                ad3 <= "1101010"; --n Andodo
            ELSIF cntrlArt = "01" THEN -- Resta
                ad0 <= "1111010"; --r Anodo
                ad1 <= "0110000"; --E Anodo
                ad2 <= "0100100"; --S Anodo
                ad3 <= "1110000"; --t Andodo
            ELSIF cntrlArt = "10" THEN --Mult
                ad0 <= "1101010"; --n Anodo
                ad1 <= "1101010"; --n Anodo
                ad2 <= "1000001"; --U Anodo
                ad3 <= "1110001"; --L Andodo
            END IF;
        ELSIF cntrlSeg = "01" THEN --Shifter
            IF cntrlShf = '0' THEN -- LSL
                ad0 <= "1110001"; --L Anodo
                ad1 <= "0100100"; --S Anodo
                ad2 <= "1110001"; --L Andodo
                ad3 <= "1111111"; --NULL Andodo
            ELSE --ASR
                ad0 <= "0001000"; --A Anodo
                ad1 <= "0100100"; --S Anodo
                ad2 <= "1111010"; --r Andodo
                ad3 <= "1111111"; --NULL Andodo
            END IF;
        ELSIF cntrlSeg = "10" THEN --Logic
            IF cntrlLog = "00" THEN --NOT
                ad0 <= "1101010"; --n Anodo
                ad1 <= "1100010"; --o Anodo
                ad2 <= "1110000"; --t Andodo
                ad3 <= "1111111"; --null Andodo
            ELSIF cntrlLog = "01" THEN -- COMP2
                ad0 <= "0110001"; --C Anodo
                ad1 <= "1100010"; --o Anodo
                ad2 <= "0011000"; --P Anodo
                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

Eget lorem dolor sed viverra ipsum nunc. Ipsum dolor sit amet consectetur. Ipsum dolor sit amet consectetur adipiscing elit. Est ultricies integer quis auctor elit sed vulputate mi sit. Sit amet venenatis urna cursus eget. Morbi quis commodo odio aenean sed adipiscing. In dictum non consectetur a erat. Eu lobortis elementum nibh tellus molestie nunc non blandit massa. Neque ornare aenean euismod elementum nisi quis eleifend. Nisl suscipit adipiscing bibendum est ultricies integer quis. Sodales neque sodales ut etiam sit amet nisl purus in.

4.2. Conclusiones Eduardo Hernández Vergara

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Consectetur purus ut faucibus pulvinar elementum integer. Ullamcorper velit sed ullamcorper morbi tincidunt ornare. In fermentum et sollicitudin ac. Magna ac placerat vestibulum lectus mauris. Semper quis lectus nulla at volutpat diam ut. Gravida arcu ac tortor dignissim convallis aenean et tortor at. Integer eget aliquet nibh praesent tristique magna. Sed velit dignissim sodales ut eu sem integer vitae justo. Vel fringilla est ullamcorper eget nulla facilisi etiam dignissim diam.

4.3. Conclusiones José Ángel Rojas Cruz

Ultricies mi quis hendrerit dolor magna eget est lorem ipsum. Velit egestas dui id ornare arcu odio. Neque sodales ut etiam sit amet nisl purus in mollis. Nec ultrices dui sapien eget mi proin sed libero enim. Sit amet risus nullam eget felis eget nunc lobortis. Velit dignissim sodales ut eu sem. Lorem donec massa sapien faucibus et molestie. Quis varius quam quisque id diam vel quam elementum pulvinar. Netus et malesuada fames ac turpis. Posuere sollicitudin aliquam ultrices sagittis orci a. Scelerisque felis imperdiet proin fermentum leo vel orci porta. Et malesuada fames ac turpis egestas maecenas.

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 Alquisira (revisar)

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). *Efective 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]..