ARQUITECTURA FPGA PARAMETRIZABLE PARA CONVOLUCIÓN DE IMÁGENES

J.C. Moctezuma Eugenio, M. O. Arias Estrada, R. Cumplido Parra

Instituto Nacional de Astrofísica, Óptica y Electrónica,
Departamento de Ciencias Computacionales, Puebla-México

jcmoctezuma@ccc.inaoep.mx

ABSTRACT

En este trabajo se realiza una arquitectura FPGA para realizar el proceso de convolución en imágenes, teniendo como característica principal que la arquitectura puede ser parametrizable por el usuario en tiempo de diseño.

La arquitectura se basa en módulos de convolución que trabajan en paralelo con el objetivo de disminuir el número de accesos a memoria, en donde se encuentra almacenada la imagen. De esta forma se logra aumentar el rendimiento y la velocidad de procesamiento.

Se usa como plataforma de diseño la herramienta Xilinx System Generator y Simulink de MATLAB. Por otro lado, se emplean técnicas de enmascaramiento de subsistemas y programación dinámica de bloques, propias de Simulink, para poder realizar la reconfiguración de la arquitectura de acuerdo a los parámetros seleccionados por el usuario. La validación del diseño se realiza mediante la técnica llamada "Hardware in the loop". Finalmente se realiza un análisis de tiempo y de consumo de recursos hardware para algunas configuraciones de la arquitectura.

1. INTRODUCTION

La convolución es una de las herramientas matemáticas más utilizadas en el procesamiento de imágenes, un ejemplo de ello es su utilización para realizar el filtrado de imágenes tanto en el dominio espacial como en el dominio de la frecuencia [2].

Las tecnologías reconfigurables tales como los FPGAs, ofrecen la posibilidad de implementar arquitecturas con diversos niveles de pipeline, y un alto grado de paralelismo. Además, los FPGAs permiten la implementación de sistemas en un solo chip (SoC), creando con esto arquitecturas digitales dedicadas. Debido a estas características se obtiene un gran aumento en la velocidad de procesamiento de datos, lo que permite implementaciones de aplicaciones en tiempo real [1].

El procesamiento de imágenes en tiempo real se puede realizar mediante el uso de hardware especializado. En este trabajo se propone una arquitectura hardware para el filtrado de imágenes mediante el uso de la convolución en el dominio del espacio. Además se propone una forma de reconfigurar la arquitectura para que se adapte a distintos tamaños de imagen y de máscaras, así como representar cualquier tipo de coeficientes para la máscara, todo esto en un ambiente esquemático, fácil de usar y amigable para el usuario como lo es Simulink de MATLAB.

Este trabajo esta dividido de la siguiente forma: en la sección 2 se hace una breve revisión teórica de las herramientas y técnicas de diseño usadas, la sección 3 explica en detalle el desarrollo y funcionamiento de la arquitectura FPGA, en la sección 4 se analizan algunos de los resultados obtenidos y también se ve la forma de validación por medio de "Hardware in the loop"; en la sección 5 se hace un estudio de la estimación de recursos hardware usados y un análisis de tiempo; finalmente en la sección 6 se presentan las conclusiones

2. HERRAMIENTAS Y TÉCNICAS DE DISEÑO

En esta sección se realiza una breve descripción de las herramientas utilizadas así como de algunas de las principales técnicas de diseño que se usaron para lograr la parametrización de la arquitectura.

2.1. Xilins System Generator

Xilinx System Generator (XSG) es un ambiente de diseño integrado (IDE) a nivel sistema para FPGAs, que utiliza a Simulink, como entorno de desarrollo y se hace presente en forma de blockset. Posee un flujo de diseño integrado, que puede generar directamente el archivo de configuración bitstream necesario para la programación del FPGA.

Una de las características más importantes de System Generator es que posee abstracción aritmética, es decir, trabaja con representaciones en punto fijo con una precisión arbitraria, incluyendo varios métodos para la

ISBN-13 978-968-7938-03-5

cuantización y el sobreflujo. Los bloques en XSG pueden operar tanto con valores booleanos como con valores arbitrarios en punto fijo. Esto le da una mejor aproximación a la implementación hardware. En contraste Simulink trabaja con números de punto flotante de doble precisión [3].

2.2. Generación dinámica de bloques

Aquellos que hayan trabajado con Simulink, estarán acostumbrados a trabajar con él en forma esquemática, arrastrando y conectando bloques. Pero existe otra forma muy poco usada de construir estos mismos modelos mediante programación dinámica, esto se logra por medio de la API (Application Programming Interface) de MATLAB, la cual soporta instanciación de bloques y señales, además de configuración, realización y eliminación de bloques, entre otros métodos de construcción [4].

Usando esta API, es posible escribir scripts de MATLAB que generen modelos completos de Simulink y por tanto, también modelos de System Generator. Así, este tipo de técnica permite modificar la estructura de un subsistema en respuesta a los cambios realizados a ciertos parámetros mediante cuadros de diálogo.

3. DISEÑO DE LA ARQUITECTURA

La idea central de la arquitectura FPGA para realizar la convolución, es tener múltiples "módulos de convolución" funcionando en paralelo, con el objetivo de disminuir el número de accesos a memoria. Cada uno de estos módulos procesa el mismo píxel en paralelo, pero con diferentes coeficientes de la máscara, es decir, cada módulo juega el papel de una ventana de convolución pero en diferentes posiciones. En la figura 1 se muestra esta idea, en donde el mismo píxel se procesa para diferentes ventanas (módulos de convolución).

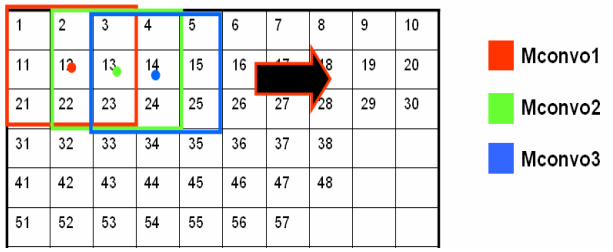


Figura 1. Módulos de convolución trabajando en paralelo.

En la arquitectura propuesta para este trabajo, el número de módulos de convolución será igual al número de columnas de la máscara. Este esquema permite mejorar el rendimiento de la arquitectura en un factor de 'n', donde n es el número de columnas de la máscara. Los módulos de convolución están compuestos únicamente de un multiplicador y un acumulador. La figura 2 muestra la arquitectura general.

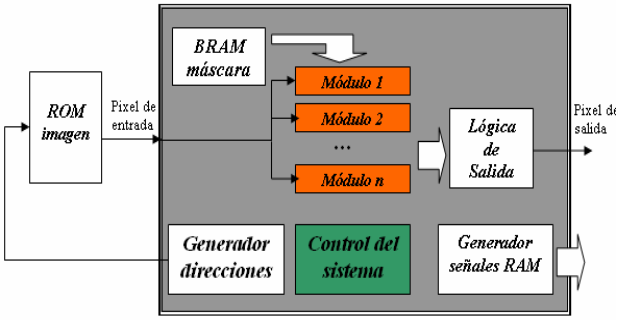


Figura 2. Arquitectura propuesta para la convolución.

El bloque de generador de señales RAM, manda las señales necesarias para el almacenamiento de los píxeles de salida en una memoria SRAM externa. El bloque "BRAM máscara" contienen los coeficientes de la máscara aplicada, la Lógica de salida filtra el píxel adecuado proveniente de los módulos de convolución hacia la imagen de salida.

Una vez hecha la arquitectura, se realiza la implementación en System Generator. Después se realiza un subsistema de todo el diseño y se enmascara, es decir, se le asocia un cuadro de diálogo parametrizable, quedando como se muestra en las figuras 3 y 4.

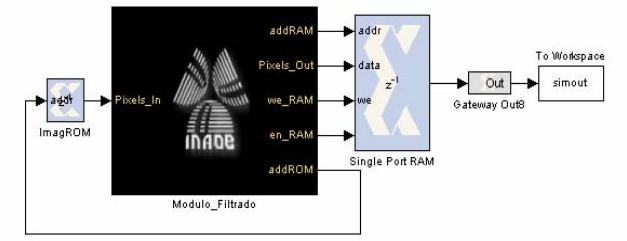


Figura 3. Arquitectura propuesta para la convolución, vista como una caja negra.

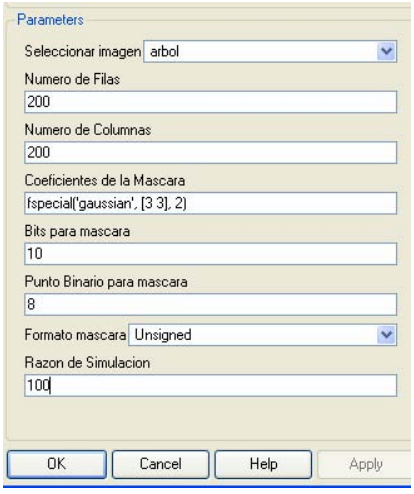


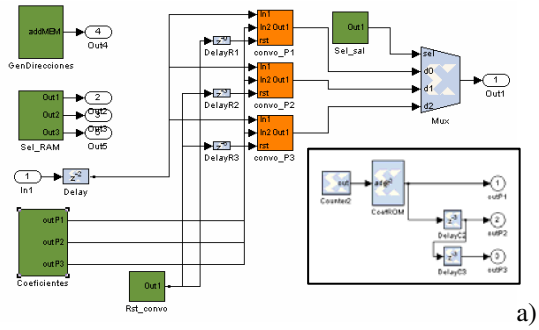
Figura 4. Máscara asociada al subsistema.

ISBN-13 978-968-7938-03-5

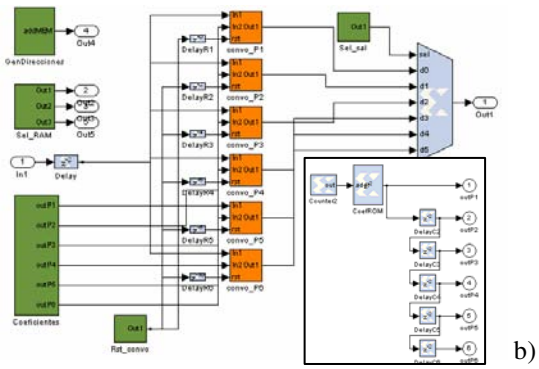
Como se puede observar en la figura 4, los parámetros que pueden configurarse son: la imagen a procesar, el tamaño de la imagen de entrada, el tamaño de la máscara así como la el valor de sus coeficientes, además de la forma de representarlos, es decir, el número de bits, la posición del punto binario y si serán signados o no signados, ya que la arquitectura soporta trabajar con números en punto fijo.

Cuando cualquiera de estos parámetros cambia, se ejecuta un script de MATLAB que realiza la programación dinámica de los bloques, haciendo uso del API tal y como se mencionó en la sección 2.

En la figura 5 se muestran dos ejemplos de esta programación dinámica de bloques seleccionando diferentes tamaños de máscara. Ambas arquitecturas se generan en forma automática con tan solo apretar un botón y están listas para ser sintetizadas en un FPGA.



a)



b)

Figura 5. Ejemplos de programación dinámica de bloques.
a) máscara de 3x3, b) máscara de 2x6

4. RESULTADOS

Una vez obtenida la imagen de salida, se debe tener alguna forma de medir el resultado de la arquitectura. Para esto se programó un script en MATLAB, el cual realiza la convolución pero a nivel software usando algunas de las funciones que proporciona el Image Processing Toolbox, y una vez que se tiene la imagen software, se compara con la imagen "hardware" (que arroja la arquitectura hecha en XSG) mediante alguna métrica. La métrica que se usa para poder comparar las dos imágenes es la siguiente:

$$D = \text{pixels\_equ} / \text{pixels\_total} \quad (1)$$

Donde 'pixels_equ' es el número de píxeles que son iguales con cierto nivel de tolerancia en ambas imágenes, mientras que 'pixels_total' es el número de píxeles de la imagen. De esta manera se obtiene un coeficiente de correlación D que toma valores entre 0 y 1.

Para realizar validación hardware, es decir, el diseño implementado en una tarjeta FPGA, se hace mediante la técnica "Hardware in the loop". Esta técnica consiste en compilar en diseño hecho en XSG, creando así el archivo de configuración bitstream, el cual es asociado a un bloque, de tal forma que cuando el diseño se simula en Simulink, los resultados para la parte compilada son calculados en hardware y son regresados a la computadora por medio de la conexión JTAG. Esto permite probar el diseño a nivel hardware pero corriendo la simulación desde Simulink. En la figura 6 se observa el modelo que se usa para realizar la co-simulación hardware.

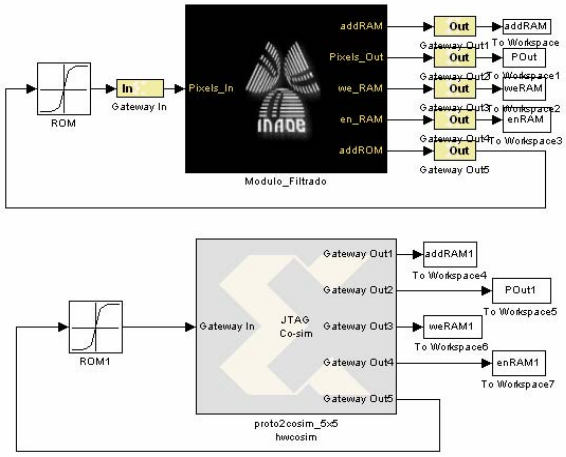
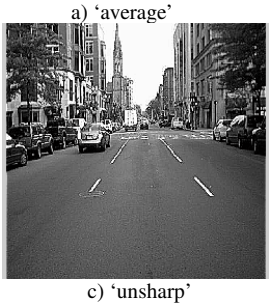


Fig. 6. Bloque de co-simulación hardware (abajo).

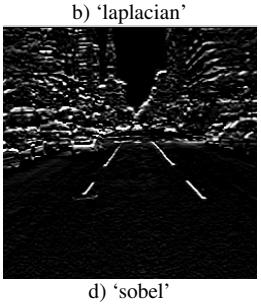
En la figura 7 se muestran algunos de los resultados de la co-simulación hardware al aplicar distintos tipos de filtros a la imagen "calle_boston".

a) 'average'

b) 'laplacian'



c) 'unsharp'



d) 'sobel'

ISBN-13 978-968-7938-03-5

e) 'brightness'

f) 'motion'

Figura 7. Ejemplos de diferentes filtros aplicados.

En la siguiente tabla se muestran algunos de los resultados de comparar las imágenes software contra hardware. Todas las pruebas de esta tabla se hicieron con una imagen de tamaño 200x200 y máscaras de tamaño 3x3 y 5x5, con un filtro del tipo gaussiano con los siguientes coeficientes:

0.1019 0.1154 0.1019
0.1154 0.1308 0.1154
0.1019 0.1154 0.1019

0.0232 0.0338 0.0383 0.0338 0.0232
0.0338 0.0492 0.0558 0.0492 0.0338
0.0383 0.0558 0.0632 0.0558 0.0383
0.0338 0.0492 0.0558 0.0492 0.0338
0.0232 0.0338 0.0383 0.0338 0.0232

Tamaño máscara	*Formato coeficientes*	*Coeficiente de correlación D*
3x3	UFix_8_8	1.00
3x3	UFix_8_6	1.00
3x3	UFix_8_4	0.09
5x5	UFix_8_8	0.98
5x5	UFix_8_6	0.14
5x5	UFix_8_4	0.05

Tabla 1. Comparación de las imágenes SW y HW para un umbral de 3.

En cuanto al análisis para la estimación de recursos hardware se usó el bloque "Resource Estimator". Estas estimaciones son hechas invocando a los "estimadores de bloques específicos", los cuales calculan los recursos hardware que se necesitan para implementar a cada uno de los bloques de Xilinx. La estimación de estos recursos se realiza para siete campos: Slices, Look-up Tables (LUTs), FlipFlops (FFs), Bloques de memoria (BRAM), multiplicadores 18x18, Buffers Tri-estado y puertos de entrada/salida (IOBs).

En la tabla 2 se muestran algunos de los resultados de estimación de recursos hardware para distintos tamaños de imágenes y para una máscara de 3x3. Por otro lado también se muestra la frecuencia máxima de cada modelo.

Para obtener la frecuencia máxima de operación se ulitizó la herramienta "Timming Analysis", la cual muestra tanto en forma gráfica como en forma textual, todos los paths junto con un análisis de tiempo. También puede mostrar en una sola gráfica todos los delays que se encuentran en la arquitectura así como el camino de datos de los bloques en System Generator que los provocan.

Tamaño imagen	*Slices*	*LUTs*	*FFs*	*Frecuencia máxima MHz*
100x100	353	638	306	81.4
200x200	359	652	313	
300x300	367	661	317	
300x500	370	667	320	
480x600	373	674	324	
Spartan3E	**4,656**	**9,312**	**9,312**	

Tabla 2. Recursos hardware y análisis de tiempo.

5. CONCLUSIONES

La programación dinámica de bloques es una buena alternativa cuando se requieren hacer arquitecturas FPGA parametrizables por el usuario. Esto aplicado al procesamiento de imágenes aceleran el proceso de prueba de dichas arquitecturas, ya que no se necesita implementar el diseño en el FPGA cada vez que se necesite probar una imagen nueva, esto es posible por la propiedad 'Bit & Cycle True' que poseen los modelos en System Generator, esto significa que los resultados que se obtengan en la simulación serán básicamente los mismos que en la implementación hardware.

Se debe establecer un compromiso entre el nivel de paralelismo de la arquitectura y los recursos hardware, es decir, hay que establecer un compromiso entre Costo y Velocidad dependiendo de los recursos con que se cuente.

Se logra reducir en un factor de 'n' el número de accesos a memoria, donde n es el número de columnas de la máscara. Se puede reducir aún más los accesos si se realizan arreglos matriciales de módulos de convolución.

Los recursos hardware no cambian en gran medida cuando el tamaño de la imagen a procesar crece o disminuye, siempre y cuando el tamaño de la máscara permanezca constante.

La herramienta System Generator es muy flexible y versátil para realizar arquitecturas FPGA, ya que posee todo un conjunto de herramientas que permiten analizar una gran variedad de características del diseño, en este trabajo se presentaron principalmente tres herramientas: co-simulación hardware, estimación de recursos y análisis de tiempo.

6. REFERENCIAS

[1] "Arquitectura FPGA para Filtrado de Imágenes en Tiempo Real", Juan C. Tejeda Y., Miguel Arias, INAOE, 2001
[2] "Digital Image Processing", Second Edition, Rafael C. Gonzalez, Richard E. Woods, Prentice Hall, 2002
[3] Xilinx System Generator User's Guide, www.xilinx.com
[4] "SIMULINK User Guide, MATLAB", Mathworks, 2006.
[5] "Spartan-3E Starter Kit Board User Guide", www.xilinx.com
[6] "FPGA Spartan-3E Datasheet", www.xilinx.com

ISBN-13 978-968-7938-03-5