EAI 805 — Embedded AI Hardware

Self-Assessment Model Answers

Companion answer key to the weekly lecture notes · platform: AMD/Xilinx PYNQ-Z2 (Zynq-7020)

Full model answers to this week's self-assessment questions, at the level a strong student should reach; usable as a marking guide. Bracketed weeks show where each idea was taught.

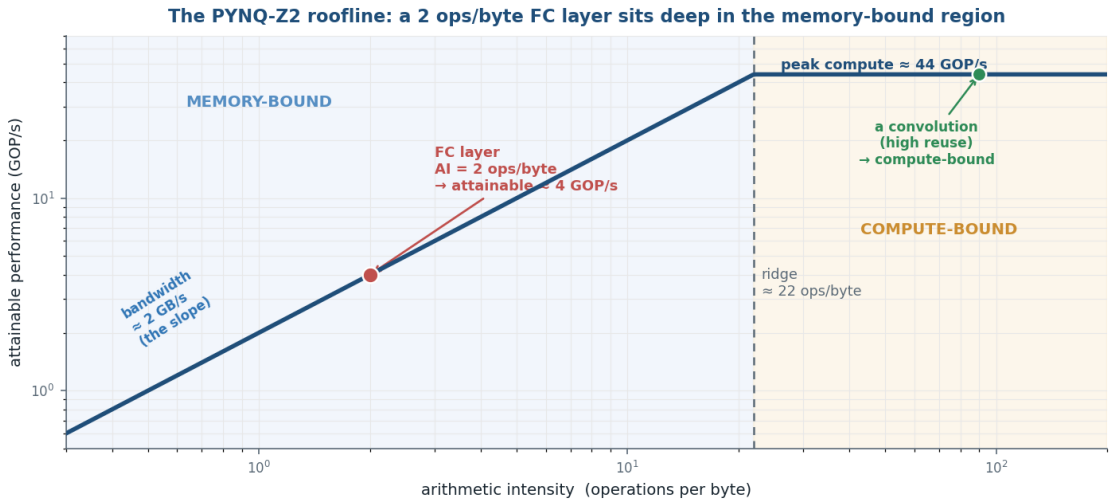
Week 3 — Platforms, the Roofline and Benchmarking

Q1. *Platform A quotes 4 TOPS at 2 W; platform B quotes 40 TOPS at 15 W. Give two reasons why A might deliver lower latency than B on your network, and two reasons why B might win.*

A. A wins on latency when: (1) peak TOPS is a ceiling rarely reached — if your network is **memory-bound** or maps poorly onto B's larger array, B's extra compute sits idle while A's smaller, better-matched engine finishes a single inference sooner (latency ≠ throughput); and (2) for one small inference, **fixed overheads** (host↔device transfer, dispatch) dominate, and A's leaner path or better-matched precision has less overhead. B wins when: (1) for a **large, highly-parallel, high-intensity** network, B's 10× compute genuinely lowers latency and raises throughput; and (2) B likely has far more **memory bandwidth and on-chip memory** to keep its array fed, sustaining more of its peak where A would hit the memory wall. Core point: peak TOPS/W is marketing — real latency depends on utilisation, bandwidth, precision match and overheads.

Q2. *A fully-connected layer has an arithmetic intensity of about 2 ops/byte. Using the PYNQ-Z2 roofline from §4.3, is it compute- or memory-bound? What is the attainable performance, and what would you change first?*

A. The PYNQ-Z2 ridge point is around **22 ops/byte** (compute ceiling ≈ 44 GOP/s over ≈ 2 GB/s bandwidth). An intensity of 2 ops/byte is far left of the ridge, so the layer is **memory-bound**. Attainable performance = bandwidth × intensity ≈ 2 GB/s × 2 ≈ **4 GOP/s** — well below the 44 GOP/s ceiling. Change first: **raise arithmetic intensity, not add multipliers**. Fully-connected layers reuse each weight only once, so batch several inputs to reuse each weight across the batch (or keep weights on-chip). Adding DSPs would do nothing while you are memory-bound.



The PYNQ-Z2 roofline: at 2 ops/byte the FC layer sits deep in the memory-bound region, attaining ≈ 4 GOP/s — far below the 44 GOP/s ceiling. The fix is to raise arithmetic intensity (batch), not add DSPs.

Q3. *Explain why a mean latency of 8 ms may still fail a 10 ms hard deadline. What should you have reported instead?*

A. A hard deadline must be met **every time**, not on average. A mean of 8 ms can hide a long tail — cache misses, contention, DVFS transitions or DMA variance can push the 99th-percentile or worst case to, say, 12 ms, so some inferences miss the deadline and the system fails even though the average looks safe. Report the **tail instead**: the p95/p99 and the worst-case (maximum) latency, ideally the full distribution, because hard real-time is governed by the worst case, not the mean. (This is why the course benchmarks report median and p95.)

> **Q4.** *Choose a platform for a battery-powered wildlife camera doing occasional species classification, and justify it against each of the five axes.*

A. The workload is mostly idle with occasional single-image classification, battery-powered, latency-lax — so **energy dominates**. Choose a tiny always-on-capable NPU/MCU-class accelerator (e.g. an Ethos-U-style microNPU or a Coral Edge TPU, depending on model size) over the PYNQ-Z2. Against the five axes: **Performance** — unimportant, only occasional inference; **Power/energy** — decisive: must sleep at ~µW and wake to classify cheaply (race-to-idle); **Flexibility** — low need, one fixed model, so a fixed accelerator is fine; **Cost** — a cheap MCU+NPU beats an FPGA/GPU for a deployed fleet; **Ecosystem** — a simple deployment path (TFLite Micro / vendor tool). The FPGA's parallel throughput is wasted on an occasional trigger and its static power hurts battery life — the reasoning, not the exact chip, is the point.

> **Q5.** *Your accelerated design is 3× faster than the PS baseline but the peak-TOPS ratio suggested 20×. Give three plausible explanations.*

A. (1) **Memory-bound**: the network's low arithmetic intensity means the accelerator cannot be fed fast enough — it sits on the memory-bound slope, so the extra compute is idle and you get a bandwidth-limited, not compute-limited, speed-up. (2) **Low utilisation / poor mapping**: layers fold onto the array inefficiently (small or odd dimensions, underused PEs), so effective TOPS ≪ peak. (3) **Amdahl / overheads**: DMA transfer, host pre/post-processing and dispatch now dominate end-to-end time, so a large kernel speed-up yields a small system speed-up. (Also possible: the software baseline was better optimised than assumed, or precision conversions add cost.)

Marking note. *Award credit for reasoning that links a principle to a concrete design choice and for correct cross-week connections (the bracketed weeks), not merely for definitions.*