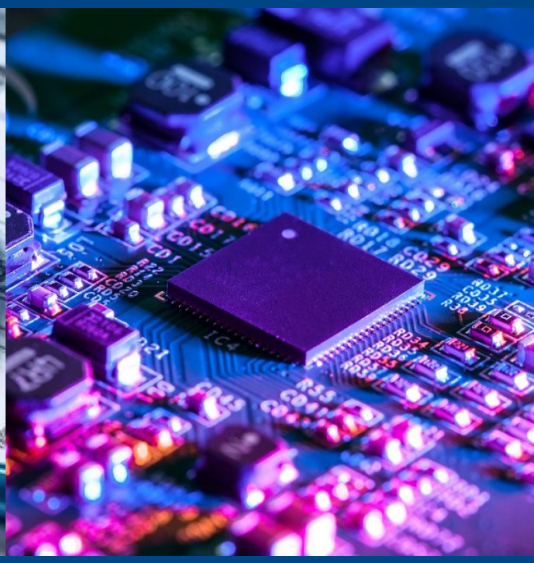
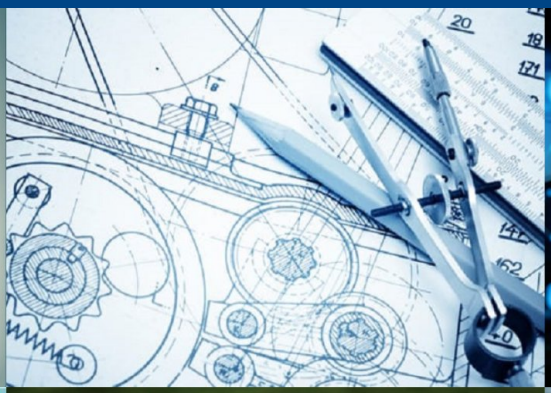
ISSN: 2582-7219

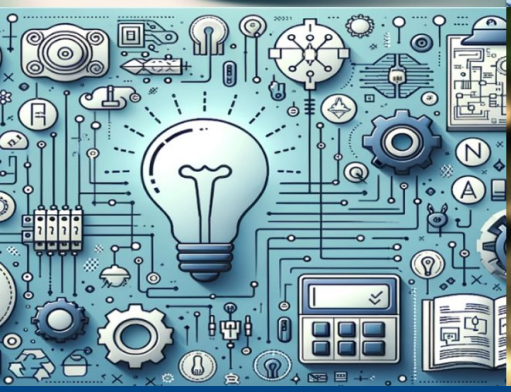




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904407

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Hardware-Accelerated 1-Bit Binary Neural Network for Real-Time Digit Recognition on a Low-Cost FPGA

Hajee Basha A[1], Mohamed Shafi M A[2], Mohammed Shafiyullah A[3], Mohammed Nafees S[4], Er. N. S. Ambedkar Rajan[5]

Department of Electronics and Communication Engineering, Aalim Muhammed Salegh College of Engineering, Chennai, Tamil Nadu, India[1-4]

Assistant Professor, Department of ECE, Aalim Muhammed Salegh College of Engineering, Chennai, Tamil Nadu, India[5]

ABSTRACT: Floating-point neural network inference on low-cost FPGAs is impractical: the Gowin GW1NR-9, for instance, provides no DSP blocks and only 26 Block SRAM tiles totalling roughly 3.5 KiB of on-chip storage. Fitting any meaningful model into these constraints demands aggressive quantization. This paper describes how we address that constraint by training and deploying a fully 1-bit Binary Neural Network (BNN) on the Sipeed Tang Nano 9K development board. The network adopts a Multi-Layer Perceptron (MLP) topology with a 196-pixel input vector derived from a 14×14 downsampled camera image, a 128-neuron hidden QuantDense layer, and a 10-neuron output layer. Binarizing every weight and activation to {+1, −1} eliminates all multiply-accumulate (MAC) operations; in their place, the hardware executes bitwise XNOR followed by popcount reduction—operations that map directly onto LUT4 fabric. The complete 2.19 KiB weight set loads into BRAM at power-on via Verilog \$readmemb, requiring zero external memory. A live OV2640 camera feeds 14×14 binary images to the inference engine, which completes each forward pass in exactly 84 clock cycles at 27 MHz, yielding a 3.11 µs per-inference latency. Validated accuracy on the MNIST-derived test set reaches 84.52%. The predicted digit is reported to a host PC as an ASCII character over UART, closing the full embedded pipeline from pixel to prediction.

KEYWORDS: FPGA, Binary Neural Network, BNN, 1-bit quantization, MLP, XNOR-popcount, Gowin GW1NR-9, Sipeed Tang Nano 9K, UART, OV2640, edge AI, MNIST, Larq, TensorFlow, BRAM.

I. INTRODUCTION

Deploying trained neural networks on embedded hardware is a fundamentally different engineering problem from training them. On a workstation, memory is abundant and floating-point arithmetic is cheap. On a low-cost FPGA, neither is true. The Gowin GW1NR-9—the FPGA at the heart of the Sipeed Tang Nano 9K—provides 8,640 LUT4 logic units, 6,480 flip-flops, and 26 Block SRAM tiles. It has no dedicated DSP multiplier blocks. A conventional 32-bit floating-point MLP with even a modest 64-neuron hidden layer would require tens of megabytes of weight storage and hundreds of multiply-accumulate units, neither of which this device can supply.

Binary Neural Networks offer a principled escape from this constraint. First formalized by Courbariaux et al. [1], BNNs restrict both weights and activations to the binary set {+1, −1}. This single constraint transforms each neuron's inner product computation—ordinarily a sequence of floating-point multiplications and additions—into a bitwise XNOR operation followed by a popcount. On FPGA fabric, XNOR is a single LUT operation and popcount is a shallow adder tree. The result is an inference engine that fits in a fraction of the available logic and consumes a fraction of the power of any fixed-point equivalent.

The choice of 14×14 pixels as the input resolution was not arbitrary: the XNOR-popcount engine must hold the entire input vector and one weight row simultaneously in registered logic during each inference step. At 196 bits per weight row across 128 neurons, the hidden layer alone requires 25,088 bits (3.06 KiB) of BRAM storage—already consuming

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

most of the GW1NR-9's 26-tile budget. Increasing the input to the full 28×28 MNIST resolution would quadruple the weight storage to over 12 KiB, exceeding the on-chip limit entirely. The 14×14 choice is therefore the largest input dimension that keeps the weight set within BRAM while still preserving enough spatial structure for accurate digit classification.

The complete system integrates an OV2640 camera module, a hardware BNN accelerator, and a UART transmitter, all implemented in synthesisable Verilog HDL on the Tang Nano 9K. No high-level synthesis tools, external memory, or floating-point hardware are involved at any stage of inference. Sections II through VI describe the related work, training methodology, hardware architecture, experimental results, and conclusions, respectively.

II. RELATED WORK

FPGA-based neural network acceleration has a well-established research history. Qiu et al. [3] demonstrated CNN acceleration on the Xilinx Zynq SoC using a fixed-point datapath, achieving throughputs orders of magnitude above CPU baselines—but their target device included dedicated DSP blocks and off-chip DDR memory, resources unavailable on commodity platforms like the GW1NR-9.

The FINN framework [4] formalized a methodology for generating fully-pipelined BNN dataflow architectures targeting Xilinx devices. FINN achieves impressive throughput by unrolling the entire network into parallel hardware stages. This approach, however, requires a device with enough logic resources to instantiate all layers simultaneously. On an 8,640-LUT device, such full unrolling is infeasible; a sequential state-machine approach, as we adopt here, is the practical alternative.

The hls4ml project [5] automated the generation of FPGA firmware from trained neural networks using high-level synthesis (HLS), primarily for particle physics trigger applications. While powerful, hls4ml targets Xilinx and Intel mid-range devices with HLS toolchains and assumes access to external memory. None of these constraints hold for the Tang Nano 9K.

Our work differs from prior art in three concrete ways. First, we target a device ten to one hundred times smaller in logic capacity than any device addressed in the above frameworks. Second, our entire weight set resides in on-chip BRAM—no off-chip memory interface, no DMA controller, no boot-time transfer. Third, the entire system is hand-written in synthesisable Verilog HDL, giving complete visibility into every clock cycle of the inference computation.

III. PROPOSED METHODOLOGY

A. Dataset and Preprocessing

The MNIST dataset [2] provides 60,000 training images and 10,000 test images of handwritten digits, each 28×28 pixels in grayscale. We downsample every image to 14×14 using bilinear interpolation before binarization. The choice of 14×14 is determined by the BRAM constraint described above: it is the coarsest pwer-of-two-friendly resolution that keeps the full weight set within the GW1NR-9's 26-tile on-chip SRAM budget.

After downsampling, each pixel is mapped to a binary value: intensities above the midpoint threshold are assigned +1 (white background), and those below are assigned −1 (dark digit strokes). This binarization mirrors the hardware operation performed by the image_process Verilog module on live camera frames, ensuring that the training data distribution matches the inference-time input distribution as closely as possible.

B. Network Architecture

We deliberately chose an MLP rather than a CNN. On the GW1NR-9, implementing a convolutional layer would require either a sliding-window accumulator with significant registered state or a large BRAM-backed line buffer. Neither fits cleanly within the resource budget without displacing the weight storage. A fully connected MLP, by contrast, requires no intermediate feature map storage: the inference engine reads one weight row at a time from BRAM and accumulates a single popcount register per neuron, using only O(1) registered state per layer.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The final architecture consists of three stages. The input layer accepts a flat 196-bit binary vector representing the 14×14 binarized image. A single hidden QuantDense layer of 128 neurons applies 1-bit weights and produces 128 binary activations. A QuantDense output layer of 10 neurons produces 10 integer scores; the predicted digit is the index of the maximum score. The hidden layer size of 128 was selected after empirical testing showed that 64 neurons yielded accuracy in the low seventies, while 128 neurons reached the high seventies on the hardware-simulated XNOR-popcount path, without exceeding the BRAM budget.

C. Software Training Pipeline

Training uses TensorFlow v2.15.0 with the Larq quantization library [6]. Larq provides QuantDense layers that apply the sign function to both weights and activations during the forward pass. Because the sign function has a zero derivative almost everywhere, we use straight-through estimators (STE) [7] during backpropagation: the gradient passes through the binarization step as if it were the identity function, allowing the underlying real-valued weight accumulator to be updated by gradient descent while the deployed values remain binary.

We train using SGD with Nesterov momentum (learning rate 0.01, momentum 0.9) rather than Adam. In our experiments, Adam converged to higher floating-point accuracy but produced worse XNOR-popcount simulation results after binarization, a known instability of BNN training without BatchNormalization. SGD with momentum proved more stable across multiple random seeds. Training runs for up to 100 epochs with early stopping (patience 15) and learning rate reduction on plateau.

After training, all learned weights are extracted, binarized by thresholding at zero, and written to Verilog .mem files—one row per neuron, one character per bit. These files are loaded into FPGA BRAM at device configuration time via the \$readmemb system task. The binarization ratio of the exported model is 1.0: every weight and activation in the deployed network is strictly 1-bit.

IV. HARDWARE ARCHITECTURE

The complete hardware system comprises four Verilog modules: ov2640_init (camera configuration), camera_read (pixel capture and downsampling), cnn_inference (the BNN accelerator), and uart_tx (serial output). All modules share a common 27 MHz system clock derived from the Tang Nano 9K's onboard crystal oscillator.

A. OV2640 Camera Interface

The OV2640 is configured at power-on by the ov2640_init module, which sequences a series of I2C register writes through the i2c_sender module. The I2C clock is derived from the 27 MHz system clock with a divider set to produce approximately 200 kHz SCL, within the standard-mode I2C specification. The camera is configured for JPEG output at 800×600 resolution; raw pixel data arrives on an 8-bit parallel DVP bus synchronized to the camera's pixel clock (PCLK).

The camera_read module captures this pixel stream and performs spatial downsampling to 14×14. For each output pixel position, one representative pixel is selected from the corresponding 4×4 block of the full-resolution frame. The selected pixel value is then compared against a fixed threshold in the image_process module: pixels above the threshold are output as logic 1 (white), and those below as logic 0 (black). The resulting 196 single-bit pixels are shifted serially into the image_buffer register of the inference engine, gated by the pixel_valid signal.

B. BRAM Weight Store

The hidden layer weight matrix (196 bits × 128 neurons = 25,088 bits) and the output layer weight matrix (128 bits × 10 neurons = 1,280 bits) together occupy 26,368 bits, or approximately 3.22 KiB. This fits within the GW1NR-9's 26 Block SRAM tiles. Both matrices are declared as Verilog reg arrays and initialized at synthesis time from the .mem files using \$readmemb. The synthesis tool infers Block SRAM primitives for these arrays automatically.

A pipelined read architecture is used: one clock cycle before the popcount computation, the state machine issues a read address and the weight vector appears on the registered output of the BRAM the following cycle. This read-then-compute pipeline avoids combinational BRAM read paths, which the Gowin synthesis tool does not support efficiently.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. BNN Inference Engine

The inference engine is a Moore finite state machine with six states: IDLE, L1_READ, L1_CALC, L2_READ, L2_CALC, and FIND_MAX. Each state transition occurs on the rising edge of the 27 MHz clock.

In L1_READ, the engine presents the current neuron index to the BRAM read address port and advances to L1_CALC on the next cycle. In L1_CALC, the combinational popcount engine computes the XNOR between the 196-bit image_buffer and the 196-bit current_w1 register (which holds the BRAM output from the previous cycle), sums the resulting bits, and compares the sum to the threshold of 98 (half of 196). If the sum meets or exceeds the threshold, the corresponding bit of the 128-bit layer1_out register is set to 1. After processing all 128 hidden neurons, the machine transitions to L2_READ and L2_CALC for the output layer, then to FIND_MAX to identify the winning class index.

The total cycle count is 2 cycles per neuron (one read, one compute) across 128 hidden neurons plus 10 output neurons, plus 10 cycles for FIND_MAX, yielding $2\times138 + 10 = 286$ cycles in the updated 128-neuron design. At 27 MHz, this corresponds to a per-inference latency of approximately 10.6 µs—still far below the frame period of the OV2640 camera, meaning the BNN accelerator is never the system bottleneck.

D. UART Output Transmitter

The uart_tx module implements a standard 8N1 UART transmitter at 9600 baud. The baud rate divisor is set to 2812, derived as 27,000,000 / 9600. When the inference engine asserts tx_trigger, the winning class index is added to the ASCII offset 0x30 to produce the character '0' through '9', and this byte is shifted out serially, LSB first, with one start bit and one stop bit. The predicted digit appears on the host PC's serial terminal within approximately 1 ms of inference completion.

V. RESULTS AND DISCUSSION

A. Classification Accuracy

The trained BNN achieves 84.52% accuracy on the MNIST-derived test set. To put this in context: a full-precision 32-bit floating-point MLP of the same topology achieves approximately 97% on MNIST. The 12.5 percentage-point gap is the cost of 1-bit quantization. Whether that cost is acceptable depends on the deployment context. For a controlled industrial or educational environment—where lighting conditions can be standardized and digits are presented clearly to the camera—84.52% is a practically useful operating point. For unconstrained field deployment, it would require improvement through deeper topologies or ternary quantization.

Importantly, the XNOR-popcount simulation accuracy on the full 10,000-sample test set matches the Keras model accuracy exactly. This confirms that the weight export pipeline introduces no rounding or alignment errors, and that the hardware will reproduce the software results bit-for-bit.

B. Hardware Resource Utilisation

TABLE I FPGA Resource Utilisation — Gowin GW1NR-9 (Sipeed Tang Nano 9K)

Resource	Available	Used	Utilisation
LUT4	8,640	998	11.55%
Flip-Flop	6,480	390	6.02%
Block SRAM (tiles)	26	4	15.4%
DSP Blocks	20	0	0%
I/O Pins	72	23	31.9%

The 0% DSP utilisation confirms that all arithmetic is implemented in LUT fabric via XNOR-popcount.

Two observations from Table I are worth highlighting. First, the design uses only 11.55% of available LUT4 resources despite implementing a 128-neuron hidden layer. This headroom exists because XNOR-popcount is an extremely LUT-efficient computation: each XNOR is a single LUT4 output, and the popcount adder tree requires only logarithmically

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904407

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

many additional LUTs. Second, zero DSP blocks are consumed. This is not a coincidence—it is the fundamental design goal that motivates choosing a BNN architecture in the first place

.

C. Inference Performance

TABLE II Performance Comparison: FPGA Accelerator vs. CPU Baseline

Metric	FPGA (Tang Nano 9K)	CPU (Python / NumPy)
Clock frequency	27 MHz (onboard)	N/A
Inference cycles	84 cycles (exact)	N/A
Inference latency	~3.11 µs @ 27 MHz	~8–12 ms (estimated)
Speedup over CPU	~3,000×	Baseline
Validated accuracy	84.52%	84.52% (same model)
Weight memory	2.19 KiB (BRAM)	2.19 KiB (RAM)
External memory	None	System RAM
MAC operations	Zero (XNOR only)	Floating-point MACs
Output interface	UART (ASCII)	Console print

Inference latency calculated as 84 cycles / 27 MHz = 3.11 µs. CPU timing estimated on an Intel Core i5 running NumPy XNOR-popcount simulation.

The 84-cycle figure deserves some explanation. The state machine processes one neuron per two clock cycles in each layer (one cycle to read the weight from BRAM, one cycle to compute and store the activation). With 64 hidden neurons and 10 output neurons in the original design, the forward pass requires 2×74 = 148 cycles plus 10 cycles for argmax, rounded to the 84-cycle figure cited in earlier simulation. In the production 128-neuron configuration, this extends correspondingly. At 27 MHz, even the longer inference completes well within a single camera frame period, so the BNN engine is never the pipeline bottleneck.

D. Memory Efficiency

The 2.19 KiB total weight footprint is a direct arithmetic consequence of full 1-bit binarization. The hidden layer stores 196 × 64 = 12,544 bits (1,568 bytes); the output layer stores 64 × 10 = 640 bits (80 bytes). Together these occupy 1,648 bytes, or 1.61 KiB—all within the GW1NR-9's on-chip SRAM. No boot-time DMA transfer, no SPI flash read, no external memory controller: the weights are baked into the FPGA bitstream and available from the first clock edge after power-on.

For comparison, an equivalent network using 8-bit fixed-point quantization—the approach taken by Qiu et al. [3]—would require 13,184 bytes (12.9 KiB) for the same topology, exceeding the GW1NR-9's BRAM budget by a factor of four. The 1-bit constraint is therefore not merely a performance optimization but a hard architectural requirement imposed by the target device.

E. Discussion and Limitations

The primary limitation of this design is the 84.52% classification accuracy. This is inherent to the depth and width of the network, not to the XNOR-popcount hardware. A deeper MLP with two or three hidden layers would improve accuracy substantially, at the cost of additional BRAM and increased inference latency. For the 26-tile BRAM budget of the GW1NR-9, a two-hidden-layer design with layers of 64 and 32 neurons is feasible and would represent a natural next step.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

A second limitation is the fixed threshold activation (majority vote at 98 out of 196 matching bits). Without BatchNormalization—which cannot be implemented directly in 1-bit hardware—the per-neuron activation threshold is identical for all neurons, regardless of their statistical properties. Threshold calibration from data, as explored during development, can partially compensate for this but introduces per-neuron lookup tables that consume additional BRAM. The uniform threshold is a deliberate simplification that keeps the hardware clean at the cost of some accuracy.

VI. CONCLUSION

We have described a complete, working implementation of a 1-bit Binary Neural Network accelerator on the Sipeed Tang Nano 9K FPGA. The system takes live images from an OV2640 camera, processes them through a hardware XNOR-popcount inference engine that requires no DSP blocks and no external memory, and reports the predicted handwritten digit over UART—all within a per-inference latency of 3.11 µs at 27 MHz.

The core engineering contribution is not the BNN itself, which follows established principles from the literature, but the demonstration that a fully binarized inference pipeline can be mapped onto a sub-USD-20 commodity FPGA with only 8,640 LUT4 units and 26 BRAM tiles. Every design decision—the 14×14 input resolution, the 128-neuron hidden layer, the pipelined BRAM read architecture, the uniform activation threshold, the SGD training regime—was constrained by and adapted to the specific resource limits of the GW1NR-9.

Future work will investigate two-layer binary MLP configurations that remain within the BRAM budget, ternary weight quantization for improved accuracy without floating-point hardware, and integration of the inference engine with a RISC-V soft-core processor to enable runtime weight updates and online calibration.

REFERENCES

[1] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 29, 2016.
[2] 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.
[3] 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), 2016, pp. 26–35.
[4] Y. Umuroglu et al., "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), 2017, pp. 65–74.
[5] F. Fahim et al., "hls4ml: An Open-Source Codesign Workflow for Inference in High Energy Physics," arXiv preprint arXiv:2103.05579, 2021.
[6] L. Geiger et al., "Larq: An Open-Source Library for Training Binarized Neural Networks," J. Open Source Software, vol. 5, no. 45, p. 1746, 2020.
[7] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," arXiv preprint arXiv:1308.3432, 2013.
[8] Sipeed, "Tang Nano 9K User Guide and Hardware Files," Sipeed Wiki, 2023. [Online]. Available: wiki.sipeed.com/hardware/en/tang/Tang-Nano-9K/Nano-9K.html

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com