Arquitectura Básica de 8 bits

I.G.Ramírez Jiménez, J.Colín Ramiro, J.C. Hernández Reyes
Departamento de ingeniería en sistemas computacionales, ESCOM IPN
Unidad Profesional Adolfo López Mateos, 07320 Ciudad de México, CDMX
itzelraji12@gmail.com, joelcolinr@gmail.com, jhernandezr1923@alumno.ipn.mx

Resumen*— *Para esta primera práctica se realizó una arquitectura de 8 bits el cual contiene un total de 20 instrucciones entre las que se encuentran de tipo logarítmicas, aritméticas, manejo de registros, etc. Con las entradas siendo los switches de la tarjeta proporcionada por el docente y las salidas son a través del display de LCD. Esta práctica se llevó a cabo mediante el software Quartus II y basándose en los conocimientos adquiridos a lo largo de los cursos de Diseño Digital que previamente se nos otorgaron en la Institución.

Palabras Clave*— *Arquitectura, Display, Entrada, Instrucción, Salida

Abstract*— *In this first practice, an 8-bit architecture was made, which contains a total of 20 instructions, among which are logarithmic, arithmetic, register management, etc. With the inputs being the switches of the card provided by the teacher and the outputs are through the LCD display. This practice was carried out using the Quartus II software and based on the knowledge acquired throughout the Digital Design courses that were previously given to us at the Institution.

Keywords*—*Architecture, Display, Input, Instruction, Output

I. Introducción

La arquitectura de computadoras es el diseño conceptual y la estructura operacional fundamental de un sistema que conforma una computadora. Es decir, es un modelo y una descripción funcional de los requerimientos y las implementaciones de diseño para varias partes de una computadora, con especial interés en la forma en que la unidad central de proceso (CPU) trabaja internamente y accede a las direcciones de memoria, explica la situación de sus componentes y permite determinar las posibilidades de un sistema informático, con una determinada configuración, pueda realizar las operaciones para las que se va a utilizar.

La unidad aritmética lógica, también conocida como ALU (siglas en inglés de arithmetic logic unit), es un circuito digital que calcula operaciones aritméticas (como suma, resta, multiplicación, etc.) y operaciones lógicas (si, y, o, no), entre dos números.

Muchos tipos de circuitos electrónicos necesitan realizar algún tipo de operación aritmética, así que incluso el circuito dentro de un reloj digital tendrá una ALU minúscula que se mantiene sumando 1 al tiempo actual, y se mantiene comprobando si debe activar el sonido de la alarma, etc.
En arquitectura de computadoras, 8 bits es un adjetivo usado para describir enteros, direcciones de memoria u otras unidades de datos que comprenden hasta 8 bits (1 octeto) de ancho, o para referirse a

una arquitectura de CPU y ALU basadas en registros, bus de direcciones o bus de datos de ese ancho. Las CPU de 8 bits normalmente usan un bus de datos de 8 bits y un bus de direcciones de 16 bits lo que causa que su memoria direccionable esté limitada a 64 kilobytes; sin embargo esto no es una "ley natural", ya que existen excepciones.

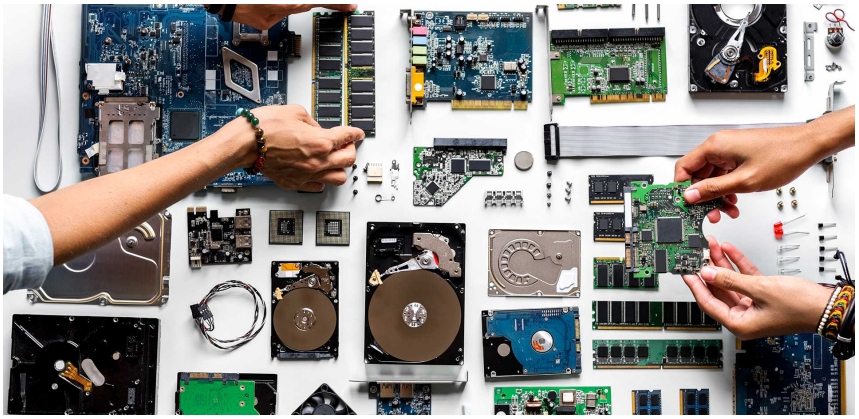

Figura 1.- Elementos que componen una arquitectura de computadoras

Con ALU se busca que podamos ejecutar instrucciones y mandar mensajes a la Fpga Altera De2 115 que está orientada al desarrollo de aplicaciones en lógica combinacional y secuencial; esta tarjeta maneja posee un excelente balance entre bajo costo, bajo consumo de energía y desempeño.

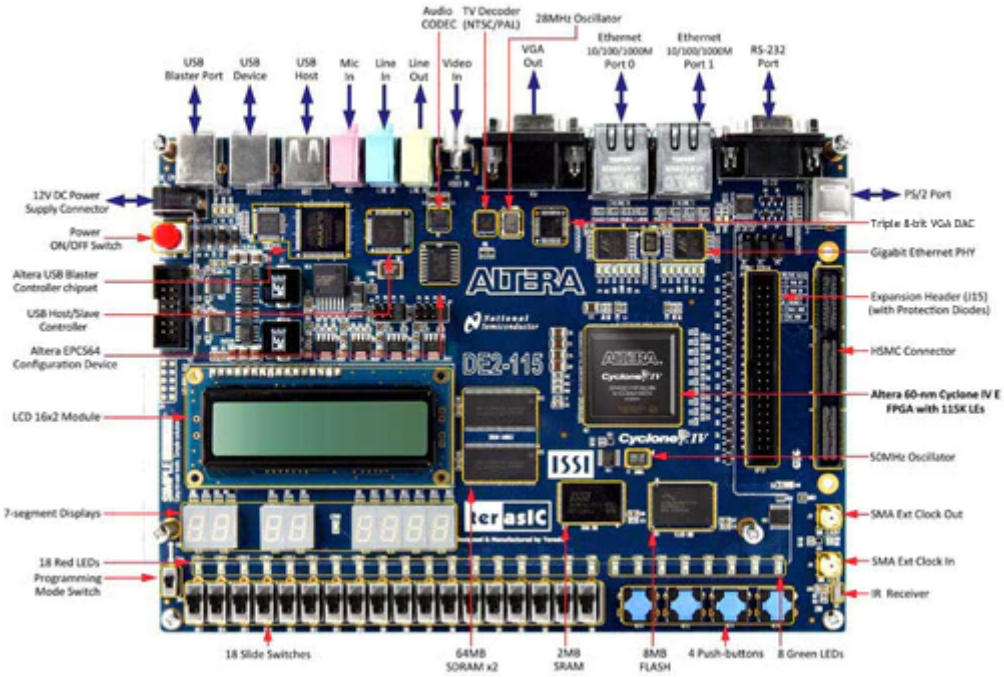

Figura 2.-ALTERA De2-115

II. Metodología/Desarrollo

Para la realización de esta primera práctica se realizaron 3 programas por separado. Se manejó el concepto de paquetes y procedimientos para poder trabajar con algunas “librerías”, creadas por los miembros del equipo en cuestión. Esto debido a que si lo trabajamos todo dentro de un solo programa, este eventualmente se desorganizaria, podría confundir a algún integrante y resultaría poco práctico

Ahora bien el primero y principal es el programa de la Unidad de Control. Para darle comienzo al desarrollo del mismo programa, se declararon las librerías, más concretamente las IEEE, que permitan, operaciones sin signo y la que es más conocida por nuestra comunidad, la 1164. Posterior a estas declaraciones, pasamos a la fase de declaración de entidades, con las cuales se estuvo trabajando a lo largo del desarrollo de la práctica. Estas entidades van desde la clásica señal de reloj, hasta los

switches(botones) de entrada, como lo pueden ser el de ejecución de instrucción, el de la entrada de datos, etc. También se declara una señal de salida dirigida hacia un display de 7 segmentos como actividad extra. Ya para finalizar con este programa, o bien para concluir con ello, se declara lo que viene siendo la arquitectura del mismo. En esta sección se implementaron 5 registros, cada uno de ellos, realiza una tarea en particular.

Para el segundo programa, se definió lo que es la ALU(Unidad Aritmética Lógica), en el cual se definieron todas las operaciones posibles para los registros que se tienen. Como se mencionó anteriormente, este programa es uno de los que es considerado como librería ya que para su desarrollo recurrimos a realizar su implementación a través de paquetes en VHDL. Como bien es sabido, para poder hacer uso de los paquetes, es necesario por primera parte, declarar la cabecera del paquete y posteriormente, definir el cuerpo del mismo. Ya con esta información, definimos todas las operaciones lógicas, aritméticas y de desplazamiento que utilizamos para las instrucciones, como lo son la Suma, la Resta, Corrimiento izquierda y derecha, compuertas, AND, OR, NOT, etc.

Y el último pero no menos importante es el programa del **LCD**, donde se definió a detalle, cómo se deben comportar las salidas del programa. En esta parte se necesitaron crear los códigos para que se mostraran las palabras en el display, para lo cual, se tenía que formar cada palabra que representaba a la instrucción con la unión de sus símbolos, así entonces, se necesitaron los códigos de cada símbolo, en caso nuestro, se realizó un filtro general, de cuáles eran los necesarios para el programa según las instrucciones que manejamos. Los cuales se describen a continuación:
Números: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Letras mayúsculas: A, C, D, I, M, N, O, R, S, T, X.
Letras minúsculas: a, c, e, i, l, m, n, o, p, q, r, s, t, u, z.
Otros Símbolos: _
Una vez se obtuvieron los símbolos necesarios se recurrió a la tabla de caracteres para **LCD (Figura 2)** en la cual se encuentran los códigos en binario para cada letra o símbolo que se requiera, una vez teniendo los códigos de estos elementos en binario se pasó a utilizar la conversión de binario a hexadecimal para obtener los códigos que se implementarán en vhdl; el resultado de lo anterior mencionado se ve reflejado en la **tabla 1.**

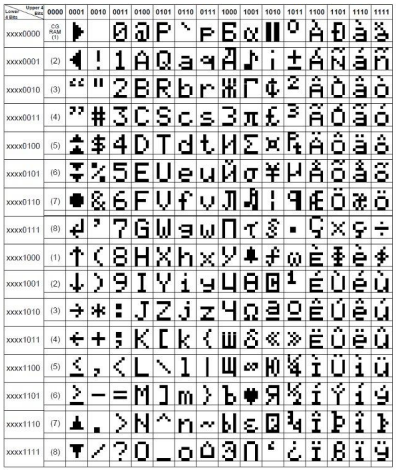

Figura 3.- Tabla de caracteres para LCD

Caracter	Binario	Hexadecimal
0	0011 0000	X"30"
1	0011 0001	X"31"
2	0011 0010	X"32"
3	0011 0011	X"33"
4	0011 0100	X"34"
5	0011 0101	X"35"
6	0011 0110	X"36"
7	0011 0111	X"37"
8	0011 1000	X"38"
9	0011 1001	X"39"

Tabla. 1- Tabla de conversión Binario <- -> Hexadecimal

Los codigos resultados de las palabras usadas en el programa son los siguientes:
Cla (Limpiar Display)
(X"FE",X"FE",X"FE", X"43",X"6C",X"41","X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Baj (Cargar la parte baja del acumulador con un dato)
X"FE",X"FE",X"FE",X"42",X"61",X"6A","X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Inx (Cargar el registro índice con una dirección)
(X"FE",X"FE",X"FE", X"49",X"6E",X"58""X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Sum (Suma)
(X"FE", X"FE", X"FE", X"53", X"75", X"6D", X"FE",X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Res (Resta)

(X"FE", X"FE", X"FE", X"52", X"65", X"73", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Mul (Multiplicación)
(X"FE", X"FE", X"FE", X"4D", X"75", X"6C", X"FE",X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
AND (Compuerta AND)
(X"FE", X"FE", X"FE", X"41", X"4E", X"44", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
NAND (Compuerta NAND)
(X"FE", X"FE", X"FE", X"4E", X"41", X"4E", X"44", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
OR (Compuerta OR)
(X"FE",X"FE", X"FE", X"4F", X"52", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
NOR (Compuerta NOR)
(X"FE", X"FE", X"FE", X"4E", X"4F", X"52", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
XOR (Compuerta XOR))
(X"FE", X"FE", X"FE", X"58", X"4F", X"52", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
XNOR (Compuerta XNOR)
(X"FE", X"FE", X"FE", X"58", X"4E", X"4F", X"52", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
NOT (Compuerta NOT))
(X"FE", X"FE", X"FE", X"4E", X"4F", X"54", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Der (Corrimiento a la Derecha)
(X"FE", X"FE", X"FE",X"44",X"65", X"72, "X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Izq (Corrimiento a la Izquierda)
(X"FE",X"FE",X”FE, X"49",X"7A",X"71","X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE", X"FE");
Espacio vacío
X"FE"

III. Resultados y Discusión

En el archivo de ALU_UC.vhd implementamos 15 instrucciones que se pedían para elaborar la práctica, en la primera parte hemos declarado los diferentes paquetes IEEE estándar que son los que describen los valores lógicos digitales y resuelven conflictos de estados, posteriormente declaramos ALU y las instrucciones, nos apoyamos de la biblioteca ieee.numeric_std.all que nos proporciona diferentes funciones aritméticas, este archivo se mandará a llamar en U_Control.vhd donde van declarados los comandos para la tarjeta y su manejo para poder mostrar las diferentes instrucciones y los mensajes que se desean mostrar en el LCD y en el display de 7 segmentos. Dentro del When mandamos a llamar las instrucciones de ALU_UC.vhd, para posteriormente ejecutarse en la tarjeta a 7 segmentos aunque se intentó al final de intentar todo lo posible antes del límite de tiempo para la entrega de la práctica, no se logró que funcionaran los mensajes en conjunto con el LCD, pues al no tener mucha experiencia con esta parte no se logró unificar las dos partes. Sin embargo sabemos como lo debimos de realizar para que el programa funcione junto con el LCD y quedó claro como se tenía que implementar el hexadecimal para mostrar los diferentes mensajes en el LCD.

En primer lugar se supone que el archivo ctrl_LCD.vhd es donde se declaran todos los mensajes posibles que se pueden mostrar en la pantalla pero también donde se declara la lógica de que mensaje se muestra dependiendo de la entrada recibida en el paquete, se debe cambiar la entrada **ent** por un vector de 5 bits el cual nos posibilita la selección de 20 mensajes distintos más los números del 0 al 9.

Después se tendría que fusionar todo el contenido del archivo LCD_M.vhd con el archivo U_Control.vhd; lo que se tendría que funcionar es la parte de los when (correspondiente a cada instrucción) con el código necesario para la configuración inicial, la limpieza y la configuración del modo de entrada del LCD así como la impresión de los mensajes línea a línea según la instrucción.

En la parte de la impresión de mensajes se tendría que actualizar la llamada al paquete de mensaje para que nos acepte dependiendo la instrucción el vector de 5 bits que identifica cual mensaje es el que se imprimirá; en esta misma parte se necesita modificar que en la línea 1 del LCD se imprima la instrucción actual junto con su código, y en la línea 2 del LCD se imprima la salida obtenida al realizar lo indicado por la instrucción.

Haciendo los cambios anteriores se cree que el funcionamiento del programa ahora si hará uso del LCD y de los mensajes dependiendo de qué instrucción se seleccione, esperando que lo que se vea en la pantalla sea:

Instrucción	Codigo_Instruccion
Resultado	

IV. Conclusiones

Durante la elaboración de este trabajo, se analizó y estudió el comportamiento de una arquitectura de 8 bits concretamente, así como los componentes que se requieren para dicha implementación. Pudimos observar que hay múltiples caminos o formas de elaborar una arquitectura como esta, diferentes métodos con diferentes elementos lo cual requiere que se analice y planee cómo se va a elaborar, con que requisitos, que tipo de registros, cual va a ser el formato de instrucciones, que entradas y que salidas tendrán, entre muchas otras cosas que se deben considerar.

A su vez, también es importante analizar el comportamiento que dicha arquitectura va a tener. Como sabemos, casi todos los dispositivos digitales si no es que **todos** los que se consideran "modernos", contienen una arquitectura similar a la que realizamos, evidentemente de un número mayor de bits y con una cantidad de recursos mayor. Es por eso de la importancia de comprender a fondo este tópico de las arquitecturas, porque como también sabemos, existen múltiples tipos, con diferentes características que se pueden implementar en diferentes situaciones y/o dispositivos.

Agradecimientos

Los autores agradecen al Instituto Politécnico Nacional por el apoyo brindado respecto a los materiales provistos durante la realización de esta mencionada práctica. También se agradece a las familias de cada uno de los autores por el apoyo del material electrónico y digital antes, durante y posterior a la conclusión de la mencionada.

Referencias

[1] Cardoso Llach, Daniel, Capdevila Werning, Remei Arquitectura,diseño y computación. Dearq [en línea]. 2009, (4), 136-140[fecha de Consulta 10 de Marzo de 2022]. ISSN: .

[2] Cifuentes Quin, Camilo Andrés Arquitectura y computación ¿determinismo o mediación?: del paradigma informacional hacia una tectónica digital. Dearq [en linea]. 2012, (10), 22-35[fecha de Consulta 10 de Marzo de 2022]. ISSN: .