Descripción VHDL de un Microcontrolador

Luis Martinez Belot[1], **Alejandro Leyes**[2]
Facultad de Informática, UNLP
[1] lmartinezbelot@hotmail.com [2] alejandroleyes@hotmail.com

Horacio A. Villagarcía Wanza
Profesor Adjunto DE, Facultad de Informática, UNLP.
Profesional Principal CIC, III-LIDI, UNLP
Calles 50 y 120, 2do piso La Plata, Argentina
hvw@info.unlp.edu.ar

Resumen

En este trabajo se presenta una descripción utilizando el lenguaje de descripción de hardware VHDL (Very High Speed Integrated Circuit Hardware Description Language - VHSIC HDL) de un microcontrolador. Se define una estructura interconectada de componentes para describir la mayor parte los integrantes de una familia de microcontroladores comerciales. Cada componente y la estructura propuesta fueron simulados y sus resultados fueron los esperados. Dado que los elementos que constituyen el microcontrolador tienen las características de un sistema complejo (CPU, Memoria y puertos de E/S), la implementación física de un modelo completo nos llevaría a obtener un sistema en chip (System On a Chip - SOC). La descripción completa también permite definir un módulo IP utilizable en diseños con lógica programable.

Palabras claves: VHDL, Microcontrolador, RISC, System On a chip (SOC), Lógica programable.

Abstract

In this paper a microcontroller description in Very High Speed Integrated Circuit Hardware Description Language (VHDL) is presented. An interconnected component-based structure is defined to include most members of a commercial microcontroller family. Every component and the proposed structure were simulated and its results were the awaited ones. Provided that the elements that constitute the microcontroller have the characteristics of a complex system (CPU, Memory and I/O's ports), the physical implementation of a complete model would lead us to obtain a system on a chip (SOC). The whole description also defines an IP module usable in programmable logic designs.

Keywords: VHDL, Microcontroller, RISC, System On a chip (SOC), Programmable logic.

1 INTRODUCCIÓN

En la actualidad, cuando se desarrolla un dispositivo comercial se busca la mejora en la calidad de los diseños obtenidos y el incremento de la productividad. Asimismo, la necesidad de integrar sistemas cada vez más complejos en dimensiones más reducidas y con menores tiempos de desarrollo ha llevado a tener que desarrollar técnicas y metodologías de trabajo que permitan obtener una mayor productividad. Este objetivo ha dado lugar a un desarrollo cada vez más orientado al diseño de módulos que puedan ser reutilizados, con el fin de alcanzar nuevos modos, no para diseñar más rápido sino para diseñar menos.

El constante avance tecnológico y las características del desarrollo motivaron la aparición en el mercado de sistemas en un chip (System On a Chip - SOC), que como su nombre indica son soluciones que integren en un único chip todo un sistema. Las nuevas metodologías de diseño basadas en SOC se fundamentan en la existencia de librerías con bloques pre-diseñados y pre-verificados que poseen propiedad intelectual y se denominan bloques de "Intellectual Property" (IP). La reusabilidad de los bloques IP es uno de los tópicos que facilita la creación de un nuevo SOC [1][2][3][4].

Los dispositivos de Lógica Programable, que en la actualidad superan el millón de compuertas, son una excelente opción para la implementación de soluciones integradas System on Programmable Chip (SOPC) debido a la flexibilidad de la personalización de los mismos luego de la fabricación. Es importante entender que cualquier diseño SOPC involucra un conjunto complejo de decisiones de diseño hardware-software, teniendo en cuenta el costo del producto, la flexibilidad del mismo y las restricciones que puedan existir dependiendo del escenario donde se desarrollare el proyecto [5].

Una de las decisiones es la selección del procesador del sistema. Algunas compañías ofrecen los mismos como soluciones 'soft-core' con descripciones de procesadores propietarios (NIOS [6]) o estándar (8051 y otros [7]). También ofrecen soluciones 'hard-core' para el caso de aplicaciones intensivas, con procesadores de alta prestación pregrabados (como MIPS32_4Kc o ARM922 [8]).

Otra decisión posible es la utilización de módulos IP de biblioteca que cumplan y provean determinadas funcionalidades útiles para el proyecto en diseño. En este artículo se realiza una descripción en VHDL de un procesador con arquitectura RISC para ser implantado en un dispositivo de lógica programable y que pueda ser utilizado como elemento de biblioteca IP.

2 DISEÑO DE UN BLOQUE IP

Como se mencionó en la sección anterior la existencia de bibliotecas con módulos IP son el mecanismo por el cual las soluciones SOC/SOPC evolucionan constantemente y la información y acceso a las distintas soluciones IP de mercado puede obtenerse en forma rápida [9]. También se pueden mencionar diseños y propuestas realizadas de módulos IP con objetivos académicos [10] [11] a los cuales nos propusimos complementar mediante el diseño de un elemento procesador con características de arquitectura y programación diferentes a los mas difundidos. Nuestro propósito fué describir un sistema con arquitectura Harvard con un procesador de conjunto de instrucciones reducido (RISC) [15].

2.1 Características Harvard y RISC

Una arquitectura tipo Harvard, posee la memoria de programa separada de la memoria de datos, es

decir, el procesador (CPU) ejecuta las instrucciones que se encuentran en su memoria de programa, y por otra vía separada accede a los datos necesarios para su ejecución (Figura 1). Un procesador RISC posee un conjunto de instrucciones acotado (reducido), con acciones muy específicas y la unidad de control podría ser más sencilla, y ocuparía mucho menos espacio físico (superficie del circuito integrado) que un procesador de conjunto complejo de instrucciones (CISC).

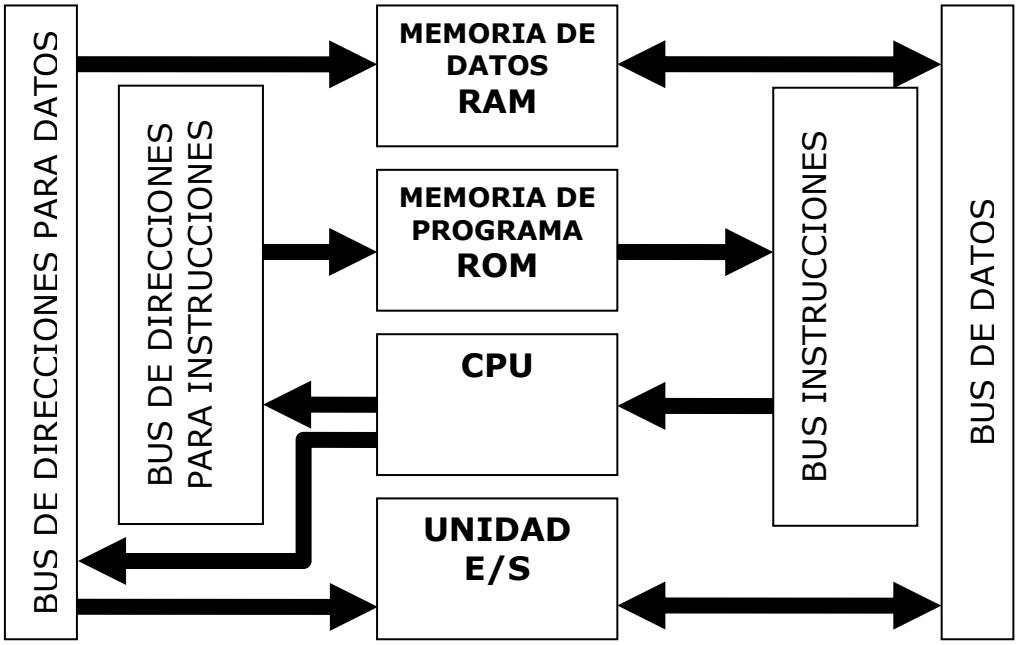

Figura 1: Arquitectura simplificada de un procesador de tipo Harvard.

Por lo tanto, implementar un sistema de este tipo e integrarlo como un bloque IP dentro de un dispositivo de lógica programable ampliaría la oferta de soluciones obtenibles en SOC.

2.2 Microcontrolador

Un microcontrolador, como su nombre indica, está destinado a controlar un sistema determinado, o una parte del mismo. Tiene un único objetivo, ejecutar el programa en su memoria y controlar así el sistema donde esté incluido. Dadas estas características, su capacidad de procesamiento es limitada (poca variedad en el conjunto de instrucciones), necesita memoria de programa (ROM), elementos de almacenamiento temporal (registros y/o RAM) y requiere variada interconexión con el medio que lo rodea (distintos puertos de entrada/salida) e inclusive poseen temporizadores y conversores A/D o D/A todos en un mismo chip. En versiones complejas un microcontrolador puede ser visto como un sistema en si mismo. En nuestra búsqueda de un elemento tipo nos centramos en los difundidos microcontroladores de la empresa Microchip [16].

2.3 Descripción del Bloque IP

El uso de un lenguaje para la descripción de hardware como el VHDL [12][13] nos permite diseñar, modelar, y comprobar sistemas de variado objetivo como [14], [11] o [10] desde un alto nivel de abstracción bajando hasta el nivel de definición estructural de puertas utilizando metodología Top-Down. Es decir, describir (por modelado) el comportamiento de bloques de alto nivel, analizarlos (por simulación), y refinar la funcionalidad de alto nivel requerida antes de llegar a niveles más bajos de abstracción de la implementación del diseño y su síntesis en lógica programable.

3 ARQUITECTURA DE LA FAMILIA PIC17C4X

Los microcontroladores de la familia PIC17X son de alta gama con palabras de 8 bits e instrucciones de 16 bits, arquitectura RISC (58 instrucciones simples) y modelo Harvard. Utilizan segmentación (“pipeline”) en la ejecución de las instrucciones (que son ortogonales), poseen una pila con profundidad de 16 niveles, direccionamiento directo, indirecto e inmediato para datos e instrucciones. Pueden tener hasta 454 registros de propósito general (memoria de datos RAM) y direccionan hasta 8Kx16 de memoria de programa (EPROM/ROM). También, poseen multiplicador 8x8 en hardware y controlador de interrupciones.

La arquitectura de los PIC17X se muestra la Figura 2. En ella se observa la memoria de datos (DATA RAM) que engloba los registros especiales (SFR) y los de propósito general (GPR) y la memoria de programa (PROGRAM MEMORY). La ALU de 8 bits permite realizar operaciones aritméticas y lógicas entre el Registro de Trabajo (WREG) y otro registro. También, se observan los numerosos periféricos que posee esta familia como los Timer y la puerta serie.

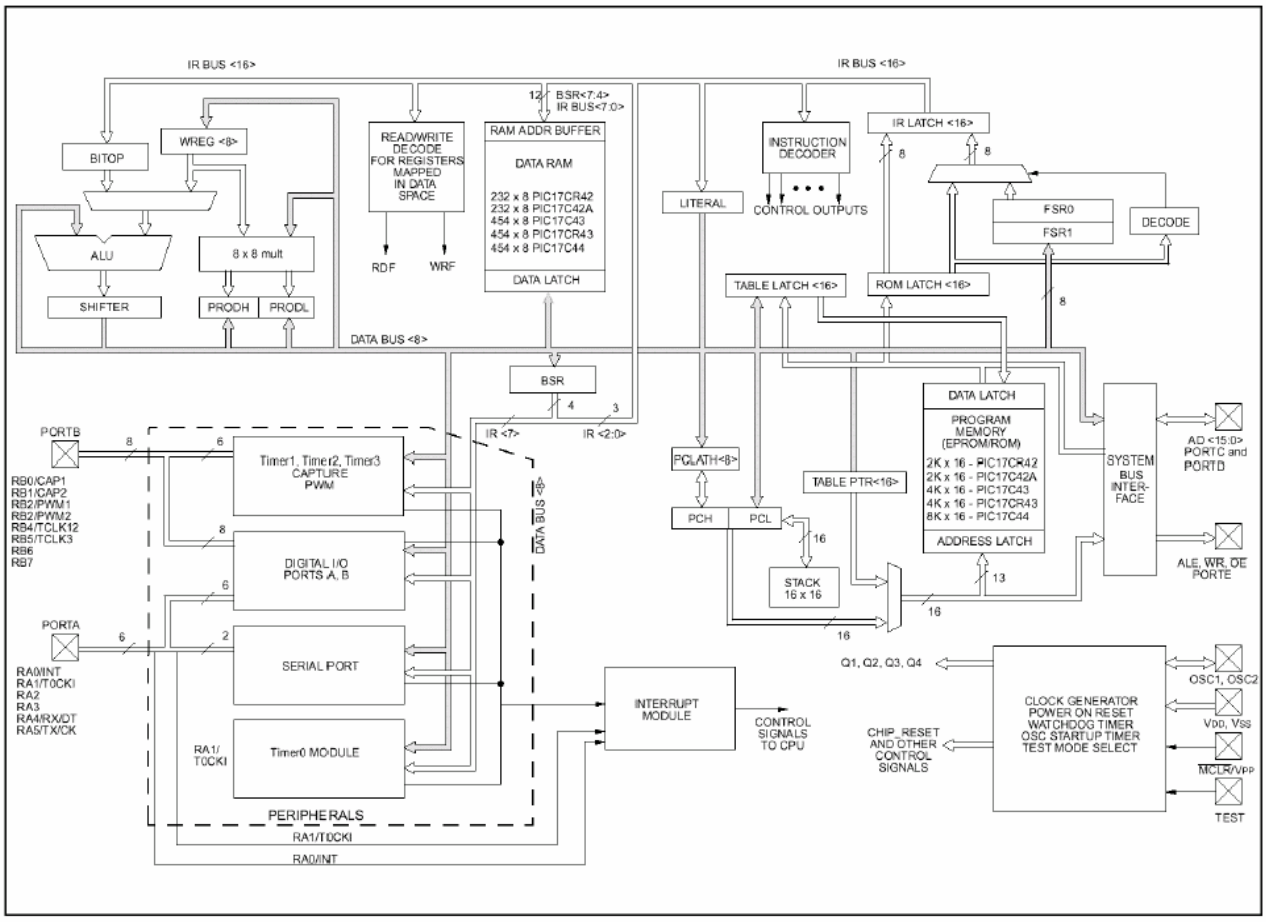

Figura 2: Arquitectura de un microcontrolador PIC17C4X

4 DESCRIPCIÓN EN VHDL DEL PIC17C4X

Como vemos en la sección anterior, tomamos como base la hoja de datos del microcontrolador provista por el fabricante y con ella desarrollamos la arquitectura de componentes a utilizar para describir en VHDL el microcontrolador.

Algunos componentes, no han sido descriptos como el multiplicador por hardware, el Perro Guardián (Watch Dog), los Timers adicionales que poseen este tipo de microcontroladores y la pila. El esqueleto del diseño se basa en lo que denominamos BUS A, BUS B y BUS C. Todas las instrucciones se ejecutan en un ciclo de CLK, excepto las de salto, que duran dos. La Figura 3 presenta un esquema de los componentes de la implementación en VHDL del microcontrolador PIC17C42.

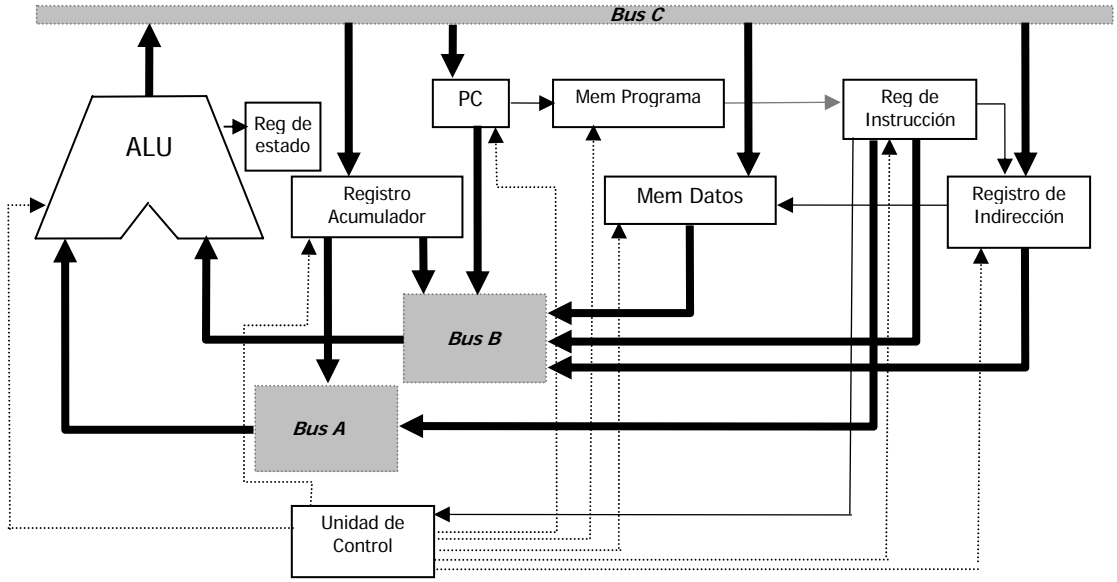

Figura 3: Arquitectura del microcontrolador descripto en VHDL

A continuación describimos brevemente algunos de los bloques.

4.1 Unidad aritmética lógica (ALU)

La ALU, es un sistema combinacional e independiente del reloj. La Figura 4 muestra un esquema de las señales que ingresan y egresan del mismo.

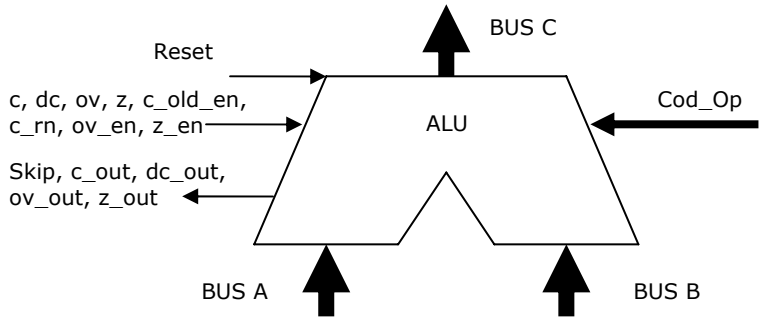

Figura 4: Unidad Aritmético Lógica (ALU)

La unidad implementada puede realizar operaciones aritméticas de suma y resta, lógicas tales como not, and, or y xor, de desplazamiento como rotación a izquierda, a derecha y SWAP, operaciones orientadas a bit.y de transferencia. La ALU descripta tiene asociada un registro de estado que guarda la configuración de la ALU. Es mismo es un registro de 8 bits en los que los cuatro bits más significativos están dedicados a definir el modo de direccionamiento indirecto y en los restantes 4

se encuentran las tradicionales banderas: C (Carry), Z (Zero), DC (Digit Carry) y OV (Overflow).

4.2 Unidad de Control (UC)

Es el bloque encargado de decodificar las instrucciones y generar una secuencia de ordenes (activación de señales) para ejecutarlas. En la Figura 5 se puede observar que recibe una señal CodOP[15:0] proveniente del registro de instrucción, en función de esta señal de entrada, generará las micro-instrucciones necesarias para que se complete la instrucción. Es un bloque combinatorio.

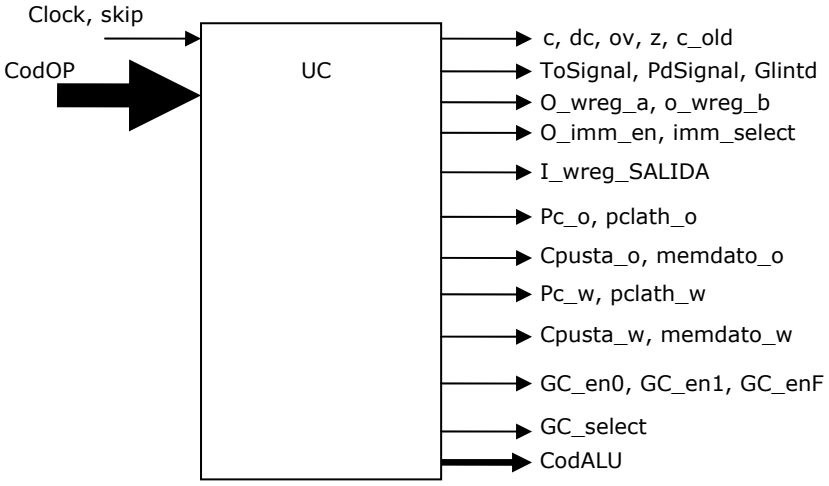

Figura 5: Unidad de Control

La estructura de este bloque en VHDL la dividimos en dos partes bien diferenciadas, que actúan en paralelo. La primera parte determina cual es el registro involucrado en la instrucción, tanto en la entrada como en la salida y genera las señales que corresponda en cada caso. La segunda parte del bloque describe el decodificador de instrucciones y su código VHDL tiene una estructura de tipo IF-THEN-ELSIF para verificar que instrucción debe decodificarse:

SI (Instrucción = ADD)
    ENTONCES *Activación de las señales que ejecuten ADD*
SINO-SI (Instrucción = SUB)
    ENTONCES *Activación de las señales que ejecuten SUB*
    ….

4.3 Contador de programa (RegPC)

En la Figura 6 se observan las señales de la unidad descripta cuya función es sumar ‘1’ al contenido del registro cada vez que se ejecuta una instrucción. La salida es tomada por el bloque de la memoria de programa (ROM) para ubicar la instrucción que debe enviarse a la unidad de control.

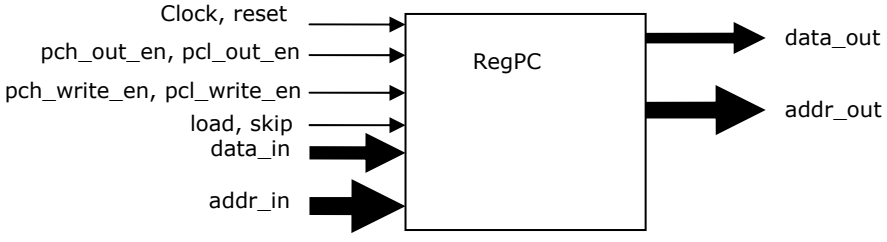

Figura 6: Registro PC

4.4 Registro de trabajo (RegAcum)

La Figura 7 muestra que a este registro de 8 bits ingresarán sólo 3 señales de control. Dos para habilitar las salidas a BUS A y BUS B. La tercer señal habilita la escritura del registro desde BUS C

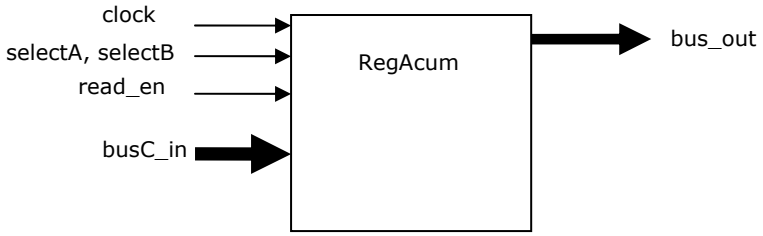

Figura 7: Registro Acumulador

El cuerpo principal de la descripción en VHDL del registro podría ser:

```
Process (clock, read_en, busC_in)
  begin
    if clock'event and clock='1' then
      t <='0';
    end if;
    if  read_en='1' and t='0' then
      W_Reg <= busC_in;
      t <= '1';
    end if;
  end process;
 Bus_out <= W_Reg when selectA='1' or selectB='1';
```

4.6 Memoria de datos (RAM)

Este bloque consiste de registros de propósito general que definen la memoria de datos. Recibe señales de control que le indican cuando realizar una acción de escritura o de lectura y posee los canales de comunicación que permiten seleccionar el registro dentro del bloque con el cual se debe operar. La Figura 8 muestra las señales que ingresan y egresan del bloque que denominamos RAM.

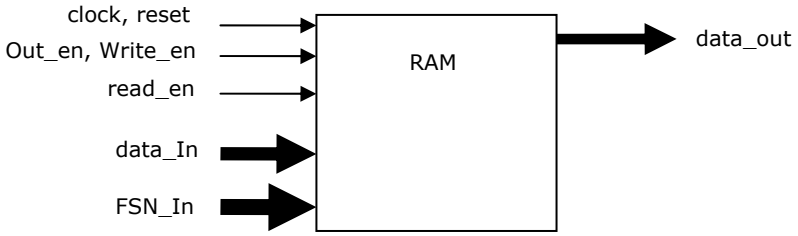

Figura 8: Memoria de datos RAM

4.7 Memoria de programa (ROM)

Este bloque es una memoria ROM convencional, donde debe residir el programa a ejecutar. Recibe la dirección de la celda buscada (generada por el registro PC con pc_in[15:0]) y devuelve el binario almacenado en esa celda (código de máquina de una instrucción en inst_out[15:0]) que se transferirá al registro de instrucción.

Los contenidos de las celdas de ROM son los valores binarios obtenidos por herramientas de

ensamblado ó compilación que realizan traducción a código de instrucciones de máquina del microcontrolador de instrucciones en lenguaje assembly ó sentencias de un lenguaje de alto nivel. Para realizar esta tarea de una forma amigable al programador se desarrolló una herramienta que permite generar la descripción del bloque en VHDL del componente de memoria de programa del microcontrolador de una forma rápida y sencilla; garantizando que la traducción de las instrucciones escritas en assembly del microcontrolador a su respectivo código de máquina se realiza en forma correcta. De este modo el programador debe centrarse únicamente en la lógica del algoritmo que se cargará en la ROM y desentenderse del problema de la traducción de cada una de las instrucciones.

Un ejemplo de la descripción de este módulo puede ser como la siguiente:

```
ARCHITECTURE Rtl OF Rom IS
 SIGNAL int: integer;
BEGIN
 int <= conv_integer(pc_in);
 inst_out <=
            "0000000000000000" when int = 0 else
            "1011000001100000" when int = 1 else
            "0000000100010010" when int = 2 else
            "0000011100010010" when int = 3 else
            "0000011100001010" when int = 4 else
            "0000000000000000" when int = 5 else
            "0000000000000000" when int = 6 else
            "0001010100010010" when int = 7 else
            "0001010100001010" when int = 8 else
            "0000011100010010" when int = 9 else
            "0000011100001010" when int = 10 else
            "0001010100010010" when int = 11 else
            "0001010100001010" when int = 12 else
            "0000000000000000" when int = 13;
end Rtl;
```

5 PRUEBAS REALIZADAS

Se diseñaron diferentes tipos de pruebas que permitieron verificar el funcionamiento de cada uno de los componentes desarrollados (pruebas de unidad) como así también la sincronización y comunicación que debe existir entre todos los componentes de la arquitectura del microcontrolador (pruebas de integración).

En cada una de las pruebas, se utilizaron diferentes secuencias de instrucciones a ejecutar y verificar del microcontrolador, las mismas se cargaron en el componente ‘memoria de programa (ROM)’, luego de actualizado el modelo VHDL se realizaron las correspondientes ejecuciones en el simulador. Para el desarrollo y simulación del modelo VHDL se utilizó el software Max +Plus II de ALTERA.

A continuación se muestran dos de los resultados obtenidos luego de las simulaciones efectuadas con las diferentes configuraciones del microcontrolador. En cada caso se efectuó un análisis en base al resultado obtenido y el resultado esperado, como así también de los errores detectados.

5.1 Ejemplo de prueba 1: operación suma sobre registro de trabajo

En este caso, se prueba el funcionamiento de la Unidad Aritmético Lógica, registro de estado de la Alu, registro de trabajo y generador de constantes. La Figura 9 muestra los resultados obtenidos en

una simulación. Las instrucciones que se ejecutan, realizan sucesivas sumas sobre el registro de trabajo (WREG), obteniendo el dato desde el propio registro y reutilizando el valor obtenido en la siguiente operación. El lenguaje de ensamblado correspondiente fue el siguiente:

```
nop
movlw  50
addwf  wreg, 0
addwfc wreg, 0
addwfc wreg, 0
addwfc wreg, 0
clrf   wreg, 0
nop
```

La sucesión de valores de salida registrados son correctos, observando que se generan las indicaciones de acarreo correspondientes.

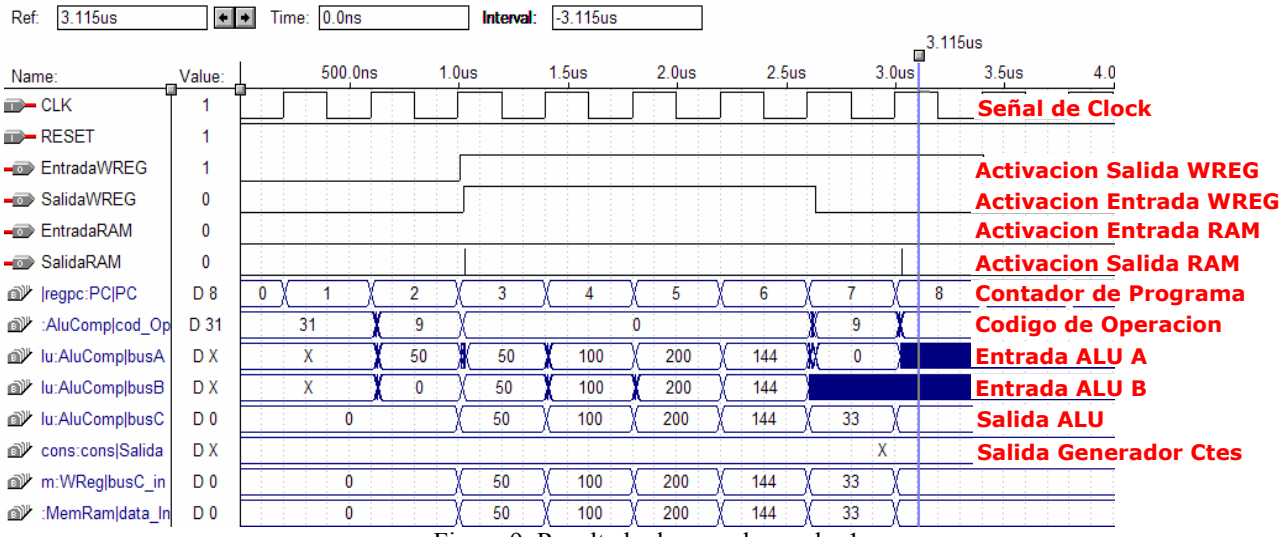

Figura 9. Resultado de caso de prueba 1

5.2 Ejemplo de prueba 2: operaciones lógicas

En este caso de prueba se intentó demostrar el funcionamiento de la memoria de datos y el registro de acumulador trabajando conjuntamente con operaciones lógicas. Para ello se ejecutarán instrucciones que permitirán cargar los registros con diferentes valores y luego realizar operaciones lógicas que requieran de ambos registros. El lenguaje de ensamblado correspondiente fue el siguiente:

```
nop
movlw 98
movwf 18
movlw 97
andwf 18,0
andwf 18,1
iorwf 18,0
iorwf 18,1
xorwf 18,0
xorwf 18,1
nop
```

Si observamos la Figura 10, que muestra los resultados obtenidos durante la simulación del caso de prueba, se puede verificar que los valores obtenidos hacia el final de la ejecución no son correctos (marcado con 1) debido a una de-sincronización de las señales de control de los diferentes componentes. La falta de sincronización se produce en el valor de entrada A de la ALU, por lo que se hicieron ajustes en el componente que define el valor de entrada a la misma para corregir el error..

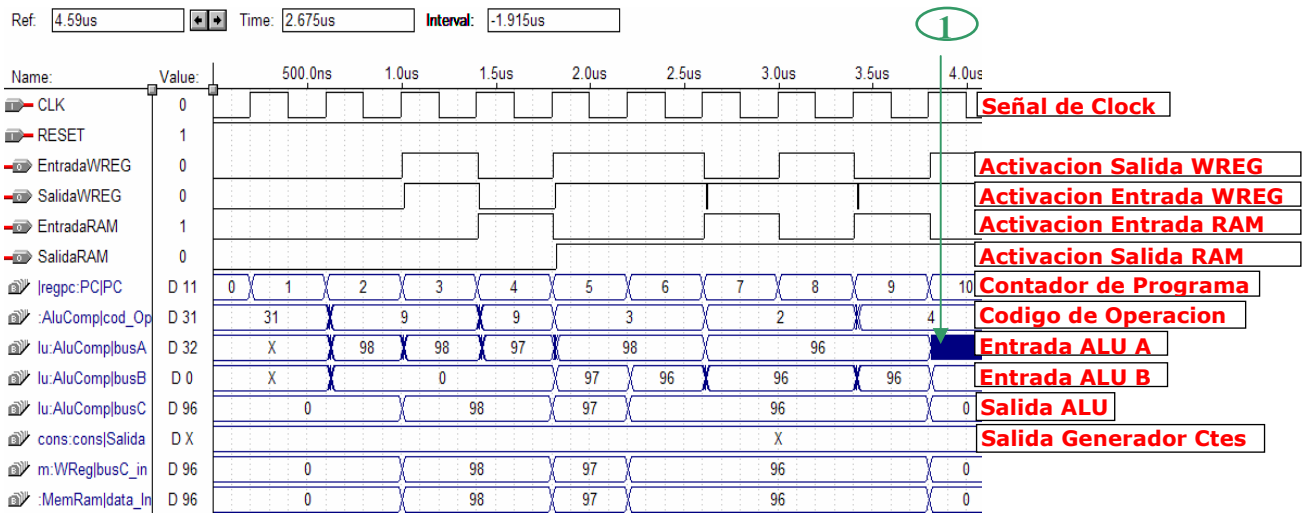

Figura 10. Resultado de caso de prueba 2

6 CONCLUSIONES

El trabajo presentado resume una descripción en lenguaje VHDL de un microcontrolador que pueda ser utilizado como bloque IP de una biblioteca. Se tomo como base un dispositivo comercial que cumpliera características distintivas de otros elementos de biblioteca. Se propuso un conjunto de componentes para formar la estructura del microcontrolador, se describió y verificó el funcionamiento de cada uno de ellos por simulación.

Asimismo, dado que al momento de ejecutar cada simulación, la carga de las instrucciones en la memoria de programa ROM era proclive a errores (y requerían un tiempo considerable) se desarrolló una herramienta que permite, de una forma rápida, la generación automática del componente ROM a partir del código en lenguaje assembly de las instrucciones a ejecutar.

También se realizó la síntesis del modelo VHDL mediante la herramienta Max Plus II provista en el University Program Design Laboratory Package de Altera, obteniéndose resultados dispares, tal como se pudo comprobar en algunos casos de prueba. En estos casos, donde se encontraron diferencias se hicieron ajustes al modelo de componentes para llegar a los resultados esperados.

Los resultados obtenidos fueron satisfactorios y concuerdan con los resultados esperados en cuanto a poder realizar una descripción en VHDL de un microcontrolador existente en el mercado. Sin embargo, sería conveniente profundizar y desarrollar en trabajos futuros la descripción de un multiplicador por hardware, dado que algunos de los controladores pertenecientes a la familia lo poseen, y la descripción de los puertos de entrada/salida a periféricos permitiendo que el modelo pueda interactuar con el mundo exterior o con otros dispositivos.

REFERENCIAS

[1]. Keating M., Bricaud P., "Reuse Methodology Manual For System-On-A-Chip Designs, Second Edition", Kluwer Academic Publishers 1999, USA, ISBN 0-7923-8558-6.

[2]. Meerwein M. et al, "Linking Codesign and Reuse in Embedded Systems Design", Proc. Of the 8th. Intl. Workshop on Hardware/Software Codesign. San Diego, USA, May 2000, pp.93-97.

[3]. Seepold R, Martinez Madrid N. (Editores), "Virtual Components Design and Reuse", Kluwer Academic Publishers 2000, USA, ISBN 0-7923-7261-1.

[4]. Villagarcía H., Bria O., "Diseño de bloques IP: Programabilidad y Reutilización". WICC2001, San Luis, Argentina, May 2001, pp.2-5.

[5]. Wolf W., "Computer as Components: Principles of Embedded Computer Systems Design", Morgan Kaufmann 2000, USA, ISBN 1-55860-541-X.

[6]. ALTERA Corp., "NIOS Soft core Embedded Processor Data Sheet. Version 1". San José, CA, USA, 2000.

[7]. ALTERA Corp., "Intellectual Property Catalog", M-CAT-AIPS-01, San José, CA, USA, 1999.

[8]. ALTERA Corp., "ARM-based Embedded Processor Device Overview. Version 1.1", "MIPS based Embedded Processor Device Overview. Version 1.1". San José, CA, USA, 2000.

[9]. Design & Reuse, sitio de internet: http//: www.design-reuse.com/

[10]. Schweers R., Villagarcia H., Bria O., "Arquitectura CORDIC para calcular seno y coseno", VIII Congreso Argentino de Ciencias de la Computación, Octubre 2002, Ciudad Autónoma de Buenos Aires, Argentina, Publicado en Resúmenes de Comunicaciones, pág. 129.

[11]. Jaquenod G. A., Villagarcia H., Bria O., De Giusti M., "Adapting an IP MC6805 core for multiprocessing and multitasking", IX Workshop IBERCHIP, Marzo 2003, La Habana, Cuba, Publicado en Anales (CDROM), Trabajo 090.

[12]. Villar E. et al., "VHDL Lenguaje estándar de diseño electrónico", McGraw-Hill 1998, ESPAÑA. ISBN 84-481-1196-6.

[13]. Pardo, F. y Boluda J., "VHDL Lenguaje para síntesis y modelado de circuitos", Alfaomega 2000, MEXICO. ISBN 970-15-0443-7.

[14]. Bonadero J., Liberatori M., Bria O., Villagarcia H., "Expansión de la clave en Rijndael: Diseño y optimización en VHDL", XI Workshop IBERCHIP, Marzo 2005, Salvador de Bahía, Brasil. Publicado en Actas, pág. 150 a 153.

[15]. Martinez Belot L. y Leyes A. "Descripción VHDL de una Arquitectura RISC", Tesina de Grado, Facultad de Informática, UNLP. Octubre 2007.

[16]. Microchip Technology Inc, sitio de la empresa en Internet http//: www.microchip.com/