A Scalable Neuromorphic Architecture to Efficiently Compute Spatial Image Filtering of High Image Resolution and Size

M. Abarca, G. Sánchez, L. García, J. Avalos, T. Frías, K. Toscano, and H. Pérez, *Senior Member, IEEE*

***Abstract*—In this work, we propose a spiking P neuron which is capable of performing spatial filtering operations by using new variants of the spiking neural P systems, such as synaptic weights and rules on the synapses. The inclusion of these variants have allowed us to create a compact spiking P neuron with minimal number of synapses and low computational complexity of the spiking rules. In addition, we propose a multi-FPGA neuromorphic system to support an array of very large-scale spiking P neurons to process high image resolution at high processing speeds. These neurons can be simulated by using a scalable configurable parallel hardware architecture, where its basic processing unit is a single spiking P neuron. Our results show that the proposed architecture is up to 54 and 12 times faster when compared to advanced Graphical Processing Units (GPU), and high performance CPUs, respectively. On the other hand, our proposal is $55x10^3$ times faster than the best of existing FPGA-based neuromorphic solution.**

***Index Terms*—Spiking neural P systems, Spatial image filtering, FPGA.**

I. Introducción

En los últimos años, el uso de la computación paralela se ha incrementado significativamente, especialmente en el campo de procesamiento de imágenes-video. Esto con el fin de desarrollar aplicaciones avanzadas, tales como visión robótica, interacción hombre-máquina, video vigilancia, entre otras. Recientemente, algunos autores han propuesto modelos informáticos altamente paralelos inspirados en las características intrínsecas de computación paralela distribuida de la corteza visual humana, en donde la adquisición y el procesamiento de la información se lleva acabo de manera paralela. Desde el punto de vista de la ingeniería, estas características podrían resultar de gran utilidad en el desarrollo de sistemas de visión avanzados. Específicamente, algunos de estos modelos, los cuales se basan en los denominados sistemas P neuronales (SN P), han sido propuestos para simular filtros espaciales, ya que estos son una parte fundamental en el desarrollo de sistemas avanzados de visión.

Las redes neuronales denominadas tipo P spiking (en inglés: spiking neural P systems) son el resultado de la combinación de dos áreas de cómputo natural, las cuales son: cómputo de membrana y cómputo neuronal tipo spiking [1]. Específicamente, el cómputo de membrana se enfoca únicamente en el modelado del soma o la membrana, la cual es la unidad principal de procesamiento de la neurona. Por otra parte, el cómputo tipo spiking modela la comunicación entre neuronas, la cual se establece mediante pulsos eléctricos denominados spikes. Recientemente, varios autores han demostrado que las redes neuronales tipo P spiking son idóneas para ser utilizadas en el procesamiento de imágenes en paralelo mediante filtros digitales [2,3]. Sin embargo, hasta la fecha estas redes neuronales tipo P spiking no han sido implementadas en hardware debido a que se requiere de un gran número de conexiones sinápticas y complejos núcleos de procesamiento para simular este tipo de red neuronal en tiempo real. Una posible solución en la implementación eficiente de este tipo de redes neuronales en hardware, la cual contenga múltiples núcleos de procesamiento, se podría basar en dispositivos GPUs, dado que estos dispositivos han surgido como una solución para realizar el cómputo de algoritmos de filtrado espacial de imágenes en paralelo [4]. Sin embargo, las GPUs tienen varias limitaciones dado que la velocidad de procesamiento de sus procesadores está condicionada por el ancho de banda de la memoria, el número de hilos y la transferencia de los datos entre procesadores. Por otra parte, las soluciones basadas en FPGA se han vuelto una alternativa importante para el procesamiento de algoritmos de filtrado espacial de imágenes en paralelo debido a sus avanzados sistemas de memoria e interfaces de comunicaciones de alta velocidad, los cuales los convierten en herramientas adecuadas para el procesamiento de imágenes en paralelo [5,6]. Específicamente, la implementación en paralelo de filtros digitales para el procesamiento de imágenes de gran tamaño y

M. Abarca is with the Universidad Politécnica de Texcoco, Texcoco, México (e-mail: marco.a.abarca.r@gmail.com).

G. Sánchez is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: gsanchezriv@ipn.mx).

L. García is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: jorgeluis102008@hotmail.com).

J. G. Avalos is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail:javaloso@ipn.mx).

T. Frías is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: thania.frias05@gmail.com

K. Toscano is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: caaann@gmail.com).

H. M. Pérez is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: hperez.meana@gmail.com).

Corresponding author: Giovanny Sánchez (gsanchezriv@ipn.mx).

alta resolución demanda principalmente una gran cantidad de circuitos aritméticos como: multiplicadores, divisores, sumadores y restadores. Sin embargo, en los dispositivos digitales actuales (FPGAs y GPUs) el número disponible de estos circuitos aritméticos es limitado. Recientemente, diversos investigadores, inspirados en el procesamiento y la transmisión de información a través de las neuronas, han propuesto poderosos métodos computacionales [1]. Particularmente, han propuesto circuitos aritméticos neuronales seriales basados en redes neuronales tipo P spiking, los cuales puedan realizar multiplicaciones, divisiones, sumas, restas, con el objetivo de mejorar la eficiencia computacional y el consumo de hardware de los circuitos aritméticos binarios convencionales [7]. Sin embargo, estas propuestas solo se han basado únicamente en las capacidades computacionales del soma. A consecuencia de esto, los circuitos aritméticos existentes hacen uso de varias sinapsis/neuronas y reglas complejas. Por lo tanto, estos factores han limitado su implementación, ya sea en software o en hardware, por lo que su rendimiento no ha sido validado. Recientemente, varios autores han propuesto nuevas variantes para incrementar la capacidad computacional del soma. Algunas de estas variantes están inspiradas por varios fenómenos neuro-biológicos tales como: sistemas neuronales tipo P spiking con reglas en las sinapsis [8], sistemas neuronales tipo P spiking con pesos en las sinapsis [9], con retardos dendríticos [10], conexiones de realimentación [10], entre otros. Por lo tanto, la inclusión de estas nuevas variantes en la estructura de los actuales circuitos aritméticos neurales podrían mejorar sus capacidades computacionales al reducir el número de conexiones sinápticas / neuronas y la complejidad de las reglas del soma.

En general, existen dos grandes retos en el desarrollo de circuitos neuronales para ser utilizados en el procesamiento de imágenes:

1. El primer reto es crear circuitos aritméticos neuronales tipo P, en los cuales sus operandos sean adquiridos y procesados en paralelo.
2. El segundo reto se centra en el diseño de arquitecturas paralelas y su eficiente implementación en hardware para simular los nuevos circuitos neuronales.

En este trabajo se consideran estos dos puntos críticos con el fin de crear una arquitectura escalable, configurable y altamente paralela y su eficiente implementación en dispositivos FPGAs utilizando neuronas tipo P spiking para realizar el cómputo de diversos algoritmos de filtrado espacial de imágenes de gran resolución y tamaño a grandes velocidades de procesamiento y teniendo un bajo consumo de área/potencia. La alta velocidad de procesamiento y el bajo consumo de hardware/potencia por cada neurona se han conseguido debido a la consideración de neuronas tipo P con reglas simples y con un mínimo número de sinapsis por neurona, y al uso del método de codificación unaria de la información. Este método es denominado unario debido a que se basa en la representación de un solo símbolo (uno = spike), por lo tanto, la información es codificada en trenes de spikes. Por ejemplo, el número 5 es convertido en un tren de 5 spikes (“11111”). Por consiguiente, la arquitectura paralela propuesta podría potencialmente mejorar las prestaciones de las aplicaciones de procesamiento de imágenes en tiempo real.

II. Cómputo Convolucional

La convolución es una operación matemática fundamental para el desarrollo de diversos algoritmos de procesamiento de imágenes. Esta operación matemática multiplica dos arreglos de números con diferentes tamaños, para así generar un tercer arreglo. En procesamiento de imágenes, se puede definir a una imagen como una función espacial $f(x,y)$ de dos dimensiones (2-D), donde x y y son números naturales. Cada punto de la imagen (2-D) es llamado pixel y presenta diferentes valores de intensidad (niveles grises). El valor de salida de los pixeles es el resultado de una simple combinación lineal de los valores de entrada de pixeles específicos de la imagen, los cuales se obtienen usando los operadores de convolución. La operación de convolución es parte de los algoritmos considerados como filtros espaciales. El proceso de filtrado se realiza al mover una máscara (kernel) a través de los pixeles de una imagen. El modelo matemático de convolución se define como:

$$g(x,y) = \sum_{s=-m}^{m} \sum_{t=-m}^{m} w(s,t) f(x+s, y+t) \tag{1}$$

donde $g(x,y)$ representa la imagen filtrada, $w(s,t)$ es la máscara del filtro, $f(x+s,y+t)$ es un segmento de la imagen igual al tamaño de la máscara y m es el índice del kernel, donde m = (kernel-1)/2 para índices impares y m = (kernel)/2 para índices pares.

La ecuación (1) denota la operación de convolución de 3x3 pixeles con un kernel de 3x3. Dicha operación se describe como:

$$\begin{aligned} g(x,y) = {} & f(x-1,y-1)w(-1,-1) + f(x-1,y)w(-1,0) \\ & + f(x,y)w(0,0) + f(x-1,y+1)w(-1,1) \\ & + f(x,y-1)w(0,-1) + f(x,y+1)w(0,1) \\ & + f(x+1,y-1)w(1,-1) + f(x+1,y)w(1,0) \\ & + f(x+1,y+1)w(1,1) \end{aligned} \tag{2}$$

De acuerdo con el ejemplo anterior, una operación de convolución de 3x3 pixeles y un kernel de 3x3 requiere de 9 multiplicaciones y 8 sumas. En el caso de realizar la convolución en paralelo de una imagen de 512x512 con un kernel de 3x3, se requiere de 2,097,152 sumas y 2,359,296 multiplicaciones, es decir, en este ejemplo se aplica un gran número de kernels en paralelo a la imagen, en lugar de mover un solo kernel sobre una imagen. Por lo tanto, la implementación de este ejemplo demanda una gran cantidad de hardware (sumadores y multiplicadores). En particular, los multiplicadores producen un consumo significativo de área en los actuales dispositivos digitales (FPGAs y GPUs), lo que limita su uso en el procesamiento en paralelo de imágenes de gran tamaño y resolución [5,6]. En este trabajo se utilizan neuronas spiking tipo P con el objetivo de disminuir el consumo de área de los multiplicadores y los divisores dado que son los circuitos que demandan más consumo en hardware

para el procesamiento de diferentes tipos de filtros espaciales.

III. Convolución en Paralelo utilizando Redes Neuronales Tipo P Spiking

Antes de presentar nuestra propuesta de red neuronal tipo P spiking, la cual realiza el cómputo de la convolución en paralelo, definiremos de manera breve las bases teóricas de estos modelos neuronales. Los modelos de redes neuronales tipo P spiking están basados principalmente en el procesamiento del soma, el cual maneja la información mediante una codificación unaria de eventos o pulsos eléctricos (del inglés: spikes). El soma procesa los spikes por medio de dos reglas (disparo y olvido). Ambas reglas se definen como $E/a^c \rightarrow a^p : d$, donde E es una expresión regular sobre un símbolo a y c, d y p son números naturales, donde $c \geq 1,\ d \geq 0, p \geq 1$ y $c \geq p$. La regla de disparo del spike indica que, si una neurona contiene k spikes (a^k), donde $k \geq c$, entonces consume c spikes y produce p spikes de salida, después de un retraso de d pasos. La regla de olvido con la forma $a^s \rightarrow \lambda$ implica que $s \geq 1$ spikes son eliminados (λ), siempre que la neurona contenga exactamente s spikes [1].

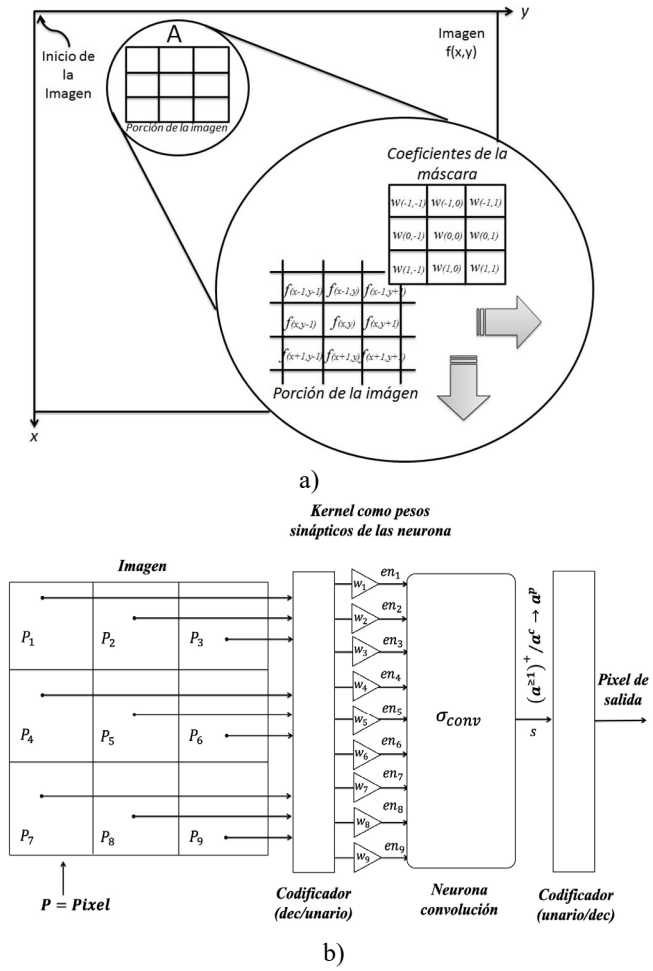


Fig. 1. a) Operación de convolución convencional, b) operación de convolución usando spikes.

Diversos algoritmos de filtrado espacial de imágenes emplean cuatro operaciones aritméticas básicas (resta, suma, multiplicación y división) para realizar la operación de convolución. En este trabajo, una sola neurona denominada neurona convolución σ_{conv} tipo P realiza las cuatro operaciones aritméticas antes mencionadas. Esto con el fin de crear un kernel con grandes capacidades computacionales que pueda realizar el filtrado de imágenes de alta resolución y gran tamaño en paralelo (ver Figura 1b). Para realizar la convolución con neuronas tipo P se tomaron en cuenta los siguientes criterios:

1.- El valor de los pixeles son convertidos en trenes de spikes, los cuales son procesados por la neurona convolución σ_{conv} de manera serial y en codificación unaria.

2.- El número de sinapsis de una sola neurona convolucional σ_{conv} define el tamaño del kernel m.

3.- Se propone utilizar los coeficientes del kernel $w(s,t)$ como pesos sinápticos de la neurona para el procesamiento de valores positivos o negativos del kernel. Cabe señalar que la consideración de los pesos sinápticos en los sistemas neuronales tipo P es considerada una nuevas variante [9].

4.- La regla en la sinapsis de salida de la neurona convolucional σ_{conv} define la salida del filtro. Cabe señalar que utilizamos esta nueva variante [8] para disminuir la complejidad computacional del soma.

La neurona convolucional σ_{conv} se puede defirnir formalmente como:

$$\sigma_{conv} = \left(O, en_1, en_2, en_3, en_4, en_5, en_6, en_7, en_8, en_9, s\right) \quad (3)$$

Donde:

- $O = \{a\}$;
- $sin = (origen, destino, peso\ sináptico, regla\ en\ sinapsis) = (en_1, \sigma_{conv}, w_1, 0), (en_2, \sigma_{conv}, w_2, 0), (en_3, \sigma_{conv}, w_3, 0), (en_4, \sigma_{conv}, w_4, 0), (en_5, \sigma_{conv}, w_5, 0), (en_6, \sigma_{conv}, w_6, 0), (en_7, \sigma_{conv}, w_7, 0), (en_8, \sigma_{conv}, w_8, 0), (en_9, \sigma_{conv}, w_9, 0), (\sigma_{conv}, s, 0, R_{conv}) \Rightarrow R_{conv} = \{(a^{\geq 1})^+ / a^c \rightarrow a^p\}$.

Donde: a representa un spike o impulso eléctrico, *sin* indica la sinapsis teniendo en cuenta el origen y destino de la conexión, w es su peso sináptico, R_{conv} define la regla en la sinapsis de salida de la neurona σ_{conv}, c indica el número de spikes que serán consumidos o eliminados del soma de la neurona σ_{conv} y p el número de spikes que serán generados, siempre y cuando se cumpla la regla.

La neurona σ_{conv} calcula la convolución de una imagen 3x3 y una máscara 3x3 como se muestra a continuación:

El codificador (dec/unario) se encarga de convertir los pixeles en su equivalente tren de spikes. Cada spike es enviado a la neurona σ_{conv} a través de su respectiva sinapsis de manera serial, sin embargo, todos los trenes de spikes son enviados de forma paralela a la neurona σ_{conv}, de tal manera que la neurona σ_{conv} puede recibir hasta 9 spikes al mismo tiempo mediante sus 9 sinapsis. Cabe señalar que la neurona σ_{conv} incrementará o decrementará el número de spikes contenidos en su membrana en función de sus pesos sinápticos y la regla

en su sinapsis de salida indicará la respuesta del filtro de la siguiente manera:

- *Operación de convolución*. La regla en la sinapsis de salida es configurada como: $(a^{\geq 1})^+/a^{c=9} \rightarrow a^{p=9}$, para indicar el resultado de la operación de convolución. En este caso, la regla en la sinapsis evalua el contenido del soma de la neurona para verificar que existan más de un spike $(a^{\geq 1})$. En caso de que existan nueve spikes, la neurona consumirá nueve spikes a^c, c=9, y generará a su salida nueve spikes a^p, p=9.
- *Filtro promedio.* En este caso la regla de la sinapsis de salida es definida como: $(a^{\geq 1})^+/a^{c=9} \rightarrow a^{p=1}$, lo que significa que cada vez que el soma de la neurona contenga más de un spike $(a^{\geq 1})$, la regla en la sinapsis se activa. En este caso, la condición de disparo se activa cuando el soma contiene nueve spikes. Por lo tanto, nueve spikes serán eliminados del soma a^c, c=9 y producirá un spike a la salida a^p, p=1. Por lo tanto, la condición de disparo es equivalente a la operación de división, es decir, la neurona σ_{conv} dispara un solo spike al recibir 9 spikes, entonces la operación que se realiza es $(9/9 \rightarrow 1)$.
- *Filtro Laplaciano-Gausiano.* La salida de la neurona provee la salida de este filtro cuando se configura la regla en sinapsis $((a^{\geq 1})^+/a^{c=1} \rightarrow a^{p=1})$. En este caso, la regla en la sinapsis evalua el contenido del soma de la neurona σ_{conv} para verificar que exista más de un spike $(a^{\geq 1})^+$, y en caso de que existan spikes, solo eliminará un spike a^c, c=1 y producirá un spike a la salida a^p, p=1.

En general, el uso de esta regla en la sinapsis de salida de la neurona σ_{conv}, permite crear un circuito compacto y con un gasto computacional mínimo dado que solo se requiere configurar c y p para obtener la respuesta deseada de diferentes filtros espaciales. Cabe señalar que estos valores se determinaron analizando los valores del kernel de cada filtro espacial. En el caso específico del filtro Laplaciano-Gausiano, la mayor contribución negativa esta en el centro del kernel, por lo que, cada spike p representa la diferencia en una unidad para distinguir un cambio de color. Además, se requiere configurar los pesos sinápticos de acuerdo con los valores del kenel del filtro espacial deseado. Finalmente, la función del codificador (unario/dec) es proporcionar el resultado en formato decimal, es decir el codificador (unario/dec) incrementa su contador interno en función de los spikes, los cuales son generados por la regla en la sinapsis de la neurona σ_{conv}.

IV. Arquitectura paralela basada en sistemas neuronales tipo P y su implementación en dispositivos FPGA

De acuerdo con la Figura 1b, nosotros diseñamos un elemento de procesamiento (EP) para implementar la neurona convolución σ_{conv}, tal como se muestra en la Figura 2. Este EP está compuesto principalmente por seis elementos: un vector de registros (P), un codificador (dec/unario), un vector de registros (w), un sumador, un comparador y un codificador de spikes (unario/dec). Los detalles de cada elemento del EP son presentados a continuación:

1. **Vector de registros (*P*)**. Estos registros almacenan los pixeles de la imagen. El número de registros (P) está en función del número total (n) de elementos del kernel.
2. **Codificador (dec/unario)**. Este circuito convierte los pixeles de la imagen (P) en trenes de spikes usando un conjunto de contadores, es decir, cada contador incrementa su cuenta y al mismo tiempo genera un spike (uno) en cada ciclo de reloj hasta alcanzar el valor del pixel.
3. **Vector de registros (*w*).** Estos registros almacenan los pesos sinápticos de la neurona σ_{conv}. El número de registros está en función del número total (n) de elementos del kernel.
4. **Sumador de 10 bit**s. Este elemento imita el comportamiento del soma de la neurona σ_{conv}. Por lo tanto, este elemento realiza la operación de la suma o resta de los potenciales sinápticos dependiendo del valor de cada peso sináptico (w), es decir, en el caso de pesos sinápticos positivos, se lleva a cabo la operación de suma. En caso contrario se realiza la operación de resta para pesos sinápticos negativos. El uso de esta estrategia evitar usar la operación de multiplicación, puesto que, solo hay que incrementar o decrementar el sumador proporcionalmente al valor de los potenciales sinápticos (w).
5. **Comparador 10 bits.** Este elemento implementa la regla en la sinapsis $((a^{\geq 1})^+/a^c \rightarrow a^p)$ de la neurona σ_{conv}. Por lo tanto, el contenido del sumador es comparado con el valor c para producir p spikes.
6. **Codificador (unario/dec).** Este codificador está compuesto básicamente por un sumador. Este sumador cuenta los p spikes, los cuales han sido generados por el comparador, para dar el resultado de la imagen filtrada en formato decimal.

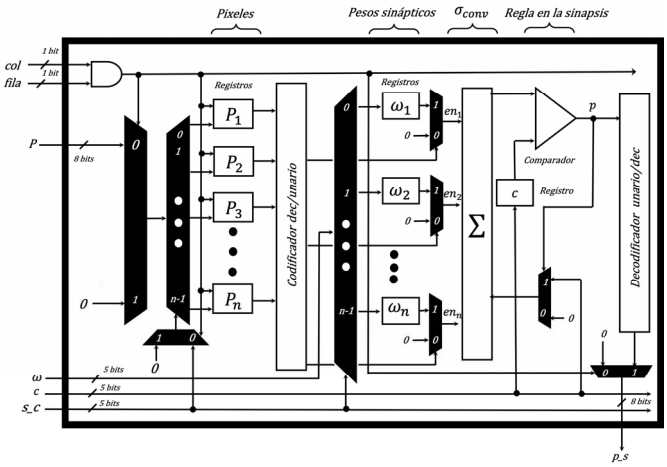


Fig. 2. Estructura del procesador para la implementación de la neurona convolucional σ_{conv} utilizando circuitos digitales binarios.

Una vez validado el EP, diseñamos la arquitectura paralela, la cual está compuesta de una matriz de EPs para simular diversos filtros espaciales. Esta matriz es etiquetada como módulo de procesamiento (ver Figura 3). Además de este módulo se utilizaron una unidad de configuración y un módulo Gigabit Ethernet para enviar los pesos sinápticos y los pixeles

de la imagen desde un computador hacia la arquitectura paralela mediante un protocolo Ethernet. Una vez filtrada la imagen, estos módulos se encargan de enviar los pixeles de la imagen filtrada (*p_s*) desde la arquitectura paralela hacia el computador.

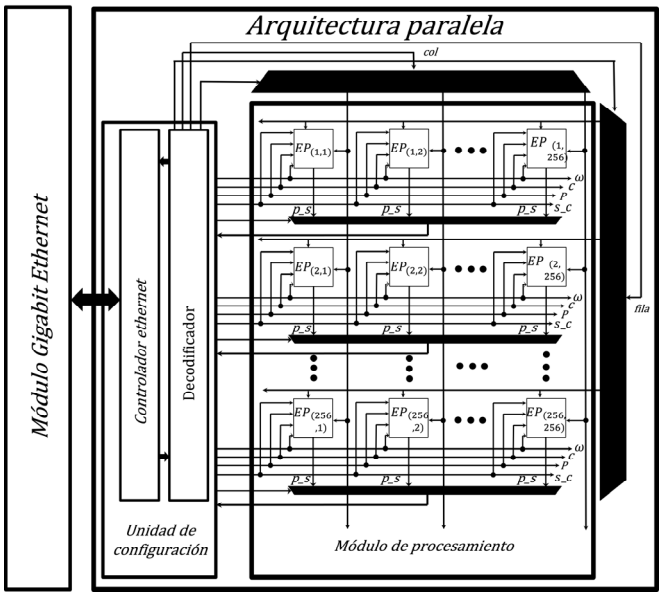


Fig. 3. Esquema de la arquitectura paralela.

A. Detalles del Protocolo de Comunicación Gigabit-Ethernet para Establecer la Comunicación entre el FPGA y el Computador

Como se menciono anteriormente, el FPGA cuenta con una interfaz de datos Gb Ethernet (una unidad de configuración y un módulo Gb Ethernet) para realizar tres tareas: configuración de los EPs, envio de los pixeles de la imagen y visualización de la imagen filtrada. Primeramente, el usuario ordena los pesos sináticos de cada neurona en una trama Ethernet utilizando un script en C, tal como se muestra en la Figura 4a. Como se puede observar, los pesos sinápticos son ordenados por cada fila de procesadores, iniciando por la fila = 0 y columna 0. Una vez enviada la trama desde el computador hacia el FPGA, el controlador Ethernet recibe el paquete para que el decodificador distribuya a cada procesador sus respectivos pesos sinápticos. Para realizar dicha tarea, el decodificador utiliza las señales (fila y col) para controlar sus respectivos multiplexores e indicar al procesador destino sus respectivos datos de entrada. Específicamente, con estas señales, el decodificador habilita la escritura de los pesos sinápticos en los registros (*w*) de cada procesador mediante una compuerta lógica AND, tal como se muestra en la Figura 2. De la misma manera, el usuario ordena e envía las tramas, las cuales contienen los pixeles de la imagen, por cada fila de procesadores y donde cada procesador recibe los pixeles de entrada como en el caso anterior. Evidentemente, en este caso los pixeles son almacenados en los registros *P* mediante la señal *s_c*. Cabe señalar que esta señal indica el índice del registro al cual se desea escribir. Una vez que la imagen es filtrada, el decodificador envía los pixeles de salida *p_s*, por cada fila, como se muestra en la Figura 4c. En general, la organización de los datos en sus respectivas tramas ha facilitado su distribución dentro de la matriz de procesadores.

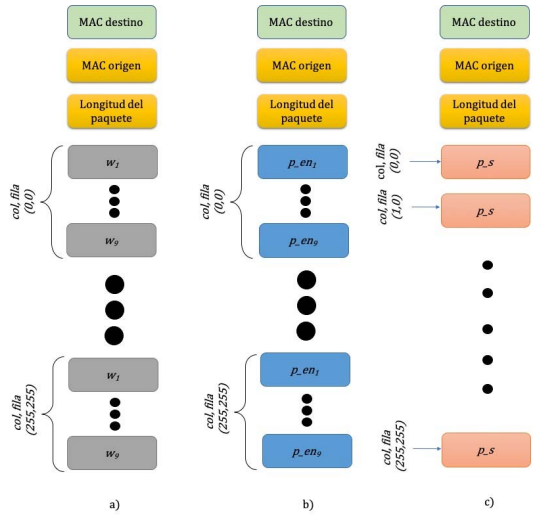


Fig. 4. Esquema general de la organización de los datos y parámetros de configuración en la trama Ethernet: a) trama con los pesos sinápticos (*w*), b) trama con el valor de los pixeles de la imagen de entrada. Ambas tramas son enviadas desde el computador hacia cada uno de los FPGAs en configuración multi-FPGA, y c) trama con los pixeles de salida, la cual es enviada desde cada FPGA hacia el computador.

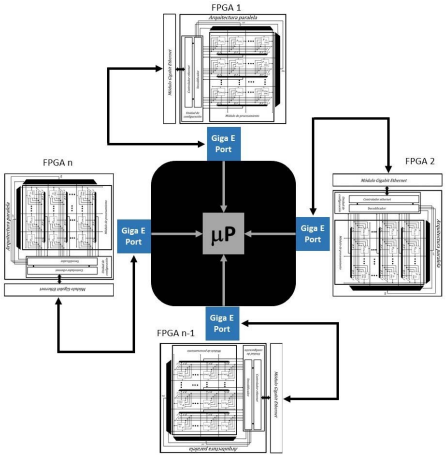


Fig. 5. Configuración multi-FPGA.

B. Implementación de la Aarquitectura Paralela en FPGA

En la presente versión la arquitectura paralela es implementada en una tarjeta de Xilinx. Esta tarjeta tiene embebido un FPGA Kirtex-7 KC705, el cual soporta una matriz de 65,536 procesadores. Por lo tanto, un solo FPGA puede soportar hasta 65,536 neuronas σ_{conv} para procesar una imagen de hasta 256x256 pixeles en paralelo. La implementación de esta cantidad de neuronas en el FPGA Kintex-7 requiere de 66% LUTs y de 37% registros para realizar la convolución de una imagen de 256x256 pixeles con un kernel (3x3), para el caso de un kernel 5x5 y una imagen de 256x256 se requiere de alrededor de 84% de LUTs y 56% de registros. Cabe mencionar que la estrategia propuesta no necesita de un gran ancho de banda de la memoria, y no

requiere de multiplicadores DSPs comparado con otras propuestas [5,6], lo cual es muy valioso para circuitos neuromórficos implementados en FPGA.

En el caso de requerir un mayor número de neuronas, *n* FPGAs se pueden conectar a través de buses Gb Ethernet a un computador para configurar y visualizar en la pantalla las imágenes filtradas, como se muestra en la Fig. 5. La presente configuración multi-FPGA permite conectar hasta 4 FPGAs dado que nuestro computador tiene una tarjeta de red de 4 puertos Gb Ethernet. Sin embargo, se podría conectar un mayor número de FPGAs mediante el uso de un switch.

V. Resultados

Esta sección muestra un ejemplo arbitrario usando la red neuronal propuesta, la cual realiza el cómputo de dos algoritmos de filtrado (filtro promedio y filtro Laplaciano-Gausiano), para procesar una imagen de 256x256 en escala de grises de 8 bits. Para realizar el cómputo del filtro promedio, utilizamos un kernel de 3x3, el cual contiene coeficientes positivos (*w*). Por otra parte, utilizamos un kernel 5x5 para realizar el cómputo del filtro Laplaciano-Gausiano, el cual tiene coeficientes (*w*) positivos y negativos, como se muestra en la Fig. 6. Cabe señalar que en este trabajo no se considero el uso de cero-padding, el cual es comúnmente utilizado para rellenar los bordes de la imagen, dado que la convolución se realizó solo dentro de la imagen, por lo que hubo una reducción en el tamaño de la imagen filtrada. Por otra parte, se realizó la operación de convolución con stride = 1, es decir, se considero el peor de los casos dado que la operación de convolución se llevo a cabo entre pixeles consecutivos. Para comprobar la escalabilidad y configurabilidad del sistema propuesto se habilitó la configuración multi-FPGA, por lo que utilizamos dos FPGAs para realizar el cómputo de los dos algoritmos de filtrado espacial de manera paralela.

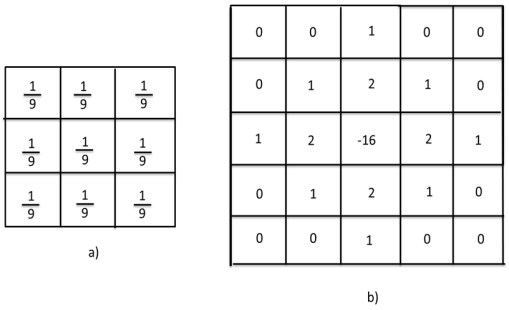


Fig. 6. Dos tipos de máscaras: a) Promedio y b) Laplaciano-Gaussiano.

Fig. 7. Respuesta de los filtros: a) Original, b) Promedio y c) Laplaciano-Gaussiano.

La Figura 7 muestra la respuesta de los filtros especiales de imágenes (promedio y Laplaciano-Gaussiano). Cada ejemplo arbitrario utiliza 65,536 neuronas para realizar el computo de los filtros espaciales en paralelo de imágenes de hasta 256x256 pixeles.

El tiempo de procesamiento de cada algoritmo es de 1.28 µs. Este tiempo de procesamiento se obtuvo multiplicando el valor máximo del pixel 2^n, donde $n=8$ define la resolución de la imagen en bits, por el reloj del sistema (5 ns). Además de este tiempo de procesamiento se ha calculado el tiempo que se requiere para enviar las tramas Ethernet desde el ordenador hacia el FPGA y viceversa. Para obtener este tiempo, primeramente, se ha calculado el número total de bits N_{b_T}, los cuales se requieren para enviar todos los pixeles de la imagen a la matriz de procesadores, mediante la siguiente ecuación:

$$N_{b_T} = \left(N_{pi} \times N_{bp}\right) \times N_{pro} \tag{4}$$

Donde: N_{pi} es el número de pixeles en cada procesador, N_{bp} es la resolución en bits de cada pixel y N_{pro} es el número total de procesadores de la matriz. Para este caso, $N_{pi} = 9$, $N_{bp} = 8$ y $N_{pro} = 65{,}536$. Sustituyendo estos valores en la ecuación 4, el número total $N_{b_T} = 4{,}718{,}592$ bits. Considerando que una trama Ethernet puede contener hasta N_b =12,144 bits = 1,518 bytes * 8. Por lo tanto, es necesario calcular el número total de tramas (N_{TE}) requeridas para enviar 4,718,592 bits, con la siguiente ecuación:

$$N_{TE} = \left(N_{b_T}/N_b\right) \tag{5}$$

En este caso el número total de tramas $N_{TE} = 389$. Por otra parte, es importante considerar que utilizamos una velocidad de transmisión de 1 Gigabit Ethernet. Por lo tanto, en cada segundo se envían N_T = 82,345 tramas. Para obtener el tiempo que se requiere por una trama (T_T) se utilizó la siguiente ecuación:

$$T_T = 1 \,/\, N_T \tag{6}$$

Sustituyendo N_T en la ecuación 6, cada trama se envía en un tiempo de 12.144 µs. Por lo tanto, se requiere de 4.7 ms para enviar 389 tramas desde el computador hacia el FPGA ya sea para enviar los pixeles o los pesos sinápticos. Para recibir la imagen filtrada se requiere de 524 µs dado que solo se envía un pixel de salida por cada procesador. Esto representa 1/9 parte del tiempo de envío de los pixeles de entrada. En conclusión, el tiempo total de procesamiento requerido para enviar los pesos sinápticos (4.7 ms), pixeles de entrada (4.7 ms), filtrado (1.28 µs) y envio de los pixeles de salida (524 µs) es de 9.9 ms. Por lo tanto, nuestra propuesta puede ser vista como una potencial solución en el procesado de video dado que la resolución temporal entre secuencia de imágenes es de 33 ms = 1 s / 30 secuencias de imágenes. Evidentemente, se puede mejorar este tiempo de procesamiento mediante el uso de multiples puertos Gb Ethernet en paralelo dado que la

adquisición de los datos en la presente arquietctura se realiza mediante un solo puerto Gb Ethernet, lo que disminuye considerablemente el rendimiento de la arquitectura paralela.

Además nosotros programamos dos filtros (promedio y Laplaciano-Gausiano) utilizando sus modelos matemáticos convencionales en un CPU y una GPU para comparar el rendimiento de nuestra propuesta, la cual está basada en pulsos o spikes, con respecto a los sistemas binarios convencionales. En particular, los modelos matemáticos convencionales fueron programados en C y simulados en un servidor con un procesador Xeon E5-2630 de 6 núcleos, a 2.6 GHz y con una memoria de 64 GB RAM. De igual forma, estos algoritmos fueron programados en C y simulados en una tarjeta NVIDIA GeForce GTX 1080 2560 CUDA, a 1.733 GHz con 8GB GDDR5X. La Tabla I muestra el tiempo de procesamiento de cada uno de los algoritmos implementados y simulados en sus respectivos dispositivos, considerando que la imagen se encuentra almacenada en la memoria de cada dispositivo con el fin de hacer una comparación coherente y efectiva. Para realizar la comparativa se considero una imagen de 256x256 con una resolución de 8 bits. Como se puede observar, la arquitectura propuesta es 34 y 54 veces más rápida que el software simulado en el servidor y 6 y 12 veces más rápido que los resultados obtenidos con la GPU. Por lo tanto, nuestra propuesta exibe un mejor rendimiento en comparación con el rendimiento de un GPU a pesar de que este último dispositivo contiene más de mil procesadores y un sistema de reloj de mayor frecuencia comparado con el reloj del FPGA. Nos hemos dado cuenta que varios factores limitan el rendimiento de la GPU. Estos factores están relacionados con el número de hilos, la transferencia de datos en memoria y el cálculo de operaciones en punto flotante.

TABLA I
COMPARACIÓN EN TÉRMINOS DE VELOCIDAD DE PROCESAMIENTO ENTRE UN SERVIDOR, GPU Y EL SISTEMA NEURONAL BASADO EN FPGA

Dispositivo	Tipo de filtro	
	Promedio (µs)	Laplaciano-Gaussiano (µs)
FPGA	1.2	1.2
GPU	8.2	14.2
Servidor	34	54

VI. DISCUSIÓN

Actualmente, varios autores han desarrollado herramientas de procesamiento de imágenes utilizando arquitecturas paralelas y otros basados en el comportamiento neuronal y su implementación en FPGAs. Por ejemplo, Yangjie Qi *et al*. [5] desarrollaron una arquitectura neuromórfica paralela con una comunicación interna entre los núcleos de procesamiento mediante el uso de enrutadores [5]**.** La principal diferencia entre el sistema neuromórfico existente y el propuesto en este trabajo esta relacionada con el uso de multiplicadores, dado que la propuesta de Yangjie Qi et al. [5] utiliza multiplicadores, lo que limita la implementación de un gran número de núcleos de procesamiento dado que cada núcleo usa un multiplicador y el FPGA tiene un número limitado de estos circuitos. Nuestra arquitectura no utiliza multiplicadores, por lo que, se puede implementar un gran número de procesadores.

La Tabla II muestra la comparación entre la arquitectura neuromórfica presentada por Yangjie Qi *et al.* [5] y nuestra propuesta en términos de consumo de área, rendimiento y consumo de potencia.

TABLA II
COMPARACIÓN ENTRE LA ARQUITECTURA NEUROMÓRFICA EXISTENTE [5] Y NUESTRA PROPUESTA

Autor	Filtro	Núcleos	Neuronas por núcleo
Este trabajo	Pasa altas	65,536	1
Qi *et al* [5].	Pasa altas	9	16
	Sinapsis por neurona	Throughput (Millones de pixeles/segundo)	Consumo de potencia
Este trabajo	9, 25	54613	0.98 – 1.56 mW
Qi *et al* [5].	512 Registros	0.98 Multiplicadores	No disponible LUTS
Este trabajo	228256 (56%)	0	171192 (84%)
Qi *et al* [5].	9896 (8%)	288 (54%)	17531 (15%)

Como se puede observar en la Tabla II, nuestra propuesta tiene un mayor consumo de área en comparación con la arquitectura de Yangjie Qi *et al.* [5] debido a que nuestra propuesta es una implementación totalmente paralela. Sin embargo, el número de multiplicadores en arquitecturas avanzadas FPGA es limitado. Esto confirma que los FPGAs pueden soportar un número reducido de núcleos de procesamiento para procesar imágenes de gran tamaño y resolución en paralelo tal como sucede en la arquitectura de Yangjie Qi *et al.* [5]. Cabe mencionar que la arquitectura de Yangjie Qi no puede ser utilizada para procesamiento de video puesto que una imagen de hasta 1024x960 pixeles se procesa en un segundo. Mientras que nuestra propuesta podría procesar este tamaño de imagen en microsegundos, puesto que cada FPGA puede procesar un segmento de la imagen de hasta 256x256 y varios segmentos serían procesados en paralelo en la configuración multi-FPGA, lo que es significativamente útil para el procesamiento de video. Aparte de la propuesta de Yangjie Qi et al. [5], otros autores han propuesto arquitecturas hardware implementadas en FPGA. Por ejemplo, Hasan [6] realiza una implementación de una arquitectura multiprocesador, la cual comparamos con nuestra propuesta para realizar una comparación coherente. Como se puede ver en la Tabla III, nuestra propuesta consume una mayor área pero menos potencia. Además, la velocidad de procesamiento es significativa comparada con la de Hasan [6].

VII. CONCLUSIONES

Este trabajo presenta una arquitectura paralela escalable y configurable para realizar el cómputo de filtrado espacial de imágenes de alta resolución y gran tamaño. Las contribuciones de este trabajo son:

-Propuesta de una neurona con reglas simples, con pesos en las sinapsis y con codificación unaria, la cual realiza cuatro operaciones aritméticas (suma, resta, multiplicación y

división) en paralelo.

-Diseño de un procesador configurable para procesar diferentes algoritmos espaciales de filtrado.

-Diseño de un sistema Multi-FPGA para procesar imágenes de gran tamaño.

TABLA III
COMPARACIÓN ENTRE ARQUITECTURAS PARALELAS EN FPGA Y ESTE TRABAJO

Autor	Dispositivo	Filtro	Throughput (Millones de pixeles/segundo)
Este trabajo	Kintex-7	Laplaciano	54613
Hasan [6]	Virtex 6	Laplaciano	84
	Numero de procesadores	LUTS	Registros
Este trabajo	65536	171192	228256
Hasan [6]	9	6392	9616
	Consumo de potencia		
Este trabajo	0.96 – 1.56 mW		
Hasan [6]	212 mW		

Además, este trabajo puede ser visto como un primer paso en el desarrollo de nuevos esquemas de procesamiento neuronal tipo P spiking y su inclusión en aplicaciones de procesamiento de imágenes en tiempo real. Nuestros resultados demuestran que es altamente factible simular nuevos modelos neuronales tipo P spiking en los actuales sistemas embebidos digitales para crear arquitecturas altamente paralelas, las cuales pueden ser usadas para dichas aplicaciones y mejorar sus prestaciones y rendimiento. En general, la propuesta presenta características de escalabilidad y multi-modelo, con la finalidad de crear una herramienta que pueda ser utilizada en aplicaciones avanzadas en procesado de imágenes en tiempo real dado que la arquitectura requiere de 9.9 ms, en una imagen de hasta 256x256 pixeles, independientemente de cualquier filtro espacial, lo que hace factible su uso en procesamiento de video, donde la resolución temporal minima entre cuadros es de 33 ms.

REFERENCIAS

[1] M. Ionescu, G. Paun, T. Yokomori, “Spiking neural p systems”, *Fundamenta informaticae*, 71 (2), pp. 279-308, 2006.

[2] D. Díaz-Pernil, F. Peña-Cantillana, and M. A. Gutiérrez-Naranjo, “A parallel algorithm for skeletonizing images by using spiking neural P systems,” *Neurocomputing*, vol. 115, pp. 81–91, 2013.

[3] T. Song, S. Pang, S. Hao, A. Rodríguez-Patón, and P. Zheng, “A parallel image skeletonizing method using spiking neural P systems with weights,” *Neural Processing Letters*, pp. 1–18, 2018.

[4] J. Ke, A. Sowmya, Y. Guo, T. Bednarz, M. Buckley, “Efficient gpu computing framework of cloud filtering in remotely sensed image processing”, *Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1-8, 2016.

[5] Y. Qi, B. Zhang, T. M. Taha, H. Chen and R. Hasan, "FPGA design of a multicore neuromorphic processing system," NAECON 2014 - IEEE National Aerospace and Electronics Conference, Dayton, OH, pp. 255-258, 2014.

[6] S. Hasan, "Performance-Aware Architectures for Parallel 4D Color fMRI Filtering Algorithm: A Complete Performance Indices Package," *in IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 2116-2129, July 1 2016.

[7] G. Zhang, Q. Yang, and L. Pan, “A Bibliography of Technological Applications of Spiking Neural P Systems”, *Bulletin Webpage*, p. 69.

[8] T. Song, Q. Zou, X. Liu, X. Zeng, “Asynchronous spiking neural P systems with rules on synapses”, *Neurocomputing*, 151, pp. 1439-1445, 2015.

[9] X. Zeng, L. Xu, X. Liu, L. Pan, “On languages generated by spiking neural P systems with weights”, Information Sciences, 278, pp. 423-433, 2014.

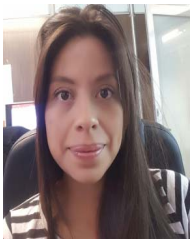
[10] T. Frias, C. Diaz, G. Sanchez, G. Garcia, G. Avalos and H. Perez, "Four Single Neuron Arithmetic Circuits based on SN P Systems with Dendritic Behavior, Astrocyte-like control and rules on the synapses," in *IEEE Latin America Transactions*, vol. 16, no. 1, pp. 38-45, Jan. 2018.

Marco Abarca recived the M.S. degree and the PhD at Instituto Politecnico Nacional, Mexico in 2013 and 2019, respectively. He is currently a Porfessor in the Universidad Politecnica de Texcoco. His research interest range from signal and image processing to bio-inspired neural processing.

Giovanny Sánchez received the M.S. degree at Instituto Politecnico Nacional, Mexico, in 2008, and the Ph.D. degree at Universitat Politecnica de Catalunya, Spain, in 2014. His research is focused on developing early auditory neural processing systems, neural-based cryptosystems in neuromorphic hardware, image and audio processing. Currently, he is an Associate Professor in the Instituto Politecnico Nacional, Mexico

Jorge Luis García Peña received the BS degree at Instituto Politecnico Nacional, Mexico, in 2018. Currently, he is a Master student at the Instituto Politecnico Nacional, Mexico. His research interest is linked to robotics and neuromorphic circuits. He participated in international competitions, such as INFOMATRIX, ROBOMATRIX. He won first place in both competitions.

Juan Gerardo Avalos received in 2008 the M.Sc. degree in microelectronics and the Ph.D. degree in electronics and communications engineering in 2014 from the National Polytechnic Institute. His current research interests are signal processing and adaptive filtering applied to speech, audio, acoustics and early auditory and vision neural processing.



Thania Frias recived the M.S. degree and the PhD degree at Instituto Politecnico Nacional, Mexico in 2013 and 2019, respectively. Her research interest is linked to the development of arithmetic circuits implemented in neuromorphic hardware.

Karina Toscano received the BS degree in Computer Science Engineering and the PhD degree in Electronic and Communications in 1999 and 2005, respectively, from the National Polytechnic Institute, Mexico City. She is a member of the National Researchers System of Mexico. Her principal research interest is related to artificial neural networks

Hector Perez-Meana received the M.S. degree from the University of Electro-Communications, Tokyo Japan, a Ph. D. degree in Electrical Engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989. His principal research interests are adaptive systems, image processing, pattern recognition and information security