http://dx.doi.org/10.20983/culcyt.2018.3.10

Implementación de Algoritmos de Procesamiento Digital de Señales en Hardware Paralelo: Artículo de revisión

Gabriel Bravo Martínez

Jesús Martin Silva Aceves

Soledad Vianey Torres Argüelles

Francisco Javier Enríquez Aguilera

Instituto de Ingeniería y Tecnología

Universidad Autónoma de Ciudad Juárez

RESUMEN

Sobre el procesamiento digital de señales con sistemas de computadoras con capacidades genéricas, en su mayoría de un solo procesador multinúcleo

Palabras clave: Procesamiento digital de señales, Algoritmos, Hardware paralelo

Introducción

La enseñanza del procesamiento digital de señales (PDS) hasta la fecha se ha realizado con sistemas de computadoras con capacidades genéricas, es decir, computadoras en su mayoría de un solo procesador (Kehl *et al.*, 2017), el cual es generalmente multinúcleo (Ramalakshmi and Kompala, 2017). El tipo de procesador define la forma en que el algoritmo del PDS sea programado; si el procesador es multinúcleo, el algoritmo se puede implementar en el número de procesos definido por la cantidad de núcleos físicos del procesador, los que llegan a ser incluso dieciocho núcleos en computadoras de escritorio (Intel, 2017), esto se obtiene haciendo uso de interfaces de programación de aplicaciones (API por sus siglas en inglés), como lo es OpenMP (Dagum and Menon, 1998), que aprovecha los sistemas conocidos como de memoria compartida. Los algoritmos utilizados en el área de PDS son en su mayoría altamente realizables en paralelo. Para aplicar la característica de paralelismo de los algoritmos en sistemas de cómputo genérico, la enseñanza se ha apoyado en diferentes técnicas de programación en paralelo, como la predicción de saltos (*branch prediction*) (Kalla *et al.*, 2017), recuperación previa (*prefetching*) (Xu *et al.*,

2018), y segmentación (*pipeline*) (Arndt, Linde and Blume, 2015).

Referente a los procesadores multinúcleo, el avance tecnológico ha llevado al incremento en número de núcleos, así como de la generación de unidades de procesamiento gráfico (GPU por sus siglas en inglés) que incluyen una gran cantidad de núcleos CUDA (siglas del inglés de Arquitectura Unificada de Dispositivos de Cómputo) (Yu *et al.*, 2015), haciendo en la implementación en paralelo por hardware especializado la opción más rápida y de mejor desempeño comparada con la implementación en paralelo en hardware convencional. Dentro de los trabajos realizados de implementación de algoritmos de PDS en paralelo por hardware tenemos combinaciones de clúster de CPU, CPU/GPU, GPU (Arndt, Linde and Blume, 2015) y sistemas embebidos, que incluyen CPU, FPGA (matriz de puertas programables) y coprocesadores multinúcleo integrados en una sola tarjeta. Mustafa *et al.* (Mustafa, Shahana and Ahmed, 2015) realizaron la paralelización del algoritmo Doolittle utilizando la herramienta de implementación en paralelo para computadoras con procesadores multinúcleo llamada OpenMP, consiguiendo una aceleración en el desempeño del algoritmo al 50% con matrices del orden de 6000 elementos; el procesador utilizado fue un Intel i5. Por su parte Kena Xu *et al.* (Xu *et al.*, 2016) proponen un algoritmo de planificación de tareas para sistemas multinúcleo con el que se pretende conseguir que las tareas que se asignan a los núcleos del sistema tengan la misma duración, con el fin de optimizar el hardware; dicho estudio reporta que la implementación del algoritmo en un procesador, de cuatro núcleos, mostró una aceleración del proceso y una mayor eficiencia comparado con los algoritmos citados en su trabajo. En cuanto al uso de CUDA, Jiansen li *et al.* (J. Li *et al.*, 2015) presentan una combinación de CPU-CUDA y la plataforma CUDA con dos versiones diferentes de optimización de paralelización para la aceleración del algoritmo de reconstrucción de imagen por resonancia magnética, sus resultados muestran la ventaja de la implementación del algoritmo en la tecnología CUDA comparada con la realizada en CPU. El desarrollo de plataformas embebidas también ha tomado auge con el lanzamiento de la plataforma Parallella, la cual es una computadora de placa única que integra un procesador multinúcleo y un FPGA, con la que Sethakarn Prongnuch *et al.* (Prongnuch and Wiangtong, 2016) realizaron el procesamiento de datos utilizando diferentes combinaciones de los elementos embebidos con el fin de evaluar el desempeño de la misma.

Adaptación de la implementación de algoritmos de PDS en la evolución del HW

Los procesadores son el cerebro del hardware, son quienes se encargan de llevar a cabo las operaciones básicas de cómputo para lograr que el algoritmo que les ocupa se ejecute. La ejecución del procesador está caracterizada por las métricas de desempeño (Mastelic, Brandic and Jaarevic, 2015)(Rinku, 2017) como la velocidad del procesador (Kumashiro *et al.*, 2017), potencia disipada, número de ciclos por instrucción y número de hilos (procesos ejecutados simultáneamente en un solo núcleo o en varios núcleos) que puede realizar el procesador (Hellestrand, 1996). La implementación de algoritmos de PDS en esta arquitectura es reportada en la literatura logrando tiempos de ejecución diez veces más rápidas comparadas con la implementación del algoritmo tradicional (Ramalakshmi and Kompala, 2017). El factor que ha tomado relevancia actualmente es la potencia disipada de los procesadores, que al tener más núcleos se incrementa; existen diversas técnicas para bajar la potencia disipada y mejorar la eficiencia energética; dos técnicas son el escalamiento dinámico de voltaje y frecuencia (DVFS por sus siglas en inglés) y potencia canalizada (PG por sus siglas en inglés) (Chen *et al.*, 2017)(Kanduri, … and 2017, 2017). Con el fin de descargar de trabajo al procesador y aumentar la potencia de ejecución de algoritmos se introdujo el uso de las unidades de procesamiento gráfico, con altas velocidades de procesamiento y gran cantidad de núcleos (480 en GeForce GTX 295). Se ha reportado en la literatura una aceleración en la

ejecución de algoritmos implementados en los GPUs de hasta 1024 veces (sobre matrices de 100x100) (Bozejko, Dobrucki and Walczynski, 2010). Las GPU son tarjetas que consumen alta potencia y son de gran tamaño, lo que las hacen poco atractivas para ciertas áreas, por ejemplo, cuando se desea tener movilidad en los dispositivos de procesamiento (Kurth *et al.*, 2016), o cuando la disminución de la potencia disipada es primordial (Gao, Huang, Z. Wang, *et al.*, 2017). El surgimiento de las computadoras de placa única (SBC por sus siglas en inglés) ha dado lugar a placas con sistemas embebidos que incluyen procesadores multinúcleo, FPGAs (matriz de puertas programables) y coprocesadores multinúcleo con la característica de baja potencia y alta movilidad (Nakata and Ito, 2017)(Mendat *et al.*, 2016)(Prongnuch and Wiangtong, 2016).

El estado del arte referente al cómputo paralelo contiene información limitada con enfoques particulares, por ejemplo, Lajiao Chen *et al*. (L. Chen *et al.*, 2015) realizaron un artículo de revisión acerca del procesamiento paralelo en el área de imágenes de mosaicos (*image mosaicking*), donde exponen las técnicas de procesamiento paralelo más recientes utilizadas en el acoplamiento de las imágenes, recalcando su importancia en la gran cantidad de información que se requiere procesar. Por otra parte, la medición experimental óptica y mecánica requiere de procesar grandes cantidades de datos y a alta velocidad, con el propósito de obtener una respuesta en tiempo real. Qian Kemao *et al*. (Wang and Kemao, 2017)(Wang and Kemao, 2018) cubren en su revisión de literatura varias plataformas y técnicas utilizadas en el cómputo paralelo que se emplean en medición experimental, entre las que destacan el uso de CPUs, GPUs, FPGAs. En esta revisión se expondrán los trabajos más significativos en el área del procesamiento digital de señales implementados en procesadores multinúcleo, con las diferentes técnicas de programación, se cubrirá la implementación en GPUs y sistemas heterogéneos de GPU-CPU-FPGA y finalmente se mostrara los trabajos más recientes en la computadora de placa única Parallella, computadora que unifica tres diferentes tecnologías a baja potencia (Olofsson, Nordström and Ul-Abdin, 2015).

La implementación de algoritmos en paralelo en procesadores multinúcleo

La principal tarea en la implementación de algoritmos en paralelo en hardware es la de acelerar las ejecuciones en comparación con las implementaciones que utilizan técnicas de software. En (Atweh *et al.*, 2018) se presenta una implementación en paralelo del algoritmo de detección de bordes Sobel, realizado en un procesador de 4 núcleos Intel Core i5. La implementación se realizó con diferentes particiones de los datos con el fin de mejorar el desempeño en los diferentes núcleos, logrando una aceleración de un 1.7 que era reportado en las referencias a un 2.49. Entre los factores que afectan el desempeño de las implementaciones en paralelo se encuentra la planificación, por lo que una buena estrategia de planificación es clave en la reducción de tiempo de ejecución en los procesadores multinúcleo, y en consecuencia la reducción de consumo en energía. La partición de datos se puede realizar en forma automática, pero generalmente no es eficiente. En (Jing *et al.*, 2018) proponen un método de partición automática para sistemas multinúcleo, la integración del método en algoritmos como lo es Función y declaración genera una partición de toda la aplicación en muchas sub tareas con la tarea de maximizar el paralelismo en tiempo y espacio, así como disminuir la comunicación entre las tareas. Los resultados experimentales muestran una aceleración del proceso que va de 1.99 con dos núcleos, hasta 5.79 con ocho núcleos. El algoritmo de programación de prioridad y falta de cache en CPU y el algoritmo de programación de prioridad en cambio de contexto en CPU originalmente ofrecen la ventaja de reducción de consumo de energía cerca del 20%. Con el uso de procesos dinámicos de prioridad como el criterio de programación, (Datta and Patel, 2014) lograron un ahorro de energía de ejecución entre un 24.15 a un 52%. En (Hwang and Pedram, 2016) realizaron un estudio comparativo entre dos

algoritmos que reducen el gasto de energía consumido en un servidor virtual multinúcleo. Los algoritmos analizados son el llamado Voltaje dinámico y escalado de frecuencia y el algoritmo Consolidación de CPU. Adicionalmente se presentó un nuevo algoritmo, variante del Consolidación de CPU. La comparación dio hasta un 13% de mejora en la variante del algoritmo introducida en comparación con el algoritmo original. La optimización intra-nodo e inter-nodo forman parte de la optimización de desempeño en velocidad y consumo de energía del procesador. Estas dos propiedades mantienen una relación compleja y por lo tanto es una variable importante que considerar. La realización del algoritmo para la Optimización del problema bi-objetivo para el desempeño y la energía (BOPE) dada por (Manumachu and Lastovetsky, 2018) es una solución para aplicaciones de datos en paralelo en clústeres homogéneos de procesadores multinúcleo. Este algoritmo, que es llamado ALEPH, tiene como entradas funciones discretas de desempeño y consumo de energía dinámico y da como salida el conjunto global pareto-óptimo de las soluciones. Se reporta l desempeño de la optimización en una reducción de la energía de consumo dinámica. Con el uso de la librería OpenBlast, el promedio y el máximo porcentaje de reducción de la energía fue de 12 y 68 porciento. Una propuesta de métodos y algoritmos para la minimización de tiempo de ejecución y energía para los modernos clústeres multinúcleo son propuestos en (Lastovetsky and Reddy Manumachu, 2017), que formulan el problema de desempeño y la optimización de energía con algoritmos de complexidad $O(p^2)$ donde p es el número de procesadores. Uno de los algoritmos presentados es EOPTA, que es un algoritmo similar al POPTA (Pasado antes de llegar), fue analizado por medio de dos aplicaciones de datos paralelos, OpenBLAS y FFTW en un servidor Intel Haswell. Los porcentajes de mejoramiento promedio estuvieron en el 1%, 17% y 97%. En la minería de datos, el algoritmo Patrón secuencial frecuente cercano es una parte importante en la secuencia de minería. Una paralelización eficiente de este algoritmo se presenta en (Huynh, Vo and Snasel, 2017), con una significante reducción en el tiempo de ejecución. En (Ichnowski and Alterovitz, 2014) se presentaron dos algoritmos, El explorador paralelo rápido de árbol (PRRT por sus siglas en inglés) y el RRT* paralelo. Estos dos algoritmos son utilizados en los planeadores de movimiento basados en muestreo y las adaptaciones en paralelo son factibles y óptimas para los procesadores multinúcleo. La implementación genero una tasa mayor y mejor calidad en la solución con el incremento de núcleos utilizados. La identificación de parámetros físicos es crucial para los diseños de control, monitoreo y diagnósticos de fallas en la industria, en (Immune, Learning and Optimization, 2016) se propone un algoritmo de optimización inteligente, bio-inspirado basado en arquitectura paralela. El algoritmo esta optimizado para ejecutarse en unidades de procesamiento central multinúcleo. El tiempo promedio de ejecución del algoritmo en un CPU de un solo núcleo es de 125, mientras que, en dos, tres y cuatro núcleos se lograron tiempos de 87, 78 y 61 segundos respectivamente. La técnica de Representación jerárquica de comprensión de volumen es requerida en diferentes aplicaciones que requieren acceso aleatorio de datos en modelos masivos. Un modelo novedoso es presentado en (Kim *et al.*, 2010), el cual cuenta con soporte paralelo que se puede utilizar en procesadores multinúcleo. La implementación en una CPU de cuatro núcleos alcanzo una aceleración de 2x con respecto a la versión de un solo núcleo. Para corregir errores de transmisión en sistemas de comunicaciones digitales se utilizan los códigos de baja densidad revisores de paridad (LDPC). Una implementación en multinúcleo se realiza en (Le Gal and Jego, 2016), donde la comparación de los resultados al implementar el algoritmo en un simple núcleo de un procesador INTEL i7 alcanzó 170 Mbps, lo que queda rebasado por la implementación en cuatro núcleos del mismo procesador con un resultado de 560 Mbps. La cuantificación del uso del CPU en las diferentes tareas que se realizan es importante cuando se parametriza una ejecución. Si el procesador que se utiliza es de un solo núcleo, la tarea es sencilla, pero cuando se utilizan procesadores multinúcleo, la cuantificación tiene varios factores a considerar, como lo es la cantidad de recursos asignados a la tarea. Un nuevo

mecanismo de cuantificación es presentado en (Luque *et al.*, 2012), en la que se mejora la precisión cuando se mide la utilización de los núcleos en el CPU. Los resultados indican que los mecanismos establecidos llegan a tener 16% de error en promedio, cuando la propuesta llega a tenar un error menor del 2.8% en un sistema de procesador con ocho núcleos. El comprobador de notificación de eventos contra las subscriciones recibidas es parte fundamental de toda infraestructura de publicación y suscripción. Los algoritmos de comprobación propuestos difieren entre sien gran parte, excepto en que están diseñados para ejecutarse en hardware convencional. En (Margara and Cugola, 2014) se propone un algoritmo denominado comparación paralela de contenido, algoritmo diseñado para hardware paralelo, el que se implementó en CPU multinúcleo y en GPGPU. Los resultados de los algoritmos fueron: SFF tarda 1.353 ms y BETree 0.326 ms, mientras que los algoritmos propuestos solo tardan 0.0205 para la implementación en CPU multinúcleo y 0.0091 ms en GPGPU, dando una aceleración de 66x y 148.7x respectivamente. Una opción novedosa para a programación de los núcleos de los CPU es utilizar Java 8, cuya nueva funcionalidad esta presentada en (Masegosa, Martinez and Borchani, 2016). Está versión de java (JDK 8.0) pretende proveer de una API para realizar la codificación de manera simple y en perfecto paralelismo para el público en general. Este artículo menciona varias implementaciones en las que se utilizan las diversas características e Java 8.0 en la programación de CPU multinúcleo, abriendo el camino para la gran parte de los programadores noveles.

El procesamiento en procesadores especializados (DSP)

El procesamiento digital de señales ha sido una tarea que se distingue por su alto requerimiento de procesamiento de datos, lo que lo hace un área de interés en el diseño de hardware especializado. Para escribir algoritmos de decodificación de audio AVS-P3 en el chip del procesador de señal digital (DSP) que admite algoritmos de punto fijo, los algoritmos de punto flotante de decodificación de audio AVS-P3 deben convertirse a algoritmos de punto fijo. Debido a la alta complejidad de las ventanas del módulo IMDCT (Transformada Coseno Discreta Modificada Inversa) en el algoritmo del decodificador AVS-P3, en (Li *et al.*, 2013) proponen un algoritmo de ventana mejorado en este documento. Los resultados de la prueba muestran que la complejidad de tiempo del módulo optimizado se reduce en un 17,04% y la complejidad del espacio del módulo optimizado se reduce en un 32,9%. En (Li and Huang, 2017) se implementa un método de comunicación de datos de alta velocidad entre DSP y FPGA para cumplir con los requisitos en tiempo real de la transmisión de datos y el procesamiento de algoritmos en un sistema de radio definido por software. El sistema usa AD9361 para adquisición de datos, FPGA de alto rendimiento para procesamiento previo de datos, procesador DSP de baja potencia para lograr algoritmos complejos y parámetros de extracción, procesamiento combinado de datos de alta velocidad de DSP con respuesta FPGA en tiempo real para cumplir con el rendimiento óptimo del sistema. Los resultados experimentales demuestran que el esquema tiene alta viabilidad y estabilidad. Hoy en día, los accionamientos con motor síncrono de imanes permanentes (PMSM) se usan ampliamente en muchas aplicaciones industriales. Dado que la mayoría de los sistemas de accionamiento PMSM con técnicas de control vectorial en lazo cerrado se controlan con controladores proporcionales más integrales (PI), existen demandas crecientes para obtener parámetros óptimos de ganancia PI para lograr un alto rendimiento. Para eliminar las desventajas de las técnicas tradicionales de optimización PI, en (Qiwei Cao and Liuchen Chang, 2016) se propone una novedosa metodología de optimización de controlador PI basada en el algoritmo genético multi-objetivo NSGA-II (algoritmo genético de clasificación no dominado II) para mejorar el rendimiento del sistema de accionamiento PMSM en varios las condiciones de trabajo. El algoritmo NSGA-II se adapta con éxito para encontrar ganancias

óptimas de PI para el sistema de accionamiento PMSM. El proceso de diseño del controlador PI se simplifica y la intervención del diseñador se reduce en gran medida. Tanto la simulación como los resultados experimentales han confirmado la efectividad de la metodología de optimización genética propuesta. Además, con solo cambiar los modelos de motor, el programa de optimización del controlador PI se puede aplicar fácilmente a cualquier otra unidad de motor para lograr el rendimiento deseado. En (Caffarena *et al.*, 2010) se presenta un estimador de ruido de cuantificación de punto fijo que apunta a algoritmos de procesamiento de señal digital (DSP). El estimador permite una reducción significativa en el tiempo de cálculo requerido para realizar optimizaciones de longitud de palabras complejas a la vez que proporciona una alta precisión. El algoritmo *Affine Arithmetic* (AA) se utiliza para proporcionar una estimación de relación de señal a ruido de cuantificación (SQNR) para algoritmos diferenciables no lineales con y sin retroalimentación. El estimador se prueba usando un subconjunto de algoritmos no lineales, tales como operaciones vectoriales, filtro IIR de computación de potencia, filtros adaptativos y ecualizadores de canal. El tiempo de cálculo de la optimización de longitud de palabras se incrementa en tres órdenes de magnitud, mientras que el error de estimación promedio se reduce al 6% para la mayoría de los casos. En (Ng, Tay and Mok, 2009) se presentan los algoritmos de verificación del iris que utilizan la característica de textura y su implementación y optimización en el procesador de señal digital (DSP). El método evalúa las imágenes del iris tomadas de la base de datos de imágenes del iris CASIA versión 1.0. La optimización se realiza mediante el ajuste del código fuente, la optimización de bucles y la optimización del código condicional. Los resultados experimentales demuestran que el sistema de verificación de iris es capaz de completar la verificación en menos de un segundo. El sistema de verificación de iris basado en DSP proporciona una solución de autenticación rápida y baja potencia dentro de un dispositivo compacto. (Ling-bin *et al.*, 2010) introdujo el principio y el proceso del algoritmo de detección de rostros humanos, así como su implementación y optimización utilizando DSP. Aplicó el algoritmo AdaBoost para detectar el rostro humano. Antes de la detección, se usaron algunos métodos para optimizar los datos de imagen originales, como la escala de la imagen, el filtro de la mediana, la ecualización del histograma y la detección de bordes. Después de la detección, juzgó y modificó los resultados para mejorar aún más la precisión de la detección. Luego usó muchos métodos de optimización para cumplir con el requisito en tiempo real. Los resultados experimentales muestran que este sistema DSP puede cumplir los requisitos en tiempo real incluso si detecta caras en imágenes de alta resolución de 720x576, al mismo tiempo tiene una alta precisión y una baja tasa de errores en un entorno complejo. Una tarea frecuente en el procesamiento digital de señales es ajustar la frecuencia de muestreo de acuerdo con la señal de interés. Los sistemas con diferentes tasas de muestreo se conocen como sistemas multi-tasa. En (Gawande, Metkar and Khanchandani, 2016) se presentan diferentes estructuras de filtros multi-tasa utilizando varias técnicas de optimización que resultan en alta velocidad, alto rendimiento y alta tasa de computación. La mejor estructura de filtro de diezmado entre las estructuras comparadas es *Transpps Pipeline*, que proporciona una mejora en la velocidad de casi 100 MHz para las especificaciones de filtro seleccionadas. Se concluye en el artículo que la velocidad, el rendimiento y la velocidad de cálculo se alcanzan de manera óptima mediante la estructura del filtro de decimación de *Transpps Pipeline*. En (Luo and Chen, 2014) se presenta un método para mejorar el algoritmo clave SIFT en el sistema DSP multinúcleo. El sistema de hardware tiene 8 núcleos y abundantes recursos, y se adapta al cálculo de la cantidad de algoritmo SIFT de big data. El algoritmo SIFT y los datos se dividen en muchos bloques de ejecución al mismo tiempo en multinúcleo y de esta manera son las características de optimización más importantes en este documento. Los resultados experimentales muestran que la velocidad del algoritmo de la tecla SIFT mejoró. En un escenario reciente, aumenta la complejidad del diseño, lo que motiva al diseñador del sistema a innovar continuamente. Al diseñar un sistema

complejo con acoplamiento ajustado, el diseño de varios bloques se convierte en una opción. La arquitectura reconfigurable de grano grueso (CGRA), de clase fuertemente emergente, actualmente está recibiendo la atención debida con un excelente rendimiento y flexibilidad en la fabricación. En (Khorgade and Dakhole, 2016) se presenta el diseño de un tejido reconfigurable con varias combinaciones de elementos de procesamiento. Es probable que la comprensión de las arquitecturas SoC evolucione con el tiempo. El diseño se aplicará a otras aplicaciones, como las comunicaciones, la decodificación de imágenes criptográficas y video, que también brindan reconfiguración y escalabilidad. En (Wang, Xie and Pan, 2013) Muestran que un espectro de coseno raíz elevado con un factor de caída de 0.1 es óptimo para señales Nyquist 16QAM considerando la compensación entre el rendimiento del sistema y las complejidades, y un filtro FIR digital con 37 derivaciones es suficiente para generar tales señales. Con un filtro eléctrico super gaussiano de cuarto orden espectral, los DAC con 1,2 muestras / velocidad de muestreo de símbolos pueden generar señales 16QAM Nyquist-WDM con un espaciado de canal de 1,05 símbolos, con una penalización OSNR por debajo de 0,2 dB en comparación con 2 muestras/símbolo. En (Li, Zhu and Tian, 2010) se logró y mejoró un algoritmo de reconocimiento rápido de malezas en el DSP de DM6437 con núcleo C64 + en este documento. El algoritmo es capaz de identificar malezas entre las filas de cultivo con información de las ubicaciones y áreas.

Las GPGPU como soporte de programación en hardware paralelo

La Eliminación de Contenedores es un marco que abarca varios algoritmos, incluyendo el algoritmo de la Propagación Confiada. En (Bistaffa, Bombieri and Farinelli, 2017) se presenta una aproximación para acelerar la Eliminación de contenedores en GPUs. El reporte indica que la aceleración conseguida utilizando GPU vs CPU fue de 39x, una aceleración que es 466% mayor que la contraparte. En (G. Chen *et al.*, 2015) exponen un nuevo marco de software para automatizar la localización de datos en un GPU.el software denominado Porple proporciona la capacidad de especificar el tipo de memoria requerido en los datos para proporcionar su ubicación, un compilador tipo C que permite programar las porciones de código relevantes y un mecanismo en línea para ubicar en tiempo de ejecución s datos en la memoria adecuada a sus características. Implementado en tres diferentes tarjetas de Nvidia, Tesla K20, Tesla M2075 y Tesla C1060, el desempeño del marco de software fue incluso en ocasiones mayor al dado por el mejor posicionamiento obtenido por medición exhaustiva. La implementación del algoritmo de la aproximación geométrica discreta en el dominio del tiempo es una solución numérica para resolver las ecuaciones de Maxwell en redes no estructuradas. La implementación del algoritmo en GPGPU es presentado en (Cicuttin *et al.*, 2018) debido a su alta implementación en paralelo. La aceleración obtenida llego 16.58x comparado con la implementación en CPU. El análisis de la distribución de potencia en redes paralelas en integrado es realizado en una plataforma de varios GPU con varios núcleos en (Feng, Zeng and Li, 2011). La red es fragmentada y desarrollada por diferentes GPUs que, al trabajar una pequeña porción de la red, el cálculo en pasos fluidos. El análisis se realizó con diferentes cantidades de nodos en las redes, entre las que tenemos dos, una de cuatro mil y otra de ocho mil nodos. El resultado de la aceleración fue de 140x utilizando cuatro GPUs en comparación con un procesador de ocho núcleos. En el área de la navegación satelital, la implementación del rastreo de los rayos emitidos por el Sistema Global de Navegación por Satélite es calculado para determinar los caminos de las señales a través de la atmosfera. El cálculo de los rayos es una tarea que requiere una gran cantidad de cálculos, por lo que en (Gegout *et al.*, 2014) se propone la implementación en GPS utilizando CUDA. La aceleración reportada en comparación con la implementación en CPU es de 75x, cuando se aplican 130 000 rayos. La transposición de

matrices es una tarea común en los datos de las señales a ser procesados digitalmente, por lo que su implementación en un dispositivo que disminuya su tiempo de ejecución es conveniente. En (Gomez-Luna *et al.*, 2016) muestran los resultados de implementar la transposición de matrices en línea, utilizando una GPU. Los resultados dependen del tamaño de la matriz, para matrices pequeñas la aceleración es buena, pero para matrices grandes, la aceleración es de incluso 20x comparado con la implementación en un procesador AM Hawaii. Regularmente, la optimización de los algoritmos es buena para matrices grandes de datos, pero cuando las matrices son pequeñas, los algoritmos optimizados tienen una respuesta pobre. En (Haidar *et al.*, 2017) se ofrece una guía en la que se proporciona una serie de recomendaciones de cómo organizar los núcleos para el cálculo de matrices pequeñas, como por ejemplo, si se trabaja con matrices de 16x16, la recomendación es un entrelazado donde la esquina final de la primer matriz se encuentra con la primer esquina de la segunda matriz, esto dará columnas de 32 elementos, lo que es una alineación de 128-byte, lo que es un almacenamiento eficiente para la implementación en GPU. Las propuestas de implementación para mapear eficazmente los rayos físicos ópticos de disparo y rebote (SBR-PO) son varias, pretendiendo acelerar el proceso de solución de problemas de dispersión. En (Kee and Wang, 2013) se discute la implementación del SBR-PO en la arquitectura de computo de dispositivos unificada y OptiX de Nvidia para reducir el tiempo de cálculo para el modelado de dispersión mono estático a partir de objetos eléctricamente grandes y de forma arbitraria. La comparación contra los resultados de la literatura y los resultados de propuestas comerciales se obtiene un resultado sobresaliente al llegar a una utilización del 99% del GPU. La búsqueda de datos en arboles es un algoritmo frecuente en los juegos de computadora, por lo que su aceleración es muy conveniente comercialmente. En (L. Li *et al.*, 2015) se realizó la implementación del algoritmo de búsqueda en dos juegos, El ajedrez y Conecta6. La aceleración conseguida en el ajedrez fue de 89.95x, mientras que en el Conecta6 alcanzo 11.43x. El mapeo isométrico para la clasificación espectral es utilizado ampliamente en el análisis hiperespectral de imágenes para visualización y reducción de dimensiones. La implementación paralela del algoritmo para GPU es propuesta en (Li *et al.*, 2017) y los resultados de la clasificación son utilizados para verificar los resultados de los vectores embebidos. Los resultados sugieren que la rápida solución propuesta para el problema de eigen-descomposición corrió hasta 317 veces más rápido que en el método convencional en CPU. La presentación de una nueva versión del algoritmo de disparo-Newton implementado en paralelo con aceleración basado en GPU se da en (Liu, Yu and Tan, 2015). La implementación es desarrollada en la tarjeta Nvidia Kepler K40 que cuanta con 2880 núcleos. Los resultados arrojan que la aceleración conseguida fue de 8x comparado con la versión secuencial en CPU. La transferencia de la radiación electromagnética a través de una atmosfera planetaria se calcula utilizando un modelo de transferencia radioactiva atmosférica (RTM). Las aplicaciones del código de transferencia radiante de banda ancha para aplicaciones de modelo de circulación general se basan en el código de referencia de una sola columna (RRTM). El modelo de banda corta es implementado en paralelo en un GPU y es reportado en (Mielikainen *et al.*, 2016). El algoritmo es alojado en una tarjeta Nvidia Tesla K40 y se reporta una aceleración de 202x comparado con la contra parte de un solo hilo Fortran en un Intel Xenon E5-2603. El diseño y análisis de arquitecturas de hardware escalables para el entrenamiento de los parámetros de aprendizaje se utilizan para clasificar grandes conjuntos de datos. El diseño de una arquitectura de hardware escalable del algoritmo K-means es realizado en FPGA y en GPU (Mohammadi *et al.*, 2018). Estas arquitecturas en paralelo con tuberías escalonadas son capaces de implementar conjuntos de datos sin restricciones en sus dimensiones. Las implementaciones de hardware son típicamente rápidas comparadas con cualquier implementación en software. En particular, la implementación en FPGA, un Xiling Virtex-7, utilizando coma flotante de 28 bit obtuvo una aceleración de un 15.94x, mientras que la implementación en la GPU fue

entre 1.33x y 14.6x comparado con CPU. El problema de las *N* reinas es un pasatiempo en el que se debe de colocar las *N* reinas sin amenazarse. El algoritmo para resolver este problema es el Sistema de simulación de membrana. Una mejora del algoritmo del sistema de simulación de membrana en GPU es presentado en (Muniyandi and Maroosi, 2015). La aceleración con memoria global con $N = 10$ fue de 10.6 veces y utilizando memoria compartida y memoria entejada fue de 33 veces. La simulación de problemas electromagnéticos y térmicos acoplados con alta resolución requiere esquemas numéricos eficientes, el uso de lenguajes de alto rendimiento son una ayuda en la ejecución paralela de tales algoritmos de simulación, como en (Richter, Schops and Clemens, 2014) utilizan una GPU para acelerar los cálculos. El tiempo de computación para la solución por fase tiene una dependencia directa con la cantidad de núcleos implementados. Cuando se ejecuta el algoritmo en una GPU, el factor de aceleración es de 5.2 comparado con el mismo código ejecutado en un CPU. La rotoscopia es una técnica de post-procesamiento muy antigua, compleja y que consume tiempo, utilizada por un animador para producir manualmente máscaras de segmentación para una secuencia de video. Específicamente, es el acto de trazar o delinear objetos que aparecen en los marcos. Una manera ventajosa de implementar la rotoscopia se presenta en (Rzeszutek *et al.*, 2010). Para resolver el sistema linear se utilizó el método del gradiente conjugado (CGM). Debido a que el CGM se compone de operaciones vectoriales fácilmente paralelizadas, aún puede beneficiarse de una implementación de GPU. Lo que una implementación paralela hará es acelerar las operaciones individuales y no la propia CGM. Un procesador serie y un procesador multinúcleo aún tomarán el mismo número de iteraciones para completarse. Sin embargo, la versión paralela completará esas iteraciones más rápido ya que las operaciones individuales se están ejecutando mucho más rápidamente en la GPU. Uno de los principales problemas del análisis de datos hiperespectrales es la presencia de píxeles mixtos debido a la baja resolución espacial de dichas imágenes. El filtrado espectral lineal tiene como objetivo la obtención de firmas espectrales puras y sus fracciones en cada píxel de la escena. Los enormes volúmenes de datos adquiridos por los sensores hiperespectrales imponen requisitos estrictos sobre los métodos de procesamiento y desmezcla. En (Sevilla, Martin and Nascimento, 2016) se muestra una implementación eficiente del método llamado identificación simplex con separación Lagrangian aumentado, el cual es altamente paralelo, lo que explotara las características de la GPU. Los resultados indican para los conjuntos de datos hiperespectrales reales y simulados una aceleración de hasta 49 veces, lo que demuestra que la implementación de la GPU acelera significativamente la ejecución del método sobre grandes conjuntos de datos. En (Su *et al.*, 2016) se muestra la implementación eficiente de GPU del análisis de punteros basado en la inclusión de todo el programa de Andersen, un análisis fundamental en el que se basan muchos otros, incluyendo la optimización de compiladores, detección de errores y análisis de seguridad. El algoritmo de Andersen realiza modificaciones extensas al gráfico que representa las instrucciones de manipulación del puntero en un programa. Para paralelizar el análisis de Andersen de manera eficiente en GPU, se realizó un esquema de partición de carga de trabajo consciente de desequilibrio que divide su carga de trabajo dinámicamente entre las deformaciones simultáneas, inicialmente de forma centrada en *warp*. Para un conjunto de puntos de referencia de 14 “C” evaluados, nuestra implementación paralela del análisis de Andersen logra una aceleración significativa del 46 por ciento en promedio sobre el estado de la técnica en una GPU NVIDIA Tesla K20c. La identificación del subespacio de señal es una operación crucial en las imágenes hiperespectrales, que permite una reducción de dimensión correcta que a menudo produce ganancias en el rendimiento y la eficiencia del algoritmo. En (Torti *et al.*, 2014) se presentan nuevas implementaciones paralelas de una identificación de subespacio hiperespectral ampliamente utilizada con un algoritmo de error mínimo (HySime) en diferentes tipos de arquitecturas de computación de alto rendimiento, incluyendo CPU multinúcleo de propósito general, unidades de procesamiento de gráficos (GPU) y procesadores de señal digital (

DSP). Las implementaciones se probaron en arquitecturas múltiples, principalmente GPU basadas en la arquitectura NVIDIA Fermi, CPU multinúcleo Intel i7 y un DSP TI C6678. Los resultados muestran que las versiones de CPU GPU y multinúcleo son más rápidas que el código original de MATLAB. Estos resultados también sugieren que el empleo de estas tecnologías en una plataforma aérea sería más efectivo, en términos de costos y rendimiento, que las actuales placas de cuatro núcleos, especialmente con el uso de futuras generaciones de GPU que reducirán aún más las tasas de consumo de energía. La categorización visual es importante para administrar grandes colecciones de imágenes digitales y video, donde los metadatos textuales a menudo están incompletos o simplemente no están disponibles. El modelo de bolsa de palabras se ha convertido en el método más poderoso para la categorización visual de imágenes y video. La tendencia a aumentar la potencia de cómputo en arquitecturas de CPU y GPU más nuevas es aumentar su nivel de paralelismo, explotar este paralelismo se convierte en una dirección importante para manejar el costo computacional del enfoque de bolsa de palabras, como se realizó en (Van De Sande, Gevers and Snoek, 2011). Al optimizar un sistema basado en el enfoque de bolsa de palabras, el objetivo es minimizar el tiempo que lleva procesar lotes de imágenes. Los autores utilizaron una implementación paralela en el GeforceGTX260GPU, la clasificación de imágenes invisibles fue 4.8 veces más rápida que una versión de CPU de cuatro núcleos en el Core i7 920, a la vez que proporcionó exactamente los mismos resultados numéricos. Incluso se consideró cómo los algoritmos se pueden generalizar a otras aplicaciones, como la recuperación de texto y la recuperación de videos. Cuando la aceleración obtenida se utilizó para procesar cuadros de video adicionales en un punto de referencia de recuperación de video, la exactitud de la categorización visual mejoró en un 29%. La amplia aplicación de la investigación óhmica ha producido un estallido de datos biológicos en los últimos años, que a su vez ha aumentado la necesidad de inferir redes biológicas a partir de los datos. El aprendizaje de redes biológicas a partir de datos experimentales puede ayudar a detectar y analizar vías de señalización aberrantes, que pueden utilizarse en el diagnóstico de enfermedades en una etapa temprana. Sin embargo, debido a su naturaleza combinatoria, el aprendizaje computacional de las relaciones dependientes subyacentes a redes complejas es NP-completo. Para reducir la complejidad, en (Wang *et al.*, 2016) se propone utilizar los métodos de la cadena de Markov Monte Carlo (MCMC) para muestrear el espacio de la solución. Los métodos de MCMC garantizan la convergencia y la capacidad de transferencia. Sin embargo, MCMC no es escalable para redes con más de 40 nodos debido a la complejidad computacional. Los resultados reportan una aceleración de 2.46x al optimizar el algoritmo y una aceleración adicional de 58 veces al implementarlo en una GPU. En total, la aceleración del algoritmo resulto en 143x. En (Wu, Song and Jeon, 2014) se propone una implementación paralela masiva eficiente basada en GPU del método de des-entrelazado en el campo adaptativo dirigido al borde que interpola los píxeles faltantes basados en la covarianza des-entrelazada estimada a partir de la covarianza entrelazada de acuerdo con la dualidad geométrica entre la covarianza entrelazada y des- entrelazada. El método propuesto interpola más de un píxel faltante a la vez con el fin de obtener una aceleración significativa en comparación con el caso de la interpolación de solo un píxel faltante a la vez. Los resultados experimentales muestran que se obtuvo una aceleración de 94.6x cuando se consideró el tiempo de transferencia de E / S, en comparación con el código original de CPU C de un hilo. En el pasado, los simuladores de física MRI se han utilizado para optimizar los protocolos de imágenes, el seguimiento de fuentes de artefactos, pero también para fines de capacitación. En la mayoría de los casos, las simulaciones se utilizaron para responder a un problema metodológico particular y para proporcionar información con respecto a la futura secuencia de pulso o el desarrollo del protocolo de imágenes. En (Xanthis *et al.*, 2014) el objetivo fue desarrollar un simulador de imágenes por resonancia magnética (MRI) que no haga suposiciones con respecto a la secuencia de pulso subyacente y también permite el análisis complejo a gran escala en una sola

computadora sin requerir simplificaciones del modelo de MRI. Se supone que tal la plataforma de simulación podría desarrollarse con aceleración paralela del núcleo ejecutable dentro del entorno de la unidad de procesamiento gráfico (GPU). La plataforma de simulación se desarrolló en MATLAB, mientras que los servicios centrales de demanda computacional se desarrollaron en CUDA-C. El alto poder computacional de las simulaciones basadas en GPU se comparó con otras configuraciones de computadora. Se logró una aceleración de aproximadamente 228 veces en comparación con el código C ejecutado en serie en la CPU, mientras que se logró una aceleración entre 31 y 115 veces en comparación con el código C ejecutado en paralelo OpenMP en la CPU, dependiendo del número de subprocesos utilizados en multi-hilos. En (Pikacz and Gambrych, 2014) presenta dos implementaciones de la transformada rápida de Fourier descompuesta en operaciones vectoriales. Este enfoque es apropiado para los casos en que los datos que se van a transformar se almacenan en un orden poco ortodoxo y, como tales, son muy adecuados para el procesamiento de datos de radares y sonares. Los procedimientos descritos que realizan un vector FFT se implementaron para las plataformas TigerSHARC DSP y NVIDIA CUDA. Ambas implementaciones presentadas demostraron ser más eficientes que los procedimientos de biblioteca altamente optimizados. Dependiendo de la plataforma y el tamaño de los datos, la aceleración alcanzó del 10% al 200%. Por ejemplo, para el radar de seguimiento que emite 128 impulsos coherentemente y procesa 512 celdas de rango, la aceleración fue de aproximadamente 40% para la plataforma CUDA y de aproximadamente 100% para el procesador TigerSHARC. La unidad de procesamiento de gráficos (GPU) se ha aplicado con éxito en muchos ámbitos de la informática científica debido a su rendimiento superior en el cálculo de la flotación y el ancho de banda de la memoria, y tiene un gran potencial en aplicaciones de sistemas de potencia. El análisis de seguridad estático N-1 (SSA) parece ser una aplicación válida en la que es necesario resolver los problemas masivos de flujo de potencia de corriente alterna (ACPF). Sin embargo, al aplicar algoritmos acelerados GPU existentes para resolver el problema N-1 SSA, el grado de paralelismo es limitado porque las investigaciones existentes se han dedicado a acelerar la solución de un único ACPF. En (Zhou *et al.*, 2017) se propone una solución acelerada por GPU que crea una capa adicional de paralelismo entre los ACP por lotes y consecuentemente alcanza un nivel mucho más alto de paralelismo general. Se establecen dos principios básicos para determinar algoritmos de GPU bien diseñados, a través de los cuales se demuestra la limitación de la solución secuencial ACPF acelerada por GPU. Para mejorar aún más la eficacia de la solución de SSA, se desarrolla y optimiza cuidadosamente un cribado acelerado por lote de GPU-matriz Jacobiana y un cribado de contingencia. En comparación con la contraparte UMFPACK basada en una sola CPU que utiliza Intel Xeon E5-2620, la estructura SSA acelerada por GPU propuesta que utiliza NVIDIA K20C logra hasta 57.6 veces la aceleración. Incluso puede lograr una aceleración cuatro veces mayor en comparación con una de las soluciones de computación paralela de CPU multinúcleo más rápidas que utilizan la biblioteca KLU.

Plataformas heterogéneas de dispositivos paralelos

El cálculo de plantilla se usa ampliamente en cálculos científicos y se han propuesto muchos aceleradores basados en CPU multinúcleo y GPU. La computación por plantilla tiene una pequeña intensidad operativa, por lo que normalmente se requiere un gran ancho de banda de memoria externa para un alto rendimiento. Los FPGA tienen el potencial de resolver este problema utilizando una gran memoria interna de manera eficiente. Sin embargo, se requiere un tiempo de depuración, diseño y prueba muy grande para implementar con éxito una arquitectura FPGA. Para resolver este problema, en (Waidyasooriya *et al.*, 2017) proponen una

plataforma FPGA que utiliza un lenguaje de programación tipo C llamado lenguaje informático abierto (OpenCL). Los resultados experimentales indican un logro de 119 a 237 Gflop/s de potencia de procesamiento y mayor velocidad de procesamiento en comparación con las implementaciones convencionales de GPU y CPU multinúcleo. En el diseño conjunto de hardware / software (HW / SW), el particionamiento de hardware / software es un paso esencial ya que determina qué componentes se implementarán en el hardware y cuáles en el software. La mayoría de los problemas de particiones HW / SW son NP duros. Para problemas de gran tamaño, se deben utilizar métodos heurísticos. En (Hou *et al.*, 2017) presentan un algoritmo genético paralelo con corrección de dispersión para partición HW / SW en CPU-GPU. Primeramente, se presenta un algoritmo genético mejorado con corrección de dispersión. Los individuos con restricción limitada son llevados a una región factible paso a paso. De esta forma, la intensificación se puede mejorar y también se puede manejar el problema de restricción. La estrategia propuesta calcula los costos de cada individuo en paralelo en la GPU y corrige a los individuos con restricciones menores en paralelo en la CPU multinúcleo. De esta forma, se puede lograr una computación en paralelo altamente eficiente en la que se asignan docenas de pasos de cálculo de corrección irregular a la CPU multinúcleo y se mapean miles de pasos de cálculo de costo regular a la GPU de muchos núcleos. En (Dine *et al.*, 2016) se trata de la cuestión de la complejidad computacional de la localización y el mapeo simultáneos basados en gráficos (SLAM). SLAM permite que un robot que esté navegando en un entorno desconocido construya un mapa de este entorno al mismo tiempo que determina la posición del robot en este mapa. La gráfica basada en SLAM es un método de suavizado que usa un gráfico para representar y resolver el problema de SLAM. Se propone un mapeo de arquitectura de algoritmo para implementar eficientemente SLAM basado en gráficos en una arquitectura heterogénea incorporada de bajo costo. Este mapeo aprovecha la construcción de gráficos incrementales. Hemos demostrado que aprovechar la GPU integrada puede acelerar el SLAM basado en gráficos. La construcción del sistema en la GPU es hasta cinco veces más rápida en comparación con la CPU. Sin embargo, para mejorar más nuestro diseño, planeamos investigar una implementación del bloque de solución del sistema que podría explotar la estructura de bloque dispersa del problema. Las imágenes hiperespectrales se usan en diferentes aplicaciones en la ciencia de la Tierra y del espacio, y muchas de estas aplicaciones presentan limitaciones de tiempo reales o casi reales. Un problema al analizar imágenes hiperespectrales es que su resolución espacial generalmente no es suficiente para separar diferentes constituyentes espectralmente puros (miembros finales); como resultado, varios de ellos se pueden encontrar en el mismo píxel. En (Torti *et al.*, 2016) se describe una cadena de filtrado hiperespectral en tiempo real basada en tres pasos principales: 1) estimación del número de miembros finales que utilizan la identificación del subespacio hiperespectral con error mínimo; 2) estimación de las firmas espectrales de los miembros finales utilizando el análisis del componente del vértice (VCA); y 3) estimación de abundancia no restringida. Desarrollamos nuevas implementaciones paralelas de los algoritmos antes mencionados y los ensamblamos en una cadena de mezcla en tiempo real totalmente operativa utilizando unidades de procesamiento de gráficos (GPU), explotando la arquitectura de dispositivo unificado de NVIDIA (CUDA) y su biblioteca de subrutinas de álgebra lineal básica (CuBLAS). Como resultado, la cadena en tiempo real explota los paradigmas CPU (multinúcleo) y GPU en la optimización. Los resultados revelan que esta implementación híbrida de la CPU GPU paralela cumple totalmente con las limitaciones de tiempo real en las aplicaciones de imágenes hiperespectrales. Los CPU multincleo pueden combinarse con GPUs para realizar computaciones sobre mallas 3d no estructuradas en clústeres de GPU/CPU heterogéneas. En (Yang *et al.*, 2015) explican cómo desbloquear la potencia informativa del CPUs sin retrasar otras tareas relacionadas con el movimiento de datos. Al solucionar la ecuación de difusión representativa mediante el método de volúmenes finitos centrados en células, los autores demuestran que combinando

la capacidad de computación de CPUs y GPUs ofrece una ventaja de rendimiento sobre el enfoque único de la GPU. Los resultados de la implementación heterogénea, el rendimiento obtenido con el uso exclusivo de GPU, y con una instancia del código heterogéneo donde todas las comunicaciones están deshabilitadas indican una mejora considerable en tiempo de ejecución. La aceleración para 128 nodos es 98.7x. Actualmente, la rápida evolución de las cámaras en los últimos años las ha convertido en herramientas prometedoras para el diagnóstico de Tokamak. La solución presentada en (Zhang *et al.*, 2018) consiste en un prototipo de sistema de adquisición y procesamiento de imágenes visibles de alta velocidad (HVIAP) dedicado para el control de posición y forma de tokamak superconductores avanzados experimentales. La unidad de procesamiento de gráficos (GPU) y la matriz de puertas programables en campo (FPGA) se usan típicamente como aceleradores o coprocesadores además de una CPU. Un sistema informático tan heterogéneo puede combinar las ventajas de sus componentes individuales. Los resultados del proceso de imagen fuera de línea se comparan con el código de ajuste de equilibrio, que es un método de reconstrucción comúnmente utilizado, con un error promedio de 1,5 cm. El tiempo de procesamiento total para una trama es inferior a 0.3 ms. BLAST, abreviación de *Basic Local Alignment SearchTool*, es una herramienta difundida que se utiliza en las ciencias de la vida para la búsqueda de secuencias por parejas. Sin embargo, con el advenimiento de la secuenciación de próxima generación (NGS), ya sea al principio o al final de NGS, el crecimiento exponencial de las bases de datos de secuencias está superando nuestra capacidad de analizar los datos. Mientras que los estudios recientes han utilizado la unidad de procesamiento de gráficos (GPU) para acelerar el algoritmo BLAST para buscar secuencias de proteínas (es decir, BLASTP), estos estudios usan paralelismo de grano grueso, donde una alineación de secuencias se asigna a un solo hilo. Tal enfoque no utiliza eficientemente las capacidades de una GPU, particularmente debido a la irregularidad de BLASTP en ambas rutas de ejecución y patrones de acceso a la memoria. Para abordar las deficiencias anteriores, (Zhang, Wang and Feng, 2017) presenta un enfoque detallado para paralelizar BLASTP, donde cada fase individual de búsqueda de secuencia se asigna a muchos hilos en una GPU. Las estructuras de datos centrales aprovechan la última arquitectura NVIDIA Kepler. Los autores optimizaron las fases restantes de cuBLASTP en una CPU multinúcleo con pthreads. En un nodo de cómputo con una CPU Intel Sandy Bridge de cuatro núcleos y una GPU NVIDIA Kepler, cuBLASTP logra una aceleración de hasta 7,9 veces y 3,1 veces sobre FSA-BLAST de un solo subproceso y NCBI-BLAST de subprocesos múltiples con cuatro subprocesos. En (Possa *et al.*, 2014) se presenta una nueva arquitectura flexible parametrizable para procesamiento de imágenes y video con requisitos de latencia y memoria reducidos, que admite una resolución de entrada variable. La arquitectura propuesta está optimizada para la detección de características, más específicamente, el detector de bordes Canny y el detector de esquina Harris. La arquitectura contiene extractores de vecinos y operadores de umbral que se pueden parametrizar en tiempo de ejecución. Las simplificaciones de algoritmo se emplean para reducir la complejidad matemática, los requisitos de memoria y la latencia sin perder la fiabilidad. Un análisis de rendimiento de las implementaciones de FPGA y GPU, y una implementación adicional de referencia de CPU, muestra el rendimiento competitivo de la arquitectura propuesta incluso a una frecuencia de reloj mucho más baja que las de la GPU y la CPU. Los resultados muestran una clara ventaja de la arquitectura propuesta en términos de consumo de energía y mantienen un rendimiento confiable con imágenes ruidosas, baja latencia y requisitos de memoria. Los nodos de extremo loT requieren un alto rendimiento y una eficiencia energética extrema para hacer frente a los algoritmos de análisis de datos de sensores cercanos complejos. El procesamiento en múltiples procesadores programables que operan en un umbral cercano está surgiendo como una solución prometedora para explotar el impulso de energía que brinda la operación de bajo voltaje, mientras se recupera la degradación de frecuencia relacionada con el paralelismo. En (Das *et al.*, 2018) se presenta una arquitectura de clúster heterogénea que extiende un clúster de

procesador paralelo tradicional con un acelerador de matriz programable integrada (IPA) reconfigurable. Mientras que los procesadores programables garantizan la herencia de programación para administrar fácilmente los periféricos, las pilas de software de radio y el flujo de programa global, la descarga de núcleos intensivos en datos y control intensivo al IPA lleva a un rendimiento del sistema mucho mayor y eficiencia energética. Los resultados experimentales muestran que el clúster heterogéneo propuesto supera a una arquitectura homogénea de 8 núcleos hasta en 4.8x en rendimiento y 4.5x en eficiencia energética cuando ejecuta una combinación de núcleos intensivos en control de datos y datos típicos de aplicaciones de análisis de datos cercanos al sensor. Al combinar capacidades encontradas en unidades de procesamiento de gráficos (GPU) y unidades de procesamiento central (CPU) multinúcleo tradicionales de von Neumann, se están desarrollando y optimizando enfoques para proporcionar velocidades de procesamiento en tiempo real o casi real para aplicaciones de proyectos de investigación. Los algoritmos están diseñados para dividir el trabajo en recursos mejor diseñados para manejar la carga de procesamiento. El uso de recursos básicos permite que el diseño sea flexible durante todo el ciclo de vida sin las demoras costosas y que requieran mucho tiempo asociadas con el desarrollo del Circuito Integrado Específico de Aplicación (ASIC). Este paradigma permite una rápida transferencia de tecnología a los usuarios finales. En (Park *et al.*, 2011) se describe un algoritmo de generación de imágenes de radar de reconstrucción de impulso síncrono que ha sido diseñado para el procesamiento híbrido de CPU-GPU. Se reporta que la aceleración lograda por este enfoque excede los requisitos de rendimiento en tiempo real. El emparejamiento estéreo es un problema fundamental de la visión por computadora, y las aplicaciones emergentes, como el reconocimiento de gestos 3-D y la navegación automotriz, exigen una sincronización estéreo rápida y de alta calidad. Los enfoques basados en el campo aleatorio de Markov (MRF) son ampliamente utilizados, pero los solucionadores de software convencionales son lentos. Los solucionadores de propagación de creencias (BP), que usan patrones de mensajes locales que transmiten MRF, se han estudiado en hardware, pero su rendimiento no es confiable. Se muestra en (Choi and Rutenbar, 2016) cómo un método superior, el reenvío de mensajes secuencial de árboles secuenciados (TRW-S), se puede representar en hardware. TRW-S tiene una convergencia confiable, garantizada por su llamado cálculo secuencial. Se dividió el procedimiento de ajuste estéreo en la CPU y los FPGA y aplicaron optimizaciones de nivel de cuadro, como la reutilización de mensajes basada en la detección de cambios de escenas, la implementación en paralelo de niveles de marcos y la canalización de niveles de funciones. Los resultados experimentales muestran que el sistema alcanzó una velocidad de 22.8 cuadros/s para una desafiante tarea de emparejamiento estéreo de video QVGA. La inversión de matriz densa es un procedimiento básico en muchos algoritmos de álgebra lineal. Cualquier algoritmo de inversión de matriz densa basado en factorización implica la inversión de una o dos matrices triangulares. En (Ries, De Marco and Guerrieri, 2012) se presenta una implementación mejorada de una inversión de matriz triangular paralela para sistemas heterogéneos de CPU / GPU dual multinúcleo. Se reporta una aceleración de hasta 57x en comparación con la aplicación de referencia de CPU dual basada en LAPACK. Los sistemas de monitoreo de salud personalizables pueden ofrecer una solución rentable para la atención de la salud humana. Estos sistemas deben monitorear constantemente las señales fisiológicas de los pacientes y proporcionar un procesamiento y entrega de la gran cantidad de datos dentro de un espacio de energía y área muy Estas aplicaciones biomédicas personalizadas requieren el muestreo y procesamiento de múltiples flujos de señales fisiológicas con un número variable de canales y frecuencias de muestreo. precisas y rápidas. En base lo anterior, en (Kulkarni *et al.*, 2017) se propone un acelerador de múltiples puntos pequeño, eficiente en energía y específico de dominio, denominado nano clústeres de potencia eficiente (PENC), para mapear y ejecutar los núcleos de estas aplicaciones. Los resultados de la simulación muestran que el PENC puede

reducir el consumo de energía hasta en un 80% y 25% para los kernels DSP y ML, respectivamente, cuando se paraleliza de manera óptima. La dosis de radiación asociada con la tomografía computarizada (TC) es significativa. Los métodos de detección compresiva (CS) proporcionan enfoques matemáticos para reducir la exposición a la radiación sin sacrificar la calidad de imagen reconstruida. Sin embargo, los requisitos computacionales de estos algoritmos son mucho más altos que los enfoques de reconstrucción de imágenes convencionales, como la proyección retrospectiva filtrada (FBP). En (Chen *et al.*, 2012) se describe un nuevo algoritmo de reconstrucción tridimensional de imágenes con compresión basado en la maximización de expectativas y la variación total, denominado EM + TV, y también presenta una prometedora implementación de arquitectura híbrida para este algoritmo que incluye la combinación de CPU, GPU y FPGA. Los resultados de rendimiento indican que este enfoque proporciona un menor consumo de energía y una mejor calidad de reconstrucción, e ilustra un ejemplo de las ventajas que se pueden obtener a través de la informática específica del dominio.

Computadora de placa única Parallella

Dentro de los sistemas heterogéneos, la computadora de placa única tiene su merecido espacio por ser un sistema que incluye tres tecnologías y la ventaja del bajo consumo de energía. En el modelo de cálculo del flujo de datos, las instrucciones o tareas se activan de acuerdo con sus dependencias de datos, en lugar de seguir el orden del programa, lo que permite la explotación natural del paralelismo. El flujo de datos se ha utilizado, en diferentes niveles de sabores y abstracción (desde procesadores hasta bibliotecas de tiempo de ejecución), como una alternativa interesante para aprovechar el potencial de los sistemas informáticos modernos. Sucuri es una biblioteca de flujo de datos para Python que permite a los usuarios especificar su aplicación como un gráfico de dependencia y ejecutarla de forma transparente en clústeres de multinúcleo, al tiempo que se ocupan de los problemas de programación. Las últimas tendencias en computación de niebla e in situ asumen que los dispositivos de almacenamiento y de red estarán equipados con elementos de procesamiento que generalmente tienen un menor consumo de energía y rendimiento. Una decisión importante sobre dicho sistema es si mover los datos a los procesadores tradicionales (pagar los costos de comunicación) o realizar un cálculo donde están los datos, usando un procesador potencialmente más lento. Por lo tanto, los entornos de tiempo de ejecución que se ocupan de esa compensación son extremadamente necesarios. En (Carvalho, Ferreira and Franca, 2017) se da un primer paso hacia una solución que considera In-situ en un tiempo de ejecución de flujo de datos. Se utilizó Sucuri para administrar la ejecución en un sistema pequeño con una PC común y una placa Parallella. Dado que la placa Parallella tiene menor potencia de procesamiento que la PC, los autores evaluaron la solución propuesta realizando un conjunto de experimentos con dos aplicaciones con diferentes costos computacionales: (i) el recuento de caracteres cuenta el número de líneas que aparece en un archivo de entrada, usando búsqueda del lenguaje Python; y (ii) filtro encuentra números en líneas de un archivo de entrada, usando una expresión regular, y los suma. Ambas aplicaciones realizan búsquedas en un par de archivos del mismo tamaño, de modo que las búsquedas se pueden realizar en paralelo, aprovechando los núcleos disponibles en la PC host o en la placa Parallella. Los experimentos usaron archivos de entrada de 250 bytes, 12MiB y 48Mib para el carácter de filtro y recuento. La aplicación de caracteres de conteo también usó archivos de entrada de 476MiB. No hubo pruebas para la aplicación de filtro utilizando el archivo 476MiB, ya que las pruebas preliminares no mostraron diferencias significativas en los resultados. En (Bartok and Vasarhelyi, 2015) se analiza un método de interpolación de reglas difusas implementado en diferentes plataformas. Los sistemas de control basados en interpolación de reglas difusas

desempeñan un papel importante en el control de robots y tienen varios tipos de implementaciones. Las aplicaciones de control difuso basadas en la interpolación de reglas difusas con las contribuciones realizadas por Johanyak permitieron la implementación de varios modelos difusos. En la operación Parallella en los análisis aparece el resultado de que la operación de división de coma flotante "ni siquiera coincide con el rendimiento en coma flotante de los procesadores de PC de escritorio modernos (Core i7-4770K (Haswell), 4 núcleos a 3.5 GHz A VX2: 177 GFLOPS). En (Gener, Yildiz and Goren, 2016) se examina las ventajas de arquitecturas de múltiples núcleos de bajo costo en unidades de procesamiento gráfico (GPU) para aplicaciones masivamente paralelas. Se muestra que las unidades de procesamiento gráfico (GPU) podrían ser costosas e ineficientes en términos de energía para cargas de trabajo tales como tareas de filtrado de video de dominio espacial que aprovechan la computación paralela. Se demostró que es posible utilizar una plataforma de bajo costo para este tipo de cargas de trabajo sin demasiado esfuerzo de programación. La consistencia del tiempo de cada nodo y cada núcleo es uno de los problemas del kernel que debe abordarse para promover el desarrollo de un sistema de clúster de muchos núcleos que mantenga el sistema distribuido en marcha, y es la base de la evaluación del rendimiento. En (Gao, Huang, S. Wang, *et al.*, 2017) se estableció un nuevo modelo de arquitectura híbrida de sincronización de tiempo para mejorar la precisión de la sincronización del reloj interno de un clúster de muchos núcleos y la precisión del tiempo. Se realizó un experimento para evaluar la precisión y precisión de la arquitectura Parallella. El resultado muestro que la arquitectura híbrida logró una mayor precisión de tiempo de nivel y consistencia de reloj interno que un método de sincronización de reloj común como NTP. La arquitectura Adapteva Epiphany MIMD es una matriz 2D escalable de núcleos RISC con una rápida red en chip (NoC) para procesamiento en paralelo. En (Labowski *et al.*, 2016) se analiza la idoneidad de la arquitectura para manejar aplicaciones de radio definidas por software (SDR), como los filtros de respuesta de impulso finito (FIR). La intensidad computacional involucrada en el procesamiento de video excedió el escenario de prueba simple, en cuanto las velocidades de datos y los tamaños de búfer requeridos estuvieron dentro de lo esperado. Esta comparación solo establece que se requiere más investigación sobre tareas complejas de codificación y decodificación en Epiphany ya que el ancho de banda limitado dentro y fuera de Epifanía no ha excedido el tiempo de procesamiento asignado. En (Brauer, Lundqvist and Mallo, 2016) se utilizó el chip Adapteva Epiphany manycore para demostrar cómo se puede optimizar el rendimiento y la latencia de una cadena de procesamiento de señal de banda base, que normalmente se encuentra en LTE o WiFi, mediante una combinación de paralelización de tareas y datos y canalización de datos. La implementación en paralelo y la canalización de datos se ven facilitadas por la arquitectura de memoria compartida de Epiphany, y por el hecho de que un procesador en un núcleo puede escribir directamente en la memoria de cualquier otro núcleo del chip. Las plataformas integradas basadas en aceleradores disponibles en el mercado ofrecen una solución competitiva de bajo consumo energético para cálculos livianos de aprendizaje profundo en sistemas basados en CPU. Los clasificadores de baja complejidad utilizados en escenarios con restricciones de potencia y de rendimiento limitado se caracterizan por operaciones en mapas de imágenes pequeños con 2 a 3 capas profundas y pocas etiquetas de clase. Para estos casos de uso, los autores en(Hegde, G. (School of Computer Science and Engineering, Nanyang Technological University *et al.*, 2016) consideraron una gama de sistemas incorporados con presupuestos de energía de 5-20W, como la placa Xilinx ZC706 (con procesador de vectores blandos MXP), NVIDIA Jetson TX1 (GPU), TI Keystone II (DSP) como así como la placa Adapteva Parallella (multi-núcleo personalizado con NoC). Implementación más rápida y más eficiente de la energía en el viejo DSP TI Keystone II de 28nm sobre el nuevo SoC NVIDIA TX1 de 20nm en todos los casos, sin embargo, Parallella fue la opción de menor consumo de energía. En (Faber and Boryczko, 2016) se evalúa Parallella, una pequeña tabla con el coprocesador multiprograma Epiphany que consta de dieciséis núcleos MIMD

conectados por una red de malla en un chip. Las pruebas se basaron en algoritmos genéticos clásicos. Si bien se logró mejoras significativas en la velocidad, existen problemas, como el tamaño limitado de la memoria local y el acceso lento a la memoria, que dificultan la implementación de un código eficiente para Parallella. Dado que la eficiencia energética y el consumo de energía son el impedimento principal en el camino a los sistemas de escalados, los sistemas embebidos de alto rendimiento de baja potencia son cada vez más interesantes. En (Varghese *et al.*, 2017) se evalúa el rendimiento del sistema Epiphany para una variedad de operaciones básicas de computación y comunicación. Con los futuros sistemas que albergarán 4096 eCores, se comparan los méritos de la arquitectura Epiphany como un camino a los escalados con otros sistemas competitivos de eficiencia energética. En los experimentos, se observó el ancho de banda de DMA en chip de 2 GBytes/s, el ancho de banda de acceso a memoria compartida fuera de chip de 150 MBytes/s, la latencia de memoria en chip de 11 ns para el vecino más cercano.

Referencias

Arndt, O. J., Linde, T. and Blume, H. 2015 'Implementation and analysis of the histograms of oriented Gradients algorithm on a heterogeneous multicore CPU/GPU architecture', *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1402–1406. doi: 10.1109/GlobalSIP.2015.7418429.

Atweh, H. K. *et al.* 2018 'Parallelization of gradient-based edge detection algorithm on multicore processors', in *2018 Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*. IEEE, pp. 59–64. doi: 10.1109/DINWC.2018.8356996.

Bartok, R. and Vasarhelyi, J. 2015 'A fuzzy rule interpolation base algorithm implementation on different platforms', in *Proceedings of the 2015 16th International Carpathian Control Conference (ICCC)*. IEEE, pp. 37–40. doi: 10.1109/CarpathianCC.2015.7145041.

Bistaffa, F., Bombieri, N. and Farinelli, A. 2017 'An Efficient Approach for Accelerating Bucket Elimination on GPUs', *IEEE Transactions on Cybernetics*, 47(11), pp. 3967–3979. doi: 10.1109/TCYB.2016.2593773.

Bozejko, W., Dobrucki, A. and Walczynski, M. 2010 'Parallelizing of digital signal processing with using GPU', pp. 29–33.

Brauer, P., Lundqvist, M. and Mallo, A. 2016 'Improving Latency in a Signal Processing System on the Epiphany Architecture', *Proceedings - 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016*, pp. 796–800. doi: 10.1109/PDP.2016.51.

Caffarena, G. *et al.* 2010 'Fast fixed-point optimization of DSP algorithms', *Proceedings of the 2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip, VLSI-SoC 2010*, pp. 195–200. doi: 10.1109/VLSISOC.2010.5642659.

Carvalho, C. B. G., Ferreira, V. C. and Franca, F. M. G. 2017 'Towards a Dataflow Runtime Environment for Edge, Fog and In-Situ Computing', *Performance Computing*. doi: 10.1109/SBAC-PADW.2017.28.

Chen, G. *et al.* 2015 'Enabling Portable Optimizations of Data Placement on GPU', *IEEE Micro*, 35(4), pp. 16–24. doi: 10.1109/MM.2015.53.

Chen, J. *et al.* 2012 'A Hybrid Architecture for Compressive Sensing 3-D CT Reconstruction', *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(3), pp. 616–625. doi: 10.1109/JETCAS.2012.2221530.

Chen, J. *et al.* 2017 'A Hybrid Power-Performance Adjustment Strategy for Clustered Multi-threading Architecture', *Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016*, pp. 292–300. doi: 10.1109/HPCC-SmartCity-DSS.2016.0050.

Chen, L. *et al.* 2015 'A review of parallel computing for large-scale remote sensing image mosaicking', *Cluster Computing*. Springer US, 18(2), pp. 517–529. doi: 10.1007/s10586-015-0422-3.

Choi, J. and Rutenbar, R. A. 2016 'Video-Rate Stereo Matching Using Markov Random Field TRW-S Inference on a Hybrid CPU+FPGA Computing Platform', *IEEE Transactions on Circuits and Systems for Video Technology*, 26(2), pp. 385–398. doi: 10.1109/TCSVT.2015.2397198.

Cicuttin, M. *et al.* 2018 'GPU Accelerated Time-Domain Discrete Geometric Approach Method for Maxwell's Equations on Tetrahedral Grids', *IEEE Transactions on Magnetics*, 54(3). doi: 10.1109/TMAG.2017.2753322.

Dagum, L. and Menon, R. 1998 'OpenMP: an industry standard API for shared-memory programming', *IEEE Computational Science and Engineering*, 5(1), pp. 46–55. doi: 10.1109/99.660313.

Das, S. *et al.* 2018 'A Heterogeneous Cluster with

Reconfigurable Accelerator for Energy Efficient Near-Sensor Data Analytics', in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–5. doi: 10.1109/ISCAS.2018.8351749.

Datta, A. K. and Patel, R. 2014 'CPU Scheduling for power/energy management on multicore processors using cache miss and context switch data', *IEEE Transactions on Parallel and Distributed Systems*, 25(5), pp. 1190–1199. doi: 10.1109/TPDS.2013.148.

Dine, A. *et al.* 2016 'Graph-Based Simultaneous Localization and Mapping: Computational Complexity Reduction on a Multicore Heterogeneous Architecture', *IEEE Robotics & Automation Magazine*, 23(4), pp. 160–173. doi: 10.1109/MRA.2016.2580466.

Faber, Ł. and Boryczko, K. 2016 'Efficient parallel execution of genetic algorithms on Epiphany manycore processor', in, pp. 865–872. doi: 10.15439/2016F255.

Feng, Z., Zeng, Z. and Li, P. 2011 'Parallel on-chip power distribution network analysis on multi-core-multi-GPU platforms', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(10), pp. 1823–1836. doi: 10.1109/TVLSI.2010.2059718.

Le Gal, B. and Jego, C. 2016 'High-Throughput Multi-Core LDPC Decoders Based on x86 Processor', *IEEE Transactions on Parallel and Distributed Systems*, 27(5), pp. 1373–1386. doi: 10.1109/TPDS.2015.2435787.

Gao, F., Huang, Z., Wang, S., *et al.* 2017 'A hybrid clock synchronization architecture for many-core cluster system based on GPS and IEEE 1588', *2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016 - Proceedings*, pp. 2645–2649. doi: 10.1109/CompComm.2016.7925177.

Gao, F., Huang, Z., Wang, Z., *et al.* 2017 'An object detection acceleration framework based on low-power heterogeneous manycore architecture', *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, pp. 597–602. doi: 10.1109/WF-IoT.2016.7845407.

Gawande, G. S., Metkar, S. S. and Khanchandani, K. B. 2016 'Performance enhancement of multirate digital filter structures using advanced DSP optimization techniques', *Proceedings - IEEE International Conference on Information Processing, ICIP 2015*, pp. 307–311. doi: 10.1109/INFOP.2015.7489398.

Gegout, P. *et al.* 2014 'Ray-tracing of GNSS signal through the atmosphere powered by CUDA, HMPP and GPUs technologies', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(5), pp. 1592–1602. doi: 10.1109/JSTARS.2013.2272600.

Gener, Y. S., Yildiz, A. and Goren, S. 2016 'Low-cost and low-power video filtering with parallel many cores', in *ELECO 2015 - 9th International Conference on Electrical and Electronics Engineering*, pp. 921–925. doi: 10.1109/ELECO.2015.7394502.

Gomez-Luna, J. *et al.* 2016 'In-Place Matrix Transposition on GPUs', *IEEE Transactions on Parallel and Distributed Systems*, 27(3), pp. 776–788. doi: 10.1109/TPDS.2015.2412549.

Haidar, A. *et al.* 2017 'A Guide For Achieving High Performance With Very Small Matrices On GPU: A case Study of Batched LU and Cholesky Factorizations', *IEEE Transactions on Parallel and Distributed Systems*, 29(5), pp. 973–984. doi: 10.1109/TPDS.2017.2783929.

Hegde, G. (School of Computer Science and Engineering, Nanyang Technological University, S. 639798) *et al.* 2016 'CaffePresso: An optimized library for Deep Learning on embedded accelerator-based platforms', *International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*, pp. 1–10. doi: 10.1145/2968455.2968511.

Hellestrand, G. R. 1996 'VLSI Register , Ins tion and Data Caches Suited to on Chip Support for Real-Time Multi-Media Applications', pp. 1–4.

Hou, N. *et al.* 2017 'A Parallel Genetic Algorithm with Dispersion Correction for HW/SW Partitioning on Multicore CPU and Many-core GPU', *IEEE Access*, XX(c). doi: 10.1109/ACCESS.2017.2776295.

Huynh, B., Vo, B. and Snasel, V. 2017 'An Efficient Parallel Method for Mining Frequent Closed Sequential Patterns', *IEEE Access*, 5, pp. 17392–17402. doi: 10.1109/ACCESS.2017.2739749.

Hwang, I. and Pedram, M. 2016 'A Comparative Study of the Effectiveness of CPU Consolidation Versus Dynamic Voltage and Frequency Scaling in a Virtualized Multicore Server', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(6), pp. 2103–2116. doi: 10.1109/TVLSI.2015.2499601.

Ichnowski, J. and Alterovitz, R. 2014 'Scalable multicore motion planning using lock-free concurrency', *IEEE Transactions on Robotics*, 30(5), pp. 1123–1136. doi: 10.1109/TRO.2014.2331091.

Immune, U., Learning, D. and Optimization, S. 2016 'An Enhanced Approach for Parameter Estimation', *IEEE Systems, Man, and Cybernetics Magazine*, 2,(June), pp. 26–33. doi: 0.1109/MSMC.2015.2472915.

Intel Corporation (2017) POWER YOUR CREATIVITY WITH THE INTEL ® CORE TM X-SERIES INTEL ® CORE TM X-SERIES PROCESSOR FAMILY. Available at: https://www.intel.la/content/dam/www/public/us/en/documents/product-briefs/core-x-series-processor-family-product-brief.pdf (Accessed: 10 August 2018).

Jing, M. *et al.* 2018 'An Automatic Task Partition Method for Multi-core System', in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–5. doi: 10.1109/ISCAS.2018.8351528.

Kalla, B. *et al.* 2017 'A Probabilistic Monte Carlo Framework for Branch Prediction', *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 2017–Septe, pp. 651–652. doi: 10.1109/CLUSTER.2017.29.

Kanduri, A. *et al.* 2017 'Accuracy-aware power management for many-core systems running error-resilient

applications', *Ieeexplore.Ieee.Org*, 25(10), pp. 2749–2762. Available at: http://ieeexplore.ieee.org/abstract/document/7914763/.

Kee, C. Y. and Wang, C. F. 2013 'Efficient GPU implementation of the high-frequency SBR-PO method', *IEEE Antennas and Wireless Propagation Letters*, 12, pp. 941–944. doi: 10.1109/LAWP.2013.2274802.

Kehl, W. *et al.* 2017 'Real-Time 3D Model Tracking in Color and Depth on a Single CPU Core', *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 465–473. doi: 10.1109/CVPR.2017.57.

Khorgade, M. P. and Dakhole, P. 2016 'Optimization of reconfigurable fabric of DSP processor with image processing', *International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016*, pp. 1799–1801. doi: 10.1109/ICEEOT.2016.7754997.

Kim, T. J. *et al.* 2010 'RACBVHs: Random-accessible compressed bounding volume hierarchies', *IEEE Transactions on Visualization and Computer Graphics*, 16(2), pp. 273–286. doi: 10.1109/TVCG.2009.71.

Kulkarni, A. *et al.* 2017 'An Energy-Efficient Programmable Manycore Accelerator for Personalized Biomedical Applications', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(1), pp. 96–109. doi: 10.1109/TVLSI.2017.2754272.

Kumashiro, S. *et al.* 2017 'An Accurate Metric to Control Time Step of Transient Device Simulation by Matrix Exponential Method', pp. 37–40.

Kurth, A. *et al.* 2016 'Mobile Ultrasound Imaging on Heterogeneous Multi-Core Platforms', *Proceedings of the 14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia - ESTIMedia'16*, pp. 9–18. doi: 10.1145/2993452.2993565.

Labowski, K. L. *et al.* 2016 'Implementing Hilbert transform for Digital Signal Processing on epiphany many-core coprocessor', *2016 IEEE High Performance Extreme Computing Conference, HPEC 2016*. doi: 10.1109/HPEC.2016.7761638.

Lastovetsky, A. and Reddy Manumachu, R. 2017 'New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters', *IEEE Transactions on Parallel and Distributed Systems*, 28(4), pp. 1119–1133. doi: 10.1109/TPDS.2016.2608824.

Li, C. *et al.* 2013 'IMDCT optimization of AVS-P3 decoding algorithm in DSP', *2013 IEEE International Conference on Information and Automation, ICIA 2013*, (12), pp. 1364–1368. doi: 10.1109/ICInfA.2013.6720506.

Li, J. *et al.* 2015 'Accelerating MRI reconstruction via three-dimensional dual-dictionary learning using CUDA', *Journal of Supercomputing*. Springer US, 71(7), pp. 2381–2396. doi: 10.1007/s11227-015-1386-z.

Li, K., Zhu, Y. and Tian, Y. 2010 'Implementation and optimization of a weed identification algorithm on the DSP with C64+ core', *ISPACS 2010 - 2010 International Symposium on Intelligent Signal Processing and Communication Systems, Proceedings*, 6437, pp. 8–11. doi: 10.1109/ISPACS.2010.5704792.

Li, L. *et al.* 2015 'A Parallel Algorithm for Game Tree Search Using GPGPU', *IEEE Transactions on Parallel and Distributed Systems*, 26(8), pp. 2114–2127. doi: 10.1109/TPDS.2014.2345054.

Li, W. *et al.* 2017 'GPU Parallel Implementation of Isometric Mapping for Hyperspectral Classification', *IEEE Geoscience and Remote Sensing Letters*, 14(9), pp. 1532–1536. doi: 10.1109/LGRS.2017.2720778.

Li, Y. and Huang, X. 2017 'High speed communication and realization between FPGA and DSP in software-defined radio system', in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, pp. 2329–2332. doi: 10.1109/WiSPNET.2017.8300176.

Ling-bin, K. *et al.* 2010 'Realization and Optimization of Face Detection Algorithm Based on DSP', *Information Science and Management Engineering (ISME), 2010 International Conference of*, 1, pp. 246–249. doi: 10.1109/ISME.2010.74.

Liu, X. X., Yu, H. and Tan, S. X. D. 2015 'A GPU-accelerated parallel shooting algorithm for analysis of radio frequency and microwave integrated circuits', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(3), pp. 480–492. doi: 10.1109/TVLSI.2014.2309606.

Luo, Y. and Chen, Y. 2014 'Optimization of the SIFT key algorithms on multi-core DSP systems', *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology, ICCSNT 2013*, pp. 969–973. doi: 10.1109/ICCSNT.2013.6967265.

Luque, C. *et al.* 2012 'CPU accounting for multicore processors', *IEEE Transactions on Computers*, 61(2), pp. 251–264. doi: 10.1109/TC.2011.152.

Manumachu, R. R. and Lastovetsky, A. 2018 'Bi-Objective Optimization of Data-Parallel Applications on Homogeneous Multicore Clusters for Performance and Energy', *IEEE Transactions on Computers*, 67(2), pp. 160–177. doi: 10.1109/TC.2017.2742513.

Margara, A. and Cugola, G. 2014 'High-performance publish-subscribe matching using parallel hardware', *IEEE Transactions on Parallel and Distributed Systems*, 25(1), pp. 126–135. doi: 10.1109/TPDS.2013.39.

Masegosa, A. R., Martinez, A. M. and Borchani, H. 2016 'Probabilistic Graphical Models on Multi-Core CPUs Using Java 8', *IEEE Computational Intelligence Magazine*, 11(2), pp. 41–54. doi: 10.1109/MCI.2016.2532267.

Mastelic, T., Brandic, I. and Jaarevic, J. 2015 'CPU performance coefficient (CPU-PC): A novel performance metric based on real-time CPU resource provisioning in time-shared cloud environments', *Proceedings of the International Conference on Cloud*

Computing Technology and Science, CloudCom, 2015–Febru(February), pp. 408–415. doi: 10.1109/CloudCom.2014.13.

Mendat, D. R. *et al.* 2016 'Neuromorphic sampling on the SpiNNaker and parallella chip multiprocessors', *LASCAS 2016 - 7th IEEE Latin American Symposium on Circuits and Systems, R9 IEEE CASS Flagship Conference*, pp. 399–402. doi: 10.1109/LASCAS.2016.7451094.

Mielikainen, J. *et al.* 2016 'GPU Compute Unified Device Architecture (CUDA)-based Parallelization of the RRTMG Shortwave Rapid Radiative Transfer Model', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(2), pp. 921–931. doi: 10.1109/JSTARS.2015.2427652.

Mohammadi, M. *et al.* 2018 'A Hardware Architecture for Radial Basis Function Neural Network Classifier', *IEEE Transactions on Parallel and Distributed Systems*, 29(3), pp. 481–495. doi: 10.1109/TPDS.2017.2768366.

Muniyandi, R. C. and Maroosi, A. 2015 'Enhancing the simulation of membrane system on the GPU for the n-queens problem', *Chinese Journal of Electronics*, 24(4), pp. 740–743. doi: 10.1049/cje.2015.10.012.

Mustafa, B., Shahana, R. and Ahmed, W. 2015 'Parallel implementation of Doolittle Algorithm using OpenMP for multicore machines', *Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 2015*, pp. 575–578. doi: 10.1109/IADCC.2015.7154772.

Nakata, K. and Ito, Y. 2017 'An evaluation of the parallella architecture for the convex hull computation', in *Proceedings - 2016 4th International Symposium on Computing and Networking, CANDAR 2016*. IEEE, pp. 704–706. doi: 10.1109/CANDAR.2016.8.

Ng, R. Y. F., Tay, Y. H. and Mok, K. M. 2009 'DSP-based implementation and optimization of an iris verification algorithm using textural feature', *6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2009*, 5, pp. 374–378. doi: 10.1109/FSKD.2009.757.

Olofsson, A., Nordström, T. and Ul-Abdin, Z. 2015 'Kickstarting high-performance energy-efficient manycore architectures with Epiphany', *Conference Record - Asilomar Conference on Signals, Systems and Computers*, 2015–April, pp. 1719–1726. doi: 10.1109/ACSSC.2014.7094761.

Park, S. J. *et al.* 2011 'Hybrid core acceleration of UWB SIRE radar signal processing', *IEEE Transactions on Parallel and Distributed Systems*, 23(1), pp. 46–57. doi: 10.1109/TPDS.2010.117.

Pikacz, B. and Gambrych, J. 2014 'Vector implementation of the fast Fourier transform on DSP and NVIDIA CUDA platforms', in *2014 10th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. IEEE, pp. 1–4. doi: 10.1109/PRIME.2014.6872729.

Possa, P. R. *et al.* 2014 'A multi-resolution FPGA-based architecture for real-time edge and corner detection', *IEEE Transactions on Computers*, 63(10), pp. 2376–2388. doi: 10.1109/TC.2013.130.

Prongnuch, S. and Wiangtong, T. 2016 'Heterogeneous Computing Platform for data processing', *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 1–4. doi: 10.1109/ISPACS.2016.7824762.

Qiwei Cao and Liuchen Chang 2016 'Genetic Algorithm Optimization for High-Performance VSI-Fed Permanent Magnet Synchronous Motor Drives', *37th IEEE Power Electronics Specialists Conference*, pp. 1–7. doi: 10.1109/PESC.2006.1711959.

Ramalakshmi, E. and Kompala, N. 2017 'Multi-threading image processing in single-core and multi-core CPU using R language', in *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, pp. 1–5. doi: 10.1109/ICECCT.2017.8117873.

Richter, C., Schops, S. and Clemens, M. 2014 'GPU acceleration of algebraic multigrid preconditioners for discrete elliptic field problems', *IEEE Transactions on Magnetics*, 50(2), pp. 1–4. doi: 10.1109/TMAG.2013.2283099.

Ries, F., De Marco, T. and Guerrieri, R. 2012 'Triangular matrix inversion on heterogeneous multicore systems', *IEEE Transactions on Parallel and Distributed Systems*, 23(1), pp. 177–184. doi: 10.1109/TPDS.2011.103.

Rinku, D. R. 2017 'Analysis of multi-threading time metric on single and multi-core CPUs with Matrix multiplication', pp. 3–6.

Rzeszutek, R. *et al.* 2010 'An advantageous rotoscoping method', *IEEE Signal Processing Magazine*, 27(2), pp. 34–39. doi: 10.1109/MSP.2009.935392.

Van De Sande, K. E. A., Gevers, T. and Snoek, C. G. M. 2011 'Empowering visual categorization with the GPU', *IEEE Transactions on Multimedia*, 13(1), pp. 60–70. doi: 10.1109/TMM.2010.2091400.

Sevilla, J., Martin, G. and Nascimento, J. M. P. 2016 'Parallel Hyperspectral Unmixing Method via Split Augmented Lagrangian on GPU', *IEEE Geoscience and Remote Sensing Letters*, 13(5), pp. 626–630. doi: 10.1109/LGRS.2016.2522561.

Su, Y. *et al.* 2016 'An Efficient GPU Implementation of Inclusion-Based Pointer Analysis', *IEEE Transactions on Parallel and Distributed Systems*, 27(2), pp. 353–366. doi: 10.1109/TPDS.2015.2397933.

Torti, E. *et al.* 2014 'Real-time identification of hyperspectral subspaces', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6), pp. 2680–2687. doi: 10.1109/JSTARS.2014.2304832.

Torti, E. *et al.* 2016 'A Hybrid CPU-GPU Real-Time Hyperspectral Unmixing Chain', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(2), pp. 945–951. doi:

10.1109/JSTARS.2015.2485399.

Varghese, A. *et al.* 2017 ‘Programming the Adapteva Epiphany 64-core network-on-chip coprocessor’, *International Journal of High Performance Computing Applications*, 31(4), pp. 285–302. doi: 10.1177/1094342015599238.

Waidyasooriya, H. M. *et al.* 2017 ‘OpenCL-based FPGA-platform for stencil computation and its optimization methodology’, *IEEE Transactions on Parallel and Distributed Systems*, 28(5), pp. 1390–1402. doi: 10.1109/TPDS.2016.2614981.

Wang, J., Xie, C. and Pan, Z. 2013 ‘Optimization of DSP to generate spectrally efficient 16QAM Nyquist-WDM signals’, *IEEE Photonics Technology Letters*, 25(8), pp. 772–775. doi: 10.1109/LPT.2013.2251329.

Wang, T. and Kemao, Q. 2017 ‘Parallel computing in experimental mechanics and optical measurement: A review (II)’, *Optics and Lasers in Engineering*. Elsevier, 50(4), p. doi: https://doi.org/10.1016/j.optlaseng.2017.06.002.

Wang, T. and Kemao, Q. 2018 ‘Parallel computing in experimental mechanics and optical measurement: A review (II)’, *Optics and Lasers in Engineering*. Elsevier Ltd, 104(April 2017), pp. 181–191. doi: 10.1016/j.optlaseng.2017.06.002.

Wang, Y. *et al.* 2016 ‘A Learning Algorithm for Bayesian Networks and Its Efficient Implementation on GPUs’, *IEEE Transactions on Parallel and Distributed Systems*, 27(1), pp. 17–30. doi: 10.1109/TPDS.2014.2387285.

Wu, J., Song, Z. and Jeon, G. 2014 ‘GPU-parallel implementation of the edge-directed adaptive intra-field deinterlacing method’, *IEEE/OSA Journal of Display Technology*, 10(9), pp. 746–753. doi: 10.1109/JDT.2014.2319232.

Xanthis, C. G. *et al.* 2014 ‘MRISIMUL: A GPU-based parallel approach to MRI simulations’, *IEEE Transactions on Medical Imaging*, 33(3), pp. 607–617. doi: 10.1109/TMI.2013.2292119.

Xu, K. *et al.* 2016 ‘A real-Time task scheduling algorithm for multicore embedded systems’, *Proceedings - 2015 Chinese Automation Congress, CAC 2015*, pp. 1165–1170. doi: 10.1109/CAC.2015.7382674.

Xu, S. *et al.* 2018 ‘PIMCH : Cooperative Memory Prefetching in Processing-In-Memory Architecture’, *Asp-Dac 2018*, pp. 209–214.

Yang, W. *et al.* 2015 ‘Performance Optimization Using Partitioned SpMV on GPUs and Multicore CPUs’, *IEEE Transactions on Computers*, 64(9), pp. 2623–2636. doi: 10.1109/TC.2014.2366731.

Yu, D. *et al.* 2015 ‘A fast parallel matrix inversion algorithm based on heterogeneous multicore architectures’, *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 903–907. doi: 10.1109/GlobalSIP.2015.7418328.

Zhang, H. *et al.* 2018 ‘High-Speed Visible Image Acquisition and Processing System for Plasma Shape and Position Control of EAST Tokamak’, *IEEE Transactions on Plasma Science*, 46(5), pp. 1312–1317. doi: 10.1109/TPS.2018.2805911.

Zhang, J., Wang, H. and Feng, W. C. 2017 ‘CuBLASTP: Fine-Grained Parallelization of Protein Sequence Search on CPU+GPU’, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(4), pp. 830–843. doi: 10.1109/TCBB.2015.2489662.

Zhou, G. *et al.* 2017 ‘GPU-accelerated batch-ACPF solution for N-1 static security analysis’, *IEEE Transactions on Smart Grid*, 8(3), pp. 1406–1416. doi: 10.1109/TSG.2016.2600587.