Descripción en VHDL de la interconexión de componentes de un procesador básico

Carlos Federico Hernández Farfán

Instituto Tecnológico Superior de Irapuato, Carr. Irapuato-Silao km 12.5, 462 60 6 79 00

cahernandez@itesi.edu.mx

Jonathan Paris Vargas Mosqueda

Instituto Tecnológico Superior de Irapuato, Carr. Irapuato-Silao km 12.5, 462 60 6 79 00

parís.47@hotmail.com

Resumen

Dentro del estudio de un curso básico de Arquitectura de Computadoras se abordan temas como lógica digital, componentes digitales, transferencias de registros y micro-operaciones, así como la interconexión de los componentes de un procesador básico. Debido a la necesidad de realizar un balance entre los conceptos teóricos, su simulación e implementación, en este trabajo se presenta la descripción de la interconexión de componentes de un procesador básico empleando el lenguaje de descripción de hardware VHDL y la técnica de diseño asistido por computadora (CAD). Se hace énfasis en la descripción estructural, el uso de sólo una librería estándar y un sólo paquete de la librería con el objeto de construir el sistema digital a partir de los componentes digitales básicos, las pruebas del sistema digital se basan en las transferencias entre registros y las operaciones necesarias para completar el ciclo de una instrucción. Se puede observar que la capacidad empleada de la tarjeta en la implementación física en un dispositivo de arreglos de compuertas programables en campo (FPGA) es menor al 1%.

Palabras Claves: Arquitectura de computadoras, FPGA, VHDL.

1. Introducción

En este trabajo se emplea la técnica CAD para el desarrollo de un sistema digital que apoya el estudio básico de Arquitectura de Computadoras. La técnica CAD de sistemas digitales permite reducir el tiempo de desarrollo de un diseño, facilita la comprensión y verificación del funcionamiento del sistema digital. También permite la implementación del diseño del sistema digital en un dispositivo programable [1]. Esta técnica requiere la descripción del sistema por medio de un lenguaje, en este caso VHDL; posteriormente se realiza la simulación del funcionamiento y finalmente su implementación en un dispositivo programable. Existen diversos dispositivos lógicos programables, que pueden reemplazar a los circuitos de pequeña, mediana y muy alta escala de integración ahorrando espacio y costo del diseño [2]. En este caso se emplea por su versatilidad, rendimiento y alta capacidad un dispositivo FPGA.

Los conceptos básicos cubiertos en un curso de Arquitectura de Computadoras incluyen compuertas lógicas, circuitos combinacionales, flip-flop´s, circuitos secuenciales; así como decodificadores, registros, multiplexores, contadores y memorias. La implementación física de éstos circuitos no constituye mayores contratiempos, pero a medida que se avanza en conceptos como la transferencia entre registros, implementación de un bus o camino de datos, unidad aritmética-lógica y de corrimiento (ALU) construidos a partir de componentes básicos para conformar un procesador, se complica la implementación física para adquirir los componentes, interconectarlos, realizar algún cambio o identificar fallas, el tiempo requerido aumenta e incluso el costo llega a ser significativo para los estudiantes.

La técnica CAD permite describir los componentes digitales, simularlos independientemente y posteriormente integrarlos en un proyecto. Las transferencias de datos entre los componentes se realizan en tiempos sumamente pequeños por lo que es no es posible visualizarlos; sin embargo la simulación permite observar transferencias de datos que ocurren en tiempos de micro y nanosegundos. También la descripción en VHDL de la unidad de control permite facilitar la comprensión del sistema digital ya que

se establece cuáles son los las señales que se deben generar para controlar la transferencia de los datos y lograr un operación en particular del procesador.

2. Desarrollo

A continuación se muestra el proceso de interconexión de los componentes de un procesador básico basado en la organización y diseño básico de computadoras de [3]. Mediante este procesador básico se puede mostrar el proceso de diseño sin demasiadas complicaciones [3]. Con fines ilustrativos y por simplicidad en este caso se describe un procesador que tiene un sistema de bus para datos y direcciones de 4 bits; 8 registros de 4 bits tales como el registro de direcciones, registro contador de programa, registro de datos, acumulador, registros de entrada y salida, registro de instrucciones y registro temporal; una memoria RAM y una unidad lógica-aritmética y de corrimiento. La capacidad de esta arquitectura se puede ampliar mediante la descripción en VHDL y básicamente no afecta la estructura general del sistema. En la Fig. 1 se muestra la estructura del procesador.

La interconexión de los componentes internos del procesador se realiza mediante la técnica de diseño jerárquico top-down. Se establece una jerarquía de funcionamiento de orden superior constituida por bloques con una función específica, a continuación se realiza la descripción del bloque o de segundo nivel el cual puede estar integrado por bloques que a su vez requerirán su propia descripción hasta llegar a la descripción de los componentes más simples; en el primer nivel se determinan las entradas y salidas del sistema digital y la interconexión entre los componentes del segundo nivel continuando este proceso hasta llegar a la descripción de los componentes individuales [4]. Cada componente se describe empleando una sola librería estándar y un solo paquete de la librería con el fin de construir el sistema a partir de sus componentes básicos.

Todo el proceso se realiza mediante el uso del software Quartus® II versión 11.1 Web Edition, las simulaciones se realizaron mediante el uso del software Altera® U.P Simulator Qsim versión 11.1 Web Edition y se empleó la tarjeta Cyclone® II. Todos estos productos tecnológicos desarrollados por la compañía Altera® Corporation [5].

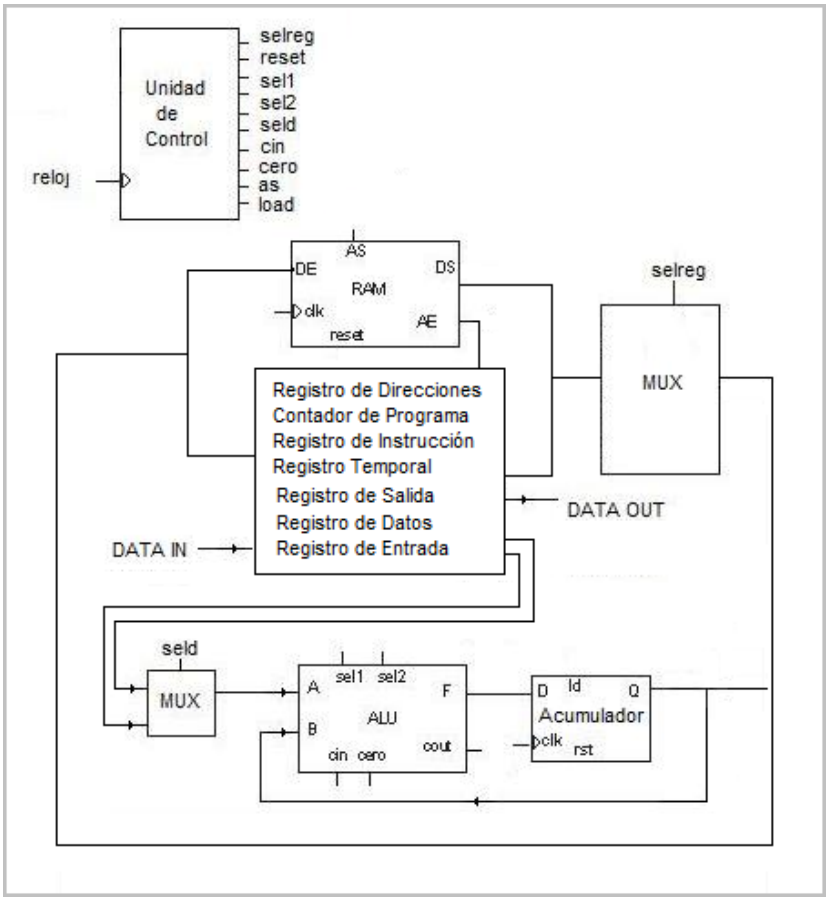

Fig. 1. Componentes del procesador básico basado en [3].

En la Fig. 2 se muestran los componentes del cual está constituido el procesador básico. Se requiere un archivo con la descripción de primer nivel jerárquico en donde se declaran las entradas, salidas y señales intermedias del sistema digital; en este archivo también se declaran cada uno de los componentes y la forma en que se interconectan [6]. Observe que tanto la memoria RAM como la ALU requieren una descripción de segundo nivel jerárquico. En la parte derecha se observan los componentes de la ALU.

En la Fig. 3 se muestra la forma en que se declaran en el código VHDL de primer nivel jerárquico algunos de los componentes del procesador, por ejemplo registros, ALU, etc. Cada uno se encuentra descrito en un archivo aparte que puede ser compilado, probado y posteriormente integrado al proyecto.

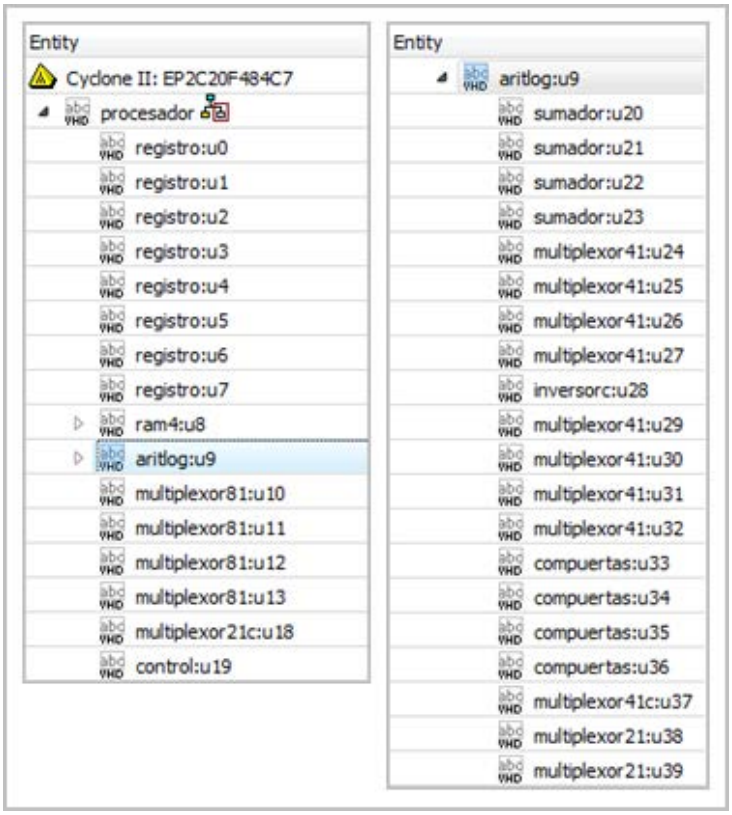

Fig. 2. Componentes del procesador básico.

En la Fig. 4 se muestra la forma en que en el archivo de primer nivel jerárquico se describe la interconexión de algunos componentes como registros y memoria RAM.

```
component registro port (
   di : in std_logic_vector (3 downto 0);
   ld,rst,clk : in std_logic;
   q : out std_logic_vector (3 downto 0));
end component;

component aritlog port (
   a,b : in std_logic_vector (3 downto 0);
   sel1 : in std_logic_vector (1 downto 0);
   sel2 : in std_logic_vector (1 downto 0);
   cin,cero : in std_logic;
   f : out std_logic_vector (3 downto 0);
   cout : out std_logic);
end component;

component multiplexor81 port (
   m7,m6,m5,m4,m3,m2,m1,m0 : in std_logic;
   s : in std_logic_vector (2 downto 0);
   y : out std_logic );
end component;
```

Fig. 3. Declaración de componentes del procesador básico.

```
u0: registro port map (bus0,load(0),reset,reloj,datoout);
u1: registro port map (bus0,load(1),reset,reloj,dato1);
u2: registro port map (bus0,load(2),reset,reloj,dato2);
u3: registro port map (datoin,load(3),reset,reloj,in1);
u4: registro port map (alu,load(4),reset,reloj,dato4);
u5: registro port map (bus0,load(5),reset,reloj,dato5);
u6: registro port map (bus0,load(6),reset,reloj,dato6);
u7: registro port map (bus0,load(7),reset,reloj,dato7);
u8: ram4 port map (bus0,dato7,reloj,as,dato8,reset);
```

Fig. 4. Interconexión de componentes.

Como se mencionó anteriormente cada componente puede ser probado aparte y posteriormente integrado al proyecto. En la Fig. 5 se muestra la simulación del funcionamiento de la ALU, en la cual se realizan ocho operaciones aritméticas, cuatro operaciones lógicas, corrimiento a la derecha y corrimiento a la izquierda. Los datos de entrada a la ALU corresponden a las variables a y b con los valores en decimal 5 y 3 respectivamente, con la variable sel2 se selecciona el tipo de operación ya sea aritmética (sel2 = 00), lógica (sel2 = 01), de corrimiento a la derecha (sel2 = 10) o corrimiento a la izquierda (sel2 = 11). Las 8 operaciones aritméticas se determinan a partir de la variable sel1 y el carry de entrada denominado cin, las cuales son suma, suma con carry, resta con préstamo, resta, transferir, incrementar, decrementar y nuevamente transferir.

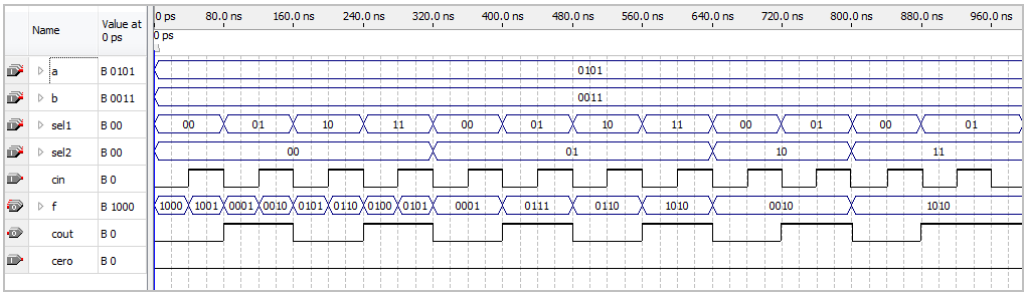

Fig. 5. Operaciones de la unidad aritmética-lógica y de corrimiento.

Las operaciones lógicas se determinan empleando las variable sel1 y sel2, las cuales son operación and (sel1 = 00), or (sel1 =01), xor (sel1 = 10) y not (sel1 = 11).

Finalmente para este ejemplo en el código de nivel 3 jerárquico se describe cada componente individual del cual está conformada la arquitectura, por ejemplo se describen multiplexores, registros, decodificadores, sumadores y compuertas lógicas. En la Fig. 6 se muestra el ejemplo de la descripción en VHDL de un multiplexor de 4 a 1 cuádruple.

```
library ieee;
use ieee.std_logic_1164.all;

entity multiplexor41c is port (
   e0,e1,e2,e3 : in std_logic_vector (3 downto 0);
   s : in std_logic_vector (1 downto 0);
   y : out std_logic_vector (3 downto 0));
end multiplexor41c;

architecture arqmultiplexor41c of multiplexor41c is
begin
  process (s,e0,e1,e2,e3)
  begin
    case s is
      when "00" => y <= e0;
      when "01" => y <= e1;
      when "10" => y <= e2;
      when others => y <= e3;
    end case;
  end process;
end arqmultiplexor41c;
```

Fig. 6. Multiplexor 4 a 1 cuádruple basado en [4].

Una vez descritos los componentes se realiza la asignación de las variables de entrada y salida a los correspondientes pines de la tarjeta con el dispositivo FPGA [7]. En la Fig. 7 se muestra la asignación de pines para el circuito.

	Status	From	To	ssignment Nam	Value	INTERRUPTOR / LED
1	Ok		datoin[3]	Location	PIN_W12	SW [4]
2	Ok		datoin[2]	Location	PIN_V12	SW [3]
3	Ok		datoin[1]	Location	PIN_M22	SW [2]
4	Ok		datoin[0]	Location	PIN_L21	SW [1]
5	Ok		reloj	Location	PIN_L22	SW [0]
6	Ok		datoout[3]	Location	PIN_Y19	LEDR [3]
7	Ok		datoout[2]	Location	PIN_U19	LEDR [2]
8	Ok		datoout[1]	Location	PIN_R19	LEDR [1]
9	Ok		datoout[0]	Location	PIN_R20	LEDR [0]

Fig. 7. Asignación de pines

3. Resultados

En los diferentes ejemplos de diseño realizados, se debe poner atención en que la compilación se realice de forma exitosa y que los componentes descritos puedan ser realizables físicamente en la tarjeta. En la Fig. 8 se muestra el resumen del informe de la compilación de la descripción del procesador básico. Como puede observarse se completaron exitosamente todas las tareas de compilación y el porcentaje de utilización para la tarjeta es menor del 1% de su capacidad física.

Se prepararon dos pruebas, una basada en la simulación para comprobar las transferencias de datos internas del procesador y otra en la implementación física de la arquitectura en el dispositivo FPGA.

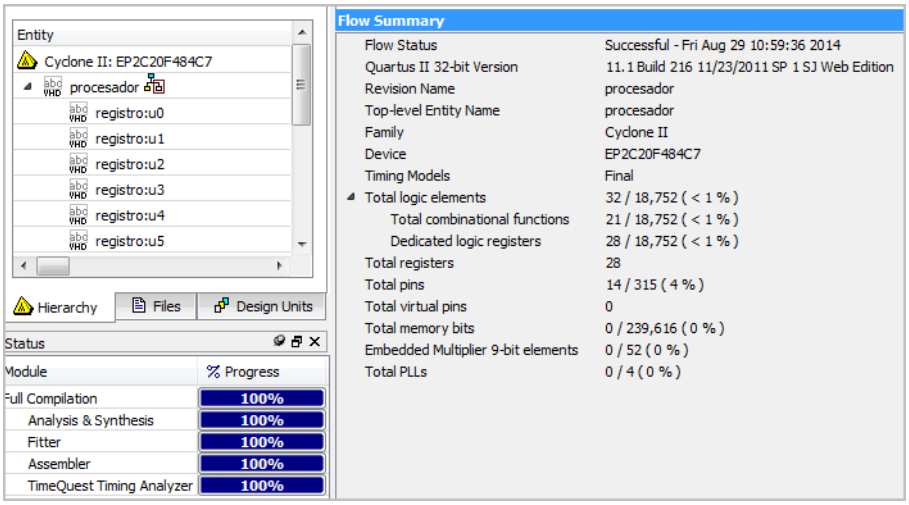

Fig. 8. Resumen de compilación.

En la Fig. 9 se muestra la simulación de una prueba sencilla del funcionamiento de la descripción de la arquitectura del procesador. Se se inicia con un reset a todos los componentes mediante la señal reset activa en bajo para asegurarse que el contenido de todos los registros y el contenido de la memoria sea cero. La variable datoin se emplea para proporcionar el dato de entrada al sistema. En este caso se fija en un valor decimal

de 2 después del reset. Mediante la señal selreg se selecciona uno de los ocho registros o memoria del cual ha de aparecer su contenido en el bus, el valor 011 selecciona al registro acumulador. Mediante la señal load se determina cuál registro esta habilitado para cargar el dato en el momento en que ocurra el pulso de reloj. Al momento de ocurrir el segundo pulso de reloj se carga el registro de entrada con el dato de entrada. Las señales sel1, sel2 y cin determinan el comportamiento de la ALU la cual se encuentra conectada directamente al registro de entrada, el resultado de la ALU se puede cargar directamente al acumulador y el contenido del acumulador retroalimentarse a la ALU.

De esta manera el dato recién cargado en el registro de entrada, pasa a través de la ALU si modificarse, se carga en el acumulador derante el tercer pulso y es mostrado en el bus. Una vez que el dato se encuentra en el bus puede estar disponible para cualquiera de los registros o como dato de entrada a la memoria. A continuación, se selecciona el registro contador de programa para que en el siguiente pulso de reloj cargue el dato, el cuarto pulso de reloj también provoca que en la localidad cero de la memoria se guarde el dato del bus. En el bus se muestra ahora el contenido del contador de programa y en el quinto pulso de reloj el contenido del contador de programa a través del bus es cargado en el registro de direcciones, el contenido del registro de direcciones es mostrado en el bus.

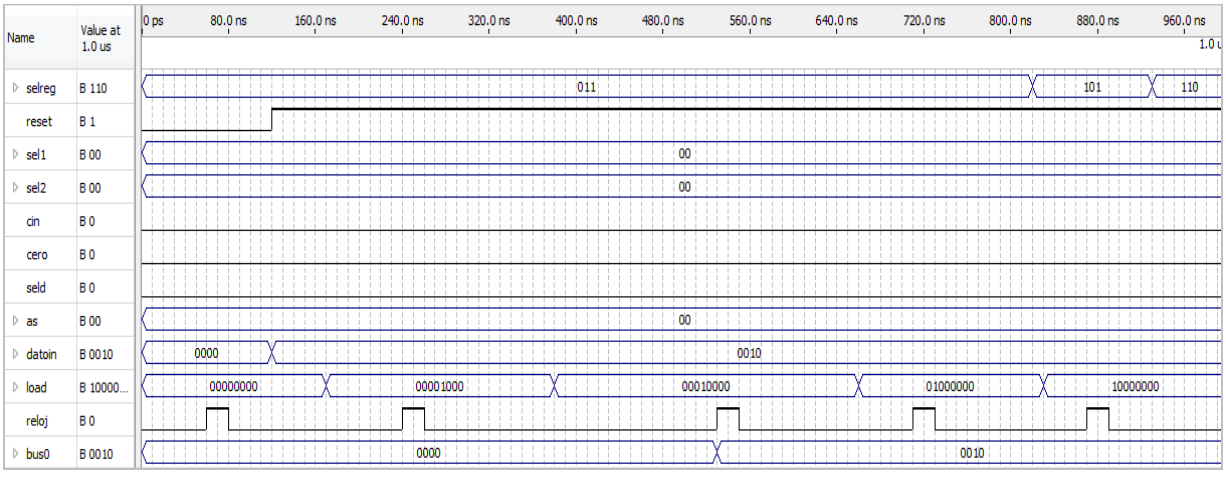

Fig. 9. Dato de entrada y transferencia entre registros.

En la Fig. 10 se muestra la simulación del ingreso de un dato que es retroalimentado hacia la ALU. Se inicia con un reset, el dato es cargado en registro de entrada, durante el tercer pulso de reloj el dato es cargado en el acumulador, es seleccionado para aparecer en el bus y también se encuentra retroalimentado a la ALU. De esta manera durante el cuarto pulso de reloj se aplica a la ALU el dato de entrada así como el contenido del acumulador y se selecciona la operación suma, cuyo resultado es cargado nuevamente en el acumulador y mostrado a través del bus.

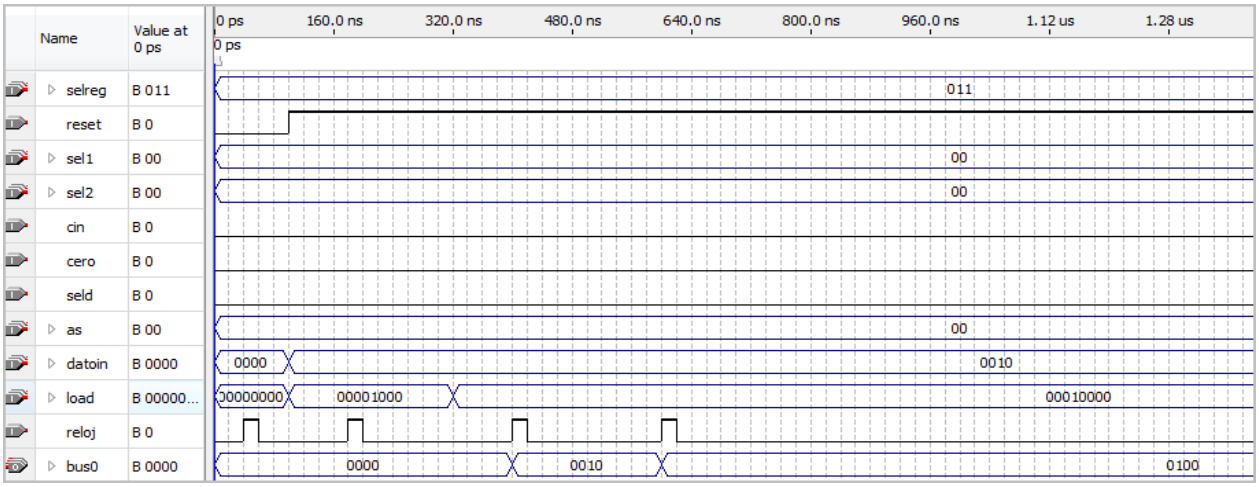

Fig. 10. Retroalimentación del contenido del acumulador a través de la ALU

En la Fig. 11 se muestra la prueba de la transferencia de datos entre registros y la memoria RAM. Nuevamente se inicia con un reset, a continuación se carga el registro de entrada con el dato de entrada el cual pasa a través de la ALU sin modificarse para cargarse en el acumulador en el tercer pulso de reloj, el contenido del acumulador es seleccionado por la variable selreg para aparecer en el bus. Una vez que el dato se encuentra en el bus, se selecciona el registro contador de programa para que durante el cuarto pulso de reloj sea cargado, durante este mismo pulso de reloj las señales aplicadas a la memoria provenientes del contenido del registro de direcciones y el contenido del bus provocan que en la localidad 00 se almacene el contenido del bus. Durante el sexto pulso de reloj el contenido del contador de programa a través del bus es cargado en el

registro de direcciones proporcionándole a la memoria una nueva dirección para almacenar un dato, además el contenido del registro de direcciones es mostrado a través del bus. Mediante la señal as se aplica la dirección del dato de salida de la memoria y mediante la señal selreg se selecciona la memoria para que su contenido aparezca en el bus. De esta manera puede mostrarse el contenido de las primeras localidades de la memoria. A continuación el contenido de la memoria es cargado en el registro de datos, mostrado en el bus y aplicado como un operando a la ALU. Finalmente durante el octavo pulso de reloj, el dato pasa a través de la ALU, es cargado en el acumulador y mostrado en el bus.

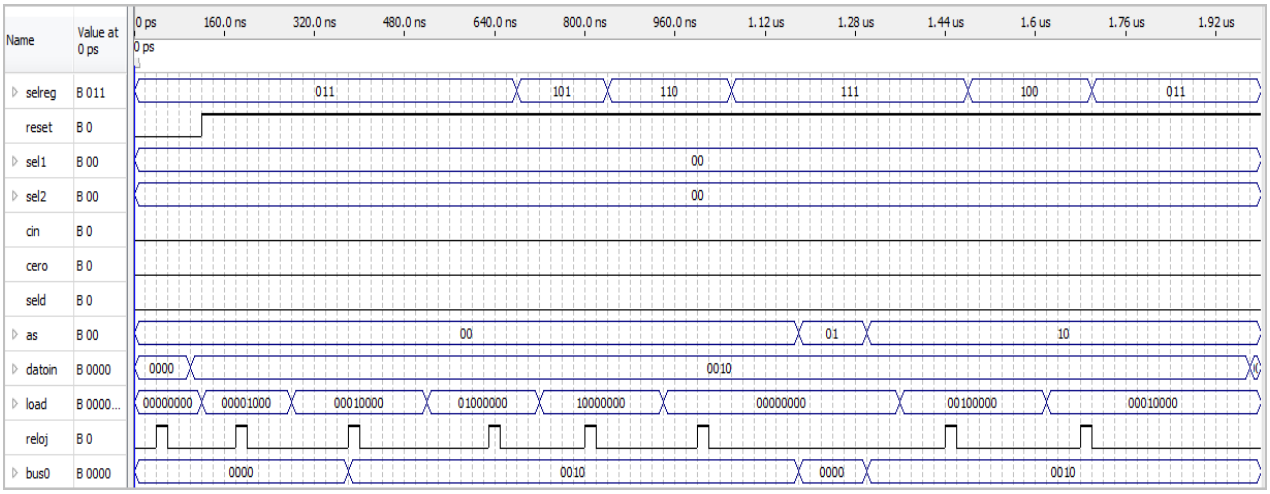

Fig. 11. Transferencia entre registros y memoria

En la Fig. 12 se muestra la simulación para la prueba que se va a realizar físicamente en la tarjeta, la cual consiste en ingresar un dato a través del registro de entrada, pasarlo a través de la ALU sin modificaciones y cargarlo en el acumulador; mostrarlo en el bus y guardarlo en la memoria. A continuación el dato se lee de la memoria y se deja disponible en el bus, se carga en el registro de datos y se pasa como operando a la ALU en donde se suma con el contenido del acumulador y se envía al registro de salida. La anterior secuencia de transferencias se basa en la unidad de control que genera las señales

internas para lograr las transferencias entre registros, la microoperaciones en la ALU, la escritura y lectura en la memoria, el ingreso de un dato y mostrar el dato a la salida.

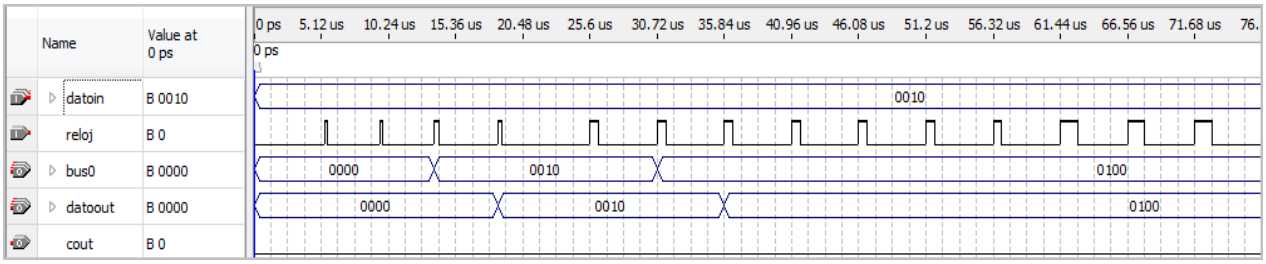

Fig. 12. Simulación para la implementación física

En la Fig. 13 se muestra la implementación correspondiente a la simulación de la Fig. 12. Se puede observar en la parte inferior derecha el interruptor sw[0] empleado para aplicar el pulso de reloj, los interruptores sw[4] … sw[1] se emplean como dato de entrada y los leds rojos son indicadores del dato de salida. De acuerdo a la simulación al introducir el dato con el valor en decimal 2 y al realizarse las transferencias indicadas anteriormente se obtiene el dato de salida con el valor en decimal 4.

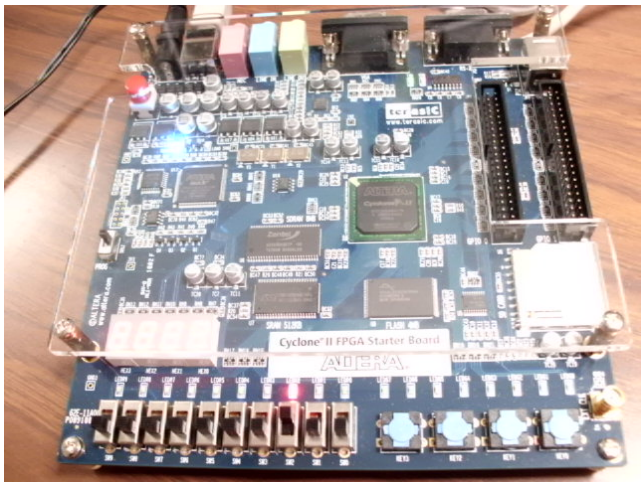

Fig.13. Implementación del procesador en la tarjeta con FPGA

4. Discusión

En el procesador básico el ciclo de instrucción consiste en: 1. Leer la instrucción de la memoria, 2. Determinar la instrucción a realizar, 3. Determinar la dirección a utilizar por la instrucción, 4. Ejecutar la instrucción [3]. Considerando las fases del ciclo de instrucción, puede observarse que mediante la descripción en VHDL de la interconexión de los componentes de un procesador básico, pueden ser realizadas las transferencias de datos entre los componentes que la integran; es decir es capaz de almacenar datos en la memoria que se pueden tratar de instrucciones de un programa, ser transferidos a cualquiera de los registros como por ejemplo al registro de instrucciones para su posterior decodificación.

Un conjunto de instrucciones está completo si el procesador incluye instrucciones de los diferentes tipos: 1. Instrucciones aritméticas, lógicas y de corrimiento, 2. Instrucciones para escribir y leer la memoria y los registros, 3. Instrucciones de control del programa como saltos condicionales e incondicionales, 4. Instrucciones para ingresar datos al sistema o enviar datos a la salida [3]. Mediante el control de la ALU es posible realizar ocho instrucciones aritméticas, cuatro lógicas y dos de corrimiento. Es posible la transferencia de datos entre cualquiera de los ocho registros y hacia o desde la memoria. Se puede controlar el curso del programa al cargar el registro contador de programa con una dirección en particular y transferirse al registro de direcciones para obtener de la memoria el dato correspondiente. También es posible ingresar datos a través del registro de entrada, así como enviar datos de salida. Se prueba la interconexión de los componentes del procesador básico mediante su capacidad de realizar la transferencia de datos entre los registros, memoria y entrada y salida de datos. Se pueden realizar todas las transferencias necesarias para la ejecución de un ciclo de instrucción.

La etapa de temporización y control está determinada por la unidad de control la cual realiza las transferencias necesarias para la prueba física. Para el uso posterior del procesador se debe realizar la unidad de control con la secuencia en particular requerida por el ciclo de instrucción. A través de las simulaciones se permite el estudio de la función

de cada componente del procesador y la prueba física determina su factibilidad de implementación.

5. Conclusiones

Mediante la técnica CAD se realizó la descripción de la interconexión de los componentes un procesador básico. Se empleó el lenguaje de descripción de circuitos VHDL. Se realizaron las transferencias entre registros y microoperaciones necesarias para completar el ciclo de una instrucción. Se considera que el empleo de ésta técnica facilita el estudio básico de arquitectura de computadoras ya que permite abordar los temas de forma gradual iniciando desde los componentes más sencillos los cuales se pueden probar por separado e integrarlos posteriormente a un proyecto más complejo. El uso de esta técnica es viable ya que se emplea una versión software accesible para los estudiantes, la tarjeta es de bajo costo y no se requiere equipamiento de laboratorio adicional.

Se puede observar que la capacidad empleada de la tarjeta es mínima por lo que se puede proponer ampliar el panorama de los estudiantes para buscar aplicaciones de cómputo en diferentes áreas. Con la realización de este proyecto se pretende adquirir conocimientos de la técnica CAD por parte de la comunidad estudiantil y académica del Instituto Tecnológico Superior de Irapuato para incorporarla en la dinámica de los cursos del área de sistemas digitales de la carrera de Ingeniería en Sistemas Computacionales y servir como base para el planteamiento de proyectos más robustos donde se requiera el cómputo de alto desempeño.

6. Referencias

[1] Quartus® II Introduction for VHDL Users. Altera® Corporation. Estados Unidos de America. 2011.

[2] D. G. Maxinez, J. Alcalá, VHDL EL arte de programar sistemas digitales. Primera Edición. Año 2003. Cecsa. México. P 4.

[3] M. M. Mano, Arquitectura de Computadoras. Tercera Edición. Año 2000. Prentice Hall. México. P 131, 139, 148, 143.

[4] R. de J. Romero Troncoso, Electrónica Digital y Lógica Programable. Primera Edición. Año 2007. Universidad de Guanajuato. México. P 358, 359, 118.

[5] www.altera.com. Enero 2012.

[6] F. Pardo Carpio, J. A. Boluda Grau, VHDL Lenguaje para síntesis y modelado de circuitos. Primera Edición. Año 1999. Alfaomega. México. P 32.

[7] Cyclone® II FPGA Starter Development Board Reference Manual. Altera® Corporation. Estados Unidos de America. 2011.

7. Autores

M. en I. Carlos Federico Hernández Farfán es Ingeniero en Electrónica por el Instituto Tecnológico de Celaya, obtuvo su título de Maestría en Ingeniería Eléctrica por la Universidad de Guanajuato y es profesor de la carrera de Ing. en Sistemas Computacionales del Instituto Tecnológico Superior de Irapuato.

Jonathan Paris Vargas Mosqueda es estudiante de la carrera de Ingeniería en Sistemas Computacionales en el Instituto Tecnológico Superior de Irapuato.