Relación área-potencia en la implementación con aritmética distribuida de un Filtro FIR en FPGA

F. Angarita[1], J. Marin-Roig[1], E. Todorovich[2], E. Boemo[2]

[2] School of Engineering at Gandía, Universidad Politécnica de Valencia, Spain
faanpre@doctor.upv.es, jomara@eln.upv.es
[1] School of Engineering, Universidad Autónoma de Madrid, Spain
{etodorov,eduardo.boemo}@uam.es

Resumen

En este artículo se ha evaluado el efecto que tiene sobre el consumo de energía la implementación basada en aritmética distribuida con arquitecturas que procesan los datos con bits o dígitos en serie, o en paralelo. Para ello se ha aplicado dicha aritmética a un filtro FIR de 64 coeficientes que procesa los datos a una velocidad de 12.5 MHz. Se ha medido físicamente su consumo y se ha evaluado el mismo con la herramienta A-DyP. Como conclusión cabe resaltar que el consumo total es menor en la versión paralela. Sin embargo la relación área-consumo es más favorable en la implementación serie.

1. Introducción

Los filtros FIR se utilizan en la mayoría de las aplicaciones de tratamiento digital de la señal. Para su implementación en FPGA, se suele utilizar aritmética distribuida debido principalmente, a que los dispositivos poseen abundantes recursos de almacenamiento (*look-up tables*) distribuidos por todo el chip y a la facilidad de automatizar el proceso de implementación [1,6].

Originalmente, la aritmética distribuida fue propuesta como un método de procesado en serie de los datos con el fin de computar algoritmos basados en la suma de productos con un coste en área reducido. El mismo método permite aumentar la velocidad de cómputo de los datos, procesando dígitos (agrupaciones de bits), en serie o todo el dato en paralelo, a costa de un aumento del área de los circuitos que lo implementan.

En este artículo se ha evaluado el consumo de energía de un algoritmo (un filtro FIR) implementado con aritmética distribuida en función del tipo de procesado: serie, dígitos en serie o paralelo. El supuesto del que se parte es que, con los dispositivos FPGA de última generación, es posible aplicar procesado serie o paralelo en las aplicaciones que requieren velocidades moderadas (del orden de algunas decenas de MHz). Por tanto, si el objetivo de la implementación es reducir el consumo, es necesario constatar la influencia de cada una de las posibilidades arquitecturales.

El consumo medio dinámico del circuito está dado por:

$$P = \frac{1}{2} V_{dd}^2 f_{clk} \sum_i C_i E(sw)_i \qquad (1)$$

Donde C_i es la capacidad del nodo i del circuito, V_{dd} es la tensión de alimentación, que se supone constante en todo el circuito, $E(sw)$ es el número medio de transiciones por ciclo de reloj (*switching activity*), y f_{clk} es la frecuencia de reloj. El sumatorio se hace para todos los nodos del circuito.

Las implementaciones del filtro propuesto tienen diferentes tamaños de digito a frecuencia de muestreo constante. Entonces, a medida que se hace menor el tamaño de dígito, se debe aumentar la frecuencia de trabajo interna. Pero también, el área va a ser menor a medida que se reduce el tamaño de dígito.

Observando la Ec. 1, se puede pensar por un lado, que el consumo va a aumentar por el incremento en la frecuencia de reloj. Pero por otro lado, y en la misma medida que aumente la frecuencia de reloj, se va a reducir el área, y consecuentemente, la capacidad.

Lo que se va a ver en este artículo, es cual de los dos efectos va a predominar: menor área con menos consumo o mayor frecuencia con un consumo más alto.

La organización del artículo es la siguiente. En la sección 2 se introduce la base teórica de la aritmética distribuida aplicada a los filtros FIR. En la sección 3 se presenta el experimento realizado para evaluar el consumo en función de la arquitectura. En la sección 4 se detallan los resultados. Finalmente, se resumen las conclusiones del trabajo.

2. Aritmética Distribuida

Un filtro FIR con coeficientes constantes de M etapas (coeficientes) está caracterizado por la siguiente ecuación:

$$y[n] = \sum_{n=0}^{M-1} h[n] \cdot x[n-k] \qquad (2)$$

Una manera eficiente de implementar este filtro sobre una FPGA es utilizar aritmética distribuida [4], ya que ésta se basa en almacenar los productos parciales de las multiplicaciones en tablas (*look-up tables*), evitando así el uso de multiplicadores, que representan una reducción de la velocidad y un incremento del área.

La aritmética distribuida se basa en la computación de una suma de productos.

$$y = \overline{a} \cdot \overline{x} = \sum_{i=0}^{N} a_i \cdot x_i \qquad (3)$$

$$y = \sum_{i=1}^{N} a_i \cdot \left[-x_{i_0} + \sum_{K=1}^{W_d-1} x_{i_K} \cdot 2^{-K} \right] \qquad (4)$$

$$y = -\sum_{i=1}^{N} a_i \cdot x_{i_0} + \sum_{K=1}^{W_d-1} \left[\sum_{i=1}^{N} a_{i_K} \cdot x_{i_K} \right] \cdot 2^{-K} \qquad (5)$$

$$y = -F_0(x_{10},..,x_{N0}) + \sum_{K=1}^{W_d-1} F_K(x_{1K},..,x_{NK}) \cdot 2^{-K} \qquad (6)$$

F_K sólo puede tener un número finito de valores (2^N) y, por lo tanto, puede ser computada y almacenada en una memoria.

La implementación de la suma de productos se ilustra en la figura 1. La memoria ROM es de tamaño 2^N x W_{ROM}, donde $W_{ROM} \leq W_C + \log_2(N)$, siendo N el número de operaciones (sumas). El tiempo de computación es de W_D ciclos de reloj. Esta memoria se implementa en la FPGA de manera distribuida, utilizando las LUTs (*look-up tables*).

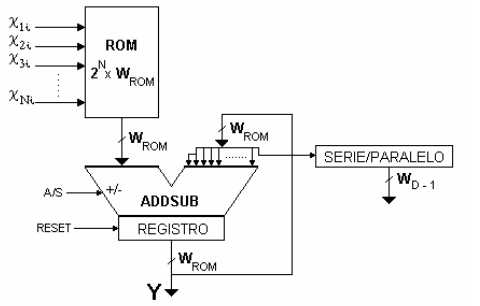

Figura 1. Suma de productos con aritmética distribuida.

Las operaciones a realizar son: una secuencia de búsquedas en tablas, sumas, restas y desplazamientos de la señal de entrada. Todas estas operaciones se pueden implementar en una FPGA de forma eficiente.

El tamaño de la tabla puede llegar a ser muy grande. Una manera de reducir este tamaño es fraccionar la memoria. Para N/k particiones de k bits se pasa de necesitar una memoria de tamaño 2^N a N/k memorias de tamaño 2^K y, además, hay que añadir un acumulador multioperando.

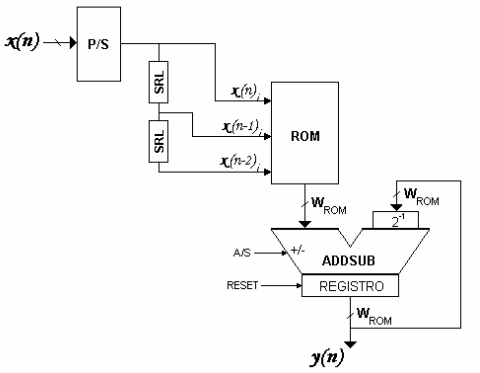

Figura 2. Filtro FIR Serie de 3 etapas.

Para mejorar la velocidad se puede aumentar el número de bits por entrada. De este modo se trabaja con *Digit-Serial* y tamaño de dígito D. En

este caso se obtiene una salida cada W_D/D ciclos de reloj. La máxima velocidad se obtiene cuando D es igual a W_D, que es el caso paralelo.

En la Figura 2 se muestra un ejemplo de la implementación de un filtro FIR de 3 etapas usando aritmética distribuida.

3. Experimento

Las medidas de consumo se hicieron sobre diferentes implementaciones de un filtro FIR basado en aritmética distribuida, paso bajo, de 64 coeficientes, con una frecuencia normalizada de corte de 2/3. El tamaño de los coeficientes es de 6 bits, y para la entrada y salida la cuantificación es de 8 bits. La frecuencia de muestreo, no normalizada f_s, es siempre de 12.5 MHz. Las implementaciones difieren en el tamaño del dígito y en la frecuencia de trabajo, a medida que se aumenta el dígito, la frecuencia de trabajo disminuye, para así mantener la frecuencia de muestreo de 12.5 MHz. La implementación en paralelo trabaja a una frecuencia de 12.5 MHz y la serie a una frecuencia 8 veces mayor, o sea 100 MHz.

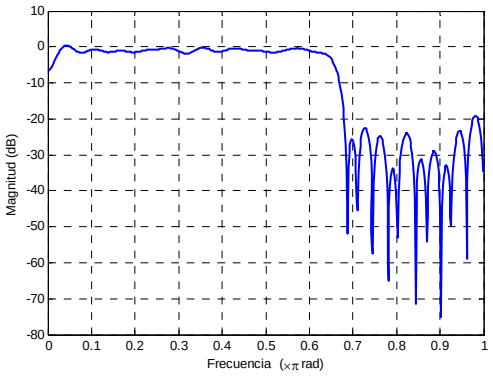

Figura 3. Respuesta en frecuencia del filtro.

Los filtros sobre los que se hicieron las medidas, son los presentados en [1], que han sido diseñados utilizando el emplazado relativo (RPM) para minimizar el efecto sobre la velocidad, de los caminos críticos y aprovechar los recursos al máximo, evitando el uso de hardware innecesario.

Las medidas de consumo total se hicieron físicamente. Las estimaciones y el análisis de consumo, mediante la herramienta A-DyP presentada en [5]. Las medidas físicas se realizaron, implementado los filtros en un dispositivo Virtex-E xcv400E -8 de Xilinx montado en la placa de desarrollo HW AFX PQ240-110 del mismo fabricante. Las señales de entrada y el reloj se generaron mediante el modulo generador de patrones del equipo TLA-715 de Tektronix. Con este mismo equipo, pero utilizando el modulo de analizador lógico, se verificó la salida de cada filtro, para confirmar su correcto funcionamiento. Las medidas tensión y de corriente se realizaron con dos multimetros con una resolución en la medida de tensión de 1 mV y en corriente de 1 mA.

La Figura 4 muestra el diagrama de conexión que se utilizó para el experimento. Las medidas de tensión y corriente se realizaron sobre el núcleo de la FPGA, sin incluir los puertos de entrada salida.

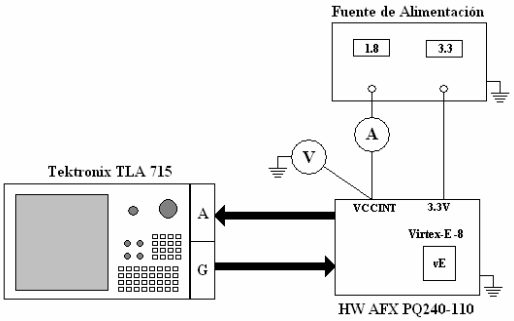

Figura 4. Montaje del experimento.

Un factor importante a tener en cuenta a la hora de realizar las medidas, es que la tensión en el núcleo debe ser la misma para todos los casos. Con esto se garantiza el mismo entorno de experimento para todos los casos, y hace que los resultados sean comparables.

4. Resultados

Dependiendo de la aplicación es posible elegir entre los diferentes tipos de implementación del filtro, teniendo en cuenta el consumo y el área que ocupan a la misma frecuencia de muestreo. Como se ve en la Figura 5, el área es inversamente proporcional al consumo: a mayor área, menor consumo. Si la prioridad en el diseño es el área, se debería usar la implementación serie, ya que es casi W_d (en este caso W_d=8) veces menor que la implementación paralela. Pero, si al contrario, la prioridad es el consumo de potencia o energía por

operación (EPO), la implementación en paralelo es la ideal. La EPO en el caso serie es casi el doble que para la versión paralela.

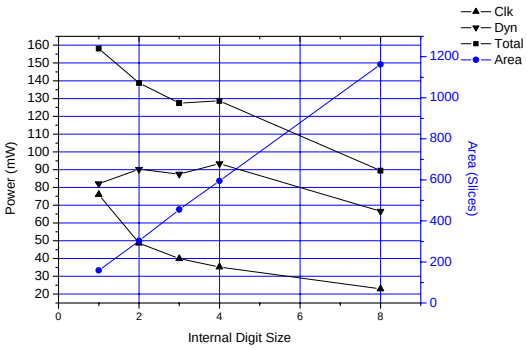

Figura 5. Relación área – potencia del filtro FIR

Es interesante observar por separado la potencia de sincronización del resto. En la gráfica se observa que la potencia dinámica de la lógica varía muy poco. Se incrementa en las versiones donde se serializa debido a la lógica adicional. Se puede asociar una cantidad de energía al cómputo de datos [3]. Al realizarse la misma cantidad de computación, se necesita la misma cantidad de energía. En este punto, es importante aclarar que la segmentación minimiza, en todas las implementaciones, la actividad espuria.

En cuanto a la potencia de sincronización, la versión serie consume poco más de 3 veces mas potencia que la versión paralela. Del mismo modo que en la segmentación [2], se necesita potencia adicional para gestionar (serializar, en este caso) los datos. La potencia total, combinando sincronización y potencia dinámica, crece de tal forma que para la versión serie no llega a ser el doble que en la versión paralela. Entonces, si se optimiza área y potencia, conviene usar la versión serie (Figura 6).

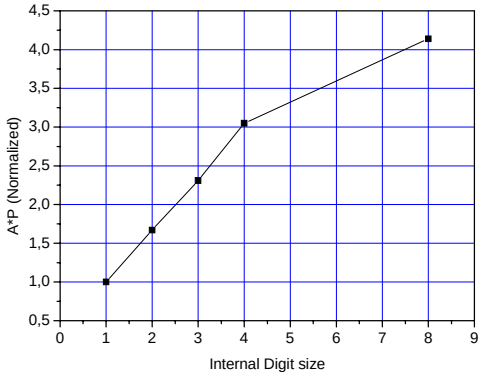

Figura 6. Producto área-potencia.

La herramienta A-DyP para la estimación del consumo medio, permite construir los mapas de consumo medio de los circuitos estudiados. En la Figura 7 y 8 se ven los mapas para las versiones serie y paralela respectivamente. Se ve gráficamente lo expuesto en esta sección: la reducción de área va acompañada de un aumento de consumo (eje z). Se observa adicionalmente, que estos mapas permiten identificar los elementos del diseño que más energía disipan. De esta manera se puede aplicar alguna técnica de diseño para bajo consumo adecuada [2]. El caso serie presenta un pico de consumo en la zona donde se encuentra la memoria distribuida. En el caso paralelo, el pico de consumo está en el sumador-acumulador a la salida del filtro.

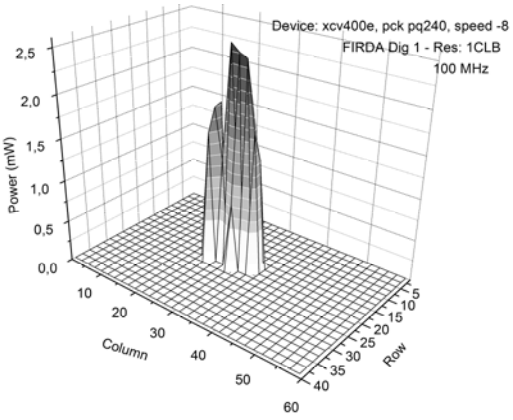

Figura 7. Mapa de consumo para la versión serie.

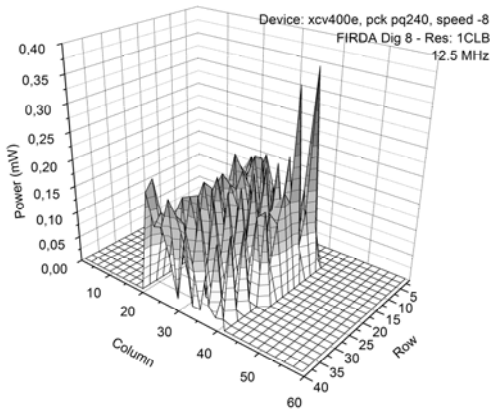

Figura 8. Mapa de consumo para la versión parelela.

5. Conclusiones

En este artículo se ha evaluado desde el punto de vista de consumo de energía, el efecto que tiene en un filtro digital basado en aritmética distribuida su implementación con arquitecturas procesando los bits o dígitos en serie y en paralelo. Se ha propuesto un experimento, se ha medido físicamente su consumo y se ha evaluado el mismo con la herramienta A-DyP. Se concluye que el consumo total va a ser mayor en la versión serie. Pero esto se debe solamente al aumento de la potencia de sincronización. La potencia de la lógica es prácticamente la misma en todas las versiones porque la cantidad de energía necesaria para este mismo cómputo, se haga en serie o en paralelo, es la misma. Entonces, el factor de aumento de potencia es menor que el factor de reducción de área a medida que se serializa. Por lo tanto la relación área-consumo es más favorable en la implementación serie. Cabe destacar que estos resultados son extrapolables a cualquier algoritmo implementable con aritmética distribuida.

Referencias

[1] Xilinx: Distributed Arithmetic FIR Filter v 9.0 LogicCore Product Specification. DS-240, (May. 2004).

[2] Angarita F.E., Canet M.J., Valls J., Vicedo F: Implementación de un Core-IP "Filtro FIR Basado en Aritmética Distribuida", III Jornadas de Computación Reconfigurable y FPGAs - JCRA 2003. Madrid, (Sep. 2003).

[3] Stanley A. White: Applications of Distributed Arithmetic to Digital Signal Processing: A tutorial Review, IEEE ASSP Magazine, (Jul. 1989).

[4] Todorovich, E., M. Gilabert, G. Sutter, S. Lopez-Buedo, and E. Boemo: A Tool for Activity Estimation in FPGAs, Lecture Notes in Computer Science, Vol. 2438. Springer-Verlag, Berlin Heidelberg, p340-349, (2002).

[5] Richard P. Feynman, "Feynman Lectures on Computation", Perseus Books, 1996.

[6] Gustavo Sutter, "Aportes a la Reducción de Consumo en FPGAs", Ph. D. Thesis, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, April 2005.