1.1 Understanding NPU Architecture

Before talking about agents on NPUs, we need to talk about the NPU itself — what makes it a distinct class of accelerator, and why the architectural choices ripple all the way up to how you design an agent loop. This book uses **Intel Core NPU** as its primary anchor and **Facebook AI's M2M-100** as its primary worked-example model. Every concept in this chapter ladders back to those two.

What an NPU Actually Is

Neural Processing Units are domain-specific accelerators built for the matrix-multiplication and activation workloads that dominate neural network inference. They sit between CPUs (which are flexible but inefficient at dense matmul) and GPUs (which are powerful but power-hungry and latency-spiky). The NPU's pitch is **sustained matrix throughput at a fraction of the GPU's power budget** — useful for always-on, on-device workloads where battery and thermal headroom matter more than peak FLOPS.

The internal recipe varies by vendor, but every modern NPU combines three things: a **MAC array** (the dense-matmul workhorse, typically 1K–4K multiply-accumulate units per cycle), **on-chip SRAM** sized to hold a tile of activations and weights without round-tripping to DRAM, and a **fixed-function or near-fixed-function activation pipeline** (GELU, Sigmoid, Tanh, sometimes Softmax in hardware). What an NPU does *not* have, by design, is a general-purpose programmable shader array. Generality is the GPU's job.

How the Major Families Compare

Five NPU families currently matter in commercial deployments, and they descend from recognizably different lineages:

Intel Core NPU is the only x86-native NPU and the focus of this book. It inherits its architecture from Movidius — Intel acquired the company in 2016 — and pairs a MAC array with programmable SHAVE VLIW DSPs in the same compute engine. The SHAVEs handle transcendentals, type conversion, and FP32 fallback. Three generations exist: NPU 3720 (Meteor Lake, December 2023, ~11.5 TOPS INT8 claimed but measured at 9.5 TOPS at 1.16 GHz by Chips and Cheese), NPU 4 (Lunar Lake, September 2024, 48 TOPS INT8, on the same compute tile as the CPU and Xe2 iGPU), and NPU 5 (Panther Lake, CES 2026, 50 TOPS INT8, Intel 18A process, with native FP8 support).

Apple Neural Engine is a fixed-function tensor accelerator tightly bound to Core ML on macOS and iOS. The M4 family ships a 16-core Neural Engine at 38 TOPS. Developer access is gated through Core ML — there's no equivalent of OpenVINO that lets you reach the silicon directly.

Qualcomm Hexagon NPU descends from a phone DSP (Hexagon QDSP6) with a bolted-on Tensor Accelerator and Vector eXtensions. Snapdragon X Elite reaches 45 TOPS. The architecture is fundamentally optimized for power efficiency at phone scale; bringing it to laptops is a relatively recent push.

AMD XDNA descends from Xilinx Versal AI Engine tiles arranged in a 2D spatial array. XDNA 2 in Ryzen AI 300 (Strix Point) hits 50 TOPS INT8 plus 50 TOPS Block FP16. Unlike Intel and Apple, XDNA sits as a separate IP block rather than on the main compute die — a different integration model with implications for memory contention.

Google Edge TPU is a fixed-function systolic-array ASIC, primarily for Coral devices and on-device TensorFlow Lite. It's a different deployment story (small embedded modules) and outside the scope of consumer-PC agents.

What's Distinctive About Intel

Four things set Intel apart, and each has practical consequences for agent design:

OS support spans Windows and Linux. The in-tree `intel/linux-npu-driver` makes Intel NPU usable on Ubuntu and other Linux distributions without proprietary blobs in user space. Apple's ANE is macOS-only; Qualcomm's NPU is largely Windows-on-Arm. This matters when your agent's deployment target isn't a consumer laptop — embedded kiosks, industrial edge boxes, server racks running Linux all become viable on Intel NPU.

Developer access is ungated. Every Core Ultra Series 2 or Series 3 SKU exposes the NPU. OpenVINO is Apache-2.0 open source. There's no equivalent of needing a Mac to develop for ANE or a specific Snapdragon SKU to access Hexagon at full capability.

Single-die integration on Lunar Lake and Panther Lake. CPU, Xe2 (or Xe3 on Panther Lake) iGPU, and NPU all sit on the same compute die, sharing an 8 MB memory-side L4 cache on Lunar Lake. AMD's XDNA, by contrast, is a separate block. The integration matters because **agents that hop between devices** — say, NPU for prefill and iGPU for decode — pay less for the hop on a single-die SoC.

OpenVINO ecosystem coverage. OpenVINO is the only unified toolkit that targets CPU, iGPU, NPU, dGPU (Arc), and Gaudi from the same source intermediate representation, with native Hugging Face Optimum-Intel integration. No competing vendor offers this breadth.

The Intel NPU Generation Table

The differences between NPU 3, NPU 4, and NPU 5 are large enough that "the Intel NPU" is not one target — it's three. Code that runs well on NPU 4 may fail to compile on NPU 3, and FP8 paths that work on NPU 5 won't exist on either predecessor.

Generation	SoC / launch	NCEs	SHAVE DSPs	INT8 TOPS (claimed)	INT8 TOPS (measured)	Distinctive feature
NPU 3720	Meteor Lake, Dec 2023	2	4 (Movidius SHAVE)	~11.5	**9.5 @ 1.16 GHz**	First Intel NPU; INT8/FP16; 4 MB total scratchpad
NPU 4	Lunar Lake, Sep 2024	6	12 (SHAVE-V, 4× wider)	**48**	back-calc ~1.95 GHz	4× MACs; NF4 weight compression; single-tile integration
NPU 5	Panther Lake, CES 2026	3 (each 2× wider)	not disclosed	**50**	—	Native FP8 (E4M3/E5M2); programmable LUT for activations; Intel 18A

Per-engine, the MAC array is 2048 INT8 MAC/cycle on every generation. What changes is the count of engines, the SRAM, and the supported data types. NPU 3 totals 4,096 INT8 MAC/cycle; NPU 4 totals 12,288; NPU 5 consolidates back to roughly 12,288 with wider per-engine units and the same Intel-18A area efficiency win.

The Copilot+ certification line (≥40 TOPS) draws cleanly across the table: NPU 4 and NPU 5 qualify; NPU 3 doesn't. If your agent depends on Phi Silica or other Copilot+ OS features, your floor is Lunar Lake or later.

The Hidden Constraint: Memory Bandwidth

TOPS is the marketing number. Bandwidth is the engineering number. **Lunar Lake ships LPDDR5X-8533 on a 128-bit on-package bus, yielding 8,533 MT/s × 128 bits / 8 = 136.5 GB/s of total platform bandwidth shared among CPU, iGPU, and NPU.** There is no private DRAM for the NPU and no published per-device bandwidth quota. This is the single most important number for understanding why LLM decode tops out where it does on Intel hardware.

Intel does not say "decode is DRAM-bandwidth-bound on NPU" in marketing copy — that specific phrasing is a gap in vendor literature. The closest official analog is **Microsoft's Phi Silica blog** (Windows Experience Blog, December 2024): "*Context processing involves intense parallel computation, mainly matrix multiplications, requiring high computational power. In contrast, the*

token iteration stage demands substantial memory for storing and accessing the KV cache for each token generation step. While it needs less computation, efficient memory access is crucial." That's the canonical quotable framing. The roofline becomes tangible on DeepSeek-R1-Distill-Llama-8B INT4: 4 GB of weights streamed at 6.10 tok/s equals about 24.4 GB/s of sustained DRAM read, roughly 18% of platform peak. The NPU does not saturate LPDDR5X; it saturates its scheduling-quota share plus driver overhead.

We'll return to this ceiling in Chapter 1.3 (where it sets the ITL floor) and Chapter 2.1 (where it sets the KV cache wall).

Why M2M-100 as the Worked Model

A book about agentic AI on NPUs needs a concrete model to keep referencing, and we'll use **Facebook AI's M2M-100** — specifically the 418M and 1.2B variants. M2M-100 is a 100-language many-to-many translation model released by Meta in 2020 with three properties that make it a useful teaching example:

It is **encoder-decoder seq2seq**, which forces us to confront the asymmetric NPU/CPU partition that the rest of the field is converging on — encoders fit NPU constraints well (static shape, single forward pass), decoders do not (dynamic shape, autoregressive). M2M-100 makes the partition visible in code, not just in theory.

It uses **full multi-head attention with no GQA or MQA** — the architectural choice that defines its KV cache footprint. In Chapter 2.1 we'll show that M2M-100 1.2B has the same per-token decoder KV bandwidth as Phi-3-mini-3.8B, because Phi-3 uses GQA with one-quarter the KV heads. The KV cache wall is set by attention design, not parameter count, and M2M-100 makes this visceral.

It is **MIT-licensed** (unlike its successor NLLB-200, which is CC-BY-NC 4.0 and unusable in commercial products). The licensing distinction matters more than the technical successor relationship.

It is **not on Intel's validated NPU model list.** This is a feature for our purposes. Production NPU deployment guides usually anchor on validated models (Llama, Phi, Qwen) where everything works. Real engineering happens at the edge of the validated set, and M2M-100 is squarely there.

The honest deployment recommendation, which we'll build up to, is encoder on NPU and decoder on CPU or iGPU. The mechanics of that split are the through-line of the book.

What This Section Bought You

You should now understand:

- **An NPU is a domain-specific matmul accelerator** with on-chip SRAM and fixed-function activations — not a general-purpose shader array
- **Intel Core NPU is distinctive** in OS coverage, ungated developer access, single-die integration, and ecosystem breadth
- **There are three Intel NPU generations** (3720, 4, 5) with materially different capabilities — "the Intel NPU" is not one target
- **Memory bandwidth, not TOPS, is the binding constraint** on Lunar Lake's 136.5 GB/s LPDDR5X-8533
- **M2M-100 is the worked model** for the book — encoder-decoder with full MHA, MIT-licensed, deliberately at the edge of NPU validation

The next section moves from architecture to consequence: given these properties, what computational constraints fall out, and what does optimization look like for an encoder-decoder model on Intel NPU specifically?

Next: *1.2 Computational Constraints & Model Optimization*

Revision #4
Created 2026-05-12 15:55:27 UTC by Admin
Updated 2026-05-12 19:13:48 UTC by Admin