Unidad aritmética en coma flotante para sistemas auto-reconfigurables dinámicamente sobre Spartan-3 basados en Microblaze.

Lumbiarres López R. [1], López García M. [1], Cantó Navarro E.[2], Ramos Lara R.[1]

[1] Dptos. Ing. Automática y Electrónica, Escuela Politécnica Superior de Vilanova i la Geltrú, España,
Ruben.Lumbiarres@upc.es
Lopezg@eel.upc.es
Lara@eel.upc.edu
[2] Universidad Rovira i Virgili , Tarragona, España,
ecanto@urv.net

Resumen. El presente artículo muestra la implementación de una unidad en coma flotante (FPU) que actúa como coprocesador dentro de un sistema auto-reconfigurable dinámicamente. La FPU tiene capacidad para resolver operaciones básicas como la suma, la resta, el producto, el cociente, la raíz cuadrada, la inversa y el cuadrado. Además, dispone de un registro en el que se almacena el último resultado obtenido con la intención de utilizarlo como operador en el siguiente cálculo, de modo que se reducen los accesos a los buses de comunicación en la resolución de las operaciones matemáticas. El diseño emplea Microblaze como microprocesador del sistema y su implementación se ha realizado sobre una FPGA Spartan 3 de bajo coste. El artículo muestra resultados experimentales en relación al área total ocupada, así como los tiempos de ejecución obtenidos con un ejemplo particular basado en un algoritmo de CORDIC resuelto en coma flotante.

1. Introducción

El uso de aritmética en coma flotante es habitual en la programación de algoritmos orientados al procesado donde la precisión de las operaciones es determinante para su correcta resolución. La codificación de los operandos en este formato viene definida por la norma IEEE 754 [1]. La forma más simple que tiene el microprocesador de resolver operaciones en este formato es la ejecución de un algoritmo puramente software. Como contrapartida los tiempos de procesado, en comparación con la aritmética con enteros, pueden llegar a ser significativos. Otra posibilidad es utilizar microprocesadores que incorporen una unidad en coma flotante interna destinada a resolver dichas operaciones por hardware. La mejora en el tiempo de ejecución es notable a costa de utilizar un área específica mayor orientada a la implementación de la propia FPU. La idoneidad de una u otra alternativa depende de la precisión y la funcionalidad del algoritmo, y del uso más o menos intensivo que haga de operaciones en coma flotante. Así, en un proceso donde prácticamente todo el tiempo de ejecución se opere con este tipo de aritmética será adecuado y sobradamente justificable utilizar un microprocesador que incorpore una FPU. Por el contrario, en procesos donde por ejemplo sólo se emplee esta unidad durante un intervalo de tiempo reducido o únicamente en un pequeño grupo de operaciones será más adecuada su resolución por software.

Por otro lado, el uso de una FPGA posibilita la utilización de técnicas de auto-reconfiguración dinámica que permiten aumentar la densidad funcional (procesado por unidad de área y tiempo) de la aplicación. Esta característica consiste en permitir la reconfiguración dinámica de parte de la FPGA, de forma que un

área del dispositivo actúe como un coprocesador dinámico con capacidad para adoptar en tiempo de ejecución diferentes funcionalidades. El resto del dispositivo mantiene un comportamiento estático configurado como microprocesador embebido, cuya función es el control del propio proceso de reconfiguración dinámica y la ejecución por software de las tareas de menor coste computacional. El resultado más destacable es un notable ahorro del área ocupada, si lo comparamos con el resultado que se obtendría al implementar el mismo sistema sobre un dispositivo estático no reconfigurable. Las herramientas de desarrollo y la arquitectura interna de Virtex_II y superiores, han dado lugar a muchos trabajos de investigación relacionados con la auto-reconfiguración dinámica. Sin embargo, no es tan numerosa la producción científica relacionada con la familia Spartan 3 de bajo coste, debido a sus limitaciones internas y a la falta de herramientas software [2][3][8]. Los autores de este trabajo han presentado previamente varios artículos donde se propone un flujo de diseño y una arquitectura basada en Microblaze que permite la auto-reconfiguración dinámica para Spartan 3 (véase [2]) muy adecuada en el co-diseño software-hardware de sistemas complejos. La ventaja principal de esta propuesta es que permite reconfigurar la sección dinámica de la FPGA con tantos coprocesadores como sea necesario, estando su número únicamente limitado por el tamaño de la memoria externa necesaria para almacenar los bit-streams parciales (existen obviamente otras limitaciones adicionales relacionadas con el tamaño de los coprocesadores y su interficie).

El microprocesador embebido, además de controlar el proceso global, ejecuta las funciones asociadas, con la parte software del algoritmo, obtenidas tras el particionado software-hardware. El microprocesador podría incorporar una FPU interna, dependiendo de las restricciones y requerimientos en cuanto a área y velocidad de la aplicación. Sin embargo, dada la capacidad de auto-reconfiguración del sistema cabe la posibilidad de diseñar una FPU externa, cuya implementación podría llevarse a cabo en la sección dinámica de la FPGA justo antes de ejecutar la parte software del algoritmo. Durante la ejecución del programa las operaciones en coma flotante se trasladan a la FPU externa, que tras su procesado devuelve el resultado obtenido al microprocesador. La ventaja de esta solución es que el sistema puede beneficiarse de las prestaciones intrínsecas que se desprenden del uso de una FPU, sin necesidad de incrementar el área relacionada con la parte estática. El objetivo de este artículo es describir el diseño de esta arquitectura corroborando mediante resultados reales sus propiedades y características más relevantes.

En la sección 2 se muestra la arquitectura del sistema y las características principales que permiten la reconfiguración dinámica. La sección 3 describe la arquitectura interna de la unidad en coma flotante diseñada, y finalmente la sección 4 muestra los resultados obtenidos a nivel experimental con un ejemplo concreto basado en la resolución de una función exponencial.

2. Arquitectura del sistema

Spartan 3 es una FPGA de bajo coste que a diferencia de otras familias de Xilinx, tales como Virtex-2-4-5, no dispone de soporte hardware (ICAP, reconfiguración libre de glitches) y un flujo de diseño definido (PlanAhead Partial Reconfiiguration) para abordar la realización de sistemas auto-reconfigurables. Estas características particulares han hecho que la implementación de sistemas embebidos auto-reconfigurables sobre este tipo de FPGAs requiera de un esfuerzo notable, a pesar de las ventajas en términos de precio que presentan frente a las familias Virtex-2-4-5.

La arquitectura interna del sistema auto-reconfigurable sobre Spartan 3 está ampliamente documentada en las referencias [2][3], siendo esta sección una breve descripción acerca de sus principales características.

El sistema reconfigurable dispone de dos áreas o secciones claramente diferenciadas: la parte estática y la parte dinámica. En la sección estática se implementa el microprocesador Microblaze sin cache con 3 etapas en pipeline, el bus de comunicaciones, el controlador de memoria externa y el controlador de re-

configuración del sistema. La parte dinámica incorpora diversos coprocesadores hardware cuya implementación se produce de forma multiplexada en tiempo (cuando su funcionalidad es requerida por parte del microprocesador). Los "bit-streams" necesarios para la reconfiguración se encuentran almacenados en la memoria externa ubicada fuera de la FPGA, tal y como se muestra en la figura 1.

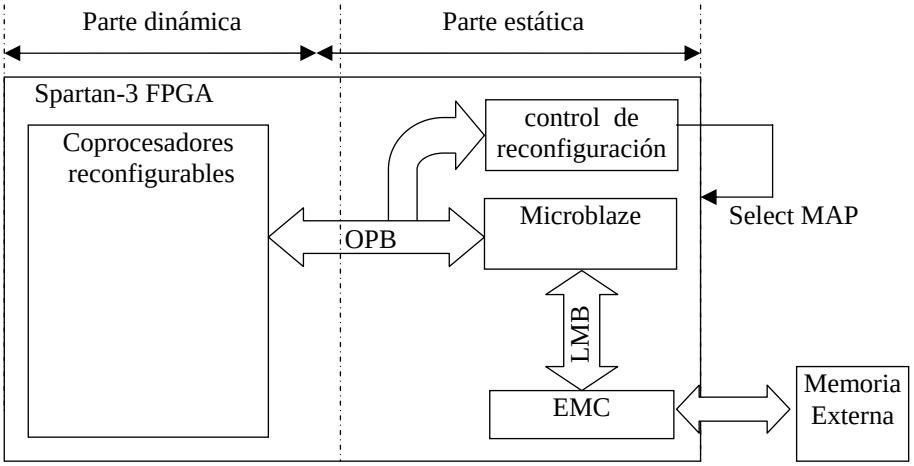

Figura 1: Arquitectura del sistema reconfigurable

Cuando es necesario mapear un coprocesador en la sección dinámica, Microblaze lee el bit-stream desde memoria externa y ejecuta la rutina de auto-reconfiguración. Esta rutina permite reconfigurar columna por columna, de forma que la memoria externa pueda usar pines de la FPGA que pertenezcan tanto a la sección estática como a la dinámica. El proceso consiste básicamente en leer desde memoria externa el bit-stream de un coprocesador, analizarlo para almacenar las tramas correspondientes a cada columna en memoria interna BRAM, y programar adecuadamente el controlador para configurar una de las columnas, repitiéndose el proceso de forma iterativa hasta la completa reconfiguración de todas las columnas correspondientes a la sección dinámica.

Al tiempo de resolución de la funcionalidad asociada con el coprocesador debe sumarse el tiempo necesario para su reconfiguración, que depende del tamaño del bit-stream, y que debe ser lo más pequeño posible en comparación con el primero. Como dato orientativo configurar una sección dinámica que ocupe el 66% de una Spartan 3 XC3S1500 puede realizarse en un tiempo de 10.2 ms a razón de 319.8 Mb/s [7].

3. Unidad de coma flotante

El diseño de la unidad de coma flotante se ha realizado atendiendo íntegramente al estándar IEEE 754 y según las premisas que a continuación se detallan:

- Las entidades que resuelven las operaciones básicas de suma, producto, cociente y raíz cuadrada han sido diseñadas como módulos independientes entre sí, con el objeto de permitir la implementación separada de cada una de ellas. Es posible, por tanto, generar "bit-streams" diferentes con unos u otros módulos para cargar en la parte dinámica aquella que se adapte mejor a las necesidades del algoritmo.

- Implementación de un control lógico que gestione el enrutamiento de los datos y que permita obtener operaciones derivadas de los bloques básicos tales como la resta, la potencia al cuadrado y la inversa.
- Inclusión de un registro en el que se almacena el último resultado obtenido para que pueda usarse como operador en la siguiente operación, con objeto de reducir la cantidad de ciclos de escritura-lectura.

Tal y como muestra la figura 2, el modelo funcional de la FPU dispone básicamente de tres entradas (además de las señales de reloj, inicio y reset), a saber: *op_a* y *op_b* que se corresponden con los operandos expresados en coma flotante de simple precisión (32 bits) y *operación* que representa directamente la operación a realizar (8 bits). La unidad dispone de un registro interno de 32 bits que memoriza el último resultado hallado. El diseño permite usar este registro en el caso de que uno de los operandos implicados corresponda al resultado de la operación anterior, lo que permite ahorrar un ciclo de escritura. Por otro lado, la FPU dispone de dos salidas donde se indica el resultado de la operación expresado en coma flotante de simple precisión (32 bits) y una señal adicional que indica que el cálculo ha concluido y que se encuentra disponible para ser leído por el microprocesador.

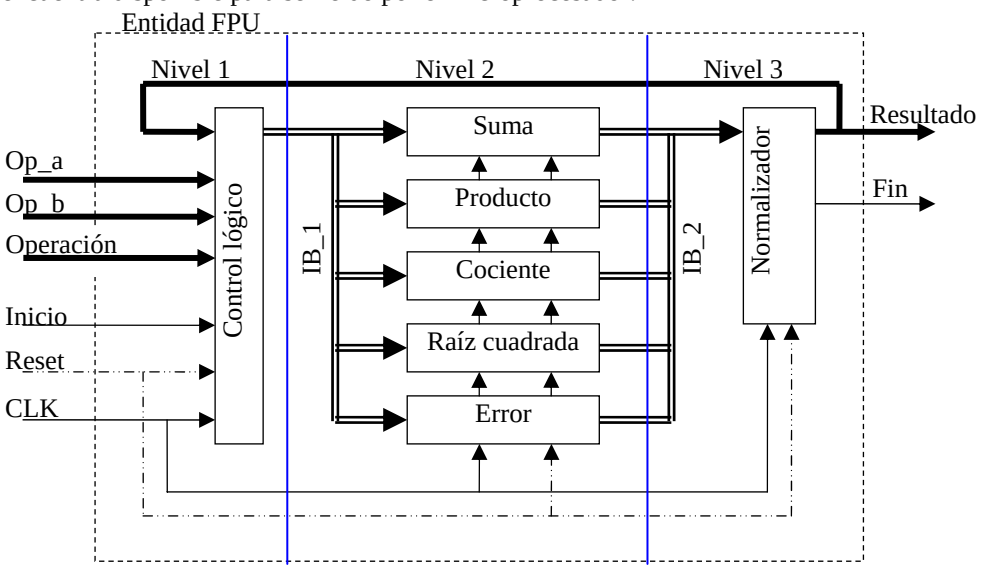

Figura 2: Estructura interna de la unidad de coma flotante

Estructuralmente la unidad en coma flotante dispone de un control lógico que se encarga de gestionar el "data path" y secuenciar su funcionamiento interno [4][5]. Además dispone de 4 bloques básicos de operación: suma, producto, cociente y raíz cuadrada, junto con un bloque especial que devuelve resultado "NaN" si la operación programada no corresponde a ninguna de las que la unidad realiza. Finalmente se dispone de un bloque que tiene la misión de normalizar el resultado de la operación, aplicar el redondeo y proporcionar el resultado obtenido. Pueden distinguirse dos tipos de operaciones diferentes:

- Operaciones básicas, que disponen de un bloque funcional que las resuelve, tales como la suma, el producto, el cociente y la raíz cuadrada.
- Operaciones no básicas, que se resuelven mediante el uso de uno de los bloques funcionales previa manipulación de los datos recibidos. Estas operaciones son: la resta (que se resuelve con el bloque suma previo cambio del signo del segundo operador), la inversa (se programa el valor

"1" en el primer operador del bloque división) y el cuadrado (se programa el mismo valor en los dos operadores del bloque producto). Es misión del control lógico gestionar estas operaciones.

Cabe destacar que cuando se operan valores en coma flotante el resultado se obtiene fuera de normas por la inclusión del bit implícito y de los bits de guarda para el posterior redondeo, siendo necesario procesar el resultado para que cumpla la norma de nuevo tras la operación [4]. El funcionamiento de la unidad se ha estructurado en tres niveles:

- Un primer nivel en el que se decodifica la operación programada, se detecta si se está solicitando operar con la memoria, se comprueba si alguno de los operadores corresponde con algún valor singular (concretamente: NaN, cero o infinito[1]) y, finalmente, se preparan los operandos para el siguiente nivel.
- En el segundo nivel se realiza la operación solicitada.
- En el tercer nivel se normaliza el valor y se aplica el redondeo "Nearest Even".

Los niveles están interconectados entre ellos mediante dos buses internos. El primero de ellos, IB_1, proporciona los operadores a los bloques básicos e incorpora señales de sincronización interna. El segundo, IB_2**,** traslada el resultado de la operación realizada hacia el bloque de normalización y señales de sincronización interna.

La FPU se comunica con el microprocesador mediante el bus OPB (On-Chip Peripheral Bus) [6]. Los dos bits más significativos de bus de direcciones, junto con la señal de escritura/lectura, se utilizan para transmitir los operandos, indicar la operación a realizar y recibir el resultado, tal y como muestra la tabla I.

Acción	MSB bus direcciones	Señal lectura-escritura RNW	Numero bits
Escritura operando A	00	0	32 bits
Escritura operando B	01	0	32 bits
Escritura código operación	10	0	8 bits
Lectura resultado	-- (No importa)	1	32 bits

Tabla 1: Direcciones del mapa de memoria asociadas al dispositivo

4. Resultados experimentales

Para las pruebas de funcionamiento se ha creado una rutina en C que resuelve la función exponencial mediante el algoritmo de CORDIC utilizando números en coma flotante de simple precisión. El sistema completo se ha implementado sobre la placa de desarrollo de AVNET que contiene una FPGA Spartan 3 XC3S2000. La frecuencia de reloj es de 40MHz. Concretamente el código realiza 100 iteraciones, donde en cada una de ellas se calcula la exponencial de un número aleatorio comprendido entre 0 y 10. La ejecución de este código se realiza de tres modos diferentes, a saber: en la primera se tiene únicamente el microprocesador en ausencia de FPU externa, donde la función exponencial se calcula ejecutando directamente el algoritmo de CORDIC (no se utilizan las funciones de la librería math.h). La segunda es exactamente igual a la primera, pero utilizando la unidad en coma flotante presentada en el apartado 3 como coprocesador implementado en la sección dinámica de la arquitectura y la tercera, a modo de comparación, se resuelve el algoritmo CORDIC usando la FPU interna del procesador. En la segunda prueba las operaciones en coma flotante se sustituyen por escrituras (operandos y código de operación) y lecturas

(obtención del resultado) en la dirección de memoria donde se mapea el coprocesador. Por otro lado, el software de desarrollo de sistemas embebidos suministrado por Xilinx (EDK) permite mediante un fichero Linker Script distribuir el código y los datos entre la memoria externa (SRAM) e interna (BRAM). Los resultados mostrados en la tabla 2 corresponden al valor medio del tiempo de ejecución de la función exponencial obtenido después de realizar las 100 iteraciones, según la configuración del Linker Script y de la implementación del sistema con o sin FPU. Nótese que la resolución de la función exponencial con FPU externa es cómo mínimo 5.5 veces más rápida que en ausencia de ésta, independientemente de la distribución del código y los datos en memoria SRAM o BRAM y que la FPU interna es 2 veces más rápida que la presentada en el apartado 3.

Distribución de memoria	Microprocesador sin FPU	Microprocesador con FPU externa	Microprocesador con FPU interna
Código en memoria interna, variables y constantes en externa.	1058 µs	186 µs	97 µs
Código, variables y constantes en interna.	948 µs	160 µs	83 µs

Tabla 2: Tiempos de ejecución del algoritmo de prueba

Módulo funcional	CLB Slices	Flip-flops	LUT's
Control lógico	131	80	264
Suma	225	146	435
Producto	108	172	236
Cociente	277	147	387
Raíz Cuadrada	201	153	399
Error	3	2	5
Normalización	91	76	192
Total	1036	776	1918

Tabla 3: Área ocupada por los distintos módulos funcionales de la FPU

La tabla 3 muestra el área que ocupa cada uno de los bloques descritos en la figura 2. Como puede observarse la FPU completa ocupa 1036 ó 835 Slices en ausencia del módulo que calcula la raiz cuadrada. La herramienta de desarrollo EDK permite configurar Microblaze con una FPU interna con capacidad para resolver las operaciones básicas de suma/resta, producto y cociente. Esta FPU ocuparía un tamaño aproximado de 704 Slices, formando parte de la sección estática del sistema. Implementando la FPU como un coprocesador externo permitiría utilizar este área como parte de la sección dinámica, ofreciendo la posibilidad de implementar coprocesadores más grandes o bien utilizar FPGAs de tamaño y coste menor.

5. Conclusiones

Los sistemas reconfigurables dinámicamente permiten, mediante la multiplexación en tiempo de diferentes coprocesadores, aumentar la densidad funcional de una aplicación sin necesidad de incrementar el

área necesaria para su implementación. Estas arquitecturas pueden en la actualidad implementarse de forma sistemática en FPGAs de bajo coste como Spartan 3, siendo muy adecuadas para el co-diseño software-hardware. El microprocesador de estos sistemas resuelve por software las tareas de menor coste computacional, que según el algoritmo implementado pueden contener operaciones en coma flotante. Para evitar la inclusión de una FPU interna al microprocesador, ahorrando espacio en la sección estática, puede diseñarse un coprocesador que realice las operaciones básicas que se describen en el estándar IEEE 754 configurable dentro de la sección dinámica de la arquitectura (FPU externa). Este articulo ha mostrado que está solución permite acelerar de forma sustancial la ejecución del algoritmo. Concretamente ha presentado resultados numéricos para una función exponencial concluyendo que para este caso particular la velocidad de procesado puede aumentarse en un factor superior a 5.5.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia de España, TEC2006-12365-C02-02.

Referencias

1. IEEE 754: Standard for Binary Floating-Point Arithmetic
2. Enrique Cantó, Francesc Fons and Mariano López,. Dep. of Electronics, Electrical & Automatics University Rovira i Virgili, Tarragona, Spain: SELF-RECOFIGURABLE EMBEDDED SYSTEMS ON SPARTAN-3. IEEE International Conference on Field Programmable Logic and Applications, pp- 571-574, Heidelberg, Germany (2008)
3. Enrique Cantó, Francesc Fons and Mariano López,. Dep. of Electronics, Electrical & Automatics University Rovira i Virgili, Tarragona, Spain: RECONFIGURABLE OPB COPROCESSORS FOR MICROBLAZE SELF-RECONFIGURABLE SOC MAPPED ON SPARTAN-3 FPGAs, 32ND annual Conference on the IEEE Electronics Society, pp. 3496-3501, Paris, France (2006)
4. Mario E. Vera Lizcano, Jaime Velasco Medina, Gustavo Ordoñez, Grupo de Bio-Nanoelectrónica, Escuela EIEE Universidad del Valle, Cali, Colombia. : DISEÑO DE OPERADORES ARITMÉTICOS EN PUNTO FLOTANTE USANDO FPGAs. XI Workshop IBERCHP, Salvador de Bahía, Brasil (2005)
5. T. Pollán, B. Martín y J. Ponde De León. Escuela Universitaria de Ingeniería Técnica Industrial de Zaragoza. Departamento de Ingeniería Electrónica y Comunicaciones. Universidad de Zaragoza. España: MÁQUINAS ALGORÍTMICAS COMO OPCIÓN DIDÁCTICA DE SISTEMAS DIGITALES COMPLEJOS
6. On-Chip Peripheral Bus Architecture Specifications Version 2.1 IBM Corporation
7. Enrique Cantó, Mariano Fons, Mariano López, Rafael Ramos, Dep. of Electronics, Electrical & Automatics University Rovira i Virgili, Tarragona, Spain “ACCELERATION OF COMPLEX ALGORITHMS ON A FAST RECONFIGURABLE EMBEDDED SYSTEM ON SPARTAN-3 19th The International Conference on Field Programmable Logic and Applications (FPL) Lugar: Praga, Republica Checa, (se publicará el 31 Agosto de 2009)”
8. I. Gonzalez, F. J. Gomez-Arribas and S. Lopez-Buedo, "HARDWARE-ACCELERATED SSH ON SELF-RECONFIGURABLE SYSTEMS", Escuela Politecnica Superior, Universidad Autonoma de Madrid, 2005.