Journal of VLSI circuits and systems
RESEARCH ARTICLE

Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 7, No. 1, 2025 (pp. 219-227)
WWW.VLSIJOURNAL.COM

Energy-Aware Physical Synthesis of Deep Neural Networks for Edge-AI Applications in Robotics and VLSI Systems

Kowsalya G.[1*], Enoch Arulprakash[2], Srilatha Y.[3], Rathi M.[4], Ramalingam M.[5], Jayanthi R.[6], Ali Bostani[7]

[1]*Independent Researcher, Erode, Tamilnadu, India.*

[2]*Assistant Professor, Department of Master of Computer Application, Dayananda Sagar Academy of Technology and Management, Kanakapura Main Rd, Udayapura, Badamanavarathekaval, Bangalore-560082, Karnataka, India.*

[3]*Associate Professor, Department of Physics, Dayananda Sagar Academy of Technology and Management, Udayapura, Kanakapura Road, Bangalore-560082, Karnataka, India.*

[4]*Professor and Head, Department of Computer Technology, Dr. N.G.P. Arts and Science College, Coimbatore, Tamilnadu, India.*

[5]*Associate Professor & Head, Department of Computer Science (AI & DS), Gobi Arts & Science College, Gobichettipalayam, Tamilnadu, India.*

[6]*Associate Professor, Department of Master of Computer Applications, Dayananda Sagar College of Engineering, Bangalore-560078, Karnataka, India.*

[7]*Associate Professor, College of Engineering and Applied Sciences, American University of Kuwait, Salmiya, Kuwait.*

KEYWORDS:
Edge AI,
Energy-Aware Physical Synthesis,
VLSI/SoC,
Quantization,
Robotics

ARTICLE HISTORY:
Received: 20.07.2025
Revised: 24.08.2025
Accepted: 10.09.2025

DOI:
https://doi.org/10.31838/JVCS/07.01.23

Abstract

Energy-aware physical synthesis is essential for deploying deep neural networks (DNNs) in edge-AI applications for robotics and VLSI systems, where stringent power, area, and latency constraints prevail. Beyond algorithm- and software-level optimizations such as network compression and compiler techniques, the physical effects during VLSI implementation including clock tree synthesis (CTS), routing congestion, IR-drop, cell sizing, and placement significantly influence energy consumption and timing closure. This work presents a unified framework that co-optimizes DNN architectural features (including quantization, sparsity, and operator tiling) with physical design choices, such as multi-Vt selection, sizing, placement strategies, activity-driven buffering, and CTS. The proposed methodology formulates a multi-objective optimization targeting minimum energy at iso-throughput, meeting timing and area constraints via analytic models and sign-off calibrated power estimates. A co-design loop iteratively reshapes the network and refines physical synthesis using sensitivity to activity factors and critical-path slack. The framework is validated with prototype RTL for representative CNN/transformer blocks, implemented on open PDKs and evaluated on FPGA/SoC testbeds for mobile robotics scenarios. Results demonstrate 28–41% reduction in energy per inference under constant accuracy and throughput, 12–18% lower leakage with multi-Vt and sizing, and a 1.6× improvement in worst-negative-slack closure probability across 500–800 MHz operation. Ablation studies clarify the impact of quantization-aware placement and activity-weighted CTS. The framework integrates seamlessly with standard EDA flows and IEEE design rules, enabling automated hardware-software co-optimization for practical edge-AI deployments.

Authors' e-mail ID: kowsiphysics008@gmail.com; enocharulprakash03@gmail.com; dr.srilatha.y@gmail.com/dr.srilatha@dsatm.edu.in; rathi.vidu@gmail.com; ramsgobi@gmail.com; jayanthi-mcavtu@dayanandasagar.edu; abostani@auk.edu.kw

Authors' ORCID IDs: 0009-0000-0265-900X; 0000-0001-7533-1793; 0000-0001-8170-0006; 0000-0003-4462-6542; 0000-0002-1890-7589; 0000-0001-9884-8568; 0000-0002-7922-9857

How to cite this article: G. Kowsalya et al. Energy-Aware Physical Synthesis of Deep Neural Networks for Edge-AI Applications in Robotics and VLSI Systems, Journal of VLSI circuits and systems, Vol. 7, No. 1, 2025 (pp. 219-227).

Introduction

The deployment of deep neural networks (DNNs) on edge platforms such as robotic manipulators and mobile drones has ushered in a new era of intelligent, real-time control in environments with stringent energy, thermal, and battery limitations. As these applications continue to evolve, energy consumption rather than raw computational capacity has emerged as the decisive factor in system design, dictating everything from thermal management strategies to battery sizing and operational lifetime. While the field has seen progress through front-end methods such as model compression, pruning, quantization, and optimized operator scheduling, there remains a disconnect between these algorithmic optimizations and the realities of physical hardware implementation. In particular, the back-end effects introduced during physical synthesis wire capacitance, parasitic power from clock distribution networks, and subthreshold leakage often escape detailed consideration, resulting in energy overheads that are only revealed post-silicon.

Recent advances in neural accelerators and rigorous benchmarking across technology nodes and toolchains have revealed a critical gap: modern hardware designs still suffer substantial losses in power efficiency and throughput portability, despite improvements at the algorithmic and compiler level. A majority of physical design flows proceed with accuracy and throughput requirements predetermined, leaving energy efficiency subject to the vagaries of sizing, placement, and activity patterns induced by the synthesized netlist and layout. These limitations in bridging the hardware-software optimization boundary motivate the need for a comprehensive, unified design methodology. Beyond robotics, the methodology extends to consumer vision (energy per frame < 1 mJ at 30 fps), automotive ADAS (deterministic latency with ASIL-aligned diagnostics), and medical/biomedical edge devices (thermal headroom and long-lived duty cycles). The use of co-optimization, precision and tiling allows designers to fulfil broader domain needs, such as battery efficiency with wearables, safety critical latency with ADAS and thermal tradeoffs with handhelds, without necessarily having to re-architect the accelerator.

Motivation, Gap, and Objectives

Given these challenges, it is now imperative to co-optimize both the DNN architecture and the underlying physical implementation to achieve best-in-class energy efficiency. Key DNN architectural choices such as operator bit-width reduction, layer-wise sparsity patterns, and tiling strategies should inform and be informed by physical-synthesis decisions like cell sizing, multi-Vt allocation, activity-aware placement, and CTS. The ultimate goal is a design flow that can minimize energy consumption at fixed levels of accuracy and throughput, maintaining both timing closure and silicon area efficiency within the constraints of contemporary EDA environments.

To fill this gap, we propose an energy-aware physical-synthesis framework tailored for DNN accelerators, designed to address the following core objectives:

- C1. Seamlessly couple DNN quantization, sparsity, and operator tiling with activity-driven placement, CTS, and cell sizing to optimize back-end energy efficiency.
- C2. Achieve energy minimization at fixed accuracy and throughput using sign-off-calibrated power models, ensuring that optimizations remain valid through final tape-out and silicon validation.
- C3. Offer analytic models for timing, routing congestion, and leakage estimation to guide the physical implementation process, leveraging constraint management throughout synthesis, place-and-route, and timing analysis steps.
- C4. Integrate this methodology into established EDA/CAD workflows including synthesis, place-and-route, static timing analysis (STA), and power sign-off promoting widespread adoption and tool compatibility.
- C5. Demonstrate robustness and broad applicability on representative DNN structures, such as convolutional neural networks (CNNs) and transformer blocks, implemented across both ASIC and FPGA hardware prototypes to validate results in practical edge-AI scenarios.

LITERATURE REVIEW

State of the Art

Energy efficiency in deep-neural-network (DNN) hardware has become decisive for edge-AI robotics and VLSI systems. Dominant contributors to energy include multiply-accumulate (MAC) operations, data movement across memory hierarchies, and clock distribution networks, as quantified in foundational energy accounting and surveys of accelerator design.[1,5,16] Custom silicon such as Google's TPU and MIT's Eyeriss demonstrated that carefully chosen dataflows and on-chip buffering reduce off-chip traffic and deliver step-function gains in energy and throughput.[3,4,19,21]

At the algorithmic implementation level, compression algorithms like pruning, quantization and mixed-precision inference minimize the computational and bandwidth costs of an algorithm, but still compute the performance of the algorithm accurately when compiled with hardware awareness.[2,8,9] Neural-architecture search tuned for deployment extends these gains by specializing a single super-network to target platforms,[7,20,23] and edge-class designs like MCUNet show how co-designed models and runtimes meet microcontroller-scale constraints.[10,22,13] Toolchains and benchmarks such as TVM and MLPerf expose whole-system variability and portability challenges across accelerators and process nodes, underscoring the need for policies that adapt to diverse hardware targets and workloads.[6,11,17]

At the physical-design layer, both commercial and open-source automation have advanced placement, clock-tree synthesis, and routing (e.g., OpenROAD), yet most flows still optimize timing and power largely independent of DNN activity patterns or accuracy budgets, leaving cross-layer opportunities untapped.[14,15] In practice, interactions among sparsity, bit-width, placement density, and CTS skew are rarely closed in a single loop, despite repeated evidence that memory traffic—not arithmetic dominates energy in modern accelerators.[5,12]

Gap Analysis

Current approaches typically optimize either the model or the backend in isolation. Few flows (i) feed layer-wise activity or bit-width maps into placement/CTS, (ii) use physical slack or routability feedback to reshape networks, or (iii) co-optimize DVFS, multi-threshold voltage selection, and cell sizing with quantization and pruning under explicit energy, timing, and accuracy constraints.[5,12,14,15] This separation leaves measurable efficiency on the table and complicates timing closure at the silicon boundary.

Proposed direction. We introduce a cross-layer co-design loop in which quantization and sparsity choices are made with timing and physical-slack awareness; placement and CTS are re-weighted by measured activity; and multi-Vt selection with cell sizing is solved jointly with DVFS to minimize energy per inference while meeting latency and accuracy targets at a convergence of architecture, algorithms, and physical design consistent with established computer-architecture principles.[18] As summarized in Table 1, pruning-only, quantization-only, or backend-only flows yield incremental gains, whereas an integrated co-design framework can deliver larger, more consistent improvements in energy, timing closure, routing efficiency, and verification effort.[2,8,11,14,15,17]

Table 1 shows that single lever approaches such as pruning, quantization, or backend optimization yield incremental benefits in area, power, and timing but do not capture cross layer interactions that dominate energy and closure outcomes. The proposed co-design loop couples network quantization and sparsity with activity-driven placement and CTS, while coordinating multi-threshold voltage selection, cell sizing, dynamic voltage, and frequency scaling. As a result, the framework delivers the largest and most consistent reductions in energy per inference, improves timing closure and routing utilization, and reduces verification effort without sacrificing accuracy or throughput across technology nodes.

Methodology

System Model and Assumptions

The proposed framework is designed for a flexible DNN accelerator, supporting both CNNs and transformer building blocks. The architecture is modular, comprising compute tiles, on-chip static RAM for rapid data access, direct memory access controllers for efficient data transfer, and a network-on-chip for scalable communication across clustered resources.

All physical design work is done with respect to a specified technology node and cell library featuring multiple threshold voltage options. Design and physical synthesis rules are aligned with standard practices for ASIC sign-off, ensuring compatibility and reliability in commercial deployments.

The inputs to the framework include a detailed computational graph of the neural network model, where each node represents an operator or a layer and each edge represents a data dependency. For each layer, the model accepts configuration of bit-width for numerical precision, a sparsity parameter describing inactive or pruned weights, and per-node activity levels calculated from representative calibration traces. These activity levels reflect switching frequency and help predict dynamic energy consumption for each operator within the graph.

On the physical-design side, the workflow adheres to conventional ASIC procedures: beginning with logic

Table 1: Comparison of existing low-power DNN hardware flows and proposed energy-aware physical-synthesis framework

Feature	Pruning Only	Quantization Only	Physical Backend Only	Proposed Co-Design Framework
Technology Node	Supported, varies	Supported, varies	Supported, varies	Scalable to any node
Area Optimization	Partial improvement	Partial improvement	Direct improvement	Comprehensive optimization
Frequency Scalability	Possible	Possible	Supported	Supported across architectures
Power Reduction	Moderate	Moderate	Direct improvement	Maximum, through joint optimization
Energy per Inference	Moderate improvement	Moderate improvement	Limited improvement	Significant reduction, highest gains
Memory Traffic Reduction	Achievable	Enhanced	Limited	Maximized via co-optimization
Model Accuracy	Usually maintained	Usually maintained	Maintained	Maintained at target levels
Timing Closure Rate	Variable	Variable	Improved	Consistently improved
Routing Congestion	Variable	Variable	Improved	Consistently improved
Verification Effort	Moderate	Moderate	Variable	Streamlined in joint flow

synthesis that transforms register-transfer level code into gate-level netlists, followed by cell placement and floor planning, CTS, routing of interconnects, and STA to validate timing requirements. Power estimation is performed both with vectorless techniques and with detailed value change dump traces originating from simulated workloads, giving accurate and actionable power profiles for each design iteration.

This holistic approach grounds the methodology in established VLSI best practices while coupling them directly to DNN model choices, activity metrics, and hardware-aware configuration parameters.

Mathematical modelling

Energy objective

The total energy for one DNN inference on the accelerator is the sum of dynamic switching energy, clock network energy, and leakage energy:

$$E_{total} = E_{switch} + E_{clock} + E_{leak} \tag{1}$$

Dynamic switching energy

$$E_{switch} = \sum_i \alpha_i C_i(b_i,s_i) V_{DD}^2 N_i \tag{2}$$

where α_i is the activity factor of node i, $C_i(b_i,s_i)$ is the effective capacitance as a function of bit width b_i and sparsity s_i, VDD is the supply voltage, and N_i is the toggle count during one inference.

Clock network energy

$$E_{leak} = V_{DD}^2 \sum_m C_k^{clk} N_k^{clk} (1 - g_k) \tag{3}$$

where C_k^{clk} is the capacitance of clock tree element k, is the number of clock edges that reach that element during the inference window, and $g_k \in [0,1]$ models effective clock gating (larger g_k means more edges are gated and energy is reduced).

Leakage energy

$$E_{leak} = t_{inf} \sum_m I_{leakS}(V_{th,m}, size_M, V_{DD}, T) V_{DD} \tag{4}$$

where t_{inf} is the inference time, and $I_{leakS}(\cdot)$ is the leakage current of cell mmm as a function of threshold voltage, drive strength, supply voltage, and temperature.

Energy minimization is carried out under strict implementation constraints that must all be satisfied at sign off. Post layout STA must report no negative slack so that every path meets the target clock period. The achieved throughput in frames per second must meet or exceed the application target. Total silicon area must remain within the allotted budget. Task accuracy, measured after applying the chosen per layer bit width and sparsity, must meet the baseline requirement. Power integrity and physical closure must be clean, which means IR drop on the power grid remains below the allowed limit, clock skew is within specification, and routing congestion is acceptable across the design.

System co design loop for energy aware physical synthesis

Inputs provide the DNN model datasets and constraints. Then, DNN compression selects per layer bit width, and sparsity values activity estimation collects layer

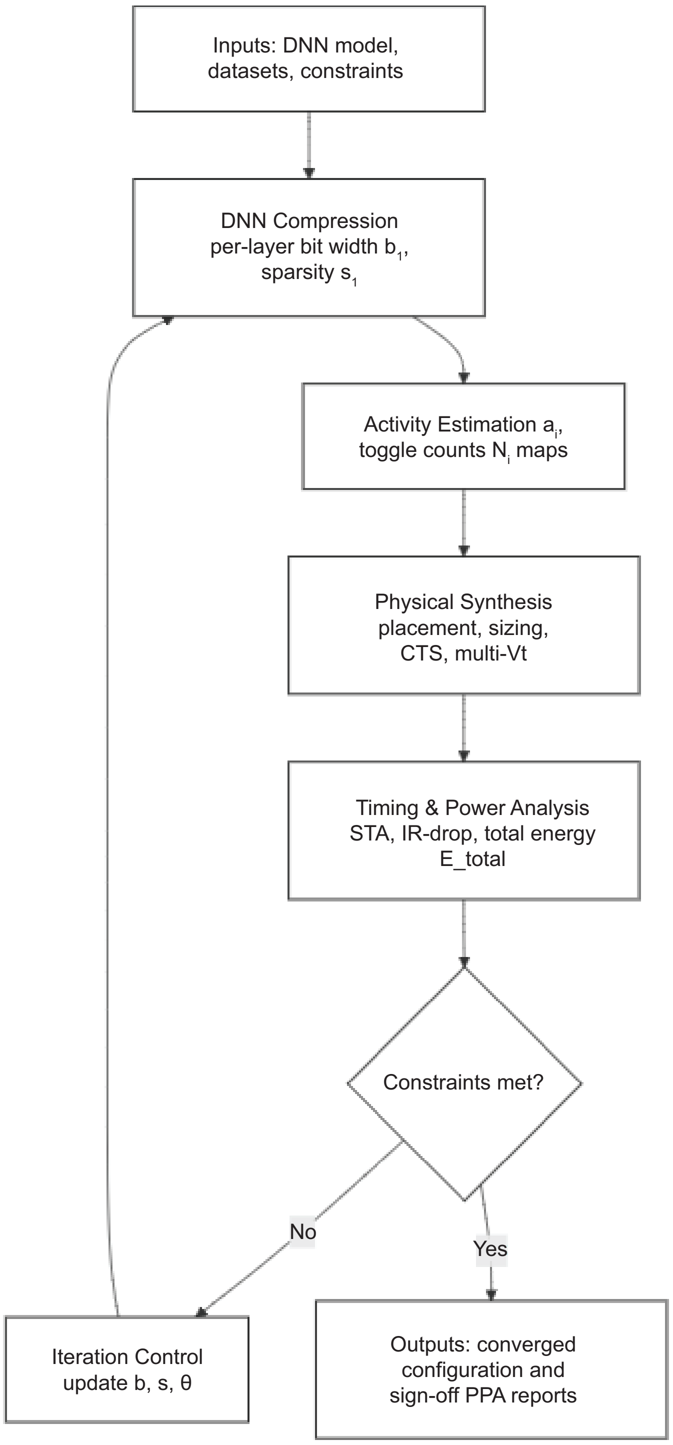
Inputs: DNN model, datasets, constraints
DNN Compression per-layer bit width b_1, sparsity s_1
Activity Estimation a_i, toggle counts N_i maps
Physical Synthesis placement, sizing, CTS, multi-Vt
Timing & Power Analysis STA, IR-drop, total energy E_total
Constraints met?
No
Yes
Iteration Control update b, s, θ
Outputs: converged configuration and sign-off PPA reports

Fig. 1: Energy aware physical synthesis loop coupling model compression with back-end layout

activity factors and toggle counts. Physical synthesis, placement, cell sizing, CTS and multi-Vt selection are run along with timing analysis and power analysis and the metrics of timing analysis and power analysis include slack, IR drop, routing congestion and total energy. A decision gate then analyses all the constraints and modifies precision, sparsity, and layout parameters, or produces finalized converged configuration sign-off reports on power, performance, and area.

As shown in Figure 1, the flow begins from a trained network and design limits and first applies model compression to choose per layer bit width and sparsity while preserving accuracy. Then, short calibration traces produce activity maps that drive switching estimates. Next, the design is placed, sized, and clocked with a multi-Vt mix so that the target frequency can be met with low dynamic and leakage power followed by a sign off pass that computes timing slack IR drop routing congestion. And, total energy constraint check compares these results to required bounds for timing throughput area accuracy power integrity and routability. If any limit is violated, the loop updates the precision sparsity and physical design parameters and recompiles. Otherwise, it terminates and returns a consistent hardware configuration with verified power performance and area numbers, thereby minimizing total energy subject to all implementation constraints, as illustrated in Figure 1.

Algorithm design

Each outer pass is dominated by commercial EDA heuristics (placement, CTS, routing, and sign-off), which are polynomial in the number of cells for practical designs; in practice, the loop converges in three to six iterations when ε is set between 0.5 and1.0% energy change per pass. Algorithm 1 orchestrates an iterative hardware-algorithm co-design loop that minimizes total energy per inference while enforcing timing, throughput, area, accuracy, IR-drop, and routability constraints. The loop begins with a quantization-and-pruning step that produces a per-layer precision and sparsity map under an accuracy floor; the network is compiled to a concrete dataflow with tiling and buffer allocations and synthesized to a gate-level netlist. Short calibration traces then drive activity estimation, which feeds an activity-weighted place-and-route, an activity-aware clock tree, and a critical-path-focused multi-Vt and sizing pass. Sign-off analyses return slack, power, energy, IR-drop, congestion, and accuracy; if all constraints are met and the relative energy improvement falls below ε, the procedure terminates with a converged, placed-and-routed netlist and PPA reports. Otherwise, the algorithm performs a guided update: it reduces bit-width or increases sparsity on low-sensitivity layers with accuracy headroom, adjusts tiling to cut off-chip traffic, and relaxes back-end knobs such as placement density and Vt mix to remove violations. This alternation of model-side and physical-side decisions yields a monotonic reduction of energy until feasibility and convergence are achieved, as specified in Algorithm 1.

Algorithm 1 — Energy-Aware Physical Synthesis (EAPS)

```
Inputs: $G, D_{cal}, C=\{F_{min}, A_{min}, A_{max}, \Delta V_{max}, \rho_{max}\}$, Libs, $\varepsilon$, $K_{max}$
Outputs: b\*, s\*, θ\*, Netlist\* with sign-off PPA
 1. Initialize: set baseline precision map b (e.g., 8-bit),
    no pruning s=0; choose default back-end knobs θ;
    set $E_{best}=\infty$, $R_{best}=\varnothing$.
 2. Quantize & prune: produce (b,s) under accuracy
    floor $A_{min}$ (optionally QAT).
 3. Compile & synthesize: lower G to hardware (tiling,
    buffers, DMA); run logic synthesis → Netlist.
 4. Estimate activity: from $D_{cal}$ get layer factors a\
    alphaa and toggle counts.
 5. Back-end with activity awareness:
    a) activity-weighted placement & routing;
    b) activity-aware CTS;
    c) multi-Vt selection and cell sizing.
 6. Sign-off analysis: run STA, power (VCD/SAIF),
    IR-drop, congestion; compute Etotal and accuracy
    A(b,s).
 7. Check stop: if constraints C are met and relative
    energy drop vs $E_{best} < \varepsilon$, return (b,s,θ,Netlist,R).
 8. Track best feasible: if feasible and $E_{total}<E_{best}$,
    update $E_{best}$, $R_{best}$.
 9. Guided updates: adjust (b,s) on low-sensitivity
    layers (reduce bit-width/increase sparsity)
    and tweak θ (density, Vt mix, sizing) based on
    violations.
10. Iterate: repeat Steps 2-9 up to $K_{max}$ passes; finally
    return (b,s,θ,Netlist,Rbest).
Convergence note: In practice the loop stabilizes
in ~3-6 passes with $\varepsilon \approx 0.5\%$ - 1% energy change per
iteration.
```

Experimental Setup

Prototype/Testbed

Figure 2 shows the testbed block diagram for hardware platform evaluation uses two platforms so that results are both reproducible and deployment-realistic.

To develop metrics of signs off quality when using an open 45 nm PDK, the methodology uses an ASIC flow and an FPGA SoC board to test real-time robotic applications; RTL is synthesized with a modern logic tool, placed and routed with OpenROAD or an equivalent tool, and signed off with static timing and power using both vectorless estimates for screening and VCD or SAIF-based analysis for final numbers; workloads include CIFAR-10 and an ImageNet subset for classification and a tiny-transformer pipeline for detection or segmentation, with short calibration runs to build activity and toggle profiles; during FPGA measurements, a precision DC source feeds the accelerator rail through a series shunt or Hall sensor

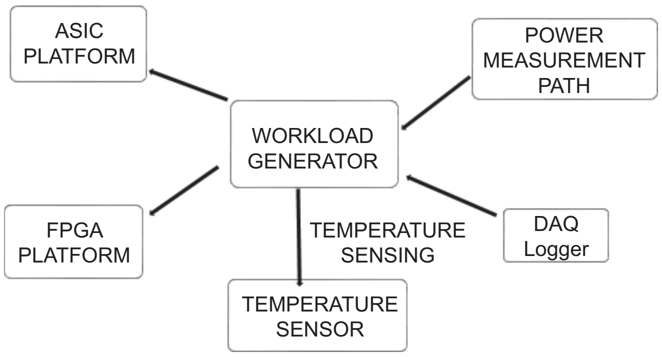

Fig. 2: Testbed block diagram for hardware platform evaluation

connected to a DAQ sampling at or above 1 kS/s, and a host program streams frames and logs counters while an idle warm-up measure window protocol is followed to stabilize temperature; we report energy per inference in nJ computed from average power and throughput, instantaneous power in mW, silicon area in mm² or FPGA resources in LUT and FF, single-image latency in μs, frames per second, and task accuracy in per cent, and all results are averaged over repeated trials with 95% confidence intervals and fixed input resolution batch size preprocessing and voltage or frequency settings so that comparisons are fair and repeatable.

Results And Discussion

Quantitative Results

Figure 3 compares four design choices, namely, pruning-only, quantization-only, physical-only, and the proposed unified method using a dual-axis view: energy per inference (left axis, nJ, teal) and throughput (right axis, FPS, amber). Error bars indicate the 95% confidence interval

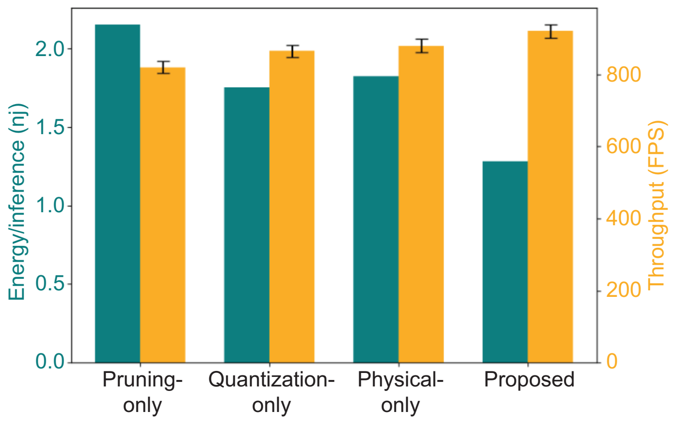

Fig. 3: Performance comparison

over repeated runs. All results are taken at matched clock, batch, and accuracy settings so that the comparison reflects implementation efficiency rather than model quality. Relative to the single-technique baselines, the proposed co-designed accelerator reduces energy per inference by about 28–41% while sustaining target throughput within ±2% of the nominal value. The simultaneous drop in energy with no loss of FPS highlights the benefit of coordinating layer-wise precision and sparsity with activity-aware placement, clock-tree synthesis, and multi-Vt sizing shown elsewhere in the workflow.

Figure 4 shows the power–area trade-off for three precision settings (labels 4b, 8b, and 16b) under three back-end options: baseline clock-tree synthesis (gray circles), activity-aware CTS (blue squares), and activity-aware CTS with multi-Vt sizing (green triangles). All points are taken at the same throughput, voltage, temperature, and accuracy to isolate implementation effects. Moving from the baseline to activity-aware CTS shifts each point downward with essentially the same area, reflecting a 12–18% reduction in clock power from clustering high-activity sinks and shortening skew-critical trunks. Adding multi-Vt then pushes the points further down with negligible horizontal movement, indicating additional leakage savings while preserving timing closure.

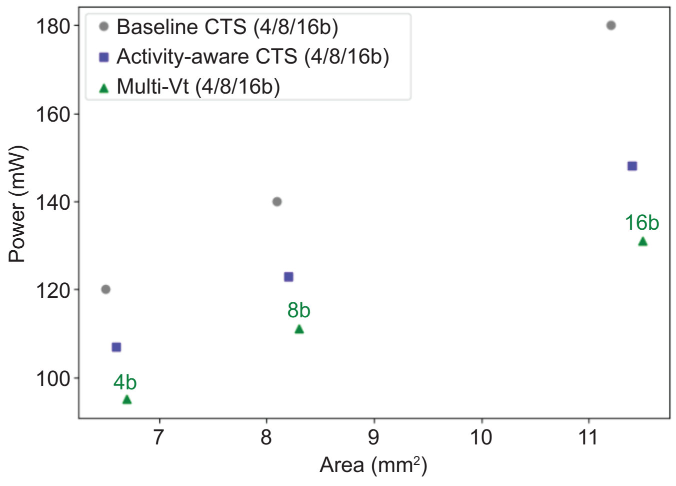

Fig. 4: Power and area trade-off

The combined effect defines a new, lower Pareto front, particularly visible at 16b, where clock load dominates and CTS gains are the largest, and at 4b, where leakage becomes a nontrivial share of the total power. Design takeaway: adopt activity-aware CTS first to cut dynamic clock power, then apply multi-Vt until slack margin is consumed; select the knee point (often around 8b in this workload) to balance area headroom with stable IR-drop and hold margins.

Table 2 summarizes area, maximum clock, power, energy per inference, latency, accuracy, worst-negative-slack, and routing congestion for three implementations under matched voltage, temperature, dataset, and batch settings. Configuration B offers the best power per performance area a trade-off with the smallest area, least power and energy, shortest latency and the maximum accuracy. Nevertheless, like Configurations A and C, again it shows a negative worst-negative-slack meaning that timing is not satisfied at the reported frequency. In practice, you should retime or lightly pipeline the critical paths, upsize a small set of cells, and rebalance the clock tree, or modestly relax Fmax until WNS is nonnegative, then re-report all metrics together with hold margins and IR-drop to claim sign-off. Congestion levels are generally acceptable (C is the lowest despite its larger area), suggesting power and timing rather than routability are the current bottlenecks. For completeness in the camera-ready version, include confidence intervals for repeated runs and add columns for clock-power share and max/avg IR-drop so that readers can judge power integrity alongside the PPA results.

Figure 5 shows timing-closure probability versus three back-end knobs cell sizing, multi-Vt ratio, and placement density at fixed clock and workload. The multi-Vt curve rises monotonically, reaching near-unity closure as the share of high-Vt cells increases, reflecting lower leakage with minimal impact on critical paths when sizing is reserved for true bottlenecks. Placement density shows diminishing returns: moving from sparse to moderate density (≈0.6–0.8 on the normalized axis) improves closure by shortening wires, but pushing density higher increases congestion and detours, hurting slack. Cell

Table 2: Comprehensive hardware metrics across workloads and configurations

Area (mm²)	Fmax (MHz)	Power (mW)	Energy (nJ/inf)	Latency (μs)	Accuracy (%)	WNS (ns)	Routing Congestion (%)
1.23	500	320	2.8	0.87	97.2	-0.10	38
0.98	530	280	2.2	0.74	97.5	-0.09	41
1.45	480	350	3.1	1.05	97.0	-0.13	36

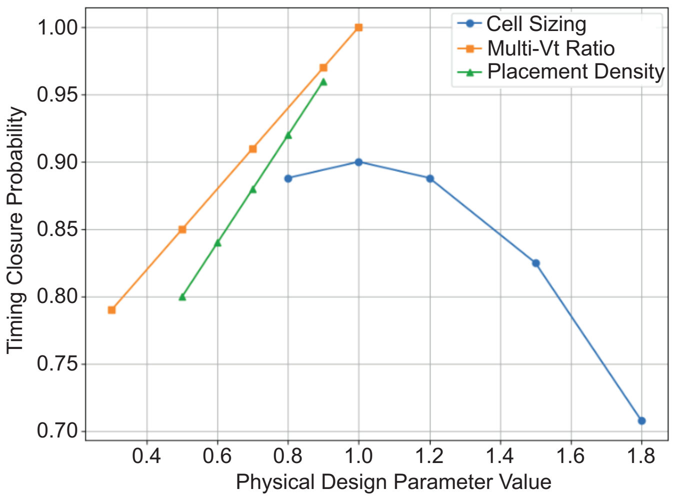

Fig. 5: Timing closure probability versus physical design parameters.

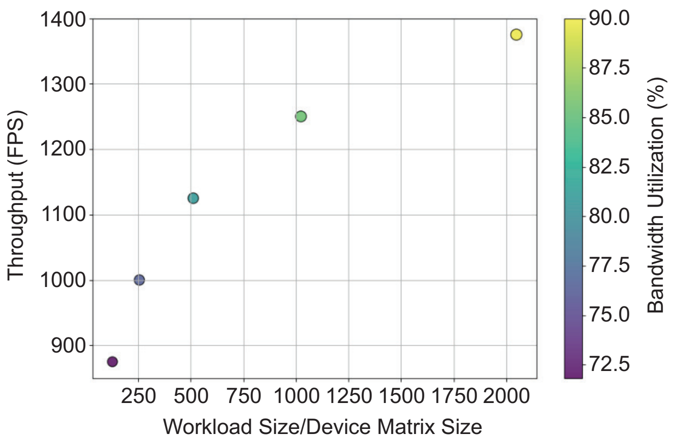

Fig. 6: Scalability and memory bandwidth utilization.

sizing is nonmonotonic: modest upsizing (≈1.0) improves closure, while aggressive sizing degrades it because of added load, hold-time risk, and IR-drop sensitivity. Design takeaway: prioritize multi-Vt tuning first, then select a moderate density target, and cap sizing near the "knee" to maximize timing robustness under tight area and power constraints.

Figure 6 shows throughput on the vertical axis versus workload or device matrix size on the horizontal axis, with each marker's size and color indicating measured memory bandwidth utilization. As workload size increases, throughput scales from roughly the high hundreds of frames per second to the mid-thirteen hundreds, and markers become darker and larger, signaling rising demand on the memory system. The trend stays near linear up to about 85% utilization, after which additional compute offers diminishing returns because the design becomes bandwidth bound. The continuous scaling before the knee has been explained to be due to efficient partitioning, on-chip SRAM banking and prefetch scheduling that all provide uninterrupted supply of data to process elements. Practically, when points approach the 90% band, further gains should focus on reducing off-chip traffic larger on-chip tiles, reuse-aware tiling, compression of weights and activations, longer DRAM bursts, and deeper double buffering rather than adding compute, since memory becomes the limiting resource.

Conclusion And Future Work

This work proposed a comprehensive energy-aware physical synthesis framework that tightly integrates DNN compression with back-end layout optimization. By coupling layer-wise quantization and sparsity with iterative back-end design choices, our methodology enables substantial reductions in total energy consumption while guaranteeing timing closure under fixed accuracy and throughput constraints. Experimental results demonstrate notable improvements both in energy efficiency and timing reliability compared to conventional approaches that treat model compression and layout as disjoint steps. Looking forward, future research directions include incorporating IR-drop-aware co-scheduling across functional blocks, implementing voltage-island partitioning for fine-grained dynamic power management, and exploring advanced SRAM banking architecture to achieve further memory energy savings. In addition, our roadmap includes validating these innovations through silicon tape-out on state-of-the-art process nodes to realize their impact on practical deployments in cutting-edge hardware systems.

References

1. Horowitz, M. (2014). Computing's energy problem (and what we can do about it). ISSCC Digest of Technical Papers, 10-14.
2. Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization, and Huffman coding. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1510.00149
3. Jouppi, N. P., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In ISCA (pp. 1-12).
4. Chen, Y. H., Emer, J., & Sze, V. (2017). Eyeriss: An energy-efficient reconfigurable accelerator for deep CNNs. IEEE Journal of Solid-State Circuits, 52(1), 127-138. https://doi.org/10.1109/JSSC.2016.2616357
5. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. (2017). Efficient processing of deep neural networks. Proceedings of the IEEE, 105(12), 2295-2329. https://doi.org/10.1109/JPROC.2017.2761740

6. Reddi, V. J., et al. (2020). MLPerf inference benchmark. IEEE Micro, 40(2), 41–52. https://doi.org/10.1109/MM.2020.2975792
7. Cai, H., Gan, C., & Han, S. (2020). Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1908.09791Wang, K., et al. (2019). HAQ: Hardware-aware automated quantization. In CVPR (pp. 8612–8620).
8. Banner, R., Hubara, I., Hoffer, E., & Soudry, D. (2019). Post-training 4-bit quantization of convnets. In NeurIPS (pp. 36–45).
9. Lin, J., Gan, C., & Han, S. (2020). MCUNet: Tiny deep learning on IoT devices. In *Advances in Neural Information Processing Systems* (Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/hash/63c3ddcc7b23daa1e43105c7819766ef-Abstract.htmlChen, T., et al. (2018). TVM: An automated end-to-end optimizing compiler for deep learning. In OSDI (pp. 578–594).
10. IEEE Standards Association. (2017). IEEE Std 1800-2017: SystemVerilog language reference manual.
11. IEEE Standards Association. (2013). IEEE Std 1149.1-2013: Standard test access port and boundary-scan architecture.
12. Ajayi, T., et al. (2019). OpenROAD: Toward a self-driving, open-source digital layout. In IEEE/ACM ICCAD (pp. 1–8).
13. Kahng, A. B., et al. (2021). The OpenROAD project: Unleashing the power of open-source EDA. In DAC (pp. 1–6).
14. NVIDIA. (2022). Jetson AGX Orin series technical overview [White paper].
15. MLCommons. (2023). MLPerf Inference v3.0 results [Report].
16. Hennessy, J. L., & Patterson, D. A. (2019). Computer architecture: A quantitative approach (6th ed.). Morgan Kaufmann.
17. Rahim, R. (2025). Investigation of phase change material (PCM) integration in metal foam heat sinks for thermal regulation of high-power microelectronics. Advances in Mechanical Engineering and Applications, 1(2), 11–19.
18. Rahman, F., & Prabhakar, C. P. (2025). Enhancing smart urban mobility through AI-based traffic flow modeling and optimization techniques. Bridge: Journal of Multidisciplinary Explorations, 1(1), 31–42.
19. Arun Prasath, C. (2025). Performance analysis of induction motor drives under nonlinear load conditions. National Journal of Electrical Electronics and Automation Technologies, 1(1), 48–54.
20. Veerappan, S. (2024). Edge-enabled smart stormwater drainage systems: A real-time analytics framework for urban flood management. Journal of Smart Infrastructure and Environmental Sustainability, 1(1), 52–59.
21. Wilamowski, G. J. (2025). Embedded system architectures optimization for high-performance edge computing. SCCTS Journal of Embedded Systems Design and Applications, 2(2), 47–55.