Ternary-Weight MLP for FPGA-Based Buck Converter Control

Pramodh G
Electronics and Communication
REVA University
Bangalore, India
pramodhgandhinathan2013@gmail.com

Sayantam Sarkar
Electronics and Communication
REVA University
Bangalore, India
sayantam.61@gmail.com

***Abstract*—Deploying full-precision multilayer perceptron (MLP) controllers for DC–DC converters on low-cost edge FPGAs is constrained by look-up table (LUT) and digital signal processor (DSP) budgets, particularly when fully-parallel datapaths are required at the converter switching rate. We present a ternary-weight MLP for buck converter duty-cycle control in which every weight is constrained to $\{-1, 0, +1\}$, mapping the forward pass to wires, inverters, and pruned connections at synthesis time. The network is trained by behavioral cloning from a Tustin-discretized lag compensator and realized as Q15.16 synthesizable Verilog on a Xilinx Zynq-7020, with closed-loop verification through System Generator co-simulation. The ternary core uses 2,830 LUTs (5.3%) and zero DSPs, versus 84,769 LUTs (159%) and 70 DSPs for an architecturally matched Q15.16 baseline that does not fit on the device. These results indicate that ternary weight quantization can enable fully-parallel neural converter control on low-cost edge FPGAs.**

***Index Terms*—FPGA, neural network hardware, ternary weight quantization, DC–DC converters, buck converter control, behavioral cloning.**

I. Introduction

Field-programmable gate arrays (FPGAs) are a preferred substrate for high-frequency DC–DC converter control thanks to their parallel computation, deterministic timing, and reconfigurability. Classical compensators designed from averaged plant models [3] dominate practice but require manual tuning and assume linear plant behavior. Neural controllers offer a model-agnostic alternative [1], [2], but a fully-parallel multilayer perceptron (MLP) in fixed-point arithmetic readily consumes tens of thousands of look-up tables (LUTs) and dozens of digital signal processor (DSP) blocks—beyond the budget of the low-cost edge FPGAs typically targeted for converter control. This work treats such an MLP as a given target and addresses its efficient hardware realization, not the merits of neural over classical control.

Ternary weight quantization $\{-1, 0, +1\}$ reduces every multiplication to a sign change or zero selection, allowing the synthesis tool to map the forward pass onto LUT fabric with zero DSP usage [5], [10]. Ternary-FPGA implementations have shown striking resource savings for classification [8]–[10], but to the authors' knowledge no prior work has applied ternary quantization to closed-loop power converter control.

This paper presents a ternary-weight MLP for buck converter duty-cycle control on a Xilinx Zynq-7020 SoC, deployed in Q15.16 fixed-point precision. The controller is trained by behavioral cloning from a Tustin-discretized lag compensator [4], with weights constrained via the Ternary Weight Networks (TWN) formulation [5] and the straight-through estimator (STE) [6], optimized with Adam [7]. The trained network is realized as fully-unrolled pipelined Verilog with an adder-tree datapath and a 1024-entry ROM-based sigmoid. The main contributions are:

1) The first ternary-weight MLP, to the authors' knowledge, controlling a DC–DC converter in closed loop on FPGA hardware.
2) A $\sim$30$\times$ LUT reduction (84,769 $\rightarrow$ 2,830) and complete DSP elimination (70 $\rightarrow$ 0) versus a Q15.16 baseline that overflows the Zynq-7020's 53,200 LUTs at 159% utilization and cannot be placed.
3) A training-to-deployment pipeline—behavioral cloning, ternarization-aware training (STE, Adam), and synthesizable Verilog cores verified via Vivado / System Generator co-simulation.

Section II surveys related work; Section III the plant and compensator; Section IV the MLP design and training; Section V the FPGA implementation; Section VI the results.

II. Related Work

A. Neural Networks for Power Converter Control

Rajamallaiah *et al.* [1] survey deep reinforcement learning for converter control, and Hingu *et al.* [2] demonstrate a 32-bit RNN inverter controller on FPGA, confirming the feasibility of neural converter control on reconfigurable hardware. Both employ either floating-point networks or DRL policies with no extreme weight quantization. This work instead trains an MLP from a classical lag compensator [3], [4] and applies ternary quantization for hardware compression.

B. Ternary Weight Networks and FPGA Realizations

Li *et al.* [5] introduced TWN with a per-layer scaling factor $\alpha = E[|W|]$ and a thresholded ternarization rule that assigns each weight to $\{-1, 0, +1\}$, with gradients propagated through the non-differentiable quantization via the STE [6]. On FPGA, Tridgell *et al.* [8] reported up to 90% operation reduction from fully unrolled ternary CNNs; Woo *et al.* [9] confirmed substantial MAC reduction for RF modulation recognition;

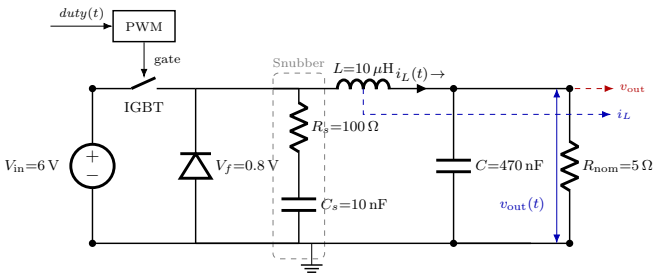

Fig. 1: Buck converter schematic (parameters in Table I).

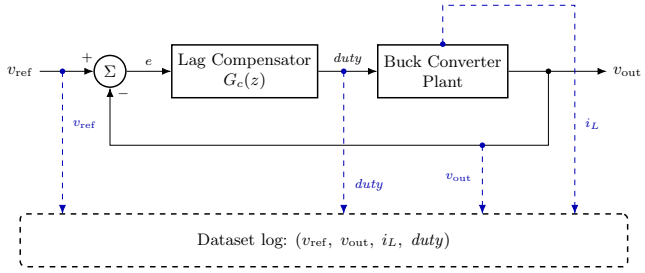

Fig. 2: Closed-loop arrangement for training-data generation.

and the hls4ml platform [10] demonstrates that binary and ternary networks deploy on FPGA with zero DSP utilization at sub-microsecond latency across a range of classification benchmarks. All target classification; none deploy ternary networks for closed-loop power converter control—the gap addressed here.

C. FPGA Inference Architectures and Sigmoid Implementation

Pipelined FPGA MLP architectures [11]–[13] and lightweight inference engines [14], [15] have addressed area–performance trade-offs for classification. The hls4ml framework [10] establishes that constant-weight matrix-vector multiplication with weights as compile-time constants lets the synthesis tool fold zero-weight multiplications and shift-and-add specific constants, and that binary and ternary weights need only AND/XOR gates; production-scale FPGA inference deployments have demonstrated the practicality of this approach [16]. Kordi *et al.* [17] show Q1.15 MAC units cut LUT use by over 67% and DSPs by 50% versus floating-point, motivating Q15.16 here. For the sigmoid, LUT-based ROM lookup [18], bit-level mapping [19], and piecewise-linear interpolation [20] have all been studied; this work adopts the ROM approach with saturation logic for the clamped range.

III. Buck Converter and Control Setup

A. Converter Topology and Parameter Selection

The buck converter is a step-down topology with a controlled switch, freewheeling diode, and RC snubber, shown in Fig. 1. All operating parameters are listed in Table I. Passive component values follow from the standard ripple equations [3]:

$$L = \frac{V_{\text{out}}(V_{\text{in}} - V_{\text{out}})}{\Delta I_L \, f_s \, V_{\text{in}}}, \qquad C = \frac{\Delta I_L}{8 \, f_s \, \Delta V_{\text{out}}}, \tag{1}$$

yielding the values in Table I for targets $\Delta I_L \approx 0.1$ A and $\Delta V_{\text{out}} \approx 10$ mV. The LC filter resonant frequency $f_{\text{res}} \approx 73$ kHz sits roughly a decade below the switching frequency.

B. Lag Compensator for Training-Data Generation

A continuous-time lag compensator

$$G_c(s) = G_{c0} \cdot \frac{1 + s/\omega_z}{1 + s/\omega_p} \tag{2}$$

places its zero at the LC resonance for active damping and its pole below for low-frequency gain (values in Table I). Tustin discretization [4] at $T_s = 1$ μs gives

$$u[n] = b_0 \, e[n] + b_1 \, e[n-1] + a_1 \, u[n-1], \tag{3}$$

with $e[n] = v_{\text{ref}}[n] - v_{\text{out}}[n]$. The integrator uses backward Euler, and internal signals are clamped to physically realizable bounds ($i_L \in [-0.2, 3]$ A, $v_{\text{out}} \in [0, 6]$ V, $|e| \le 6$ V) to prevent windup. The closed-loop arrangement of Fig. 2 produces the input–output trajectories used for supervised training.

TABLE I: Buck converter and lag-compensator parameters.

Parameter	Symbol	Value
Converter		
Input voltage	V_{in}	6 V
Nominal output voltage	V_{out}	5 V
Nominal load resistance	R_{nom}	5 Ω
Load sweep range (datasets)	R	3–20 Ω
Inductance	L	10 μH
Output capacitance	C	470 nF
Switching frequency	f_s	1 MHz
Sampling period	T_s	1 μs
LC resonant frequency	f_{res}	$\approx$ 73 kHz
Nominal duty cycle	D_{nom}	0.833
IGBT on-resistance	R_{on}	1 mΩ
Diode forward voltage	V_f	0.8 V
Snubber R_s, C_s	—	100 Ω, 10 nF
Lag compensator		
Zero frequency	$\omega_z/2\pi$	70 kHz
Pole frequency	$\omega_p/2\pi$	5 kHz
DC gain	G_{c0}	0.02
Tustin coeff. ($e[n]$)	b_0	2.402×10^{-2}
Tustin coeff. ($e[n-1]$)	b_1	1.536×10^{-2}
Tustin coeff. ($u[n-1]$)	a_1	0.9691

IV. MLP Controller Design

A. Network Architecture

The controller is a fully-connected MLP with two hidden layers (Table II). The input is $\mathbf{x} = [\, e, \, i_L \,]^\top$ with $e = v_{\text{ref}} - v_{\text{out}}$; the output is the duty cycle in $[0, 1]$. In the deployed ternary network, the forward pass at layer ℓ is

$$z_\ell = \alpha_\ell \, (W_{t,\ell} \, a_{\ell-1}) + b_\ell, \quad a_\ell = \varphi_\ell(z_\ell), \tag{4}$$

where $W_{t,\ell} \in \{-1, 0, +1\}^{n_\ell \times n_{\ell-1}}$, $\alpha_\ell \in \mathbb{R}$ is a per-layer scaling factor, and φ_ℓ is ReLU for $\ell \in \{1, 2\}$ and sigmoid for $\ell = 3$. Biases and α_ℓ are stored in Q15.16 in the deployed

TABLE II: MLP topology; the full-precision baseline uses the same neuron counts but without weight ternarization or α-scaling.

Layer	In	Out	Weights	Biases	Act.
Input	—	2	—	—	—
Hidden 1	2	16	32	16	ReLU
Hidden 2	16	32	512	32	ReLU
Output	32	1	32	1	Sigmoid
Total			**576**	**49**	

core. The lag compensator that produces the training labels acts only on e; supplying i_L as a second input gives the MLP the full second-order plant state and, because $i_L \approx v_{\text{out}}/R$ at quasi-steady state, an implicit observation of load condition without R as an explicit input.

B. Training Dataset Generation

Six operating profiles are generated by simulating the closed-loop system of Fig. 2 at T_s, each spanning the load range $R \in [3, 20]\ \Omega$ (0.25–1.67 A of nominal output current): A (constant load, constant reference), B (constant load, step-sweep reference), C (slow load sweep, constant reference), D (stepped load, step-sweep reference), E (per-sample random load and reference), and F (stepped load, constant reference). Training uses four randomized runs of each profile ($\sim$100k samples per run) drawn from A, B, C, D, and F—20 CSV files in total. Profile E is held out from training as configurations trained with E included gave higher residual training error on the remaining profiles, consistent with its broadband random reference and load providing largely unfocused training signal. The test set comprises one run per profile across all six (A–F).

C. Ternarization-Aware Training

Following Li *et al.* [5], the scaling factor and ternarized matrix at each layer are obtained from the latent real-valued weights W_ℓ as

$$\alpha_\ell = \tfrac{1}{N_\ell} \textstyle\sum_{i,j} |W_{\ell,ij}|, \tag{5}$$

$$W_{t,\ell,ij} = \begin{cases} +1, & W_{\ell,ij} > \delta_\ell, \\ -1, & W_{\ell,ij} < -\delta_\ell, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

with $\delta_\ell = 0.7\,\alpha_\ell$. Since (6) is not differentiable, the STE [6] passes gradients through it as the identity, and Adam [7] updates the latent W_ℓ directly. Training runs in single-precision floating-point; quantization to ternary weights and Q15.16 biases/scalings is applied when exporting the trained network to Verilog. The MSE loss $\mathcal{L} = (1/B)\sum_i (\hat{y}_i - y_i)^2$ is minimized for 100 epochs; the lowest-test-MSE configuration is retained for deployment. Hyperparameters are listed in Table III; the forward/backward pipeline is shown in Fig. 3. A full-precision baseline is trained with identical settings but without weight ternarization or α-scaling, so its layer rule simplifies to $z_\ell = W_\ell\, a_{\ell-1} + b_\ell$.

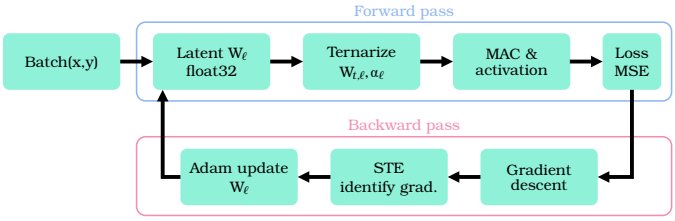

Fig. 3: Ternarization-aware training: forward pass ternarizes W_ℓ; backward pass uses the STE.

TABLE III: Training hyperparameters (both networks).

Hyperparameter	Symbol	Value
Loss function	$\mathcal{L}$	MSE
Optimizer	—	Adam [7]
Learning rate	η	1×10^{-4}
First-moment decay	β_1	0.9
Second-moment decay	β_2	0.999
Numerical stability	ε	1×10^{-8}
Minibatch size	B	256
Epochs	—	100
Ternarization threshold	δ_ℓ/α_ℓ	0.7

Threshold and α-scaling apply to the ternary network only; the full-precision baseline uses neither.

V. FPGA Implementation

The ternary controller (Section V-B) and full-precision baseline (Section V-C) share architecture, pipeline depth, and bias precision; they differ in how the weight–activation products are realized and in whether per-layer α-scaling is present. Both use the Q15.16 arithmetic of Section V-A, and both end in the ROM sigmoid of Section V-D.

A. Q15.16 Fixed-Point Arithmetic

All signals are signed Q15.16: 32-bit two's-complement, with the integer x_{int} interpreted as $x_{\text{int}} \cdot 2^{-16}$. The range $[-32768, +32767.99998]$ covers every physical quantity in the controller. A Q15.16 product is realized as

$$p = (a \times b) \ggg 16, \tag{7}$$

where $\ggg$ is sign-preserving arithmetic right shift; for operands within the controller's bounded operating range the result fits in 32-bit Q15.16, and no overflow is observed across the training and test datasets.

B. Ternary Datapath

At each adder-tree leaf, the ternary weight $W_{t,ij}$ selects between a_j, $-a_j$, and zero. Source-level this is a 3:1 multiplexer (Fig. 4); because the weights are constant after training, the code generator emits them as compile-time constants in Verilog and each leaf resolves before synthesis: $+1 \to$ wire, $-1 \to$ two's-complement negation, $0 \to$ pruned input. No runtime selection logic and no DSP blocks are needed for the weight–activation product.

The partial products feed a balanced adder tree of depth $\lceil \log_2(n_{\ell-1}) \rceil$. The tree output is multiplied by α_ℓ via (7) (the only multiplication in the layer) and summed with b_ℓ to form

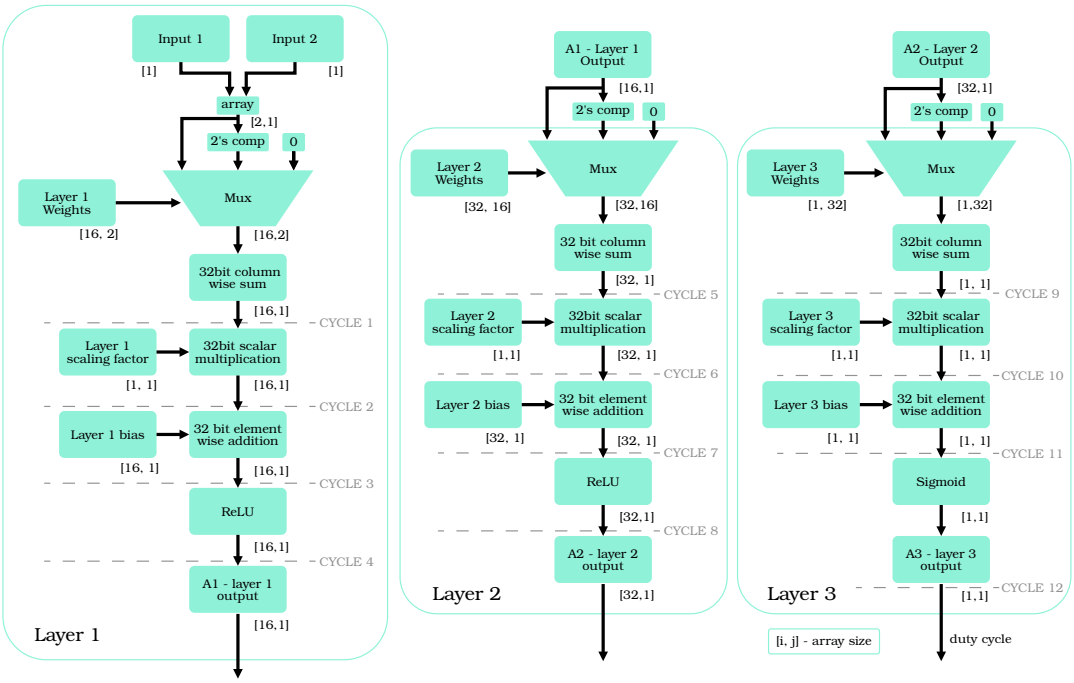

Fig. 4: Pipelined ternary MLP inference datapath.

z_ℓ. ReLU is a sign-bit comparison feeding a 2:1 multiplexer. End-to-end inference latency is twelve clock cycles; the initiation interval is one. With all 576 ternary weights resolved to wires, inverters, or pruned connections, the synthesis tool emits no DSP blocks for the forward path; the rest of the arithmetic maps to LUTs and carry chains.

C. Full-Precision Baseline Datapath

The baseline shares the topology, pipeline depth, and bias precision of the ternary network but differs in two respects at each layer: (i) every weight–activation product is computed in full Q15.16 as $p_{ij} = (W_{\ell,ij} \times a_{\ell-1,j}) \ggg 16$, with $W_{\ell,ij}$ stored as a Q15.16 *localparam*; and (ii) per-layer α-scaling is omitted, since the full-precision weights already carry the necessary magnitude information. The 576 products are constant-coefficient multipliers, which Vivado distributes between LUTs and DSPs. Both designs store weights as compile-time *localparams* (no BRAM), so the synthesis-stage resource difference is attributable to the weight representation (and the absent α-stage) rather than to the memory strategy.

D. Sigmoid via Lookup Table

The output sigmoid is a 1024-entry ROM (Fig. 5) populated at design time by sampling $\sigma(x) = 1/(1 + e^{-x})$ uniformly over $x \in [-8, +8]$ in Q15.16 and emitting the table as *localparam* assignments. The clamp range is chosen because $|\sigma(\pm 8) - \{0, 1\}| < 4 \times 10^{-4}$, below the Q15.16 quantization step. At inference, z_3 is range-clamped (saturating to 0.0 or 1.0); within range, the ROM address is $(z_3 + 8) \ggg 10$—equivalently bits $[19 : 10]$ of the offset operand, dividing the integer code by 1024 to index a 10-bit address. The clamp,

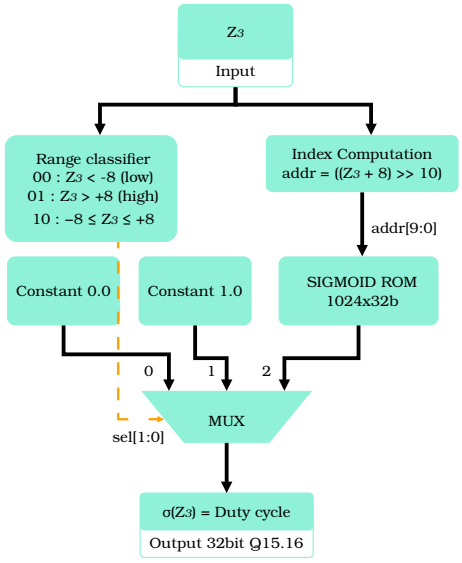

Fig. 5: Combinational ROM-based sigmoid activation.

address generation, and ROM read all complete within one cycle—the sigmoid path is combinational—and its output is captured into the registered `duty` signal at the next clock edge.

VI. Results

A. Training Accuracy

Both networks were trained on the five training profiles (Section IV-B) for 100 epochs with the hyperparameters of

TABLE IV: Final test-set accuracy.

Network	Weights	MSE	RMSE	Δ duty
Ternary	$\{-1, 0, +1\} + \alpha$	9.94×10^{-4}	0.0315	$\pm 3.15\%$
Full-prec.	32-bit Q15.16	2.77×10^{-4}	0.0167	$\pm 1.67\%$

Δ duty is RMSE as a fraction of the duty range $[0, 1]$.

Table III; results are summarized in Table IV and the loss trajectories in Fig. 6. The ternary network exhibits approximately 3.6× higher test MSE than the full-precision baseline, the expected outcome of constraining weights to a discrete three-valued set. The penalty is modest: a ternary RMSE of 0.0315 corresponds to a sub-3.2% duty-cycle deviation, comparable to typical PWM duty resolutions. Section VI-B verifies that this training-set accuracy yields well-behaved closed-loop behavior; Section VI-C reports the resource savings that justify the trade-off. The baseline is not further tuned, as it serves as a resource-consumption reference rather than a performance target.

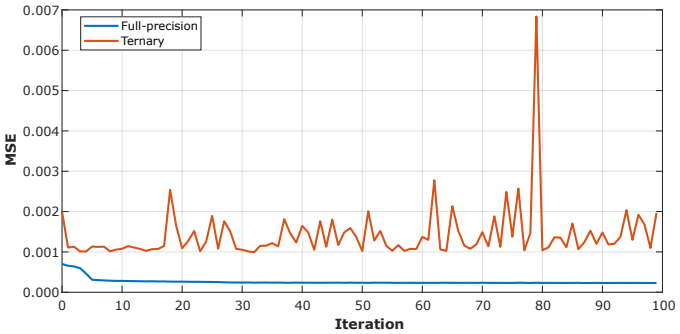

Fig. 6: Training MSE versus iteration.

B. Closed-Loop Verification

The Verilog inference core was instantiated as a System Generator HDL block (Fig. 7) and run in closed loop against the buck plant at $T_s = 1$ μs, communicating with the Simulink model via the System Generator JTAG bridge. Three step disturbances were applied independently: an input-voltage step in V_{in} (Fig. 8), a reference step in v_{ref} (Fig. 9), and a load-resistance step in R (Fig. 10). In each scenario the FPGA controller is well-behaved and returns v_{out} to track v_{ref}. The small steady-state offset visible in the traces is inherited from the teacher: $G_c(s)$ is type-0 (finite DC gain $G_{c0} = 0.02$), so the lag compensator itself does not eliminate steady-state error against constant disturbances, and the cloned MLP faithfully reproduces this behavior; the plant's diode forward drop V_f contributes additionally. Eliminating the offset would require a teacher with integral action or a plant-side correction, not a change to the controller representation. The full-precision baseline is not co-simulated as an HDL block: as established in Section VI-C, it does not fit on the target device.

C. FPGA Resource Utilization

Both controllers were synthesized in Vivado for the Xilinx Zynq-7020 SoC (XC7Z020). Post-synthesis utilization is given

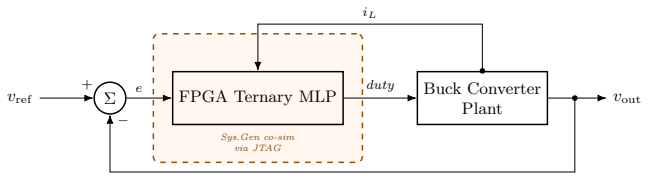

Fig. 7: System Generator JTAG co-simulation arrangement.

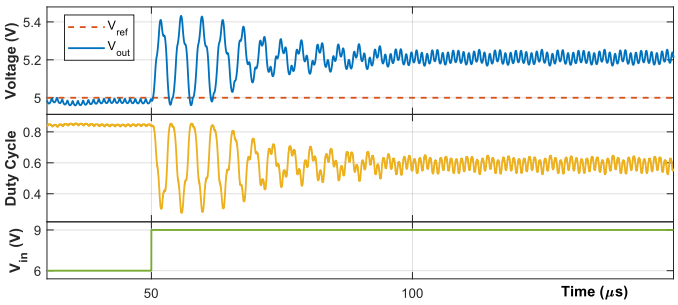

Fig. 8: Closed-loop response to an input-voltage step in V_{in}.

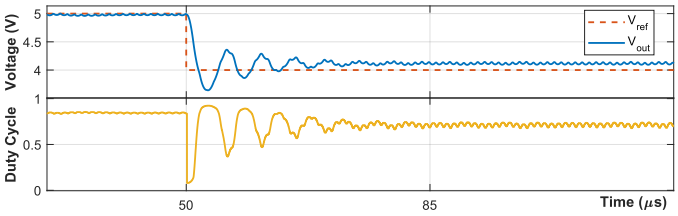

Fig. 9: Closed-loop response to a reference step in v_{ref}.

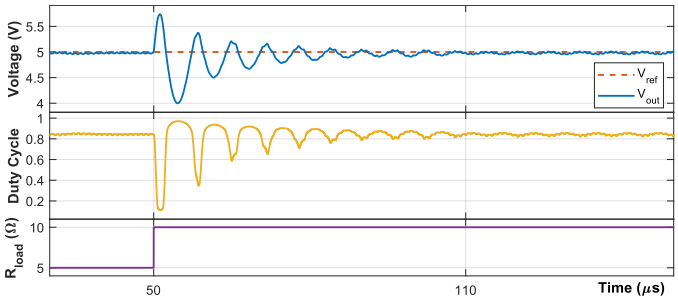

Fig. 10: Closed-loop response to a load-resistance step in R.

in Table V. The ternary core fits comfortably; the full-precision baseline exceeds the available LUTs at 159% utilization and does not fit on the target device, so place-and-route was not attempted. Three points follow: (i) the ternary implementation reduces LUT use by ∼30×; (ii) DSP usage is eliminated entirely; (iii) the baseline is not deployable on the Zynq-7020, whereas the ternary implementation uses under 6% of the LUTs. Flip-flops decrease ∼2.4× as register usage downstream of the pruned-weight paths shrinks. On higher-capacity device families (UltraScale+) both implementations would synthesize successfully, but the ternary advantage—and the associated power and cost benefits—would persist.

Post-implementation results for the ternary core confirm the

synthesis estimate: 2,781 LUTs (5.23%) after place-and-route, with FF and DSP counts unchanged from Table V. At a 10 ns target clock period (100 MHz), static timing analysis reports a worst-case slack of 0.659 ns, equivalent to a critical-path delay of 9.34 ns and a maximum operating frequency $F_{\max} \approx 107$ MHz; hold and minimum-pulse-width constraints are also met with positive slack. At the 100 MHz operating point, the twelve-cycle inference pipeline produces a duty-cycle update in 120 ns, well within the $T_s = 1$ μs converter sampling period.

TABLE V: Post-synthesis utilization on Xilinx Zynq-7020.

Resource	Ternary	Full-precision	Available
LUT	**2,830 (5.32%)**	84,769 (**159.3%**†)	53,200
FF	2,076 (1.95%)	5,072 (4.77%)	106,400
DSP	**0 (0.00%)**	70 (31.82%)	220

†Exceeds device capacity; place-and-route not attempted.

VII. Conclusion

This paper presented a ternary-weight MLP for buck converter duty-cycle control, realized as synthesizable Verilog on a Xilinx Zynq-7020 SoC and verified by closed-loop System Generator co-simulation. Against an architecturally matched Q15.16 baseline, the ternary implementation cuts LUT use by $\sim 30\times$ and eliminates DSP usage entirely; the baseline exceeds device capacity, while the ternary core occupies only 5.3% of available LUTs. These results indicate that ternary weight quantization can enable fully-parallel neural converter control on low-cost edge FPGAs. Fabric remaining after deployment leaves room for a full single-chip control stack (PWM, ADC interface, supervisory logic) on the same SoC. Hardware-in-the-loop validation on physical converter hardware and full-system co-deployment are the subjects of ongoing work.

References

[1] A. Rajamallaiah, S. V. K. Naresh, Y. Raghuvamsi, S. Manmadharao, K. Bingi, R. Anand, and J. M. Guerrero, "Deep reinforcement learning for power converter control: A comprehensive review of applications and challenges," *IEEE Open Journal of Power Electronics*, vol. 6, pp. 1769–1802, 2025.

[2] C. Hingu, X. Fu, P. Vangala, R. Mishan, and P. Fajri, "32-bit fixed and floating-point hardware implementation for enhanced inverter control: Leveraging FPGA in recurrent neural network applications," *IEEE Access*, vol. 12, pp. 111 097–111 110, 2024.

[3] R. W. Erickson and D. Maksimović, *Fundamentals of Power Electronics*, 3rd ed. Cham, Switzerland: Springer, 2020.

[4] M. Talha and I. A. Makda, "Frequency-domain modeling and Tustin discretization method based controlling of DC step-up chopper," in *2019 4th International Conference on Power Electronics and their Applications (ICPEA)*, 2019, pp. 1–5.

[5] F. Li, B. Liu, X. Wang, B. Zhang, and J. Yan, "Ternary weight networks," 2022. [Online]. Available: https://arxiv.org/abs/1605.04711

[6] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013. [Online]. Available: https://arxiv.org/abs/1308.3432

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[8] S. Tridgell, M. Kumm, M. Hardieck, D. Boland, D. Moss, P. Zipf, and P. H. W. Leong, "Unrolling ternary neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1909.04509

[9] J. Woo, K. Jung, and S. Mukhopadhyay, "Efficient hardware design of DNN for RF signal modulation recognition employing ternary weights," *IEEE Access*, vol. 12, pp. 80 165–80 175, 2024.

[10] J.-F. Schulte, B. Ramhorst, C. Sun, J. Mitrevski, N. Ghielmetti, E. Lupi, D. Danopoulos, V. Loncar, J. Duarte, D. Burnette, L. Laatu, S. Tzelepis, K. Axiotis, Q. Berthet, H. Wang, P. White, S. Demirsoy, M. Colombo, T. Aarrestad, S. Summers, M. Pierini, G. Di Guglielmo, J. Ngadiuba, J. Campos, B. Hawks, A. Gandrakota, F. Fahim, N. Tran, G. Constantinides, Z. Que, W. Luk, A. Tapper, D. Hoang, N. Paladino, P. Harris, B.-C. Lai, M. Valentin, R. Forelli, S. Ogrenci, L. Gerlach, R. Flynn, M. Liu, D. Diaz, E. Khoda, M. Quinnan, R. Solares, S. Parajuli, M. Neubauer, C. Herwig, H. F. Tsoi, D. Rankin, S.-C. Hsu, and S. Hauck, "hls4ml: A flexible, open-source platform for deep learning acceleration on reconfigurable hardware," 2025. [Online]. Available: https://arxiv.org/abs/2512.01463

[11] A. P. d. A. Ferreira and E. N. d. S. Barros, "A high performance full pipelined architecture of MLP neural networks in FPGA," in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*, 2010, pp. 742–745.

[12] L. F. G. Cardenas, N. Vazquez, L. E. Rojo, S. E. P. Castillo, H. J. C. L. Tapia, and R. O. Guerrero, "FPGA implementation of a multi-layer perceptron," in *2025 International Conference on Electrical Drives and Power Electronics (EDPE)*, 2025, pp. 1–5.

[13] S. Guo, Y. Huang, K. Mai, and G. Mo, "A superscalar six-stage pipeline neural network accelerator: Design and implementation based on FPGA," in *2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2024, pp. 126–133.

[14] A. Rehman, M. P., and P. B. S., "FPGA-based efficient MLP neural network for digit recognition," in *2023 International Conference on Integration of Computational Intelligent System (ICICIS)*, 2023, pp. 1–7.

[15] Y. Wang, Y. Qian, and X. He, "Design and implementation of lightweight neural network inference accelerator based on FPGA," in *2024 International Conference on Control, Electronic Engineering and Machine Learning (CEEML)*, 2024, pp. 98–104.

[16] J. Duarte, P. Harris, S. Hauck, B. Holzman, S.-C. Hsu, S. Jindariani, S. Khan, B. Kreis, B. Lee, M. Liu, V. Lončar, J. Ngadiuba, K. Pedro, B. Perez, M. Pierini, D. Rankin, N. Tran, M. Trahms, A. Tsaris, C. Versteeg, T. W. Way, D. Werran, and Z. Wu, "FPGA-accelerated machine learning inference as a service for particle physics computing," *Computing and Software for Big Science*, vol. 3, no. 1, 2019.

[17] F. Kordi, P. Fortier, and A. Miled, "Optimized fixed point MAC unit for neural network on FPGA," in *2024 International Conference on Microelectronics (ICM)*, 2024, pp. 1–5.

[18] R. Pogiri, S. Ari, and K. K. Mahapatra, "Design and FPGA implementation of the LUT based sigmoid function for DNN applications," in *2022 IEEE International Symposium on Smart Electronic Systems (iSES)*, 2022, pp. 410–413.

[19] I. V. Ushenina and E. A. Danilov, "Implementation of the sigmoid function on current FPGAs using the bit-level mapping method," in *2024 International Russian Automation Conference (RusAutoCon)*, 2024, pp. 295–300.

[20] P. Mukherji, S. Rajput, and V. Mudaliar, "Optimization of sigmoid activation function on FPGA: An analysis of linear interpolation with fixed-point representations," in *2025 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, 2025, pp. 238–246.