6º Jornadas de Desarrollo e Innovación Tecnológica | 2007
Instituto Nacional de Tecnología Industrial
ELECTRÓNICA
Desarrollo e Innovación

Microcontrolador compatible con *PIC* 16C84, descripción de hardware

Tropea, S. E.[(i)]

[(i)]INTI-Electrónica e Informática

Introducción

En este trabajo presentamos el desarrollo de un microcontrolador compatible con el *PIC* 16C84 de Microchip[1]. Los microcontroladores son similares a los microprocesadores o *CPUs* utilizados en computadoras. Las principales diferencias radican en su capacidad, mucho más modesta, su consumo, mucho menor, y el hecho de que sus componentes periféricos se encuentran incluidos en el mismo circuito integrado, por ejemplo: memoria de programas.

El 16C84 es un microcontrolador muy popular y que posee una arquitectura muy simple. Pertenece a la familia de microcontroladores de tipo *RISC*. En nuestro país es ampliamente utilizado para pequeños desarrollos. Por estas razones se lo seleccionó como base para nuestro diseño.

Este desarrollo fue autogenerado y posteriormente transferido para su uso no exclusivo a la empresa INVAP S.E.[2]. Actualmente es utilizado en el WRND, un sistema de medición de flujo de neutrones de amplio rango para seguridad y control de reactores nucleares, calificado para aplicaciones espaciales[3].

El objetivo principal de este desarrollo fue el de disponer de un microcontrolador que pudiéramos utilizar en desarrollos basados en el uso de *FPGAs* (*Field Programmable Gate Arrays* = Arreglo de Compuertas Programable en Campo), el mismo debía poseer una arquitectura familiar para los otros integrantes de nuestro laboratorio.

Metodología

Las *FPGAs* son circuitos integrados reconfigurables y son los miembros más avanzados de la familia de circuitos lógicos programables. Una *FPGA* está compuesta por lógica combinacional, registros y mecanismos de interconexión para unir estos dos últimos. Todos estos recursos son reconfigurables de manera tal que se puede modificar el funcionamiento para que se ajuste a nuestras necesidades. Estos elementos se complementan con celdas de entrada y salida que permiten conectar nuestro circuito con el exterior.

Las *FPGAs* presentan una alternativa muy interesante cuando es necesario desarrollar un circuito integrado a medida. Los circuitos integrados realizados a medida poseen costos muy elevados y sólo se justifican cuando el volumen de producción supera las decenas de miles. Esto es válido aún para los *ASIC* (*Application Specific Integrated Circuit* = Circuito Integrado de Aplicación Específica) que utilizan técnicas de fabricación de las conocidas como *semi-custom* en las que sólo una parte del proceso es específico del cliente y el resto son estándares. Cuando los volúmenes de producción son pequeños las *FPGAs* aparecen como una excelente alternativa. Las mismas no poseen grandes gastos iniciales o del tipo *NRE* (*Non-Recurring Engineering*) como en el caso de los *ASICs*. Por otra parte si se detectara un error en el diseño, o fuera necesario introducir otro tipo de cambio o mejora, las *FPGAs* pueden reconfigurarse sin ser necesario reemplazar el circuito integrado. Como contrapartida el costo por unidad de las *FPGAs* es superior, su velocidad es inferior y el consumo de energía es mayor cuando se las compara con los *ASICs*.

Dentro de una *FPGA* se puede incluir la funcionalidad de varios circuitos integrados. Esta funcionalidad puede ser desarrollada por el mismo equipo de trabajo o adquirida a través de un tercero. Debido a que estas funcionalidades son como componentes electrónicos, pero sin su parte física, se los suele llamar componentes virtuales. En la industria se los conoce como bloques de propiedad intelectual o *IP cores*.

El desarrollo de un *IP core* para una *FPGA* es muy similar al de un *ASIC*. En el caso de la *FPGA* el proceso se encuentra simplificado gracias a que nuestro circuito utilizará recursos ya probados y algo sobredimensionados. Este diseño se realiza utilizando lo que se conoce como lenguajes de descripción de hardware. Los mismos tienen cierto parecido con los lenguajes de programación de computadoras pero poseen diferencias conceptuales muy importantes. No se trata de programar un dispositivo sino de describir el

comportamiento del mismo. Luego la descripción se convierte en una configuración para la *FPGA* utilizando herramientas de síntesis.

En nuestro desarrollo utilizamos el lenguaje de descripción de hardware *VHDL* (*Very high speed integrated circuit Hardware Description Language* = Lenguaje de Descripción de Hardware para Circuitos Integrados de Muy Alta Velocidad). Se trata de un lenguaje muy utilizado por agencias gubernamentales y en el área aeroespacial. El mismo fue originalmente desarrollado por el Departamento de Defensa Norteamericano y hoy se encuentra estandarizado por la IEEE (normas 1076, 1164 y accesorias).

El desarrollo fue verificado utilizando *FPGAs Spartan* II[4] y *Spartan* 3[5] de *Xilinx*. Aún así el mismo fue realizado de manera tal que pudiera ser sintetizado con cualquier *FPGA*. Esto fue de vital importancia a la hora de transferir el desarrollo a INVAP S.E. debido a la necesidad de sintetizar el microcontrolador utilizando *FPGAs* de *Actel*[6] tolerantes a radiación, muy importante debido al hecho de que la aplicación final debe funcionar en el espacio.

Las herramientas de desarrollo utilizadas fueron las recomendadas por el proyecto FPGALibre[7][8][9] impulsado por nuestro laboratorio. Para este desarrollo se utilizaron estaciones de trabajo que corren Debian[10] GNU[11]/Linux.

Para la implementación del microcontrolador se buscó modelar el diagrama en bloques mostrado en las hojas de datos del 16C84[12]. Así mismo se optó por implementar la misma secuencia de operación. De esta manera programas escritos para el 16C84 y que dependieran de detalles de bajo nivel, lazos de demora por ejemplo, podrían correr sin modificaciones en nuestra implementación.

El 16C84 es un microcontrolador tipo *RISC* capaz de ejecutar una instrucción por ciclo de máquina. La única excepción son los saltos que son efectivos luego de dos ciclos. Cada ciclo de máquina corresponde a cuatro ciclos de reloj y se divide en:

—Decodificación.

—Lectura de operandos.

—Ejecución (modificación)

—Escritura del resultado

De esta manera el 16C84 posee instrucciones capaces de realizar lectura, modificación y escritura de resultados en un sólo ciclo de máquina.

La memoria de programa se encuentra separada de la memoria de datos y son de anchos diferentes. La memoria de programas es de 14 bits y la de datos de 8 bits.

Resultados

Se obtuvo una descripción del microcontrolador escrita en *VHDL* estándar útil para ser utilizada con cualquier *FPGA*. Nuestra implementación modela el diagrama en bloques expuesto en la hoja de datos con las siguientes excepciones:

—No se implementaron: la memoria *EEPROM*, el *Power-on Reset*, el *Power-up Timer*, la instrucción *SLEEP* ni la programación serie de la memoria de programa.

—Se mejoraron detalles tales como: el número de entradas/salidas, la memoria de datos y la frecuencia máxima de operación.

—La ALU (*Arithmetic Logic Unit* = Unidad Aritmético-lógica) no se encuentra muy detallada en la hoja de datos por lo que se optó por una solución adecuada para implementar el *set* de instrucciones *(ver Fig. 1)*.

La implementación incluye los siguientes módulos de particular interés:

—*Stack* de 8 saltos, tamaño configurable de 1 a 256.

—Sistema de interrupciones. Opcionalmente configurable para que las interrupciones sean por nivel y no por flanco.

—*Watch Dog*. A diferencia del *PIC* original se implementó a partir del *clock*.

—Temporizador/Contador.

—Entrada externa de interrupciones. Opcionalmente se puede indicar que pin de I/O se usa.

—Interrupción por cambio en un grupo de pines. Opcionalmente se puede indicar un grupo diferente al original.

—Tamaño de la memoria de datos configurable.

—Hasta 64 k *words* de memoria de programa.

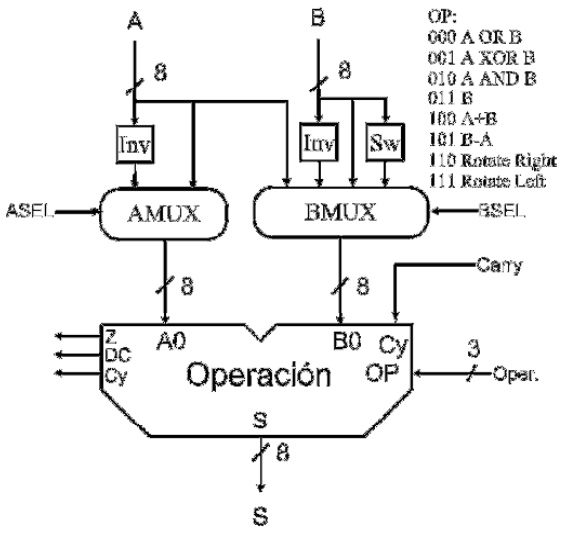

Fig. 1: Esquema simplificado de la ALU implementada.

A los fines de permitir agregar periféricos al microcontrolador se incluyó un bus de expansión WISHBONE[13]. El mismo permite acceder a un máximo de 256 registros de 8 bits. Este bus es el utilizado por el proyecto OpenCores[14] y por lo tanto nos permitió conectar periféricos pertenecientes a dicho proyecto. El agregado de este bus permitió realizar pruebas en las que se hizo funcionar el microcontrolador junto con diferentes periféricos. Los periféricos utilizados en dichas pruebas fueron:

—Maestro I^2C, adaptado a partir de uno disponible en OpenCores.

—Controlador de interrupciones[15], desarrollado por nuestro laboratorio.

—Generador de video en modo texto[16], compatible con monitores VGA, desarrollado por nuestro laboratorio.

—UART, una versión mejorada de la disponible en OpenCores.

El área ocupada depende de que características se encuentren habilitadas y la velocidad depende en gran parte del tipo de *FPGA* utilizada. Para *FPGAs Spartan* II se obtuvo frecuencias de trabajo del orden de 30 MHz y en el caso de *Spartan* 3 fue posible correr el microcontrolador a 50 MHz. Esto último se realizó indicando al programa de síntesis que se deseaba obtener esta velocidad aún a costa de incrementar el área utilizada.

El área mínima que puede ocupar el microcontrolador es de 124 *slices*, esto es sólo el 6,5 % de una pequeña *Spartan* 3 200 y menos del 1 % de una *Spartan* 3 1500. Cuando se sintetizó el microcontrolador conectado a todos los periféricos antes mencionados el área ocupada fue de 622 *slices,* periféricos incluidos, esto es alrededor de una tercera parte de una *Spartan* 3 200 o bien algo más de un 4 % de una *Spartan* 3 1500.

Conclusiones

Se obtuvo el *IP core* de un microcontrolador pequeño y versátil. El mismo permite reusar código, herramientas y conocimiento adquiridos trabajando con microcontroladores para trabajar con tecnologías más avanzadas como las *FPGAs*.

El microcontrolador obtenido es más poderoso que el 16C84 original, en la Tabla I se muestra una comparación de la implementación utilizando una *Spartan* II.

Tabla I. Comparación entre el 16C84 de Microchip y nuestra implementación.

	Original	Implementación (máximos)
Memoria de registros	36 bytes	464 bytes
Pines de I/O	13	34
Frecuencia máxima de reloj	10 MHz	31 MHz
Memoria de programa	1 k	64 k

La versatilidad de las *FPGAs* permitió incorporar periféricos poderosos con un esfuerzo relativamente bajo. Tal es el caso del generador de video.

En cuanto al uso de WISHBONE para el bus de expansión se observó que el área insumida es baja, no implica un esfuerzo importante ni insume una cantidad importante de recursos. Por otro lado las ventajas de utilizar WISHBONE son importantes, las mejoras en la reusabilidad y la posibilidad de utilizar *IP cores* disponibles libremente en internet son apreciables.

La utilización de *VHDL* estándar permitió que la descripción de hardware pudiera ser sintetizada para *FPGAs* de *Actel* aún cuando se desarrolló utilizando *FPGAs* de *Xilinx*.

La utilización de las herramientas propuestas por el proyecto FPGALibre mostró ser adecuada para este desarrollo.

Referencias

[1] Microchip Technology Inc., http://www.microchip.com/
[2] INVAP S. E., http://www.invap.net/
[3] S. E. Tropea, R. M. Cibils, "Contador de pulsos nucleares para uso espacial", 6° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2007.

[4] *Xilinx Spartan* II *FPGA*, http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan2_fpgas/index.htm
[5] *Xilinx Spartan* 3 *FPGA*, http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas/
[6] *Actel* Corporation, http://www.actel.com/
[7] S. E. Tropea, D. J. Brengi, J. P. D. Borgna, "FPGALibre: Herramientas de Software Libre para diseño con *FPGAs*", *FPGA* Based Systems, ISBN 84-609-8998-4, pp 173-180, 2006.
[8] S. E. Tropea, D. J. Brengi, J. P. D. Borgna, S. N. Gwirc, "FPGALibre: Herramientas de Software Libre para diseño con *FPGAs*", 6° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2007.
[9] Proyecto *FPGA* Libre, http://fpgalibre.sourceforge.net/
[10] Debian project, http://www.debian.org/
[11] GNU project, http://www.gnu.org/
[12] Microchip Technology Inc., "8 bits CMOS EEPROM Microcontroller", p 8 (http://ww1.microchip.com/downloads/en/devicedoc/30445c.pdf, verificado 27/06/2007).
[13] Silicore and OpenCores.Org, "WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores", http://prdownloads.sourceforge.net/fpgalibre/wbspec_b3-2.pdf?download
[14] OpenCores.Org, http://www.opencores.org/
[15] S. E. Tropea, "Controlador de interrupciones, descripción de hardware", 6° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2007.
[16] S. E. Tropea, "Controlador de video en modo texto compatible con VGA, descripción de hardware", 6° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2007.

Para mayor información contactarse con:

Ing. Salvador E. Tropea - salvador@inti.gov.ar