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

Periférico para el cálculo del logaritmo, descripción de hardware

Tropea, S. E.[(i)]

[(i)]INTI-Electrónica e Informática

Introducción

En este trabajo presentamos el desarrollo de un periférico capaz de realizar el cálculo del logaritmo de un valor representado en punto fijo.

Este desarrollo fue realizado para uso no exclusivo de la empresa INVAP S.E.[1] para ser utilizado en un sistema de seguridad y control de reactores nucleares, calificado para operar en ambiente espacial[2].

El objetivo de este periférico es el de realizar un cálculo matemático muy útil y a altas velocidades, sin ocupar el tiempo de ejecución del microcontrolador principal. Este tipo de cálculos es muy usado en aplicaciones tales como filtros digitales, cálculos de potencia (dB) y sistemas numéricos logarítmicos (LNS = *Logarithmic Number System*). El cálculo del logaritmo es un componente clave de los LNS que ofrecen buenas ventajas sobre los sistemas de coma flotante tradicionales[3].

Como objetivo particular el cliente exigía que este periférico no ocupara más del 8% del área disponible en una *FPGA Spartan* II[4] 200 (XC2S200-6). Adicionalmente la especificación exigía un error máximo de 0,05% referido al fondo de escala. Los requerimientos de velocidad eran que el mismo debía computar un nuevo valor en menos de 4 ms cuando se utilizara un reloj de 20 MHz.

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 de *Xilinx*. Aún así el mismo fue realizado de manera tal que pudiera ser sintetizado con cualquier *FPGA*. Esto fue de vital importancia debido a que el cliente utiliza *FPGAs* de *Actel*[5] tolerantes a rayos cósmicos, 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[6][7][8] impulsado por nuestro laboratorio. Para este desarrollo se utilizaron estaciones de trabajo que corren Debian[9] GNU[10]/Linux.

De acuerdo con las especificaciones del cliente los valores de entrada estaban en el rango 0,01 a 10.000.000.000 representados en formato de punto fijo con 34 bits de parte entera y 7 bits de parte decimal. El resultado del logaritmo decimal para estos valores pertenece al rango de -2 a 10 y la resolución requerida fue de 10 bits de parte decimal.

Existen dos mecanismos principales para el cálculo del logaritmo con circuitos electrónicos. El primero es el uso de tablas con valores conocidos, estas tablas pueden complementarse con algún tipo de interpolación. El segundo mecanismo consiste en realizar aproximaciones que van acercando el resultado obtenido al buscado, luego de un cierto número de iteraciones se obtiene un valor aproximado. El primer método es muy rápido, pero suele consumir grandes cantidades de recursos para las tablas. El segundo es más lento pero no necesita de tablas de gran tamaño. Usualmente se utilizan mecanismos híbridos que explotan ventajas de ambos métodos[11][12][13][14]. El tipo de *FPGA* utilizado por el cliente no dispone de memoria embebida y por lo tanto el uso de tablas medianas o grandes insumiría demasiados recursos de la misma. Por esta razón se descartó este tipo de algoritmos y se buscó un algoritmo que utilizara aproximaciones.

Un mecanismo muy utilizado para realizar este cálculo es el de las series de Taylor, este mecanismo no es bueno para su implementación en *hardware* debido a su lenta convergencia y alta complejidad de cálculo. Otro algoritmo comúnmente usado en sistemas digitales es el CORDIC (*Coordinate Rotation Digital Computer*), el mismo permite calcular logaritmos naturales para un pequeño rango de valores de entrada. Para utilizar el algoritmo CORDIC en un rango de valores tan amplio, como el planteado por este proyecto, es necesario realizar cambios de escala muy complejos. Una alternativa de rápida convergencia y de implementación compacta es la utilización de la denominada "normalización multiplicativa"[15][16][17]. Este algoritmo consiste en utilizar una pequeña tabla, de nueve valores en nuestro caso, que contiene los resultados para algunos valores en particular. El truco consiste en combinar los escasos valores de la tabla para obtener el resultado correspondiente a cualquier valor de entrada. El algoritmo permite converger al resultado en pocas iteraciones, ocho en nuestro caso. Los cálculos realizados en dicha iteración son simples sumas y desplazamientos[18].

Este tipo de aproximaciones son válidas para un rango pequeño de valores por lo que es necesario realizar un cambio de escala para extender dicho rango. Otro detalle importante es que la aproximación sólo sirve para una única base. Debido a que el cambio de escala debe ser compensado al terminar el cálculo se seleccionó la base 2 para la aproximación, esto simplifica notablemente el cambio de escala y su ajuste. Para obtener el logaritmo decimal es necesario realizar un cambio de base que consiste en la multiplicación del resultado por una constante. El la *Fig. 1* se muestra un diagrama en bloques del cálculo.

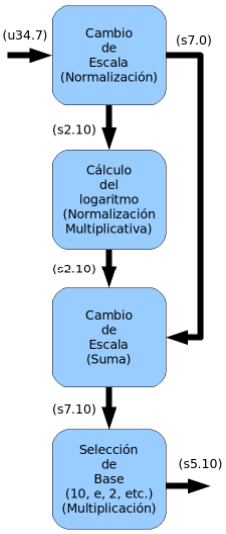

Fig. 1: Diagrama en bloques del cálculo. Los valores entre paréntesis indican el tipo de representación en punto fijo. La u indica sin signo y la s con signo, el primer valor corresponde a la cantidad de bits usados para la parte entera y el segundo a la cantidad de bits usados para la parte decimal.

Luego de seleccionar este algoritmo se procedió a su implementación en *hardware* y a la creación de bancos de prueba que permitieran verificar el correcto funcionamiento del mismo.

A los fines de poder mejorar la reusabilidad del periférico el mismo se diseñó de manera tal que el tamaño del valor de entrada y la resolución del valor de salida pudieran ser configurables. También se tuvo en cuenta la posibilidad de modificación del periférico para poder calcular logaritmos naturales y base 2.

Resultados

Se obtuvo una descripción de un periférico capaz de calcular el logaritmo de una magnitud representada en punto fijo, escrito en *VHDL* estándar, y por lo tanto útil para ser utilizado con cualquier *FPGA*.

El mismo puede ser configurado para obtener entre 8 y 26 bits de parte decimal a su salida. Al mismo tiempo es posible configurarlo para computar logaritmos decimales, naturales o base 2. Otras bases son posibles realizando pequeñas modificaciones.

En el caso requerido por el cliente el periférico posee una latencia de entre 23 y 63 ciclos de reloj, una vez que se encuentra en régimen puede calcular un nuevo valor cada 12 ciclos de reloj en promedio. Esto significa que la velocidad de cálculo promedio es de 1,7 millones de valores por segundo, cuando se utiliza un reloj de 20 MHz. Aún cuando el periférico no se utilice en régimen constante y aún cuando se cuenten los ciclos de reloj extra necesarios para la implementación de la interfaz requerida por el cliente este periférico es capaz de computar el resultado en 3,4 µs.

El área ocupada por el periférico fue de 156 *slices* lo que corresponde al 6,63 % de la *FPGA* antes mencionada. El resultado contiene un error menor al 0,02 % del fondo de escala.

Conclusiones

El periférico realizado logró cumplir con los objetivos de área ocupada, velocidad de cálculo y precisión de cálculo.

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.

El algoritmo seleccionado y adaptado demostró ser compacto, 6,63 % de la *FPGA* usada como referencia y menos de 1/500 de una moderna Virtex 4 LX 200. Al mismo tiempo el algoritmo ofrece una velocidad de cálculo alta, 1,7 millones de cálculos por segundo en las condiciones planteadas y aproximadamente 16 millones de cálculos por segundo cuando se considera un dispositivo como la Virtex 4. La relación área/velocidad del periférico es excelente. Para aplicaciones donde fuera necesario una mayor velocidad es posible utilizar más de un periférico en paralelo.

La cantidad de periféricos que se pueden utilizar en paralelo queda limitada sólo por la cantidad de celdas lógicas disponibles y no por recursos especiales de la *FPGA* tales como memoria embebida. Los algoritmos basados en grandes tablas son más rápidos, pero consumen recursos de memoria embebida por lo que la cantidad que se pueden utilizar en paralelo queda limitada.

Referencias

[1] INVAP S. E., http://www.invap.net/
[2] 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.
[3] M. Haselman, M. Beauchamp, A. Wood, et al, "A Comparison of Floating Point and Logarithmic Number Systems for *FPGAs*," Proc. of the 13th Annual IEEE Symp. on Field-Prog. Custom Comp. (FCCM'05) 0-7695-2445, 2005.
[4] *Xilinx Spartan* II *FPGA*, http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan2_fpgas/index.htm
[5] *Actel* Corporation, http://www.actel.com/
[6] 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.
[7] 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.
[8] Proyecto *FPGA* Libre, http://fpgalibre.sourceforge.net/
[9] Debian project, http://www.debian.org/
[10] GNU project, http://www.gnu.org/
[11] Y. Wan and C. L. Wey, "Efficient algorithms for binary logarithmic conversion and addition," IEE Proc.-Comp. Digit. Tech., vol. 146, no. 3, 1999.
[12] T. C. Chen, "Automatic computation of exponential, logarithms rations and square roots," IBM J. Res. Develop., pp. 380-388, 1972.
[13] H.-Y. Lo and Y. Aoki, "Generation of a precise binary logarithm with difference grouping programmable logic array," IEEE Trans. Comput., vol. C-34, pp. 681-691, 1985.
[14] S.-Y. Shi, "Shortcut to logarithms combine table lookup and computation," Comput. Design., pp. 186-189, 1976.
[15] I. Koren, "Computer arithmetic algorithms, 2nd edition," ISBN 1-56881-160-8, pp. 225-232.
[16] B. Parhami, "Computer Arithmetic Algorithms and Hardware Designs," ISBN 0-19-512583-5, pp 378-381.

[17] M. Pascale, "Microcontrollers & CORDIC methods,", Dr. Dobb's Journal, http://www.ddj.com/184404244, Jul. 2001.
[18] S. E. Tropea, "*FPGA* Implementation of Base-N Logarithm", III Southern Conference on Programmable Logic - SPL 2007, Proceedings (ISBN 978-1-4244-0606-7), pp 27-32, 2007.

Para mayor información contactarse con:

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