TeCEO
Revista Tecnología, Ciencia y Estudios Organizacionales

TeCEO, 6(12), 2024.
Revista Tecnología, Ciencia y Estudios Organizacionales / ISSN: 2594-1410
https://doi.org/10.56913/teceo.6.12.235-246

Sistema basado en hardware reconfigurable para el cómputo paralelo de la operación convolución en imágenes.

Hardware structure of parallelism in image processing convolution operation

- **Melesio Reyes-Pérez,** es estudiante de la maestría en ciencias en Ingeniería Electrónica en el Tecnológico Nacional de México en Celaya, Gto. (México) (m2303054@itcelaya.edu.mx), Licenciado.
- **Moisés Arredondo-Velázquez,** Facultad de Ciencias en Física y Matemáticas, Benemérita Universidad Autónoma de Puebla, en Puebla, Pue. (México) (https://orcid.org/0000-0003-0198-274X), Doctor.
- **Javier Díaz-Carmona,** Tecnológico Nacional de México en Celaya, Celaya, Gto. (México) (https://orcid.org/0000-0002-7363-3349), Doctor.

Resumen: En el presente documento se analiza una estructura propuesta en hardware reconfigurable, más específico en FPGA, para realizar de manera paralela la operación de convolución aplicada en imágenes, con el propósito de optimizar el procesamiento de imágenes de redes neuronales de convolución, lo cual se le conoce como un acelerador hardware. La estructura propuesta consta de dos etapas principales: el Line Buffer que se centra en llenar los datos de la imagen de entrada al sistema y el bloque de convolución o de bloques MAC, que se encarga de hacer la convolución multiplicando la máscara de convolución creada en el Line Buffer con el filtro que se desee aplicar. Adicionalmente, se presenta el pretratamiento que se le debe aplicar a las imágenes de entrada para que puedan ser analizadas y procesadas correctamente por la estructura hardware, la cual consiste básicamente en obtener sus valores en binario en formato Q1.8, un bit para la parte entera y ocho bits para la parte fraccionaria. Los resultados del análisis de la investigación muestran que existen múltiples formas de realizar una optimización basada en el paralelismo. Siendo dos las principales: el paralelismo al momento de aplicar un filtro consistente en poder aplicar más de un filtro a la vez, y el paralelismo en operaciones, que de igual manera consiste en el filtro, pero al momento de hacer la multiplicación de la máscara de convolución con el filtro se multiplican diferentes elementos del mismo filtro con la máscara de convolución generada.

Palabras clave: Convolución, Red Neuronal, RNC, FPGA, Hardware, Binario, Paralelismo, Bloques MAC, Mascara de Convolución, Matriz, Line Buffer, Filtro.

Cómo citar: Reyes Pérez, M.; Arredondo Velázquez, M.; Díaz-Carmona, J. (2022). Sistema basado en hardware reconfigurable para el cómputo paralelo de la operación convolución en imágenes**.** *Tecnología, Ciencia y Estudios Organizacionales*, 6(12), pp. 235–246. https://doi.org/10.56913/teceo.6.12.235-246

Recepción: 30-09-24
Aprobación: 23-10-24

Abstract: This paper analyzes what is a more specific hardware structure in FPGA of parallelism in the convolution operation, in order to optimize the image processing of neural networks, which would be considered as a hardware accelerator, the proposed structure consists of two main cores, the Line Buffer that focuses on filling the data from the input image to the system and the convolution block or MAC blocks, which is responsible for making the convolution by multiplying the convolution mask created in the Line Buffer with the filter to be applied, additionally shows the pre-treatment that must be given to the input images so that they can be analyzed and processed correctly by the hardware structure, which is basically to obtain their values in binary format Q1,8, which consists of an integer bit and 8 bits in fractional unsigned. The results of the research analysis show that there are multiple ways to perform an optimization based on parallelism, the main ones obtained are parallelism when applying a filter, which consists of being able to apply more than one filter at a time. And parallelism in operations, which likewise consists of the filter, but at the time of multiplying the convolution mask with the filter, different elements of the same filter are multiplied with the convolution mask generated.

Keywords: Convolution, Neural Network, CNN, FPGA, Hardware, Binary, Parallelism, MAC's Block, Convolution Mask, Matrix, Line Buffer, Filter.

Introducción

Las redes neuronales de convolución (RNCs) son uno de los algoritmos más utilizados para la clasificación y detección de objetos en imágenes (LeCun et al., 2015). Estos algoritmos son modelos matemáticos biológicamente inspirados cuya alta eficiencia ha cambiado el paradigma de diseño en el área de procesamiento digital de imágenes. Anteriormente los algoritmos de clasificación y detección requerían de una selección manual de las etapas de procesamiento, por lo que la experiencia del diseñador era un factor determinante (Sve et al., 017). Ahora, las RNCs se someten a procesos de entrenamiento automático en donde se obtienen los coeficientes (o pesos) que determinan la habilidad de la RNCs para clasificar imágenes. Aunque el entrenamiento de las redes neuronales es muy importante, de forma muy general se tiende a focalizar su entrenamiento y análisis en una capa o sección de software, cuando la operación se basa o tiene mucho de los recursos físicos con los que cuenta, o mejor dicho, el hardware disponible para llevar a cabo la tarea. Por lo cual la optimización de algoritmos focalizados en hardware es indispensable al momento de analizar una red neuronal en operación o en su etapa de entrenamiento. En la actualidad, la operación de convolución es un tema que ha cobrado gran relevancia debido a las áreas en las que se utiliza, como el procesamiento de imágenes, las redes neuronales e incluso la inteligencia artificial. Debido a su relevancia clave en el procesamiento de datos, existen actualmente varios trabajos de investigación que buscan diversos métodos computacionales para optimizar y realizar eficientemente el algoritmo de convolución

En los últimos años se han realizado investigaciones y propuestas en este sentido para lograr un procesamiento eficiente de los datos cuando se ejecuta una RNC. Se han realizado avances a través de diferentes enfoques propuestos en el procesamiento paralelo por convolución. Algunas investigaciones se han centrado en la optimización de algoritmos. Esto incluye técnicas como la convolución Winograd, la convolución basada en la transformada rápida de Fourier (FFT) y la convolución separable en profundidad (An et al., 2023). Otro enfoque de la investigación se basa en hardware de aceleración especializado, en el que se están diseñando y mejorando unidades de procesamiento especializadas para acelerar la convolución paralela. Esto incluye desarrollos en unidades de procesamiento sensorial (TPU) y unidades de procesamiento neural (NPU) (Huang et al., 2023). Aunque el algoritmo de convolución es donde se centra la mayor parte de la optimización de algunas investigaciones, la optimización del consumo energético se centra en reducir el consumo de energía de la convolución paralela. Se están explorando estrategias como la

cuantización de modelos, la compresión de redes neuronales y el diseño de hardware de bajo consumo para mejorar la eficiencia energética de los sistemas hardware que ejecutan operaciones de convolución paralela (Hong et al., 2023). Muchos otros investigadores se centran en aplicaciones generales, o más concretamente en aplicaciones FPGA del algoritmo de convolución, añadiendo y modificando las mismas etapas del algoritmo de diferentes maneras, modificando el Max pooling, añadiendo elementos hardware, e incluso optimizando los recursos de memoria del algoritmo y los elementos de memoria del componente hardware. Por lo tanto, se puede considerar que el tema de la convolución paralela es un tema de gran actualidad y que merece la pena desarrollar más propuestas de optimización por el gran alcance y aplicaciones que se pueden desarrollar en el futuro.

Otro concepto necesario de mencionar es lo que es un acelerador hardware, lo cual es un dispositivo o circuito especializado diseñado para realizar ciertas tareas más rápido y de manera más eficiente que una CPU general. Los aceleradores de hardware están dedicados a mejorar el rendimiento de aplicaciones específicas, descargando ciertas tareas que consumen muchos recursos, como la computación gráfica, el procesamiento de señales, o la inteligencia artificial. Existen múltiples tipos de aceleradores hardware, los más comunes que existen en la actualidad son;

GPU (Unidad de procesamiento gráfico): Utilizadas principalmente en el cálculo bruto de gráficos frecuentemente empleadas en programas de diseño 3D, animaciones o para el diseño de videojuegos, pero también en el procesamiento paralelo intensivo, como el aprendizaje automático, tanto para aplicaciones como para el desarrollo de inteligencia artificial por hardware.

Algunas de las plataformas de implementación de las RNCs son: FPGA (Field Programmable Gate Arrays): hardware reconfigurable para realizar diversas funciones específicas, cuyo diseño e implementación realiza a través de lenguajes de descripción de hardware (VHDL o Verilog), ASIC (Application-Specific Integrated Circuit): circuitos integrados diseñados para una aplicación en particular, TPU (Tensor Processing Unit): aceleradores diseñados por Google para mejorar el rendimiento de los modelos de machine learning, específicamente en redes neuronales.

Como se puede observar algunos aceleradores hardware, son de uso muy específico o particular, o incluso su desarrollo tiende a ser particular y privado, por tal motivo el desarrollo e investigación se basa en FPGA’s ya que son hardware mucho más accesible para el público en general, en su gran mayoría. Al momento de decidir por un acelerador hardware debemos considerar las razones o motivos para ello. Algunas de las ventajas que proporciona el uso de un acelerador hardware, es un incremento sustancial en el rendimiento ya que puede realizar operaciones especializadas dedicando hardware en específico a realizar las operaciones deseadas, con esto viene también un gran ahorro en el consumo energético, ya que al tener dividido específicamente su hardware, se optimizan las tareas de forma más eficiente que con un procesador general, con lo que al hacer tareas concretas puede utilizar mucho menos consumo energético, Adicionalmente tenemos que un CPU al tener que realizar múltiples operaciones con la misma prioridad de tiempo, al querer realizar una tarrea en especifica el uso o del CPU estará siempre más elevado a lo que realmente seria en el caso de un FPGA con la misma tarea, ya que no necesita estar considerando tareas secundarias al momento de entrar en operación. También puede dar el caso en el que en una placa convergen lo que es un CPU y un FPGA como en el caso de las placas de desarrollo altera, en donde al delegar tareas complejas a aceleradores (FPGA) la carga del CPU cae y puede centrarse en otras tareas (Armeniakos et al., 2022), (Lavaine, 2024).

Método

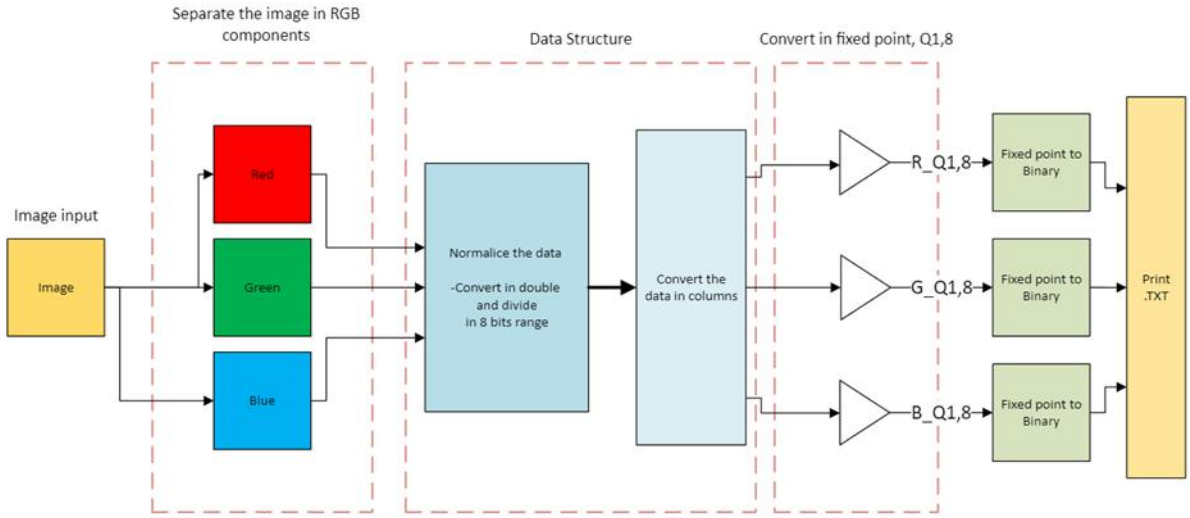
La optimización de redes neuronales es un concepto ampliamente requerido he investigado, más aún en los últimos años, ya que muestra ser la base de muchas tecnologías e investigaciones que parten de ella, tales como, inteligencia artificial, machine learning, y reconocimiento de imágenes (objetos). Debido a la necesidad de optimizar se tienen diferentes propuestas o metodologías para lograr mejorar los tiempos de ejecución, recursos energéticos, o en general costos, el enfoque depende de la necesidad a cubrir, en este caso como se busca la aceleración de una red de procesamiento de imágenes, en base a aceleradores de hardware. Lo primero requerido en todo procesamiento de imágenes es una etapa de preprocesamiento de la imagen de entrada previa a la estructura hardware (Ma et al., 2023).

Conversión de datos de imagen a binario:

El primer paso es hacer un preprocesamiento de la imagen de entrada, que consiste básicamente en transformar la información de la imagen a un formato que un FPGA pueda leer y procesar adecuadamente, el tipo de datos requerido es binario, por lo que es necesario convertir cada pixel de una imagen es un valor en binario de punto fijo (Zhang, 2024)

En la figura 1 se muestran las etapas propuestas para realizar la conversión binaria de los datos de la imagen. El algoritmo fue implementado MATLAB (Cazacu, 2021), (Gao, 2013).

Figura 1. *Descripción del proceso de conversión de una imagen a datos en binario*



El primer paso es separar la imagen a sus componentes RGB, lo cual en sí resulta en tres matrices diferentes. Posteriormente en la etapa de estructura los datos de cada matriz se convierten a formato doble (número de punto flotante de doble precisión que se almacenan en 64 bits) para posteriormente ser normalizados mediante su división entre 255 (8 bits). Dentro de esta misma etapa las matrices de datos son convertidos a columnas para poder ser introducidos de forma serial al FPGA, para ello todos los datos normalizados de la imagen son convertidos en una columna vertical. La tercera etapa consiste en convertir la columna de datos normalizados al formato de punto fijo, siendo en este caso el formato Q1.8 donde los valores son sin signo con un bit para la

 ISSN: 2594-1410

parte entera y ocho bits para la parte fraccionaria. Como última etapa en la conversión es escribir todos los valores de la columna en un archivo de texto para poder ser procesada a futuro, porque al final de este proceso obtenemos como resultado tres archivos de texto por imagen.

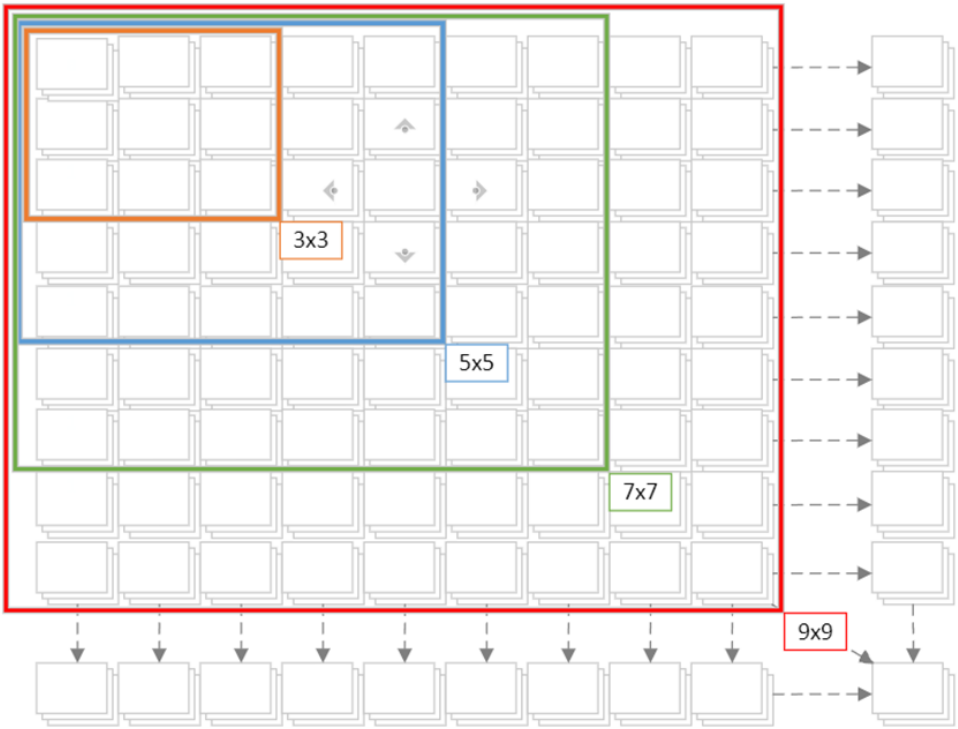
Matriz De Una Imagen

Antes de explicar la propuesta de convolución, es importante destacar cómo se consideran los datos de entrada, o más concretamente, cómo se considera la información de una imagen o serie de imágenes. Actualmente es de conocimiento común que las imágenes pueden tener diferentes tamaños, en altura y longitud, la resolución multimedia más común es 1920 x 1080, que simplemente se refiere a su tamaño en número de píxeles, más explícitamente sería, 1920 píxeles de largo por 1080 píxeles de alto. Entendiendo cómo se dimensiona una imagen es posible extrapolar esa percepción a una matriz de 1920 columnas y 1080 filas, a efectos explicativos es mejor considerar algo mucho más manejable (Gonzalez & Woods, 2018).

Entrando más en conceptos de lo que es la convolución, se tiene el concepto de máscara de convolución, o a veces también llamado filtro, que solo se refiere a una pequeña matriz que hará el barrido sobre la imagen de entrada (fig. 2). Estas matrices comúnmente son matrices cuadradas como por ejemplo matrices de 3x3, 5x5, 7x7 y 9x9, podrían extenderse a matrices mucho más grandes, pero a efectos de optimización y paralelismo en el algoritmo de operación se consideran éstas como las más comunes (Smith & Doe, 2020), (Szeliski, 2010).

Esa matriz cuadrada hará un barrido de izquierda a derecha columna a columna y fila a fila, sin embargo, no siempre será tan exacto ya que ese recorrido puede modificarse en función de lo especificado en la convolución.

Figura 2. *Descripción general de una matriz general y de filtros*



Concepto de Paralelismo en Convolución

El concepto de realizar operaciones paralelas es bastante sencillo de entender, se trata simplemente de realizar múltiples operaciones de esta o distinta naturaleza al mismo tiempo. Sin embargo, cuando queremos aplicar este concepto a operaciones hardware, nos encontramos con el hecho de que disponemos de elementos físicos limitados, por lo que el concepto de paralelismo es bastante difícil de implementar literalmente. Por ello, en los últimos años se han utilizado técnicas de optimización en procesos y operaciones lógicas en FPGA's y DSP's, una de las más populares o conocidas es la técnica pipeline, que centra su paralelismo en etapas desfasadas, por lo que optimiza mucho el proceso.

Teniendo en cuenta el fenómeno que se produce en la técnica pipeline, donde el objetivo es liberar elementos hardware para iniciar etapas de otro proceso, una pregunta importante sería qué pasaría si cambiamos el paralelismo de etapas, para liberar elementos hardware o sector, para realizar operaciones de diferente magnitud realmente en paralelo al mismo tiempo (Krizhevsky et al., 2017), (Hennessy & Patterson, 2019).

Estructura Propuesta FPGA

El primer paso en la estructura de la convolución es la entrada de los datos. La imagen de entrada se considera como una matriz de cualquier dimensión (fig. 2) pero en hardware no es posible transferir la información completa al mismo tiempo, entonces la imagen se separa en píxeles, y esos datos se introducen en una conjunción de flip-flops como bloques de memoria conformando un registro de desplazamiento numeroso, con el propósito de tomar cada paquete de datos de la imagen de entrada. Todo el proceso lleva un cierto tiempo de procesamiento y algunos recursos, cada DSP y FPGA tiene una velocidad de reloj diferente, por esa razón el tiempo medio de procesamiento que se considera para el análisis es de alrededor de 3 ciclos de reloj para ejecutar un comando o una acción en el proceso, con el fin de hacer cálculos posteriores del tiempo que tardara en procesar una imagen. La estructura en general está formada de dos elementos que se consideran el núcleo de todo el procesamiento el cual es el Line Buffer y el arreglo de bloques MAC, que es donde se lleva a cabo la operación de convolución (Chetlur et al., 2014), (Chen et al., 2015).

Line Buffer

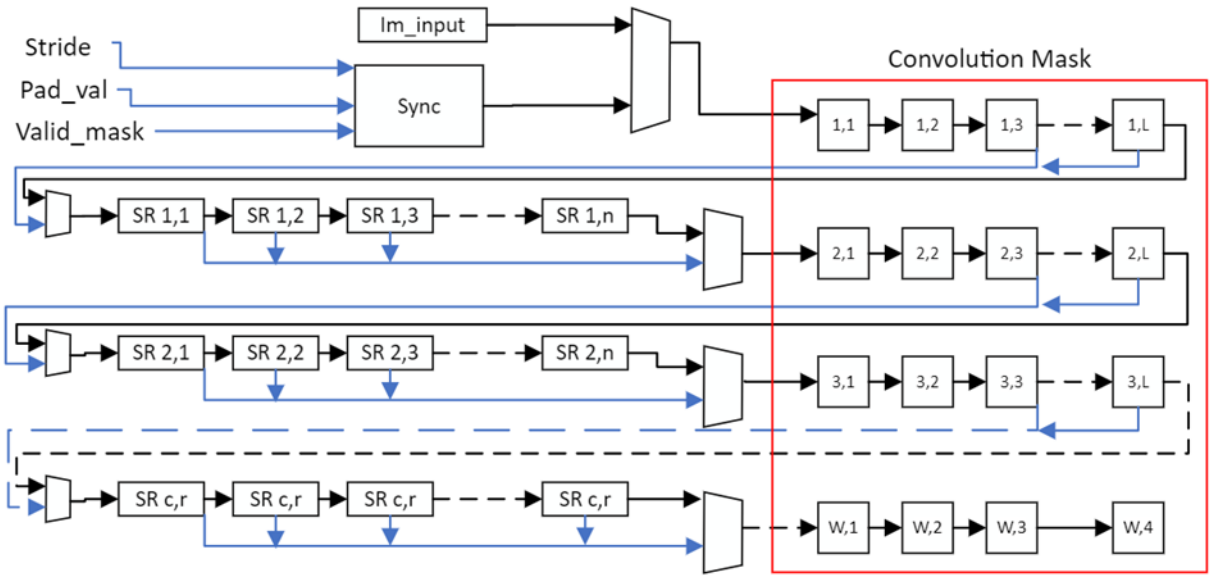
El line buffer es la sección dedicada al llenado de los datos de la imagen, lo primero a considerar es cada paquete de datos de la imagen, y cómo entra en la estructura propuesta. Los datos se llenan en los flip-flops y se recorren debido al uso del registro de desplazamiento, después de haber terminado de almacenar toda la información de la imagen de entrada, tenemos ciertos datos almacenados en los flip-flops, que en conjunto crean lo que para nosotros es la máscara de convolución de la imagen (Fig. 3). Cabe destacar que esta máscara de convolución tiene la característica de ser flexible, lo que significa que puede extenderse a ser aplicada en una capa de convolución de cualquier dimensión. Para que esta flexibilidad sea posible en la capa de convolución se puede observar en la figura 3, como hay unas conexiones de líneas punteadas, estas conexiones son derivaciones que se colocan en los terminales impares de los flip-flops, y en las salidas del registro de desplazamiento para que su valor sea fácil de modificar en cualquier momento.

En esta sección se obtiene como resultado lo que es la máscara de convolución, que es la matriz de datos a la cual se le aplicara la operación de convolución con los datos de otra matriz que la

 ISSN: 2594-1410

consideramos como el filtro, este proceso consiste en generar múltiples máscaras de convolución, una en cada siclo de reloj, lo cual haría una acción de barrido en todos los datos de la imagen que tengamos de entrada, hasta que ya no haya datos en la entrada. Posterior a esto los datos tendrán que pasar a la siguiente etapa, sin embargo, es importante tener en cuenta que los datos de esta primera etapa tienen que entrar en sincronía con los datos de la siguiente etapa para que no exista un desface y un futuro error al momento de recuperar los datos de la imagen al final del proceso.

Figura 3. *Diagrama de la estructura encargada de el llenado de los datos de la imagen de entrada*



Bloques MAC

Después de obtener la máscara de convolución, el siguiente paso es multiplicar la máscara de convolución con una matriz diferente, en otras palabras, las matrices de los filtros. Si se realiza la multiplicación por la matriz el resultado técnicamente será el resultado de la convolución de un filtro, esta operación se realiza en hardware mediante el concepto del bloque MAC (Multiplicar y Acumular). La estructura de propósito utiliza los bloques MAC para combinar la información de la máscara de convolución y la información del filtro (fig. 4). La diferencia y la razón de usar los bloques MAC es porque en hardware los datos de la matriz se seleccionan por separado, la matriz se toma fila por fila, y cada fila se toma dato por dato. A continuación, el primer elemento de los datos de convolución se multiplicará por el primer elemento de la matriz de filtro, luego por el segundo elemento, y así sucesivamente. Después de la multiplicación, los resultados se acumularán y formarán una nueva matriz de elementos MAC (Fig. 5).

Figura 4. *Diagrama de la estructura de generación de los bloques MAC*

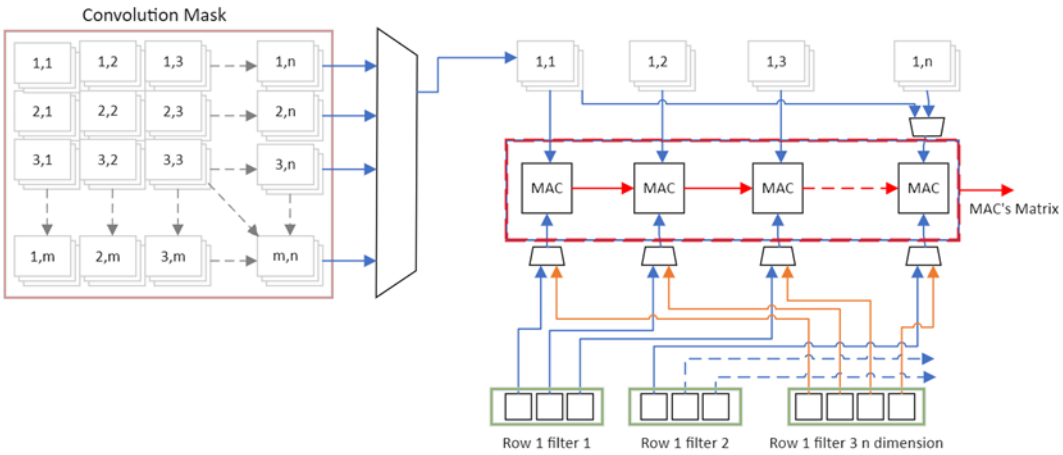
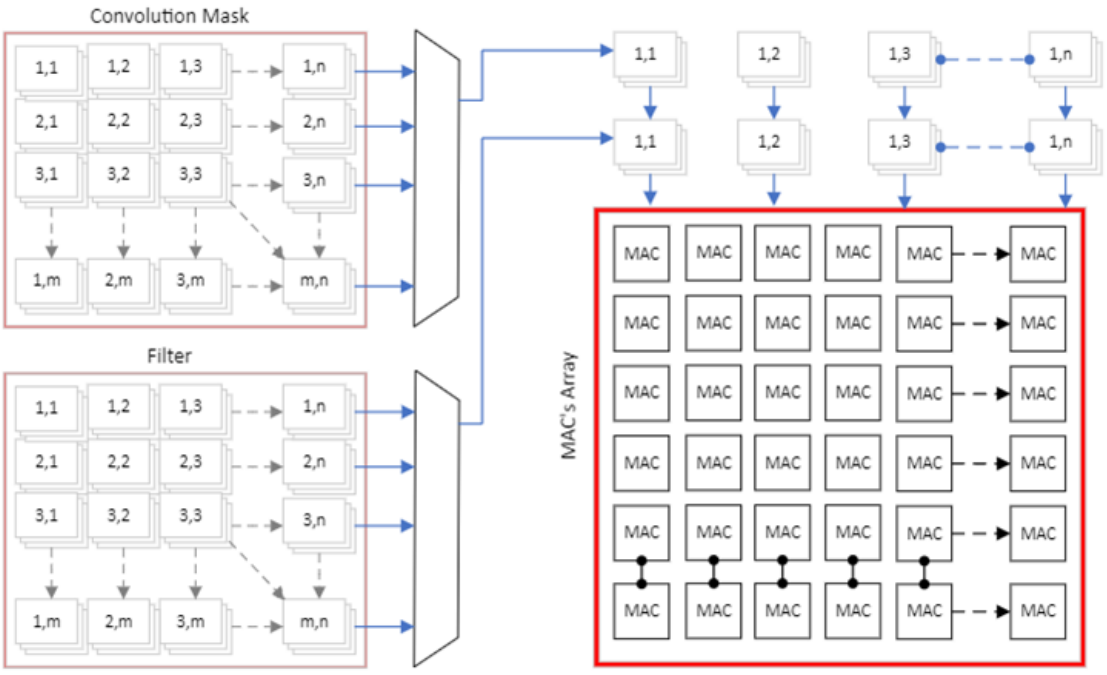


Figura 5. *Diagrama de la estructura de la generación de la matriz de bloques MAC*



Al momento de realizar la multiplicación del filtro con la máscara de convolución, el número de filtros aplicables depende del número de elementos en hardware, pero la dimensión del filtro debe ajustarse a la dimensión de la máscara de convolución. Pero como puede observarse en la figura 4 es posible aplicar múltiples filtros al mismo tiempo, a causa del sobrante de elementos hardware de manera que se puede diseñar un algoritmo capaz de utilizarlos para optimizar el proceso.

 ISSN: 2594-1410

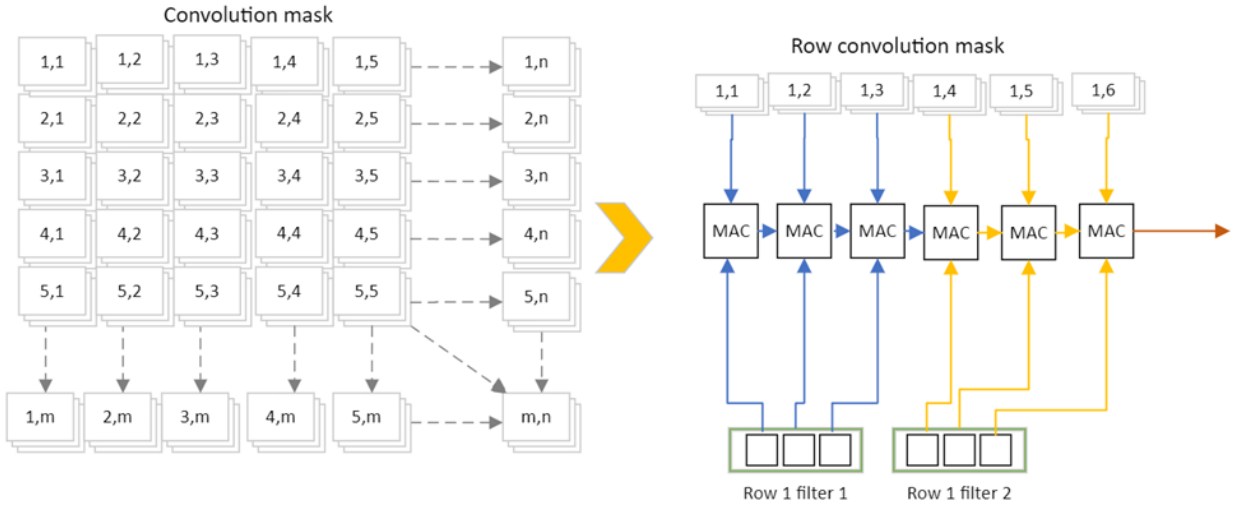
Resultados

Tras el análisis de la estructura propuesta para llevar a cabo la convolución y el procesamiento de una imagen en un FPGA, es posible determinar que, si buscamos optimizar el proceso, es necesario utilizar la flexibilidad que ofrece la operación de convolución y los elementos físicos al momento de considerar el hardware. Con ello surge el concepto de paralelismo, y si buscamos diferentes formas de paralelismo podemos destacar dos tipos de paralelismos que surgen naturalmente en la operación de convolución y al momento de crear la matriz de los bloques MAC, los cuales son: aplicar múltiples filtros o aplicar diferentes operaciones a la vez. Cabe resaltar que aún existen más tipos de paralelismos que se podrían aplicar para seguir optimizando el proceso.

Paralelismo en Filtros

Este tipo de paralelismo se refiere a la cantidad de filtros que se pueden aplicar a la máscara de convolución al mismo tiempo, retomando el concepto de la flexibilidad al crear la máscara de convolución, se tiene que esta capa puede ser muy grande o muy pequeña, esto da la opción de poder aplicar filtros no solo de la misma dimensión, sino que también filtros por ejemplo de 3x3 y 5x5 sí la máscara de convolución es de al menos 5x5 (fig. 6). Esto último esto se puede extrapolar a más y más filtros. Como siguiente etapa están los bloques MAC, ambas etapas están relacionadas ya que, si se aplican filtros en paralelo, es porque en la etapa de multiplicación se multiplica a la vez la fila de los filtros en paralelo con la fila de la máscara de convolución, y se puede ampliar la aplicación de filtros en paralelo dependiendo de los elementos hardware que aún se tienen disponibles.

Figura 6. *Diagrama que muestra la multiplicación de la máscara de convolución por elementos de dos filtros a la vez*

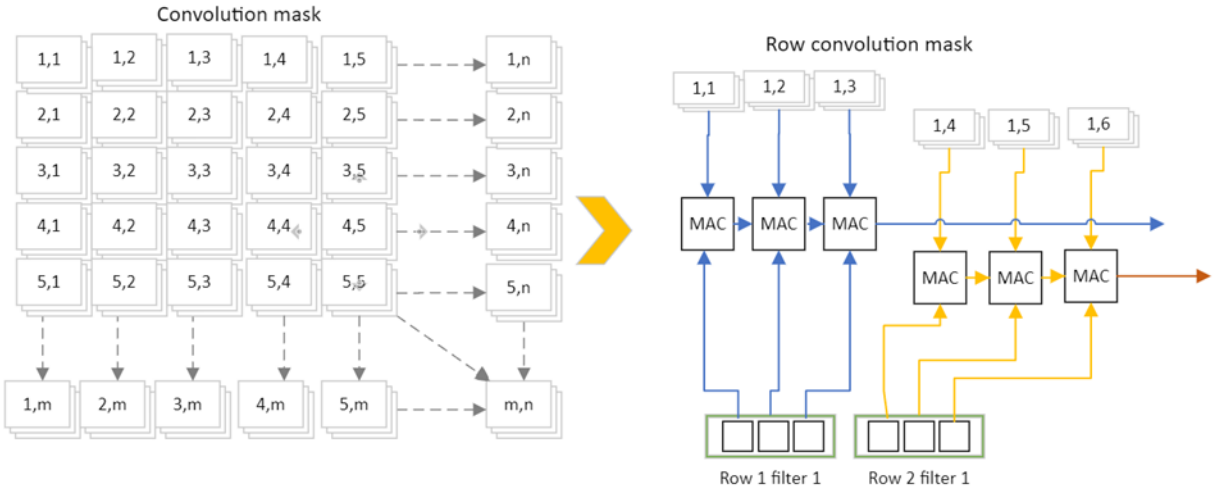


Esta forma de realizar la multiplicación de filtros se puede extrapolar a una dimensión superior y a la mezcla de múltiples tamaños de filtros, esto representa una reducción en la aplicación de múltiples filtros en una máscara de convolución. Sin embargo, hay que tener en cuenta que no es la única forma de paralelismo a la hora de realizar la convolución.

Paralelismo en Operaciones

Otra forma de paralelismo al calcular la convolución es en el instante de realizar las operaciones MAC, si se considera que se tiene una máscara de convolución de cualquier tamaño, y una cantidad de filtros a aplicar, entonces se puede centrar el paralelismo en las multiplicaciones de los elementos de un mismo filtro, con los elementos de la máscara de convolución, y como cada elemento de las matrices se pueden considerar casi como independientes, entonces se puede multiplicar la primera fila del filtro por la primera fila de la máscara de convolución, y al mismo tiempo podemos multiplicar la segunda fila del filtro con la segunda fila de la máscara de convolución (Fig. 7), y así sucesivamente hasta ocupar todos los elementos hardware disponibles hasta cubrir toda la información de la imagen multiplicada por el filtro.

Figura 7. *Diagrama que muestra la multiplicación de 2 filas del mismo filtro con la máscara de convolución*



Del mismo modo, este paralelismo en las operaciones puede extrapolarse a un filtro de mayor dimensión y a una máscara de convolución de mayor tamaño, para cubrir la mayor cantidad de información de la imagen de entrada en el menor tiempo posible para el filtro o filtros que se esté aplicando.

Discusión

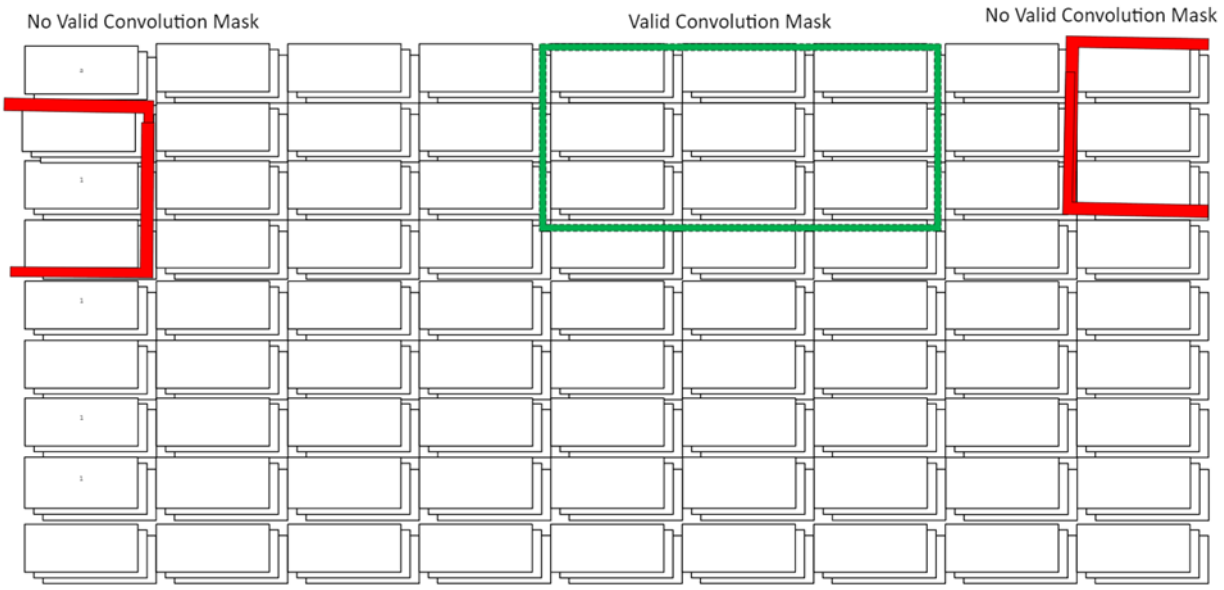
En la actualidad el tema de optimización de algoritmos en relación con las redes neuronales tiene demasiado desarrollo y distintos campos de enfoque, por lo que este trabajo no busca más que ofrecer una perspectiva de optimización referente al área del hardware, sin dejar de lado los demás campos y las investigaciones emergentes.

Las operaciones de convolución así como el llenado de datos en el Line buffer presentan un inconveniente que representa una oportunidad de mejora, los cuales son el control de datos en el Line buffer, el cual se encarga de verificar que los datos estén entrando correctamente, y que además presente un elemento extra de validación en las mascas de convolución, ya que analizando la forma en la que se generan las máscaras de convolución es posible darse cuenta que si no se limita el rango de creación de las máscaras de convolución, puede darse el caso en el que se generen máscaras de convolución con información de dos diferentes filas de datos de una imagen (Fig. 8)

 ISSN: 2594-1410

lo que entorpecería la reconstrucción de la imagen final, por lo que se necesita un bloque en la etapa del Line Buffer que pueda distinguir entre mascas de convolución validas y no validas.

Figura 8. *Diagrama que muestra ejemplos de mascara de convolución valida y no valida*



Tras analizar la estructura de control, es posible destacar de que puede suceder una desincronización de los datos entre el Line Buffer y la sección de convolución, por lo que es necesario tener un elemento que pueda transferir los datos entre ellos sin que se sobrepongan datos debido su flujo constante y los ciclos de reloj a los que obedezcan las instrucciones de la estructura.

Referencias

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. arXiv preprint arXiv:1703.09039. https://doi.org/10.48550/arXiv.1703.09039

An, F., Wang, L., & Zhou, X. (2023). A high performance reconfigurable hardware architecture for lightweight convolutional neural network. Electronics, 12(13), 2847. https://doi.org/10.3390/electronics12132847

Huang, J., Liu, X., Guo, T., & Zhao, Z. (2023). A high-performance FPGA-based depthwise separable convolution accelerator. Electronics, 12(7), 1571. https://doi.org/10.3390/electronics12071571

Hong, E., Choi, K.-A., & Joo, J. (2023). Efficient two-stage max-pooling engines for an FPGA-based convolutional neural network. Electronics, 12(19), 4043. https://doi.org/10.3390/electronics12194043

Armeniakos, G., Zervakis, G., Soudris, D., & Henkel, J. (2022). Hardware approximate techniques for deep neural network accelerators: A survey. ACM Computing Surveys, 55(4), 1–36. https://doi.org/10.1145/3527156

Lavaine, C. (2024, 5 de agosto). FPGA vs. ASIC: Choosing the right technology for your project. Reflex CES. https://www.reflexces.com/newsroom/fpga-vs-asic-choosing-the-right-technology-for-your-project.

Ma, Y., Xu, Q., & Song, Z. (2023). Resource-efficient optimization for FPGA-based convolution accelerator. Electronics, 12(20), 4333. https://doi.org/10.3390/electronics12204333

Repositorio CICESE. (2017, 10 de junio). Item 1007/1348. https://cicese.repositorioinstitucional.mx/jspui/handle/1007/1348

Zhang, Z. (2024). Binary representation learning on visual images. Springer. https://doi.org/10.1007/978-981-97-2112-2

Cazacu, R. (2021). Matlab framework for image processing and feature extraction: Flexible algorithm design. MDPI. https://doi.org/10.3390/proceedings2020063072

Gao, S. (2013). Research on medical image processing method based on MATLAB. En Lecture Notes in Electrical Engineering (pp. 269–276). Springer. https://doi.org/10.1007/978-1-4471-4850-0_35

Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing (4.ª ed.). Pearson Education.

Smith, J., & Doe, A. (2020). The matrix representation of digital images in computer vision systems. Journal of Image Processing, 25(3), 123–140. https://doi.org/10.1016/j.jip.2020.06.00

Szeliski, R. (2010). Computer vision: Algorithms and applications. Springer. https://doi.org/10.1007/978-1-84882-935-0

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90. https://doi.org/10.1145/3065386

Hennessy, J. L., & Patterson, D. A. (2019). Computer architecture: A quantitative approach (6.ª ed.). Morgan Kaufmann.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). cuDNN: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759. https://doi.org/10.48550/arXiv.1410.0759

Chen, T., Xu, B., Zhang, C., & Guestrin, C. (2015). Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174. https://doi.org/10.48550/arXiv.1604.06174