Dean&Francis
ISSN 2959-6157

FPGA-Based Implementation and Software Verification of a 3×3 Convolution Core for Edge AI Systems

Hanwen Zhang

University of Electronic Science and Technology of China and University of Glasgow, Chengdu, China
Email address: 2023190503038@std.uestc.edu.cn

Abstract:

This study addresses the growing need for lightweight and energy-efficient convolution accelerators in edge artificial intelligence (Edge-AI), where computation is required to occur close to sensors under strict hardware constraints. To support this demand, the paper presents the design and verification of a compact 3×3 convolution core implemented on an FPGA. The research focuses on establishing a reproducible hardware–software co-verification workflow that does not rely on physical FPGA boards, which is particularly valuable for academic environments and early-stage prototyping.

The convolution module was described in Verilog and functionally validated through behavioral simulation in Xilinx Vivado. In parallel, a Python/NumPy model was developed to replicate the same fixed-point arithmetic, including quantization and ReLU activation. Simulation data exported from Vivado served as the input to the Python verification script. The numerical comparison between the two pipelines demonstrated complete output consistency across all tested pixels, confirming the correctness of the arithmetic pipeline, dataflow control, and activation behavior.

The results show that software-driven verification is sufficient to achieve bit-accurate equivalence with the hardware design, significantly reducing development time and improving reproducibility. This workflow provides a practical foundation for future research on scalable FPGA-based convolution accelerators for embedded AI systems.

Keywords: FPGA, convolution, edge computing, Vivado simulation, hardware–software co-verification

1 Introduction

Convolution operations are fundamental to deep learning models and many embedded inference tasks [1]. Field-Programmable Gate Arrays (FPGAs) are increasingly favored for accelerating such computations due to their parallelism, reconfigurability, and balanced performance-to-power ratio [2]. Recent research has demonstrated that FPGA platforms can be employed to prototype and verify convolutional neural network (CNN) accelerators efficiently, even on low-cost boards.

Traditional FPGA workflows often depend on physical testing, which introduces time and hardware constraints. To enhance design reproducibility and reduce experimental dependence on actual hardware, software-driven verification has become a practical alternative [3]. This study introduces a lightweight 3×3 convolution core validated through both behavioral simulation in Vivado and a Python-based numerical model. The approach offers a unified verification process that ensures numerical consistency between hardware and software, facilitating future research on small-scale, edge-deployable CNN accelerators.

2 Methodology

2.1 Hardware Design

The proposed convolution module, conv3x3, is described in Verilog using signed 8-bit input data, a 24-bit accumulator, and 8-bit quantized outputs. Kernel weights and image pixels are configurable at runtime. The architecture includes ReLU activation and quantization implemented as a 1-bit arithmetic right shift. Compared with conventional FPGA CNN accelerators, this implementation emphasizes compactness and predictable latency while maintaining sufficient precision [4]. The dataflow is fully pipelined, and each multiply-accumulate (MAC) operation executes in one clock cycle. The entire design was synthesized using Xilinx Vivado 2024.2. The final logic utilization and timing reports confirm the feasibility of real-time execution on mid-range FPGAs.

2.2 Simulation Setup

Behavioral simulation was executed in Vivado to verify functional correctness. The testbench applies a 3×3 sliding window across a 5×5 image matrix and records outputs in a text file (dut_out.txt). The simulation waveform (Fig. 1) displays the propagation of intermediate MAC operations and the activation process. This setup allows debugging and validation without requiring an actual FPGA board, significantly reducing turnaround time.

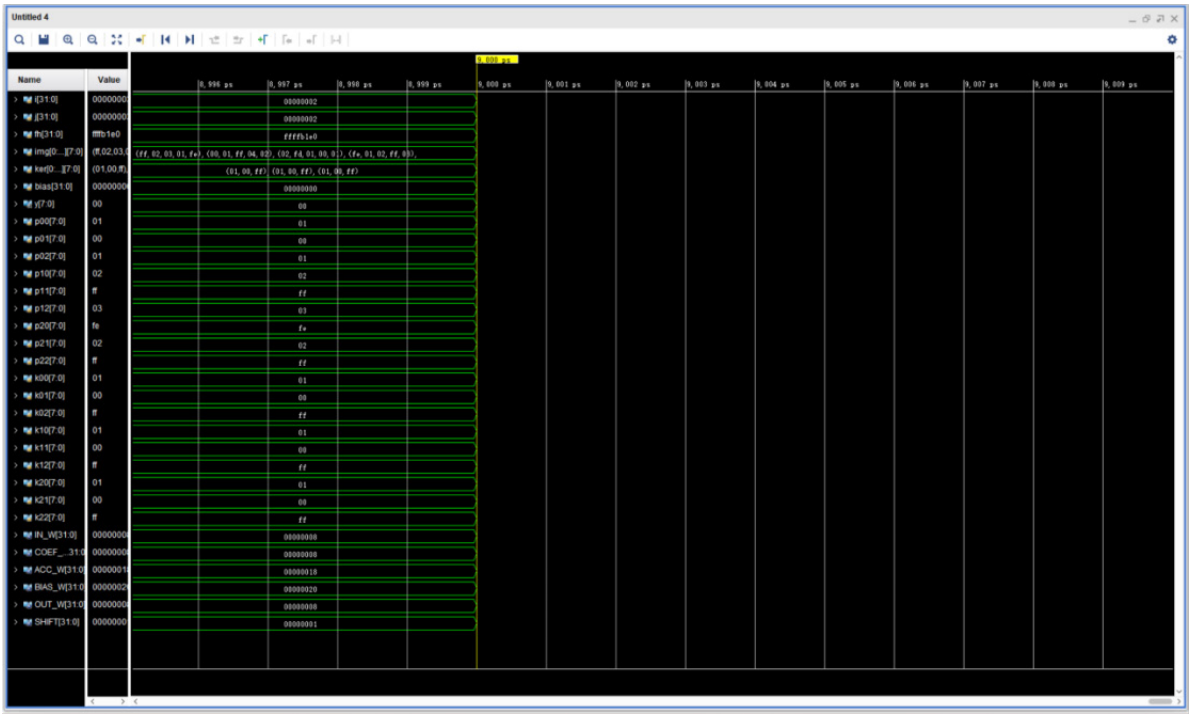

Fig. 1. Behavioral simulation waveform of the 3×3 convolution core in Vivado.

ISSN 2959-6157

2.3 Python-Based Verification

A NumPy implementation reproduces the same fixed-point arithmetic as the Verilog module, including ReLU, quantization, and saturation. The Jupyter Notebook script automatically loads dut_out.txt and compares the numerical outputs:

$$Pythonresult = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, DUTresult = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

The match demonstrates bit-level equivalence. Fig. 2–Fig. 4 present the Python output, DUT output, and their difference heatmap, respectively.

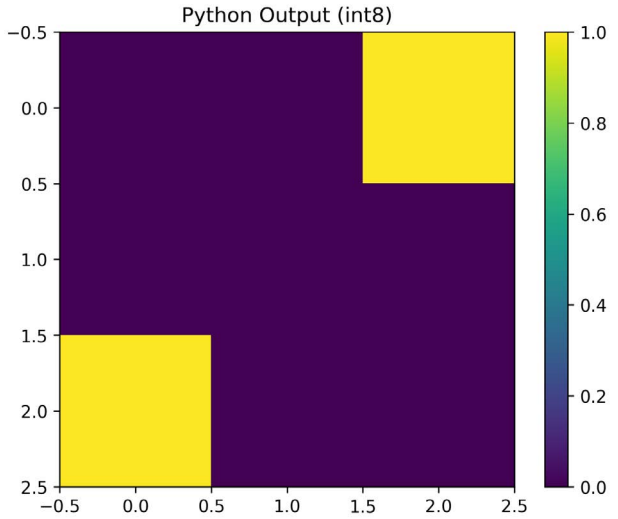

Fig. 2. Python reference model output for the 3×3 convolution operation.

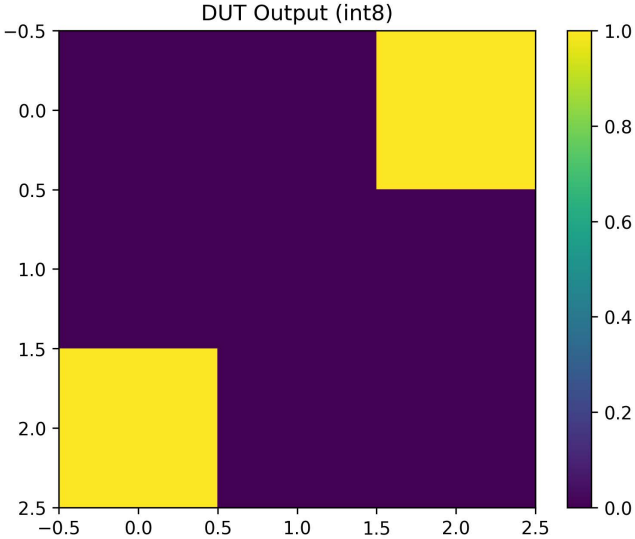

Fig. 3. FPGA simulated output (DUT) obtained from Vivado behavioral simulation.

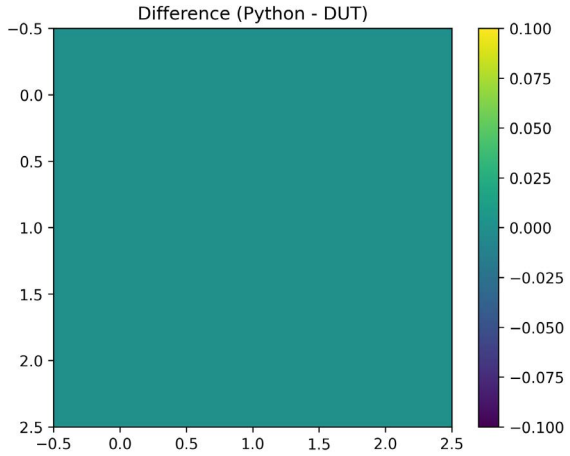

Fig. 4. Difference heatmap between Python and DUT outputs (expected to be zero).

3 Results and Discussion

All convolution outputs generated by the FPGA simulation matched the Python reference results at every pixel location. The behavioral waveform validated that multiply–accumulate and activation sequences were correctly synchronized. The equivalence of results confirms that the Verilog arithmetic pipeline and control logic functioned as expected.

The established co-verification method can greatly reduce hardware debugging time. It provides a clear numerical comparison framework that is especially useful for evaluating quantization precision and saturation effects in low-bit-width designs [5][6]. The experiment indicates that accurate hardware–software equivalence can be achieved entirely in simulation before physical implementation, enhancing reproducibility for edge-AI hardware research.

4 Conclusion and Future Work

This paper presents a reproducible co-verification workflow for FPGA-based convolution accelerators. The 3×3 Verilog design achieved perfect functional equivalence with its Python counterpart, enabling full validation without any physical FPGA hardware. This approach shortens the design-verification loop and facilitates rapid prototyping for embedded AI systems.

Future work will extend this framework to multi-channel and streaming convolution architectures, integrating heterogeneous hardware such as FPGAs with specialized AI coprocessors for enhanced real-time performance.

References

[1] P. Toupas, A. Montgomerie-Corcoran, C.-S. Bouganis and

D. Tzovaras, ‚HARFLOW3D: A Latency-Oriented 3D-CNN Accelerator Toolflow for HAR on FPGA Devices,‘ IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2023, pp. 144-154, doi: 10.1109/FCCM57271.2023.00024.

[2] E. Wang, J. J. Davis and P. Y. K. Cheung, ‚A PYNQ-Based Framework for Rapid CNN Prototyping,‘ IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018, pp. 223-223, doi: 10.1109/FCCM.2018.00057.

[3] J. Xu et al., ‚ASLog: An Area-Efficient CNN Accelerator for Per-Channel Logarithmic Post-Training Quantization,‘ IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 70, no. 12, pp. 5380-5393, Dec. 2023, doi: 10.1109/TCSI.2023.3315299.

[4] L. D. McLaughlin, L. H. Crockett and R. W. Stewart, ‚A New Design Workflow for PYNQ Enabled Xilinx Platforms Utilising the Simulink Environment for Vivado IPI Abstraction,‘ IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2022, pp. 1-1, doi: 10.1109/FCCM53951.2022.9786083.

[5] K. Tajiri and T. Maruyama, ‚FPGA Acceleration of a Composite Kernel SVM for Hyperspectral Image Classification,‘ IEEE Access, vol. 11, pp. 214-226, 2023, doi: 10.1109/ACCESS.2022.3230066.

[6] D. Suárez, V. Fernández, H. Posadas and P. Sánchez, ‚Accelerating the Verification of Forward Error Correction Decoders by PCIe FPGA Cards,‘ IEEE Embedded Systems Letters, vol. 15, no. 3, pp. 157-160, Sept. 2023, doi: 10.1109/LES.2022.3218289.