PROCESADOR DE IMÁGENES CNN SIMPLICIAL: IMPLEMENTACIÓN EN FPGA

M. Di Federico, P.M. Julián, P. Mandolesi, P.D. Pareja Obregón

Instituto de Investigaciones en Ingeniería Eléctrica - IIIE
(UNS-CONICET)
Departamento de Ingeniería Eléctrica y de Computadoras
Universidad Nacional del Sur
Avda. Alem 1253 (8000) Bahía Blanca
mdife@uns.edu.ar

Resumen—En este artículo se muestra el diseño de una estructura CNN Simplicial para el procesamiento de imágenes. Esta estructura fue descripta en VHDL, sintetizada en una FPGA.

1. INTRODUCCIÓN

El procesamiento de imágenes requiere administrar una gran cantidad de información, por esta razón el análisis y procesamiento de imágenes realizado por sistemas de cómputo requiere una gran cantidad de tiempo de procesamiento. Esto presenta una desventaja en sistemas de tiempo real, donde una interpretación inmediata es necesaria, como por ejemplo en inspecciones en la industria, sistemas de navegación autónoma, etc [12]. El enfoque tradicional de procesamiento de datos secuencial, comúnmente utilizado en computadores y microprocesadores, presenta una limitación, dado que es necesario recolectar toda la información en un mismo lugar para proceder a su procesamiento mediante una única unidad de procesamiento. El transporte de la información torna inevitablemente lento el procesamiento. Es aquí que surge el paradigma del cálculo a nivel local, donde cada unidad realiza un cierto procesamiento, utilizando la información propia e información de las unidades más cercanas solamente. Esto ha permitido desarrollar sistemas que alcanzan velocidades de operación muy elevadas [9][13].

2. CNN Y SIMPLICIAL-CNN

La llamada Red Celular No-lineal (o CNN por su sigla en inglés), introducida originalmente por Chua y Yang [1], es una arquitectura que permite realizar procesamiento en paralelo de información. En la CNN, cada celda desarrolla una misma ecuación diferencial no-lineal dinámica, y el procesamiento se realiza por la acción coordinada de todas ellas [2]. De esta manera, es posible implementar tareas de cálculo complejas. Las CNN bidimensionales constituyen una atractiva arquitectura, sobre la que se pueden implementar varios procesos para extraer información o realizar una interpretación de la imagen. Esta arquitectura se puede desarrollar en formato analógico [9], o en formato digital, lo cual resulta atractivo por su escalabilidad. Se puede definir un arreglo de procesadores no lineales localmente interconectados (celdas) que operan en el dominio digital, dando lugar a lo que ha dado en llamarse Redes Neuronales Celulares de Tiempo Discreto (CNN-DT) [7]. Las CNN-DT presentan importantes ventajas como su inherente robustez y fácil control de la estructura, que es definido por una función externa y es la misma para todas las celdas. Se han realizado varias aplicaciones al filtrado de imágenes, segmentación [6] [10] [11], detección de: bordes, puntos aislados, conectividad; generación de sombra, modificación de histograma, estimaciones [5], reconocimiento de patrones [3], etc.

Las CNN son una herramienta poderosa para visión robótica y el procesamiento de imágenes y de video. Cada celda posee un valor de estado, teniendo el sistema completo un estado definido. El cálculo del estado siguiente del sistema está determinado por una ecuación que cada celda debe resolver, lo que implica que deben tener el poder de realizar los cálculos necesarios. En este caso el sistema dinámico posee un estado local, entradas y salidas. Las ecuaciones de estado del CNN Simplicial (S-CNN) utilizan una función lineal a tramos para calcular el valor del próximo estado. Este algoritmo fue originalmente propuesto en [4].

3. DISEÑO IMAGER

Cada celda cuenta con un valor de estado, entradas y salidas. Las entradas de cada celda son los valores de intensidad de luz (o valor del píxel de la imagen) de la celda respectiva y de los vecinos pertenecientes a la esfera de influencia. Un píxel puede tener cuatro u ocho vecinos dependiendo de la definición de de conectividad. La Fig. 1 muestra un píxel conectado con sus 8 vecinos.

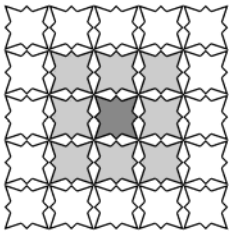


Figura 1: Una celda y su esfera de influencia.

El valor del estado actual esta se almaceno en un registro, en el cual se puede encontrar valores previos de entradas, valores ya procesados o valores arbitrarios. La entrada de cada celda es la concatenación de la información que recibe de los vecinos que conforman la esfera de influencia. La salida es la información que cada celda envía a sus vecinos. Basado en todas estas variables, la ecuación de estado, produce una

evolución en tiempo de la celda, usando una función lineal a tramos. El valor del estado de la celda esta codificado en una palabra de 8 bits y almacenado en un registro. La entrada es una palabra digital de 5 bits. El valor del siguiente estado es calculado con el algoritmo descrito en [8], el cual es explicado brevemente a continuación. Un grafico de la celda que realiza este algoritmo es mostrado en la Fig. 4.

Las señales de entrada y el valor del estado son comparadas con una rampa digital durante un ciclo llamado *Ciclo de Programa*. La señal resultante de la comparación, contiene información de los valores de los registro codificada en tiempo. Estas señales y las equivalentes pertenecientes a los vecinos, son concatenadas en un vector de 5 bits *(gr, fr)*. Este vector es utilizado para seleccionar una dirección de memoria y obtener un valor. El mismo se utiliza para realizar la función lógica a implementar por la S-CNN, *G(gr)=Gdr* y *F(fr)=Fdr*. Como la memoria no se encuentra en la celda, para cada paso del *Ciclo de Programa*, una rampa digital con los valores de la memoria es distribuida a todas las celdas a través del bus. Este ciclo es llamado *Ciclo de Evaluación*. Para cada paso del valor del *Ciclo de Programa*, el valor de las funciones de *F(fr)* y de *G(gr)* son obtenidas y son operadas por una función lógica. El valor obtenido es integrado y al final del *Ciclo de Programa* el resultado de la integración es el valor del nuevo estado.

Cada celda del arreglo debe tener lo necesario para:

- Almacenar el estado y el valor de entrada
- Comparar el estado con la rampa digital
- Obtener la señal PWM para enviar a sus vecinos
- Obtener el valor de la función
- Realizar la operación entre funciones
- Integrar el valor obtenido de la función

4. ARQUITECTURA

La estructura contiene un arreglo de 16x16 celdas interconectadas entre sí. En este caso la esfera de influencia es de 4 vecinos y configurable de Vertical-Horizontal a Diagonal. Cada celda se puede comunicar con las celdas Norte-Sur-Este-Oeste (Horizontales y Verticales) o las Noreste-Noroeste-Sureste-Suroeste (Diagonales). Los tipos de conexión con las celdas vecinas forman vecindades llamadas $S_{(+)}$ y $S_{(\times)}$ respectivamente, tal como lo muestra la Fig. 2 .

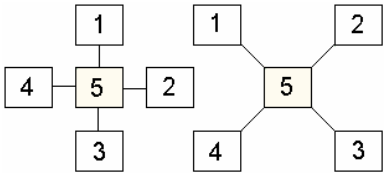


Figura 2: Configuración de vecindad $S_{(+)}$ y $S_{(\times)}$.

Esto significa que se dispone de un arreglo de celdas interconectadas (con conexión variable) que realizan cálculo computacional en paralelo. El procesamiento esta basado en la operación lógica de dos funciones aplicadas a dos registros de la celda y sus vecinas. El cálculo del próximo estado se realiza con una función PWL Simplicial. El valor final de la función se obtiene integrando los vértices. Esta estructura permite realizar una gran variedad de procesamiento con imágenes, desde traslaciones y filtros, hasta reconocimiento de patrones simples. Combinando varias funciones se pueden realizar tareas mas complejas como esqueletización, binarización, apertura, cierre, eliminar puntos aislados, etc. Una lista de algunos procesos implementado se muestra en la Tabla 1.

Tabla 1: Algunos Procesos que realiza la arquitectura

Filtros	Traslación	Reconocimiento:	
Copiar	Arriba	Bordes	Puntos Aislados
Negar	Abajo	Esquinas	Fin de Líneas
Erosión	Izquierda	Bordes Verticales	Curvas
Mediana	Derecha	Bordes Horizontales	Bifurcaciones
Dilatación	Arriba-Derecha	Bordes Diagonales	Trifurcaciones
Apertura	Arriba-Izquierda	Lineas Verticales	
Cierre	Abajo-Derecha	Lineas Horizontales	
Binarizar	Abajo-Izquierda	Lineas Diagonales	
Esqueletizar			

La Fig. 3 muestra el diagrama en bloques del sistema, donde se pueden ver todos los bloques que lo componen. Los valores de las funciones F y G se almacenan en un registro serie. Las señales para controlar el circuito son generadas por un microcontrolador PicoBlaze. La comunicación con la PC se produce por medio de una UART.

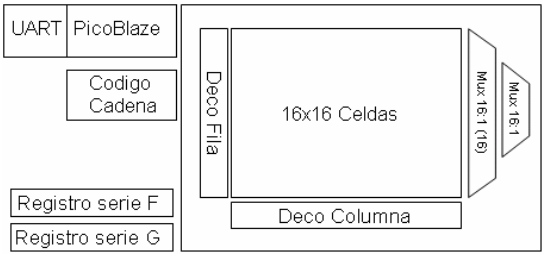


Figura 3: Bloques de la arquitectura

Como se puede ver en la Fig. 3, junto con el arreglo de celdas se sintetizó un bloque extra pare obtener el código cadena de una imagen. Este bloque está explicado en la sección 4.2

4.1 Diseño de Celda

Cada celda contiene 3 registros de 8 bits, llamados X, U y W. Los valores de estos registros pueden ser cargados externamente. Las funciones F y G pueden ser aplicadas a cualquier registro y el resultado puede ser operado por la función OR, XOR y AND. La integral es realizada por un contador de 8 bits. La celda en total tiene 34 FlipFlops, 24 para los 3 registros. 8 para el contador 2 para guardar el valor de las señales Fpwm y Gpwm, y 2 para almacenar los valores de *Fdr* y *Gdr*. Un esquemático de la celda se muestra en la Fig. 4.

Para obtener el valor del vértice a integrar se debe comparar el valor del registro con la el valor correspondiente del *Ciclo de Programa*, para esto se utiliza un solo comparador lo que significa que se necesitan 2 ciclos de reloj (uno para cada registro) para realizar las comparaciones. El valor resultante de la comparación, se guarda en el registro Fpwm y Gpwm. El valor de los vértices *Fdr* y *Gdr* se

obtienen en el *Ciclo de Evaluación*, cada uno de los dos comparadores se encarga de comparar el valor de la entrada de la celda con el del barrido de la rampa y así latchear el valor correspondiente en los Registros F y G.

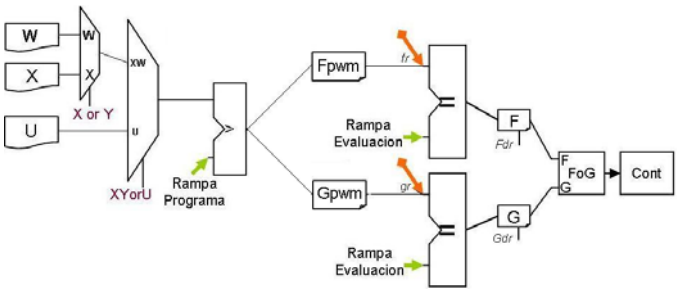


Figura 4: Esquemático Celda

El *Ciclo de Evaluación* tiene una longitud de 32 pasos (desde 00000 hasta 11111), pero si por cada paso del *Ciclo de Evaluación* se envían 2 valores de función, serán necesarios menos ciclos (16) para obtener todos los datos. Para esto se necesita un multiplexor que seleccione el valor del vértice correcto. Con esta configuración, el comparador usado para verificar la dirección tendrá un bit menos. Un análisis aproximado de áreas que ocuparía la celda enviando 1, 2, 4, 8, 16 o 32 valores a la vez es mostrado en la tabla 2.

En esta prueba de concepto se ha seleccionado enviar 2 señales para testear su performance y en un próximo paso se agregarán más señales.

Tabla 2:Relación cantidad de señales área y velocidad.

Señales	Transistores	%	Ciclos	Velocidad
1	1078	100	8704	1
2	1070	99.3	4608	1.8
4	1070	99.3	2560	3.4
8	1086	101	1536	5.6
16	1142	106	1024	8.5
32	1214	113	256	34

El bloque FoG realiza función lógica entre F y G, esta operación es seleccionable entre AND, OR y XOR.

4.2 Bloque Generador Código Cadena

El código cadena es una forma más eficiente de representar una lista de puntos mas allá de $[(x_1, y_1), (x_2, y_2), ...]$. Este código guarda los movimientos que se realizan al recorrer la imagen. Para realizar este proceso el bloque necesita la dirección del píxel donde se comenzará (se puede obtener con el proceso "Fin de línea") y la dirección de comienzo de recorrido (Abajo, Arriba, Izquierda o Derecha).

Antes de comenzar a realizar el código cadena hay que almacenar en cada píxel la información de existencia o no de los vecinos de cada píxel. Los 5 bits menos significativos del registro se utilizan para codificar esta información. La información del píxel superior se almacena en el bit menos significativo. La Fig. 5 muestra donde se guarda la información de cada uno de los vecinos. Por Ejemplo, si el registro tiene un valor de 0101, los vecinos de la Izquierda y de la derecha forman parte de la imagen.

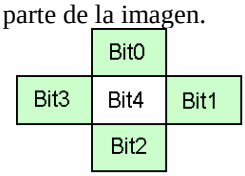
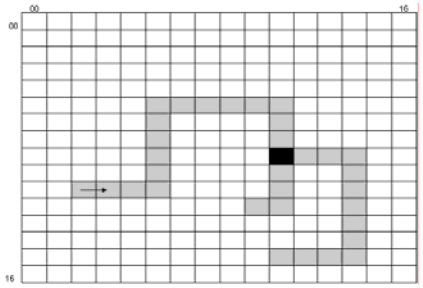


Figura 5: Información de la presencia de vecinos.

Si la imagen tiene una bifurcación, la dirección preferencial es la actual. Si se llega a una T, siempre la prioridad es avanzar aumentando filas o aumentando columnas. La Fig. 6 muestra una imagen, y su código cadena. El píxel marcado con negro es una bifurcación. El circuito, al pasar por un píxel con una bifurcación, envía una señal indicando que tuvo que tomar una decisión. El Resultado final del proceso es un registro indicando la cantidad total de píxeles recorridos y los movimientos realizados se almacenan en un registro serie.



$DDDA_rA_rA_rA_rA_rDDDDDA_bA_bA_bA_bA_bA_bA_bI_z$

Figura 6: Trazo con bifurcación y su código cadena.

El bloque cuenta con una unidad de decisión, un generador de direcciones, una unidad de control, un registro serie, y un acumulador. Para seleccionar el valor del próximo píxel a visitar se utiliza la información que viene del píxel indicando cuales de sus vecinos están activos y la dirección actual del movimiento. Se utilizan 2 bits para guardar la dirección del recorrido, las cuales pueden ser "00"=baja, "01"=sube, "10"=izquierda y "11"=derecha.

5. IMPLEMENTACIÓN E INTERFAZ

El circuito se implementó en una FPGA Spartan 3-1500, que es parte de una placa de desarrollo MEMEC. Se utilizó comunicación via USB con la PC con una interfaz desarrollada en MATLAB. Para realizar el testeo del circuito, se realizó una interfaz por computadora. Esta interfaz permite cargar imágenes, visualizar la información de todos los registros de las celdas, cargar los valores de las funciones F y G, realizar cualquier proceso a la imagen y crear animaciones paso a paso de la evolución del valor de la celda.

El sistema fue sintetizado y testeado con éxito. Algunas de las imágenes procesadas se muestran a continuación.

La Fig. 7 muestra la imagen original cargada en el arreglo, y el negativo de la imagen calculado usando el proceso "Negar".

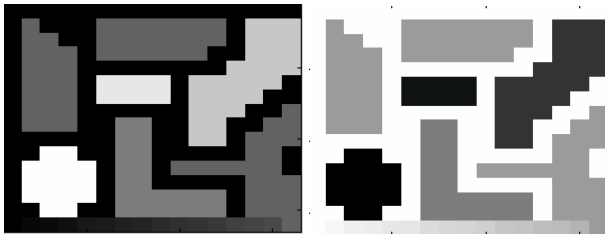


Figura 7: Imagen original y negativo.

La Fig. 8 muestra la binarización de la imagen utilizando distintos niveles de umbral, pudiendo realizar distintas segmentaciones básicas de la imagen. La Fig. 9 muestra la detección de bordes internos y externos. La primera se logra con el proceso "Bordes" la detección de bordes externos se logra dilatando la imagen y restando la original. La Fig. 10 muestra la detección de bordes direccionales, en este caso se muestran los bordes verticales, se puede elegir que el borde sea cualquiera de las 8 direcciones posibles si el borde sea Oeste-Este o Este-Oeste. La Fig 11. Muestra la esqueletización de la imagen junto con los puntos finales de las líneas.

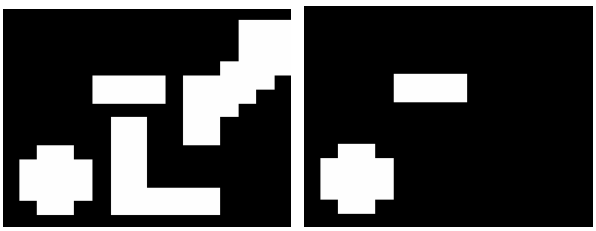


Figura 8: Binarización con varios umbrales.

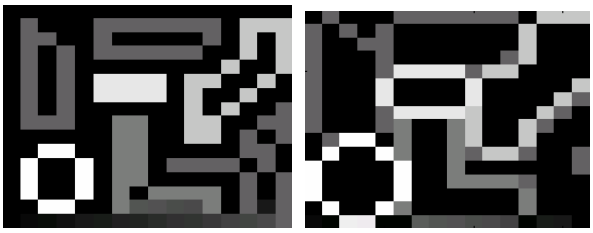


Figura 9: Detección de bordes, internos y externos.

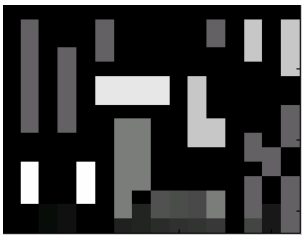


Figura 10: Detección de Bordes verticales.

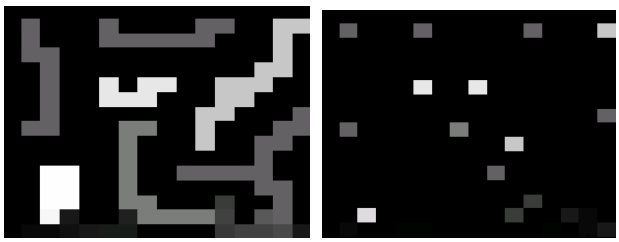


Figura 12: Esqueletización y detección de final de línea.

6. REFERENCIAS

[1] L.O. Chua, L. Yang, "Cellular neural networks: Theory", IEEE Trans. Circuits and Systems, Abril 1988, vol. 35, pp. 1257-1272

[2] L. O. Chua, "*CNN: A vision of complexity"*, Int. Journal of Bif. and Chaos, Oct. 1997, vol. 7, pp. 2219-2425

[3] Qun Gao, P. forster,K. R. Mobus, G. S. Moschytz, "*Fingerprint recognition using CNNs: fingerprint preprocessing"*, IEEE Int. Symposium on Circuits and Systems, vol 3, 6-9 May 2001 pp. 433 - 436

[4] P. Julián, R. Dogaru, Leon O. Chua, "*A piecewise-linear simplicial coupling cell for CNN gray-level image processing*", IEEE Trans. on Circuits and Systems, July 2002, pp. 904-913.

[5] G. Grassi, L- A. Grieco, "*Object-oriented image analysis using the CNN universal machine: new analogic CNN algorithms for motion compensation, image synthesis, and consistency observation* ", IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, Abril 2003, vol 50, No 4, pp. 488 - 499

[6] P. Arena, A. Basile, M. Bucolo, L. Fortuna, "*An object oriented segmentation on analog CNN chip"*, IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, Julio 2003, vol 50, No 7, pp. 837 - 846

[7] V. M. Brea, D. L. Vilarino, A. Paasio, D. Cabello, "*Design of the processing core of a mixed-signal CMOS DTCNN chip for pixel-level snakes"*, IEEE Trans. on Circuits and Systems I: Regular Papers, May 2004, vol 51, No 5, pp. 997 - 1013

[8] P. S. Mandolesi, P. Julian, A. G. Andreou, "*A scalable and programmable simplicial CNN digital pixel processor architecture*", IEEE Trans. on Circuits and Systems I: Regular Papers, Mayo 2004, vol 51, No 5, pp. 988- 996. ISSN: 1057-7122.

[9] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, S.E. Meana, *" ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs"*, IEEE Trans. on Circuits and Systems I: Regular Papers, May 2004, vol 51, No 1, pp 851 - 863

[10] G. Grassi, E. Di Sciascio, L.A. Grieco, P. Vecchio, "*New object-oriented segmentation algorithm based on the CNN paradigm***",** Transaction on Circuits and Systems II: Express Briefs, IEEE, April 2006, vol 53, Issue 4, pp 259 – 263.

[11] R. Perfetti, E. Ricci, D. Casali, G. Costantini, " *Cellular Neural Networks With Virtual Template Expansion for Retinal Vessel Segmentation"*, IEEE Transactions on Circuits and Systems II: Express Briefs, Feb. 2007, vol 54, No 2, pp. 141 – 145

[12] G. Xu, Y. Yin, L. Yin, Y. Hao, Z. Wang, *"Visual Information Processing Using Cellular Neural Networks for Mobile Robot"*, Proceedings of 2007 IEEE International Conference on Grey Systems and Intelligent Services, November 18-20, 2007, Nanjing, China

[13] J. Dubois, D. Ginhac, M. Paindavoine, B. Heyrman, "*A 10 000 fps CMOS Sensor With Massively Parallel Image Processing"*, IEEE Journal of Solid-State Circuits, Mar 2008, vol 43, No 3, pp. 706 – 717