Viabilidad de acelerar la migración sísmica 2D usando un procesador específico implementado sobre un FPGA

The feasibility of speeding up 2D seismic migration using a specific processor on an FPGA

Sergio Alberto Abreo Carrillo[1] y Ana Beatriz Ramírez Silva[2]

RESUMEN

Este artículo describe el estado actual del proceso de migración 2D, desde el punto de vista del *software* y del *hardware*. Así mismo, presenta la actualidad del procesamiento específico usando FPGA, para luego poder concluir la viabilidad de implementar completamente el proceso de migración sísmica 2D sobre un FPGA a través de un procesador específico. Con el fin de obtener una visión global del estado actual del procesamiento específico usando FPGA, se usaron trabajos que muestran el desempeño de éste, en diversas áreas del conocimiento. Adicionalmente, como el proceso de migración sísmica 2D trabaja con datos en formato de punto flotante, en este artículo se presentan varios trabajos que muestran las tendencias en operaciones de punto flotante, tanto en procesadores de propósito general como específico; la información en él contenida permite concluir de manera general que los FPGA en esta área tienen un gran futuro, pues las empresas de la industria del petróleo han comenzado a desarrollar sus propias herramientas con el fin de optimizar aún más los procesos relacionados con la exploración de campos.

Palabras clave: optimización de camino de datos, FPGA, optimización de *hardware*, migración en tiempo de Kirchhoff.

ABSTRACT

This paper was aimed at describing the state of the art regarding 2D migration from a software and hardware perspective. It also gives the current state of specific processing using field programmable gate array (FPGA) and then concludes with the feasibility of fully implementing 2D seismic migration on a FPGA via a specific processor. Work was used showing performance in different areas of knowledge to gain an overview of the current state of specific processing using FPGAs. As 2D seismic migration employs floating-point data, this article thus compiles several papers showing trends in floating-point operations in both general and specific processors. The information presented in this article led to concluding that FPGAs have a promising future in this area due to oil industry companies having begun to develop their own tools aimed at further optimising field exploration.

Keywords: data path optimisation, field programmable gate array (FPGA), hardware optimisation, Kirchhoff time migration.

Recibido: abril 12 de 2009
Aceptado: marzo 15 de 2010

Introducción

Hoy en día las empresas relacionadas con la industria del petróleo usan clusters de computadores para calcular la migración 2D y 3D debido al gran costo computacional ocasionado por los elevados volúmenes de datos que se deben procesar. Para hacernos una idea de este costo computacional, normalmente el proceso de migración en 3D "gasta por lo menos un mes" para poder obtener los resultados finales usando un clúster con "una capacidad de procesamiento de 1GFLOPS". Esta demora se debe a que típicamente los volúmenes de datos que se procesan son del orden de gigabytes (He *et ál*., 2004).

El principal problema de estas empresas consiste en que el proceso de migración se realiza cada vez que se explora una nueva área en búsqueda de hidrocarburos (llega un nuevo *set* de datos), debido a que con las imágenes que dicho proceso entrega el equipo de geólogos realiza las respectivas interpretaciones del subsuelo en búsqueda del petróleo. Además hay ocasiones en las que es necesario migrar el mismo *set* de datos más de una vez, para poder resaltar algunos detalles que previamente no aparecen. En pocas palabras, este proceso se realiza frecuentemente. Las empresas que trabajan en la industria del petróleo siempre están buscando reducir estos tiempos de procesamiento para poder agilizar y aumentar sus exploraciones anuales. Una muestra de ello son sus considerables inversiones económicas en clusters cada vez más grandes que a su vez requieren de sistemas de enfriamiento y su uso acarrea elevados consumos de energía (Panetta *et ál*., 2007). Por esta razón, la idea de tener un sistema de procesamiento específico que ayude a acelerar este proceso sin ocupar mucho espacio y cuyo consumo de potencia sea bajo, empezó a tener acogida dentro de la industria del petróleo. Por otra parte, el crecimiento de los procesadores específicos ha estado vinculado con el desarrollo de los FPGA. Una muestra de ello son los trabajos publicados sobre procesadores específicos, implementados en FPGA, en los que se han obtenido índices de aceleración muy buenos, comparados

[1] Ingeniero electrónico, Universidad Distrital Francisco José de Caldas, Colombia. Estudiante M.Sc., en Ingeniería Electrónica, Universidad Industrial de Santander, Colombia. abreosergio@gmail.com, mae3t24@uis.edu.co
[2] Ingeniera electrónica, Universidad Industrial de Santander, Colombia. M.Sc., in Electrical Engineering, Universidad de Puerto Rico, Puerto Rico. Profesora, Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones e integrante del Grupo de Investigación en Conectividad y Procesado de Señales - CPS, Universidad Industrial de Santander, anaberam@uis.edu.co.

con los procesadores de propósito general. Estos trabajos abarcan campos como: análisis de secuencias genéticas (Puttegowda *et ál*., 2003; Bogdán *et ál*., 2008; Hoang y Lopresti, 1993), filtrado digital (Tessier y Burleson, 2001; Yamada y Nishihara 2001; Wang y Shen, 2008), criptografía (Patterson, 2000; Anghelescu *et ál.,* 2008a; Anghelescu *et ál.,* 2008), filtrado de paquetes de red (Sinnappan y Hazelhurst, 2001; Cho y Mangione-Smith, 2008, 2005), reconocimiento automático de objetos (Jean *et ál*., 1999; Villasenor *et ál*., 1996), identificación de patrones (Baker y Prasanna, 2004), entre otros. Los resultados de estas investigaciones muestran claramente que cuando se cumplen las condiciones especificadas en la sección "Procesamiento específico usando FPGAs" el procesador específico siempre será superior al de propósito general.

Las razones expuestas anteriormente explican el interés de universidades como Texas A&M en realizar trabajos de investigación en esta área (migración sísmica) usando FPGA.

En su primer trabajo He *et ál*. (2004) desarrollaron un coprocesador para acelerar el proceso de migración. En él alcanzaron "un índice de aceleración de 15,6 veces comparado con una estación de trabajo Pentium 4 de 2,4 GHz" (He *et ál*., 2004). A parte de esta investigación es difícil encontrar más trabajos aplicados en ese campo que usen FPGA. Esta falta de difusión se debe a que la mayoría de ellos están sujetos a cláusulas de confidencialidad por parte de sus patrocinadores. Como se busca analizar la viabilidad de implementar completamente el proceso de migración sísmica 2D sobre un FPGA, se presentarán por separado los estados actuales tanto del proceso de migración (*software* y *hardware*) como de los procesadores específicos implementados sobre FPGA actuales, para luego obtener algunas conclusiones a partir de dichos indicadores. El presente artículo está organizado de la siguiente manera: inicialmente se presenta el concepto de migración y sus desarrollos a la fecha desde el punto de vista del *software* y del *hardware*. Luego se muestra los campos de acción de los procesadores específicos usando FPGA y algunas estrategias de optimización empleadas por los diseñadores de *hardware*. Como se desea analizar la viabilidad de implementar el proceso de migración sobre un FPGA. Posteriormente, se revisan las posibles familias de FPGA sobre las que se podría realizar dicha implementación y se presenta algunas tendencias de rendimiento de los FPGA y los PC en operaciones de punto flotante. Finalmente, las conclusiones, los agradecimientos y las referencias cierran el artículo.

Proceso de migración 2D preapilado en tiempo de *Kirchhoff*

Definición y desarrollos de software.

Para poder entender el proceso de migración 2D, primero se debe tener claro cómo es el experimento sísmico. Para ello miraremos la descripción que hacen de él He *et ál*. (2004): "Las ondas acústicas generadas por una fuente de energía intensa, tal como una carga de dinamita o una pistola de aire, son dirigidas hacia el interior de la tierra y propagadas hacia abajo a una velocidad que depende de las propiedades elásticas del medio. Cuando esta onda llega a una interfaz entre capas de roca, donde la densidad o la velocidad cambian, una porción de la energía es reflejada hacia la superficie".

Mientras que en la superficie, "un arreglo de cientos de geófonos están regularmente distribuidos sobre el suelo para detectar la intensidad y el tiempo de las ondas reflejadas. Las series de tiempo grabadas por cada geófono durante un experimento explosivo (un disparo), son llamadas traza sísmicas. El conjunto de trazas grabadas por todos los receptores durante un disparo forman un agrupamiento de disparo común" (He *et ál*., 2004).

Después de que las trazas se capturan y se procesan, aparecen los reflectores (puntos donde se refleja parte de la energía de la onda) que permiten analizar el subsuelo. El problema es que estos reflectores no aparecen en sus posiciones reales debido a una "ilusión acústica". Entonces, la función de la migración consiste en desplazar a los reflectores de estas posiciones falsas a sus posiciones reales (Claerbout y Green, 2008a). Actualmente se encuentran dos clases de programas que son usados en el proceso de migración. La primera es de tipo industrial, en la que el código fuente es privado porque son desarrollos de *software* que dichas empresas venden (por ejemplo, PRO, 2009; OME, 2009; SEI, 2009; GEO, 2009) e incluso patentan (USP, 2009) como es el caso de la empresa Tsunami (TSU, 2009). La segunda clase es de tipo académico, luego el código fuente es de libre uso y ha sido desarrollado en universidades como Colorado School (Estados Unidos) (CWP, 2009).

Al revisar el código fuente que es de libre uso, se encuentra que matemáticamente esta migración implica operaciones como suma, resta, multiplicación, división, potenciación y radicación. Esto se puede apreciar en la ecuación 1, la cual muestra las operaciones necesarias para calcular el tiempo de vuelo[3].

$$t = \frac{1}{2}\left(\sqrt{\tau^2 + \left[(y-yo)-h\right]^2 / v_{half}^2}\right) + \frac{1}{2}\left(\sqrt{\tau^2 + \left[(y-yo)+h\right]^2 / v_{half}^2}\right) \quad (1)$$

Tomada de (Claerbout y Green, 2008*b*)

El objetivo de revisar esta ecuación es el de mostrar por qué actualmente este proceso tiene un costo computacional alto, ya que "estas dos operaciones de radicación consumen mucho tiempo para la mayoría de CPU" (He *et ál*., 2004). El proceso completo de migración, revisando el código fuente libre, se puede dividir en siete etapas:

1) La primera se encarga de calcular el tiempo de vuelo.
2) La segunda calcula el factor indirecto.
3) La tercera calcula el filtro anti-*aliasing*.
4) La cuarta trae los datos de la traza de entrada.
5) La quinta aplica el filtro anti-*aliasing* sobre estos datos.
6) La sexta escala estos datos por el factor indirecto.
7) Y la séptima suma los resultados.

Estas etapas se presentan en la figura 1 (He *et ál*., 2005).

Otro aspecto importante que sobresale al revisar el código fuente de libre uso es que el algoritmo de migración se puede separar en dos secciones, la secuencial y la paralela, donde "la sección secuencial se encarga de alimentar a la sección paralela con datos de entrada y resultados almacenados, mientras que la parte paralela se encarga de ejecutar el algoritmo de migración" (Panetta *et ál.*, 2007).

La razón por la cual una parte del algoritmo de migración se puede paralelizar es porque cada traza se puede migrar de forma independiente. Luego se pueden tomar varias estrategias de trabajo, que van desde migrar traza por traza, hasta migrar grupos de trazas. Obviamente, la selección de una de estas estrategias termina

[3] Uno de los sub-procesos que conforman el proceso de migración.

afectando el número total de operaciones de lectura y filtrado sobre las trazas de entrada (Panetta *et ál*., 2007).

```
For every input trace in a field data
  Prepare parameters
  Download input trace and parameters to SONIC
  For every output trace allocated to this board
      For every point on this output trace
          Calculate travel time
          Calculate oblique factor
          Calculate anti-aliasing filter Tap
          Fetch data from input trace
          Anti-aliasing filtering
          Scaling by oblique factor
          Summation
      End
  End
End
```

Figura 1. Tomada de (He *et ál*., 2005)

Finalmente, para cerrar esta sección, se puede concluir que los algoritmos de migración de tipo industrial tienen un mayor desempeño que los de tipo académico debido a que estas dos líneas de trabajo buscan objetivos diferentes. Mientras los desarrollos de tipo industrial pretenden ofrecer cada día más servicios, los de tipo académico buscan mostrar cómo se puede implementar el proceso de migración.

Actualidad del procesamiento de datos sísmicos

Para la implementación del proceso de migración (2D o 3D) en la industria del petróleo, típicamente se han usado supercomputadores[4], los cuales han suplido esta necesidad a cambio de altos cos-tos de mantenimiento y operación. En los últimos años, con la aparición de *clusters* de computadores tradicionales (PC), ha surgido una nueva opción para implementar este proceso de migración a un costo más bajo (He *et ál*., 2004).

Estos *clusters* están compuestos por nodos esclavos o estaciones de trabajo y un nodo principal o *frontend*. Típicamente estos recursos son administrados por un sistema operativo Linux. La función del *frontend* consiste en repartir las cargas de trabajo entre los nodos del sistema a través de una herramienta que puede ser MPI (*Message Passing Interface*) o PVM (*Parallel Virtual Machine*) (Sloan, 2004).

Para el caso específico del proceso de migración 2D, el *frontend* distribuye las trazas que se van a procesar hacia los nodos. Cada nodo recibe estas trazas y procede a aplicar el algoritmo de migración. Después de finalizado el proceso, cada nodo devuelve los resultados hacia el *frontend* nuevamente.

Este proceso se repite hasta migrar todas las trazas (Panetta *et ál*., 2007).

Por último, el *frontend* se encarga de recopilar todos los resultados recibidos para entregarlos organizados como un solo paquete. Por obvias razones esta implementación está sujeta a las ventajas y desventajas que ofrecen los clusters de computadores.

Finalmente, se puede decir que los últimos avances en implementación (*hardware*) orientados hacia el proceso de migración han sido desarrollados por la Universidad de Texas A&M. En su trabajo, He *et ál*. (2004) elaboraron un coprocesador usando FPGA. Con él lograron aumentar la capacidad de procesamiento de una estación de trabajo PC. El gran aporte de esta investigación es el de que, si se piensa en un escalamiento, la estación de trabajo puede representar un nodo que forma parte de un *clusters* de computadores, mostrando así una forma de aumentar la capacidad de procesamiento de todo el clusters a partir de sus nodos.

Procesamiento específico usando FPGA

Evolución

A continuación se presenta la evolución de los procesadores específicos implementados sobre los FPGA, tomando como referencia el formato de datos con el que ellos funcionaban. La importancia de revisar esta evolución desde el punto de vista del formato de los datos radica en que el proceso de migración sísmica 2D trabaja con datos en punto flotante de precisión sencilla.

Los primeros procesadores específicos implementados sobre FPGA, se elaboraron para aplicaciones que realizaban operaciones de punto fijo u operaciones con enteros. Esto es debido a que la implementación de esas operaciones dentro del FPGA, no consumen muchos recursos lógicos (CLB), por lo tanto (dentro del FPGA), se podían replicar muchos módulos. Como resultado de la replicación interna de módulos y de una frecuencia de trabajo (interna) elevada, estos primeros procesadores lograron índices de aceleración muy buenos, si se compara su rendimiento con un procesador de propósito general (Puttegowda *et ál*., 2003; Tessier y Burleson, 2001; Patterson, 2000; Sinnappan y Hazelhurst, 2001).

Por otra parte, si se trata de operaciones de punto flotante, inicialmente los FPGA no tenía mucho futuro en este campo debido a que la implementación de un módulo de punto flotante consume muchos recursos lógicos y las primeras FPGA no tenían suficiente densidad en sus recursos. Es por ello que los FPGA no habían logrado incursionar en muchos campos de aplicación científica (Underwood, 2004).

A medida que las empresas fabricantes (Xilinx, Altera) empezaron a desarrollar nuevas familias de FPGA (Virtex 2 Pro, Virtex 4 y Virtex 5) con mayor densidad en sus recursos lógicos y comenzaron a incluir multiplicadores embebidos, los FPGA fueron incursionando poco a poco en el área de operaciones de punto flotante.

Los primeros trabajos con un rendimiento superior al logrado por los procesadores de propósito general, en aplicaciones con operaciones de punto flotante, usaron estrategias como la de no implementar completamente el estándar IEEE 754, bien sea reduciendo el *set* de operaciones (eliminando aquellas que no son necesarias para cada aplicación), o disminuyendo la precisión de los resultados (trabajando con formato simple en lugar del doble) (Dido *et ál*., 2002). Otra estrategia usada para realizar operaciones matemáticas exigentes, como funciones hiperbólicas o trigonométricas, es emplear módulos Cordic[5], los cuales ayudan a reducir significativamente los recursos lógicos de dichas operaciones (He *et ál*., 2004). Después de que se optimizaron los módulos que realizaban las operaciones matemáticas, el siguiente paso en la metodología de diseño de estos trabajos era elaborar un *datapath* a la medida, pensando en mejorar el flujo de datos de dicha aplicación.

Finalmente, elaborar la máquina de estados, que controlara el flujo de datos a través del *datapath*. La importancia de la máquina de estados radica en emplear de la mejor manera el *datapath*, por cada ciclo de reloj.

[4] Es una computadora con capacidades de cálculo muy superiores a las disponibles por las máquinas de escritorio de la misma época

[5] Coordinate Rotation Digital Computer.

De esta forma, los procesadores de propósito específico poco a poco fueron aumentando su campo de acción, ofreciendo un mayor rendimiento, comparado con los procesadores de propósito general. Esto explica el auge en el uso de FPGA para solucionar muchos problemas.

Descripción en *hardware*

Tomando como referencia la metodología planteada por Xilinx (2008e), la descripción en *hardware* se divide en cuatro etapas:

-Entrada del diseño.

-Síntesis del diseño.

-Implementación del diseño.

-Programación del dispositivo de Xilinx.

La entrada del diseño es el proceso de describir el comportamiento o la estructura del circuito deseado, usando el lenguaje de descripción de *hardware* (HDL). Luego de que el circuito es descrito, el siguiente paso es el de sintetizar dicha descripción para detectar y corregir tanto errores de semántica (forma) como errores de uso del lenguaje (fondo).

Posteriormente, el diseñador tiene la posibilidad de simular la descripción del diseño para revisar retardos de propagación, o puede avanzar directamente a la implementación del diseño.

La implementación del diseño es el momento en el que el diseñador le asigna a su descripción en *hardware* (diseño) un lugar dentro del FPGA.

La decisión del lugar donde se ubicaría el circuito dentro del FPGA se debe tomar pensando en los tiempos de respuesta deseados y, en algunas ocasiones, en la optimización del consumo de potencia. La razón por la cual la implementación del diseño afecta los tiempos de respuesta es la de que ésta permite ubicar cada uno de los módulos o bloques del diseño en lugares específicos dentro del FPGA. De esta manera, si cada bloque se coloca lo más cerca posible a su sucesor los tiempos de propagación de compuerta se reducen considerablemente. Ésta es la forma en la que muchos trabajos logran darle la mejor velocidad a los *datapath* (Beauchamp *et ál.*, 2006).

En cuanto al consumo de potencia, la empresa Xilinx ofrece en su *web site* un *White paper* llamado *Virtex 5 FPGA System Power Design Considerations* (wp285). Este artículo, a través del análisis de cada uno de los factores que inciden en el incremento del consumo de potencia dentro del FPGA, ofrece consejos que le permiten al diseñador optimizar el consumo de potencia en su trabajo final.

Después de implementado el diseño, finalmente se procede a la programación del dispositivo. Esta programación se hace simplemente usando la herramienta de trabajo, en este caso la de Xilinx.

Con el análisis de la importancia en la implementación final dentro del FPGA, se cierra esta sección, que busca mostrar algunas estrategias de optimización que usan los desarrolladores de *hardware* actualmente, dando paso al siguiente numeral, en el cual se indican algunas plataformas sobre las que se podría implementar el proceso de migración sísmica 2D.

Plataformas de desarrollo

Sistemas de desarrollo

Al hacer una revisión en el mercado de las empresas que fabrican FPGA se encuentran dos grandes líderes, conocidos como Xilinx (Xil, 2009) y Altera (Alt, 2009), los cuales se encargan de fabricar FPGA de propósito general. Adicionalmente se hallan cinco fabricantes de FPGA con características especiales, como Lattice Semiconductor (Lat, 2009), que fabrica FPGA basadas en tecnología *flash* (no volátil); Actel (Act, 2009), productora de FPGA que incluye mezcladores de señal basados en *Flash;* Quicklogic (Qui, 2009), en antifusibles (sólo se programan una vez); Atmel (Atm, 2009), que fabrica FPGA con microcontroladores AVR[6] internos; y Achronix Semiconductor (Ach, 2009), que fabrica FPGA muy veloces (velocidades internas de 1,5 Ghz).

Debido a que el proceso de migración sísmica 2D tiene un costo computacional alto, se debe seleccionar una empresa que ofrezca adicionalmente al sistema de desarrollo el soporte necesario para poder implementar dicha aplicación. Este soporte debe incluir seminarios en línea, *drivers*, notas de aplicación, *ipcores*[7], etcétera.

Es por ello que al utilizar el soporte como un criterio inicial de selección, Xilinx y Altera sobresalen por un amplio margen de este grupo de siete empresas, limitando así la selección a las dos empresas que tiene mayor experiencia.

Al entrar en más detalle sobre las familias de FPGA de la gama alta de Xilinx que se podrían usar para implementar este proceso, se encuentran las Virtex 5 FXT (con tecnología de 65 nm desarrollada para procesamiento embebido de alto rendimiento) (Xilinx Inc., 2009*c*) y las Virtex 6 SXT (con tecnología de 40 nm desarrollada para procesamiento digital de señales) (Xilinx Inc., 2009*d*). Por otra parte, al revisar las familias de la gama alta de Altera que servirían para implementar este proceso se encuentran las Stratix III E (con tecnología de 65 nm desarrollada para procesamiento digital de señales) (Altera Corporation., 2009) y Stratix IV E (con tecnología de 40 nm desarrollada para procesamiento digital de señales) (Altera Corporation., 2008). Cabe aclarar que cualquiera de estas cuatro familias tienen un entorno favorable para aplicar dicho proceso y que se requiere de un conocimiento mayor del proceso de migración sísmica con el fin de poder seleccionar la alternativa más conveniente y por consiguiente elegir un sistema de desarrollo efectivo.

Cada una de las empresas mencionadas ofrecen diferentes sistemas de desarrollo (*hardware*) junto con sus herramientas de diseño (*software*), y como se optó por Xilinx y Altera de acuerdo a los argumentos previos, a continuación se resumirán las herramientas de diseño que ofrecen cada una de ellas.

Xilinx, por su parte, ofrece:

-ISE: es una herramienta de descripción de *hardware* que permite realizar los diseños lógicos programables. Actualmente existen dos versiones, la WebPACK, que es gratis, y la Foundation, que es licenciada y ofrece el conjunto de librerías completas.

-ChipScope Pro: es una herramienta que permite capturar y analizar las señales internas del FPGA en tiempo real (Xilinx, 2008*b*).

-PlanAhead: permite optimizar los diseños de *hardware* a partir de la mejor ubicación de cada uno de los módulos dentro del

[6] Los AVR son una familia de microcontroladores RISC de Atmel.
[7] Nucleos de propiedad intelectual.

FPGA. Además facilita el flujo de diseño para la reconfiguración parcial (Xilinx., 2008d).

-ISE Simulator: permite simular el comportamiento de los circuitos digitales descritos usando VHDL.

-Platform Studio and the EDK: es un ambiente que ofrece todas las herramientas necesarias para diseñar sistemas de procesamiento embebido (Xilinx, 2008*c*).

-System Generator for DSP: es una herramienta que permite diseñar sistemas DSP de alto rendimiento usando FPGA a partir del entorno Symulink de Matlab (Xilinx, 2009*b*)

-AccelDSP Synthesis tool: permite migrar algoritmos de Matlab dentro de los FPGA (Xilinx, 2008*a*).

Mientras que altera tiene:

-Quartus II: esta es la herramienta de descripción de *hardware* equivalente a ISE de Xilinx.

-Modelsim-Altera: herramienta que permite simular el comportamiento de los circuitos digitales. Debido a que Xilinx (2009*a*) ofrece además una herramienta de simulación llamada Modelsim XE-III, brinda una comparación de los rendimientos de ambas herramientas.

-Nios II: esta herramienta es la equivalente al EDK de Xilinx.

-DSP Builder: es la equivalente al System Generator for DSP de Xilinx.

Como conclusión, se puede observar que, si bien ambas empresas ofrecen FPGA con capacidades muy similares, la diferencia surge en la cantidad de herramientas de desarrollo que ofrecen, inclinando así la balanza a favor de Xilinx si se toma el soporte como el primer criterio de selección.

Tendencias y comparación de rendimientos entre PCs y FPGA

Se mencionó que los FPGA de la gama alta (Virtex 4, Virtex 5) han ido ganando una mayor acogida en aplicaciones que realizan operaciones de punto flotante.

También se aclaró que este avance es producto de la gran densidad de recursos lógicos que ofrecen estas familias de la gama alta y de sus frecuencias de trabajo. Es por ello que en esta sección se presentan tres trabajos que muestran enfoques de análisis distintos de estas tendencias, empezando por el artículo de Keith Underwood.

En su trabajo, Underwood pudo estimar que "los FPGA incrementan su rendimiento en operaciones de punto flotante 4x cada dos años", mientras que "los PC incrementan su rendimiento en operaciones de punto flotante 4x cada tres años" (2004). Los valores mencionados los obtuvo después de analizar operaciones como la suma, la multiplicación, la división y la acumulación.

Este análisis lo hizo para formatos de precisión simple y precisión doble (Underwood, 2004).

Por otra parte, Underwood (2004) también deja claro que no es fácil establecer estas tendencias debido a ser a veces difícil estimar el crecimiento de los FPGA, porque algunas familias tienden a sacrificar recursos lógicos para la inclusión de más procesadores embebidos o más recursos de otra característica.

Otra forma en la que se ha tratado de medir el rendimiento de los FPGA en operaciones de punto flotante es a través de la implementación de tres rutinas BLAS6 (producto punto entre vectores, multiplicación entre vector y matriz y multiplicación entre matrices) sobre esta plataforma. En este nuevo trabajo, Underwood y Hemmert (2004) lograron concluir que el rendimiento de los FPGA era superior al de los PC tradicionales siempre y cuando se usen *memory bound functions* (Underwood y Hemmert, 2004). Finalmente, el trabajo de Craven y Athanas (2007) brinda una visión diferente de la comparación entre los FPGA y los PC. Este trabajo (Craven y Athanas, 2007) está más orientado a hacer una crítica de la forma como se comparan actualmente estas tecnologías. Lo interesante de este último trabajo es que resalta ser las comparaciones hechas hasta ahora injustas por tres aspectos.

El primer aspecto consiste en que hoy en día la academia ha dedicado mucho esfuerzo optimizando librerías para demostrar que los FPGA son mejores que los PC y enfatiza que si se dedicara el mismo esfuerzo en optimizar los programas muy seguramente los resultados de los PC serían los mejores (Craven y Athanas, 2007).

El segundo aspecto que resalta este trabajo (Craven y Athanas, 2007), son los costos actuales de los FPGA. Enfatiza en que si bien los FPGA logran mayor rendimiento en varias aplicaciones comparados con los PC, si se comparan los costos de implementación los FPGA pierden frente a los PC (Craven y Athanas, 2007).

El tercer y último aspecto son los tiempos de desarrollo de las aplicaciones. El tiempo que se gasta desarrollando una aplicación en un FPGA es superior de la aplicación en un PC. Aunque hoy en día hay unas herramientas de *software* que permiten agilizar este proceso, también es cierto que la calidad de dichas aplicaciones nunca serán igual de eficientes a las desarrolladas por un ingeniero de *hardware* (Craven y Athanas, 2007).

Quizás el único aspecto que no se analizó en este trabajo (Craven y Athanas, 2007), son los costos de mantenimiento de los clusters. Se considera que si estos costos se tienen en cuenta la diferencia en $/Gflop se acortará entre estas dos tecnologías.

Con estos tres trabajos se manifiesta por qué es difícil generalizar una comparación de rendimientos y costos de las dos tecnologías. Además, dichos trabajos (Underwood, 2004; Underwood y Hemmert, 2004; Craven y Athanas, 2007) logran contener la visión global de las tendencias de desempeño en operaciones de punto flotante de estas dos tecnologías en los últimos ocho años.

Conclusiones

En este artículo se presentó el estado actual del proceso de migración sísmica usando dos enfoques diferentes (desarrollos de *software* y de *hardware*). De la información revisada se pudo concluir que:

Los desarrollos de software orientados al proceso de migración, están bastante avanzados. En el mercado se encuentran desde versiones académicas libres y básicas, hasta versiones comerciales que son completas pero muy costosas.

También se pudo observar que los desarrollos de *hardware* se limitan en su mayoría, a la implementación de costosos *clusters* de computadores por parte de las empresas de la industria del petróleo. Dejando así un pequeño espacio a las primeras implementaciones parciales de este proceso sobre plataformas reconfigurables como los FPGA.

Además se mostró la evolución de los FPGA y las tendencias de los mismos en operaciones de punto flotante y se pudo observar que las proyecciones de rendimiento de los FPGAs superan ampliamente a los procesadores de propósito general.

También se pudo apreciar que en la actualidad las empresas que fabrican FPGA pueden ofrecer el soporte necesario (*software* y *hardware*) para poder implementar el proceso de migración sísmica 2D sobre dicha plataforma.

Se puede concluir de manera general que los FPGA en esta área tienen un gran futuro. Área que se ha ido desarrollando a medida que las empresas de la industria del petróleo han tomado la decisión de empezar a desarrollar sus propias herramientas con el fin de optimizar aun más sus procesos.

Como futuro trabajo nos enfocaremos en un estudio de viabilidad de la implementación de procesos como, migración en profundidad y migración 3D en tiempo de Kirchhoff, en un FPGA.

Agradecimientos

Los autores le agradecen a William Mauricio Agudelo Zambrano, funcionario del Instituto Colombiano del Petróleo (Bucaramanga, Colombia), por todo el apoyo ofrecido durante la realización de este trabajo bajo el convenio de cooperación tecnológica UIS-ICP número 005 de 2003.

Bibliografía

Ach., Achronix Semiconductor Corporation, 2009. Disponible en: http://www.achronix.com/. Consultado Marzo de 2009.

Act., Actel Power Matters., 2009. Disponible en: http://www.actel.com/. Consultado Marzo de 2009.

Alt., Altera Corporation., 2009. Disponible en: http://www.altera.com/. Consultado Marzo de 2009.

Altera Corporation., Stratix IV Device Family Overview., 2008. Disponible en: http://www.altera.com/literature/hb/stratix-iv/stx4siv51001.pdf

Altera Corporation., Stratix III Device Family Overview., 2009. Disponible en: http://www.altera.com/literature/ hb/ stx3/ stx3siii51001.pdf

Anghelescu, P., Ionita, S., Sofron, E., FPGA implementation of hybrid additive programmable cellular automata encryption algorithm, in 'HIS '08: Proceedings of the 2008 8th International Conference on Hybrid Intelligent, Systems', IEEE Computer Society, Washington, D.C., USA, 2008, pp. 96–101.

Anghelescu, P., Sofron, E., Rincu, C.-I., Lana, V.-G., Programmable cellular automata based encryption algorithm', Semiconductor Conference, CAS 2008, International 2, 2008, pp. 351–354.

Atm., Atmel Corporation., 2009. Disponible en: http://www.atmel.com/. Consultado Marzo de 2009.

Baker, Z. K., Prasanna, V. K., Time and area efficient pattern matching on fpgas, in FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, ACM, New York, N.Y., USA, 2004, pp. 223–232.

Beauchamp, M. J., Hauck, S., Underwood, K. D., Hemmert, K. S., Embedded floating-point units in FPGAs, in FPGA '06: Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays, ACM, New York, N.Y., USA, 2006, pp. 12–20.

Bogdán, I. A., Rivers, J., Beynon, R. J., Coca, D., High-performance hardware implementation of a parallel database search engine for real-time peptide mass fingerprinting., Bioinformatics, 24(13), 2008, pp. 1498–1502.

Cho, Y. H., Mangione-Smith, W. H., Deep network packet filter design for reconfigurable devices., Trans. On Embedded Computing Sys., 7(2), 2008, pp. 1–26.

Cho, Y., Mangione-Smith, W., Fast reconfiguring deep packet filter for 1+ gigabit network., Field-Programmable Custom Computing Machines, FCCM 2005, 13th Annual IEEE Symposium, 2005, pp. 215–224.

Claerbout, J. F., Green, I., Basic Earth Imaging., Stanford University, 2008a, pp. 60-61.

Claerbout, J. F., Green, I., Basic Earth Imaging., Stanford University., 2008b., pp. 125-128.

Craven, S., Athanas, P., Examining the viability of FPGA supercomputing', EURASIP J. Embedded Syst., No. 1, 2007, pp. 13–13.

CWP., Center for wave phenomena colorado school of Mines., 2009. Disponible en: http://www.cwp.mines.edu/cwpcodes/. Consultado en marzo de 2009.

Dido, J., Geraudie, N., Loiseau, L., Payeur, O., Savaria, Y., Poirier, D., A flexible floating-point format for optimizing data-paths and operators in FPGA based DSPS, in FPGA '02: Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays, ACM, New York, N.Y., USA, 2002, pp. 50–55.

GEO., Geocluster Seismic Processing System., 2009. Disponible en: http://www.cggveritas.com/default.aspx?cid=13. Consultado Marzo de 2009.

He, C., Lu, M., Sun, C., Accelerating seismic migration using FPGA-based coprocessor platform, in FCCM '04: Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, IEEE Computer Society, Washington, D.C., USA, 2004, pp. 207–216.

He, C., Sun, C., Lu, M., Zhao, W., Prestack Kirchhoff time migration on high performance reconfigurable computing platform, SEG Technical Program Expanded Abstracts, Vol. 24, No. 1, 2005, pp. 1902–1905. Disponible en: http://link.aip.org/link/?SGA/24/1902/1

Hoang, D. T., Lopresti, D. P., FPGA implementation of systolic sequence alignment., in International Workshop on Field Programmable Logic and Applications, 1993.

Jean, J., Liang, X., Drozd, B., Tomko, K., Accelerating an ir automatic target recognition application with FPGAs., in FCCM '99: Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines, IEEE Computer Society, Washington, D.C., USA, 1990, pp. 290.

Lat., Lattice Semiconductor Corporation., 2009. Disponible en: http//www.latticesemi.com/. Consultado Marzo de 2009.

OME., Omega Seismic Processing System., 2009. Disponible en: http://www.westerngeco.com/content/services/dp/omega/index.asp? Consultado Marzo de 2009.

Panetta, J., de Souza Filho, P. R. P., da Cunha Filho, C. A., da Motta, F. M. R., Pinheiro, S. S., Junior, I. P., Rosa, A. L. R., Monnerat, L. R., Carneiro, L. T., de Albrecht, C. H. B., Computational characteristics of production seismic migration and its performance on novel processor architectures., in in Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing, IEEE Computer Society, 2007.

Patterson, C., High performance des encryption in virtex(tm) FPGAs using jbits(tm), in FCCM '00: Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines, IEEE Computer Society, Washington, D.C., USA, 2000, pp. 113.

PRO., ProMAX Seismic Processing Family, 2009. Disponible en: http://www.halliburton.com/ps/default.aspx?pageid=.862n&navid=221n&prodid=MSE::1055450737429153. Consultado Marzo de 2009.

Puttegowda, K., Worek, W., Pappas, P., Dandapani, A., Athanas, P., Dickerman, A., A run-time reconfigurable system for gene-sequence searching, in Proceedings of the 16th International Conference on VLSI Design, IEEE Computer Society, 2003, pp. 561–566.

Qui., Quicklogic., 2009. Disponible en: http://www.quicklogic.com/. Consultado Marzo de 2009.

SEI., Seisup Seismic Processing System., 2009. Disponible en: http://www.geocenter.com/seisup/seisup.html.Consultado Marzo de 2009.

Sinnappan, R., Hazelhurst, S., A reconfigurable approach to packet filtering, in FPL '01: Proceedings of the 11th International Conference on Field-Programmable Logic and Applications., Springer-Verlag, London, UK, 2001, pp. 638–642.

Sloan, J., High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI (Nutshell Handbooks), O'Reilly Media, Inc., 2004. Disponible en: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0596005709.

Tessier, R. , Burleson, W., Reconfigurable computing for digital signal processing: A survey., Journal of VLSI Signal Processing, 28, 2001, pp. 7–27.

TSU., Tsunami development., 2009. Disponible en: http://www.tsunamidevelopment.com/prestack time mig.php. Consultado Marzo de 2009.

Underwood, K., FPGAs vs. cpus: trends in peak floatingpoint performance., in FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, ACM, New York, N.Y., USA, 2004, pp. 171–180.

Underwood, K., Hemmert, K., Closing the gap: Cpu and fpga trends in sustainable floating-point blas performance., Field-Programmable Custom Computing Machines, FCCM 2004. 12th Annual IEEE Symposium, 2004, pp. 219–228.

USP., Us patent 6,996,470, 2009. Disponible en: http://www.patentstorm.us/patents/6996470.html. Consultado Marzo de 2009.

Villasenor, J., Schoner, B., Chia, K.-N., Zapata, C., Kim, H. J., Jones, C., Lansing, S., Mangione-Smith, B., Configurable computing solutions for automatic target recognition., FPGAs for Custom Computing Machines, Proceedings IEEE Symposium, 2006, pp. 70–79.

Wang, Y., Shen, Y., Optimized fpga realization of digital matched filter in spread spectrum communication systems., in CITWORKSHOPS '08: Proceedings of the 2008 IEEE 8th International Conference on Computer and Information Technology Workshops, IEEE Computer Society, Washington, D.C., USA, 2008, pp. 173–176.

Xil., Xilinx website., 2009. Disponible en: http://www.xilinx.com/. Consultado Marzo de 2009.

Xilinx inc., AccelDSP Synthesis Tool, User Guide., 2008a. Disponible en: http://www.xilinx.com/ support/documentation/sw manuals/acceldsp user.pdf

Xilinx inc., ChipScope Pro 10.1 Software and Cores, User Guide., v10.1 edn., 2008 b. Disponible en: URL: http://www.xilinx.com/support/documentation/swmanuals/chipscope pro sw cores 10 1 ug029.pdf

Xilinx inc., EDK Concepts, tools, and Techniques., 2008c. Disponible en: http://www.xilinx.com/ support/documentation/sw manuals/edk ctt.pdf

Xilinx inc., PlanAhead Tutorial Release 10.1., 2008d. Disponible en: http://www.xilinx.com/ support/documentation/swmanuals/PlanAhead10-1 Tutorial.pdf

Xilinx Inc., Xilinx ISE 10.1 Design Suite Software Manuals and Help - PDF Collection., 2008e. Disponible en: http://www.xilinx.com/itp/xilinx10/books/manuals.pdf

Xilinx inc., Average ModelSim XE-III Simulation Performance Compared to ModelSim Altera Edition., 2009a. Disponible en: URL: http://www.xilinx.com/ ise/ verification/mxe details.html#compare

Xilinx inc., System Generator for DSP Getting Started Guide, User Guide and Reference Guide., 2009b. Disponible en: http://www.xilinx.com/support/documentation/swmanuals/sysgen bklist.pdf

Xilinx Inc., Virtex-5 Family Overview., 2009c., Disponible en: http://www.xilinx.com/support/documentation/datasheets/ds100.pdf

Xilinx Inc., Virtex-6 Family Overview., 2009d. Disponible en: http://www.xilinx.com/publications/prodmktg/Virtex6Overview.pdf

Yamada, M., Nishihara, A., A high-speed fir digital filter with csd coefficients implemented on FPGA., in ASPDAC '01: Proceedings of the 2001 conference on Asia South Pacific design automation, ACM, New York, N.Y., USA, 2001, pp. 7–8.