Arquitectura Básica de 8 bits

Maxime Ishin Montealegre Terada
Luis Ricardo Luna Lopez
Daniel Hernández Cardenas
Departamento de ingeniería en sistemas computacionales, ESCOM IPN

maxime.montealegre@gmail.com
dany2501.dhn@gmail.com

***Resumen*— La realización de esta práctica se basa en realizar una arquitectura básica de 8 bits, la cual con apoyo esencial del profesor se pudo entender y realizar. Se va a basar en un BCD de 7 segmentos, en el que el código realizado fue gracias a la ayuda del profesor de la clase, en el que nos ayuda a entender y sobre todo a realizar el código en el que nosotros tuvimos que revisar a detalle, esto debido a que contaba con algunos errores, los cuales poco a poco fueron solucionados con la ayuda en general de toda la clase. Al momento de compilar y observar que el código estaba ya corregido se realizó el paso de agregar las instrucciones que se solicitaban para realizar esta práctica, en las que poco a poco se fue entendiendo como realizar este paso, ya que durante la clase se realizaron ejemplos de cómo se introducían estas instrucciones de manera correcta.**

***Palabras Clave* — Arquitectura, BCD, Segmentos.**

I. Introducción

En un sistema existen dos tipos de buses, los cuales son de datos y de direcciones. El bus de datos realiza el intercambio de información, mientras que el de direcciones selecciona el contenido del mapa de memoria. Estos buses controlan a los elementos de memoria y los sistemas de entrada/salida o periféricos.

Figura 1. Buses.

La ALU es un circuito que permite realizar operaciones lógicas y aritméticas, además de contar con una serie de registros para almacenar los datos, y bits de información sobre los resultados, a estas funciones se les da el nombre de banderas. Las banderas más comunes son: Carry, Auxiliary, Carry, Borrow, Overflow, Parity, Zero.

La ALU solo realiza operaciones, por lo que no toma decisiones dentro del sistema. Es importante que las entradas contengan el signo y la magnitud correspondiente a la operación, además de requerir un mecanismo de control que le permita saber el tipo de operación a realizar.

Una ALU se forma por medio de una "célula" de tipo bit-slice, la cual permite realizar sumas o más operaciones lógicas entre dos bits. La única limitante respecto a esta "célula" es que a y b no estén negadas.

Las partes de la ALU serían los sumadores - restadores, los operadores lógicos, un acumulador, un registro auxiliar y uno de salida, señales de control que indiquen la operación y las banderas anteriormente mencionadas.

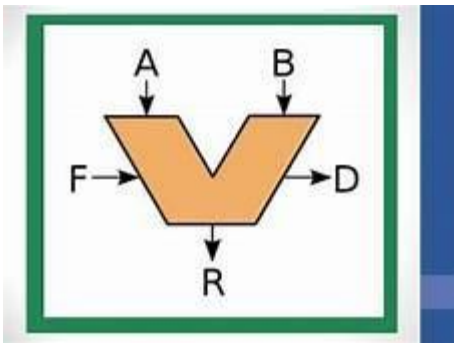

Figura 2. Diagrama de la Unidad Aritmética Lógica (ALU).

La unidad de control es un elemento capaz de realizar operaciones lógicas y aritméticas además de participar en la toma de decisiones y permitir el intercambio de datos entre localidades de memorias y/o periféricos. Permite el procesamiento de la información, la gestión de la memoria y

de los puertos de comunicación, además de ser la responsable de la interpretación y ejecución de las instrucciones contenidas en la memoria principal. La dirección hacia la memoria se realiza por medio del bus de direcciones y el bus de datos.
Se encarga de codificar el Código de operación, además de generar las señales para leer las entradas de la ALU.

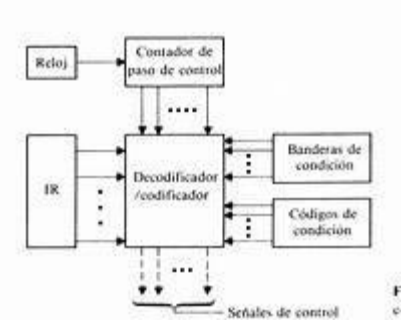

Figura 3. Diagrama de una Unidad de Control.

Los registros son localidades de almacenamiento temporal de datos de alta velocidad y están dedicados al control, y solo la unidad de control tiene acceso a ellos.

- Registro de direcciones de la memoria: Es el registro de enlace entre la CPU y el registro de direcciones. El número de bits que hay es igual al del registro de direcciones.
- Registro de datos: Proporciona un área de almacenamiento temporal en la que los datos se intercambian entre la CPU y la memoria. Los datos pueden ser instrucciones o datos del operando.
- Registro de instrucciones: Conserva el código de operación de la instrucción en todo el ciclo de la máquina, su longitud va a ser igual a la del código de operación.

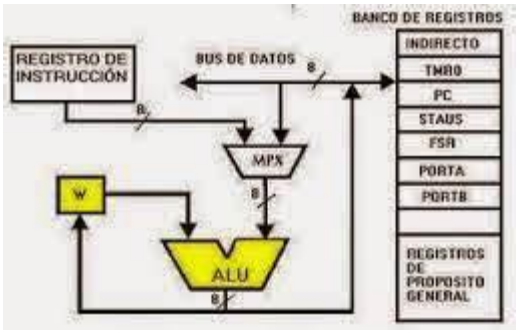

Figura 4. Diagrama.

Existen 5 tipo de instrucciones, las cuales son aritmética - lógico, movimiento de datos, operaciones de datos, control del programa y, de entrada - salida.

- **Aritmética - Lógico**: Son operaciones binarias y operaciones AND, NAND, NOR, XAND, XOR.
- **Movimiento de Datos**: Da por resultados la copia de datos desde una localidad de operando a otra, requieren información que identifique los operandos fuentes y destinos.
- **Operaciones de Datos**: Se efectúan con un conjunto de operandos y no con un solo operando. Hace posible que un programa se adapte a la secuencia inherente al ciclo de máquina de la computadora.
- **Control del Programa:** Contienen información acerca del estado inactivo, ocupado, etc. además de almacenar información de tipo control. Se utiliza principalmente para proporcionar una imagen global del hardware.
- **Entrada - Salida:** Se define como el acceso a la memoria o a un periférico en el cual se requiere el mismo conjunto de instrucciones.

La memoria SDRAM espera a la señal de un reloj del sistema antes de transferir los datos, algunas veces es usada en computadoras con velocidades de reloj de 66 y 133 MHz. Los datos se transfieren mediante la técnica de doble bombeo, es decir los datos se transfieren cuando el reloj aumenta y disminuye, además de no requiere cambios ni ajustes en la frecuencia del reloj, transfiere los datos en piezas de 64 bits. La tasa de transferencia real se calcula multiplicando la frecuencia del reloj de la memoria por dos, luego se multiplica por 64 ya que es el número de bits que se transfirieron.

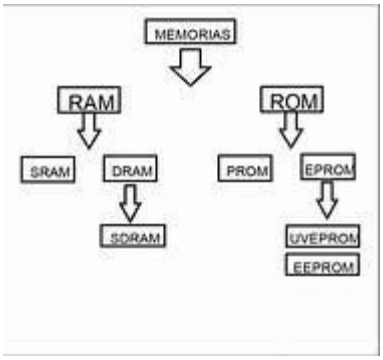

Figura 5. Tipos de Memorias.

Metodología/Desarrollo

A. *En esta práctica, utilizaremos el lenguaje VHDL, el cuál es un lenguaje de descripción de hardware. El significado de VHDL es VHSIC (Very High Speed Integrated*

B. *También estaremos utilizando la herramienta Quartos, es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en HDL (Lenguaje de Descripción de Hardware), el cuál será nuestro IDE para poder compilar y programar nuestro Altera DE2-115.*

C. *Para poder almacenar datos de forma temporal, utilizaremos la memoria RAM de tipo distribuida, en ésta,los elementos lógicos de la FPGA son conectados formando una memoria.*

D. *Se programarán 20 funciones distintas en el FPGA, los cuales son:*
*1)**Add.** Suma la parte baja del acumulador con la entrada que el usuario ingrese a través de los switches.*
*2)**Sub**. Resta la parte baja del acumulador con la entrada que el usuario ingrese a través de los switches.*
*3)**Mul**. Multiplica la parte baja del acumulador con la entrada que el usuario ingrese a través de los switches.*
*4)**Div.** Divide la parte baja del acumulador con la entrada que el usuario ingrese a través de los switches.*
*5)**Not**. Realiza la operación lógica NOT de la parte baja del acumulador.*
*6)**Or**. Realiza la operación lógica OR de la parte baja del acumulador.*
*7)**Xor.** Realiza la operación lógica XOR de la parte baja del acumulador.*
*8)**And**. Realiza la operación lógica AND de la parte baja del acumulador.*
*9)**Nand**. Realiza la operación lógica NAND de la parte baja del acumulador.*
*10)**Nor.** Realiza la operación lógica NOR de la parte baja del acumulador.*
*11)**Xnor.** Realiza la operación lógica XNOR de la parte baja del acumulador.*
*12)**Inc.** Incrementa el acumulador en una unidad*
*13)**Dec.** Incrementa el acumulador en una unidad*
*14)**Set.** Asigna al acumulador el valor "1111111111111111"*
*15)**SLL.** Realiza un shift al bit hacia la izquierda*
*16)**SLR.** Realiza un shift al bit hacia la derecha*

E. *Compilamos y cargamos nuestro programa en el FPGA*

III. Conclusiones

Durante la realización de la práctica se tuvieron varias dudas en las que el profesor siempre nos ayudaba a detectarlas o ya bien a corregirlas. A lo largo del tiempo que se estuvo realizando esta práctica se pudo notar muchas dudas respecto al manejo del lenguaje VHDL, por lo que poco a poco durante las sesiones se fueron quitando dudas a través de ejemplos y sobre todo con la práctica de algunos códigos los cuales fueron proporcionados por el profesor. Al momento de empezar a implementar esta práctica se pudo poco a poco ir visualizando cada parte del código, desde el uso y las funciones que generan las bibliotecas hasta el momento en que se ingresaron las instrucciones. Conforme comprendemos cada parte del código, poco a poco se fue analizando la forma en cómo podíamos desarrollar el código sin que en ningún momento se tuvieran errores, hablo al momento de compilar. Un detalle que notamos fue que la mayoría de los errores mostrados es debido a fallas puntuales en la lógica del código, ya sea por falta de bibliotecas o por otros más significativos y menos visibles.

Referencias

[1] Marcos Sánchez-Élez. (2014). INTRODUCCIÓN A LA PROGRAMACIÓN EN VHDL. 07/10/2021, de Facultad de Informática Universidad Complutense de Madrid Sitio web: https://eprints.ucm.es/id/eprint/26200/1/intro_VHDL.pdf

[2] Jim Lewis. (2008). VHDL-2008: Why It Matters . 07/10/2021, de verificationhorizons Sitio web: https://pld.ttu.ee/~alsu/vhdl-2008-why-it-matters_vh-v8-i3.pdf

[3] Andres Garcia Garcia (2014). Procesadores Disponible en: http://homepage.cem.itesm.mx/garcia.andres/PDF201411/Arquitectura%20Computacional.pdf

[4] David Sandoval (). RAM vs SDRAM Disponible en: https://techlandia.com/ddr-vs-sdram-sobre_98129/

[5] David Sandoval (). Diferencias entre memorias. Disponible en: https://techlandia.com/diferencia-sram-dram-info_235151