Filtro FIR con Procesamiento Paralelo Masivo

Ramón Díaz de León Zapata

Instituto Tecnológico de San Luis Potosí, Av. Tecnológico s/n Soledad de Graciano Sánchez, San Luis Potosí, C.P. 78376, México, teléfono (444) 8182136

ramondz@hotmail.com

Gloria del Carmen Rendón Sustaita

Instituto Tecnológico de San Luis Potosí, Av. Tecnológico s/n Soledad de Graciano Sánchez, San Luis Potosí, C.P. 78376, México, teléfono (444) 8182136

gloria_rendon77@hotmail.com

Resumen

El cómputo paralelo masivo ha demostrado ser una técnica viable para incrementar hasta en 10 órdenes de magnitud la velocidad de cálculo y/o procesamiento de algunas etapas de algoritmos seleccionados.

Una de las empresas pioneras tanto en la fabricación de hardware como en la innovación de software para el cómputo paralelo masivo es NVIDIA a través de sus tarjetas gráficas GeForce y su compilador CUDA, que lo hacen ideal para experimentar la paralelización de procesos de cálculo complejos a un costo razonable.

Si bien el cómputo paralelo masivo no es exclusivo para un área particular de la ciencia, la ingeniería o la técnica, ha encontrado un segmento de desarrollo especialmente interesante en el Control Digital, ya que entre otros temas, se abarca el de los filtros digitales y particularmente aquellos basados en la técnica denominada de Respuesta Finita al Impulso (FIR), que son paralelos por su naturaleza no recursiva.

Comprender, pero sobre todo aprovechar al cómputo paralelo masivo en áreas del Control Digital abre un aspecto inexplorado en nuestro Instituto y será factor de motivación tanto para alumnos como para maestros que deseen explotar esta potente tecnología.

Se expone en el presente trabajo, la descripción teórica de la paralelización de un caso aplicativo para el filtro FIR pasa banda en el rango audible humano.

Palabras Claves: CUDA, Filtro, FIR, Procesamiento Paralelo Masivo.

1. Introducción

La Unidad de Procesamiento de Gráficos (GPU por sus siglas en inglés) es la responsable de manipular y desplegar datos gráficos. Las recientes GPUs tienen centenares de Unidades Centrales de Procesamiento (CPU) y por tanto resultan excelentes para realizar operaciones en paralelo [1]. Cada CPU puede correr centenares de hilos (Threads) en paralelo. Aquellas tarjetas gráficas que permiten su uso no sólo para aplicaciones concretas de gráficos, sino para aplicaciones genéricas son llamadas Unidades de Procesamiento de Gráficos de Uso General (GPGPU), sin embargo las GPGPU fueron concebidas para usos gráficos y sus interfaces de programación así lo hacen evidente; hacer un programa genérico implicaba adaptar la sintaxis y pensar en la solución que nada tenía que ver con gráficas, como si lo fuera [2]. Con la llegada de la Arquitectura Unificada de dispositivo de Cómputo (CUDA) ya no es necesario pensar en una solución genérica como si fuera una gráfica, ya que en esencia se programa en lenguaje C con algunas adiciones propias de las tarjetas gráficas que lo soportan, específicamente las de la familia GeForce de la empresa NVIDIA [3].

Los filtros digitales se han considerado por sus características de diseño en dos grandes tipos, los IIR y los FIR. Los primeros se fundamentan en una analogía directa con los filtros analógicos realimentados y por tanto son también llamados recursivos, característica que no los convierte en los idóneos para ser tratados por métodos paralelos de cálculo, aunque incluso en estos se ha notado un incremento en la velocidad de respuesta hasta 4 veces más rápidos [4].

Por otra parte los filtros FIR tienen una naturaleza eminentemente paralela y resultan los idóneos para ser tratados por el cómputo paralelo masivo, logrando incrementar la

velocidad de respuesta hasta en 40 veces, lo que permitiría su aplicación en filtrados de alta velocidad, como pueden ser instrumentos ópticos o filtrado de imágenes o video en tiempo real sólo por mencionar algunas.

La falta de conocimiento en este rubro hace suponer una limitante para que se utilicen técnicas que aprovechen el cómputo paralelo masivo en la solución de problemas exigentes que con los procedimientos convencionales requieren ser analógicos o que incluso sean irrealizables.

Los filtros digitales con respuesta finita al impulso son considerados como la aplicación más básica del procesamiento digital de señales. Los filtros FIR pueden describirse en el dominio del tiempo como una ecuación para su señal de salida y(n):

$$y(n) = \sum_{k=0}^{M-1} h(k)\, x(n-k) \qquad (1)$$

Donde M es el orden del filtro, x(n) es la señal de entrada, h(k) es la respuesta finita al impulso y las condiciones iniciales a cero se asumen cuando (n - k) < 0.

Debido a que la naturaleza de la ecuación resulta de un proceso que puede paralelizarse directamente, en CUDA existen dos posibilidades para ser implementado:

Cada hilo de proceso enumera una sola salida dentro del m-ésimo ciclo, por ejemplo: el producto punto de los vectores del filtro h(0…m-1) y la entrada x(n-m+1…n) son multiplicados por M-hilos en paralelo y posteriormente se paraleliza la suma para calcular la salida. El valor de n se incrementa en la siguiente iteración del ciclo.

Grupos de M-hilos enumeran una salida dentro del lazo de M-ésima longitud, esto es: los vectores respuesta del filtro h(0…M-1) y la entrada x(n-M+1…n) son multiplicados entre sí por M-hilos en paralelo y entonces se procede a la paralelización de la suma para calcular la salida. El valor de n se incrementa en la siguiente iteración del ciclo.

Previas investigaciones han demostrado que el mejor rendimiento se logra con el primer modelo [5], por lo que será el seleccionado inicialmente en las pruebas del presente

proyecto, sin descuidar procedimientos alternativos que sugieran mejoras en velocidad u optimización de código.

2. Desarrollo

La radio experimentación ha proveído importantes avances en el rubro de las telecomunicaciones ya que las bandas de radiofrecuencia asignadas y su uso está reglamentado por leyes federales que exigen una licencia especial para poder operarlas. El Instituto Tecnológico de San Luis Potosí cuenta con una licencia de radio club y catedráticos con licencia para operarlo y experimentar con temas de telecomunicaciones para aplicar en la práctica cotidiana los temas teóricos que se estudian en materias como Control Digital, Telecomunicaciones, Teoría electromagnética, etc.

Una de esas bandas de experimentación se ha seleccionado por su interés particular, ya que puede ser utilizada tanto diurna como nocturnamente con aproximadamente las mismas y regulares características, sin embargo se trata también de una banda que suele llamarse en el argot de la radioexperimentación como “ruidosa”, se trata de la banda de HF de 20 metros (de longitud de onda) y escogida en particular para las pruebas en la frecuencia de 14,130 MHz ya que es la frecuencia utilizada por la Red Nacional de Emergencia [6] en casos de desastres naturales o cualesquiera otras contingencias que requieran con carácter de urgente la transmisión de comunicados, por lo que la calidad de las transmisiones debe intentar garantizarse y esto puede lograrse con el apoyo del adecuado filtrado del canal de comunicación.

El rango audible humano se considera entre los 20 y 20,000 Hz y de ese rango la voz humana (que es el de nuestro interés) cubre el espectro entre los 300 y 3400 Hz para aplicaciones tecnológicas como la telefonía o la voz por internet (VoIP) [7] por lo que se propone la construcción del filtro pasa banda en este rango de frecuencias y con frecuencias de corte inferior en 100 Hz y superior en 3600 Hz, como se muestra en la Fig. 1.

La velocidad mínima teórica necesaria para poder implementar este filtro viene determinada por el tiempo de muestreo requerido para procesar señales en la frecuencia de máximo valor, esto es en la frecuencia de los 3600 Hz, que equivaldría aproximadamente a 147 μs, sin embargo por el principio del teorema del muestreo par sistemas digitales, requerimos que el tiempo de muestreo sea por lo menos de la mitad de este valor, es decir aproximadamente 73 μs. Es pertinente señalar que este tiempo teórico no corresponde a la realidad, ya que se deben contemplar los tiempos de latencia del sistema de cómputo, así como los tiempos que toman diferentes instrucciones en ejecutarse pues es sabido que una multiplicación por ejemplo, puede tomar hasta cuatro ciclos de reloj en ejecutarse, mientras que una suma lo haría en un solo ciclo de reloj. Tomando esto en cuenta, se suele tomar un valor aún más pequeño de tiempo de muestreo, para este caso se optó por los 40 μs que es aproximadamente la mitad de este valor, con lo que aseguramos que los tiempos extras no resultrán significativos. Como podrá observarse en la tabla 2 de la sección de resultados, tanto el algoritmo serial como el paralelo son capaces de procesar el filtrado que se requiere, sin embargo el algoritmo serial se acerca mucho al valor del periodo de muestro, por lo que si se requiriera una mejoría al filtro, que se traduce en un incremento de taps, resultaría insuficiente el tiempo de procesamiento planteado.

La Fig. 2 muestra la respuesta en frecuencia esperada del filtro diseñado, donde puede apreciarse la característica “ventana” de los filtros pasa banda.

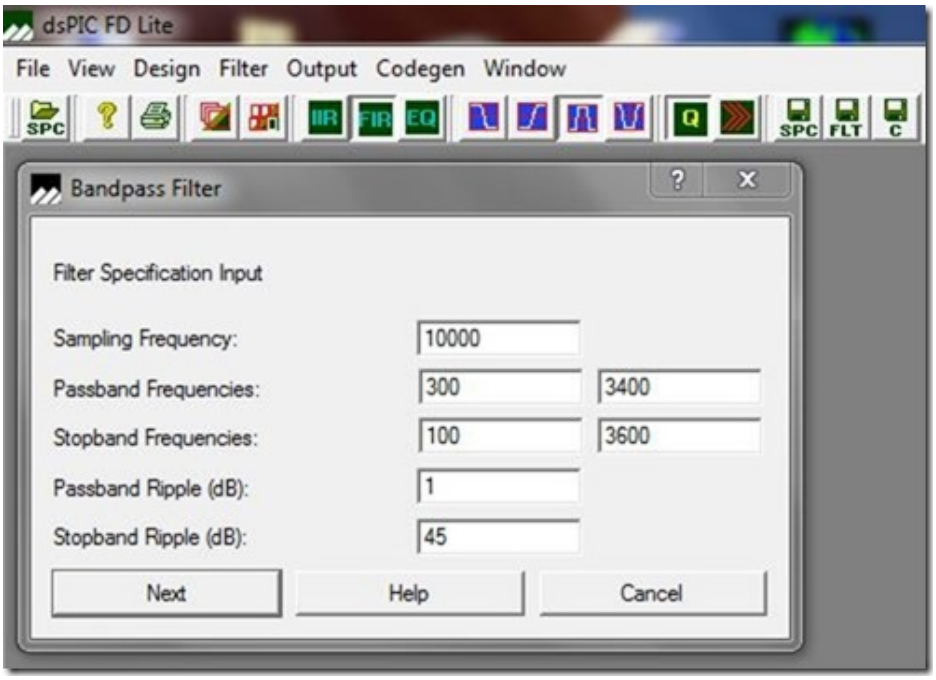

Fig. 1. Valores de diseño para el filtro FIR pasa banda.

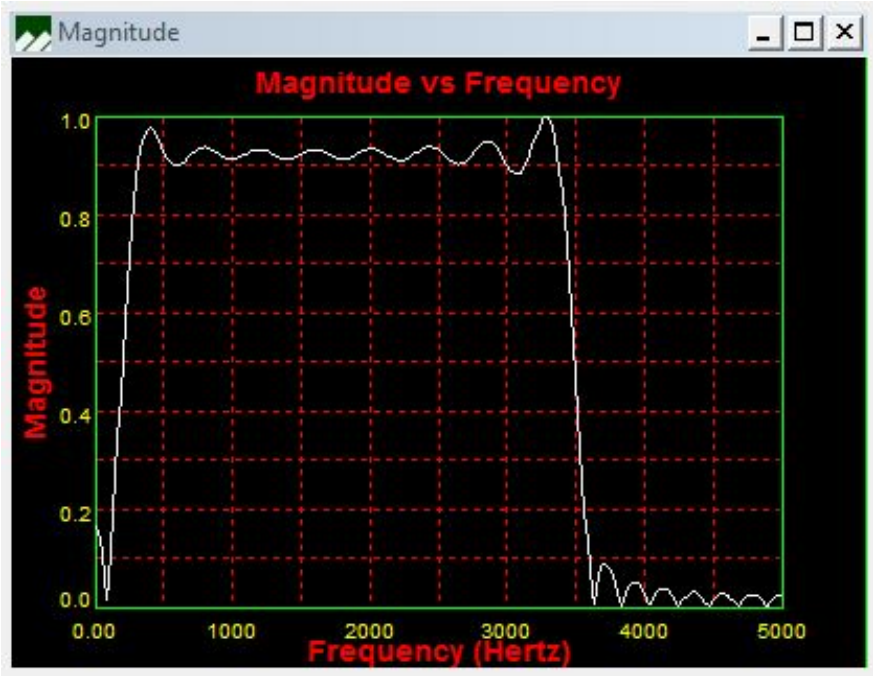

Fig. 2. Respuesta en frecuencia calculada del filtro pasa banda.

Obtención de los coeficientes o “taps” del filtro FIR.

Para obtener los coeficientes se ha utilizado el programa DSPicFD LITE de la empresa Microchip [8], seleccionando el método de ventana cuadrada (el más simple de los métodos) como estrategia para poder ser analizado y comparado más fácilmente en las etapas posteriores del proyecto. El programa genera un total de 47 taps de los cuales un extracto se muestra en la tabla 1 con fines ilustrativos y comparativos para los lectores interesados en reproducir el presente trabajo.

Tap #	Valor del coeficiente
0	0.7629394531250000E-03
1	-.1757812500000000E-01
2	0.4577636718750000E-02
3	-.8605957031250000E-02
4	-.2304077148437500E-01
5	0.2929687500000000E-02
6	-.1986694335937500E-01
7	-.2734375000000000E-01
8	0.9460449218750000E-03
9	-.3286743164062500E-01
10	-.2947998046875000E-01
11	-.1129150390625000E-02
12	-.4776000976562500E-01

Tabla 1. Fragmento de taps calculados.

Algoritmo de paralelización

Dado que el objetivo es obtener los resultados del filtrado en el momento en que la información está siendo recibida, se debe aplicar un proceso previo a los datos de descomposición en sus diferentes frecuencias individuales a través de la transformada rápida de fourier (FFT, Fast Fourier Transform), aplicar la paralelización a los datos obtenidos de la transformación previa y una vez que los cálculos de la multiplicación de cada tap ha sido obtenida, se procede a realizar la transformada inversa rápida de fourier (IFFT, Inverse Fast Fourier Transform) como lo sugieren Wefers y Berg [9] para que los datos puedan ser utilizados por la etapa de salida para su conversión digital a analógica y poderlos escuchar como sonido a través de la bocina correspondiente. La Fig. 3 muestra la estructura algorítmica y los procesos asociados.

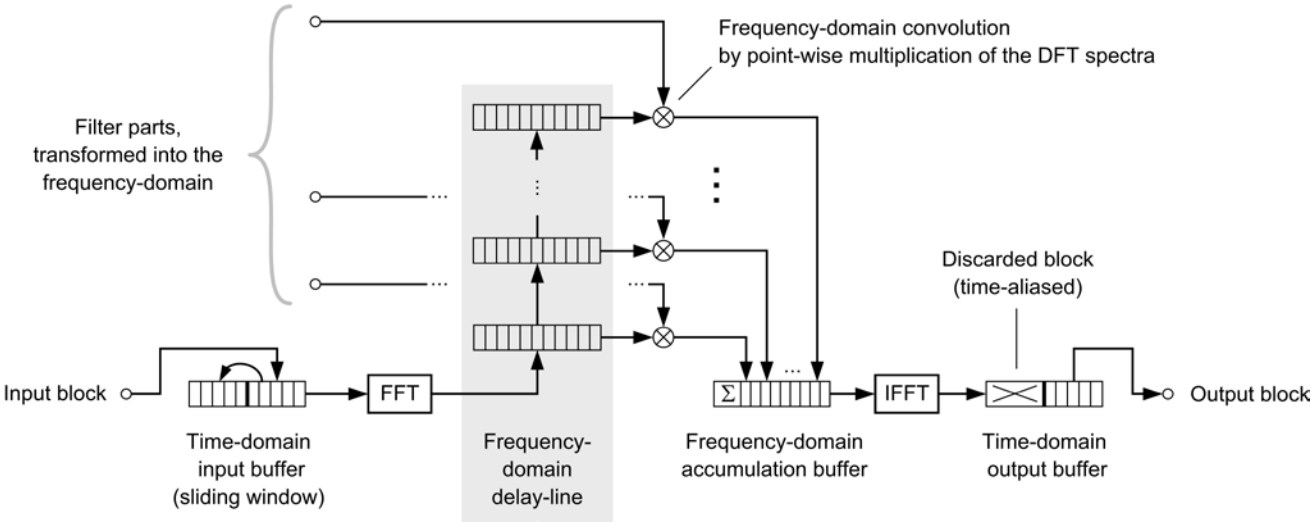

Fig. 3. Algoritmo de paralelización sugerido por Wefers y Berg [8] aplicado al proyecto.

Por otra parte, la ecuación (1), no puede ser implementada directamente en un sistema computacional, y debe tomar la forma de la ecuación (2) para lograr tal fin:

$$y[i] = \sum_{i=0}^{M} x[i * N] * tapf[i * N] \quad (2)$$

Donde $tapf[i * N]$ es la función de taps que en este caso han sido calculados por separado y colocado los valores numéricos de sus coeficientes en una variable de tipo *array* (vector lineal) en el programa. Cabe mencionar que se ha dejado la ecuación (2) con la función de taps como conveniencia para un trabajo futuro en el que se pretende implementar filtros dinámicos que podrán adaptarse a condiciones cambiantes de los datos de entrada y para ello deberán recalcularse los coeficientes de la función mientras se evalúan las entradas.

3. Implementación del algoritmo y pruebas

El listado 1 muestra el pseudo código de paralelización del filtro FIR en CUDA.

```
k = GPUThreadID; // Índice del elemento

c = channelof(k); // Número de canal

n = FDLCursor; // Último valor introducido

// Inicialiación del acumulador

accu = 0;

// Iteraciones de cada sección del filtro

for i=0 to NumFilterParts-1 {

// Multiplicación-adición de cada valor complejo

x = FDL[line=n][element=k];

y = FilterSpectrum[channel=c, part=i];

ComplexMulAdd(src=x, src=y, dest=accu);
```

```
n = (n+1) mod NumFilterParts;

}

// Escritura del resultado en la memoria

output[k] = accu;
```

Listado 1. Pseudo código de paralelización del filtro FIR.

Las implementación del filtro se realizó en una computadora MacBook Pro 15 pulgadas (Mid 2012) con Sistema oprativo OS X Mavericks versión 10.9.3 con procesador Intel Core io7 a 2.3 GHz con 4 GB de memoria RAM a 1600 MHz DDR3, tarjeta gráfica NVIDIA GeForce GT 650M con 512 MB, CUDA C V.5.5. y compilador Objective C que forma parte del Entorno Integrado de Desarrollo Xcode de Apple v.5.1.1.

La Fig. 4 muestra la pantalla de desarrollo del programa del filtro FIR en esta plataforma.

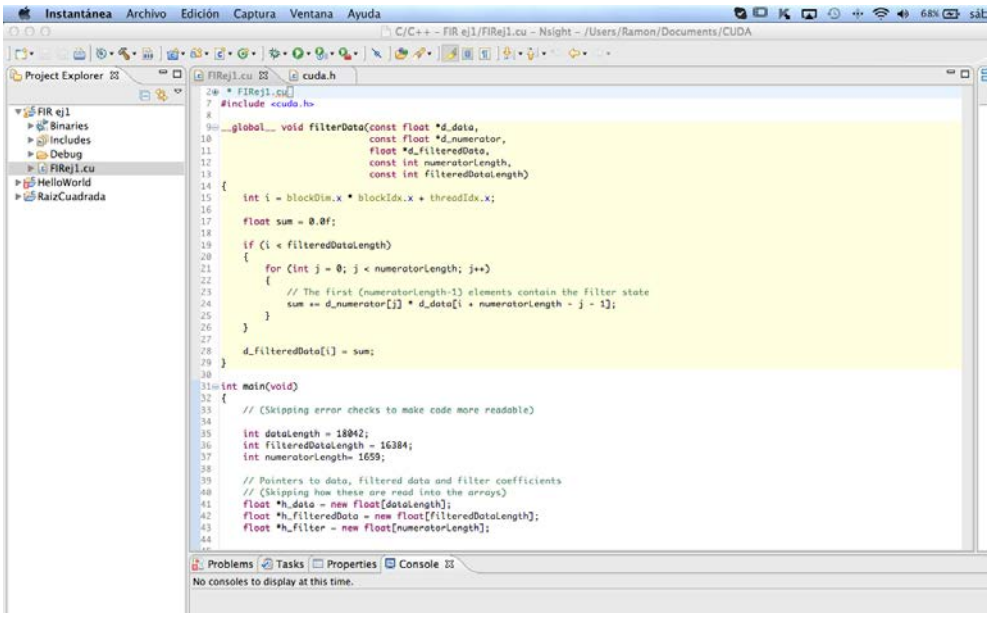

Fig. 4. Pantalla de desarrollo, pruebas y depuración del filtro FIR.

Las pruebas de la respuesta del diseño del filtro se llevaron a cabo con el apoyo del programa DSPicWorks de la empresa Microchip, únicamente con el fin de verificar que los coeficientes de la función de los taps fueran los adecuados. Con este programa podemos generar los datos correspondientes a la simulación de una señal de audiofrecuencia en el rango en que no presenta atenuación, por ejemplo en los 1500 Hz (Gráficas superiores; la izquierda es la entrada y la derecha la salida del filtro) y una entrada con una frecuencia de 3600 Hz que debería presentar una máxima atenuación (gráficas inferiores; la izquierda representa la entrada y la derecha la salida filtrada), con lo que se verifica la correcta operación de convolución de la señal de entrada con la función de taps (figura 5) para posteriormente proceder a la inclusión de los coeficientes en el programa del filtro escrito en CUDA apoyados por el algoritmo del listado 1.

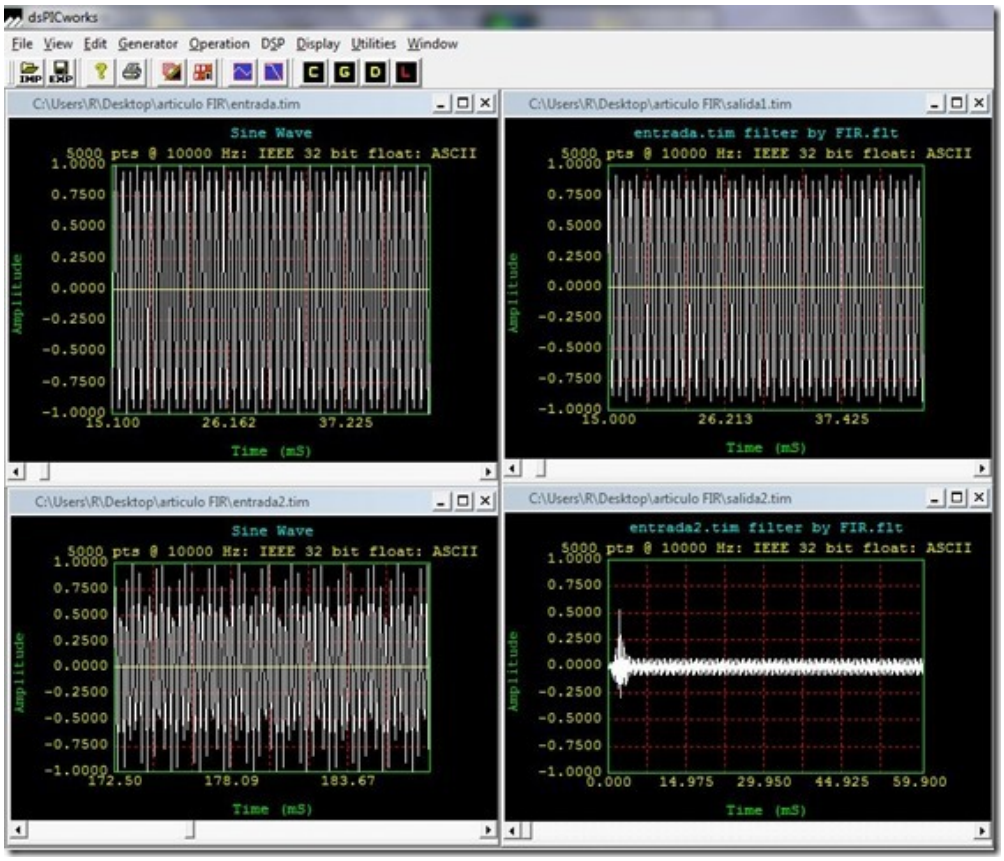

Fig. 5. Señal de prueba para el filtro FIR con componentes de alta frecuencia.

Para fines comparativos en la ganancia en velocidad de procesamiento, se implementó una prueba de ejecución sin el algoritmo de paralelización (algoritmo serial) en la misma computadora a través del CPU. Posteriormente se realizó la misma prueba a través de la GPU con el lagoritmo de paralelización. El experimento se llevó a cabo con 47. La tabla 2 muestra los resultados obtenidos.

	GPU (paralelo)	**CPU (serial)**	**Relación CPU/GPU**
Tiempo	4.08 µs	32.16 µs	7.88

Tabla 2. Tiempos de ejecución de los algoritmos y su relación de ganancia.

Tras la ejecución del algoritmo paralelo en CUDA, los datos obtenidos del proceso de filtrado (contenido de la variable *output[k]*, ver listado 1) que se guardan en un archivo de texto, se comparan con los obtenidos por el programa DSPicWorks como método de corroboración del correcto cálculo de la salida del filtro, ya que por el momento no se ha implementado un entorno gráfico que exponga estos datos sobre la misma plataforma CUDA, situación que se deja para una etapa posterior en caso de considerarlo conveniente. Independientemente de si tratase del algoritmo serial o paralelo, los resultados deben ser y de hecho fueron los mismos, como se constata en la comparación de los archivos obtenidos con DSPicWorks y CUDA.

4. Conclusiones

Un filtro pasa banda con aplicación para mejorar la calidad de recepción de las telecomunicaciones en bandas de experimentación inheremente “ruidosas” (con componentes de frecuencia que dificultan la claridad de la recepción de voz) pueden ser resueltas con filtros analógicos convencionales, sin embargo, la naturaleza cambiante de esas interferencias requeriría la adición de componentes con capacidades de variación de sus parámetros (varicaps y reóstatos) que implicarían delicados y complicados ajustes con auxilio de instrumentos de medición e incluso en ocasiones el rediseño completo.

Con la implementación en su versión digital del mismo tipo de filtro obtenemos ventajas importantes, una de ellas la capacidad de ajuste y calibración por software sin necesidad de componentes externos, pero tal ventaja demanda una capacidad de cómputo serial muy exigente o en su defecto el sacrificio de la calidad del filtrado; afortunadamente la naturaleza inherentemente paralela del tipo de filtro digital FIR puede ser implementado en una computadora que tenga instalada una tarjeta gráfica que permita el procesamiento paralelo a través de los centenares de núcleos con los que consta y obtenemos la calidad, velocidad de procesamiento y ajustes finos que deseemos por software. Otra de las ventajas es la posibilidad de permitir cálculos adicionales, como los taps de la función de coeficientes para reajustar las características del filtro dinámicamente que serían complejas, voluminosas en dimensiones físicas y prácticamente imposibles de implementar con filtros analógicos convencionales.

Este trabajo deja asentadas las bases teóricas y técnicas para la implementación de un filtro digital FIR pasa banda para un caso aplicativo real en la banda de radio-experimentación de 20 metros (frecuencia de 14,130 Hz) para aplicaciones de audio en el rango de frecuencias de la voz.

Trabajo futuro

Se Implementará el algoritmo y se construirán las interfaces correspondientes para su uso con la estación de radio-experimentación con la que cuenta el Instituto Tecnológico de San Luis Potosí y se ampliarán los estudios para conseguir el filtro dinámico que permita adaptarse automáticamente a las condiciones de otras frecuencias sin requerir ajustes adicionales por parte del usuario.

Se agregará una interfaz gráfica que muestre simultáneamente al proceso de cálculo, las señales de entrada y salida para verificar de manera inmediata la calidad del filtro.

Agradecimientos

A PROMEP, ahora PRODEP (Programa para el Desarrollo Profesional Docente) por el beneficio otorgado a través del "Apoyo a la Incorporación de Nuevos PTC".

5. Referencias

[1] Sajid Anwar et al. "Digital Signal Processing Filtering With GPU", School of Electrical Engineering Seoul National University (2010).

[2] Jason Sanders. "CUDA by Example" Addison Wesley (2011).

[3] Tomas Mazanec, "Application of CUDA in DSP". UTIA (2009).

[4] Lyons Richard G., "Understanding Digital Signal Processing". Pearson Education Inc. (2011).

[5] Mark McCurry "CUDA Based Polyphase Filter", reu program: mit haystack 2011: polyphase filter banks (2011).

[6] "Cuadro Nacional de distribución de frecuencias" IFETEL (2014).

[7] Beraneck Leo L. "Acoustics" Mc. Graw Hill (1954).

[8] Microchip Inc. www.microchip.com (2014)

[9] Wefers F. y Berg J. “high-performance real-time fir-filtering using fast convolution on graphics hardware”, Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10), Graz, Austria, September 6-10, (2010).

6. Autores

M. en C. Ramón Díaz de León Zapata obtuvo su título de Maestría en Ciencias con especialidad en Ciencias de la Computación por el Instituto Tecnológico de San Luis Potosí. Candidato a Doctor en Ciencias Experimentales con especialidad en Nanomateriales por la Universidad Autónóma de S. L. P.

Ing. Gloria del Carmen Rendón Sustaita, es Ingeniero en Sistemas Computacionales. Catedrática del ITSLP impartiendo las materias de Inteligencia Artificial y Sistemas Expertos, entre otras materias.