Diseño e Implementación en FPGA de un Microprocesador *Self-Timed* de 16 bits, utilizando el protocolo de 4 fases.

Ortega-Cisneros S. Raygoza-Panduro J.J. y Boemo E. [1]

[1] Escuela Politécnica Superior, Universidad Autónoma de Madrid.
Ctra. de Colmenar Km. 15, 28049, Madrid, España
{susana.ortega_cisneros, juan_jose.raygoza_panduro, eduardo.boemo}@ii.uam.es

Resumen. En este artículo se presenta el diseño e implementación de un microprocesador *self-timed*, utilizando el protocolo de señalización de 4 fases en FPGA. Aunque estos dispositivos están orientados para la implementación eficiente de circuitos síncronos, en la actualidad constituyen la única opción disponible para el prototipado rápido y de bajo coste de circuitos *self-timed*. Se presenta la metodología de implementación, la descripción de la arquitectura y la comunicación entre las señales de control para la transferencia de información. También se resumen los resultados de la simulación para una FPGA Xilinx Virtex II, así como los parámetros de área, distribución de pistas y *fanout*.

Keywords: Circuitos *Self-timed*, microprocesador asíncrono, FPGA, Protocolo de 4 fases.

1 Introducción

La realización de sistemas digitales autotemporizados, constituyen una alternativa para sincronizar circuitos de gran tamaño. Por esta razón la metodología de diseño *Self-timed* (en adelante ST) ha recibido un importante impulso en los últimos años. Entre sus ventajas se pueden mencionar su inherente funcionamiento en modo *stoppable-clock*, la ausencia de picos de consumo y su inmunidad al *skew* del reloj. En un circuito síncrono la transmisión o procesamiento de datos es controlada globalmente por una o más fases de reloj. Por el contrario en un sistema ST la transferencia de datos es controlada por dos señales "petición (*request*) y reconocimiento (*acknowledge*)" de la forma habitual que en cualquier sistema asíncrono [1]. La definición del formato de las señales de control da lugar a dos tipos de sincronización: Protocolo de 2 fases [2] y protocolo de 4 fases [3,4].

Este artículo presenta la implementación de un microprocesador utilizando el protocolo de 4 fases. Este ha sido seleccionado debido a la robustez de la técnica, simplicidad de implementación de los bloques de transmisión y menor ocupación en los recursos del dispositivo FPGA con respecto a su homólogo de 2 fases. [5,6]

En la actualidad el desarrollo de microprocesadores ST se ha centrado en prototipos *full-custom* o *cell-based,* entre los cuales podemos mencionar el AMULET propuesto en la Universidad de Manchester, con diferentes versiones, y el TICTAC, del Instituto de Tecnología de Tokio y la Universidad de Tokio. Entre los procesadores ST comerciales se encuentra el microprocesador 80C51 desarrollado por Phillips, utilizado en radios localizadores (*pagers*) y en aplicaciones de comunicación inalámbrica. También puede mencionarse el procesador DNP de la compañía Sharp, utilizado en receptores de televisión digital. Sin embargo, el desarrollo de procesadores con esta tecnología presenta algunos inconvenientes, tales como el tiempo de diseño y el coste de la implementación del *chip* [7,8]. En este trabajo se presenta la utilización de los circuitos programables FPGAs, como una opción de bajo coste para el desarrollo e implementación de sistemas ST embebidos.

2 Descripción del microprocesador ST

El diseño del microprocesador ST ha seguido el flujo de diseño estándar de FPGAs: 1º. Diseño de los componentes y procesos básicos; 2º. Descripción estructural de los bloques del procesador en VHDL; 3º. Síntesis del código, 4º. Verificación y simulación del sistema y finalmente, 5º. Programación del dispositivo Virtex II FPGA.

2.1 Implementación de la arquitectura

El microprocesador ST desarrollado en este trabajo tiene 16 bits, una arquitectura RISC con 6 etapas de *pipeline*, correspondientes a cada uno de los módulos (ALU, PC, MAR, OPR, GPR y memorias). Cuenta con un conjunto de 32 instrucciones, 4 de ellas con acceso a memoria, 3 aritméticas, 5 lógicas, 4 desplazamientos, 3 saltos y 12 de control. Contiene un banco de 6 registros de los cuales 4 son de propósito general y 2 específicos. Cuenta con 2 puertos de 16 bits, uno de entrada y otro de salida.

En la figura 2.1 se ilustra la distribución de cada uno de estos módulos, incluyendo el controlador de programa y los bloques de control asíncrono (BCAs) emisores y receptores, interconectados a las etapas *pipeline* a través de las líneas xi(1) a xi(32). El microprocesador se ha sintetizado utilizando VHDL estructural y realizando inferencias a los recursos específicos de los dispositivos Xilinx, tales como funciones aritméticas, *buffers*, comparadores, contadores, *flip-flops*, *latches*, multiplexores, y los bloques de memoria RAM de puerto doble RAMB16_36_36 (9 líneas de direccionamiento y 32 de datos).

2.2 Controlador

El controlador del sistema consta de 6 bloques, los cuales se ilustran en la figura 2.2. El bloque 1 contiene la unidad de control asíncrona, que esta constituida por elementos

BCAs. Este módulo suministra las señales de activación "xi" que habilitan a los registros *pipeline* y componentes del procesador.

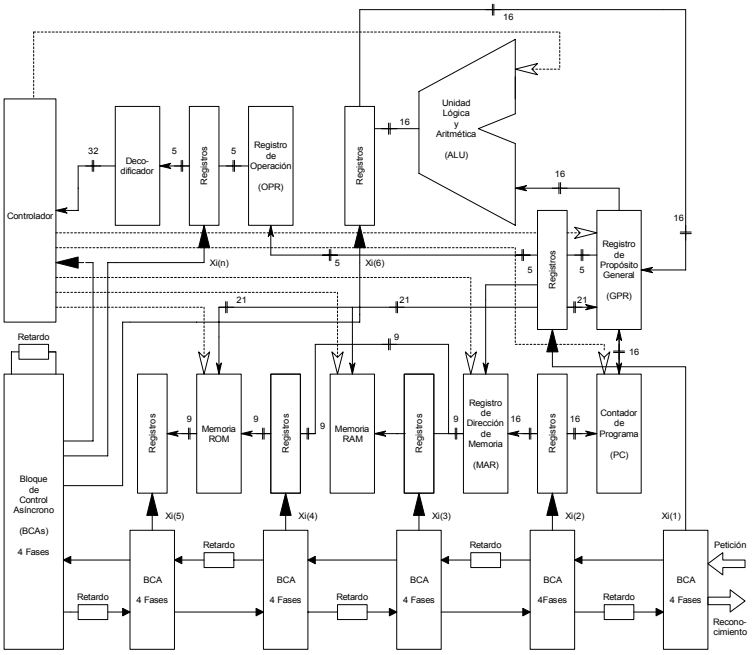

Fig. 2.1. Arquitectura del microprocesador ST de 16 bits.

El ciclo de búsqueda (bloque 2), está constituido por 4 operaciones básicas, las cuales requieren 12 señales de activación "xi" por ciclo. El bloque 3 esta constituido por el decodificador de operación, que habilita las 32 microinstrucciones disponibles, donde cada una de ellas requiere 6 señales de activación "xi" para completar su ejecución. Cada microinstrucción es seleccionada y ejecutada por el decodificador de instrucciones (bloque 4) y el ciclo de instrucción (bloque 5). La distribución de las señales de activación es responsabilidad del módulo *arbiter* (bloque 6). Por este dispositivo pasan las señales "xi" de cada microinstrucción que hacen referencia a la ALU.

Para que el sistema se inicie correctamente, se deber activar el *reset* general. Posteriormente se aplica un evento (pulso externo) a la entrada de la señal de petición "ri" del primer bloque de control asíncrono, y con este único pulso se inicia la secuencia de petición-reconocimiento de la unidad de control que activa al ciclo de búsqueda y comienza el proceso de ejecución de las microinstrucciones. Una vez terminado el primer ciclo de ejecución, la señal de petición es realimentada al BCA inicial a través de un elemento *arbiter*, el cual permite la transmisión de la señal de petición siguiente. Este proceso se realiza cíclicamente hasta que termina la ejecución del programa. Una vez concluido el programa, el microprocesador se detiene incorporando de manera inherente la característica de *stoppable clock*; es decir el

circuito se para (consumo dinámico cero) hasta que no es requerido su funcionamiento nuevamente por un señal de petición "ri" externa.

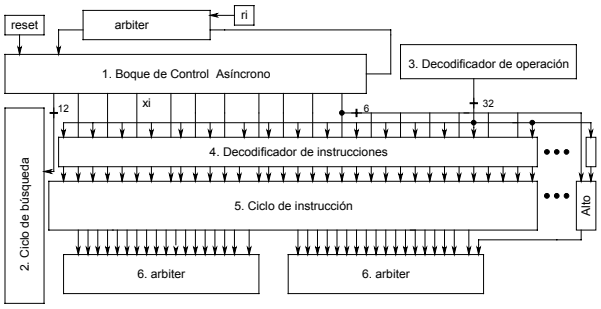

Fig. 2.2. Controlador del microprocesador ST

2.3 Control asíncrono

El control ST del microprocesador regula el flujo de datos a través de todo el sistema y utiliza un protocolo de comunicación de datos de 4 fases, también conocido como "protocolo de señalización por nivel". En este se utilizan tanto *latches* como *flip flops (FF)* como elementos de almacenamiento en la ruta de datos. Estos componentes están agrupados en módulos de control idénticos, distribuidos a lo largo de la ruta de control. En el protocolo de 4 fases sólo se produce una transferencia de datos en los flancos de subida de la señal de petición. Para ilustrar el mecanismo de sincronización del circuito es conveniente focalizar la atención sobre una etapa genérica *pipeline*, cuyo comportamiento es idéntico al resto de los bloques. El análisis se descompone en una sucesión de eventos controlada por las líneas de activación, que regulan a cada bloque de registros que segmentan al circuito. El funcionamiento de este tipo de bloques ST se describe con detalle en [9].

3 Resultados de la implementación en FPGAs

Los resultados de simulación se pueden observar en la figura 3a, en donde se presentan las señales de activación del ciclo de búsqueda y el código de operación de algunas microinstrucciones. Uno de los problemas en la construcción de circuitos ST en tecnologías con un alto grado de automatización del proceso de *placement* y *routing* es la dificultad de conocer anticipadamente los retardos de interconexión. Estos datos son fundamentales para dimensionar el retardo asíncrono que se debe asignar a la señal de petición de cada etapa del *pipeline*.
En la figura 3b se muestran los retardos en función del *fanout* de cada nodo, del microprocesador. También se presentan los histogramas de retardo pistas. La pista de *reset* tiene el *fanout* más elevado 164; no es graficado pues no interviene en la frecuencia máxima del circuito. El mayor retardo de pista es 4 ns.

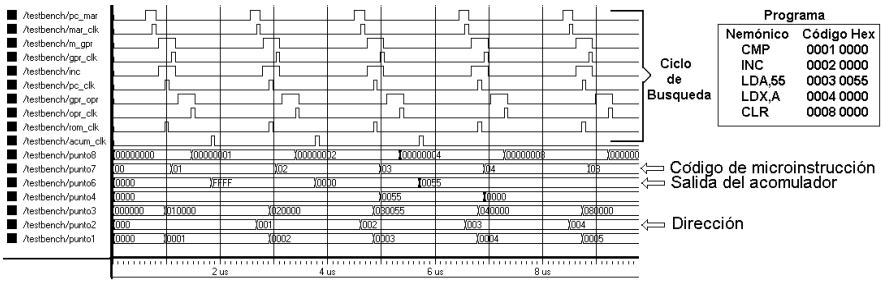

Fig. 3a) Simulación del microprocesador ST

En general, más del 80 % de las pistas tiene un retardo menor que 2 ns, siguiendo una distribución de Pareto-Levy [10].

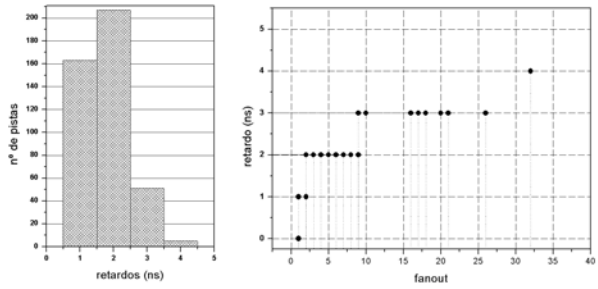

Fig. 3b) Retardo vs *fanout* del microprocesador ST.

Se puede observar que la mayoría de las pistas tienen un *fanout* que oscila en el rango 0 a 34, con retardos de entre 0 y 4 ns. En la figura 3c se muestran los resultados de la implementación para el microprocesador, en la que puede observarse una ocupación aproximada del 50% de los recursos de la FPGA Virtex II (xc2v1000-4 FG456). Los recursos utilizados son 2,815 de los 5,120 slices (54%), 171 de los 10,240 *latches* o FF (1%), 563 de los 10,240 *LUTs* (5%).

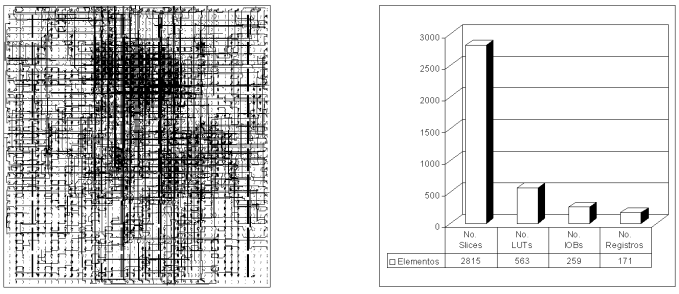

Fig. 3c) Ocupación del microprocesador ST.

4 Conclusiones

En este trabajo se muestra la factibilidad de la implementación de microprocesadores ST, en un dispositivo programable FPGA. Se han presentado algunas ideas y resultados numéricos útiles para el prototipado rápido de circuitos *self-timed* de 4 fases en FPGAs. El procesador presenta un inherente funcionamiento en bajo consumo, debido a que las señales de sincronización del microprocesador se detienen cuando termina el programa (*stoppable clock*); es decir, el consumo dinámico es cero, mientras no sea requerido nuevamente por un señal de petición "ri" externa.

Agradecimientos

Este trabajo ha sido financiado mediante el Proyecto 07T/0052/2003-3 de la Consejería de Educación de la Comunidad de Madrid, España y por la beca para estudios otorgada por el Consejo Nacional de Ciencia y Tecnología (CONACYT) de México.

Referencias

[1] I.E. Sutherland, "Micropipelines", *Communications of the ACM*, Vol. 32 No. 6, June 1989, pp. 720-738.

[2] Arechavala, L. "Design Self-Timed", MsSc. Thesis University of Manchester, Department of Computer Science. UK, 1994.

[3] Ortega S, Raygoza J.J y Boemo E. "Sincronización Self-Timed: Protocolo de 4 fases", Proc. JCRA 2003, Universidad Autónoma de Madrid 2003, pp. 517-528.

[4] Anthony J. McAuley, "Four State Asynchronous Architectures", IEEE transactions on computers, vol 41,No2, Feb. 1992.

[5] B. Furber and J. Liu, "Dynamic logic in four-phase micropipelines", in *Proc. InternationalSymposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, Mar. 1996.

[6] Kees van Berkel and Arjan Bink. "Single-track handshaking signaling with application to micropipelines and handshake circuits", In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 122-133. IEEE Computer Society Press, March 1996.

[7] Brunvand, E.; Nowick, S.; Yun, K.; Practical advances in asynchronous design and in asynchronous/ synchronous interfaces", *Proc. 36th Design Automation Conference*, page(s): 104 - 109, 21-25 June 1999.

[8] S. B. Furber, J. D. Garside, and D. A. Gilbert, "AMULET3: A high-performance self-timed ARM microprocessor," in *Proc. International Conf. Computer Design (ICCD)*, Oct. 1998.

[9] P. Day and J. V.Woods, "Investigation into micropipeline latch design styles," *IEEE Transactionson VLSI Systems*, vol. 3, pp. 264–272, June 1995.

[10] S. John K. Dagsvik and Bjorn H. Vatne. "Is the Distribution of Income Compatible with a Stable Distribution", Statistics Norway, Research Department. Discussion Papers No. 246, January 1999