

Implementación de una estructura neuronal celular en hardware reconfigurable

Luis F. Muñoz M.

Universidad de Guadalajara, Centro Universitario de Ciencias Exactas e Ingenierías,
Blvd. Marcelino García Barragán #1421, C.P. 44430, Guadalajara, Jalisco, México

Teléfono: +52 (33) 1378 5900

luisfmm@yahoo.com.mx

Juan José Raygoza P.

Universidad de Guadalajara, Centro Universitario de Ciencias Exactas e Ingenierías,
Blvd. Marcelino García Barragán #1421, C.P. 44430, Guadalajara, Jalisco, México

Teléfono: +52 (33) 1378 5900

juan.raygoza@cupei.udg.mx

J. Roberto R. Barón

Universidad de Guadalajara, Centro Universitario de Ciencias Exactas e Ingenierías,
Blvd. Marcelino García Barragán #1421, C.P. 44430, Guadalajara, Jalisco, México

Teléfono: +52 (33) 1378 5900

jose.reyes@red.cupei.udg.mx

Susana Ortega Cisneros

Centro de Investigación y de Estudios Avanzados del I.P.N, CINVESTAV, Unidad Guadalajara
Av. del Bosque 1145, colonia el Bajío, C.P. 45019, Zapopan, Jalisco, México

susana.ortega@gdl.cinvestav.mx

Resumen

El siguiente artículo presenta el diseño e implementación de una red neuronal celular (CNN) desarrollado en dispositivos reconfigurables FPGA con aplicaciones para procesamiento digital de imágenes en escala de grises a ocho bits y dimensión M x N.

La CNN es capaz de hacer el procesamiento mediante una exploración sobre todo el patrón de entrada, desplazando por bloques la matriz para procesar la imagen. La red propuesta consiste en una matriz de 24 elementos de procesamiento, constituidos por unidades aritméticas independientes que determinan su valor, colocados en una malla rectangular de cuatro filas por seis columnas. Se describe un conjunto de enlaces programables que permiten modificar la salida mediante el tipo de conectividad definido. El diseño fue desarrollado utilizando lenguaje de descripción de hardware VHDL. La implementación se realizó en un dispositivo FPGA Xilinx® de la familia Virtex-6. Se muestra la simulación, así como los resultados en área de ocupación y latencia.

Palabra(s) Clave(s): CNN, FPGA, procesamiento digital de imágenes.

1. Introducción

Las redes neuronales celulares (CNN del inglés *Cellular Neural Network*) son un paradigma de procesamiento en paralelo que permite la comunicación entre unidades vecinas, fueron desarrolladas en la universidad de Berkeley por Leon O. Chua y Yang en 1988 [1]. Este concepto involucra características de autómatas celulares y redes neuronales artificiales, de los primeros toma su organización y estructura a manera de rejillas regulares con interacción entre elementos locales, mientras que de las segundas su capacidad de procesamiento asíncrono paralelo e interacción global de sus elementos en la red [2].

Su importancia radica en el desempeño computacional, principalmente su estructura de cálculo concurrente, la cual presenta una mejoría para la solución a problemas de velocidad encontrados en métodos de computo convencional por software [3], además, un campo de investigación significativo es el procesamiento digital de imágenes, fundamental en la inteligencia artificial, robótica, visión artificial y control, áreas donde se requiere procesar una gran cantidad de datos en alta velocidad [4].

Este trabajo plantea el diseño, desarrollo e implementación sobre hardware reconfigurable de una estructura basada en CNN, con la capacidad de procesar imágenes mediante diferentes tipos de conectividad.

1.1. Arquitectura estándar

La arquitectura de la red neuronal celular permite únicamente la interconexión entre células “vecinas”, generando un arreglo el cual puede tomar diferentes formas geométricas, siempre y cuando se sigan respetando las reglas de interconexión de sus elementos, lo que cual conlleva a presentar un gran número de arquitecturas y conectividades diferentes [5,6]. En el espacio bidimensional, las células de la red pueden ser mapeadas sobre diferentes esquemas físicos, como por ejemplo, los mostrados en la Fig. 1. Con independencia de la estructura empleada, la CNN puede estar formada por células idénticas entre sí o por un conjunto reducido de células diferentes, como ocurre en el caso de la redes NUP-CNN [7]. (*Non Uniform Processor CNN*), Fig. 1d y 1f.

La vecindad de una célula es definida por el conjunto de células que se sitúan alrededor de ésta, sobre las cuales se ejerce una influencia de forma directa. A partir de lo anterior, es posible encontrar una función que determine exactamente la región de vecindad de cada célula. En general, esta región suele venir representada mediante una esfera, centrada en la propia célula, y cuyo radio r define el tamaño de la vecindad. En la práctica, sin embargo, lo habitual es que las redes formen estructuras bidimensionales de una sola capa y radios de vecindad reducidos. Un ejemplo de este tipo de redes, constituida por células con vecindad de radio $r = 1$ y con vecindad de radio $r = 2$, se muestra en la Fig. 1e.

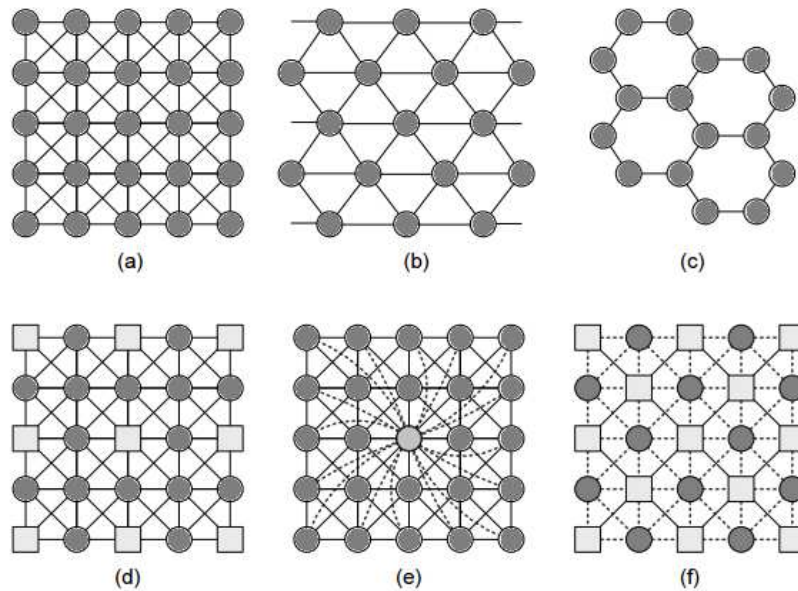


Fig. 1. Ejemplos de CNN con diferentes estructuras (a) rectangular, (b) triangular, (c) hexagonal (d) NUP-CNN, (e) diferente tamaño de vecindad y (f) conexiones distintas [7].

1.2. CNN en tiempo discreto

Las CNN de tiempo discreto (TDCNN) fueron propuestas por Hubert Harren, se definen como una clase especial de CNNs. Dichas redes, pueden ser descritas por un algoritmo recursivo cuya dinámica permite actuar en periodos definidos de tiempo [8]. La DTCNN se establece mediante las ecuaciones 1 y 2.

$$x_{ij}(k) = -x_{ij}(k) + A * y_{ij}(k) + B * u_{ij} + I \quad (1)$$

$$y_{ij}(k + 1) = f(x_{ij}(k)) = \begin{cases} 1 & \text{si } x_{ij}(k) > 0 \\ 0 & \text{si } x_{ij}(k) < 0 \end{cases} \quad (2)$$

Dónde: u_{ij}, x_{ij}, y_{ij} son las variables de entrada, estado y salida respectivamente, A (matriz de retroalimentación) y B (matriz de control) son matrices del tamaño de vecindad definida, e I es un valor de *offset*. Este conjunto de parámetros es llamado

cloning template. La ventaja primordial de las DTCNN se debe a que la descripción de la salida para el siguiente estado puede ser descrito por un conjunto de desigualdades lineales [8]. Además, la simulación en una computadora digital es más sencilla, debido a que no requiere ningún algoritmo de integración, como en el modelo original el cual está basado en ecuaciones diferenciales en tiempo continuo.

De manera general el funcionamiento de una red celular neuronal de cualquier tipo se representa mediante la Fig. 2 [9].

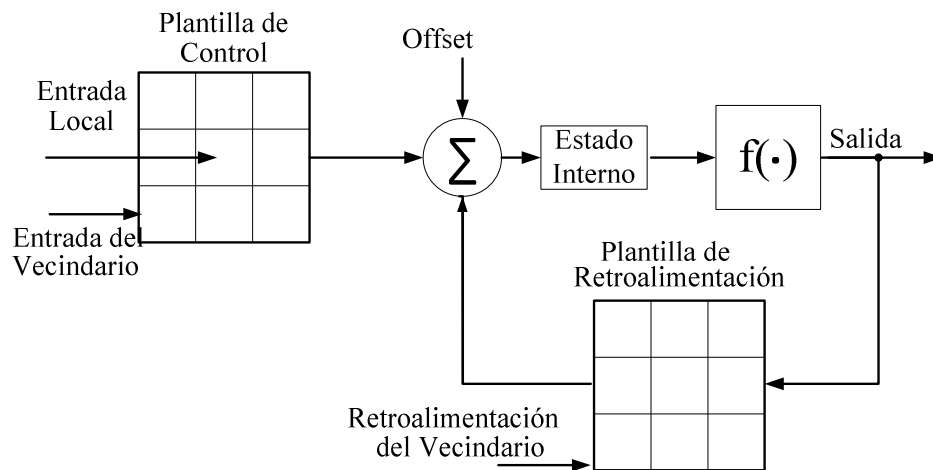


Fig. 2. Modelo funcional de una CNN.

1.3. Procesamiento digital de Imágenes y la CNN

Una imagen digital es la representación bidimensional de una imagen real, expresado como una función de la intensidad de luz $f(x,y)$, donde el valor de la amplitud f en el punto definido por las coordenadas espaciales (x,y) , está determinado por el brillo de la imagen original en ese punto, denominado píxel. Un píxel p con coordenadas (x,y) tiene dos vecinos horizontales, dos verticales y cuatro diagonales, cada uno de estos píxeles con una distancia 1 de p y en los bordes algunos de estos elementos quedan fuera de la imagen [10], lo anterior es mostrado en la Fig. 3.

Entre las primeras aportaciones de las CNNs al procesamiento digital de imágenes cabe destacar los diseños de plantillas (*templates*) que han marcado una tendencia sobre el desarrollo de algoritmos utilizados en distintos campos de la inteligencia artificial, entre otros. Estas redes definen procesos que explotan la conectividad en retroalimentación del modelo original de la CNN [1], caracterizando la red como un procesador de imágenes binarias, capaz de realizar procesamientos lineales y no lineales, operaciones morfológicas como el rellenado automático de huecos, la codificación de caracteres y la detección de elementos simples como líneas y bordes, entre otros [1,2 y 4]. Las transformaciones morfológicas, que parten de la teoría de conjuntos, son herramientas matemáticas que se encargan de modificar formas y estructuras de objetos, además de obtener características especiales de estos. Dichas transformaciones se realizan en imágenes binarias, aunque también existen casos para su aplicación en imágenes con escalas de grises.

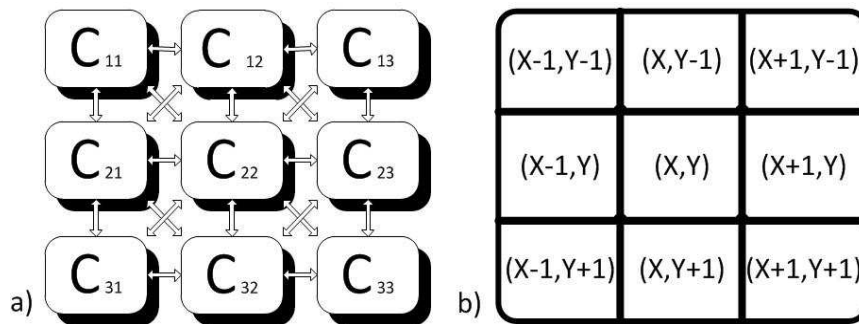


Fig. 3. Ejemplo de arreglos bidimensionales, (a) CNN 3x3 (b) imagen 3x3.

2. Desarrollo

En este trabajo se propone una estructura de una red neuronal celular para procesamiento de imágenes a escala de grises por medio de la utilización de Hardware Reconfigurable FPGA. La red consta de un conjunto de veinticuatro elementos, conectados totalmente por vecindarios de radio $r = 1$ en arreglos rectangulares, es decir, cada neurona se puede comunicar hasta con ocho vecinos, como se ilustra en la Fig. 4.

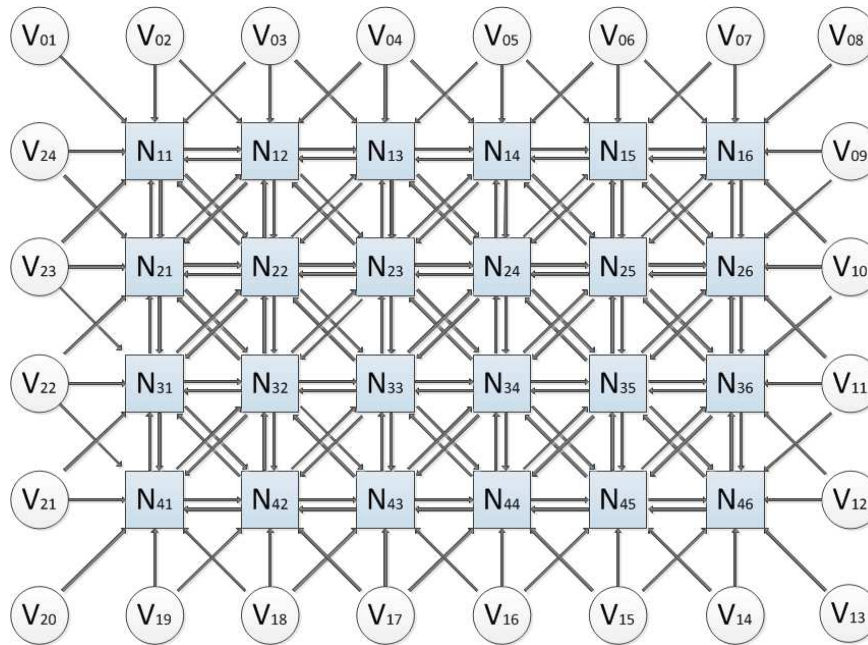


Fig. 4. Diagrama general de la red.

2.1. Registros de configuración

La comunicación entre los elementos de la red se lleva a cabo a nivel local, con la posibilidad de alterar su salida mediante la variación de los siguientes parámetros, buscando obtener operaciones que puedan ser utilizadas para el procesamiento de imágenes en escala de grises:

- Peso o valor a incrementar en la entrada original.
- Tipo de conectividad.
- Retroalimentación consigo misma y/o un valor adicional (offset).

Para ello se debe tener conocimiento por medio de registros específicos, los cuales son enviados desde una unidad de control global a toda la red (decodificador) y son recibidos en cada célula que los reconoce y actúa de la manera en que se le indique. Los cuatro tipos de registros de configuración generados se muestran en la Tabla 1.

Nombre del registro	Función del registro	Tipo	Número de bits
Registro_control_peso	Modifica directamente la entrada (peso), ya que es multiplicado por valor original.	Valor	4
Registro_control_in	Asigna los enlaces entre elementos, habilita de acuerdo a un orden en sentido de las manecillas del reloj.	Bus	9
Registro_control_SR	Define el tipo de operación entre los enlaces, suma o resta los valores vecinos, lleva el mismo orden que <i>Registro_control_in</i>	Bus	9
Registro_control_retro	Modifica directamente la salida sumando un <i>offset</i> , al resultado.	Valor	8

Tabla 1. Tipos de registros de configuración.

La Fig. 5 muestra los registros de configuración y el tipo de conectividad que genera en caso de requerir influencia con los vecinos laterales o los diagonales.

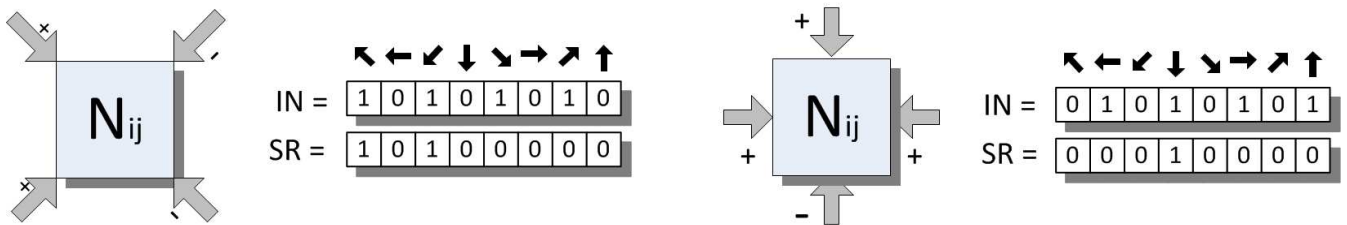


Fig. 5. Ejemplo de configuración para influencia lateral.

Cualquier alteración en algún bit de los registro representa un cambio considerable en el resultado a la salida de la red, por lo que se deben identificar las estructuras de conexión necesarias para realizar procesamientos definidos.

Con esta metodología es posible asignar conexiones tanto de manera global como local, configurar las neuronas individualmente buscando operaciones específicas a nivel pixel, o realizar conexiones homogéneas en toda la red para obtener filtrados espaciales genéricos, de igual manera realizar configuraciones híbridas, conectando una parte de la red de manera similar y el resto de manera específica.

2.2. Elementos de procesamiento

La Fig. 6 muestra el diagrama a bloques de la estructura interna de un elemento de procesamiento, donde destaca un módulo que calcula el peso asignado a la entrada original más el valor del vecindario, de acuerdo al tipo de conectividad que se desee, y otro módulo para añadir una posible retroalimentación más un valor determinado (*offset*), además de un registro de salida que almacena el dato después de cierta cantidad de flancos.

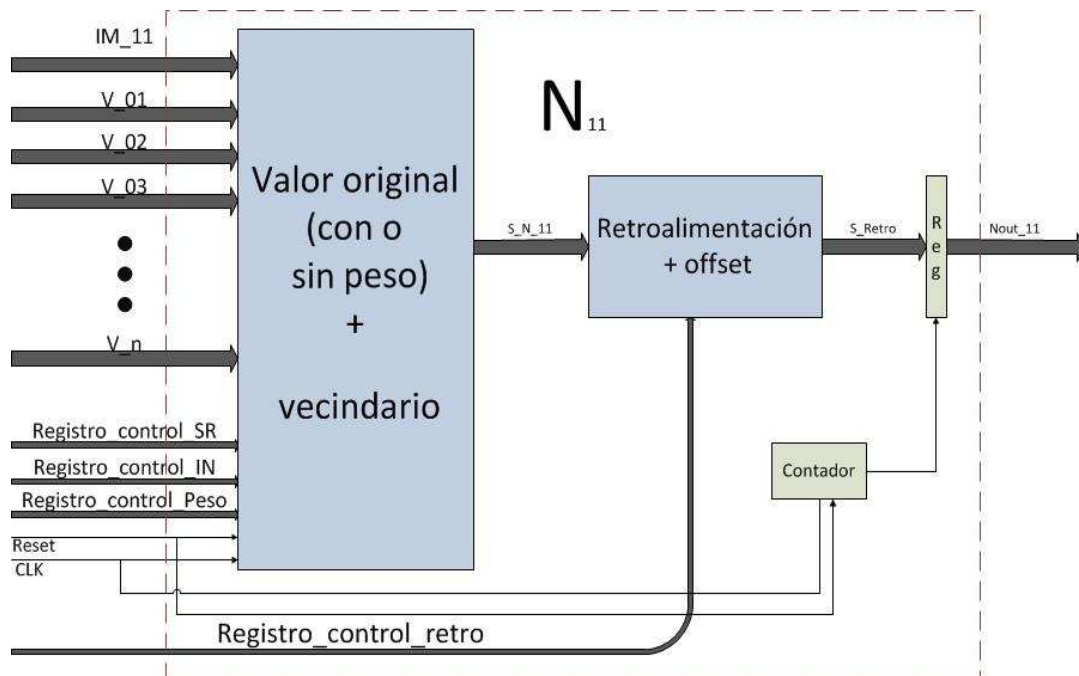


Fig. 6. Diagrama a bloques de un elemento de la red.

2.2.1. Vecindario

Este bloque recibe datos de la entrada original, de las neuronas vecinas y señales de control. El dato de entrada es almacenado en un primer registro síncrono que trabaja en flancos de subida, posteriormente es modificado por un peso predefinido, seguido de esto, se añade el valor del vecindario, el cual se obtiene mediante un arreglo de entidades aritméticas conectadas entre sí.

Los datos de los elementos vecinos se adquieren por medio de una unidad de control interna, que valida su habilitación de acuerdo a los registros de configuración recibidos a la entrada, siendo un bit del tipo *registro_control_in* para cada vecino posible, y de igual manera respetando el mismo orden se indica el tipo de operación local con *registro_control_sr*. Finalmente cuenta con otro registro síncrono de almacenamiento que trabaja en flancos de bajada, útil para evitar problemas de inestabilidad al presentarse cambios en las entradas produciendo variaciones no deseadas a la salida. El diagrama de este bloque se observa en la Fig. 7.

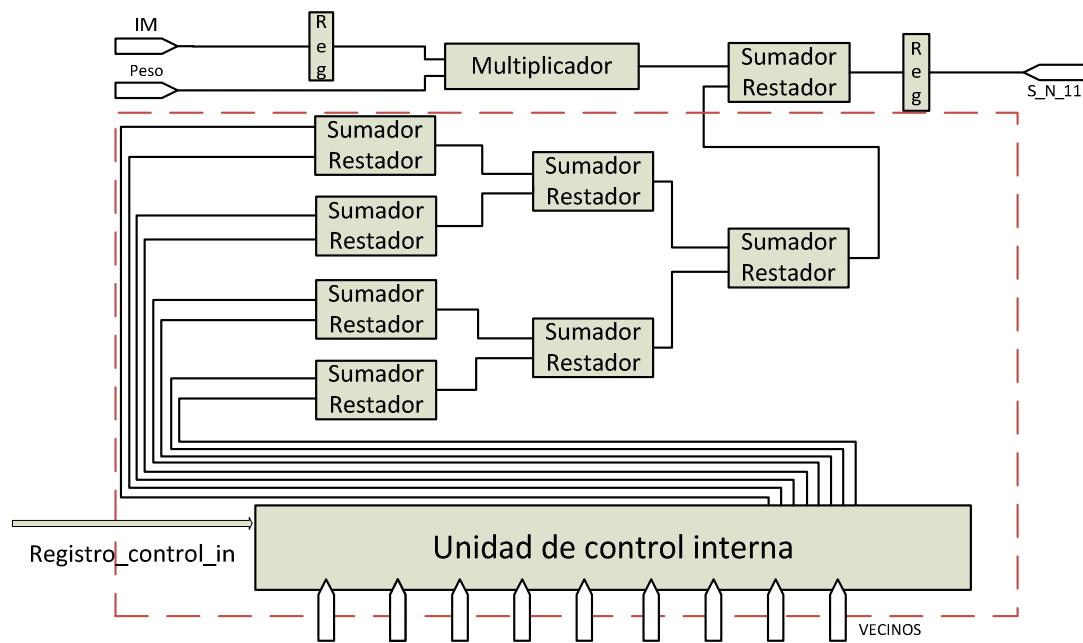


Fig. 7. Estructura de cálculo de vecindario neurona.

2.2.2. Módulo de retroalimentación y offset

En este módulo, ilustrado en la Fig. 8, se le agrega un nuevo término a la salida del cálculo de vecindario y el valor original modificado, dicho término corresponde al valor de retroalimentación, más un offset, el cual representa una cantidad predefinida que permite una mayor gama de posibilidades a la salida final de la red. Este efecto es habilitado mediante el último bit del *registro_control_in*, que indica al multiplexor si es necesario aplicarlo.

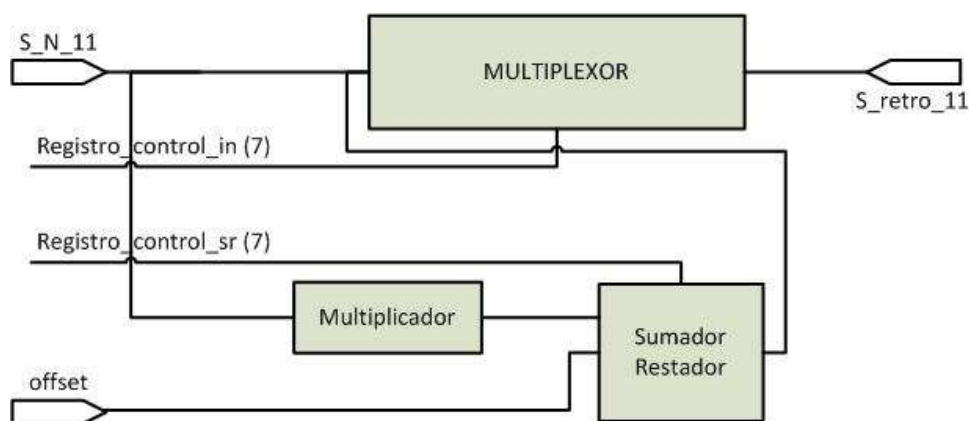


Fig. 8. Módulo de retroalimentación y offset.

Con esta estructura es posible generalizar el comportamiento de cada elemento de la red en base a los siguientes puntos:

- Modificar el valor de entrada en relación a un peso. (*registro_control_peso* > 1) o utilizar el valor original (*registro_control_peso* = 1).
- Hacer un procesamiento en base a la conexión de cada elemento con su vecindario, ya sea que todos los elementos se comporten de manera semejante (*registro_control_in*, *registro_control_sr* iguales en todos), o efectuar el procesamiento con conexiones particulares en deferentes elementos (*registro_control_in*, *registro_control_sr* independientes para cada elemento).

2.3. Funcionamiento

Para fines prácticos y poder corroborar el funcionamiento de esta metodología se acota las salidas de la red a 16 casos específicos, para lo cual se desarrolla un decodificador.

En la Tabla 2 se muestran como ejemplo los casos: 0000, 1000, 1111, donde existe una modificación de los registros de configuración para la red.

No. de caso	Código	Operaciones	Conexiones necesarias	Parámetros
1	0000	Dilatación bilateral (Homogénea)	Habilitación de vecinos izquierdo y derecho	Peso = 1 Retro = 0 offset = 0
8	0111	Detección de Contornos	Habilitación de todos los vecinos	Peso = 8 Retro = 1 offset = 0
16	1111	Conectividad específica por neurona	Cada neurona con conexión independiente	Peso = 11 Retro = 1 offset = - 2

Tabla 2. Casos específicos implementados en este trabajo.

Para realizar la demostración del funcionamiento del módulo de la red, se toma como valores de entrada los mostrados en la matriz A, y los resultados esperados para cada caso se muestran de manera independiente en relación a los pesos y las conexiones asignados por los registros de configuración.

ENTRADA

A =

1	3	1	5	1	1
3	1	5	1	3	5
1	3	1	5	1	1
3	1	5	1	3	3

Matriz de pesos

Registros de configuración

A₀₀₀₀

4	5	9	7	7	2
4	9	7	9	9	8
4	5	9	7	7	2
4	9	7	9	7	6

0	0	0
1	1	1
0	0	0

SR
IN
peso

0	0	1	0	1	0	1	0	0		
0	0	1	0	1	0	1	0	0		
							0	0	0	1

A₁₀₀₀

0	0	7	0	7	1
0	10	0	14	0	0
3	0	14	0	14	7
0	5	0	7	0	0

1	1	1
1	-8	1
1	1	1

SR
IN
peso

1	0	0	0	0	0	0	0	0		
1	1	1	1	1	1	1	1	1		
							1	0	0	0

A₁₁₁₁

3	28	0	0	6	0
28	0	51	2	10	0
0	27	0	49	0	0
29	0	52	0	28	29

0	-1	0
-1	11	-1
0	-1	0

SR
IN
peso

0	0	1	0	1	0	1	0	0		
1	0	1	0	1	0	1	0	0		
							1	0	1	1

offset = -2

1	0	1
0	-4	0
1	0	1

SR
IN
peso

1	0	0	0	0	0	0	0	0		
1	1	0	1	0	1	0	1	0		
							0	1	0	0

2.4. Procesamiento de una imagen

Para desarrollar el procesamiento de una imagen de tamaño M x N en escala de grises, se realiza el recorrido del módulo CNN por todo el patrón de entrada, debido a que el proceso se realiza solo con 24 píxeles de la imagen (tamaño de la red) a la vez, este trabajo propone un sistema que efectúe dicho recorrido.

La imagen es una matriz y cada pixel se almacena en una localidad de memoria, la cual envía datos hacia el módulo de la red mediante un bus de direccionamiento, manejado por una entidad de control encargada de la correcta distribución de los datos, entre una memoria de entrada y otra de salida, donde se guardan los datos procesados, a razón de ser requeridos para su posible visualización. El diagrama a bloques del sistema generado por esta metodología se muestra en la Fig. 9(a), y la Fig. 9(b) es un ejemplo del desplazamiento a través de la matriz de entrada, haciendo un barrido horizontal, es decir de izquierda a derecha y de arriba hacia abajo.

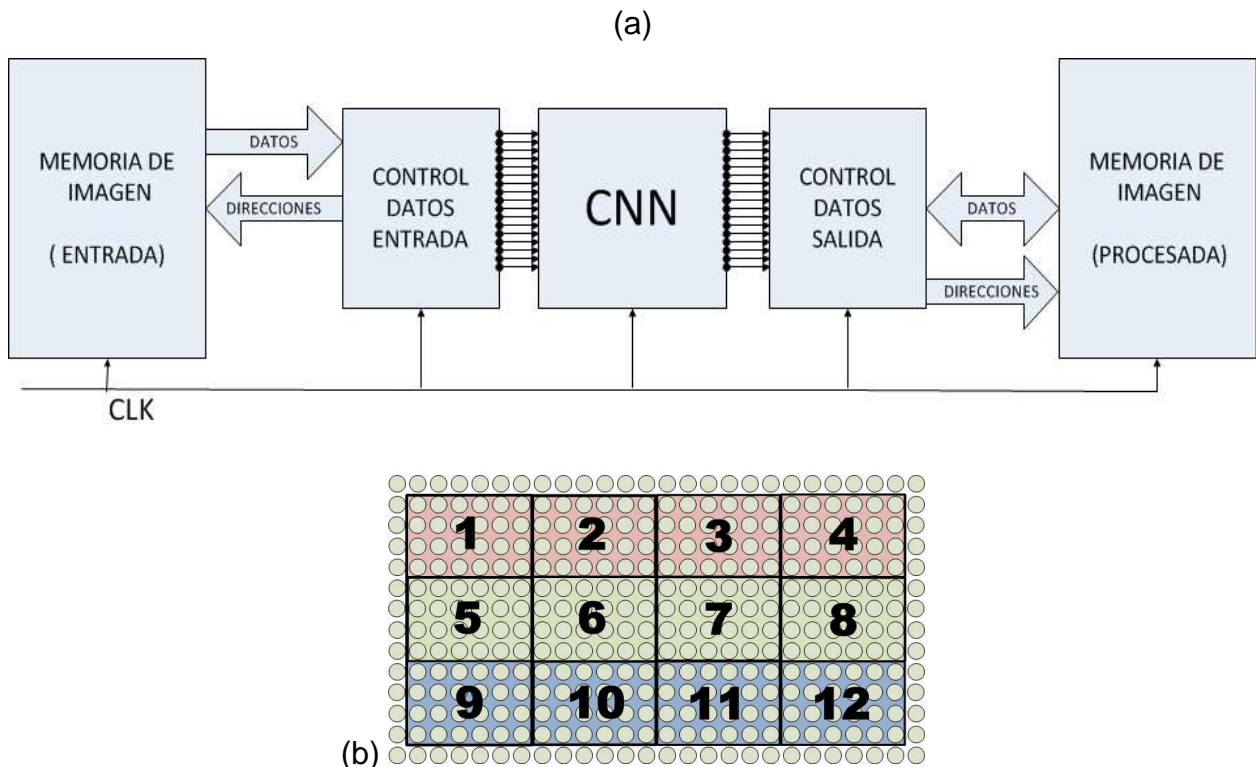


Fig. 9. (a) Diagrama a bloques del sistema completo, (b) ejemplo del barrido de la imagen.

En esta etapa del diseño, se utilizó la herramienta de software MATLAB®, para simulación y validación, como medio de lectura de imagen y generador datos para almacenar en memorias, empleando un algoritmo iterativo para colocar pixeles enteros en la memoria del FPGA, y esta pueda proporcionar los datos de manera más eficiente,

colocando anchos de palabra múltiplos de 8 bits, y localidades de memoria acorde al tamaño de la imagen a procesar.

3. Resultados

La implementación de este trabajo se realiza en una plataforma reconfigurable de la familia Virtex-6 de Xilinx® [11-13], haciendo uso de las herramientas de diseño ISE 14.6. Los resultados de la simulación para el módulo de la red se muestran en la Fig. 10. El primer caso, cuando el bus control = “0000” se realiza una dilatación lateral, posteriormente se cambia el bus control = “1000” y por último se muestra la salida cuando el bus control = “1111”.

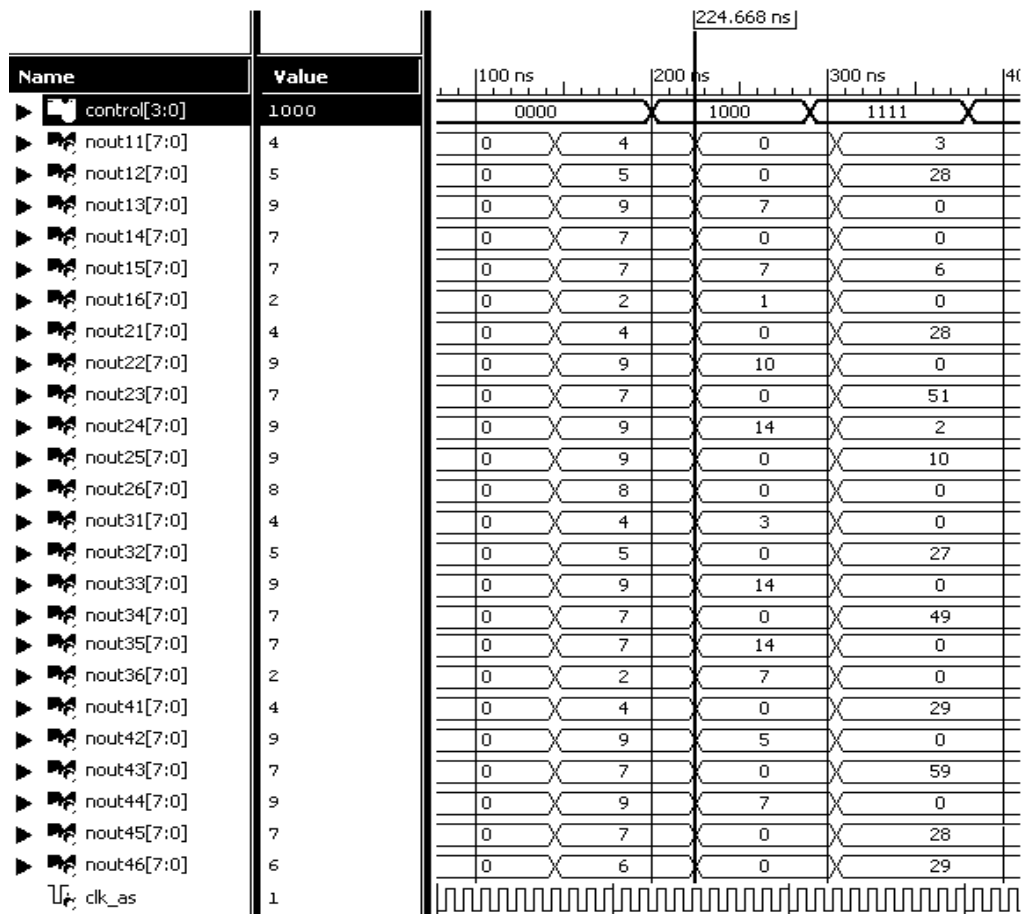


Fig. 10. Simulación del módulo CNN.

Cada tres pulsos de la señal de sincronización se produce el resultado, esto es debido a que el flujo de datos se presenta de la siguiente manera:

- En el primer flanco de subida se carga el dato a la entrada.
- En el primer flanco de bajada aún no hay actividad de los vecinos ya que se inicializan en cero por lo que solo se transfiere el valor original afectado por el peso.
- En el segundo flanco de subida ya se presentan datos del vecindario y se agregan al valor actual, es decir se realiza el procesamiento de las conexiones definidas.
- En el segundo flanco de bajada ya se cuenta con el dato más el vecindario y se envía a la retroalimentación si es que se este se habilita, de lo contrario se queda el valor actual.
- En el tercer flanco de subida se activa el último registro, perteneciente a la salida de la entidad y el valor final es mostrado.

Teniendo en cuenta lo anterior, se pueden considerar que la red tiene una latencia de alrededor de 100 ns, si se consideran pulsos de una duración no superior a los 20ns, aun cuando los cambios sean muy drásticos a la entrada, la estructura de segmentación de la red deberá poder mantener resultados estables en este lapso. Una vez obtenido el dato a la salida es necesario realizar un *reset* para poder cargar un dato nuevo y realizar otra operación, esto para evitar resultados erróneos causados por estados anteriores en los elementos de la red.

En la Tabla 3 se puede apreciar un breve resumen de los niveles de ocupación, tras la implementación del módulo CNN en diferentes FPGAs, siendo la Virtex-6 aquella con menor porcentaje de recursos utilizados.

	Usados	Totales	%	Usados	Totales	%
		Virtex-6			Spartan-6	
Registros	403	301,440	0.13369	403	54,576	0.73842
LUTs	3,525	150,720	2.33877	2536	27,288	4.67
Slices	1,419	37,680	3.76592	445	2,278	19.5347
		Artix-7			Zynq	
Registros	403	82,000	0.49146	403	35,200	1.14489
LUTs	3,472	63,400	5.47634	4,107	17,800	23.07307
Slices	1184	10250	11.55122	532	17,600	3.02273

Tabla 3. Niveles de ocupación del módulo CNN en diferentes FPGAs.

Por otro lado, al utilizar los módulos externos a la CNN, se pueden hacer pruebas para casos de imágenes completas, realizando los efectos de acuerdo a la configuración que se le asigne. En la Tabla 4 se muestran algunos ejemplos del procesamiento obtenido de acuerdo al tipo de conectividad entre elementos de la red y su respectiva matriz de pesos, todos los resultados teniendo en consideración la imagen de entrada ilustrada en Fig. 11, con una resolución 480x640. El tiempo total que el sistema obtuvo, es alrededor de 1.3 ms.



Fig. 11. Imagen de prueba.









Matriz de pesos	Registros de conexión	Resultado																											
<table border="1" style="width: 100%; text-align: center;"> <tr><td>-1</td><td>0</td><td>-1</td></tr> <tr><td>0</td><td>5</td><td>0</td></tr> <tr><td>-1</td><td>0</td><td>-1</td></tr> </table>	-1	0	-1	0	5	0	-1	0	-1	<p style="text-align: center;">Realce</p> <p style="text-align: center;">  </p> <table border="1" style="width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> </table>	1	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	
-1	0	-1																											
0	5	0																											
-1	0	-1																											
1	1	0	1	0	1	0	1	0																					
0	1	0	1	0	1	0	1	0																					
<table border="1" style="width: 100%; text-align: center;"> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>-1</td><td>11</td><td>-1</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> </table>	0	-1	0	-1	11	-1	0	-1	0	<p style="text-align: center;">Aumento de brillo</p> <p style="text-align: center;">  </p> <table border="1" style="width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </table>	1	1	1	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	
0	-1	0																											
-1	11	-1																											
0	-1	0																											
1	1	1	0	1	0	1	0	1																					
0	1	1	0	1	0	1	0	1																					
<table border="1" style="width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>-8</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	-8	1	1	1	1	<p style="text-align: center;">Detección de bordes</p> <p style="text-align: center;">  </p> <table border="1" style="width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
1	1	1																											
1	-8	1																											
1	1	1																											
1	1	1	1	1	1	1	1	1																					
1	0	0	0	0	0	0	0	0																					
<table border="1" style="width: 100%; text-align: center;"> <tr><td>-1</td><td>-1</td><td>-1</td></tr> <tr><td>-1</td><td>11</td><td>-1</td></tr> <tr><td>-1</td><td>-1</td><td>-1</td></tr> </table>	-1	-1	-1	-1	11	-1	-1	-1	-1	<p style="text-align: center;">Aumento de contraste</p> <p style="text-align: center;">  </p> <table border="1" style="width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	
-1	-1	-1																											
-1	11	-1																											
-1	-1	-1																											
1	1	1	1	1	1	1	1	1																					
0	1	1	1	1	1	1	1	1																					

Tabla 4. Resultados del procesamiento utilizando diferentes configuraciones.

4. Conclusiones

En este trabajo se presentó la descripción y los resultados de la implementación de una red neuronal celular totalmente programable en dispositivos reconfigurables FPGAs. En base a esta propuesta, es posible realizar cualquier configuración de conectividad en la CNN mediante cambios en registros específicos, diseñados para otorgar una gran flexibilidad y realizar diferentes operaciones, generando una amplia gama de posibilidades a su salida. Los resultados expuestos en la Tabla 4, así como la latencia del módulo mostrada en la Fig. 10, demuestra que el sistema de procesamiento de imágenes por medio de una red neuronal celular programable, es una alternativa viable para realizar operaciones morfológicas, y que la opción de poder configurar su conectividad, tanto en enlaces como en pesos, resulta en una estructura de procesamiento en Hardware capaz de adaptarse para ser usada en conjunto con cualquier sistema o microprocesador.

5. Referencias

- [1] L. O. Chua, T. Roska, T. Kozek, A. Sarandi, "The CNN paradigm short tutorial book of Cellular Neural Networks". John Wiley. 1993. 1- 14.
- [2] L. O. Chua, "Cellular Neural Networks: Theory". IEEE Transactions on Circuits and Systems. Vol. 35. No. 9. 1988. 1257-1272.
- [3] A. Flores, E. Gómez, "Tutorial sobre Redes Neuronales Celulares: Aplicación al Procesamiento de Imágenes". LIDETEA. Universidad La Salle. 2004.
- [4] T. B. Roska, P. Thira, L.O. Chua, "Detecting Simple Motion Using Cellular Neural Networks". Proc. First IEEE Int. Workshop on Cellular Neural Networks and their Applications. 1990. Budapest. 127-138 pp.

- [5] T. Roska, L. Chua, "Cellular Neural Networks with Non-Linear and Delay-Type Template Elements and Non-Uniform Grids". *Int'l Journal of Circuit Theory and Applications*. Vol. 20. 1992. 469-481 pp.
- [6] J. J. Raygoza-Panduro, J.J., Ortega-Cisneros S., Salazar Santos, J. C. Ibarra, De la Mora A., Cárdenas-Rodríguez R, "Prototipado de una Red Neuronal Celular Digital en FPGAs". Segundo congreso CONCIBE y CACIF. Vol.1. 2006. 1-6 pp.
- [7] A.G. Radvanyi, "On the rectangular grid representation of general CNN networks, Cellular Neural Networks and Their Applications". *Proceedings of the 2000 6th IEEE International Workshop, Digital Object Identifier No. 10.1109/CNNA.2000.877360*. May 2000. 387 – 393, 23-25 pp.
- [8] H. Harrer, "Multiple layer discrete-time cellular neural networks using time-variant templates". *IEEE Transactions on Circuits and Systems part II*. Vol. 40. 1993. 191–199 pp.
- [9] L. O. Chua and T. Roska, *Cellular neural networks and visual computing-Foundation and applications*. 2002. Cambridge University Press. 29-30 pp.
- [10] R. C. Gonzalez, R. E. Woods, *Digital Image Processing*. 3rd ed. 2008. Prentice Hall. Upper Saddle River, NJ.
- [11] Xilinx Inc. *Getting Started with the Xilinx Virtex-6 FPGA ML605 Evaluation Kit, manual XPN 0402771-01 UG533 (v1.5)*. www.xilinx.com. Oct. 20 2011.
- [12] Xilinx Inc., *ML605 Hardware User Guide UG534 (v1.8)*. www.xilinx.com. Oct. 2 2012.
- [13] "Virtex-6 Family Overview. Xilinx. 2012.