DISEÑO E IMPLEMENTACIÓN DE SOPC BASADOS EN EL MICROPROCESADOR PICOBLAZE

J. VIEJO, E. OSTUA, M. J. BELLIDO, J. JUAN, A. MILLAN
P. RUIZ-DE-CLAVIJO Y D. GUERRERO
Departamento de Tecnología Electrónica. Universidad de Sevilla
Av. Reina Mercedes, s/n (E. T. S. Ingeniería Informática) - 41012 Sevilla
{julian, ostua, bellido, jjchico, amillan, paulino, guerre}@dte.us.es
Tel.: +34 954556160 - Fax: +34 954552764
http://www.dte.us.es/gtm/

Con este trabajo pretendemos realizar una aportación a la docencia de la materias que cubren el diseño de SoPC (System on Programmable Chip). Para ello, hemos desarrollado un demostrador de diseño de un SoPC suficientemente sencillo como para que pueda integrarse en una asignatura docente. El demostrador está basado en un microprocesador sencillo como es PicoBlaze, y el sistema completo está pensado para implementarse en FPGAs de la familia Spartan-3 de Xilinx.[1]

1. Introducción

Actualmente, la tecnología microelectrónica permite incluir sistemas completos dentro de una única pastilla de silicio. Son los llamados SoC (System on Chip), dentro de los cuales son de especial interés aquellos que se denominan SoPC (System on Programmable Chip). En éstos, en un mismo chip incluimos microprocesador, memoria, lógica programable para diseño a medida, y otro tipo de componentes que realizan funciones de procesado específicas a las necesidades del sistema.

Este tipo de sistemas supone un reto no sólo para los desarrolladores de sistemas por las innumerables ventajas que supone la implementación de SoPC, sino también para los docentes de las materias relacionadas con la microelectrónica digital. Efectivamente, el diseño e implementación de un SoPC implica que hay que conocer el diseño de sistemas digitales basados en microprocesador junto con el diseño e implementación microelectrónica de sistemas digitales. Es decir, el diseñador de SoPC tiene que unir las capacidades del ingeniero de sistemas con las del ingeniero de diseño microelectrónico.

Esta cantidad de conocimientos que se manejan en el diseño de SoPC obliga a los docentes que se dediquen a estas materias a disponer de un material didáctico especialmente diseñado para facilitar la introducción de los conceptos tanto de diseño de sistemas basados en microprocesadores, como de su implementación a nivel microelectrónico.

Este trabajo está desarrollado con esta idea, es decir, que sirva como material didáctico para aquellas materias docentes donde se tenga como objetivo el diseño e implementación de SoPC. El objetivo del trabajo es presentar un ejemplo de implementación de SoPC sobre FPGA, empleando como microprocesador PicoBlaze, poniendo a disposición de la comunidad docente todo el material presentado.

El trabajo se ha organizado como sigue: las secciones 2 y 3 están dedicadas a presentar el microprocesador PicoBlaze y la metodología básica de implementación de SoPC basados en este microprocesador. En la sección 4, se introducirá el ejemplo de aplicación práctica de diseño e implementación de SoPC. Finalmente, en la quinta sección se expondrán las conclusiones más importantes.

[1]Este trabajo ha sido financiado parcialmente por el proyecto OFU TIC 1023 de la CONSEJERÍA DE INNOVACIÓN, CIENCIA Y EMPRESA de la JUNTA DE ANDALUCÍA.

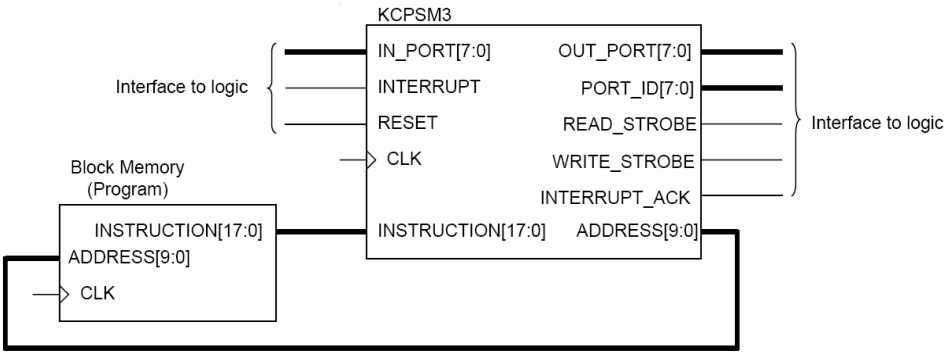

Figura 1: Diagrama de conexión de PicoBlaze con la memoria de programa

2. PicoBlaze

PicoBlaze es un Soft Core de 8 bits, es decir, un microprocesador diseñado para ser 100 % empotrable en FPGAs (Virtex-E, Virtex-II, Virtex-II/Pro, Virtex-4, Spartan-II, Spartan-IIE y Spartan-3) y CPLDs (CoolRunner-II) de Xilinx [1]. Es importante comentar que existen diferentes arquitecturas de este microprocesador (KCPSM [2], KCPSM2 [3] y KCPSM3 [4]), estando cada una de ellas optimizada para un determinado tipo de FPGA.

Al estar el sistema completo pensado para implementarse en FPGAs de la familia Spartan-3, en los siguientes tres apartados vamos a introducir las características, la arquitectura y el juego de instrucciones de la versión de PicoBlaze para este tipo de dispositivos, finalizando esta sección con una comparativa del resto de arquitecturas, que tendrá como objetivo mostrar qué posibilidades nos pueden ofrecer cada una de ellas.

2.1. Caracerísticas generales

PicoBlaze para Spartan-3 [4] tiene las siguientes características:

- ➢ 30 instrucciones (en función de todas las condiciones posibles llegamos a un total de 57).
- ➢ Cada instrucción tiene un tamaño de 18 bits.
- ➢ 16 registros de propósito general de 8 bits.
- ➢ 256 puertos direccionables tanto directa como indirectamente, que nos permiten conectar PicoBlaze con otros dispositivos.
- ➢ Señal de reset, de interrupción y de ack de interrupción (nueva en esta versión).
- ➢ Permite direccionar una memoria de programa de 1024 palabras de 18 bits cada una.

Tanto el microprocesador como la memoria de programa son completamente empotrables en nuestros diseños, por lo que en principio, PicoBlaze no requiere una memoria externa.

El diagrama de conexión de PicoBlaze con dicha memoria quedará como se muestra en la Fig. 1. En esta figura, las señales ADDRESS e INSTRUCTION representan el bus de direcciones y el de instrucciones respectivamente. Además, podemos observar el conjunto de señales que proporcionan a este microprocesador la lógica necesaria para conectarse con otros dispositivos.

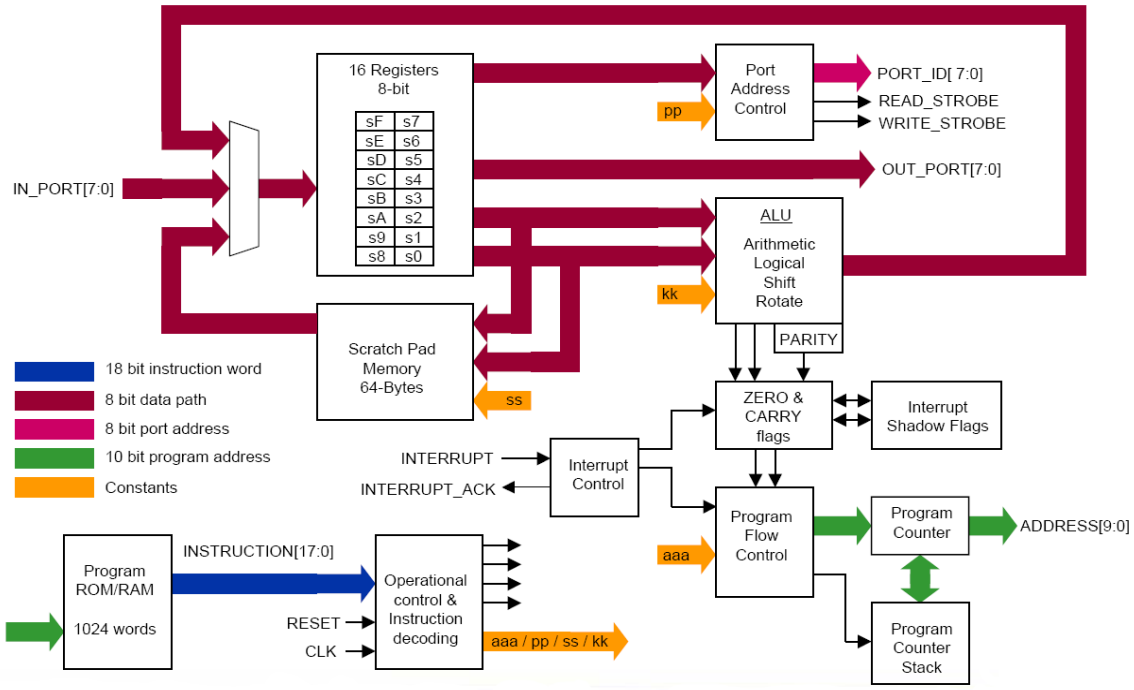

Figura 2: Arquitectura de PicoBlaze

2.2. Arquitectura

En este subapartado, vamos a describir brevemente la arquitectura de PicoBlaze. En términos generales, está compuesta por una unidad de datos y otra de control (ver Fig. 2) donde:

- La unidad de datos (parte superior de la Fig. 2) contiene el banco de registros, el módulo que controla los puertos de entrada/salida, una ALU y una memoria Scratch Pad de 64 Bytes.
- La unidad de control (parte inferior de la Fig. 2) engloba los módulos que se encargan de la decodificación de la instrucción procedente del bus de instrucciones y del control de operación, interrupciones y flujo de programa. También contiene el registro PC (Program Counter) e incluye una pila para dicho registro (Program Counter Stack). Esta pila permite realizar llamadas a subrutinas y saltos, y por tanto, modificar la secuencia normal de ejecución de un programa.

2.3. Juego de instrucciones

Dentro del juego de instrucciones de PicoBlaze podemos distinguir diferentes grupos, entre los que se encuentran:

- De control del programa: JUMP, CALL y RETURN.
- Lógicas: LOAD, AND, OR, XOR y TEST.
- Aritméticas: ADD, ADDCY, SUB, SUBCY y COMPARE.
- De desplazamiento: SR0, SR1, SRX, SRA, SL0, SL1, SLX y SLA.
- De rotación: RR y RL.

Característica	PicoBlaze para Virtex-E y Spartan-IIE	PicoBlaze para Virtex-II y Virtex-II/Pro	PicoBlaze para Spartan-3, Virtex-II/Pro y Virtex-4
Tamaño de instrucción	16 bits	18 bits	18 bits
Registros de 8 bits	16	32	16
Espacio de programa	256 instrucciones	1024 instrucciones	1024 instrucciones
Profundidad de la pila	15	31	31
Ensamblador	KCPSM	KCPSM2	KCPSM3
Tamaño	76 slices de una Spartan-IIE	84 slices de una Virtex-II	96 slices de una Spartan-3
Rendimiento	37 MIPS (Spartan-IIE)	55,8 MIPS (Virtex-II)	44 MIPS (Spartan-3) y 100 MIPS (Virtex-II/Pro)
Memoria Scratch-Pad	-	-	64 Bytes

Tabla 1: Comparativa entre las distintas arquitecturas

- ➢ De entrada/salida: INPUT y OUTPUT.
- ➢ De interrupción: ENABLE/DISABLE INTERRUPT y RETURNI ENABLE/DISABLE.
- ➢ De manejo de la memoria Scratch Pad: STORE y FETCH.

Todas las instrucciones de PicoBlaze se ejecutan bajo todas las condiciones en 2 ciclos de reloj, siendo el ancho de datos de 8 bits. Cuando realizamos operaciones lógicas y aritméticas con la ALU, el primer operando es siempre un registro, que además funciona como registro destino, es decir, aquél en el que guardamos el resultado de la operación. El segundo es, bien un registro o bien un valor constante que viene como parámetro en la instrucción que estamos ejecutando.

2.4. Comparativa entre las distintas arquitecturas

En la Tabla 1 presentamos una comparativa de las caracterásticas funcionales de las diferentes versiones de PicoBlaze para cada tipo de FPGA. Como podemos comprobar en dicha tabla, muchas de las características de PicoBlaze para Virtex-E y Spartan-IIE se han visto mejoradas en las arquitecturas para Virtex-II, Virtex-II/Pro y Spartan-3: se amplía el tamaño de la instrucción de 16 a 18 bits, el espacio de programa pasa de 256 a 1024 instrucciones y el de la pila de 15 a 31 niveles. En la Tabla 1 también observamos como la arquitectura KCPSM3 dispone de un bloque de memoria Scratch Pad y de dos nuevas instrucciones (STORE y FETCH) para control de esta memoria. En el caso de PicoBlaze para KCPSM2 podemos destacar que cuenta con el doble de registros de propósito general que las otras dos arquitecturas.

Finalmente, observamos cómo las nuevas versiones mejoran el rendimiento general, pasando de los 37 MIPS en una Spartan-IIE hasta los 100 MIPS en una Virtex-II/Pro.

3. Metodología de diseño e implementación

La estructura mínima de un SoPC basado en PicoBlaze se compone del módulo de PicoBlaze (kcpsm3.vhd) y la memoria que almacena el programa (<ROM>.vhd). A estos componentes se les tiene que unir la interfaz de conexión con las señales de entrada y de salida, así como el resto de componentes de procesado que se necesiten en cada sistema específico.

La metodología de diseño de este tipo de sistemas está basada en lenguajes de descripción de hardware, en nuestro caso en VHDL [5].

En la Fig. 3 se muestra la estructura del código VHDL que describe el sistema. Como se observa, es una descripción que bien es completamente estructural, es decir, todos los componentes están descritos en módulos individuales y en la descripción del sistema solamente se establece el interconexionado, o bien puede ser una descripción mixta estructural-comportamiento. En cualquier caso, como se puede observar en el código de la Fig. 3, el microprocesador PicoBlaze y la memoria de programa se incluyen en el código VHDL del sistema como componentes. Para el resto de elementos puede emplearse una descripción estructural o de comportamiento.

En la Fig. 4 se muestra la metodología seguida para el diseño e implementación del sistema. En dicha figura, observamos como el primer paso de la metodología propuesta es describir el sistema. En esta descripción, hay que emplear el módulo de PicoBlaze adecuado a la FPGA que se vaya a programar, y además, generar el código de la memoria de programa y del resto de componentes de interconexión y procesado.

Para generar el fichero que define la memoria de programa (<ROM>.vhd), empleamos el ensamblador proporcionado junto a PicoBlaze, quedando el proceso de generación como sigue:

1. Escribimos un fichero con el programa (<ROM>.psm).
2. Utilizamos el ensamblador de PicoBlaze para generar el fichero VHDL que define la memoria que contiene el programa (<ROM>.psm).

Una vez que tengamos descrito completamente nuestro sistema, realizaremos una verificación del mismo utilizando la herramienta de simulación ModelSim XE III [6]. Si tras finalizar la simulación con ModelSim comprobamos que el diseño no se ajusta correctamente a las especificaciones de partida, realizaremos las modificaciones que considermos oportunas del programa y/o del resto de componentes, repitiendo este paso hasta que obtengamos el comportamiento deseado.

A continuación, sintetizaremos e implementaremos nuestro diseño empleando las herramientas del entorno de Xilinx ISE [7]. Finalmente, configuraremos la placa y descargaremos el sistema implementado en la FPGA a través de la herramienta iMPACT, comprobando sobre la propia placa si el resultado obtenido se ajusta al deseado, volviendo nuevamente a un paso anterior en caso contrario.

4. Ejemplo de aplicación

En este apartado, vamos a realizar una descripción del diseños que se ha desarrollado con PicoBlaze, en concreto, una UART que envía a un terminal los caracteres recibidos desde un teclado. Así, en la Fig. 5. mostramos el diagrama de bloques del dicho diseño, donde podemos distinguir los siguientes componentes:

1. Módulo embedded_kcpsm3, donde se realiza la conexión de PicoBlaze con la memoria de programa.
2. Módulo x1, que interviene como interfaz entre los distintos controladores y PicoBlaze [8].
3. Módulo controlador_ps2. Se trata del controlador del teclado [9].

```
--Descripción externa del sistema
entity <SoPC> is
 Port (
   --Declaración de las señales de entrada al sistema
   --Declaración de las señales de salida del sistema
   clk: in std_logic
 );
end <SoPC>;
--Descripción interna del sistema
architecture Mixta_Estructural_Comportamiento of <SoPC> is
--Declaración del componente PicoBlaze
 component kcpsm3
  Port (
    address : out std_logic_vector(9 downto 0);
    instruction : in std_logic_vector(17 downto 0);
    --Declaración del resto de señales de PicoBlaze
    clk : in std_logic
  );
 end component;
--Declaración de la memoria de programa
 component <ROM>
  Port (
    address : in std_logic_vector(9 downto 0);
    instruction : out std_logic_vector(17 downto 0);
    clk : in std_logic
  );
 end component;
--Declaración de otros componentes y señales
--Descripción del sistema
begin
--Conexión de PicoBlaze con la memoria de programa
 processor: kcpsm3
  port map (
    address => address,
    instruction => instruction,
    --Conexión del resto de señales de PicoBlaze
    clk => clk
  );
 program: <ROM>
  port map (
    address => address,
    instruction => instruction,
    clk => clk
  );
--Colocación de otros componentes
--Descripción funcional de otras partes del sistema
end Mixta_Estructural_Comportamiento;
```

Figura 3: Implementación en VHDL del sistema

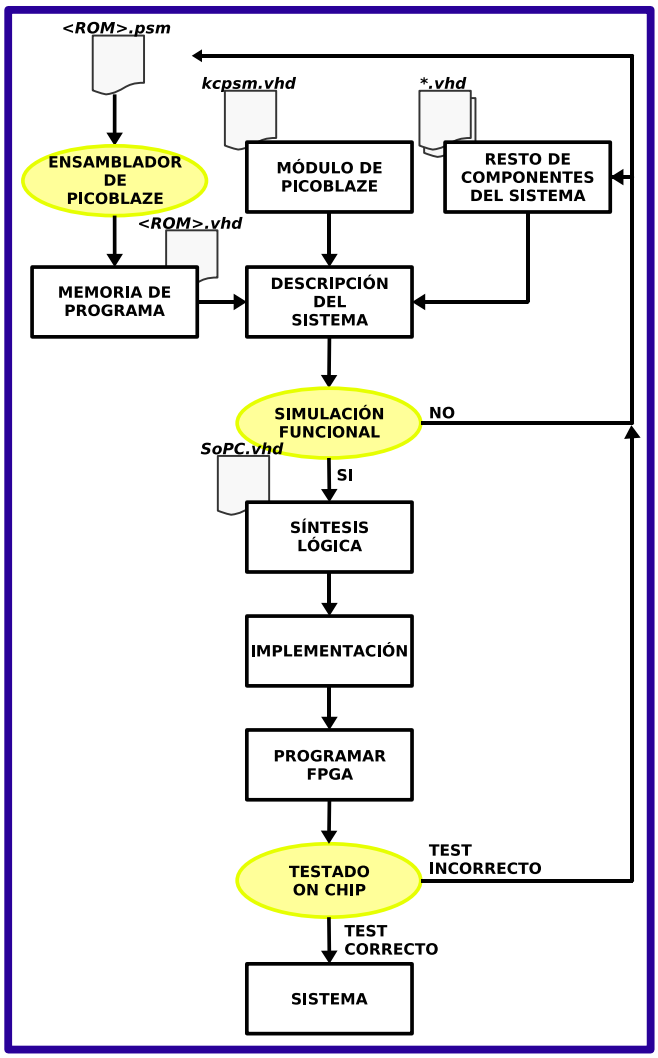

Figura 4: Metodología de diseño e implementación seguida

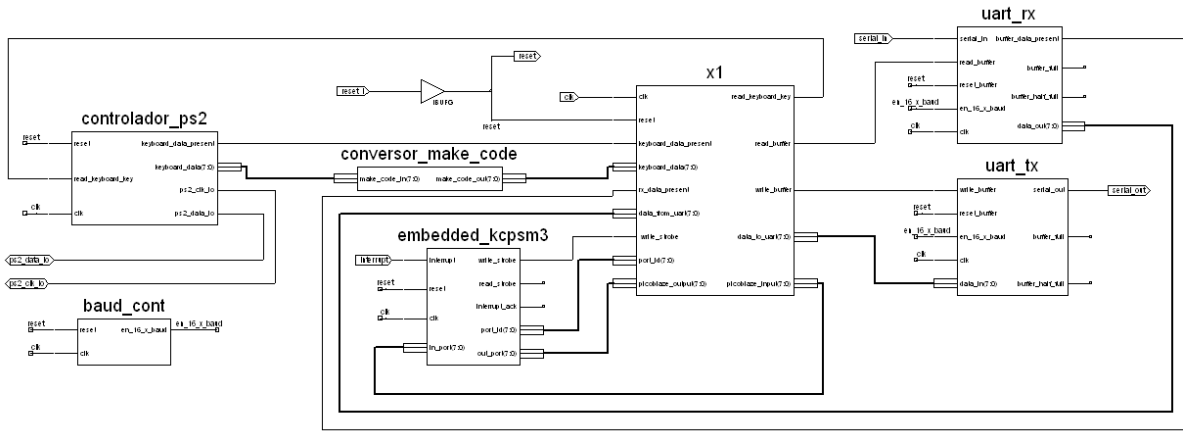

Figura 5: Esquema del diseño desarrollado

4. Módulo conversor_make_code, que sirve para convertir el make code recibido a su código ASCII correspondiente.

5. Módulos baud_cont, uart_rx y uart_tx. Estos tres módulos corresponden al controlador del puerto serie[10].

Para el desarrollo de este ejemplo, se ha empleado un dispositivo Spartan-3 Starter Kit Board de Digilent [11] que contiene una Spartan-3 XC3S200.

5. Conclusiones

Con este trabajo hemos pretendido cubrir un hueco que, desde nuestra perspectiva, creemos que existía en cuanto a la docencia del diseño e implementación de SoPC. En el trabajo hemos analizado las características de PicoBlaze, y además, hemos propuesto una metodología de diseño e implementación de SoPC basado en este microprocesador.

Por último, hemos planteado y desarrollado un ejemplo de diseño de SoPC empleando elementos que son muy accesibles: códigos del microprocesador Picoblaze y placas de evaluación con FPGAs de Xilinx tipo Spartan-3.

Referencias

[1] "Web Oficial de Xilinx." http://www.xilinx.com/.

[2] K. Chapman, "PicoBlaze 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices." http://www.xilinx.com/bvdocs/appnotes/xapp213.pdf, February 2003.

[3] K. Chapman, "PicoBlaze 8-Bit Microcontroller for Virtex-II Series Devices." http://www.xilinx.com/bvdocs/appnotes/xapp627.pdf, February 2003.

[4] K. Chapman, "PicoBlaze 8-Bit Embedded Microcontroller User Guide for Spartan-3, Virtex-II and Virtex-II PRO FPGAs." http://www.xilinx.com/bvdocs/userguides/ug129.pdf, November 2005.

[5] "VHDL International." http://www.vhdl.org/.

[6] "Modelsim Xilinx Edition III." http://www.xilinx.com/ise/optional_prod/mxe.htm.

[7] "Using the ISE Design Tools for Spartan-3 Generation FPGAs." http://www.xilinx.com/bvdocs/appnotes/xapp473.pdf, May 2005.

[8] S. Shaheen, "The X1 and X2 Modules integrate Xilinx PicoBlaze and UART Modules." http://www.opencores.org/forums.cgi/cores/2004/06/000925, June 2004.

[9] D. Quintero, "Compact and optimized PS/2 controller for Keyboard and Mice." http://www.opencores.org/projects.cgi/web/ps2core/overview, April 2003.

[10] K. Chapman, "UART Transmitter and Receiver Macros," October 2002.

[11] "Spartan-3 Starter Kit Board User Guide." http://www.xilinx.com/bvdocs/userguides/ug130.pdf, May 2005.