Microcontrolador compatible con PIC16C84, bus Wishbone y video

Salvador E. Tropea, Juan P. D. Borgna

Instituto Nacional de Tecnología Industrial, Electrónica e Informática,
Unidad Técnica Instrumentación y Control, Argentina
http://utic.inti.gov.ar.

Abstract. En este trabajo se presenta el desarrollo de un microcontrolador compatible con el PIC16C84 de Microchip. Al mismo se le agregó un bus de expansión compatible con el estándar de interconexión Wishbone. En dicho bus se agregaron periféricos desarrollados por nuestro equipo así como periféricos obtenidos de internet. Los periféricos agregados incluyen funcionalidad comúnmente encontrada en los microcontroladores PIC (UART e I²C) así como periféricos poco frecuentes para esta gama de microcontroladores (video y controlador de interrupciones). El diseño fue verificado usando simuladores y con FPGAs de Xilinx (Spartan II).

1 Introducción

Durante el proceso de búsqueda y selección de herramientas y metodologías para el desarrollo de soluciones basadas en FPGAs utilizando software libre nuestro equipo encaró el diseño de un microcontrolador. Se seleccionó el PIC 16C84 debido a que es un microcontrolador:

- Simple, por lo tanto no representaba un trabajo muy grande.
- Pequeño, por lo que podía sintetizarse en nuestra Spartan II XC2S100 dejando espacio para agregar periféricos.
- Muy popular
- Perteneciente a la familia de 14 bits de instrucciones muy usada por nuestro equipo.

Los primeros resultados referidos a la metodología de prueba fueron reportados en la edición anterior de este congreso[1]. Las herramientas seleccionadas derivaron en el proyecto FPGALibre[2], presentado en un trabajo separado en este congreso[3]. Este trabajo se centra en los detalles referentes a la arquitectura del microcontrolador, el uso de un bus de interconexión estándar, su implementación y las ventajas de dicha estrategia.

2 Microcontrolador

2.1 Arquitectura

Para la implementación del microcontrolador se buscó modelar el diagrama en bloques mostrado en las hojas de datos del 16C84. Así mismo se optó por implementar la misma secuencia de operación. De esta manera software escrito para el 16C84 y que dependiera de

detalles de bajo nivel, lazos de demora por ejemplo, podría 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.

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 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 (Fig. 1 es un esquema simplificado de nuestra ALU).

La implementación incluye los siguientes módulos de particular interés:

- Stack de 8 saltos.
- 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.

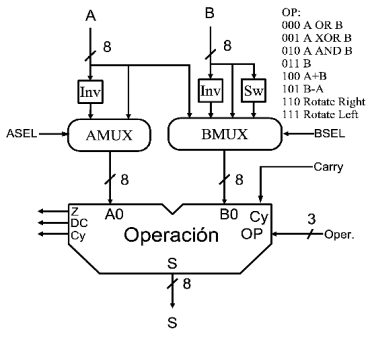

Fig. 1. Diagrama de la UART.

2.2 Resultado de Síntesis

Se utilizó el Xilinx ISE WebPack versiones 6 y 7 para sintetizar el microcontrolador. La FPGA para la que se sintetizó es la Spartan II: XC2S100-5-PQ208. Se notó que para nuestro diseño la versión 7 usa más LUTs que la 6. La diferencia radica en un criterio diferente para utilizar multiplexores en la lógica asociada a sentencias condicionales de VHDL.

La frecuencia de operación estimada por la herramienta fue de algo más de 30 MHz lo cual es tres veces la frecuencia de clock del 16C84. Las pruebas se realizaron con un clock de 25 MHz ya que esta frecuencia es la adecuada para obtener los sincronismos de video.

2.3 Comparación con el Original

La siguiente tabla compara la versión sintetizada del microcontrolador con el original:

	Original	Implementación
Memoria de registros	36	464
Pines de I/O	13	24
Frecuencia máxima de reloj	10 MHz	31 MHz

Tabla 1. Comparación entre el microcontrolador original y la implementación realizada

El incremento en la memoria de registros se debe al uso de uno de los Block RAMs de la FPGA que son de 4096 bits y por lo tanto no hubiera tenido sentido usar sólo 36 de los 512 bytes. No se pueden obtener 512 bytes libres debido a que el espacio de memoria de registros es compartido con los registros especiales donde se mapean los periféricos.

El incremento en el número de pines se hizo de manera tal que nuestro procesador fuera compatible con los procesadores de la misma familia que ofrecen más pines de I/O.

3 Bus de Expansión

En una primer aproximación creamos un bus de expansión específicamente creado para el microcontrolador. Esto tiene importantes desventajas, entre ellas:

- Para conectar periféricos creados por otros grupos de trabajo es necesario adaptarlos a la señalización del bus en cuestión.
- Periféricos diseñados para ese bus no servían para ser usados en otros diseños con buses de otro tamaño. Por ejemplo: no servían para un bus de 16 o 32 bits sin ser adaptados.
- El bus no estaba pensado para usos más avanzados como la posibilidad de que dos maestros compartieran el bus.

3.1 Selección del Bus

Con el objetivo de solucionar los problemas antes mencionados y de poder reusar nuestros diseños así como también usar diseños realizados por otros grupos adoptamos el estándar de interconexión Wishbone[4].

Wishbone fue creado para solucionar estos, y otros, problemas por el proyecto OpenCores[5]. Al utilizar este estándar de interconexión es posible incorporar cualquiera los cores IP de OpenCores en nuestros diseños. Wishbone tiene las siguientes ventajas:

- Para casos simples (un maestro y uno o más esclavos) se reduce a poca o ninguna lógica adicional.
- Fue pensado para casos más complejos (más de un maestro, reintento, notificación de error, etc.).
- No posee *royalties* y puede ser usado sin costo alguno. La especificación completa se encuentra disponible en internet.

3.2 Implementación

El mecanismo usado por los microcontroladores tipo PIC para acceder a los periféricos es mediante el mapeo de sus registros en el área de memoria de datos. El 16C84 reserva las primeras 12 posiciones de memoria para esto. En nuestra implementación dejamos de lado la memoria EEPROM proveída por el microcontrolador original y por lo tanto los registros 8 y 9 quedaron libres. Utilizamos estos dos registros para el bus Wishbone. El registro 8 se usa para indicar la dirección del periférico que deseamos utilizar en el bus Wishbone. Luego de esto cualquier operación sobre el registro 9 se transfiere a través del bus Wishbone. Esto permite acceder a hasta 256 registros de 8 bits en el bus Wishbone.

Como se explicó antes la arquitectura del 16C84 permite realizar operaciones del tipo leer/modificar/escribir en un sólo ciclo de máquina. En nuestro caso es posible realizar estas operaciones sobre cualquier registro en el bus Wishbone en un sólo ciclo de máquina.

El bus Wishbone contempla la conexión de periféricos lentos por lo que si un periférico necesita más de un ciclo de reloj para realizar la operación puede detener al microcontrolador hasta que la misma haya concluido.

4 Periféricos

Además de los que naturalmente incluye el 16C84 (puertos de I/O, interrupciones externas, interrupciones por cambio de estado y *timer/counter*) y con el objetivo de incrementar la funcionalidad del microcontrolador se agregaron al mismo un controlador de video, I²C, una UART y un controlador de interrupciones.

Todos estos periféricos se conectan al bus de expansión basado en Wishbone. Esta experiencia permitió evaluar la complejidad de realizar periféricos que cumplieran con el estándar Wishbone así como evaluar la factibilidad de utilizar *IP cores* proveídos por OpenCores.

4.1 Controlador de Video

Este periférico ya había sido desarrollado en etapas anteriores pero utilizaba el antiguo bus de expansión. La adaptación al bus Wishbone resultó extremadamente simple.

El controlador es capaz de generar una señal compatible con el estándar VGA cuando se utiliza un clock de 25 MHz. La señal obtenida tiene un sincronismo horizontal de 31.25 kHz y vertical de 69.44 Hz siendo equivalente a los sincronismos de un modo VGA de 640x400.

El modo generado es equivalente a un modo de texto con 40 columnas y 25 líneas. Debido a que se buscaba sintetizarlo en una Spartan II se acotaron sus prestaciones de manera tal que pudieran usarse los Block RAMs del dispositivo, por estas razones se redujo la resolución a 320x200 (doble *scan* y doble *pixel*) y se limitó a 40x25 caracteres (1000 caracteres). Así mismo se limitó el número de caracteres diferentes a 64 que con una resolución de 8x8 *píxeles* permite obtener un mapa de caracteres de 4096 bits que es la capacidad de un BRAM de Spartan II.

Por iguales razones y para simplificar la conexión al monitor se limitó el espacio de colores a 8 colores de primer plano y 8 de fondo. De esta manera un caracter en la pantalla es descripto por 3+3+6=12 bits con lo que la pantalla completa son 12*1000=12000 que son aproximadamente tres BRAMs.

La conexión de la FPGA a las entradas RGB del monitor se hizo intercalando una resistencia de 390 ohms que en conjunto con la impedancia de entrada del monitor permiten que la salida de la FPGA no sature estas señales.

4.2 I²C

Se adaptó un *core* de OpenCores. Los cambios que se hicieron fueron pequeños y sólo orientados a optimizar sus prestaciones ya que funcionaba perfectamente.

El *core* se probó con una memoria EEPROM 24LC02B a una frecuencia de 367,25 kHz, limitada por la memoria y no por la FPGA.

4.3 UART

Se adaptó un *core* de OpenCores. Se le agregó un controlador de *baudrate* (originalmente el *baudrate* era fijo y se configuraba usando una constante) que puede ser controlado a través de Wishbone permitiendo que la velocidad de la comunicación pueda ser modificada en pleno funcionamiento.

4.4 Controlador de Interrupciones

Tanto el periférico I²C como la UART poseen interrupciones. La arquitectura original contempla una única entrada de interrupciones externas y las interrupciones correspondientes a cada periférico que incorporan los distintos modelos se mapean en registros especiales. Este mecanismo es poco flexible y por lo tanto se decidió implementar un controlador de interrupciones muy simple que permitiera:

- Administrar tantas fuentes de interrupción como fuera necesario.
- Poder enmascarar cualquiera de ellas.
- Poseer un mecanismo para darles su correspondiente *acknowledge*.

Se desarrolló un periférico compatible con Wishbone. El mismo es parametrizable pudiéndose elegir entre una y ocho fuentes de interrupción por periférico. De ser necesario manejar más fuentes de interrupción basta con usar más de uno de estos periféricos.

5 Resultados de la Síntesis

El uso de recursos depende de cuan grande sea el programa que correrá el microcontrolador ya que en caso de no usarse todo el espacio de programa el sintetizador elimina lógica asociada a los bits sobrantes del contador de programa. Los siguientes datos se tomaron para el caso de un programa de hasta 512 *words*.

	ISE WebPack 6	ISE WebPack 7
Flip Flops	500	500
LUTs para lógica	782	825
LUTs para ruteo	46	43
BRAMs	8	8
IOBs	34	34
Clock máximo simulado	31,65 MHz	31,28 MHz

Tabla 2. Recursos utilizados. Comparación entre la versión 7 y 6 del ISE WebPack.

6 Conclusiones

- El costo de utilizar Wishbone es bajo. No implica un esfuerzo importante ni insume una cantidad importante de recursos.
- 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.
- El ISE WebPack 7 obtiene resultados menos óptimos para la mayor parte de nuestros módulos cuando se lo compara con el 6.
- Es posible reusar código, herramientas y conocimiento adquirido trabajando con microcontroladores al usar tecnologías más avanzadas como las FPGAs.
- La versatilidad de las FPGAs permite incorporar periféricos poderosos con un esfuerzo relativamente bajo. Tal es el caso del generador de video.

Referencias

1. 1st Southern Conference on Programmable Logic - SPL 2005, Mar del Plata, 14-18 marzo 2005 http://www.ii.uam.es/~mcts/Frames/Proj_SCH_UAM_2005_IntroWS_Fset.htm
2. Source Forge.net http://www.sourceforge.net/
3. Tropea, S. E., Brengi D. J., Borgna, J. P. D.: FPGALibre: Herramientas de Software Libre para diseño con FPGAs. SPL 2006, Mar del Plata, 8-10 marzo 2005.
4. Herveille, R.: WISHBONE System on Chip (SoC) Interconnection Architecture for Portable IP Cores. Revision B.3. September 7, 2002. http://prdownloads.sourceforge.net/fpgalibre/wbspec_b3-2.pdf?download.
5. OpenCores: http://www.opencores.org/