Acelerador de red neuronal convolucional (CNN) para una aplicación de clasificación de imágenes en FPGA

Presentado por:
Wilson Javier Almario Rodriguez
Director:
Ph. D. Carlos Ivan Camargo Bareño

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Proyecto de grado Ingeniería Electrónica
Universidad Nacional De Colombia

September 22, 2021

Contenido

Motivación

Motivación

Implementación de redes neuronales en FPGA para aplicaciones de sistemas embebidos, explorando diferentes técnicas de cuantización que permitan optimizar la complejidad computacional, el uso de recursos y el espacio en memoria, sin tener perdidas significativas en la precisión de la inferencia de la red neuronal.

Objetivo

Implementar un acelerador de una red neuronal convolucional (CNN) en una FPGA para una aplicación de clasificación de imágenes.

Objetivo

Objetivos específicos

1. Evaluar diferentes modelos de CNN para una aplicación de reconocimiento de imágenes usando TensorFlow en CPU y GPU, y determinar el modelo más viable para su implementación en hardware.
2. Implementar un acelerador de CNN en FPGA basado en el modelo obtenido previamente en TensorFlow, para una aplicación de clasificación de imágenes.
3. Desarrollar un wrapper para un bus de comunicaciones que permita la portabilidad del acelerador.
4. Comparar las métricas obtenidas (e.g. tiempo de inferencia) en la CPU, la GPU y el acelerador de CNN implementado en una FPGA.

Metodología

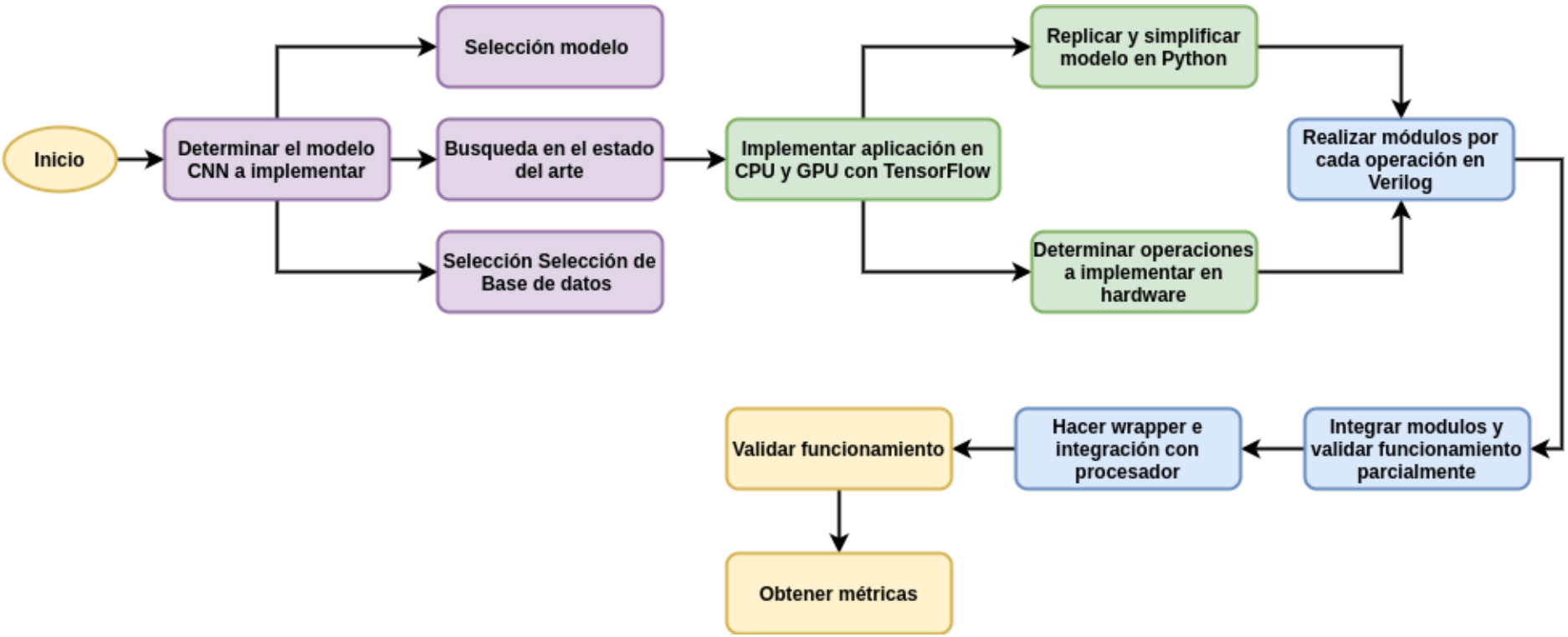

Figure 1: Diagrama de flujo de la metodología de trabajo

Modelo en software a implementar

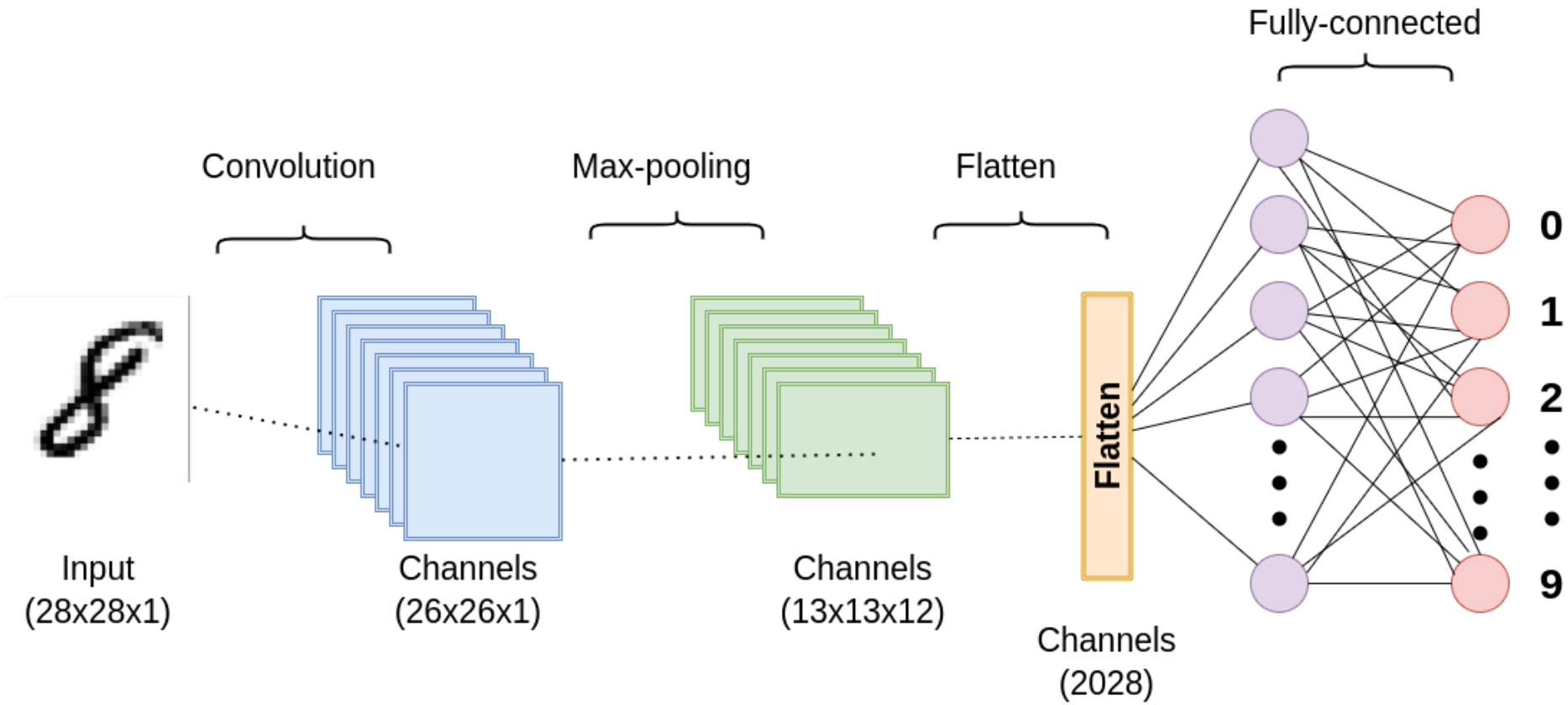

Figure 2: Diagrama del modelo de la red neuronal a implementar

Modelo en software a implementar

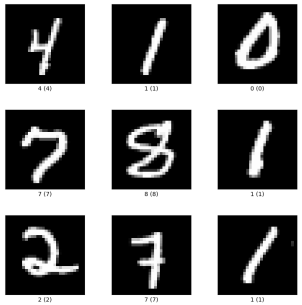

Figure 3: Ejemplo de las imágenes en la base de datos MNIST

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 28, 28, 1)	0
conv2d (Conv2D)	(None, 26, 26, 12)	120
max_pooling2d (MaxPooling2D)	(None, 13, 13, 12)	0
flatten (Flatten)	(None, 2028)	0
dense (Dense)	(None, 10)	20290

Total params: 20,410
Trainable params: 20,410
Non-trainable params: 0

Figure 4: Modelo diseñado en TensorFlow.

Cuantización

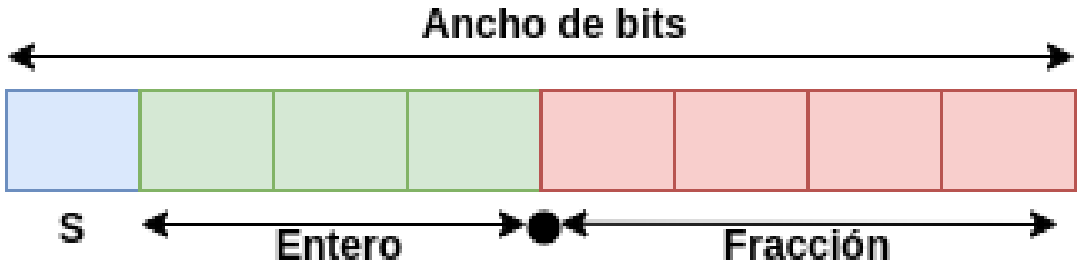

Figure 5: Representación en punto fijo con fracción.

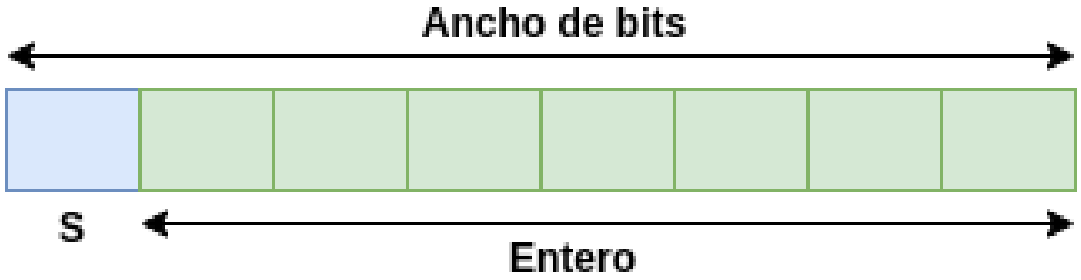

Figure 6: Representación en punto fijo solo entero.

Cuantización

Esquema de cuantización

$$r = S(q - Z) \tag{1}$$

Multiplicación Matricial

$$S_3 = Z_3 + M \sum_{j=1}^{N} (q_1^{i,j} - Z_1)(q_2^{j,k} - Z_2) \tag{2}$$

Parámetros

$$M = \frac{S_1 S_2}{S_3} \tag{3}$$

$$M = 2^{-n} M_0 \tag{4}$$

Modelo Cuantizado

Número de convolucionales	Precisión sin cuantizar	Precisión cuantizado	Número de parámetros
12	95.4%	91.6%	20410
6	94.1%	91.5%	10236
3	90.5%	87.6%	5130
1	80.5%	68.4%	1726

Table 1: Comparación de modelos variando la capa convolucional

Arquitectura implementada en Hardware

Módulo de cuantización

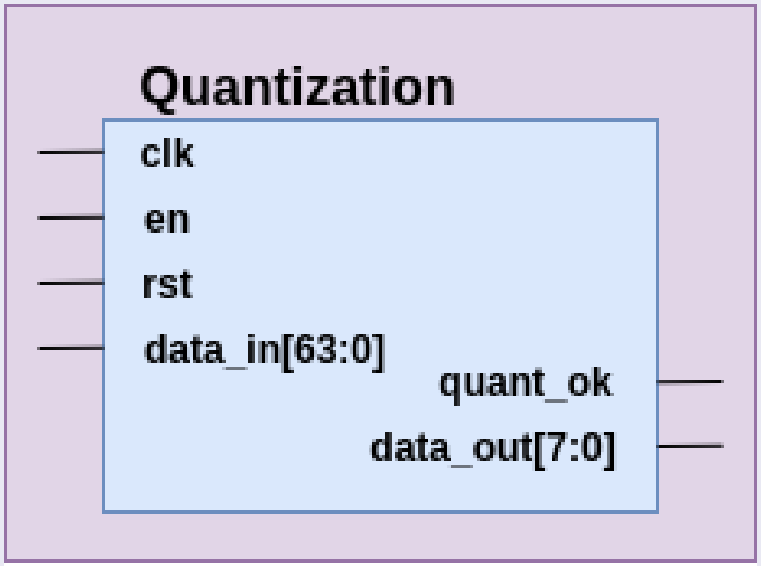

Figure 7: Módulo de cuantización implementado en hardware.

Función de activación

$$f(x) = \left\{ \begin{array}{ccc} 127 & si & x > 127 \\ -128 & si & x < -128 \\ x & si & -128 < x < 127 \end{array} \right\} \quad (5)$$

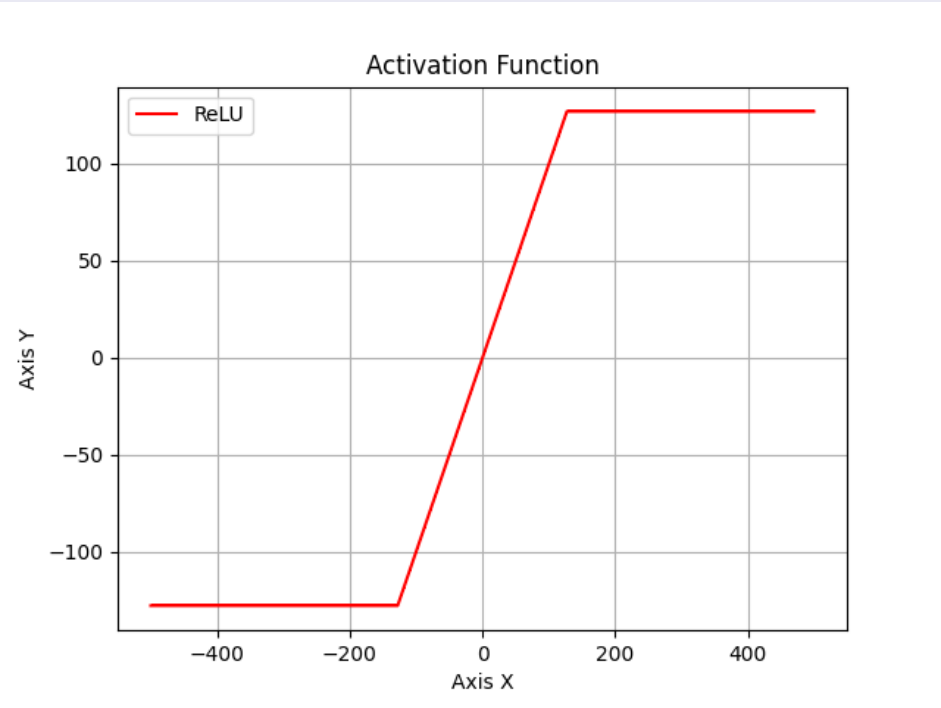

Figure 8: Función de activación adaptada al proyecto.

Arquitectura implementada en Hardware

Diseño de operaciones en hardware

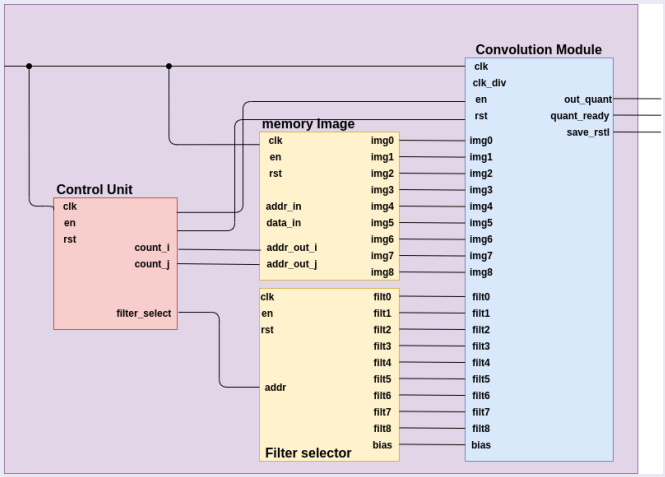

Figure 9: Diseño de forma secuencial.

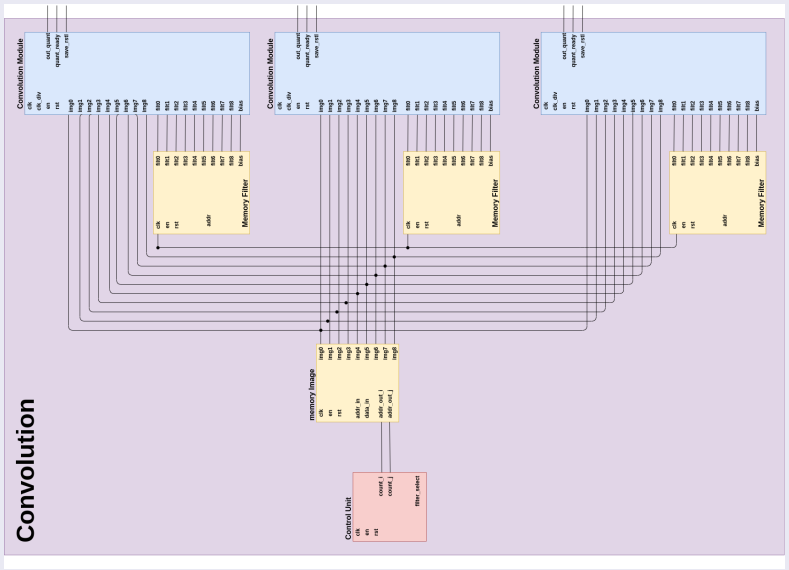

Figure 10: Diseño de forma paralela.

Arquitectura implementada en Hardware

Diseño 3 convolucionales	**Tiempo de ejecución (ms)**	**LUTs**	**LUTs as Logic**	**Slice**
Paralelo	0.173	564	564	321
Secuencial	0.520	188	265	107

Table 2: Comparación tiempo de ejecución entre el diseño en paralelo y el diseño secuencial.

Convolución en hardware

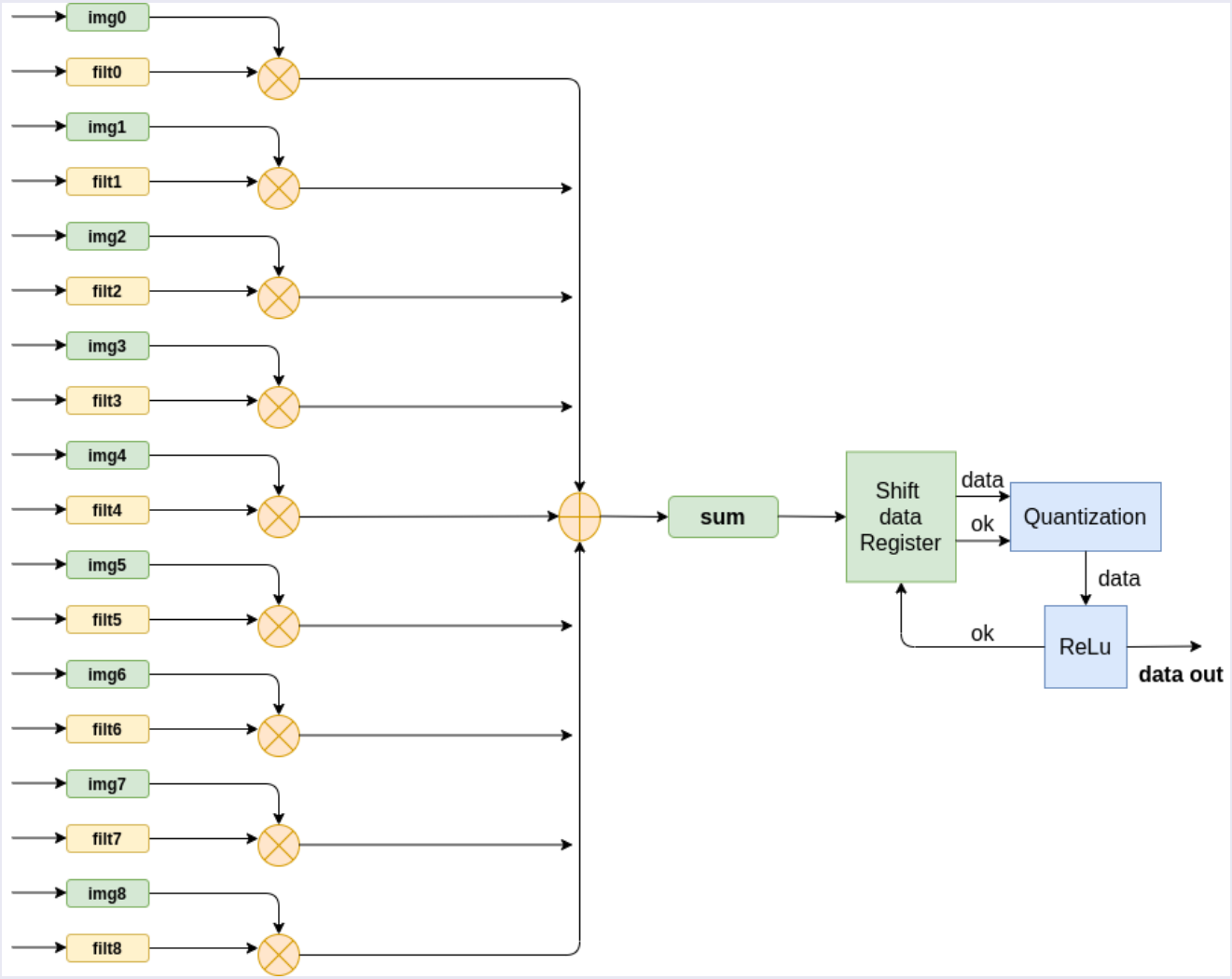

Figure 11: Operaciones en el modulo de convolución.

Arquitectura implementada en Hardware

Acelerador completo

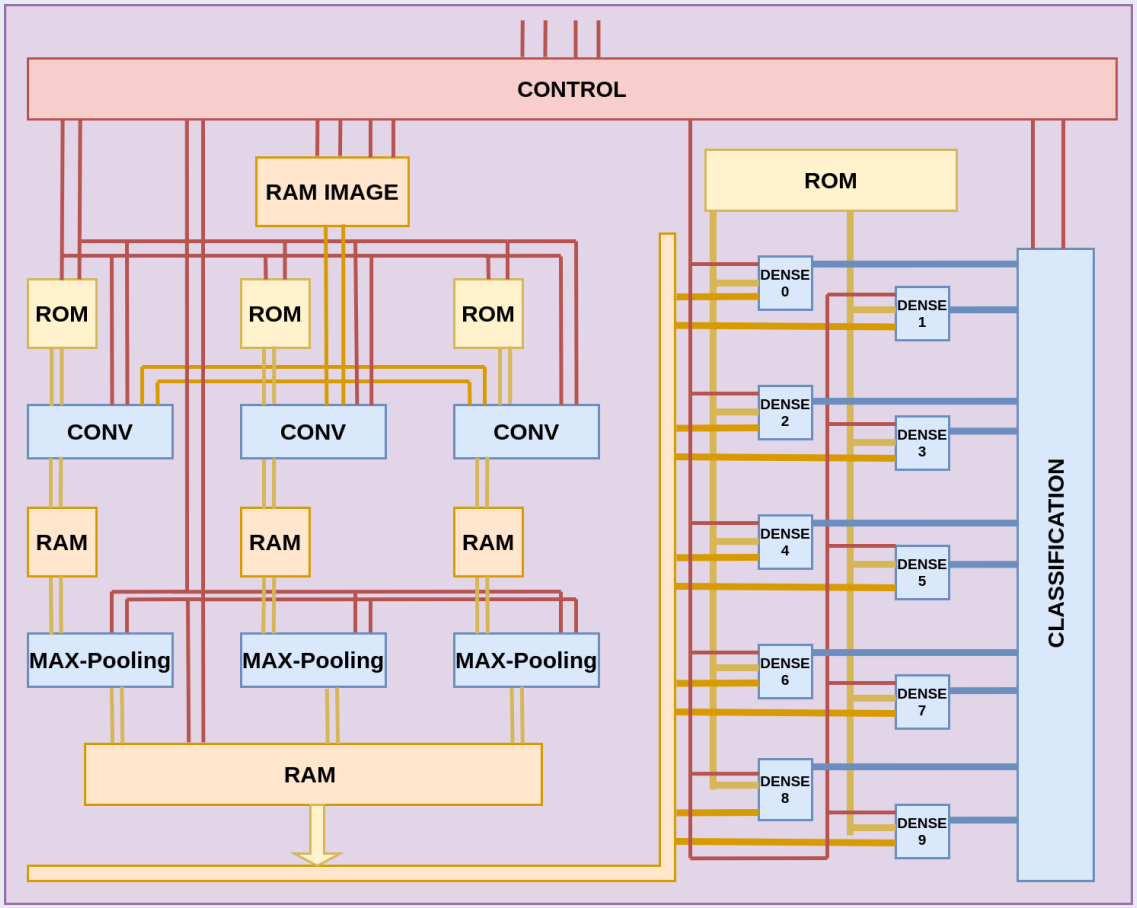

Figure 12: Acelerador de CNN completo .

Arquitectura implementada en Hardware

Acelerador con procesador Vexrisc

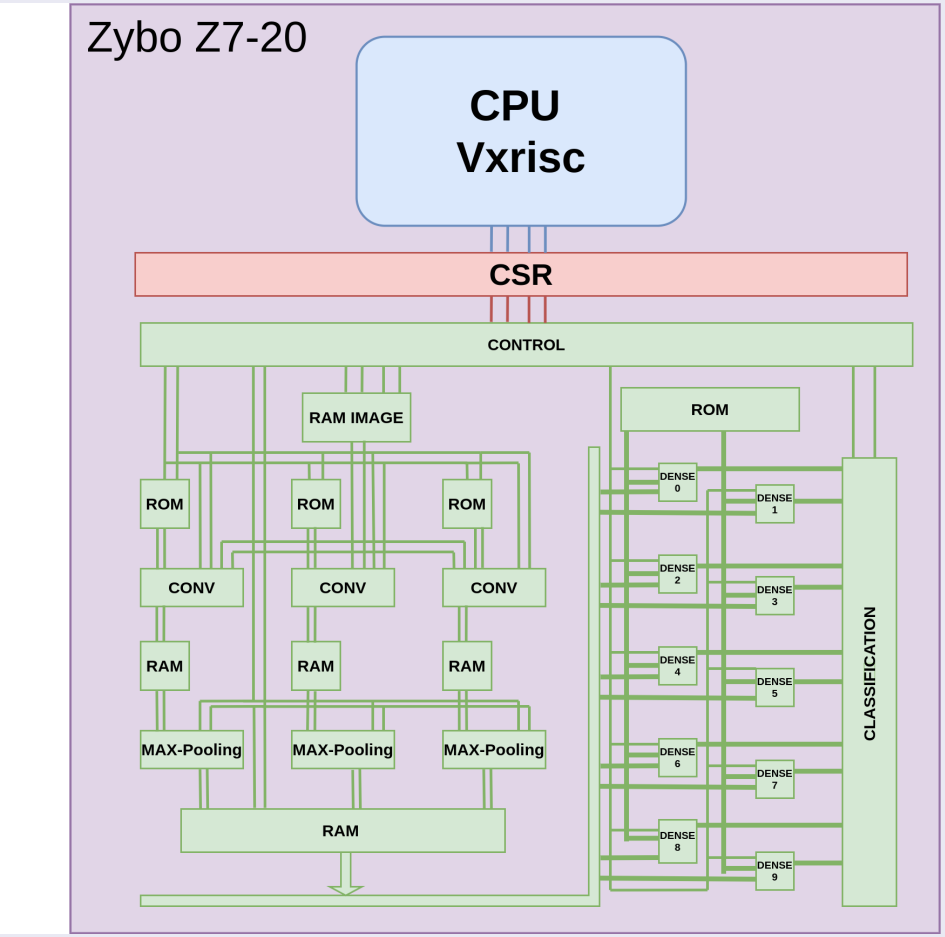

Figure 13: Acelerador de CNN con CPU comunicado a través de bus CSR.

Resultados

Espacio en memoria

	Tipo de dato	Parametros	Memoria	Precisión
CNN Referencia	Float 32-bit	20410	653.1 kB	95.4 %
CNN Cuantizada	Integer 8-bit	5130	4.1 kB	87.6 %

Figure 14: Comparación en términos de memoria utilizada entre el modelo de referencia y elimplementado en la FPGA.

Resultados

Comparación entre plataformas

	CPU	GPU	CPU	GPU	FPGA
Modelo	CNN	CNN	Q-CNN	Q-CNN	Q-CNN
Plataforma	Intel(R) i7-6700 @ 3.40GHz	GeForce GTX 750 Ti	Intel(R) i7-6700 @ 3.40GHz	GeForce GTX 750 Ti	Zynq-7020
Framenwork /Lenguaje	TensorFlow	TensorFlow	Python	Ptyhon / Numba	HDL (Verilog)
Tiempo (ms)	508.1$\pm$3	476.2 $\pm$4	71.1 $\pm$2	12.3 $\pm$3	0.190
Potencia (W)	65	60	65	60	0.057

Figure 15: Comparación de la implementación del modelo con diferentes plataformas.

Resultados

Recursos utilizados

	Acelerador Q-CNN	CPU Vexrisc	Total usado	Disponible	Util %
LUTs	12613	2321	14934	53200	28.07 %
Registers	6347	1938	8285	106400	7.79 %
F7 Muxes	536	0	536	26600	1.98 %
F8 Muxes	248	0	248	13300	1.88 %
Slice	3775	802	4577	13300	34.41 %
LUT as Logic	9753	2302	12058	53200	22.67 %
LUT as Memory	2860	10	2860	17400	16.53 %

Figure 16: Recursos utilizados en la implementación del acelerador en la Zybo-7020.

Resultados

Comparación con trabajos similares

	Acelerador Propuesto	Artículo [16]	Artículo [17]	Artículo [18]
Plataforma	Zynq - 7020	xczu9eg-ffvb1 156-2-i FPGA	XC7K325T FPGA	Virtex VC707
Frecuencia (MHz)	150	———	200	200
Modelo	1 conv 1 pooling 1 fc	3 conv 2 pooling 1 fc	3 conv 2 pooling 1 fc	2 conv 1 pooling 1 fc
Base de datos	MNIST	MNIST	MNIST	MNIST
Datos de entrada	28x28	32x32	32x32	28x28
Cuantización	Fixed-point Integer 8-bit	Fixed-point Q5.14	Fixed-point Q16.11	Fixed-point Q5.14
Precisión	87.6 %	98.64 %	98.53 %	98.66 %
Tiempo (ms)	0.190	3.58	1.043	21.27
LUTs	12613	21260	31825	55774
Potencia (W)	0.057	———	———	1.582

Figure 17: Comparación con otros trabajos.

Conclusiones

Conclusiones y trabajo futuro

- El diseño de redes neuronales cuantizadas presenta ventajas para la implementación en hardware como la reducción del tiempo de ejecución en las operaciones y el bajo consumo de energía lo que lo hace una alternativa viable para realizar sistemas embebidos en los que existe un procesador y se puede añadir un acelerador para una aplicación especifica.
- Debido a la estrategia de continuación implementada en este trabajo se logra una reducción en la complejidad de las operaciones y menor uso de la memoria para almacenar los pesos, pero se tiene un costo en la precisión, así que se recomienda buscar un equilibrio entre el uso de recursos disponibles y la precisión que se desee lograr del modelo.
- Una desventaja de este tipo de diseños es que no se puede generalizar y siempre dependerá de la aplicación que se quiera implementar, pero si se puede generalizar la metodología de cuantización la cuál como se vio en el proyecto está ligada a su diseño en software.

Bibliografía

[1] [Online]. Available: https://www.tensorflow.org/datasets/catalog/mnist

[2] S. H. David Harris, *Digital Design and Computer Architecture*, 2007. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=ebff8085ab88f864e6130720897afac0

[3] V. T. Phat, P. H. Tho, H. B. Dat, and C.-H. Chou, "Deep learning accelerator on fpga using handwritten digit recognition for example," in *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2018, pp. 1–2.

[4] T. Abtahi, C. Shea, A. Kulkarni, and T. Mohsenin, "Accelerating convolutional neural network with fft on embedded hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 9, pp. 1737–1749, 2018.

[5] K. Zhao, T. He, S. Wu, S. Wang, B. Dai, Q. Yang, and Y. Lei, "Application research of image recognition technology based on cnn in image location of environmental monitoring uav," *EURASIP Journal on Image and Video Processing*, vol. 2018, p. 150, 12 2018.

[6] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, 09 2014.

[7] S. A. Z. U. e. a. Khan, A., "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, p. 5455–5516, 04 2020.

[8] T. Wang, C. Wang, X. Zhou, and H. Chen, "An overview of fpga based deep learning accelerators: Challenges and opportunities," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019, pp. 1674–1681.

Bibliografía

[9] E. Nurvitadhi, J. Sim, D. Sheffield, A. Mishra, S. Krishnan, and D. Marr, "Accelerating recurrent neural networks in analytics servers: Comparison of fpga, cpu, gpu, and asic," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–4.

[10] A. G. Blaiech, K. Ben Khalifa, C. Valderrama, M. A. Fernandes, and M. H. Bedoui, "A survey and taxonomy of fpga-based deep learning accelerators," *Journal of Systems Architecture*, vol. 98, pp. 331–345, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762118304156

[11] A. X. M. Chang and E. Culurciello, "Hardware accelerators for recurrent neural networks on fpga," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[12] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[13] F. Chollet, *Deep Learning with Python*. Manning, 2018. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=deb175581d4a7579c3654f0a0862b5e2

[14] [Online]. Available: https://www.tensorflow.org/datasets/keras_example

[15] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 06 2018, pp. 2704–2713.

Bibliografía

[16] M. Cho and Y. Kim, "Implementation of data-optimized fpga-based accelerator for convolutional neural network," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, 2020, pp. 1–2.

[17] X. Zhen and B. He, "Research on fpga high-performance implementation method of cnn," in *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2021, pp. 1177–1181.

[18] A. Kyriakos, V. Kitsakis, A. Louropoulos, E.-A. Papatheofanous, I. Patronas, and D. Reisis, "High performance accelerator for cnn applications," in *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2019, pp. 135–140.

Anexos

Repositorio

- `https://gitlab.com/wjalmarior/fpga-accelerator-for-quantized-convolutional-neural-netwo`
  `git`

Max-pooling en hardware

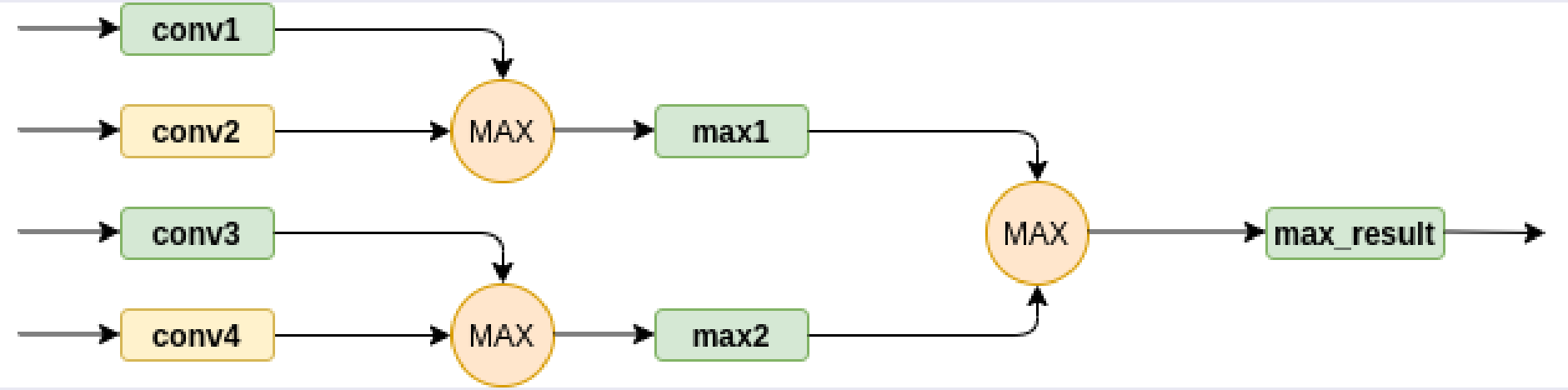

Figure 18: Operaciones en el módulo de *Max-pooling*

Fully-connected en hardware

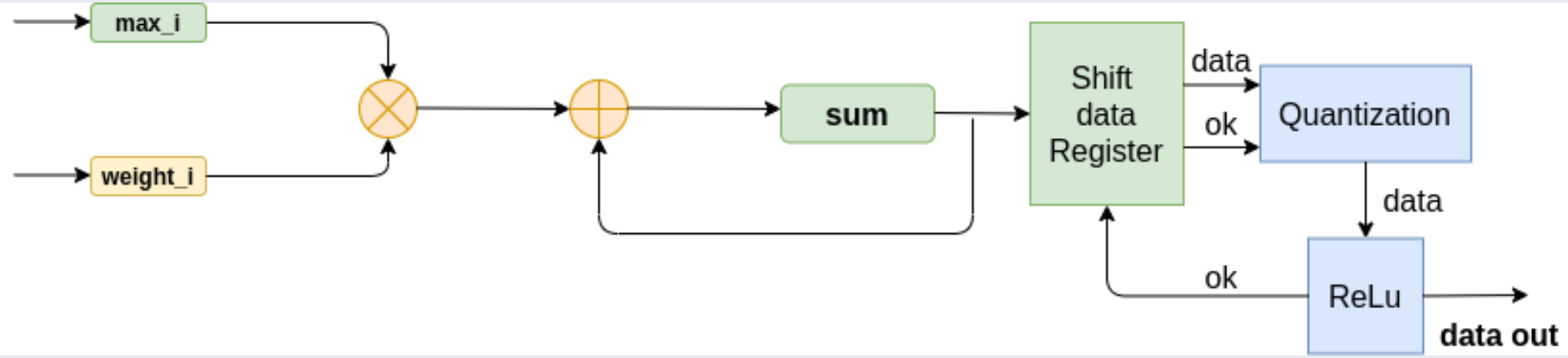

Figure 19: Diseño de operaciones para el módulo de capa densa.