Utilización de FPGAs como aceleradores de cálculo

Juan P. Oliver, Julio Pérez Acle

Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Montevideo - Uruguay

ABSTRACT

Although computers are becoming faster all the time, the need for calculation is also heavier and time consuming. There are various ways to increment the speed of a computer: a faster processor, many processors working in parallel and sharing tasks, and so on. This paper describes experiences where a different approach was used to increase speed, using electronic circuits dedicated to do part of the job and collaborate with the main processor of a computer. This alternative was so expensive until recently that its use was unthinkable in general terms. But when big sized user programmable chips (FPGAs) appeared, together with the possibility of infinite reprogramming according to the needs of the given applications, this became a realistic option. To be able to use this technology widely it is necessary to solve many stages, mainly the design of the circuits to be programmed in the chips as well as the designing of the hardware - software interfaces of application.
The architecture that was used was based on a reconfigurable board working as a coprocessor together with a PC, and hardware libraries were implemented to accelerate certain specific calculations. The tested algorithms have been artificial neural networks, image processing and data encryption.

Keywords: FPGA, reconfigurable hardware.

RESUMEN

Si bien las computadoras son cada vez más rápidas, las necesidades de cálculo también se hacen cada vez más pesadas y consumen más tiempo y recursos. Existen varias formas de aumentar la velocidad de un computador: procesador más rápido, varios procesadores trabajando en paralelo que se distribuyan tareas, etc. Este artículo describe experiencias realizadas utilizando otra forma de aumentar la velocidad con el uso de circuitos electrónicos dedicados que realicen cierta parte del trabajo y compartan la tarea con el procesador central de una computadora. Esta alternativa hasta hace poco presentaba un costo tan alto que era impensable su utilización en términos generales, pero con la aparición de chips de gran tamaño programables por el usuario (FPGAs), y la posibilidad de reprogramarlos infinitas veces de acuerdo a las necesidades de la aplicación concreta hacen que sea una opción viable. Para poder usar esta tecnología en forma amplia es necesario resolver varias etapas, principalmente el diseño de los circuitos a programar en los chips así como el diseño de las interfaces hardware - software de aplicación.
La arquitectura utilizada se basó en una placa reconfigurable funcionando como un coprocesador en conjunto con un PC, y se realizaron bibliotecas hardware para acelerar ciertos cálculos específicos. Los algoritmos probados han sido de redes neuronales, tratamiento de imágenes, y encriptado.

Palabras Clave: FPGA, lógica programable, reconfigurable.

1. INTRODUCCION

Tradicionalmente existen dos alternativas a la hora de implementar un cierto algoritmo: su ejecución en un procesador de propósito general o su realización en hardware a medida. La primera es muy barata y flexible, pero generalmente lenta; la segunda es de muy alta performance, pero por su costo sólo se justifica en aplicaciones de uso masivo (coprocesadores numéricos, chips dedicados de procesamiento de señales, etc.) y generalmente no permite flexibilidad. La lógica reconfigurable permite combinar estas dos soluciones y obtener velocidades

hardware con flexibilidad software. La posibilidad de reutilización del hardware reconfigurable abarata su costo ya que puede utilizarse exactamente el mismo hardware para varias aplicaciones cambiando exclusivamente su programación interna [1].
Este trabajo describe una serie de experiencias realizadas en los últimos cuatro años en el Instituto de Ingeniería Eléctrica (IIE) tendientes a analizar la factibilidad de realizar algoritmos en lógica reconfigurable como forma de aumentar la velocidad de ejecución de los mismos.

La utilización de lógica programable o reconfigurable para acelerar algoritmos de diverso tipo viene siendo aplicada con éxito desde hace ya algunos años [1][2]. En este trabajo se utiliza el esquema de lógica reconfigurable trabajando en paralelo con la CPU de un computador personal, la lógica reconfigurable realiza la parte más pesada de un determinado algoritmo, aumentando así su velocidad de ejecución.
Para poder ampliar la utilización de esta tecnología a aplicaciones de potenciales usuarios es necesario salvar la brecha que existe entre las herramientas de desarrollo diseñadas para especialistas en el área de diseño digital y las aplicaciones finales. En este sentido aparecen dos grandes tendencias: la creación de lenguajes y compiladores de alto nivel que sean capaces de traducir y trasladar automáticamente determinados algoritmos a hardware reconfigurable; y el desarrollo de bibliotecas o módulos reutilizables que puedan ser invocados por las aplicaciones. Esta última solución es la que se ha buscado en la mayoría de los casos descritos en este trabajo.

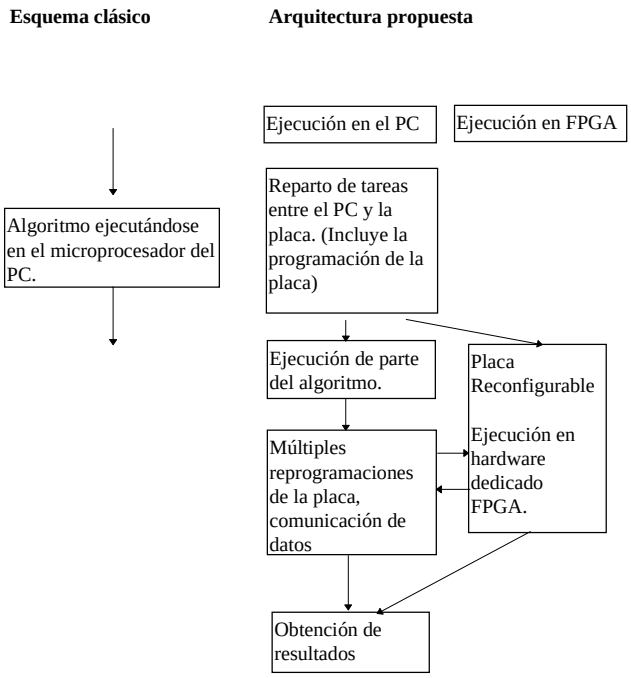

Figura 1 - Coprocesamiento

La arquitectura utilizada consta de una placa reconfigurable trabajando en conjunto con el microprocesador del PC en lo que llamamos modalidad coprocesador. La idea básica es poder partir un determinado algoritmo y realizar una parte en el microprocesador y otra en la placa reconfigurable. Además el PC se encarga del reparto de las tareas y de la reprogramación de la placa, así como de toda la interfaz con el usuario.
El esquema de la Figura 1 muestra la arquitectura utilizada en comparación con el método clásico.
Dependiendo del tipo de aplicación los puntos críticos son la capacidad de procesamiento de la placa reconfigurable, cantidad de memoria disponible en la misma, su bus de datos o lo que es lo mismo el ancho de banda de entrada salida de datos, y la velocidad de reconfiguración.

Hardware Reconfigurable
Se describen trabajos realizados con las placas RIPP10 y UP1 de Altera. Las UP1 no permiten trabajar en modo coprocesador pero nos parece interesante incluir un par de experiencias realizadas con estas placas por utilizar chips de la familia 10K.

Plataforma RIPP 10
La plataforma hardware utilizada está formada por un computador PC y una placa reconfigurable RIPP10 de ALTERA conectada en el bus ISA [9].
La placa RIPP10 tiene un grupo (“array”) de ocho FPGAs FLEX81188 interconectados totalizando 8064 celdas lógicas. Cada uno de estos chips está conectado con sus cuatro vecinos a través de buses de 32 bits. Un bus global de 36 bits provee acceso a todos los chips del array desde un FLEX8452 que es el que realiza la interfaz con el bus ISA.

La placa posee además cuatro zócalos de RAM estática con un chip de 128KB cada uno. Cada chip de RAM es accesible desde un par de 81188. El bus global está directamente conectado a una interfaz externa mediante buffers.
Tanto los chips del array (81188) así como el chip de interfaz (8452) son programados a través del bus ISA. El proceso consiste en programar primero el chip de interfaz con un circuito que permita la programación de los chips del array. Luego se programan los chips del array uno por vez a través del bus global. Finalmente, si es necesario, se puede cargar un nuevo diseño en el chip de interfaz de acuerdo a la aplicación.
La RIPP10 presenta varias ventajas y desventajas desde el punto de vista de nuestras aplicaciones. Es una plataforma flexible que permite evaluar fácilmente diferentes soluciones para un determinado problema. Tiene una gran cantidad de lógica disponible y buena interconexión entre chips. Además la simplicidad del bus ISA permite prototipar rápidamente diferentes soluciones.

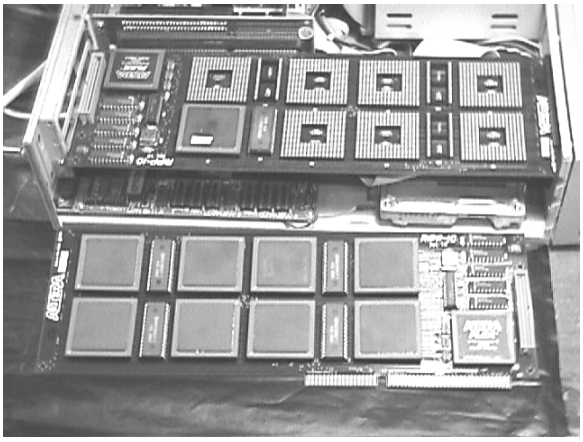

Figura 2 - Dos placas RIPP10

Por otra parte esta plataforma tiene algunas desventajas si se la compara con placas reconfigurables más modernas. No existe manera de reconfigurar en forma independiente cada uno de los chips del array, la única manera es borrar de una vez la configuración en todo el array y luego cargar los nuevos circuitos en cada uno de los ocho chips. Este es un proceso lento que limita severamente la posibilidad de multiplexar en el tiempo diferentes circuitos sobre la RIPP10. La velocidad y ancho de banda del bus ISA hoy en día han sido ampliamente superados por una interfaz PCI.

Placa UP1

La placa UP1 es una placa de uso educacional también de la empresa Altera, si bien su entrada salida es muy limitada y no permite trabajar en un modo de coprocesamiento posee un chip de la familia FLEX10K (EPF10K20) en el cual se dispone de una buena capacidad de lógica (1152 elementos lógicos) y memoria interna (6 bloques de 2048 bits de memoria cada uno).
Nos pareció interesante incluir en este análisis comparativo algunos diseños realizados con esta familia de chips porque, a diferencia con la familia FLEX8000 de la placa RIPP10, poseen RAM interna. Esto permite disponer de estructuras de almacenamiento de datos internamente sin la necesidad de consumir recursos de lógica y sin el cuello de botella que representa el bus de acceso a un chip de memoria externa.

2. ALGORITMOS IMPLEMENTADOS

No es intención de este trabajo describir en detalle cada uno de los algoritmos con los cuales se experimentó así como su implementación particular. Descripciones detalladas de los mismos y sus implementaciones pueden encontrarse en las referencias [10][11][12][13][14][15][16][17]. Trataremos entonces de extraer las principales características de los mismos para poder compararlos y de esa manera arribar a algunas conclusiones generales.

Hay que tener en cuenta que al hacer esto estamos perdiendo información sobre cada caso particular. Además la diversidad de los algoritmos realizados, los distintos enfoques para su implementación hardware y la participación de varios grupos de diseñadores hacen que sea difícil comparar los resultados de manera única y obtener conclusiones contundentes. Pero por otro lado esa misma diversidad y cantidad de trabajos realizados nos van a permitir obtener tendencias que a nuestro juicio son muy interesantes.

Desde el punto de vista del hardware utilizado ya hemos dicho que la mayoría de los algoritmos fueron implementados en la RIPP10 y que esta placa funciona conectada al bus ISA de un PC. Se presentan además dos casos de algoritmos realizados en chips de la familia FLEX10K realizados en una placa UP1.

En cuanto a las áreas de aplicación se han realizado algoritmos de redes neuronales artificiales, tratamiento de

imágenes y encriptado.

A continuación haremos una brevísima descripción de cada algoritmo realizado que luego resumiremos en una tabla.

Algoritmos de redes neuronales artificiales

Feed Forward o perceptrón multicapa

En este tipo de redes, cada nodo es básicamente un multiplicador - acumulador seguido de un bloque no lineal que implementa una función sigmoide. Las entradas de cada nodo se multiplican por un peso (determinado en la etapa de entrenamiento), luego se suman y al final pasan por el bloque sigmoide.

El parámetro crítico para estos diseños fue en todos los casos estudiados el área utilizada, debido a ello se optó por trabajar con 8 bits de datos. Es posible realizar aplicaciones con entradas de 8 bits y punto fijo, pero el problema que se plantea es que los paquetes de software para realizar el entrenamiento de redes neuronales generalmente no disponen de la posibilidad de ajustar el ancho de palabra de los datos y usualmente trabajan en punto flotante. Es necesario entonces verificar que los pesos obtenidos con un paquete estándar de entrenamiento llevados a 8 bits punto fijo pueden darnos un comportamiento aceptable de la red.
Otro problema numérico que se presenta es la saturación en una sinapsis o a lo largo de varias capas de la red. Aquí es necesario realizar un trabajo de escalado de los datos en cada capa, es posible realizar esto ajustando el bloque no lineal en cada caso y luego corrigiendo los pesos. Los multiplicadores tienen salida en 16 bits.

Se diseñaron dos arquitecturas para este tipo de redes y se analizó su mapeo en la RIPP 10. Debido a la restricción del tamaño se trabajó con una capa de 8 neuronas, 8 bits punto fijo y 8 entradas. Redes mayores se implementan multiplexando en el tiempo el uso de la circuitería de las 8 neuronas. Los datos y las funciones no lineales se cargan en las memorias RAM que se comparten entre dos neuronas. En una de las dos arquitecturas diseñadas la salida de cada neurona se calcula en un pipeline obteniendo un mejor throughput a costa de una respuesta entrada-salida más lenta. En la otra en cambio se buscó minimizar el tiempo de respuesta. Por esas características la primer arquitectura es más adecuada para aplicaciones en que se dispone un conjunto de vectores de entrada simultáneamente permitiendo mantener cargado el pipeline (por ejemplo reconocimiento de patrones), mientras que la segunda es más adecuada para aplicaciones de control en tiempo real.

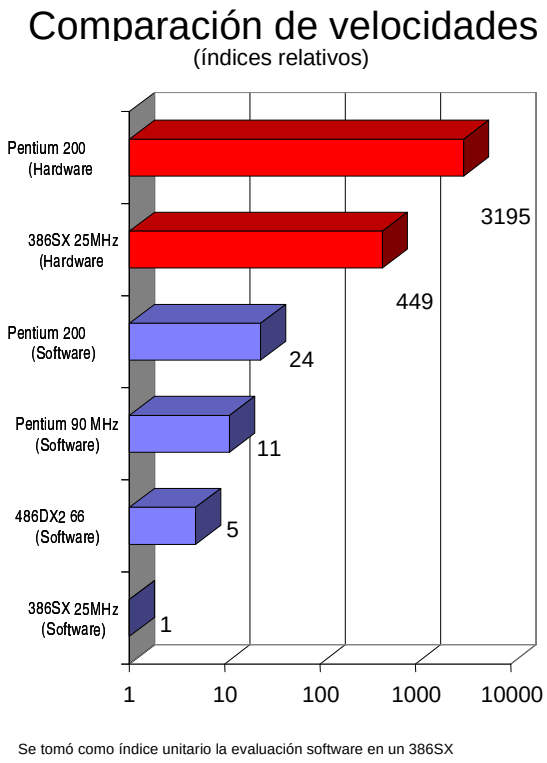

Figura 3 - Velocidades en una ALN entrenada

Redes Lógicas Adaptivas (Adaptive Logic Network ALN)

Una ALN puede describirse en general como un conjunto de árboles binarios balanceados con codificación y decodificación de entradas y salidas respectivamente y con capacidad de adaptación. Este tipo de árboles es capaz de realizar cualquier función booleana con la propiedad de ser robusto ante pequeñas variaciones de las entradas [5][6][7].

Evaluación

Una vez entrenada, una red ALN es un árbol binario donde cada nodo realiza una función booleana determinada (AND, OR, LEFT o RIGHT). Es decir que el circuito necesario para implementar un árbol ALN ya entrenado es una función combinatoria. Para problemas reales dicha función combinatoria posee una gran cantidad de nodos.
Se desarrolló una herramienta para traducir el resultado del entrenamiento obtenido con herramientas de diseño

de redes ALN a la expresión de la función combinatoria correspondiente al árbol ALN en un lenguaje de descripción hardware. Sintetizando esta descripción se obtiene el mapa binario a cargar en las FPGAs.

La Figura 3 muestra, para una red previamente entrenada, una comparación entre la velocidad en el cálculo utilizando el paquete ATREE con la función de evaluación estándar (designado como Software en la figura) y una función de evaluación modificada. Esta segunda función (resultados indicados como Hardware en la Figura 3) utiliza el circuito cargado en la placa RIPP10 para realizar la evaluación intercambiando datos y resultado a través del bus ISA.
Como puede verse en la gráfica, la aceleración de cálculo obtenida fue muy fuerte.

Aprendizaje
Durante el aprendizaje, cada nodo de un árbol ALN contiene dos contadores que almacenan el estado del nodo. La función combinatoria realizada por el nodo (AND, OR, LEFT o RIGHT) queda determinada por el valor de los contadores. Para cada patrón de entrenamiento el algoritmo de aprendizaje ajusta el valor de los contadores en función de la salida obtenida, la salida esperada, el estado del nodo y de una función de responsabilidad evaluada en el nodo padre.
Con la circuitería de aprendizaje, cada nodo ocupa algunas decenas de celdas lógicas. Se realizó un mapeo físico de cada nodo del árbol a un bloque de circuito. Dado el tamaño ocupado por cada nodo y la circuitería de control, en toda la placa RIPP10 solamente se lograron cargar árboles binarios de 7 niveles (128 hojas) lo que es totalmente insuficiente para cualquier aplicación de alguna utilidad. Igualmente se utilizó un árbol de 7 niveles para resolver una aplicación sencilla de reconocimiento de patrones. La red fue entrenada para reconocer letras mayúsculas representadas en una matriz de 5x5 pixels en blanco y negro. La red entrenada debe discernir si cierta matriz de 5x5 representa o no a determinada letra. Se entrenaron cinco redes para cada una de las vocales.
Se obtuvieron speed-ups de entre 25 y 80.

Redes neuronales competitivas
Este diseño fue realizado sobre placas UP1 con chips de la familia FLEX10K.
Las redes neuronales competitivas se caracterizan por tener una capa de nodos de entrada, que distribuyen las entradas, y una capa de nodos ocultos, conectados a los nodos de entrada, conocida como capa competitiva.
Cada conexión entre un nodo de entrada y uno oculto esta caracterizada por un valor numérico conocido como "peso".
La competición que se lleva a cabo en una red de este tipo consiste en determinar, para un vector de entrada dado (conocido como patrón de entrada), cual neurona posee el vector de pesos "más parecido" al de entrada. Esta similitud entre vectores comúnmente se caracteriza con alguna medida de distancia, es decir, que el nodo ganador será aquel que posea el vector de pesos que diste menos del presentado a la entrada.
Se trabajó con vectores binarios y se utilizó como medida de distancia la distancia Hamming (cantidad de bits en que difieren los vectores). No se realizó el procedimiento de aprendizaje. Solo se consideró la competición. Cabe destacar que en las redes competitivas, la salida viene dada por una identificación de la neurona ganadora.

El diseño resultante incluyó un bloque de control, bloques de evaluación de distancia Hamming, bloques de memoria para almacenar los pesos, un bloque de comparación y multiplexores varios. En el diseño se sacó partido de la memoria interna de los chips de la familia Flex10K para el almacenamiento de los pesos.

La simulación y prueba de la red competitiva se realizó para una red de 16 neuronas con un ancho de palabra de 8 bits.

Tratamiento de imágenes
Los algoritmos implementados fueron la FFT (transformada rápida de Fourier) de una y dos dimensiones, la convolución, la correlación, y la compresión byte-bit.

Un punto que llevó bastante tiempo en la etapa inicial de diseño de estos algoritmos fue la elección del formato de representación numérica. No existía experiencia anterior en hardware de punto flotante lo que inclinaba la balanza hacia el uso de punto fijo. Sin embargo para mantener los errores acotados trabajando en punto fijo se hace necesario o bien aumentar el ancho de palabra fuertemente con el consiguiente aumento en el tamaño de los multiplicadores; o bien detectar cuando se produce un desborde y hacer algún tipo de escalado en el resultado.

La elección final fue utilizar formato en punto flotante de 16 bits (8 mantisa, 7 exponente, 1 signo) para los algoritmos FFT y correlación. En la tabla puede compararse la cantidad de celdas ocupada por los circuitos básicos en punto fijo y punto flotante. Téngase en cuenta que el bloque básico del algoritmo FFT (butterfly) necesita 4 multiplicadores y 6 sumadores.

	Punto Fijo	**Punto Flotante**
Sumador	70	162
Multiplicador	330	246

Figura 4 - Celdas lógicas utilizadas por los bloques básicos en 16 bits

Convolución (filtro ventana)

Este algoritmo realiza la convolución de una imagen contra una ventana de 3x3 o 5x5 pixels. Es un método clásico de filtrado. Se utilizó en este caso representación en punto fijo. Con tamaños de ventana de 5x5 se obtuvo una mejora en velocidad en un factor de 4. El tiempo de cálculo está fuertemente determinado por la transferencia de datos a la placa a través del bus ISA, por lo que cuando se utilizan varios filtros de este tipo en cascada el tiempo de cálculo en hardware se mantiene prácticamente constante pudiéndose lograr en ese caso una mejora más sensible.

Compresión Byte-Bit

En este caso se buscan funciones para convertir imágenes blanco y negro (1 bit) entre una representación comprimida almacenando 8 pixels en cada byte y una representación en que se utiliza solamente un byte por pixel. Si bien la naturaleza de las operaciones (rotaciones y operaciones lógicas bit a bit) podrían hacer suponer una realización eficiente en hardware, esta ganancia no compensa el tiempo perdido en la transferencia de datos hacia y desde la placa a través del bus ISA. En otras palabras, es un algoritmo que requiere mucho pasaje de datos y poco cálculo. Los tiempos de ejecución resultantes fueron peores que en software.

Fast Fourier Transform (FFT), FFT Inversa (IFFT) y Correlación

Debido a limitaciones en la capacidad de memoria de la placa (que no permite almacenar los resultados intermedios para una imagen completa), se decidió tomar como algoritmo base el de la FFT en una dimensión (FFT-1D). La FFT en dos dimensiones se calculó en base a una serie de FFT-1D, obligando a transferir varias veces a la placa el valor de cada pixel de la imagen de entrada. Por otro lado, la correlación de dos matrices se realizó tomando primero la FFT-2D de cada matriz, calculando los productos punto-a-punto entre las transformadas y finalmente tomando la FFT-2D inversa.

Se utilizó una representación en punto flotante en 16 bits con 8 bits de mantisa y 7 bits de exponente. Para implementar los sumadores y multiplicadores se utilizaron bibliotecas de dominio público.

Los algoritmos se probaron con imágenes de tamaños entre 32x32 y 512x512. No se obtuvieron buenos resultados en la velocidad de ejecución, que resultó comparable a la obtenida con implementaciones "solo software". Es de esperar mejores resultados si se cuenta en la placa con memoria suficiente para almacenar las matrices completas, bajando fuertemente entonces la necesidad de transferencia de información entre la placa y el host.

Encriptado

Bloque DES

Este algoritmo fue implementado sobre el chip de la familia FLEX10K en la placa UP1. Se realizó en hardware el algoritmo de encriptación DES (Data Encryption Standard) descripto en el estándar FIPS PUB 46. Dado que se estaba acotado al tamaño del chip se debió llegar a un compromiso entre área ocupada contra velocidad de cálculo. La solución construida ocupa 95% del chip y evalúa cada dato de salida en 16 períodos de reloj. La frecuencia máxima de reloj es de 8Mhz.

3. CONCLUSIONES

Como puede apreciarse de la tabla resumen de resultados para cada algoritmo el abanico es muy amplio, tanto en los tipos de aplicaciones realizadas como en sus resultados. Tratando de sacar conclusiones generales podemos decir que el principal resultado buscado que es aumentar la velocidad de un cierto algoritmo en un número significativo (aumento mayor a 10 veces por ejemplo) no es fácil de lograr con la plataforma disponible.
Si bien en el caso de la evaluación de una red ALN se tiene un resultado espectacular (más de 100 veces), para otros algoritmos la mejora es mediocre (entre el doble a 10 veces), e incluso empeora en un caso (es bastante razonable que esto ocurra en casos en los cuales el cálculo es poco y sin embargo hay mucha entrada salida de datos a través del bus ISA). Las causas son varias y dependen en general del tipo de algoritmo, pero son las limitaciones ya mencionadas: cantidad de celdas lógicas disponibles, cantidad de RAM disponible, ancho de banda de entrada salida de datos.

Tabla 1 Resumen de características de cada algoritmo

Algoritmo	**Hardware**	**Tipo de dato**	**Operaciones**
RRNN: FF	RIPP 10	Punto fijo 8 bits	Multiplicador, Sumador
RRNN: ALN evaluación	RIPP 10	Bits	Operaciones binarias: AND, OR
RRNN: ALN aprendizaje	RIPP 10	Bits	Operaciones binarias, contadores
RRNN: competitiva	UP1	8 bits	Distancia Hamming, comparador
Imágenes: FFT-2D, IFFT-2D y Correlación	RIPP 10	Punto flotante 16 bits (8 mantisa, 7 exp.)	Multiplicador, sumador, tabla seno coseno
Imágenes: convolución (con ventanas 3x3 y 5x5)	RIPP 10	Punto fijo 8 bits	Multiplicador, Sumador, direccionamiento de memoria
Imágenes: compresión bit.-byte	RIPP 10	Bits	Reordenamiento de datos
Encriptado: DES	UP1	64 bits	Permutaciones, LUT

Tabla 2 Resumen de resultados de cada algoritmo

Algoritmo	**Velocidad**	**Acotado por Tamaño**	**E/S datos**	**Errores numéricos**	**Comentarios**
RRNN: FF	Sin info comparativa	SI, velocidad vs. Área	limita velocidad	Si, difíciles de manejar	No se hicieron aplicaciones reales
RRNN: ALN evaluación	Muy alta (> x100)	NO	limita velocidad	NO	Control del péndulo invertido
RRNN: ALN aprendizaje	Alta (> x25)	SI, muy acotado	limita velocidad	NO	No se hicieron aplicaciones reales
RRNN: competitiva	Sin info comparativa	NO		NO	Demostrativo. Usa RAM interna
Imágenes: FFT-2D, IFFT-2D y Correlación	Media (x2 a x10)	SI, velocidad vs. Área	limita velocidad	Aceptable, acotado	Aplicable, interfaz software
Imágenes: convolución (con ventanas 3x3 y 5x5)	Media (x3, x4)	Problemas de memoria		Aceptable, acotado	Aplicable, interfaz software
Imágenes: compresión bit.-byte	Muy mala		Graves problemas	NO	Mucho pasaje de datos, poco cálculo
Encriptado: DES	500Kdatos/seg	Da problemas		NO	Demostrativo Usa RAM interna

Los problemas que se presentan en general están interrelacionados, es así que uno de los problemas mayores que es el área o la cantidad de celdas lógicas puede mejorarse mucho si los chips tienen RAM interna, o si la reprogramación es rápida. La velocidad de programación es un parámetro que puede compensar la falta de celdas lógicas aunque esto no fue probado debido a que la placa RIPP no se puede programar en forma individual para cada chip. Además, dado que su memoria es escasa no pareció una alternativa razonable programar en varias partes un algoritmo e ir guardando resultados parciales. Si bien cada FLEX8000 de la RIPP puede reconfigurarse en menos de 100ms, reconfigurar los 8 chips desde la PC lleva algunos segundos, dependiendo del PC al cual

esté conectado.
El ancho de banda de la comunicación de la placa reconfigurable con el host es un parámetro muy importante que influye no solo en el pasaje de datos y resultados sino también en la velocidad de reprogramación de los chips, dado que la misma se realiza por el mismo bus.

En cuanto a las dificultades de diseño de los algoritmos en sí, en general se encontró que el diseño de los bloques de datos fue bastante directo, mientras que los bloques de control, el acceso a chips de memoria externa, el manejo de direcciones y el pasaje de datos entre chips requirieron un mayor esfuerzo y cuidado en el diseño.

4. TRABAJOS FUTUROS

Actualmente el IIE cuenta con una placa ARCPCI de Altera. Esta placa posee tres chips de la familia FLEX10K, uno de ellos se encarga de realizar la interfaz PCI y los otros dos están disponibles para aplicaciones de usuario. Además tiene varios SIMMS de memoria RAM de gran tamaño y buena velocidad de acceso. Esto nos permitirá resolver los problemas de ancho de banda de datos así como almacenamiento temporario de información.
La placa permite la reprogramación de cada chip de usuario por separado y en un tiempo muy breve con lo que se hace muy tentador incursionar en la reconfiguración parcial de la placa y mientras un chip sigue trabajando programar un nuevo algoritmo en el otro.

Otra alternativa en la cual nos interesa trabajar es en la utilización de herramientas de diseño de alto nivel. Existe ya un nuevo proyecto con financiamiento que permitirá ampliar las plataformas existentes, y se piensa continuar con algoritmos de redes neuronales y tratamiento de imágenes.

Un punto central de la estrategia de este grupo es poder ser el nexo entre aplicaciones con intenso procesamiento de señales originadas en grupos de especialistas de otras áreas y la implementación de las mismas sobre hardware flexible.

RECONOCIMIENTOS

Algunas de las actividades descriptas en este trabajo fueron financiadas por el Consejo de Investigación Ciencia y Tecnología de Uruguay (CONICYT) a través del proyecto BID-CONICYT Nº 355 "Procesador Neuronal Basado en Lógica Programable".

Las placas RIPP10 y ARCPCI fueron donadas por el "Programmable Hardware Development Program" de la empresa ALTERA.

Algunos de los algoritmos descriptos fueron realizados bajo la dirección de los autores como trabajos curriculares por estudiantes. En los algoritmos de tratamiento de imágenes trabajaron Jorge Myszne, Juan Andrés Míguez y Germán Calvo en su proyecto de graduación. En el diseño de la red neuronal competitiva trabajaron André Fonseca, Sebastián Gava y Javier Román, mientras que el diseño del encriptador DES fue realizado por Alejandro Pareja y Fernando Reyes. Estos dos últimos diseños se realizaron como proyecto final del curso "Diseño digital utilizando dispositivos lógicos programables"

REFERENCIAS

1. S. Guccione, *Programming Fine-grained Reconfigurable Architectures*, Ph. D. Thesis, University of Texas at Austin, 1995.
2. J. Cloutier, E. Cosatto, S. Pigeon, F. R. Boyer and P. Y. Simard , “VIP: An FPGA-based Processor for Image Processing and Neural Networks”, MicroNeuro ‘96, Lausanne, Switzerland, 1996.
3. A. G. Supynuk and W. W. Armstrong, “Adaptive Logic Networks and Robot Control”, Proc. Vision Interface Conference '92, also called AI/VI/GI '92, pp. 181 - 186, Vancouver B. C., 1992.
4. W. Armstrong and J. Gecsei, "Architecture of a Tree-based Image Processor", *12th Asilomar Conf. on*

Circuits, Systems and Computers, pp. 345-349, Pacific Grove, 1978.

5. W. W. Armstrong and G. Godbout, "Properties of Binary Trees of Flexible Elements Useful in Pattern Recognition", *IEEE 1975 International Conf. on Cybernetics and Society*, pp. 447-449, San Francisco, 1975.
6. W. W. Armstrong, A. Dwelly, J.Liang, D. Lin and Scott Reynolds, “Learning and Generalization in Adaptive Logic Networks”, *Artificial Neural Networks, Proc. of the 1991 Int. Conf. on Artificial Neural Networks (ICANN-91)*, T. Kohonen, K. Makisara, O. Simula, J. Kangas, eds, pp.1173-1176, North Holland,.Espoo, Finland, 1991.
7. G. V. Bochmann and W. Armstrong, “Properties of Boolean Functions with a Tree Decomposition”, BIT, Vol. 13, pp. 1-13, 1974.
8. W. Armstrong and J. Gecsei, "Adaptation Algorithms for Binary Tree Networks", IEEE Trans. on Systems, Man and Cybernetics, Vol. 9, pp. 276-285, 1979.
9. ALTERA RIPP10 Design Page, Brigham Young University, Electrical Engineering's Configurable Computing Laboratory, http://splish.ee.byu.edu/tutorials/altera/altera.html
10. "Síntesis Hardware de Redes ALN para Aplicaciones en Control", J. P. Oliver, A. Fonseca de Oliveira, J. Pérez, R. Canetti, VIII RPIC99, Mar del Plata, Argentina, 23 al 25 de setiembre de 1999.
11. "Implementation of Adaptive Logic Networks on an FPGA board", Juan P. Oliver, André Fonseca de Oliveira, Julio Pérez Acle, Roberto J. de la Vega, Rafael Canetti, in Configurable Computing: Technology and Applications, John Schewel, Editor, Proceedeings of SPIE Vol. 3526, page numbers 264-273 (1998).
12. Pérez, Julio. Proyecto Coprocesador Neuronal. Reporte interno No. 101. "Arquitecturas de perceptrón multicapa: Arquitectura pipeline". IIE. 1998
13. Eirea, Gabriel. Proyecto Coprocesador Neuronal. Reporte interno No. 102. "Ideas para el cálculo digital de la función sigmoide". IIE. 1998
14. Canetti, Rafael; Pérez, Julio. Proyecto Coprocesador Neuronal. Reporte interno No. 104. "Propuesta de una red multicapa elemental". IIE. 1998
15. Myszne, Jorge; Calvo, Germán; Miguez, Juan Andrés. "Implementación de algoritmos de tratamiento de imágenes en lógica reconfigurable". Proyecto de graduación, (informe disponible en http://www.iie.edu.uy/~dsp3/). IIE. 1998
16. Fonseca, André; Román, Javier y Gava, Sebastián. " Implementación en lógica programable de una red neuronal competitiva". (http://www.iie.edu.uy/ense/asign/dlp/proyectos/1998/neural/). IIE. 1998.
17. Pareja, Alejandro y Reyes, Fernando. " Implementación de un bloque DES". (http://www.iie.edu.uy/ense/asign/dlp/proyectos/1999/des/index.htm). IIE. 1999.