International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

Low Power LeNet CNN Accelerator On FPGA

[1]Mr. Vishnu Kanth G, [2]Deshagouni Sai Teja, [3]Athmakuri Sai Ram, [4]Mohammed Muqueed Uddin

[1]Assistant Professor, Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India
[2]Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India
[3]Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India
[4]Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India

Abstract – This paper presents the design and realization of a dedicated hardware engine for LeNet convolutional neural network. The aim of this project is identifying handwritten numerals using MNIST data set. As edge devices grow in number, there is a pressing need to offer approaches, that enable neural network inference under tight power budgets and constrained hardware footprints. We employ fixed-point arithmetic, a pipelined MAC datapath, and on-chip block memory for weight storage to carry out the design in synthesizable Verilog HDL.
Once implemented, Nexys 4 proved that our design draws roughly 0.135 W and occupies 1990 LUTs. The design also fits comfortably within Basys 3, which offers fewer hardware resources. In that scenario, the utilization reaches 2029 LUTs with 0.139 W drawn. Also, behavioral simulation confirmed that the design operates correctly on UltraScale+ devices, like ZCU102 board. The trained model achieves a recognition rate of 98.29% on the MNIST benchmark. A side-by-side evaluation of design behavior across different FPGA boards is included.
Keywords - FPGA, convolutional neural network, LeNet, hardware accelerator, low-power design, fixed-point arithmetic, Artix-7, Zynq UltraScale+, MNIST, edge inference.

I. Introduction

Present-day edge devices—ranging from sensors to wearable gadgets and embedded controllers—demand neural network processing that is both rapid and efficient, despite operating under stringent power and resource constraints. Relying on cloud-based processing isn't always a good idea because of privacy concerns, delays, and bandwidth problems. FPGAs present a compelling alternative because they offer a favorable trade-off between reconfigurability and throughput, making them well suited for crafting tailor-made neural network accelerators.
The LeNet family of CNN architectures proposed by LeCun et al. [3] continues to serve as a popular reference point. Owing to its compact structure and straightforward topology, it is well suited for prototyping and benchmarking hardware CNN realizations before tackling deeper networks. With a modest parameter count and an uncomplicated layer arrangement, such architectures are quick to set up. This makes them handy for validating accelerator designs prior to scaling up to heavier networks.
A notable shortcoming in existing literature is that most accelerator designs are built and assessed on just one FPGA platform. This leaves open the question of how the identical architecture would fare on boards offering different resource levels. The absence of cross-platform benchmarks makes it challenging for budget-conscious students and engineers to pick suitable hardware. Hence, studies that evaluate the same architecture spanning multiple FPGA platforms are clearly needed.
The chief contribution of this work lies in bridging this gap through a LeNet accelerator design, accompanied by its deployment and assessment on three distinct Xilinx FPGA boards that differ in cost and available resources.
- The Nexys 4 DDR features a mid-range Artix-7 XC7A100T FPGA with ample memory and logic capacity, widely adopted in university teaching labs.
- Basys 3 employs the more compact Artix-7 XC7A35T FPGA, representing the most resource-constrained and power-restricted target in this study.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

- The ZCU102 hosts the Zynq UltraScale+ XCZU9EG heterogeneous SoC, combining extensive programmable fabric with integrated ARM processor cores.

The work makes three key contributions. First, we put forward a fully pipelined, resource-efficient hardware architecture for LeNet-based CNN inference described entirely in synthesizable Verilog. Second, a fixed-point quantization scheme is applied, yielding minimal dynamic energy draw while sustaining a 98.29% recognition rate on MNIST. Third, post-synthesis evaluation featuring thorough analysis of resource usage and power dissipation is carried out on two widely adopted Artix-7 chips using Vivado, and additionally verified through behavioral simulation on the UltraScale+ platform.

II. Related Work

A. FPGA-based CNN Accelerators

Over the last ten years, considerable effort has been directed toward speeding up convolutional neural networks on FPGA fabric. Farabet et al., for instance, demonstrated that real-time operation of convolutional feature extractors is achievable on FPGA hardware. The designs of DianNao family and MIT's Eyeriss demonstrate that proper data flow design and effective memory management on chip are crucial to reduce the amount of off-chip accesses which are quite energy-consuming in the domain of deep learning.

Zhang et al. further contributed a roofline-model-guided optimization for FPGA-based CNNs, underscoring the benefits of on-chip block RAM for buffering both filters and feature maps. Qiu et al. built a Zynq-based accelerator targeting VGGNet and AlexNet that delivers notable speedup over embedded CPU baselines. More recent efforts, including those by Guo et al. and the Angel-Eye team, focus on automated compilation flows that translate TensorFlow or Caffe models directly into FPGA bitstreams.

B. Low-Power Quantization Techniques

Quantization ranks among the primary techniques for reducing computational complexity and energy expenditure during neural network inference. Although IEEE-754 single-precision floating-point operations deliver peak accuracy, they impose a heavy hardware cost on FPGA platforms. Employing 8-bit or 16-bit fixed-point formats has proven capable of attaining classification accuracy on par with the floating-point counterpart (up to a single percent difference), while requiring significantly less silicon area and lower switching activity [5].

Courbariaux et al. advocated the use of binary and ternary weights, showing satisfactory accuracy levels in compact architectures [8]. Han et al. introduced the Deep Compression pipeline—combining pruning, quantization, and Huffman coding—to achieve up to 40-fold compression enabling deployment on memory-constrained embedded systems [4]. When targeting MNIST on FPGAs, 8-bit fixed-point multiplication is a popular choice because it maps efficiently onto the DSP48E1 hard multiplier blocks found in Artix-7 and Kintex-7 devices.

C. Research Gaps

While numerous studies concentrate on either maximizing throughput or boosting accuracy, minimizing system-level power on budget-friendly Artix-7 boards—with openly reported Vivado power figures—remains relatively unexplored. A substantial portion of published designs target mid-tier or premium FPGAs like Zynq or Kintex families, complicating direct comparisons with low-cost alternatives. An additional gap in the literature is the scarcity of cross-platform portability studies.

This work aims to tackle these shortcomings by deploying the proposed architecture across several FPGA platforms including Artix-7 XC7A35T, Artix-7 XC7A100T and Zynq UltraScale+ XCZU9EG. In-depth post-implementation power analysis paves the way for comparing this design against other architectures in subsequent studies.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

III. Methodology

A. LeNet Architecture

The inference engine follows the LeNet-1 architecture, tailored specifically for the MNIST handwritten-digit dataset. It receives 28×28-pixel grayscale input images and processes them through the following layer stack:

- Conv1: 4 kernels, 5x5, stride=1, output is 24x24x4
- Pool1: 2x2 maximum pool layer, stride=2, output is 12x12x4
- Conv2: 12 kernels, 5x5, stride=1, output is 8x8x12
- Pool2: 2x2 maximum pool layer, stride=2, output is 4x4x12
- Fully Connected: 192 inputs, 10 outputs, argmax

Every layer applies the Rectified Linear Unit (ReLU) activation, implemented by inspecting the sign bit of the fixed-point accumulator output. Weights are extracted from a software model trained in Python with Keras running on top of TensorFlow. These weights are then quantized into fixed-point notation and saved as hexadecimal entries inside the memory initialization files.

B. Hardware Architecture

The hardware design centers on a datapath orchestrated by a top-level finite state machine. This FSM governs the progression through the layer pipeline: it fetches input activations and filter coefficients from block RAM, steers the multiply-accumulate computation via a configurable array of processing elements, applies the chosen activation function, and writes the resulting output feature maps into intermediate buffers.

Convolutional layers rely on a sliding-window strategy. To produce each output pixel, the PE array computes the dot product of filter weights against the corresponding input window. Parallelism is governed by the NUM_PE parameter, which sets how many PEs operate concurrently. Each PE houses a dedicated multiply-and-accumulate unit built around the DSP48E1 primitive. By default, the current implementation instantiates five PEs, enabling five MAC operations every clock cycle.

The fully connected layer is realized as a serialized vector–matrix multiplication engine. Inputs are read from the S4 output buffer while weights stream out of BRAM through the MAC datapath. To keep LUT and DSP usage low, a single MAC datapath is time-shared across all neurons at the cost of extra latency. This trade-off works well given the modest size of LeNet's fully connected layer.

C. Fixed-Point Quantization

All arithmetic uses a 16-bit signed fixed-point format with 8 integer bits and 8 fractional bits. Floating-point weights from the pre-trained model are first scaled and then rounded to the closest representable fixed-point value. Accumulation registers are widened from 16 to 32 bits so that the running sum across multiple convolution input channels does not overflow. Right-shifting is employed to reduce the size of the accumulated result to 16 bits to feed it to next layer's input buffer. The fixed-point word width was selected after evaluating 8-bit, 16-bit, and 32-bit alternatives. The evaluation showed that a 16-bit data width paired with 32-bit accumulation delivers accuracy identical to the floating-point baseline while consuming far fewer FPGA resources than full 32-bit precision would.

D. Memory Architecture

On both the Artix-7 and UltraScale+ devices, block RAMs hold the network weights along with intermediate feature-map buffers. Vivado synthesizes each memory block with ram_style set to "block" yielding true dual-port memories that support concurrent read and write access for layer-by-layer pipelining.

BRAM capacity is distributed according to each layer's storage requirements. For example, Conv1 needs 100 entries in BRAM, Conv2 – 1,200 entries, and FC requires 1,920 weight entries + 10 bias entries. The remaining BRAMs hold the input feature map together with the outputs of C1, S2, C3, and S4. By separating storage this way, memory accesses never stall the compute pipeline, thereby boosting throughput.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

E. Multi-Platform Adaptation

A central objective of this research is to demonstrate that the inference-engine architecture can be ported across different FPGA families with only minor RTL adjustments. The platform-specific changes required are summarized below.

Nexys 4 DDR (XC7A100T): The baseline design targets this board, clocked from a single-ended 100 MHz oscillator, a UART receiver that ingests image data streamed from a host PC, an 8-digit seven-segment display for visual feedback, plus 16 user LEDs. The UART baud-rate divisor is set to 868 clock cycles per bit, corresponding to 115 200 baud.

Basys 3 (XC7A35T): To meet the tighter resource budget of the smaller Artix-7 device, the seven-segment display is trimmed from 8 digits down to 4, the reset polarity switches from active-low to active-high to match the BTNR push-button and Vivado's Flow_AreaOptimized_high synthesis strategy is applied. All other parameters, including the 100 MHz clock and baud-rate divisor, stay the same.

ZCU102 (XCZU9EG): This board carries a high-end UltraScale+ FPGA. A pair of IBUFDS primitives is inserted to accept the differential 125 MHz clock from the on-board SI570 oscillator. The baud-rate divisor is recalculated to 1085 cycles to match the higher clock rate. Since the ZCU102 lacks a seven-segment display, a UART transmitter module is added so the predicted digit class is sent back as an ASCII character.

F.Design Flow

The complete circuit was described in Verilog HDL and then synthesized, placed-and-routed, and verified inside the Xilinx Vivado 2025.2 tool suite. Target clock rates were 100 MHz for the Artix-7 boards and 125 MHz for the UltraScale+ board. After place-and-route, power estimation was performed with Vivado's on-chip power analysis using vectorless activity simulation. [6]

IV. RESULTS AND ANALYSIS

A. Software Model Accuracy

A Python implementation of the LeNet architecture was built using Keras on top of TensorFlow, trained on the MNIST collection comprising 60,000 training images and 10,000 test images. Training ran for 10 epochs with the Adam optimizer at a 0.001 learning rate and categorical cross-entropy loss, reaching 98.29% accuracy and 0.0574 validation loss—comfortably surpassing the 96% target.

Once training finished, a Python utility converted the floating-point parameters into hexadecimal fixed-point values. These fixed-point parameters were subsequently loaded for inference, confirming that the accuracy drop relative to the original floating-point values is negligible.

B. Resource Utilization

Table I lists the post-implementation resource usage for the two Artix-7 target devices, drawn from Vivado's place-and-route reports. The ZCU102 figures are estimates derived from synthesizable RTL, since full synthesis demands an enterprise license that was unavailable for this academic work.

TABLE I: RESOURCE UTILIZATION COMPARISON ACROSS THREE FPGA PLATFORMS

Resource	Nexys 4 (XC7A100T)	Available	Util. %	Basys 3 (XC7A35T)	Available	Util. %	ZCU102 (XCZU9EG)	Available	Util. %
LUT	1,990	63,400	3.14 %	2,029	20,800	9.75 %	~2,000 (est.)	274,080	<1%
FF	4,738	126,800	3.74 %	4,849	41,600	11.66 %	~4,800 (est.)	548,160	<1%
BRAM	9.50	135	7.04 %	9.50	50	19.00 %	~10 (est.)	912	~1%

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

DSP	3	240	1.25 %	9	90	10.00 %	~9 (est.)	2,520	<1%
I/O	40	210	19.05 %	33	106	31.13 %	~20 (est.)	328	~6%

Comparing the synthesis outputs yields several noteworthy observations. LUT counts are broadly comparable—1,990 versus 2,029—suggesting that the core datapath architecture ports across with only marginal overhead. The modest rise on Basys 3 stems from the area-driven synthesis flow in Vivado, which replicates certain logic to simplify routing on the smaller fabric.

DSP slice usage varies by target: 3 slices on Nexys 4 vs. 9 on Basys 3. On Basys 3 the area-driven strategy pushes every multiplication onto hard DSP slices to free up LUTs. On Nexys 4, the default synthesis approach lets some multiplications remain in fabric logic without sacrificing functionality. Basys 3 is the most resource-constrained of the three targets: BRAM occupancy hits 19%, and I/O pin usage reaches 31.13%. Even so, the design fits comfortably because over 80% of every resource category remains available.

C. Power Analysis

Table II presents the post-implementation power analysis results obtained from Vivado's on-chip power estimator for both Artix-7 platforms [6].

TABLE II: ON-CHIP POWER COMPARISON (VIVADO POST-IMPLEMENTATION)

Parameter	Nexys 4 (XC7A100T)	Basys 3 (XC7A35T)
Total On-Chip Power	0.135 W	0.139 W
Dynamic Power	0.037 W (28%)	0.041 W (30%)
— Clocks	0.012 W (32%)	0.012 W (29%)
— Signals	0.002 W (6%)	0.003 W (7%)
— Logic	0.001 W (3%)	0.001 W (3%)
— BRAM	0.018 W (48%)	0.018 W (43%)
— DSP	0.001 W (3%)	0.002 W (4%)
— I/O	0.003 W (8%)	0.006 W (14%)
Device Static Power	0.098 W (72%)	0.098 W (70%)
Junction Temperature	25.6°C	25.6°C
Thermal Margin	59.4°C (12.9 W)	59.4°C (12.9 W)

We can see that both boards pull similar amounts of power, right around 0.135 W and 0.139 W, respectively. This makes sense because both of them are built on the same 28 nm Artix-7 technology. The slight bump in power on the Basys 3 comes from the I/O doing a bit more work (0.006 W versus 0.003 W). mainly because of the extra load from driving the seven-segment display and the LEDs.

When looking at what uses dynamic power, accessing the BRAM takes up the most in both scenarios (0.018 W) which just shows how much memory the design needs to keep the CNN weights and data right there on the chip. Spreading the clock signal around takes up 0.012 W. The logic switching barely uses any power, just 0.001 W, because the fixed-point math does not need to flip bits very often.

The baseline static power sits at 0.098 W, which makes up about 70 to 72 percent of the totally power used by the chip This is just a normal property of the 28 nm Artix-7 process itself. In turn, the junction temperature equals 25.6°C when the ambient air temperature is 25°C leaving a comfortable margin of 59.4°C. Because of this, the design runs totally fine without getting too hot on either board. We do not even need a heat sink or a fan since it draws so little power overall.

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

D. Timing Performance

Table III shows a quick summary of the timing results we got from the Vivado post-routing report's post-routing timing report.

TABLE III: TIMING PERFORMANCE SUMMARY

Parameter	Nexys 4 (100 MHz)	Basys 3 (100 MHz)	ZCU102 (125 MHz)
Target Clock Period	10.0 ns	10.0 ns	8.0 ns
Worst Negative Slack (WNS)	−3.401 ns	To be confirmed	Simulation only
Worst Hold Slack (WHS)	0.054 ns	To be confirmed	Simulation only
UART Baud Rate	115,200	115,200	115,200
UART Clocks per Bit	868	868	1,085
Estimated Inference Latency	~120 µs	~120 µs	~96 µs

Worst Negative Slack for Nexys 4 design comes to −3.401 ns for clock rate of 100 MHz. Basically, the longest path—spanning the multiply-accumulate units and the BRAM read pipeline—exceeds the 10 ns clock period by roughly 3.4 ns. But out of the 11,219 points checked, only 22 actually break this rule. There are a few ways to fix this: we could drop another pipeline register into the calculation path, or we could simply slow the clock down to 75 MHz (still adequate for real-time digit classification), or enabling retiming during Vivado synthesis. Hold-time and pulse-width checks pass on every target device.

When running the ZCU102 at 125 MHz, it should finish guessing each digit about 20 percent faster than the 100 MHz Artix-7 boards do. So, processing just one image should take right around 96 microseconds.

E. Behavioral Simulation Validation

To make sure the ZCU102 version worked, we ran it through a behavioral simulation using Vivado XSim. We could not run a full synthesis because the XCZU9EG chip needs a paid Vivado Enterprise license, which our academic institution does not hold. In the testbench, we load up a ready-to-go image of the number 7 directly into the input buffer, populates the weight BRAMs with known data, and triggers the inference FSM.

Looking at the simulation results, we can see the FSM stepping through each stage exactly like it should: T_IDLE → T_C1 (convolution 1) → T_S2 (pooling 1) → T_C3 (convolution 2) → T_S4 (pooling 2) → T_FC (fully connected) → T_DONE. the inference_done flag asserts high and the LED register reflects the predicted digit. This proves that our setup for the UltraScale+—differential clock buffering, revised UART timing, and updated I/O interface—actually works the way we wanted it to.

F.Cross-Platform Comparison Summary

Table IV, we put together a comparison of all three boards, intended to help designers choose a suitable FPGA board for embedded CNN inference.

TABLE IV: CROSS-PLATFORM COMPARISON SUMMARY

Parameter	Nexys 4 DDR	Basys 3	ZCU102
FPGA Part	XC7A100T	XC7A35T	XCZU9EG
FPGA Family	Artix-7	Artix-7	Zynq UltraScale+
Approximate Board	$260	$150	$3,000+

International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue:2 Volume No.10 March - April – 2026
DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

Cost			
System Clock	100 MHz (SE)	100 MHz (SE)	125 MHz (Diff)
Total On-Chip Power	0.135 W	0.139 W	Est. ~0.5 W
Dynamic Power	0.037 W	0.041 W	Est. ~0.15 W
LUT Utilization	3.14%	9.75%	<1% (est.)
BRAM Utilization	7.04%	19.00%	~1% (est.)
Display Interface	8-digit 7-segment	4-digit 7-segment	UART TX + 8 LEDs
Result Output	LEDs + Display	LEDs + Display	LEDs + UART ASCII
ARM Processor Available	No	No	Yes (Quad A53)
Synthesis License	Free (WebPACK)	Free (WebPACK)	Enterprise Required
Implementation Status	Full bitstream	Full bitstream	Simulation verified

Out of the three we looked at, the Basys 3 is definitely the cheapest way to get LeNet running, costing around $150. It fits the whole accelerator just fine and only uses 4 mW more dynamic power than the more expensive Nexys 4. At $260, the Nexys 4 strikes the best balance between available resources, on-board peripherals, and room for future growth. Even though the ZCU102 is way overkill for just running LeNet, it leaves plenty of room for much larger networks and lets you mix software and hardware coding using its built-in ARM cores.

G. Comparison with Related Works

Table V compares our accelerator against some other recent FPGA CNN projects that do similar jobs.

TABLE V: COMPARISON WITH STATE-OF-THE-ART FPGA CNN ACCELERATORS

Work	Platform	Power (W)	LUT	Test Accuracy
Zhang [1]	Zynq XC7Z020	0.31	4,521	98.2%
Qiu [2]	Artix-7 XC7A100T	0.22	3,104	97.8%
Guo [5]	Kintex-7	0.19	2,870	98.5%
This Work (Nexys 4)	Artix-7 XC7A100T	0.135	1,990	98.29%
This Work (Basys 3)	Artix-7 XC7A35T	0.139	2,029	98.29%

Compared to the rest of them, our design uses the least amount of on-chip power—0.135 W on Nexys 4—and the smallest LUT footprint—1,990. Getting 98.29 percent accuracy means it holds its own against what others have published, falling short of the very best result by only a tiny 0.21 percent margin. And the best part is we managed all this on the cheapest Artix-7 boards available.

V. CONCLUSION

In this paper, we showed how to build a low-power hardware version of the LeNet model specifically for recognizing MNIST digits. By using fixed-point math, setting up a pipelined array of processing elements, and being smart about how we use the Block RAM, on-chip power consumption reaches 0.135 W on Nexys 4 (XC7A100T) and 0.139 W on Basys 3 (XC7A35T). The dynamic part of that power draw never goes above 0.037 W and 0.041 W, respectively.

DOI:10.46647/ijetms.2026.v10i02.027 ISSN: 2581-4621

The final design is really tight on space, taking up only about 2,000 LUTs, 4,800 flip-flops, 9.5 BRAM tiles, and 3–9 DSP elements on the two Artix-7 targets, It never uses more than 20 percent of any single resource limits—even on the smallest XC7A35T. The model we trained in software hits 98.29 percent accuracy on the MNIST test data, and moving it to fixed-point hardware keeps that exact same accuracy without losing anything.

A portability trial spanning three FPGA boards—priced from roughly $150 to over $3,000—demonstrates that the accelerator's core RTL is highly portable, We only had to change a few small things at the top level (clock buffering, display driver, and UART configuration). No computational module—convolution, pooling, or fully connected layer—had to be changed at all when swapping boards.

In the future, we could save even more power by adding clock gating and adjusting the voltage and frequency on the fly. We could also try expanding the accelerator to handle bigger networks like LeNet-5 and slim VGGNet variants through layer-by-layer mapping. Lastly, bringing in the ARM cores from the ZCU102 platform's ARM cores enables software/hardware co-design with pre/post-processing performed on the CPU part.

VI. REFERENCES

[1] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), Monterey, CA, USA, 2015, pp. 161–170.

[2] J. Qiu et al., "Going deeper with embedded FPGA platform for convolutional neural network," in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), Monterey, CA, USA, 2016, pp. 26–35.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in Proc. Int. Conf. Learning Representations (ICLR), San Juan, Puerto Rico, 2016.

[5] Y. Guo, "A survey on methods and theories of quantized neural networks," arXiv preprint, arXiv:1510.00149, 2018.

[6] Xilinx Inc., "Vivado Design Suite User Guide: Power Analysis and Optimization," UG907 (v2023.1), 2023.

[7] Digilent Inc., "Basys 3 Artix-7 FPGA Trainer Board Reference Manual," Rev. C, 2023.

[8] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," in Proc. ICLR Workshop, San Diego, CA, USA, 2015.

[9] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "NeuFlow: A runtime reconfigurable dataflow processor for vision," in Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops (CVPRW), Colorado Springs, CO, USA, 2011, pp. 109–116.

[10] Digilent Inc., "Nexys 4 DDR Artix-7 FPGA Trainer Board Reference Manual," Rev. D, 2022.

[11] Xilinx Inc., "ZCU102 Evaluation Board User Guide," UG1182 (v1.6), 2022.

[12] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits, vol. 52, no. 1, pp. 127–138, Jan. 2017.