Ingeniería y Ciencia
ISSN:1794-9165
ISSN-e: 2256-4314
ing. cienc., vol. 9, no. 17, pp. 261–293, enero-junio. 2013.
http://www.eafit.edu.co/ingciencia
This a open-access article distributed under the terms of the Creative Commons Attribution License.

Reducción de los tiempos de cómputo de la Migración Sísmica usando FPGAs y GPGPUs: Un artículo de revisión

Carlos Fajardo[1], Javier Castillo Villar [2] y
César Pedraza[3]

Recepción: 27-01-2012, Aceptación: 01-02-2013
Disponible en línea: 22-03-2013

MSC:68M20 / PACS:93.85.Rt

Resumen

Este artículo hace una revisión entorno a los esfuerzos que actualmente se están realizando con el propósito de reducir el tiempo de cómputo de la MS. Nosotros introducimos los métodos más utilizados para realizar el proceso de Migración, así como también las dos arquitecturas computacionales que están ofreciendo mejores tiempos de procesamiento. Revisamos las implementaciones más representativas de este proceso sobre estas dos tecnologías y resumimos los aportes de cada una de estas investigaciones. El artículo finaliza con un análisis acerca de la dirección que deben tomar futuras investigaciones en esta área.

Palabras clave: Evaluación del desempeño, FPGA, GPGPU, Métodos sísmicos de exploración, Migración Sísmica.

[1] Ph.D.(c) cafajar@uis.edu.co, Universidad Industrial de Santander, Bucaramanga, Colombia.
[2] Doctor en Ingeniería Informática, javier.castillo@urjc.es, Universidad Rey Juan Carlos, Madrid, España.
[3] Doctor en Ingeniería Informática, cesarpedraza@usantotomas.edu.co, Universidad Santo Tomás, Bogotá, Colombia.

Aspectos relevantes

• Métodos más usados para realizar la Migración Sísmica. • Revisión de la implementación de la Migración Sísmica sobre FPGA y GPGPU. • Futuras investigaciones.

Reduction of computation time of Seismic Migration using FPGAs and GPGPUs: A review article.

Abstract

This article makes a review around the efforts that are currently being carried out in order to reduce the computation time of the MS. We introduce the methods used to make the migration process as well as the two computer architectures that are offering better processing times. We review the most representative implementations of this process on these two technologies and summarize the contributions of each of these investigations. The article ends with our analisys about the direction that future research should take in this area.

Key words: Exploration seismic methods, FPGA, GPGPU, Performance evaluation, Seismic Migration.

1 Introducción

En los últimos 150 años se han consumido alrededor de 1 billón de barriles (de equivalente) de petróleo, debido al incremento de la demanda, durante los próximos veinte años será necesario encontrar y extraer la misma cantidad de petróleo que en los 150 precedentes [1]. El petróleo y gas de fácil extracción ya han sido agotados y para ubicar nuevas reservas se tendrá que buscar en áreas geológicas complejas y que en el pasado fueron descartadas por el alto coste de extracción. Para tal fin la industria petrolera deberá realizar imágenes del subsuelo con una mayor resolución a las realizadas anteriormente.

Para la construcción de las imágenes del subsuelo, las empresas tradicionalmente han usado plataformas de cómputo hasta con miles de CPU, sin embargo, los sistemas de computo basados en procesadores tradicionales se han visto frenados principalmente por tres causas: la primera consiste en la relación cúbica entre la frecuencia y la potencia, lo cual ha detenido la carrera por la velocidad, este efecto se conoce como muro de

potencia (*power wall*); otra causa es la gran diferencia que existe entre la velocidad a la cual los datos son transferidos al procesador y la velocidad a la que éste ejecuta las instrucciones, esta limitación se conoce con el nombre de muro de memoria (*memory wall*); finalmente las posibilidades para que los procesadores continúen aumentando el paralelismo a nivel de instrucción IPL(*Instruction Level Parallelism*) no son prometedoras, este efecto se conoce como IPL wall [2].

Debido a las tres barreras mencionadas anteriormente, la computación de alto rendimiento se ha encaminado hacia la computación híbrida, en donde las CPU trabajan conjuntamente con una serie de nuevas arquitecturas que aprovechan mejor el paralelismo de las aplicaciones. Ahora bien, la pregunta para los investigadores en computación sísmica de alto rendimiento es: ¿cuál es el hardware más apropiado para implementar el proceso de Migración Sísmica (MS)? investigaciones recientes [3], [4], [5] muestran que tanto las GPGPU (*General-Purpose Computing on Graphics Processing Units*) como las en FPGAs (*Field Programmable Gate Array*) son dos opciones para acelerar el proceso de MS.

Este artículo describe el problema de la MS desde la perspectiva computacional, de igual manera revisa las fortalezas y debilidades que estas dos arquitecturas (FPGAs y GPGPUs) tienen a favor o en contra del proceso de MS. En esta revisión también se analizan las implementaciones más representativas de la MS tanto en las FPGAs como las GPGPUs encontradas en la literatura y se resumen los aportes que éstas han hecho en relación a los problemas de memoria, transferencia de datos y de cómputo.

El resto de este artículo está organizado de la siguiente manera: la sección 2 corresponde a la descripción del experimento sísmico y las diferentes estrategias para realizar el proceso de migración, las secciones 3 y 4 se dedican para introducir las FPGAs y las GPGPUs respectivamente, en la sección 5 se revisan las diferentes implementaciones de la MS sobre estas arquitecturas y se resumen los principales aportes de cada una de estas implementaciones y finalmente se muestran las conclusiones.

2 Adquisición y procesamiento de datos Sísmicos

El costo de una exploración del subsuelo está en el orden de las decenas de millones de dólares y la probabilidad de encontrar petróleo es relativamente baja, por lo cual, la industria petrolera debe realizar estudios que le permitan reducir la incertidumbre en las áreas a explorar. La principal técnica utilizada por la industria del petróleo para identificar posibles reservas de hidrocarburos es construir una imagen del subsuelo por medio de la técnica de la Reflexión Sísmica, esta técnica está basada en las propiedades de reflexión de las ondas sonoras.

2.1 Adquisición de las trazas sísmicas

Para la construcción de una imagen del subsuelo, se requiere adquirir gran cantidad de datos sísmicos, esta adquisición se inicia con la ubicación de una serie de sensores, llamados geófonos, sobre el área a explorar; si los sensores se ubican en línea recta se dice que la adquisición es 2D o si ocupan una área se dice que la adquisición es 3D. Una vez ubicados los geófonos, se activa una fuente artificial de sonido en el área a explorar (un camión vibrador, una carga explosiva o bombas de aire en el océano). El registro de un geófono durante un disparo recibe el nombre de *traza sísmica*.

2.2 La migración sísmica

Con el propósito de construir una imagen del subsuelo, las señales sísmicas detectadas por los geófonos (las trazas) deben ser procesadas en varios módulos de procesamiento, el último de ellos se denomina Migración Sísmica y es el de mayor costo computacional. Este módulo busca reubicar los reflectores (las uniones de los diferentes geológicos) de una posición virtual a su posición correcta; la importancia de la migración de los datos sísmicos fue reconocida desde 1921 [6] ya que sin este último módulo la imagen que se obtendría sería incorrecta, especialmente en zonas geológicamente complejas [7].

En el experimento sísmico se asume que la tierra está compuesta por una serie de capas z_i, como se ilustra en la Figura 1; las fuentes y los receptores se ubican en la superficie del área a explorar; la velocidad de la propagación de la onda en la tierra $v(x, y, z)$, varía en todas las direcciones, pero para simplificar el modelo algunas veces se asume que la velocidad sólo varía verticalmente y que es constante durante toda la capa z_i. La migración permite reubicar el reflector AB' supuesto en el punto medio entre el emisor y el receptor (Figura 1), a su posición correcta AB; esto se logra por medio de la solución de la ecuación de onda, Ecuación (1). La solución de esta ecuación utiliza un modelo de velocidad (v) del terreno a explorar y la información recolectada por los geófonos [8].

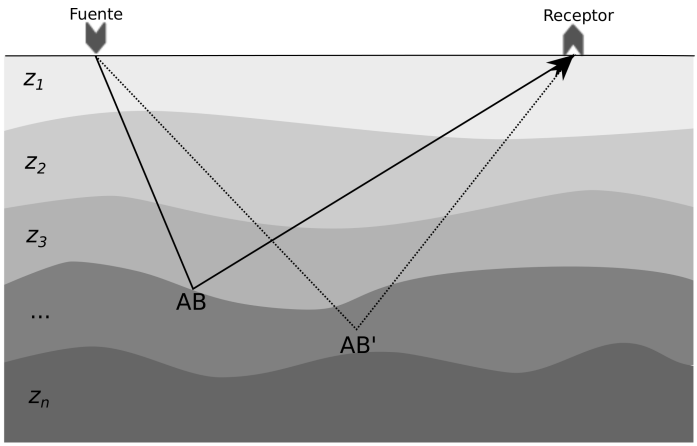

Figura 1: Problema que resuelve la migración sísmica. (Adaptado de [8])

Matemáticamente, la migración sísmica trata de encontrar una solución $P(x, y, z, t)$ a la Ecuación (1), usando la condición de frontera $P(x, y, 0, t)$, que fue recolectada en la superficie del terreno [7].

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} + \frac{\partial^2 P}{\partial z^2} = \frac{1}{v^2}\frac{\partial^2 P}{\partial t^2}\,, \tag{1}$$

La imagen del subsuelo $I(x, y, z)$ puede ser extraída de $P(x, y, z, t)$, evaluando esta última en $t = 0$ (Ecuación (2)), basado en el concepto del reflector que explota [9]:

$$I(x, y, z) = P(x, y, z, t)|_{t=0}\,, \tag{2}$$

2.3 Algoritmos utilizados para realizar la migración sísmica

Como se mencionó anteriormente, la importancia de la migración fue reconocida desde hace varios años y a partir de ahí han aparecido una serie de algoritmos que permiten realizar este procedimiento. La industria del petróleo usa paquetes comerciales de software para hacer el procesamiento de sus datos sísmicos, como por ejemplo ProMAX® Family Seismic Data Processing Software y SeisUP®. Estas herramientas procesan los datos sísmicos por medio de varios módulos, el último de ellos es el de migración.

Los algoritmos son diferentes estrategias de solución a la Ecuación (1) y utilizan alguna aproximación escalar a dicha ecuación [10],[7]. A continuación se hace una descripción de algunos de estos algoritmos desde el punto de vista computacional:

2.3.1 Migración de Kirchhoff Este es el método de migración más popular en la industria del petróleo y fue el primero en ser implementado en un equipo de cómputo y uno de los de menor costo computacional. El principio matemático que utiliza este algoritmo es bastante sencillo: se parte de un emisor y un receptor en la superficie a explorar, se mide el tiempo que se tarda el frente de onda en hacer la trayectoria emisor-reflector-receptor; con este valor se pueden calcular todos los posibles reflectores. Con velocidad constante, el lugar geométrico que ocupan estos posibles reflectores, es la mitad inferior de una elipse para el caso 2D o un elipsoide para el caso 3D (de acuerdo a la definición geométrica de una elipse) como se muestra Figura 2. Esta información (la traza no migrada) no es suficiente para decidir cuál de todos los infinitos puntos que conforman este lugar geométrico es el receptor verdadero, con otra

traza sísmica, la migración puede calcular otros posibles reflectores, la migración de Kirchhoff repite este procedimiento sobre todas las trazas y finalmente suma todas las contribuciones de las elipses (o los elipsoides) y de esta manera se forma la imagen del subsuelo [10].

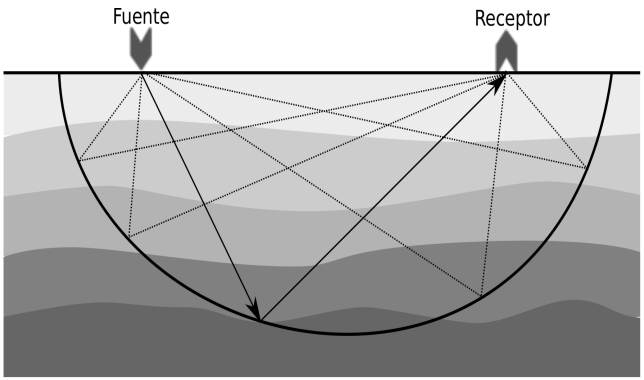

Figura 2: Lugar geométrico de los posibles reflectores (Adaptado de [11])

Matemáticamente la migración de Kirchhoff implica operaciones de suma, resta, multiplicación, división y radicación las cuales generalmente se realizan en formato de coma flotante precisión sencilla pues el formato de precisión doble no es requerido para esta aplicación.

2.3.2 Cambio de Fase El cambio de fase es un método que hace uso del concepto del *reflector que explota* [9]. Se parte de $P(x, y, 0, t)$ hasta llegar a $P(x, y, z, 0)$; en este método la extrapolación se hace en el dominio de la frecuencia y por medio de un operador de cambio de fase. Este método comienza con una transformada de Fourier de los datos, que puede ser en 2D o 3D (dependiendo del tipo de adquisición), luego se aplica el operador de cambio de fase y finalmente se utiliza una transformada inversa de Fourier [7].

2.3.3 Método por Diferencias Finitas Consiste en la discretización de la Ecuación (1) (o una aproximación de ella), reemplazando las derivadas parciales por una aproximación en diferencias centrales (esto se

logra por medio de la aproximación de la serie de Taylor). Para mejorar la precisión en este método se requiere aumentar el número de puntos que se toman en la aproximación, pero obviamente esto requiere una mayor cantidad de operaciones computacionales. De igual manera en este método se hace necesario tener en cuenta problemas relacionados con los errores de truncamiento y criterios de estabilidad con el fin de asegurar la convergencia del método [12].

2.3.4 Migración Reversa en el Tiempo Este es el método que en la actualidad ofrece una mejor calidad en la imágenes y aunque fue introducido en 1983 [13] su alto costo computacional impidió que se usara en el pasado [5]. Este consiste en resolver una aproximación de la Ecuación (1) dos veces, estas dos soluciones reciben el nombre de propagaciones hacia adelante y hacia atrás, luego los dos resultados son correlacionados para obtener la imagen [4]. Matemáticamente este método implica el operador Laplaciano, diferencias finitas y un método explícito para realizar integración en el tiempo [14].

2.4 Clases de Migración

La migración puede ser aplicada de diferentes formas, estas formas son llamadas *clases*: pos-apilado, pre-apilado, 2D o 3D, en tiempo o en profundidad. Estas clases forman ocho posibles combinaciones [15].

2.4.1 Migración Pos-apilada y Migración Pre-apilada Un proceso usado para mejorar la relación señal a ruido de la trazas sísmicas y reducir el costo de procesamiento es el apilamiento. Este proceso consiste en sumar las trazas obtenidas en varios disparos que se reflejan en un mismo punto [16]. Si la migración se aplica después del apilamiento se denomina migración pos-apilada y su principal ventaja es la reducción del tiempo de cómputo, debido a que el número de trazas a ser procesadas se decrementa significativamente; su desventaja es que este procedimiento disminuye la precisión, especialmente en terrenos geológicamente complejos. Cuando la migración se aplica antes del apilamiento (migración pre-apilada) se mejora la precisión, pero se requiere más tiempo

de cómputo (entre 60 y 120 veces más) que en la migración pos-apilada [16],[15].

2.4.2 Migración 2D o Migración 3D Dependiendo de como las trazas sísmicas son grabadas, la migración puede ser clasificada en 2D o 3D. En la migración 2D, la energía es reflejada en un plano, luego los geófonos son localizados en linea recta. En la migración 3D los geófonos ocupan un área sobre la superficie a explorar. De manera general la migración 3D puede proveer una mayor resolución en la imágenes pero requiere un mayor tiempo de cómputo, lo que puede durar días en el procesamiento en sísmica 2D puede tardar semanas con adquisiciones 3D [15].

2.4.3 Migración en tiempo o Migración en profundidad Existen dos enfoques para la migración sísmica: el tiempo y la profundidad. En términos simples la migración en tiempo localiza los reflectores midiendo el tiempo de viaje del frente de onda, mientras que la migración en profundidad lo hace en función de la profundidad. La migración en tiempo ha sido la más empleada durante los últimos años, pero algunos estudios [16] muestran que la migración en profundidad ofrecen mejores resultados pero se requiere un mejor modelo de velocidad y mayor capacidad de cómputo.

2.5 Criterios para la selección de la clase migración

Como se mencionó anteriormente si se combinan las diferentes posibilidades que se tienen actualmente para aplicar la migración, se tienen ocho diferentes posibilidades. En la Figura 3 se resumen los criterios de selección descritos por Farmer en [15] (aqui no se tiene en cuenta el tipo de adquisión 2D o 3D):

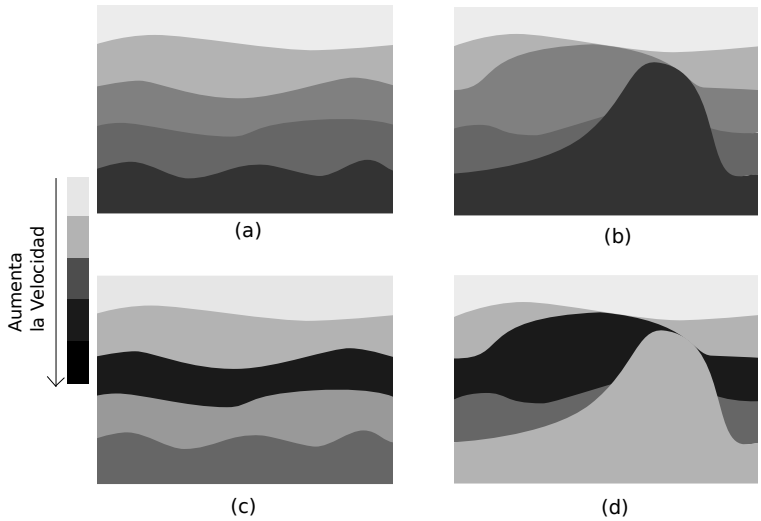

Figura 3: Esquema para la selección de la clase de migración [15]:(a) Velocidades simples + estructura simple = Migración pos-apilada en tiempo; (b) Velocidades simples + estructura compleja = Migración pre-apilada en tiempo; (c) Velocidades complejas + estructura simple = Migración pos-apilada en profundidad; (d) Velocidades complejas + estructura compleja = Migración pre-apilada en profundidad

2.6 Costo computacional de la migración sísmica

Una ventaja del proceso de la MS es que gracias a que los disparos se hacen en tiempos diferentes, cada una de las trazas sísmicas puede ser procesada de manera independiente, lo cual facilita la paralelización del proceso al ser implementado [14],[17]. Sin embargo, la implementación de la MS sobre un sistema de cómputo tiene tres retos principales:

2.6.1 Transferencia de Datos La cantidad de datos que requieren ser transferidos desde la memoria principal a los dispositivos de cómputo es elevada. una adquisición típica puede requerir 30000 geófonos y cada sensor muestrea datos a más de 2kbps, con una nueva detonación cada 10 segundos, de tal forma que la cantidad de datos que se obtienen cada día puede estar en el orden de los terabytes [18].

2.6.2 Memoria La cantidad de memoria requerida para realizar la MS fácilmente supera la capacidad de memoria disponible en el dispositivo de cómputo [14].

2.6.3 Cómputo La migración sísmica es una aplicación que requiere gran cantidad de cómputo, actualmente el tiempo requerido para migrar los datos obtenidos durante un estudio sísmico está en el orden de las semanas.

3 Dispositivos Lógicos Programables

Una alternativa para acelerar el proceso de migración sísmica son los dispositivos lógicos programables. Entre estos se destacan las FPGA (*Field Programmable Gate Array*), siendo su uso muy extendido en multitud de problemas de aceleración de algoritmos complejos.

Las FPGA permiten crear funciones complejas mediante la interconexión de una gran cantidad de recursos lógicos. Los bloques lógicos configurables (CLBs por sus siglas en inglés) son el principal recurso para la construcción de funciones lógicas. Los bloques de entrada/salida (IOBs por sus siglas en inglés) facilitan la comunicación de los pines de entrada/salidas hacia el mundo exterior y la matriz de interconexión permite interconectar los CLBs entre sí, así como también facilitar la interconexión con el mundo exterior a través de los bloques de entrada/salida.

La comunicación de las FPGA con el exterior puede realizarse mediante gran cantidad de interfaces de salida. Actualmente uno de las más utilizados son los puertos PCI-Express que permiten transferir datos entre la FPGA y la CPU a velocidades de varios Gb/s. Para las tareas de computación de alto rendimiento es también habitual que la placa donde va insertada la FPGA contenga una gran cantidad de memoria externa del tipo DDR2 o incluso DDR3.

Los proveedores de FPGAs como Xilinx®, Altera®, Atmel®, Lattice Semiconductor Corporation® y Achronix Semiconductor Corporation® -dentro de los que se destacan los dos primeros- ofrecen una serie módu-

los IP tanto softcore (descritos en un HDL) como hardcore (implementados en fábrica) que van desde módulos matemáticos hasta procesadores completos [19],[20]. De esta forma, bloques comunmente utilizados en los diseños, como unidades de punto flotante, no deben de ser diseñadas desde cero en cada nueva aplicación.

Adicionalmente desde hace un par de años, algunos fabricantes tradicionales de equipos de computación de alto rendimiento (Cray®, SGI®, Annapolis Micro Systems®, Nallatech®, Maxeler®, DRC® y Mercury Computer®) empezaron a incluir dentro de su catálogo de productos, sistemas de cómputo que incluyen FPGA. De manera general, estos equipos de cómputo cuentan con potentes CPU combinadas con FPGA.

3.1 Las FPGA y la computación de alto rendimiento

Las FPGA ofrecen muchas posibilidades para acelerar aplicaciones de cómputo intensivo, gracias a que pueden ser configurados para desarrollar tareas con un alto grado de paralelismo[21],[22], sin embargo la frecuencia a la que actualmente operan las FPGA es unas diez veces menor que la frecuencia de un procesador de escritorio, por lo cual, se requiere que las aplicaciones implementadas en la FPGA realicen al menos 10 veces más trabajo computacional por ciclo de reloj, para estar a la par con un procesador convencional.

Las FPGA ya han demostrado que tienen el potencial para acelerar aplicaciones de cómputo intensivo, prueba de esto es la constante publicación de procesos computacionales implementados sobre FPGA; estas publicaciones cubren áreas como: redes neuronales [23],[24],[25],[26], búsqueda de secuencias genéticas [27],[28], filtrado digital [29],[30],[31], [32], filtrado de paquetes de red [33],[34] y simulaciones financieras Monte-Carlo [35],[36] entre otras.

3.2 Características de las aplicaciones susceptibles de ser aceleradas en una FPGA

No todas las aplicaciones se pueden beneficiar de igual manera en una FPGA y debido a la dificultad que representa mapear una aplicación en estos dispositivos, Es aconsejable hacer algunas estimaciones previas con el fin determinar las posibilidades de aceleración que tendrá un algoritmo específico.

Las aplicaciones que requieran procesar gran cantidad de datos con poca o ninguna dependencia de datos son las ideales para implementar en las FPGA, adicionalmente se requiere que dichas aplicaciones estén limitadas por cómputo y no por transferencia de datos [37]. La migración sísmica requiere procesar gran cantidad de datos (en el orden de las decenas Terabytes) y por otro lado las trazas sísmicas pueden ser procesadas de manera independiente, lo cual disminuye sustancialmente la dependencia de los datos. Sin embargo, la migración también tienen una gran transferencia de datos y las las operaciones de entrada/salida en las FPGA están limitadas por el ancho de banda del PCI-express (2Gb/s).

Por otro lado, la precisión y el formato de los datos son otro aspecto influyente sobre el rendimiento de las aplicaciones sobre las FPGA, por lo general con una menor precisión (menor cantidad de bits para representar los datos) se aumenta el rendimiento de la aplicación dentro de la FPGA; con respecto al formato de los datos, las FPGA obtienen mejores desempeños en números en punto fijo que números en coma flotante, algunas estrategias para implementar aplicaciones en números de coma flotante en las FPGA es reducir la cantidad de bits con los cuales se representa los números [37]. La migración se ha implementado con números en coma flotante precisión simple [17] y algunas investigaciones [38], [3] muestran que es posible implementar algunas partes de este módulo utilizando punto fijo en lugar de coma flotante y obtener imágenes con patrones muy similares (visualmente idénticas).

3.3 Lenguajes de programación para las FPGA

La implementación de una aplicación sobre esta tecnología se realiza por medio de Lenguajes de Descripción de Hardware (HDLs por sus siglas en inglés) como VHDL o Verilog. La descripción de una aplicación utilizando HDLs no consiste simplemente en escribir un código sin errores sintácticos, el proceso corresponde más al diseño de un circuito a nivel RTL (*Register-Transfer Level*), en el cual se describen ciclo a ciclo de reloj todas las transferencias de datos entre los registros que componen el diseño, por lo cual la implementación de una aplicación sobre una FPGA es una tarea que consume bastante tiempo y además requiere conocimientos de hardware digital.

Actualmente existen dos enfoques para tratar de reducir el tiempo de diseño requerido en los diseños sobre FPGA:

La primera estrategia es usar compiladores CtoHDL tales como: Vivado® Design Suite (Xilinx®), Catapult Product Family Overview (Calypto Design System, Inc.), C to Verilog, ROCCC 2.0 (Jacquard Computing, Inc.), Nios® II C-to-Hardware Acceleration (Altera®) e Impulse CoDeveloper C-to-FPGA Tools (Impulse®), en los que a partir de un código C generan una descripción en un HDL (ej. VHDL o Verilog) sin necesidad de tener que realizar manualmente el RTL. En este tipo de compiladores existe una correlación directa entre: la cantidad de restricciones que tienen en cuanto al subset ANSI-C, el esfuerzo para modificar el código en C y el número de tipos de kernel soportados [39]. La otra estrategia para facilitar el uso de las FPGA usando lenguajes de alto nivel, dentro de estos lenguajes se encuentran los denominados *Like-C* como Mitrion-C (Mitrionics®). En noviembre de 2011, Altera® abrió la posibilidad de que sus FPGA sean programadas usando el estandar openCL [40]. También existen lenguajes emergentes hardware-software, como el propuesto en 2011 por Dave [41]. A pesar de su necesidad, estos lenguajes todavía no logran gran penetración en el mercado, debido a que su uso todavía compromete la eficiencia de los diseños. Las investigaciones con respecto a la compilación CtoHDL y los lenguajes de alto nivel se encuentran activas [42],[43],[44],[45] y algunos resultados han sido positivos [46], pero la posibilidad de tener acceso a todas las potencialidades de las FPGA

desde un lenguaje de alto nivel todavía está en desarrollo.

4 El hardware gráfico

Las GPGPUs son dispositivos que constan de una gran cantidad de núcleos programables en los que se ejecuta un software, sobre un flujo de datos, en forma de SIMD (Single Instruction Multiple Data). Hasta el año 2007 fueron programadas utilizando lenguajes de programación gráfica y su uso estuvo centrado en aplicaciones de este tipo; a partir del año 2008 empezaron a aparecer una serie de herramientas que han facilitado su programación [2] y por tal razón empezaron a utilizarse en computación de propósito general, de ahí que actualmente se les conozca con el nombre de General Purpose Graphics Processing Units (GPGPU).

Una GPGPU está dividida en una red de bloques y cada bloque consta de varios hilos (512 en las últimas GPGPUs de NVIDIA® [3]). El uso de hilos contribuye a optimizar el uso de la memoria: mientras un hilo está esperando por datos, otro hilo está siendo ejecutado. Cada bloque de hilos tiene su propio conjunto de registros y cada bloque de red tiene un espacio de memoria compartida, a la que se puede acceder tan rápido como se accede a un registro, adicionalmente hay un memoria compartida (con 4GB en las ultimas GPGPUs de NVIDIA®), que tiene una latencia de dos ordenes de magnitud mayor que los registros [3]. Una GPGPU es una tarjeta independiente que se conecta a un sistema de computo a través de un puerto PCI express, lo que limita la transmisión de datos al ancho de banda de este puerto.

4.0.1 Las GPGPUs y la computación de alto rendimiento El rendimiento y el ancho de banda de las GPGPUs cada vez esta más por encima de los sistemas monoprocesador, como se observa en la Figura 4. Actualmente las GPGPU alcanzan un rendimiento teórico de más de un Teraflops en precisión sencilla y más de medio en precisión doble [2],[47]. Así mismo, las GPGPUs ofrecen una mejor relación rendimiento/potencia y rendimiento/costo [2],[48],[49] lo que explica el gran interés que esta arquitectura ha despertado en las investigaciones relacionadas con el cómputo intensivo.

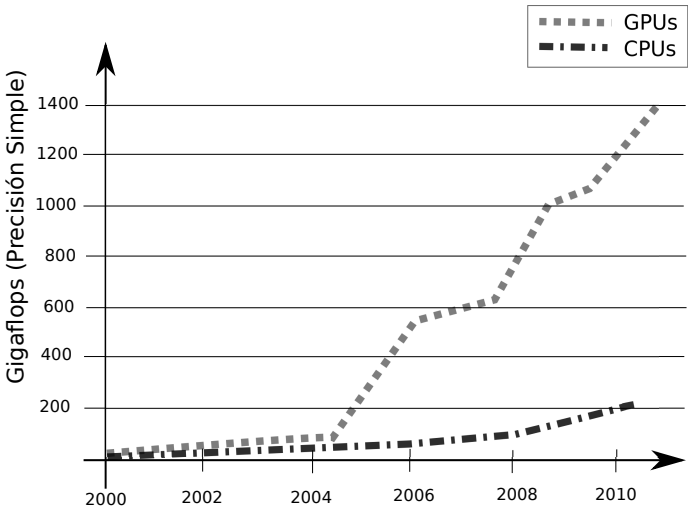

Figura 4: Rendimiento en precisión simple de las GPUs y las CPUs [2]

Las GPGPUs son una arquitectura que ofrecen características en favor de la implementación de la migración sísmica:

- Las GPGPUs están diseñadas para tomar ventaja del paralelismo de las aplicaciones y dado el gran potencial de paralelismo que tiene la migración sísmica, esta arquitectura resulta ser una de las más atractivas para la implementación de esta aplicación [14],[3].
- Las GPGPUs incluyen una unidad de procesamiento que soporta operaciones vectorizadas con números tanto en precisión simple como en presición doble [49] y su rendimiento en precisión sencilla es más de doble que en precisión doble [47] y cómo se mencionó anteriormente este tipo de aplicación no requiere números en precisión doble.
- La tasa de crecimiento en el rendimiento de las GPGPUs es de $4x$ anualmente, mientras que la tasa de crecimiento en el rendimiento de las CPUs es de $1{,}4x$ anualmente [50].

Existe una gran número de investigaciones relacionadas con el desarrollo de soluciones computacionales basadas en GPGPU y muchas de

estas empiezan a aparecer en la literatura. Dentro de las más importantes se encuentran: solución de ecuaciones simultáneas [51], diferentes tipos de transformadas [52],[53],[54], soluciones de la ecuación de onda 2D y 3D [12], [55], comunicaciones [56], simulaciones de dinámica molecular [57], entre otras.

4.1 Lenguajes de Programación para las GPGPUs

Cómo se mencionó anteriormente las GPUs sólo podían ser programadas usando lenguajes gráficos cómo OpenGL. Después, para permitir el uso general de las GPUs AMD® lanzó su programa Stream SDK y NVIDIA® publicó su lenguaje CUDA. Sin embargo, actualmente tanto las GPUs de AMD® como las de NVIDIA® soportan openCL. Otro lenguaje que se puede usar en ambos tipos de GPGPUs es DirectCompute [58], de Microsoft®, el cual permite a los desarrolladores aprovechar la extraordinaria capacidad de procesamiento paralelo de las GPUs NVIDIA® para crear aplicaciones de cálculo altamente eficientes. En este momento las GPUs de NVIDIA® están dominando la computación de propósito general, gracias a la acogida que ha tenido su lenguaje de alto nivel CUDA.

5 Implementación de la Migración Sísmica sobre FPGAs y GPGPUs

En esta sección se revisan las diferentes implementaciones de la migración sísmica tanto sobre FPGAs como sobre GPGPUs. De igual manera se revisan cómo estas investigaciones han abordado los problemas computacionales que presenta esta aplicación.

5.1 Migración de Kirchhoff, implementada sobre una Virtex II Pro (2004)

En el año 2004 en la Universidad de Texas A&M se realizó la primera implementación del proceso de migración sobre una FPGA [11], en este

trabajo se implementó la Migración de Kirchhoff Pre-apilada en Tiempo en la plataforma Xilinx Virtex II Pro. La FPGA fue conectado a una estación de trabajo a través del bus PCIe, esta plataforma (llamada SPACE) contiene dos tipos de memoria: Dos RAM *static dual-port* que hacen las veces de memoria caché de rápido acceso y cuatro módulos DDR-RAM (*Double Data Rate Ramdom Access Memory*).

5.1.1 Aportes del trabajo En este trabajo se busca mejorar el tiempo que tarda la FPGA en realizar una raiz cuadrada utilizando un módulo CORDIC [59]. En la migración de Kirchhoff se requieren realizar tres operaciones matemáticas: sumas, multiplicaciones y raices cuadradas. De las tres operaciones requeridas, la raíz cuadrada es el módulo más lento en hardware con una latencia de al menos diez veces más que un multiplicador *full pipeline*. Gracias a que algunos datos utilizados en la migración sísmica están acotados (por ejemplo los valores de tiempo no superan los 16 segundos) en este trabajo se implementó un módulo CORDIC *full pipeline* para realizar la raíz cuadrada. Este módulo primero convierte los datos de punto flotante a punto flotante alineado (los exponentes son iguales), los exponentes se mantienen constantes y las dos mantisas alimentan una unidad CORDIC para realizar la raiz cuadrada.

5.1.2 Resultados

- Con una frecuencia de trabajo de 50MHz se logró una aceleración de $15{,}6x$ comparada con un 2.4GHz Pentium 4.
- En este trabajo se realizó una comparación de los errores cometidos al usar el módulo CORDIC y un programa en C (precisión doble) y los errores no tienen mayores repercusiones en los patrones de las imágenes generadas.

5.2 Diferencias finitas en el dominio del tiempo, implementada sobre una Xilinx ML401 (2005)

La migración sísmica consiste en la solución de una ecuación de onda y las diferencias finitas es uno de los métodos más utilizados en este proceso. Esta implementación [60] es una versión mejorada de [11] y fue realizada por el mismo grupo de investigación.

5.2.1 Aportes del trabajo En este trabajo se aborda el problema relacionado con el ancho de banda entre la memoria y la FPGA, el cual impide que se puede sacar todo el potencial de las FPGAs. En este trabajo se proponen dos soluciones para mejorar el ancho de banda: reescribir la aplicación con el propósito de disminuir la cantidad de datos que requieren ser leídos y el diseño de una nueva arquitectura para el manejo de la memoria. Las ecuaciones de onda se pueden representar como un sistema de primer orden en forma de derivada, pero también se pueden presentar de segundo orden sin perder generalidad. La representación de las ecuaciones de onda como ecuaciones de segundo orden trae un beneficio en una implementación basada en FPGAs pues ofrece un mejor rendimiento en el acceso a la memoria. En este trabajo se describe la ecuación de onda usando ecuaciones de segundo orden con el propósito de mejorar el uso de ancho de banda de la memoria. Otra propuesta es el uso de las memorias DDR-SDRAM (las cuales se pueden encontrar en las plataformas reconfigurables), éstas tienen la posibilidad de ofrecer un alto *throughput* a un relativo bajo precio. Basados en la dependencia de datos que tiene la solución de la ecuación de onda, en este trabajo se introduce una interfaz entre los módulos de computo y la memoria *onboard* usando los módulos onchip de memoria de la FPGA. Este módulo permite utilizar el ancho de banda de la memoria *onboard* más eficientemente que los trabajos anteriores y puede ser implementado en la mayoría de sistemas de desarrollo basados en FPGAs. Básicamente en este trabajo se propone una nueva manera de acceder a los datos de la grid, almacenando algunos valores dentro de la FPGA con el fin de evitar el acceso a la memoria *onboard*. Los datos en la FPGA se almacenan en dos FIFOs en cascada, estos buffers se implementan en los módulos de memoria *onchip*. Los datos del grid van entrando por un puerto de

entrada al fondo del primer FIFO y se va descartando el ultimo del segundo FIFO, de tal forma que en el FIFO se tienen disponibles todos los datos necesarios para cada computo dentro de la FPGA.

5.2.2 Resultados En este trabajo se obtienen aceleraciones de hasta $8{,}26x$ comparado con un Intel P4 3.0 GHz.

5.3 *Downward continued migration*, implementada sobre la plataforma MAX1 (2007)

En [61] buscan mejorar el rendimiento de la FPGA y optimizar el uso del puerto PCIe reduciendo la cantidad de bits con la cual se representan los datos, para tal fin, se implementa una parte de la *Downward Continued Migration* utilizando un número menor de bits a los empleados anteriormente. La implementación se realizó en una plataforma MAX1 [62] la cual contiene una FPGA Virtex-4 FX100.

5.3.1 Aportes del Trabajo Se simuló todo el proceso de migración sísmica y se seleccionaron aquellas partes que se encuentran acotadas, estas partes se implementaron en punto fijo con 10, 6 y 5 bits en la parte decimal en lugar de utilizar números de coma flotante.

5.3.2 Resultados Cómo se observa en la Figura 5, las gráficas del subsuelo (para los diferentes números de bits en la parte decimal) son visualmente idénticas.

En este trabajo se alcanzaron aceleraciones de hasta $42x$ comparados con una implementación software (precisión doble) en un 2.8GHz AMD® Opteron-based PC con 12 GB de RAM.

5.4 *Downward continued migration*, implementada sobre la plataforma MAX1 (2008)

En [63] nuevamente se aborda el problema del computo desde la perspectiva de la representación de los datos. El objetivo de esta investigación

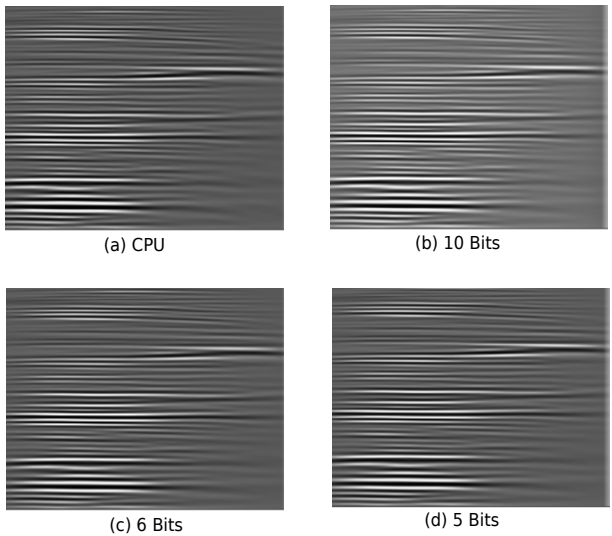

Figura 5: Implementación sobre la FPGA con diferente cantidad de bits en la parte decimal [61]

es encontrar las partes de la migración que pueden ser desarrolladas en punto fijo y determinar cuál es el número menor de bits requeridos para conseguir imagenes del subsuelo de aceptable calidad. En este trabajo se utiliza como caso de estudio la Ecuación 3, esta ecuación exponencial compleja es común a todas las clases de *Downward continued migration*.

$$U(w, k_s, k_g, z + \triangle z) = exp^{-iwk\left(\sqrt{1-\frac{vk_g}{w}}+\sqrt{1-\frac{vk_s}{w}}\right)} U(w, k_s, k_g, z) \quad (3)$$

5.4.1 Aportes del Trabajo

- En este trabajo se usan los resultados de las aproximaciones matemáticas de *Lee 2005* [64] que permiten mapear eficientemente sobre FPGAs, raíces cuadradas y funciones trigonométricas.

- Se desarrollo una herramienta que permite calcular el número menor de bits requerido para el caso propuesto.
- Se realizaron pruebas con coma flotante de menos de 32 bits.

5.4.2 Resultados

- Se establecieron como cantidad mínima de bits 12 y 16 para la raiz cuadrada y las funciones trigonométricas respectivamente.
- Con puntos fijos de hasta 12 bits se construyen imágenes con los mismos patrones que las generadas en software (precisión doble).
- Se estableció que se pueden realizar las operaciones en coma flotante de 22 bits, 6 para el exponente y 16 para la mantiza.
- Las imagenes generadas fueron comparadas por medio de filtros de error con imagenes generadas en precisión doble y se obtuvieron los mismos patrones.
- Con dos cores implementados en la FPGA se logró una aceleración de 13,7x comparado con un Intel Xeon 1.86GHz.
- Con este nuevo formato de números se reduce considerablemente el área empleada dentro de la FPGA.
- Se estableció que con el suficiente ancho de banda se pueden poner 6 cores dentro de la FPGA y lograr una aceleración de hasta 48x.

5.5 Compresión de Datos Sísmicos: Tesis de Maestría (2009)

En la tesis de maestria [65] se aborda el problema de mejorar el ancho de banda entre la memoria y el dispositivo de computo utilizando compresión de datos. En esta investigación, se utiliza el algoritmo de compresión propuesto por T. Røsten [66] para comprimir los datos sísmicos antes de ser enviados a la GPGPUs.

5.5.1 Aportes El objetivo de esta investigación es revisar si la transmisión de datos comprimidos desde una memoria principal hasta la memoria de la GPU y su posterior descompresión resulta en un tiempo más corto que el envío de la misma cantidad de datos sin comprimir. Para tal fin se implementa dicho algoritmo sobre una GPGPU Quadro FX 5800 de NVIDIA®.

5.5.2 Resultados Los resultados no fueron positivos, el envío de datos comprimidos a la memoria de la GPU y su posterior descompresión consume más tiempo que un envío de datos sin comprimir.

5.6 Migración Reversa en el Tiempo (RTM) sobre un cluster Maxeler que contiene cuatro Virtex 6 (2011)

En [1] se detalla el procedimiento para implementar la RTM dentro de una FPGA. El cluster utilizado es de la empresa Maxeler, el cual contiene 8 CPUs y cuatro Virtex 6.

5.6.1 Aportes La comunicación entre FPGAs es realizada por medio del puerto *MaxRing* y por medio de éste, se obtuvo un ancho de banda que no afectó el computo.

En esta investigación se utilizan las herramientas CAD ofrecidas por Maxeler, las cuales permiten introducir la aplicación descrita en Java. El procedimiento utilizado para la implementación sobre la FPGA es:

1. Modificar el kernel: simplificar, aplanar, convertir los lazos dinámicos en lazos estáticos.

2. El kernel modificado se compila en un Diagrama de flujo de Datos (data-flow graph). Gracias a que estos diagramas son estáticos es posible determinar los caminos críticos y la cantidad de operaciones.

3. Se convierte el Diagrama de Flujo de datos en una *synchronous data-flow machine*. Cada nodo en esta máquina es una operación

(punto fijo o flotante) un multiplexor o un buffer. Idealmente cada operación debe durar un ciclo de reloj y las operaciones complejas se les aplica pipeline a una rata de un ciclo por sub-operación.

4. Repetir los pasos anteriores para optimizar.

5.6.2 Resultados

- Este trabajo logró una aceleración $70x$ comparado contra un *dual-processor quad-core 2.9-GHz Intel Nehalem.*
- Adicionalmente en este trabajo se calculan los consumos de potencia: $500W$ para la FPGA y $700W$ para para el servidor Nehalem. La CPU consumiría $14000W$ si se acelera su proceso $40x$.
- En este trabajo también se implementa la RTM sobre una GPU, la cual obtuvo una aceleración de $5x$ comparada con la CPU.

5.7 Migración en Tiempo inverso sobre tres arquitecturas diferentes: Cell/Be, GPGPUs y FPGAs (2011)

En [14] se compara la implementación de la RTM sobre tres arquitecturas diferentes: Cell/Be, GPGPUs y FPGAs.

5.7.1 Aportes En la implementación sobre la GPGPU se comprimen los datos antes de ser enviados al disco con el propósito de ahorrar espacio. El volumen de datos es almacenado en un buffer mientras se están enviando al disco. Tener el volumen de datos almacenados en un buffer permite desarrollar de manera concurrente el envío de datos con el computo del siguiente dato. La cantidad de memoria necesaria para un disparo supera fácilmente la capacidad de memoria disponible en una GPGPU (4 GiB en este caso), la aplicación es particionada en tantos subdominios como GPGPUs existen. En la implementación sobre la FPGA, este trabajo se enfoca en el reuso de los datos, para tal fin se utilizan los tres niveles de memoria con los que cuenta la plataforma empleada. Adicionalmente se usa la compresión y descompresión de datos y se sugiere el uso de punto fijo en lugar de punto flotante.

5.7.2 Resultados Todas las implementaciones estuvieron cerca de acelerar el proceso por un factor $10x$. Sin embargo los resultados de la GPU estuvieron por encima del desempeño de la implementación sobre la FPGA.

5.8 Resumen de resultados

En la tabla 1 se resumen los aportes y resultados de las implementaciones de la Migración Sísmica sobre FPGAs y GPGPUs encontradas durante la presente búsqueda.

Tabla 1: Tabla de Resultados

Año	Aplicación	Dispositivo	Aportes	Resultados
2004	Migración de Kirchhoff Preapilada en Tiempo	Virtex 2 Pro	Uso del Módulo CORDIC para implementar la raiz cuadrada	15,6x vs 2.4GHz Pentium 4
2005	Diferencias finitas en el dominio del tiempo	Xilinx ML401	Reescribir la ecuación de onda usando ecuaciones de segundo orden. Interfaz de FIFOs implementados en la memoria on-chip con el fin de disminuir las veces que se tiene que leer la memoria	8,26x vs Intel P4 3.0 GHz
2007	*Downward continued migration*	MAX1 (Virtex-4)	Uso de punto fijo	42x vs 2.8GHz AMD® Opteron
2008	*Downward continued migration*	MAX1 (Virtex-4)	Uso de punto fijo. Aproximaciones LEE. Herramienta para calcular el número de menor de bits en la parte decimal cuando se usa punto fijo	13,7x vs Intel Xeon 1.86GHz
2009	Algoritmo de compresión Røsten	GPGPUs	Compresión de datos sísmicos	Negativos
2011	Migración Reversa en el Tiempo	Virtex 6 (Cuatro)	Herramientas CAD de Maxeler. Puerto MaxRing para la comunicación entre FPGAs	70x vs Intel Nehalem 2.9-GHz
2011	Migración Reversa en el Tiempo	GPU y FPGAs	Compresión de datos y uso de Punto fijo.	10x vs Xeon E5460 2.5 GHz

6 Conclusiones

A continuación listamos algunas lineas de investigación que consideramos importantes, con el fin de continuar optimizando el uso de las FPGA y las GPGPU en el área del procesamiento de datos sísmicos:

- El uso de la compresión de datos sísmicos como estrategia para aumentar la velocidad de transferencia de las trazas sísmicas: En este momento la velocidad a la cual se ingresan los datos sísmicos al dispositivo no permite mantener aprovechar el 100 % del potencial de cómputo de estas dos arquitecturas. El uso de la compresión para aumentar la velocidad de transferencia requiere que el tiempo de envío de los datos comprimidos más el tiempo de descompresión sea menor que el tiempo de envío de los datos sin comprimir. Al respecto, creemos que esta estrategia puede llegar a ser más viable en las FPGA, pues en estos dispositivos las operaciones adicionales de descompresión pueden llegar a convertirse en más hardware y no en más tiempo, gracias a la flexibilidad que ofrecen las FPGA para aplicar la técnica de segmentación.

- El uso de compiladores CtoHDL: El uso de las FPGA en este contexto se ha visto frenado por la dificultad que representa implementar a nivel RTL el todo algoritmo de migración (las investigaciones han implementado sólo partes de él), una estrategia que podria promover el uso de las FPGA, sería el uso de los compiladores CtoVHDL, sin embargo, se hace necesario evaluar este tipo de herramientas, a fin de determinar la eficiencia de las implementaciones en términos de área, *throughput* y *throughput*/área.

- La optimización de la implementación de operaciones matemáticas sobre estas dos tecnologías seguirán siendo claves para mejorar los tiempos de computo de la MS. En este sentido, se requieren investigaciones que giren en torno a la optimización de las operaciones matemáticas requeridas por la RTM, por ser el tipo de migración que ofrece una mayor resolución en la imágenes generadas. De igual manera las investigaciones deben contemplar la posibilidad de uti-

lizar diferentes formatos para representar los datos sísmicos (como por ejemplo punto fijo y coma flotante menor a 32 bits).

Agradecimientos

La presente revisión fue realizada al interior del grupo de investigación CPS de la Universidad Industrial de Santander, los autores agradecen a su director el Dr. Oscar Reyes y demás investigadores por sus valiosos aportes.
Este trabajo fue financiado por el convenio de cooperación tecnológica UIS-ICP número 005 de 2003 y el Departamento Administrativo de Ciencia, Tecnología e Innovación-Colciencias, Programa Nacional de formación de Investigadores Generación del Bicentenario.

Referencias

[1] O. Lindtjorn, R. G. Clapp, O. Pell, and M. J. Flynn, "Beyond Traditional Microprocessors for Geoscience High-Performance Computing Applications," *Ieee Micro*, vol. 31, no. 2, pp. 41–49, 2011. 262, 283

[2] A. Brodtkorb, "Scientific Computing on Heterogeneous Architectures," Ph.D. dissertation, University of Oslo, 2010. [Online]. Available: http://babrodtk.at.ifi.uio.no/files/publications/brodtkorb_phd_thesis.pdf 263, 275, 276

[3] R. G. Clapp, H. Fu, and O. Lindtjorn, "Selecting the right hardware for reverse time migration," *The Leading Edge*, vol. 29, no. 1, p. 48, 2010. [Online]. Available: http://link.aip.org/link/LEEDFF/v29/i1/p48/s1&Agg=doi 263, 273, 275, 276

[4] J. Cabezas, M. Araya-Polo, I. Gelado, N. Navarro, E. Morancho, and J. M. Cela, "High-Performance Reverse Time Migration on GPU," *2009 International Conference of the Chilean Computer Science Society*, pp. 77–86, 2009. 263, 268

[5] R. Abdelkhalek, H. Calandra, O. Coulaud, J. Roman, and G. Latu, "Fast seismic modeling and Reverse Time Migration on a GPU cluster," *2009 International Conference on High Performance Computing & Simulation*, pp. 36–43, 2009. 263, 268

[6] V. K. Madisetti and D. G. Messerschmitt, "Seismic migration algorithms using the FFT approach on the NCUBE multiprocessor," *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pp. 894–897, 1988. 264

[7] S. Yerneni, S. Phadke, D. Bhardwaj, S. Chakraborty, and R. Rastogi, "Imaging subsurface geology with seismic migration on a computing cluster," *Current Science*, vol. 88, no. 3, pp. 468–478, 2005. 264, 265, 266, 267

[8] V. K. Madisetti and D. G. Messerschmitt, "Seismic migration algorithms on parallel computers," *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1642–1654, 1991. 265

[9] J. F. Claerbout, "Basic Earth Imaging," p. 220, 2010. [Online]. Available: http://sepwww.stanford.edu/sep/prof/bei11.2010.pdf.2011.pdf 266, 267

[10] S. H. Gray, J. Etgen, J. Dellinger, and D. Whitmore, "Seismic migration problems and solutions," *Geophysics*, vol. 66, no. 5, p. 1622, 2001. 266, 267

[11] C. He, M. Lu, and C. Sun, "Accelerating Seismic Migration Using FPGA-Based Coprocessor Platform," *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 207–216, 2004. 267, 277, 279

[12] D. Brandao, M. Zamith, E. Clua, A. Montenegro, A. Bulcao, D. Madeira, M. Kischinhevsky, and R. C. P. Leal-Toledo, "Performance Evaluation of Optimized Implementations of Finite Difference Method for Wave Propagation Problems on GPU Architecture," *2010 22nd International Symposium on Computer Architecture and High Performance Computing Workshops*, pp. 7–12, 2010. 268, 277

[13] E. Baysal, "Reverse time migration," *Geophysics*, vol. 48, pp. 1514–1524, 1983. 268

[14] M. Araya-polo, J. Cabezas, M. Hanzich, M. Pericas, I. Gelado, M. Shafiq, E. Morancho, N. Navarro, M. Valero, and E. Ayguade, "Assessing Accelerator-Based HPC Reverse Time Migration," *Electronic Design*, vol. 22, no. 1, pp. 147–162, 2011. 268, 270, 271, 276, 284

[15] P. Farmer, S. Gray, G. Hodgkiss, A. Pieprzak, and D. Ratcliff, "Structural Imaging : Toward a Sharper Subsurface View," *Oilfield Review*, vol. 1, no. 1, pp. 28–41, 1993. 268, 269, 270

[16] A. Albertin, J. Kapoor, R. Randall, and M. Smith, "La era de las imágenes en escala de profundidad," *Oilfield Review*, vol. 14, no. 1, pp. 2–17, 2002. 268, 269

[17] S. Abreo and A. Ramirez, "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," *Ingeniería e investigación e investigación*, vol. 30, no. 1, pp. 64–70, 2010. 270, 273

[18] M. Flynn, R. Dimond, O. Mencer, and O. Pell, "Finding Speedup in Parallel Processors," *2008 International Symposium on Parallel and Distributed Computing*, pp. 3–7, 2008. 270

[19] Xilinx Inc., "Xilinx Intellectual Property." [Online]. Available: http://www.xilinx.com/products/intellectual-property/ 272

[20] Altera Corporation, "Altera: Intellectual Property & Reference Designs." [Online]. Available: http://www.altera.com/products/ip/ 272

[21] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, 2002. [Online]. Available: http://portal.acm.org/citation.cfm?doid=508352.508353 272

[22] I. Skliarova and V. Sklyrov, "Recursion in reconfigurable computing: A survey of implementation approaches," *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pp. 224–229, 2009. 272

[23] A. Gomperts, A. Ukil, and F. Zurfluh, "Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 1, pp. 78–89, 2011. 272

[24] Y. Lee and S.-B. Ko, "FPGA Implementation of a Face Detector using Neural Networks," in *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on*, May 2006, pp. 1914–1917. 272

[25] E. A. Zuraiqi, M. Joler, and C. G. Christodoulou, "Neural networks FPGA controller for reconfigurable antennas," in *Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE*, 2010, pp. 1–4. 272

[26] V. Gupta, K. Khare, and R. P. Singh, "FPGA Design and Implementation Issues of Artificial Neural Network Based PID Controllers," in *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on*, 2009, pp. 860–862. 272

[27] K. Puttegowda, W. Worek, N. Pappas, A. Dandapani, P. Athanas, and A. Dickerman, "A run-time reconfigurable system for gene-sequence searching," in *VLSI Design, 2003. Proceedings. 16th International Conference on*, 2003, pp. 561–566. 272

[28] I. a. Bogdán, J. Rivers, R. J. Beynon, and D. Coca, "High-performance hardware implementation of a parallel database search engine for real-time peptide mass fingerprinting." *Bioinformatics (Oxford, England)*, vol. 24, no. 13, pp. 1498–1502, 2008. 272

[29] S. Baghel and R. Shaik, "FPGA implementation of Fast Block LMS adaptive filter using Distributed Arithmetic for high throughput," in *Communications and Signal Processing (ICCSP), 2011 International Conference on*, 2011, pp. 443–447. 272

[30] M. Rawski, P. Tomaszewicz, H. Selvaraj, and T. Luba, "Efficient Implementation of digital filters with use of advanced synthesis methods targeted FPGA architectures," in *Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, 2005, pp. 460–466. 272

[31] Y. Wang and Y. Shen, "Optimized FPGA Realization of Digital Matched Filter in Spread Spectrum Communication Systems," *Computer and Information Technology, IEEE 8th International Conference on*, pp. 173–176, 2008. 272

[32] R. Tessier and W. Burleson, "Reconfigurable Computing for Digital Signal Processing A Survey," *Journal of VLSI Signal Processing*, vol. 28, pp. 7–27, 2001. 272

[33] R. Sinnappan and S. Hazelhurst, "A Reconfigurable Approach to Packet Filtering," in *Field-Programmable Logic and Applications*, ser. Lecture Notes in Computer Science, G. Brebner and R. Woods, Eds. Springer Berlin / Heidelberg, 2001, vol. 2147, pp. 638–642. 272

[34] Y. H. Cho and W. H. Mangione-Smith, "Deep network packet filter design for reconfigurable devices," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 2, pp. 21–26, 2008. 272

[35] X. Tian and K. Benkrid, "Design and implementation of a high performance financial Monte-Carlo simulation engine on an FPGA supercomputer," in *ICECE Technology, 2008. FPT 2008. International Conference on*, 2008, pp. 81–88. 272

[36] N. A. Woods and T. VanCourt, "FPGA acceleration of quasi-Monte Carlo in finance," in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, 2008, pp. 335–340. 272

[37] D. A. Hauck Scott, *Reconfigurable computing. The theory and practice of FPGA-BASED computing.* ELSEVIER - Morgan Kaufmann, 2008. 273

[38] H. Fu, W. Osborne, R. G. Clapp, O. Mencer, and W. Luk, "Accelerating seismic computations using customized number representations on FPGAs," *EURASIP J. Embedded Syst.*, vol. 2009, pp. 1–13, 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/382983 273

[39] A. J. Virginia, Y. D. Yankova, and K. L. M. Bertels, "An empirical comparison of ANSI-C to VHDL compilers : Spark, Roccc and DWARV," in *Anual Workshop on Circuits, Systems and Signal Processing (ProRISC),*, Veldhoven, Netherlands, 2007, pp. 388–394. 274

[40] Altera Corporation, "Implementing FPGA Design with the OpenCL Standard," p. 9, 2012. [Online]. Available: http://www.altera.com/literature/wp/wp-01173-opencl.pdf 274

[41] N. Dave, "A Unified Model for Hardware/Software Codesign," Ph.D. dissertation, Massachusetts Institute Of Technology, 2011. 274

[42] R. Sánchez Fernández, "Compilación C a VHDL de códigos de bucles con reuso de datos," Tesis, Universidad Politécnica de Cataluña, 2010. 274

[43] Y. Yankova, K. Bertels, S. Vassiliadis, R. Meeuws, and A. Virginia, "Automated HDL Generation: Comparative Evaluation," *2007 IEEE International Symposium on Circuits and Systems*, pp. 2750–2753, May 2007. 274

[44] P. I. Necsulescu, "Automatic Generation of Hardware for Custom Instructions," Ph.D. dissertation, Ottawa, Canada, 2011. 274

[45] P. I. Necsulescu and V. Groza, "Automatic Generation of VHDL Hardware Code from Data Flow Graphs," *6th IEEE International Symposium on Applied Computational Intelligence and Informatics*, pp. 523–528, 2011. 274

[46] J. Bier and J. Eyre, "BDTI Study Certifies High-Level Synthesis Flows for DSP-Centric FPGA Designs," *Xcell Journal Second*, no. 71, pp. 12–17, 2010. 274

[47] NVIDIA Tesla, "GPU Computing revolutionizing High Performance Computing," 2010. [Online]. Available: http://www.nvidia.com/docs/IO/100133/tesla-brochure-12-lr.pdf 275, 276

[48] A. Brodtkorb, C. Dyken, T. R. Hagen, and J. M. Hjelmervik, "State-of-the-art in heterogeneous computing," *Scientific Programming*, vol. 18, pp. 1–33, 2010. 275

[49] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007. 275, 276

[50] F. Warg, J. Nilsson, M. Ekman, and At An In-depth Look, "An In-Depth Look at Computer Performance Growth," *SIGARCH Comput. Archit. News*, vol. 33, pp. 144–147, 2005. 276

[51] W. Lei, Z. Yunquan, Z. Xianyi, and L. Fangfang, "Accelerating Linpack Performance with Mixed Precision Algorithm on CPU+GPGPU Heterogeneous Cluster," in *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, ser. CIT '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1169–1174. [Online]. Available: http://dx.doi.org/10.1109/CIT.2010.212 277

[52] S. Romero, M. A. Trenas, E. Gutierrez, and E. L. Zapata, "Locality-improved FFT implementation on a graphics processor," in *Proceedings of the 7th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*, ser. ISCGAV'07. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007, pp. 58–63. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364592.1364602 277

[53] Y. Su and Z. Xu, "Parallel implementation of wavelet-based image denoising on programmable PC-grade graphics hardware," *Signal Process.*, vol. 90, no. 8, pp. 2396–2411, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2009.06.019 277

[54] J. Lobeiras, M. Amor, and R. Doallo, "FFT Implementation on a Streaming Architecture," in *Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, ser. PDP '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 119–126. [Online]. Available: http://dx.doi.org/10.1109/PDP.2011.31 277

[55] P. Micikevicius, "3D Finite Difference Computation on GPUs using CUDA 2701 San Tomas Expressway," *Cell*, pp. 0–5, 2009. 277

[56] L. Jacquin, V. Roca, J.-L. Roch, and M. Al Ali, "Parallel arithmetic encryption for high-bandwidth communications on multicore/GPGPU platforms," in *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, ser. PASCO '10. New York, NY, USA: ACM, 2010, pp. 73–79. [Online]. Available: http://doi.acm.org/10.1145/1837210.1837223 277

[57] S. Hudli, S. Hudli, R. Hudli, Y. Subramanian, and T. S. Mohan, "GPGPU-based parallel computation: application to molecular dynamics problems," in *Proceedings of the Fourth Annual ACM Bangalore Conference*, ser. COMPUTE '11. New York, NY, USA: ACM, 2011, pp. 10:1—-10:8. [Online]. Available: http://doi.acm.org/10.1145/1980422.1980432 277

[58] NVIDIA, "DirectCompute para NVIDIA." [Online]. Available: http://developer.nvidia.com/directcompute 277

[59] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field*

programmable gate arrays, ser. FPGA '98. New York, NY, USA: ACM, 1998, pp. 191–200. [Online]. Available: http://doi.acm.org/10.1145/275107.275139 278

[60] C. He, W. Zhao, and M. Lu, "Time Domain Numerical Simulation for Transient Waves on Reconfigurable Coprocessor Platform," in *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE Computer Society, 2005, pp. 127–136. 279

[61] O. Pell and R. G. Clapp, "Accelerating subsurface offset gathers for 3D seismic applications using FPGAs," *SEG Technical Program Expanded Abstracts*, vol. 26, no. 1, pp. 2383–2387, 2007. 280, 281

[62] Maxeler Technologies, "Maxeler: Complete Acceleration Solutions." [Online]. Available: http://www.maxeler.com/content/solutions/ 280

[63] H. Fu, W. Osborne, R. G. Clapp, and O. Pell, "Accelerating Seismic Computations on FPGAs From the Perspective of Number Representations," *70th EAGE Conference & Exhibition*, no. June 2008, pp. 9–12, 2008. 280

[64] D.-U. Lee, A. Abdul Gaffar, O. Mencer, and W. Luk, "Optimizing Hardware Function Evaluation," *IEEE Trans. Comput.*, vol. 54, no. 12, pp. 1520–1531, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1098521.1098595 281

[65] D. Haugen, "Seismic Data Compression and GPU Memory Latency," Master Thesis, Norwegian University of Science and Technology, 2009. 282

[66] T. Rø sten, T. A. Ramstad, and L. Amundsen, "Optimization of sub-band coding method for seismic data compression," *Geophysical Prospecting*, vol. 52, no. 5, pp. 359–378, 2004. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2478.2004.00422.x 282