Diseño de un microprocesador de propósito educativo

Design of a microprocessor for educational purposes

Desenho de um microprocesador de objetivo educacional

Para citar este artículo / To reference this article / Para citar este artigo: Camacho Poveda, É. C. y Ruge Ruge, Í. A. (2015). Diseño de un microprocesador de propósito educativo. *Ingenio Magno*, *6*(**2**), **86-99.**

Édgar Camilo Camacho-Poveda
Facultad de Ingeniería, Programa de Ingeniería Electrónica extensión Tunja, Grupo de Investigación en Ingeniería GIRA, Universidad Pedagógica y Tecnológica de Colombia
edgar.camacho@uptc.edu.co

Ílber Adonayt Ruge-Ruge
Facultad de Ingeniería, Programa de Ingeniería Electrónica extensión Tunja, Grupo de Investigación en Ingeniería GIRA, Universidad Pedagógica y Tecnológica de Colombia
ilber.ruge@uptc.edu.co

Fecha de recepción: 16 Agosto 2015
Fecha aprobación: 26 Febrero de 2015

Resumen

Este artículo presenta el proceso de diseño e implementación de un microprocesador para propósitos educacionales, a partir de elementos digitales básicos. Se busca así facilitar la comprensión de la arquitectura de los microprocesadores y su funcionamiento. El proceso se realiza en tres secciones principales: instrucciones de transferencia entre registro, control de flujo (saltos incondicionales) y operaciones lógicas y aritméticas. Para realizar la comprobación del sistema, se adicionan periféricos de conversión analógico a digital (ADC) y digital a analógico (DAC), con el propósito de implementar una aplicación de medición de temperatura a partir de un sensor LM35, en función de obtener su magnitud en un protocolo industrial de comunicación de 4 mA a 20 mA. La implementación del microprocesador mencionado se realiza en el *software* ISE Design Suite 14.2 de XILINX, y se implementa en la tarjeta de desarrollo FPGA Spartan-3.

Palabras clave: diseño digital, lógica de transferencia entre registros, microinstrucciones, microprocesador.

Abstract

This article presents the design process and implementation of a microprocessor for educational purposes based on basic digital components, with the goal of making it easier to understand the architecture and functioning of microprocessors. The process is carried out in three main sections, transfer instructions between registers, control flow (unconditional jumps), and logical and arithmetic operations. In order to perform the system check, analog to digital conversion ADC and digital to analog conversion DAC peripherals were added, with the goal of implementing an application of temperature measurement with an LM35 sensor, to obtain its magnitude in an industrial protocol of communication of 4mA to 20mA. The implementation of said microprocessor is carried out with the software ISE Design Suite 14.2 from XILINX, and is implemented on the Spartan-3 FPGA development board.

Keywords: digital design, logic of transfer between registers, microinstructions, microprocessor.

Resumo

Este artigo apresenta o processo de concepção e implementação de um microprocesador para fins educativos a partir de elementos digitais básicos, com o objetivo de facilitar a compreensão da arquitetura dos microprocessadores e sua operação. O processo é realizado em três seções principais, instruções de transferência entre o logon, o controle de fluxo (incondicional pausas), e operações aritméticas e lógicas. Para executar a verificação do sistema, adicionaram-se periféricos de conversão análogico/digital ADC e digital/analógico DAC, com o propósito de implementar uma aplicação para medição de temperatura a partir de um sensor LM35, para obter sua magnitude em um protocolo industrial de comunicação 4mA a 20mA. A execução do microprocesador mencionado se realiza no software ISE Design Suite 14.2 de XILINX, e implementada no cartão de desenvolvimento FPGA Spartan-3.

Palavras Chaves: desenho digital, lógica de transferência entre registros, microinstruções, microprocesador.

1. Introducción

En la actualidad, los avances tecnológicos han alcanzado gran parte de los campos humanos, desde los cotidianos hasta los industriales, médicos, etc. Hace años, dichos avances han venido acompañados de la evolución de los sistemas digitales, los cuales hoy se encuentran en la mayoría de dispositivos electrónicos. Uno de los desarrollos que más ha marcado dicha evolución es el microprocesador, que ha logrado realizar desde pequeñas tareas en aplicaciones muy básicas, hasta grandes cálculos matemáticos y estadísticos en grandes computadores industriales. Por estas razones, la enseñanza de la arquitectura básica de los microprocesadores y su funcionamiento es esencial en el proceso de aprendizaje de la ingeniería electrónica y las disciplinas afines. Por ello, en este artículo se hace una propuesta para facilitar la comprensión de esta temática.

Un microprocesador es un sistema digital de tipo secuencial sincrónico, dominado por una única señal de reloj (Brey, 2006). Su función básica es ejecutar secuencialmente un grupo de instrucciones almacenadas en una memoria. Estas instrucciones internamente ejecutan una serie de transferencias entre registros, a las que se les llama *microinstrucciones*, que a su vez definen la función de cada instrucción (Mano y Kime, 2008).

La ejecución de dichas microinstrucciones están regidas por el reloj principal del sistema, y la sección encargada de generar las señales necesarias para la correcta ejecución de las microinstrucciones se le llama *l*ógica de *control.*

Los microprocesadores se conectan al mundo exterior a través de periféricos, los cuales pueden ser de entrada o de salida. En casos de aplicaciones básicas, estos pueden ser puertos unidireccionales o bidireccionales, pero en aplicaciones avanzadas se pueden incorporar periféricos como convertidores análogo a digital (ADC) y digital a analógico (DAC), temporizadores, módulos generadores de señales PWM, interfaces de comunicación, etc. Así se logran crear sistemas microcontrolados.

En los siguientes aparatados se describe el procedimiento de diseño, implementación y experimentación de cada una de las tres fases de desarrollo para el microprocesador propuesto.

2. Principios de diseño de microprocesadores

A. Lógica de transferencia entre registros

Esta lógica es una notación utilizada para facilitar la comprensión y documentación del flujo de datos en un sistema digital secuencial, cuando tal sistema es muy complejo para expresarlo en términos de una tabla de estados (Mano, 1982).

Las ecuaciones [1], [2] y [3] muestran ejemplos de la notación utilizada: la ecuación [1] representa una transferencia del registro A al registro B; la ecuación [2], la transferencia al registro B del valor de la memoria M en la dirección A; la ecuación [3], la transferencia al registro A de la suma algebraica del contenido de los registros B y C.

$$B \leftarrow A \quad [1]$$

$$B \leftarrow M[A] \quad [2]$$

$$A \leftarrow B + C \quad [3]$$

B. Microinstrucción

Una microinstrucción se define como una operación elemental que el sistema digital puede realizar en un único pulso de reloj (Hennessy y Patersson, 2007). La notación utilizada es la mostrada en la ecuación [4], donde una expresión de transferencia entre registros es precedida por una expresión booleana llamada *preposición condicional de control*, que especifica bajo qué condiciones se realiza la transferencia (Mano, 1982).

$x_0't_0: B \leftarrow A$ [4]

B. Elementos básicos de un microprocesador

En la figura 1 se muestran los elementos básicos que debe contener un microprocesador.

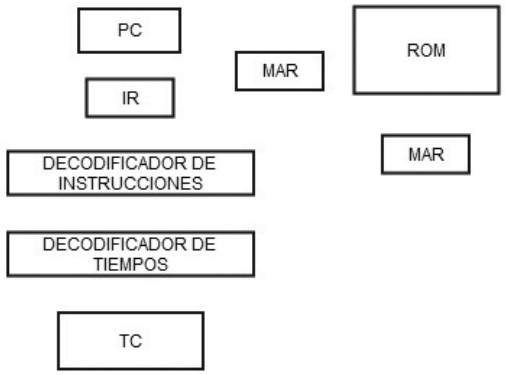

Figura 1. Elementos básicos que conforman un microprocesador

Fuente: Mano (1982).

A continuación se explican brevemente estos elementos.

1. *ROM (Read Only Memory).* Es la memoria donde se almacenan las instrucciones que han de ser ejecutadas por el microprocesador.
2. *MAR (Memory Address Register).* Es el registro encargado de almacenar la dirección de memoria a la cual se va a tener acceso.
3. *MBR (Memory Buffer Register).* Es el registro que almacena temporalmente el dato leído desde la memoria.
4. *PC (Program Counter).* Es un contador incremental de carga paralela que genera la dirección de la instrucción actual de la memoria de programa.
5. *IR (Instruction Register).* Es el registro que almacena la instrucción actual por ejecutar leída desde la memoria de programa.
6. *Decodificador de instrucciones.* Es un circuito combinacional que entrega a su salida un estado único que identifica la instrucción por ejecutar.
7. *TC (Time Counter).* Es un contador incremental que genera la base de tiempo de ejecución de las instrucciones.
8. *Decodificador de tiempos.* Es un circuito combinacional que entrega a su salida un estado único de los instantes de tiempo que utilizarán las instrucciones para ejecutarse.

De igual manera, Patersson y Hennessy (2005) muestran la arquitectura de un microprocesador MIPS; describen el concepto de *Datapath*, necesario para su análisis e implementación, así como el set de instrucciones de la arquitectura y la definición del *opcode*, o formato de código de operación respectivo. Stallings (2006) señala a profundidad las características, las funciones y el diseño del set de instrucciones de la arquitectura de un procesador PowerPC. Por su parte, Paul (1994) explica la decodificación y los formatos de instrucción de la arquitectura del procesador SPARC.

Estos aspectos son analizados a detalle, dado que la definición del set de instrucciones y de su formato de codificación es la característica inicial que debe considerarse para determinar la arquitectura del procesador por diseñar.

3. Procedimiento de diseño

Un procesador de propósito educativo se define como "un procesador que sea lo suficientemente flexible, y cuyas herramientas de diseño e implementación faciliten la realización de pruebas de verificación y desempeño" (Hincapié y Jaramillo, 2011). Debe contener aspectos básicos para que el diseñador comprenda el funcionamiento de cada uno de los elementos lógicos que conforman su arquitectura.

Diversos trabajos se han realizado desde este propósito, tal como el planteado por Varrientos (1991), que diseña un microprocesador basado en un acumulador de simple instrucción y cuya estructura consta de celdas lógicas basadas en el método de Euler, incorporando una señal única de reloj. La máquina TORO 680-16 es subdividida en cuatro bloques funcionales: set de registros, ALU, contador de programa y control lógico.

Pareja y Vera (2014) diseñan un procesador de 8 bits, con arquitectura Von Neumman, para lo cual utiliza una metodología basada en transferencia entre registros RTD, orientada hacia la implementación de algoritmos en *hardware* usando VHDL. Su estructura RTD está compuesta por tres bloques funcionales: unidad de cálculo, registros y lógica de control.

Sedef (2010) diseña un microprocesador de 16 bits usando FPGA, cuya organización de componentes se basa en arquitectura Von Neumann. Los bloques principales de su diseño son: ALU, registros (MBR, MAR y contador de programa) y unidad de control.

Por su parte, Hwang (2004) describe un microprocesador dedicado, cuyos bloques funcionales son: unidad de control basado en máquina de estado finito FSM y unidad Datapath, conformada por una unidad aritmético-lógica y un banco de registros.

A. Arquitectura de microprocesador de propósito educativo

La figura 2 muestra la arquitectura propuesta para el microprocesador por desarrollar, complementado con algunos elementos que permiten la implementación de las instrucciones definidas para la arquitectura: la unidad lógico aritmética (ALU), la pila y la lógica de control. Estas características adicionales surgen del análisis de cada arquitectura revisada en las referencias descritas en el apartado anterior.

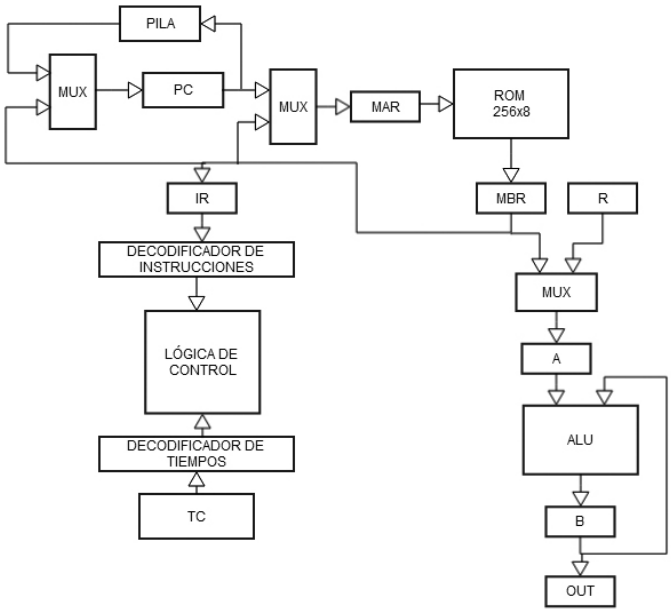

Figura 2. Arquitectura del microprocesador de propósito educativo

El registro R cumple la función de entrada de datos externos (emulación de puerto I/O en modo entrada); el registro A almacena de manera temporal el dato de entrada a la ALU; el registro B funciona como registro acumulador (emulación del registro W en la arquitectura de un microcontrolador de la familia PIC16F de Microchip®), y el registro OUT funciona como salida de datos del microprocesador (emulación de puerto I/O en modo salida).

B. Instrucciones de transferencia entre registros

La tabla 1 describe la codificación realizada a cada una de las instrucciones, el mnemónico respectivo para su uso en código y la microoperación o conjunto de microinstrucciones correspondiente. Un *mnemónico* es un conjunto de caracteres que representa una instrucción para su fácil memorización y representación en código (Palacios y Remiro, 2006).

Tabla 1. Instrucciones de transferencia entre registros

Codificación de operación	Mnemónico	Descripción	Microoperación
00000000_2	RST	Resetea el contador de programa PC	PC←0
00000001_2	MOV	Mueve R a B	B←R
00000010_2	LDI OPRD	Carga OPRD a B	B←OPRD
00000011_2	LDA ADRS	Carga en valor de la memoria en la posición ADRS en B	B←M[ADRS]

Los argumentos que requiere cada instrucción para su ejecución son almacenados en la siguiente posición de memoria. Por ejemplo, si la instrucción LDI está ubicada en la dirección de memoria 0x5A, el argumento que requiere para su ejecución está ubicado en la dirección de memoria 0x5B. La tabla 2 ilustra el caso descrito, que aplica también para las demás instrucciones.

Tabla 2. Ejemplo de ubicación de parámetros de instrucción en la memoria de programa

Dirección	Código	Mnemónico
...		
15_{16}	00000001_2	MOV
...		
$5A_{16}$	00000010_2	LDI AA_{16}
$5B_{16}$	10101010_2	
...		
74_{16}	00000011_2	LDA 55_{16}
	01010101_2	
...		
AC_{16}	00000000_2	RST
...		

Para el diseño de la lógica de control, se calculan las expresiones lógicas booleanas en términos de los estados lógicos dados por el decodificador de tiempos y el decodificador de instrucciones. Estas expresiones lógicas se obtienen mediante el análisis de cada una de las microinstrucciones establecidas para cada instrucción.

El ciclo de envío *(fetch cycle)* necesario para leer la instrucción desde la memoria de programa y cargarla en IR para su decodificación está descrito por las microinstrucciones representadas en las ecuaciones [5], [6] y [7], que son comunes a todo el set de instrucciones. Esta consideración ha de tenerse en cuenta para el diseño de la lógica de control.

$$t_0: MAR \leftarrow PC \quad [5]$$

$$t_1: MBR \leftarrow M[MAR], PC \leftarrow PC + \quad [6]$$

$$t_2: IR \leftarrow MBR \quad [7]$$

Por tanto, las microinstrucciones para cada una de las instrucciones se establecen a partir del tiempo t_3. A modo ilustrativo, las ecuaciones [8] a [12] describen las microinstrucciones correspondientes a la instrucción LDA, cuya codificación corresponde a la instrucción q_3, según lo establecido en la tabla 1.

$$q_3t_3: MAR \leftarrow PC \quad [8]$$

$$q_3t_4: MBR \leftarrow M[MAR], PC \leftarrow PC \quad [9]$$

$$q_3t_5: MAR \leftarrow MBR \quad [10]$$

$$q_3t_6: MBR \leftarrow M[MAR] \quad [11]$$

$$q_3t_7: A \leftarrow MBR, TC \leftarrow 0 \quad [12]$$

La ecuación [12] establece que una vez es realizado el ciclo de ejecución de la instrucción, el contador de programa es restablecido para que la siguiente instrucción en programa se ejecute de la misma manera que la instrucción descrita: ciclo de envío y ejecución de conjunto de microinstrucciones correspondientes.

C. Instrucciones de control de flujo

La tabla 3 muestra las instrucciones de control de flujo de programa que se proponen implementar en el microprocesador.

Tabla 3. Instrucciones de control de flujo

Codificación de operación	Mnemónico	Descripción	Microoperación
00000100_2	GOTO ADRS	Va a una dirección específica de la memoria de programa	PC←ADRS
00000101_2	CALL ADRS	Guarda la dirección actual en la pila y va a una dirección específica de la memoria de programa	PILA←PC PC←ADRS
00000110_2	RETURN	Retorna a la dirección guardada en la pila	PC←PILA

Para las instrucciones de control de flujo de programa, el ciclo de envío de instrucción se ejecuta de la misma forma que las instrucciones de transferencia entre registros. Las ecuaciones [13] a [16] representan las microinstrucciones que conforman la instrucción CALL, cuya codificación corresponde a la instrucción q_5. Esta se toma como ejemplo para ilustrar la forma como el resto de instrucciones se ejecutan.

$$q_5t_3\colon\ MAR \leftarrow PC \quad [13]$$

$$q_5t_4\colon\ MBR \leftarrow M[MAR], PC \leftarrow PC \quad [14]$$

$$q_5t_5\colon\ PILA \leftarrow PC \quad [15]$$

$$q_5t_6\colon\ PC \leftarrow MBR, T \leftarrow 0 \quad [16]$$

D. Instrucciones lógicas y aritméticas

Las instrucciones lógicas y aritméticas propuestas para conformar el set de instrucciones en esta arquitectura se describen en la tabla 4.

Tabla 4. Instrucciones lógicas y aritméticas

Codificación de operación	Mnemónico	Descripción	Microoperación
$\mathbf{00000111}_2$	INCB	Incrementa B	B←B + 1
$\mathbf{00001000}_2$	DECB	Decrementa B	B←B - 1
$\mathbf{00001001}_2$	ADDC OPRD	Suma B con OPRD y lo guarda en B	B←OPRD + B
$\mathbf{00001010}_2$	ADDR	Suma B con R y lo guarda en B	B←R + B
$\mathbf{00001011}_2$	SUBC OPRD	Resta B a OPRD y lo guarda en B	B←OPRD - B
$\mathbf{00001100}_2$	SUBR	Resta B a R y lo guarda en B	B←R - B
$\mathbf{00001101}_2$	IORC OPRD	Opera B OR OPRD y lo guarda en B	B←OPRD$\curlyvee$ B
$\mathbf{00001110}_2$	IORR	Opera B OR R y lo guarda en B	B←R $\vee$ B
$\mathbf{00001111}_2$	ANDC OPRD	Opera B AND OPRD y lo guarda en B	B←OPRD $\wedge$ B
$\mathbf{00010000}_2$	ANDR	Opera B AND R y lo guarda en B	B←R $\wedge$B
$\mathbf{00010001}_2$	XORC OPRD	Opera B XOR OPRD y lo guarda en B	B←OPRD $\oplus$ B
$\mathbf{00010010}_2$	XORR	Opera B XOR R y lo guarda en B	B←R $\oplus$ B
$\mathbf{00010011}_2$	NOTB	Complementa B	B←$\overline{B}$
$\mathbf{00010100}_2$	OUTB	Carga el valor de B en OUT	OUT←B

Como ejemplo se denotan las microinstrucciones que conforman la instrucción ADDC, cuya codificación corresponde a la instrucción q_9.

$$q_9t_3\colon\ MAR \leftarrow PC \quad [17]$$

$$q_9t_4\colon\ MBR \leftarrow M[MAR], PC \leftarrow PC \quad [18]$$

$$q_9t_5\colon\ A \leftarrow MBR \quad [19]$$

$$q_9t_6\colon\ B \leftarrow A + F, TC \leftarrow 0 \quad [20]$$

E. Lógica de control

La lógica de control se encarga de generar las señales de control para cada uno de los bloques funcionales descritos en la arquitectura mostrada en la figura 2, a partir de las microinstrucciones descritas e ilustradas en las secciones anteriores. Las ecuaciones booleanas que describen las salidas de la lógica de control se deducen a partir de las preposiciones condicionales de las microinstrucciones.

Una microinstrucción puede ser común a viarias instrucciones; por ejemplo, la microinstrucción MAR←PC forma parte del ciclo de envío y de las instrucciones LDI, CALL y ADDC, entre otras. La expresión booleana x1, correspondiente a la microinstrucción MAR←PC, queda definida como se muestra en la ecuación [20]. Esta función lógica controla la señal de reloj o de carga del registro tipo D sintetizado para el bloque MAR.

La tabla 5 describe la asignación de las señales de salida acorde con cada microinstrucción. Estas señales serán las encargadas de controlar cada uno de los bloques de la arquitectura desarrollada.

Tabla 5. Salidas de la lógica de control y las microinstrucciones que representan

Salida de la lógica de control	Microinstrucción	Salida de la lógica de control	Microinstrucción
x_1	MAR←PC	x_{12}	PC←PILA
x_2	MAR←MBR	x_{13}	B←A
x_3	MBR←M[MAR]	x_{14}	B←B + 1
x_4	IR←MBR	x_{15}	B←B – 1
x_5	A←MBR	x_{16}	B←A + B
x_6	A←R	x_{17}	B←A – B
x_7	PC←PC + 1	x_{18}	B←A ∨ B
x_8	TC←0	x_{19}	B←A ∧ B
x_9	PC←0	x_{20}	B←A ⊕ B
x_{10}	PC←MBR	x_{21}	B←$\overline{B}$
x_{11}	PILA←PC	x_{22}	OUT←B

Se presenta otro ejemplo: la señal de control x_3, que corresponde a la microinstrucción MBR ← M[MAR]. Esta microinstrucción está presente en las instrucciones LDI (q_2), LDA (q_3), GOTO (q_4) y ADDC (q_9), entre otras. La expresión booleana correspondiente queda definida como lo muestra la ecuación [21].

$$x_3 = t_1 + q_2t_4 + q_3t_4 + q_3t_6 + q_4t_4 + q_5t_4 + q_9t_4 + q_{11}t_4 + q_{13}t_4 + q_{15}t_4 + q_{17}t_4$$

[21

Esta función lógica de control es aplicada a la señal de reloj o de carga del registro tipo D sintetizada para el bloque funcional MBR. Para el caso de la señal de control x_6, correspondiente a la microinstrucción A←R, esta se obtiene gracias a la misma consideración descrita anteriormente; la señal de control resultante es aplicada a la línea de selección del multiplexor que realiza tal acción lógica.

De la misma manera se analizan cada una de las 22 salidas de la lógica de control (x_1-x_{22}) para obtener las respectivas expresiones booleanas que conforman la totalidad de la lógica de control.

4. Implementación

La implementación del microprocesador se hizo sobre la tarjeta de desarrollo DE1, basada en una FPGA Cyclone II, de Altera. Para la síntesis de los bloques funcionales del microprocesador se utilizó lenguaje de descripción de *hardware* VHDL, usando descripción tipo estructural (Pardo y Boluda, 1999; Vrenesic y Brown, 2004).

La figura 3 muestra un segmento del resultado del diseño dado por la herramienta RTL Viewer, del *software* de desarrollo Quartus II, correspondiente a los bloques contador de programa (PC), pila (*stack*) y decodificador de tiempos.

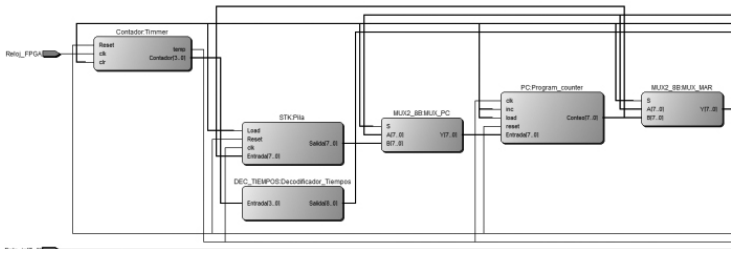

Figura 3. RTL de microprocesador didáctico

Por otro lado, la figura 4 muestra el resumen correspondiente a los recursos lógicos utilizados en la FPGA para la síntesis del procesador didáctico. Se evidencia un uso de menos del 1% de los recursos dispuestos en la Cyclone II, lo cual lo hace muy adecuado para los propósitos educativos deseados.

Flow Summary	
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	Procesador
Top-level Entity Name	Procesador
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	152 / 18,752 (< 1 %)
Total combinational functions	128 / 18,752 (< 1 %)
Dedicated logic registers	75 / 18,752 (< 1 %)
Total registers	75
Total pins	19 / 315 (6 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 4. Resumen de recursos lógicos utilizados en la síntesis del microprocesador didáctico

Para comprobar el funcionamiento de la arquitectura del microprocesador diseñado, se propone generar una aplicación en la que se utilizan instrucciones de transferencia entre registros, control de flujo y operaciones lógicas y aritméticas. Dicha aplicación consiste en medir el valor de la temperatura en un rango de 0 °C a 100 °C, y procesarlo para representarlo en una magnitud de corriente en el rango de 4 mA a 20 mA.

Los componentes utilizados son el sensor LM35 (el cual entrega 10 mV/°C), el ADC ADS7825P (de 16 bits de resolución, usando los 8 más significativos) y el DAC904U (de 8 bits, con salida de corriente de 0 a 20 mA). La frecuencia de muestreo seleccionada es de 1 Hz, con el fin de visualizar fácilmente los resultados.

La ecuación [22] determina la relación entre la temperatura del sensor y la corriente de salida (temperatura en grados Celsius y corriente en mA).

$$i\_out = \left(temp * 0.16 \frac{mA}{^{\circ}C}\right) + 4mA \quad [22]$$

La ecuación [23] es utilizada para procesar la información proveniente del ADC y entregarla al DAC.

$$data\_out = (data\_in * 4) + 51 \quad [23]$$

El conjunto de instrucciones que sintetiza la ecuación [23] y que conforma el archivo ejecutable para la memoria de programa se muestra y describe en la tabla 6.

Tabla 6. Programa de prueba para la aplicación propuesta

Dirección	Código de operación	Mnemónico	Descripción
0x00	0x01	MOVR	B = R*1
0x01	0x0A	ADDR	B = R*2
0x02	0x0A	ADDR	B = R*3
0x03	0x0A	ADDR	B = R*4
0x04	0x09	ADDC d51	B = 4*R+51
0x05	0x33		
0x06	0x14	OUT	OUT = B
0x07	0x04	GOTO 0x00	Vuelve al inicio
0x08	0x00		

5. Resultados

El ejercicio de sintetizar cada uno de los bloques funcionales de la arquitectura del microprocesador didáctico permite al estudiante comprender mejor su estructura y la funcionalidad de cada uno de los bloques. Por otro lado, al implementarse en una FPGA,

los análisis, las pruebas y las simulaciones hacen del diseño una tarea sencilla, por las características propias de un sistema que puede ser reprogramable cuantas veces se requiera, de acuerdo con las especificaciones que se deseen de la arquitectura.

En cuanto a la aplicabilidad del microprocesador didáctico, es general, dado que no se diseñó una unidad de control específica sino genérica para un set de instrucciones establecido. Esto implica que siempre y cuando el algoritmo que se desarrolle esté basado en el uso de este set de instrucciones, lo único que el programador deberá configurar es el conjunto de instrucciones que conforman el programa, que estará almacenado en la memoria de programa. Ello revalida el amplio potencial de aprovechamiento didáctico y pedagógico del microprocesador diseñado.

En cuanto a la aplicación implementada para validar la flexibilidad y el desempeño del microprocesador, al ejecutar el programa propuesto, se supervisa la señal de voltaje entregada por el sensor y la entregada por el conversor analógico a digital DAC0808. La salida de corriente se aplica a una resistencia de 50 Ω para obtener un voltaje entre 0,2 V y 1,0 V, para los valores de corriente de 4 mA a 20 mA, respectivamente. La figura 5 muestra las señales de voltaje de entrada y voltaje de salida obtenidas al aplicar una perturbación partiendo de un estado estable inicial (23,3 °C) hasta uno final (43,2 °C).

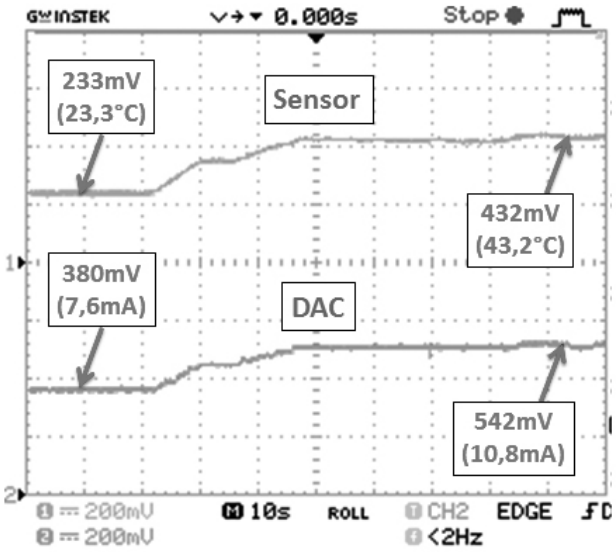

Figura 5. Señales entregadas por el sensor y el DAC

Los valores mostrados en la figura 5 se validan a partir de la evaluación de dichos valores en las ecuaciones [24] y [25], para los puntos de 23,3 °C y 43,2 °C, respectivamente.

$$\left(23.3°C * 0.16\frac{mA}{°C}\right) + 4mA = 7.73mA \quad [23]$$

$$\left(43.2°C * 0.16\frac{mA}{°C}\right) + 4mA = 10.91mA \quad [23]$$

Se observa que los valores obtenidos experimentalmente son muy cercanos a los valores esperados teóricamente, con una desviación promedio de 120 mA en el rango de medición. El ajuste de ganancia se realizó mediante

hardware a través del circuito conversor de corriente a voltaje que requiere el ADC0808.

6. Conclusión

La comprensión de la lógica de transferencia entre registros, las microinstrucciones como manera de descripción de un algoritmo en términos de *hardware* y la metodología para el diseño de la lógica de control son aspectos esenciales para entender la arquitectura de un microprocesador y el funcionamiento de los elementos que lo componen. Con respecto a la generación del código ejecutable grabado en la memoria de programa, se muestra de manera didáctica una aplicación convencional, que representa el resultado que se obtendría al utilizar un compilador convencional.

7. Financiamiento

Los resultados descritos en este artículo se derivan del proyecto de investigación "Diagnóstico tecnológico del uso de dispositivos lógicos programables y microcontroladores en la industria boyacense", financiado por la Dirección de Investigaciones de la Universidad Pedagógica y Tecnológica de Colombia, dentro del marco de la convocatoria DIN017 de 2014, para la vinculación de docentes a grupos de investigación.

Referencias

Brey, B. (2006). *Microprocesadores Intel* (7.ª ed.). Ciudad de México: Pearson.

Hennessy, J. y Patersson, D. (2007). *Computer architecture: A quantitative approach* (4.ª ed.). San Francisco: Elsevier.

Hincapie, J. y Jaramillo, J. (2011). Diseño e implementación de un microprocesador de propósito específico. *Scientia et Technica, 16*(47), 136-140.

Hwang, E. (2004). *Digital logic and microprocessor design with VHDL.* Riverside: Team Electronics.

Mano, M. (1982). *Lógica digital y diseño de computadores.* Madrid: Prentice Hall.

Mano, M. y Kime, C. (2008). *Logic and computer design fundamentals* (4.ª ed.). Upper Saddle River: Pearson Prentice Hall.

Palacios, E. y Remiro, F. (2006). *Microcontrolador PIC16F84A: Desarrollo de proyectos* (2.ª ed.). Ciudad de México: AlfaOmega.

Pardo, F. y Boluda, J. (1999). *Lenguaje para síntesis y modelado de circuitos.* Madrid: RA-MA.

Pareja, A. y Vera, M. (2014). Diseño VHDL de una procesado de ocho bits e implementación en un CPLD. Recuperado de http://www.iberchip.net/IX/Articles/POST-123.pdf

Patersson, D. y Hennessy, J. (2005). *Computer organization and design: The hardware/software interface* (3.ª ed.). San Francisco: Elsevier.

Paul, R. (1994). *SPARC Architecture assembly language programming, and C.* Nueva Jersey: Prentice Hall.

Sedef, H. (2010). Designing of a 16 - bit microprocessor by using FPGA. *National Conference on Electrical, Electronics and Computer Engineering (ELECO)*, 349-354.

Stallings, W. (2006). *Organización y arquitectura de computadores* (7.ª ed.). Madrid: Pearson.

Varrientos, J. (1991). VLSI microprocessor design for classroom instruction. *University/Government/Industry Microelectronics Symposium*, 70-75.

Vrenesic, Z. y Brown, S. (2004). *Fundamentals of digital logic with VHDL design.* Madrid: McGraw-Hill.