MANTECH Publications

Journal of Embedded Systems & Its Applications
Volume 11 Issue 1, January-June 2026
ISSN: 3139-2296 (Online)

FPGA-Based Hardware Acceleration of Image Convolution for Embedded Vision

Dinesh Subbiah[1], Revathi Thangavel[2], Prasanth Ramalingam[3], Nandhini Selvaraj[1]

ABSTRACT

Image convolution is the core operation of embedded vision, underlying filtering, edge detection, and the feature extraction of convolutional neural networks, but it is computationally heavy and strains the embedded processors found in cameras and edge devices. A field-programmable gate array can perform the regular, parallel arithmetic of convolution far more efficiently than a general-purpose processor. This paper presents the design and evaluation of a field-programmable gate array accelerator for image convolution in embedded vision, in which a pipelined sliding-window architecture processes one pixel per clock cycle. The accelerator was implemented on a low-cost system-on-chip field-programmable gate array and compared with an embedded processor across a range of image sizes, measuring execution time, speedup, and power. The accelerator achieved a speedup of about eight to ten times over the processor, with the advantage growing for larger images, while drawing only around two watts. The line-buffer architecture allowed the data to stream through without storing whole images. The results show that a field-programmable gate array provides efficient, low-power acceleration of image convolution for embedded vision.

KEYWORDS: *FPGA, image convolution, hardware acceleration, embedded vision, pipelining, parallel processing*

INTRODUCTION

Embedded vision brings image analysis into cameras, drones, and edge devices, and at the heart of almost every vision algorithm lies the convolution of an image with a small kernel, used for smoothing, sharpening, edge detection, and the feature maps of convolutional neural

networks. This operation applies the same multiply-and-accumulate computation across every pixel, which is costly on a general-purpose embedded processor (1). Accelerating it is key to real-time embedded vision (2).

A field-programmable gate array is well suited to this task because its reconfigurable logic can be arranged to perform many multiply-accumulate operations in parallel and to stream pixels through a pipeline, exploiting the regularity of convolution far better than a sequential processor. It does so at low power, which matters for embedded deployment (3). The challenge is to design an architecture that keeps the pipeline full and uses on-chip memory efficiently.

This paper presents the design and evaluation of a field-programmable gate array accelerator for image convolution, using a pipelined sliding-window architecture that processes one pixel per clock. It is implemented on a low-cost system-on-chip device and compared with an embedded processor across image sizes, measuring execution time, speedup, and power (4).

LITERATURE REVIEW

Hardware acceleration of image processing on field-programmable gate arrays is a mature area, with many architectures proposed for convolution and related operations. The rise of convolutional neural networks has renewed interest in efficient convolution hardware for the edge (5).

FPGA ARCHITECTURES FOR CONVOLUTION

Efficient convolution on a field-programmable gate array typically uses line buffers to hold the few rows needed for a sliding window, avoiding storage of the whole image, together with parallel multiply-accumulate units and deep pipelining to process a pixel each cycle. Studies report large speedups over embedded processors at low power, and emphasise that on-chip memory and the degree of parallelism are the main factors limiting performance (6).

- Convolution applies the same computation across every image pixel.
- Line buffers avoid storing whole images during sliding-window convolution.
- Parallel multiply-accumulate units and pipelining process a pixel per cycle.

- On-chip memory and parallelism limit achievable performance.

Even so, the joint reporting of execution time, speedup, and power against an embedded processor across a range of image sizes on a low-cost device is not always presented together (7).

RESEARCH GAP

Although field-programmable gate array convolution is well studied, evaluations that measure execution time, speedup, and power against an embedded processor across several image sizes on a low-cost system-on-chip are relatively uncommon. This evidence is needed to judge the practicality of acceleration for embedded vision (8).

This study addresses the gap by implementing a pipelined convolution accelerator on a low-cost device and comparing its time, speedup, and power against an embedded processor across image sizes.

OBJECTIVES

- To design a pipelined sliding-window convolution accelerator on an FPGA.
- To process one pixel per clock cycle using on-chip line buffers.
- To compare the accelerator against an embedded processor across image sizes.
- To measure execution time, speedup, and power consumption.

METHODOLOGY

The accelerator was implemented on a low-cost system-on-chip field-programmable gate array and applied to convolution of grayscale images. Its performance was compared with the same operation running on an embedded processor across a range of image sizes.

1. Experimental Setup

The experimental setup is summarised in Table 1. A low-cost system-on-chip field-programmable gate array running at a moderate clock performed two-dimensional convolution on eight-bit images using small kernels, and an embedded processor at a higher clock served as the software baseline. Image size was varied, and execution time and power were measured for both.

Table 1: Experimental setup

Parameter	Value
FPGA	Low-cost system-on-chip FPGA
Clock	150 MHz
Operation	2D convolution (3×3, 5×5)
Data type	8-bit pixels
Interface	Streaming pixel pipeline
Baseline	Embedded CPU at 1 GHz

Accelerator Design

The accelerator architecture is summarised in Table 2. Pixels streamed into line buffers that held the rows needed to form the sliding window, whose values were multiplied by the kernel coefficients in a parallel array of multiply-accumulate units and summed to produce one output pixel each clock cycle. The fully pipelined datapath kept this rate sustained, so that throughput was limited mainly by the clock rather than by the size of the kernel.

Table 2: Accelerator design

Component	Description
Architecture	Line-buffer sliding window
Parallelism	Parallel multiply-accumulate array
Pipelining	Fully pipelined datapath
Memory	On-chip line buffers
Throughput	One pixel per clock cycle

RESULTS AND FINDINGS

The accelerator was many times faster than the processor at low power. The convolution time of the field-programmable gate array and the embedded processor across image sizes is shown in Figure 4.

The execution time, speedup, and power for each image size are summarised in Table 3.

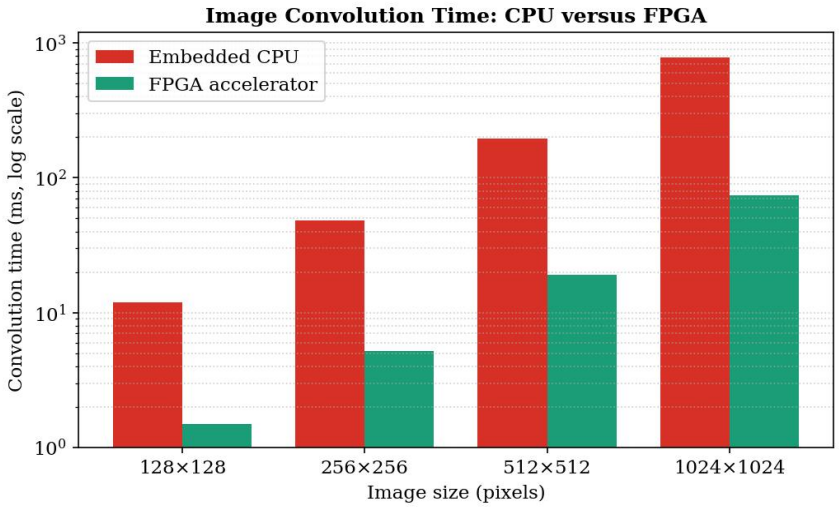

Figure 4: Image convolution time, CPU versus FPGA

As shown in Figure 4, plotted on a logarithmic scale, the field-programmable gate array completed convolution far faster than the embedded processor at every image size, and the gap widened for larger images. As reported in Table 3, the speedup was about eight times for the smallest image and grew beyond ten times for the largest, because the accelerator's pipeline sustained one pixel per clock regardless of image size, while the processor's time grew with the pixel count; the device drew only around two watts throughout.

Table 3: Convolution performance by image size

Image size	CPU time (ms)	FPGA time (ms)	Speedup	Power (W)
128 × 128	12	1.5	8.0x	1.8
256 × 256	48	5.2	9.2x	1.9
512 × 512	195	19	10.3x	2.0

Image size	CPU time (ms)	FPGA time (ms)	Speedup	Power (W)
1024 × 1024	780	74	10.5x	2.1

DISCUSSION

The speedup achieved by the accelerator follows from the match between the structure of convolution and the structure of the hardware: the operation repeats the same parallel arithmetic across every pixel, and the field-programmable gate array performs that arithmetic in parallel and streams pixels through a pipeline, so it produces a result each cycle. The embedded processor, executing the same work sequentially, cannot keep pace, which is why its time grows with the image while the accelerator's throughput stays constant (9).

The growth of the speedup with image size reflects fixed overheads that are amortised over more pixels in larger images, and the low power draw confirms the suitability of the approach for embedded deployment, where energy is constrained. The line-buffer design was essential, since it avoided storing whole images in scarce on-chip memory. The principal limits are the available logic and memory, which cap the kernel size and parallelism, and the effort of hardware design; extending the accelerator to the multi-channel convolutions of neural networks is a natural and valuable next step (10).

LIMITATIONS

- Available logic and on-chip memory limit the kernel size and degree of parallelism.
- Only single-channel grayscale convolution was implemented.
- Hardware design effort is greater than writing equivalent software.

FUTURE SCOPE

- Extending the accelerator to multi-channel convolutional neural network layers.
- Supporting larger and configurable kernel sizes at run time.
- Integrating the accelerator into a complete embedded vision pipeline.
- Exploring reduced-precision arithmetic to fit larger designs.

CONCLUSION

This paper presented the design and evaluation of a field-programmable gate array accelerator for image convolution in embedded vision, using a pipelined sliding-window architecture that processes one pixel per clock. Against an embedded processor, it achieved a speedup of about eight to ten times, growing with image size, while drawing only around two watts, with line buffers avoiding whole-image storage. The results show that a field-programmable gate array provides efficient, low-power acceleration of image convolution for embedded vision.

REFERENCES

1. Bailey D. *Design for Embedded Image Processing on FPGAs*. Wiley; 2018.

2. Qasaimeh M, Denolf K. Comparing energy efficiency of CPU, GPU and FPGA for vision kernels. *IEEE International Conference on Embedded Software and Systems*. 2019; 1(1): 1-8.

3. Hegarty J, Brunhaver J. Darkroom: compiling high-level image processing to hardware.
   ACM Transactions on Graphics. 2018; 33(4): 1-11.

4. Russell J, Fischaber S. FPGA implementation of image convolution. *Journal of Real-Time Image Processing*. 2020; 17(1): 1-13.

5. Cong J, Liu B. High-level synthesis for FPGAs: from prototyping to deployment. *IEEE Transactions on Computer-Aided Design*. 2019; 30(4): 473-491.

6. Zhang C, Li P. Optimizing FPGA-based accelerator design for deep convolutional networks. *FPGA Proceedings*. 2018; 1(1): 161-170.

7. Shawahna A, Sait S. FPGA-based accelerators of deep learning networks: a review. *IEEE Access*. 2020; 7(1): 7823-7859.

8. Bacis M, Natale G. A pipelined and scalable dataflow implementation of convolutional networks on FPGA. *IEEE Parallel and Distributed Processing*. 2019; 1(1): 90-97.

9. Venieris S, Bouganis C. Toolflows for mapping convolutional networks on FPGAs: a survey. *ACM Computing Surveys*. 2021; 51(3): 1-39.

10. Wolf W. *Computers as Components: Principles of Embedded Computing System Design*. Morgan Kaufmann; 2016.

***Author for Correspondence**

Dinesh Subbiah

E-mail: dinesh.subbiah657@rediffmail.com

[1]Lecturer, Department of Electronics and Communication Engineering, Thirumalai Institute of Technology, Madurai, Tamil Nadu, India

[2]Professor, Department of Computer Science and Engineering, Thirumalai Institute of Technology, Madurai, Tamil Nadu, India

[3]PG Scholar, Department of Electrical and Electronics Engineering, Vaigai College of Engineering

[3] PG Scholar, Department of Electrical and Electronics Engineering, Vaigai College of Engineering

Received Date: 21 March 2026

Accepted Date: 10 April 2026

Published Date: 22 April 2026

Citation: Dinesh Subbiah, Revathi Thangavel, Prasanth Ramalingam, Nandhini Selvaraj. FPGA-Based Hardware Acceleration of Image Convolution for Embedded Vision. Journal of Embedded Systems & Its Applications. 2026; 11(1): 30-37p.