Early-Exit ResNet-18 on Xilinx ZCU102

FPGA deployment + INT8 quantization + adaptive inference · NCSU ECE 591 (SW-HW Co-Design for Intelligent Systems) · Spring 2026

Topics: *FPGA deployment workflow · INT8 post-training quantization · Vitis AI toolchain · DPU subgraph compilation · custom C++ inference harness (VART API) · accuracy-latency Pareto analysis · production-engineering debugging (BGR/RGB pipeline matching)*

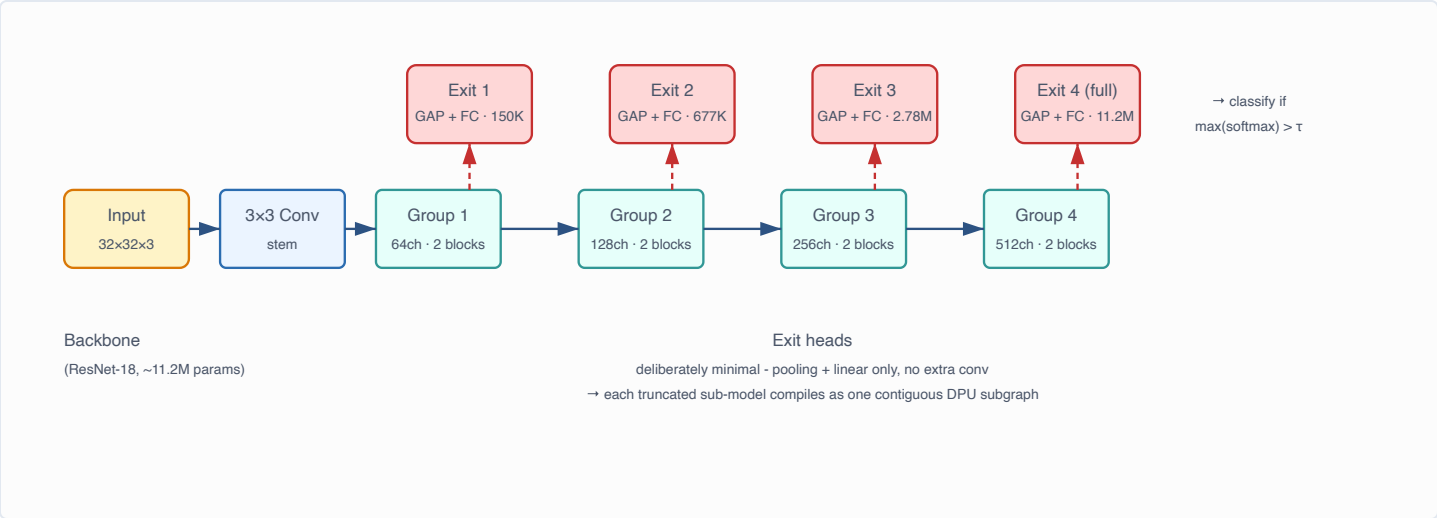
> **The one-liner:** ResNet-18 on CIFAR-10 augmented with four lightweight early-exit heads, jointly trained, quantized to INT8, and deployed on a Xilinx ZCU102 (Zynq UltraScale+ MPSoC) via Vitis AI. A softmax-confidence threshold τ governs how aggressively samples exit early.
>
> **What landed:** At τ = 0.80, 4.01× average inference speedup over the full network (1.392 ms → 0.347 ms projected average latency) while retaining 92.32% top-1 accuracy - only 0.81 pp below the no-exit baseline. Peak per-exit speedup 5.77× (Exit 1 vs. Exit 4).
>
> **My specific role:** Hardware deployment side - INT8 quantization, Vitis AI compilation, DPU-targeting strategy, custom C++ VART inference harness, on-board measurement. My project partner led model training. Architecture (early-exit heads, joint loss) was joint.

What it does

Standard ResNet-18 runs every input through the entire network - ~11.2M parameters, ~1.4 ms per image on the ZCU102. But not every input is equally difficult: many test images get a confident, correct answer well before the last layer (the "overthinking" phenomenon).

The network adds a lightweight classifier head (global average pool + one fully connected layer) at the output of each of the four residual block groups, giving four exit points. At inference, a confidence threshold τ governs the decision: if the max softmax probability at exit i exceeds τ, the prediction ends there; otherwise inference continues. **τ is a deployment-time knob - change it without retraining.**

Architecture

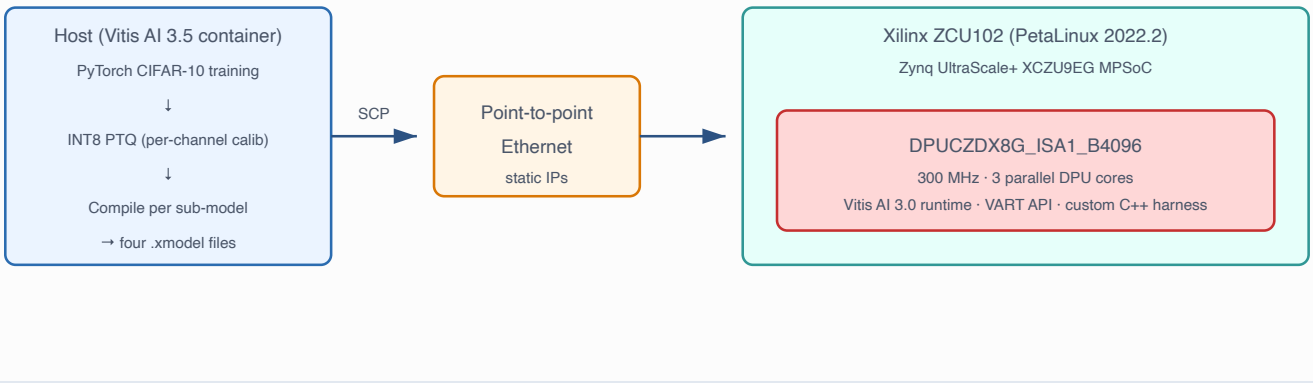


Network structure. Backbone = ResNet-18 for CIFAR-10 (3×3 stem replacing the original 7×7+maxpool). Exit heads after each of the four residual block groups. Each head is just GAP + a single FC layer - kept minimal so each truncated sub-model compiles as one contiguous DPU subgraph with no CPU fallback.

Key architectural decisions

- **Minimal exit heads (GAP + single FC, no extra conv).** Reasoning: BranchyNet-style heavier heads would improve intermediate accuracy but complicate DPU compilation. Keeping them minimal means each truncated sub-model is a single contiguous DPU kernel - no CPU fallback. Parameter cost is negligible: Exits 1-4 add only 640, 1280, 2560, 5120 weights respectively against the 11.2M backbone.
- **Equal-weighted joint loss across all four exits.** Avoids biasing the backbone toward any single exit; gives all heads uniform gradient signal. Side benefit: earlier layers receive more direct gradient signal than in a single-exit network, which encourages discriminative representations to form at shallower depths.
- **Threshold τ as a deployment-time parameter, not a learned one.** Shifts the operating point at deploy time without retraining - a single knob.

Hardware platform & toolchain

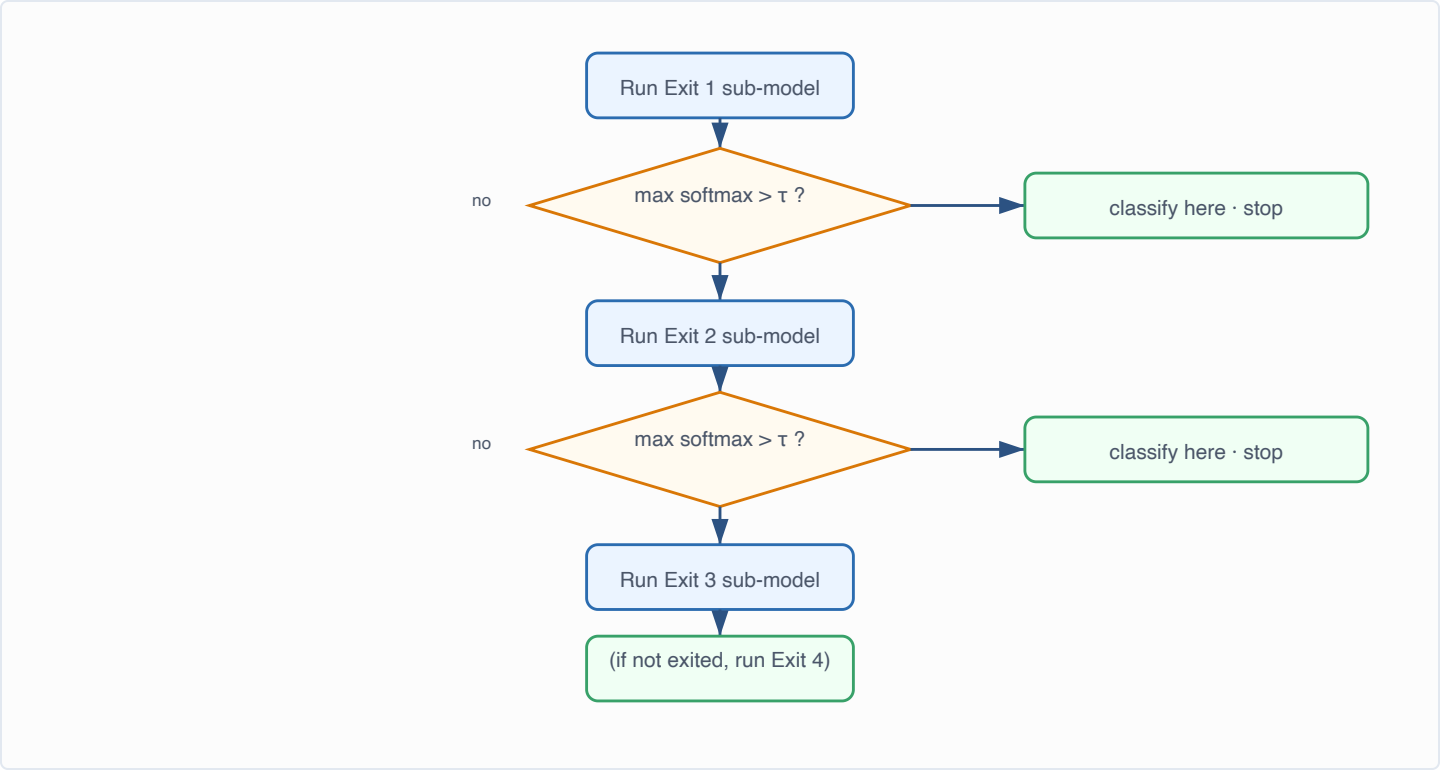


Deployment system. Host quantizes + compiles via Vitis AI 3.5 PyTorch container; `.xmodel` files SCP'd to ZCU102 over direct point-to-point Ethernet. Board runs PetaLinux 2022.2 with Vitis AI 3.0 runtime - forward-compatible with 3.5 host since the underlying DPU IP is unchanged for `DPUCZDX8G`.

Why DPUCZDX8G_ISA1_B4096

The B4096 architecture is the largest DPU config supported on the XCZU9EG; "4096" refers to the peak operations per cycle. Three parallel cores at 300 MHz with INT8 weights/activations. All four sub-models compile cleanly to a single contiguous DPU subgraph each - no CPU fallback for any quantized op.

Inference flow



Inference decision flow. Cascading early-exit logic. A higher τ keeps more samples on the deeper exits (preserves accuracy); a lower τ routes more to Exit 1 (cuts latency).

Training

- 100 epochs, SGD with momentum 0.9, weight decay 1e-4, batch 128
- LR 0.1 with step decay at epochs 50 / 75 / 90
- Joint loss = sum of cross-entropies at all four exits, equal weights
- Standard CIFAR-10 augmentation (random horizontal flip + 4-pixel pad + random crop)

Quantization & compilation

- Post-training INT8 quantization via the Vitis AI quantizer, calibrated on a subset of the CIFAR-10 training set
- Each truncated sub-model (backbone up to and including exit i) compiled *separately* into its own `.xmodel` file - lets each be measured as an independent hardware workload

Numbers

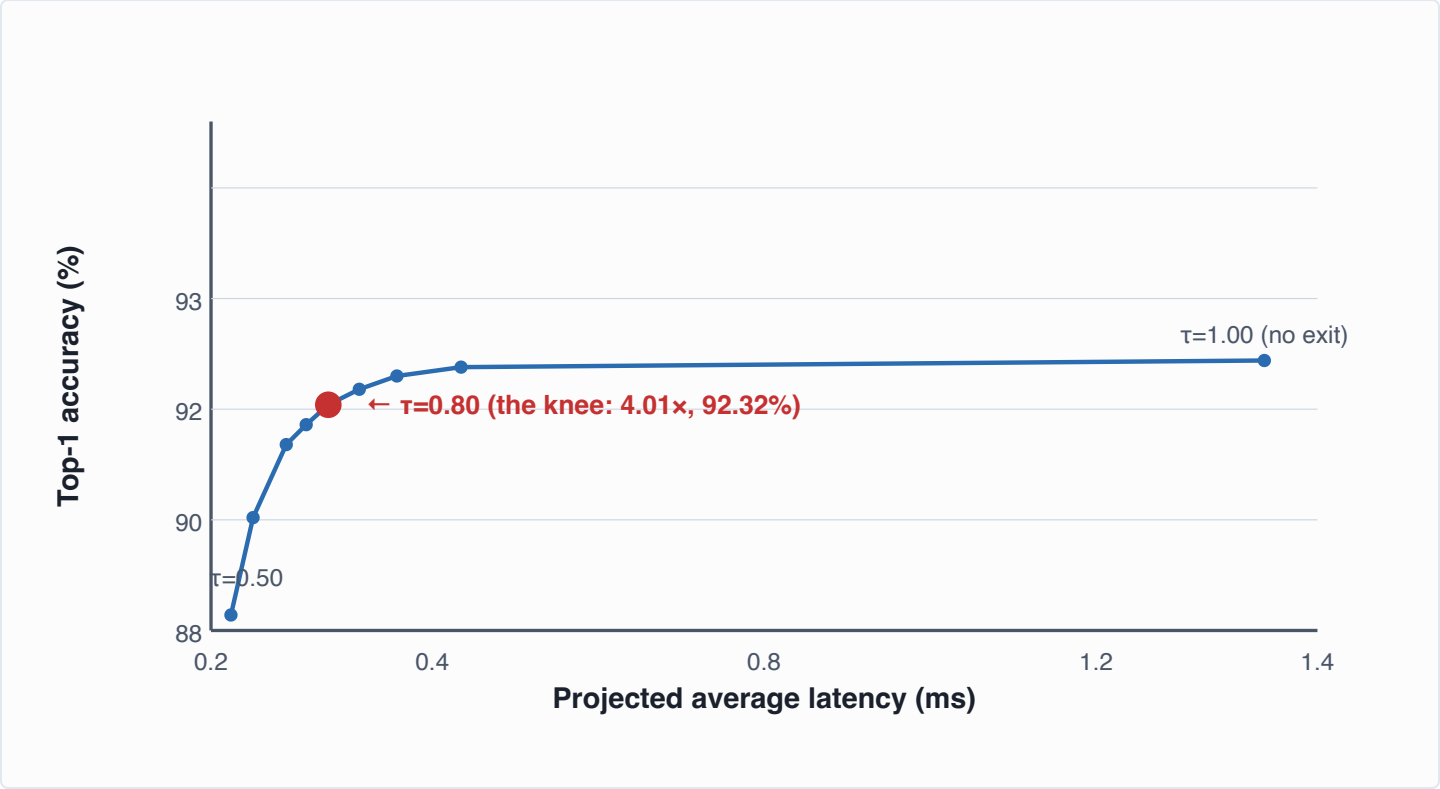
Per-exit on-board measurements (ZCU102, 5,000 CIFAR-10 test images)

Exit	Params	Size	Latency	Throughput	Top-1	Peak speedup
Exit 1	150K	273 KB	0.242 ms	4,131 FPS	83.20%	5.77×
Exit 2	677K	830 KB	0.375 ms	2,667 FPS	91.62%	3.73×
Exit 3	2.78M	2.9 MB	0.533 ms	1,878 FPS	92.14%	2.62×
Exit 4 (full)	11.2M	11 MB	1.397 ms	716 FPS	92.26%	1.00×

Threshold sweep - projected average speedup vs. accuracy (10,000-image sim)

τ	Accuracy	Avg latency	Avg speedup	Exit dist (E1 / E2 / E3 / E4)
0.50	88.50%	0.259 ms	5.37×	90.5 / 8.3 / 0.8 / 0.3
0.70	91.58%	0.309 ms	4.50×	75.7 / 17.9 / 3.6 / 2.8
0.80	**92.32%**	**0.347 ms**	**4.01×**	**67.5 / 21.9 / 5.3 / 5.2**
0.85	92.58%	0.375 ms	3.71×	62.1 / 24.5 / 6.3 / 7.1
0.95	93.01%	0.466 ms	2.99×	45.0 / 31.7 / 10.1 / 13.2
1.00	93.13%	1.392 ms	1.00×	0.0 / 0.5 / 0.0 / 99.5

The Pareto frontier



Pareto frontier of accuracy vs. projected average latency. The knee sits at τ = 0.80 - beyond this point further latency cuts come at sharp accuracy cost; before it, accuracy gains require disproportionate latency.

Why τ = 0.80 is the optimal operating point

- **4.01× average speedup** over the full network (0.347 ms vs. 1.392 ms projected average)
- **92.32% top-1**, only 0.81 pp below the 93.13% no-exit baseline
- **67.5%** of samples exit at Exit 1, 21.9% at Exit 2, 5.3% at Exit 3, 5.2% at Exit 4
- Moving to τ = 0.85 recovers 0.26 pp of accuracy but costs 0.30× speedup
- Moving to τ = 0.75 gains 0.26× speedup but costs 0.37 pp accuracy
- Diminishing returns on both sides confirm τ = 0.80 is the knee

The BGR/RGB normalization bug

> **The most instructive debugging story in this project: a normalization mismatch that the obvious suspects do not explain.**

The Vitis AI tutorial's reference inference application applies `(x/255 - 0.5) × 2` normalization in **BGR channel order** - appropriate for the original TensorFlow tutorial model. But the model was trained in PyTorch with the standard **per-channel mean / std normalization in RGB ordering**.

Without modifying the C++ harness to match training-time preprocessing exactly, on-board accuracy dropped **~17 percentage points** vs. host-side INT8 evaluation, despite identical weights and correct quantization.

The failure mode is sneaky because the obvious suspects (quantization, compilation, runtime mismatch) all check out. The fix was a few lines in the C++ harness to apply the right per-channel mean/std in RGB ordering. After the fix, on-board accuracy matched host-side INT8 within 0.07-0.35 pp across all four exits - a clean closure.

Why it matters: it shows production-engineering discipline - matching the inference pipeline end-to-end - and the kind of debugging where the obvious-looking failure modes are all wrong.

Projected vs. measured

The Vitis AI DPU does *not* natively support per-sample dynamic routing within a single compiled `.xmodel`. So the **per-exit latencies are measured directly** (each truncated sub-model timed on hardware), but the **4.01× average speedup is projected** under a CPU-side scheduler that would run Exit 1 first on the ARM cores and conditionally invoke deeper sub-models based on confidence.

This is well-bounded characterization - it describes the design space available to a heterogeneous edge deployment, not a measured end-to-end result. The distinction is worth stating plainly: the speedup is projected from per-exit throughput, not an end-to-end wall-clock measurement.

Design decisions and trade-offs

What it demonstrates:

- Deployed a PyTorch-trained ResNet-18 with four early exits to a real FPGA (ZCU102), end-to-end through quantization, compilation, PetaLinux deployment, on-board measurement.
- Wrote the C++ VART inference harness from the API up.
- Quantization losses ≤ 0.41 pp at every exit; on-board vs. host INT8 differ by ≤ 0.35 pp - production-grade pipeline consistency.
- Characterized the full accuracy-latency Pareto frontier with a 10,000-sample sweep; identified $\tau = 0.80$ as the knee with concrete justification.

Scope and limitations:

- This is FPGA *deployment*, not from-scratch RTL design. The DPU is a fixed IP block; no gateware was authored.
- The 4.01× speedup is projected under a CPU-side scheduler, not measured end-to-end (see Projected vs. measured above).
- Backbone training was my project partner's lane; the deployment side was mine.
- CIFAR-10 with 32×32 inputs is small; this doesn't directly tell you how Exit 1 would behave on ImageNet-scale data.

Common questions about this project

Question	Short answer
What does this project specifically involve?	The hardware deployment end - INT8 quantization with Vitis AI, compiling each sub-model into its own `.xmodel`, the C++ VART inference harness, on-board measurement, the threshold sweep analysis. My project partner led model training, and the early-exit architecture itself was joint.
Why early exits at all?	Not every input is equally difficult. Many test images get a confident answer well before the last layer - running the rest is wasted compute. Early exits let you trade a tiny accuracy hit for a multiple-x latency reduction without retraining.
Why keep the exit heads minimal?	Anything more than pool + linear would break single-subgraph DPU compilation. Each truncated sub-model has to compile contiguously or you eat CPU-fallback overhead.
What was the hardest engineering problem here?	The BGR/RGB normalization mismatch (see story above).
How trustworthy are the hardware numbers?	Two consistency checks: (1) on-board accuracy is within 0.35 pp of host-side INT8 across all four exits - DPU runtime introduces no real degradation; (2) the truncated Exit 4 sub-model measures 716 FPS, essentially identical to the standalone single-exit ResNet-18 baseline (716 FPS), so the truncation methodology doesn't add overhead.
Why FPGA and not GPU?	Edge / embedded constraints - power budget, latency determinism, and the ability to spec the hardware to the workload. The DPU on the ZCU102 is purpose-built for INT8 inference; for always-on edge inference an FPGA is the right platform.
What would change in a revision?	Two things. (1) Build the actual CPU-side scheduler on the PS-side ARM cores so the 4.01× is measured end-to-end, not projected. (2) Tune the per-exit loss weights - the equal-weight joint loss costs ~2 pp at Exit 4 vs. the single-exit baseline; a small reweight could buy it back.
Could this approach scale to larger models?	The mechanism scales - confidence-thresholded early exits work for any architecture with natural attachment points. The DPU subgraph constraint is the practical limit: if exit heads need to be heavier (extra conv) for larger backbones, you start paying CPU-fallback cost or splitting into multiple kernels.

Question	Short answer
Why ResNet-18 specifically?	Convenient - compact enough to fit comfortably on the ZCU102 with room for the four sub-models, well-studied baseline, and the four residual block groups provide natural attachment points for the exit heads.
How does Vitis AI compare to other FPGA ML toolchains?	The Vitis AI approach is purposeful and constrained: pre-built DPU IP, model has to map to its supported op set, quantization + compilation is largely automated. Wins on time-to-deployment; loses if you need ops the DPU doesn't natively support. For research that benefits from custom gateware (e.g., direct dataflow architectures), HLS or hand-RTL would be the alternative.