

The Computational Substrate for Physical Intelligence

The ChatGPT moment for robotics requires running robot policies on the edge.

We build the RPU (robot processing unit) — a specialized edge NPU that unlocks true embodied intelligence. Our mission is to scale robotics beyond the demo.

Executive summary

Generalist robotics policies are converging in architecture, but their real-time components cannot run on edge GPUs, forcing teams to rely on kilowatt-class datacenter GPUs and continuous network connections. That puts latency, Wi-Fi jitter, and connectivity failures directly inside the robot's control loop.

We address this by building a specialized edge NPU. Our first silicon targets **3× Thor's performance at the same 40 W**, reaching **5×** as the compiler matures. This will accelerate the real-world deployment of physical intelligence. The upside grows as model architectures converge. Rebuilding the memory system around the workload could push gains toward **~8×**, while hardening stable weights and computation into silicon, similar to Taalas's approach for language models, could reach **~15×**.

0. Alignment

The objective of this paper is to convince the reader of the following chain of reasoning:

1. General robotics policies are getting bigger, and edge GPUs can't run them in real-time.
2. Demos resort to remote inference, but this doesn't scale beyond the demo.
3. Real-world deployment is safety critical, requiring as much onboard intelligence as possible
4. Therefore, there is strong commercial incentive for a specialized edge NPU.
5. We know how to build this specialized edge NPU
6. We can dominate the market

We are not claiming that 100% of the intelligence of a robot needs to be onboard. That would be an ideal scenario. But in reality, physical intelligence is factorized into separate models operating at separate timescales of reasoning: System 2 for slow, deliberate planning, and System 1 for fast, reactive control. We are focusing on S1. We assert that S1 must run entirely onboard to

guarantee reliability, safety, and reaction latency. **Our objective with the RPU is to fit as much intelligence into onboard S1 as possible.**

There are only 2 possible scenarios for the future.

1. Models grow too big to fully fit on any edge silicon.
 - a. RPU provides a competitive advantage — robots with more onboard intelligence will always win in real-world deployments.
2. Models and/or silicon become so efficient that edge GPUs are sufficient
 - a. RPU provides a competitive advantage. The demand for onboard intelligence is effectively unbounded, because more capability wins. Like a gas expanding to fill its container, models grow to use whatever compute a robot can carry — so we expect to stay hardware-constrained for a long time. GPU will remain useful for prototyping, but RPU will dominate real-world deployment.

The scaling laws we discuss in the next section point to the first scenario. But RPU should exist in either scenario.

1. The workload and the scaling laws.

There are several different architectural bets for System 1.

- **Vision-language-action models** map visual observations, language, and often robot state to actions. π -style VLAs combine a pretrained vision-language backbone with a flow-matching transformer that generates continuous action chunks. [π0](#)
- **Generative world-action models**, such as DreamZero, adapt pretrained video backbones to jointly denoise future video latents and corresponding actions. [DreamZero](#)
- **JEPA world models** predict latent futures, but planning actions for multi-stage tasks involves costly search over candidate rollouts. WorldDP separates high-level planning from motor execution: the world model finds feasible subgoals, while a low-level diffusion policy generates the corresponding actions. [WorldDP](#)

Early robotics scaling results show substantial gains from larger models and more pretraining: Generalist reports continued improvements at 10B+ parameters, while Rhoda’s model-scaling experiments raised industrial task completion within a time limit from roughly 4% to 85%. [Generalist](#) · [Rhoda](#) Advances in data collection and human-to-robot transfer are also easing the data bottleneck. UMI enables scalable collection of human egocentric demonstrations using handheld grippers, allowing robot training datasets to grow without requiring a fleet of robots for teleoperation. [UMI](#) PI shows that sufficiently diverse robot pretraining unlocks transfer from human egocentric demonstrations, enabling robots to learn task concepts and operate in settings absent from their robot demonstrations. [PI](#) This suggests a reinforcing scaling process:

broader robot experience makes human experience more useful for training, opening a path to much larger datasets—and potentially much larger models—in the coming years.

The limitations of edge GPUs are already pushing some robotics labs toward remote inference. Physical Intelligence uses cloud inference to test models too large for its onboard GPUs. [Modal](#) Its published $\pi 0.7$ setup uses one H100 for policy inference and four H100s for the separate world model that generates visual subgoals. [\$\pi 0.7\$ paper](#) NVIDIA's own DreamZero, built on a 14B Wan2.1 video backbone, achieves approximately 150 ms inference in its published DreamZero-Flash configuration using two GB200 GPUs. [DreamZero paper](#) Many robotics startups build on DreamZero-based models. The cost of renting these datacenter GPUs can easily exceed the BOM cost of a robot over its operational lifetime.

2. Remote Inference Can't Scale Beyond The Demo

For System 1, the critical metric is reaction latency: the elapsed time between capturing an observation and executing the corresponding action. Human visuomotor reaction time is roughly 200 milliseconds, and delays substantially beyond this range are readily perceived as sluggishness or hesitation. For a robot, this "glass-to-action" latency includes every stage of the pipeline: sensor capture and onboard IPC or bus transfers; transmission of camera frames and proprioceptive state to the datacenter; the model's inference loop; transmission of the generated trajectory back to the robot; and the final onboard processing required before the command reaches the control loop.

Remote inference introduces more than a fixed delay. By the time an action chunk reaches the robot, it was generated from observations that are already stale, and the amount of staleness changes from one cycle to the next. To address variable network jitter, Pi developed real-time chunking and Rhoda developed Leapfrog Inference to overlap model computation with ongoing action execution. [PI](#) · [Rhoda](#). These techniques, as well as buffering and local stabilization, can smooth discontinuities in action chunks, but only by trading reactivity for stability. They cannot recover changes in the world that happened after the observation was captured.

Wireless reliability creates an additional bottleneck. Wi-Fi can exhibit substantial jitter, intermittent packet loss, and abrupt degradation as the robot moves through a building. A system that performs well near an access point may fail when the robot enters a basement, passes through a doorway, or encounters network congestion. During evaluations of Sunday Robotics' autonomous ACT-2 policy in real homes, unpredictable Wi-Fi latency and network jitter reportedly emerged as major bottlenecks and failure modes, motivating a shift toward local, edge-compute control. An autonomous robot cannot treat continuous datacenter connectivity as a dependable part of its control loop.

Real-world deployment is safety critical

We must not have datacenter latency and network jitter in the real-time control loop. Robots must operate in safety critical environments like households, workplaces, and public spaces. Safety requires low latency, deterministic control with ultra high reliability.

Consider three everyday scenarios:

- **Cooking an omelette:** A disruption in the connection can result in a severe fire hazard.
- **Descending stairs:** A disruption in the connection when walking down into the basement results in the robot either tumbling, or suddenly stopping. Both outcomes endanger people on the staircase.
- **Carrying a heavy object:** A disruption in the connection may result in dropping the heavy object or flinging it uncontrollably.

Techniques to make robots control more robust to network transients often trade smoothness for reactivity, but reactivity is also safety-critical for activities involving human interaction and collaboration.

Local safety controllers and reinforcement learning policies can address many of these hazards, but there is a long tail of events that can only be truly squashed by putting more intelligence on the edge, and keeping the real-time control loop onboard. This is the same reason that self-driving vehicles run their full planning loops onboard.

Engineering Complexity

Historically, engineering complexity has been the main deciding factor in engineering decisions. Coding agents have reduced this to some degree, but it is still a serious consideration. And certainly, operational burden remains tightly coupled with engineering complexity.

Some amount of remote inference may be unavoidable, but we believe it should be relegated to System 2 (non-real-time planning). Remote inference for System 1 (real-time control) introduces additional engineering requirements, including low-latency observation and action streaming, coordination with local execution, and handling network delays and interruptions. Many companies have had to build custom ultra-low-latency QUIC transport and distributed relay systems for bidirectional streaming of camera frames and trajectories. Semianalysis emphasizes that with remote inference, networking becomes the single biggest problem in robotics. They provide the example of Boston Dynamics, which has multiple teams dedicated to nothing else, covering the radio stack on the robot, the customer's infrastructure, spectrum, and protocol design. And when so many things can go wrong, real-time observability into the system becomes a very significant challenge. Unfortunately, every company is heavily inclined to create their own remote inference systems — there is a tight coupling between model inference and control stack. Data streams are proprietary and feed into other systems (e.g. teleoperation, training pipeline) as

well, and robot embodiments uniquely determine the shape of the data transport. It would be difficult to convince a robotics company to outsource this, just as it would be difficult to convince them to outsource their data pipelines. These systems require lots of customization and control authority.

3. Why Edge GPUs Are Not Good Enough: We Need a Transformer ASIC.

General-purpose hardware pays an energy cost for flexibility. Hameed et al. found that GPUs spend **less than 10% of energy on arithmetic**. Most of the energy is wasted on generality that is not needed as model architectures converge. [Hameed et al., 2010](#).

Arithmetic efficiency is especially important for diffusion transformers. At batch size 1, their inference tends to be **compute- and power-limited**, while autoregressive LLM decoding tends to be **memory-bandwidth-limited**:

- **Linear layers reuse weights across tokens.** Processing L latent tokens together performs L times the arithmetic of single-token decoding for the same weights.
- **Attention reuses keys and values across queries.** Full self-attention computes L^2 query-key pairs, or roughly half that with causal attention.

Google's original TPU demonstrated the value of combining specialized matrix hardware, software-managed memory, and predictable execution. The same principles motivate our design. Our target is **lower energy per completed inference at the required latency and model quality**.

4. Transformer ASICs Exist for Datacenters, but the Edge Requires A Different Design

LLM architectures have converged enough to make specialized datacenter chips viable. Robotics is a few years behind, but appears to be following the same path.

However, the engineering constraints are somewhat reversed. Datacenter LLM decoding is bottlenecked by the **memory-bandwidth wall**: chips can process data faster than memory can supply it. This drives innovations in HBM, caching, batching, and advanced packaging. But in a 40 watt robotic head, edge inference for diffusion transformers is bottlenecked by energy efficiency.

Robots need low-latency, **batch-1 inference**, acting on their latest observations without waiting to assemble a batch. But batch 1 does not imply low compute intensity. A diffusion transformer

processes thousands of latent tokens together, reusing its weights across those tokens. That demands far more arithmetic per byte fetched than generating one token at a time.

A DreamZero-class model makes this concrete. Under the workload assumptions here, its 14 billion parameters across 40 layers require about **7×10^{14} FLOPs per control chunk**. Each diffusion step streams **7.0 GB of 4-bit weights** and a **7.7 GB FP8 context window—roughly 44 GB across three steps**. That gives an estimated arithmetic intensity of **16,000 FLOPs per byte**.

Assuming **220–300 FP4 TFLOP/s at 40 W** and Thor’s rated **273 GB/s memory bandwidth**, its roofline crossover sits around **800–1,100 FLOPs per byte**. [NVIDIA](#) The workload sits roughly **15–20× above that threshold**, making compute the dominant constraint in this estimate. Within a fixed power budget, the priority is to deliver more useful computation per joule.

5. The Robot Processing Unit (RPU): A Technical Roadmap To Beat Jetson Thor

As mentioned in the previous sections, there is some convergence among the most widely deployed generalist robotics policies. They combine separate visual encoders with diffusion/flow-matching transformers, conditioned on latent visual representations, language, and robot state. They generate action chunks that are refreshed from new observations for responsive closed loop control and try to achieve human-like reaction latencies (200ms).

We will design the RPU around this workload. Inference runs at batch one, and all the sensor streams (proprioception, camera frames) are encoded together into a single representation. This exposes substantial parallel work across tokens and features within each layer.

First-silicon design

The RPU is an output-stationary systolic array. It uses predictable execution, specialized arithmetic, and explicit memory management to reduce the energy required for each inference:

- **Static execution.** Compile computation and data movement into a hardware schedule, keeping state between inference cycles and avoiding per-operator kernel launches. Weights and schedules remain loadable; array geometry and supported numerical formats stay fixed.
- **Specialized compute.** Target approximately 1.9 million MAC units at 1.05 GHz, or 4 PFLOP/s of theoretical dense throughput, organized into 128×128 systolic tiles. An output-stationary dataflow keeps partial sums local.
- **Low-precision arithmetic.** Support FP4 weights, FP8 activations, and FP32 accumulation, with a dedicated FP8 attention path.

- **Compiler-managed memory.** Target approximately 90 MB of SRAM for activation storage and stream buffers, backed by at least 16 GB of LPDDR5X over a 256-bit interface providing 307 GB/s peak bandwidth. Weights stream through the chip; persistent context lives in a DRAM ring buffer.
- **Local reuse and fusion.** Stage operands in SRAM, distribute shared weights and activations across MAC tiles, and keep attention intermediates on chip wherever possible.
- **Limited programmability.** Explore a small programmable path, targeting roughly **2% of the datapath**, for operations such as flow-ODE integration and CEM planning.

These are initial design assumptions, not simulation results. Simulation will determine the appropriate tile count, memory capacity, and scheduling.

Turning efficiency into lower latency

At a fixed power budget, **a 5× improvement in chip-level useful operations per joule can support up to 5× more throughput.** We aim to turn that gain into lower latency by adding parallel MAC tiles while keeping the clock frequency unchanged.

This requires sufficient parallel work within each layer, enough bandwidth to feed the tiles, and attention, interconnect, and synchronization that keep pace. More MACs increase instantaneous operand demand, so we must parallelize along dimensions that reuse data. SRAM staging and operand sharing reduce repeated DRAM reads, while increasing the demands on on-chip memory and distribution.

The area advantage remains a hypothesis. Supporting fewer numerical formats and a narrower workload family should let us simplify precision handling and general-purpose control, leaving more die area for arithmetic and memory. Physical implementation will determine how much area this saves.

Execution roadmap

We target **3× better performance per watt than Thor with first silicon**, **5× as the compiler matures**, and ultimately **15× through hardware–model codesign**. These targets will be tested through simulation and prototyping.

Stage	Target	What enables it
First silicon	~3×	Static execution, specialized arithmetic, fused attention, and efficient data reuse.
Mature compiler	~5×	Better utilization, tiling, fusion, transfer overlap, and clock gating on the same silicon.
Model-resident design	8–12×	Stack DRAM directly above compute to increase bandwidth and reduce transfer energy.
Hardware–model codesign	~15×	Implement stable weights and operator sequences in dedicated circuitry while retaining flexibility elsewhere.

Keeping memory ahead of compute

As we add parallel compute, the memory system must supply data fast enough to keep the MACs busy. We will size compute, SRAM, and external-memory bandwidth together, using actual traffic and sustained bandwidth to identify bottlenecks.

Our first priority is to get more work from each byte fetched: reuse weights and activations, keep intermediates on chip, and overlap transfers with computation. Where more bandwidth is needed, we can widen the memory interface or trade some compute area for memory resources, within the same power budget.

Longer term, stacked memory could increase bandwidth while reducing transfer energy. Cooling, capacity, cost, and package reliability will determine when that becomes practical.

Simulation and validation through FPGA

We can reuse substantial open-source infrastructure. **Chipyard already integrates Gemmini, a configurable systolic-array accelerator with output-stationary execution, banked SRAM, DMA, and a Verilator simulation workflow.** This gives us an implemented baseline for testing dataflow and memory reuse while developing the RPU’s specialized arithmetic, static scheduling, and fused attention. [Gemmini documentation](#)

Our methodology separates **measured execution and activity** from **modeled ASIC energy**:

1. **Explore with Timeloop.** Sweep geometry, accumulator depth, SRAM capacity, and scheduling. Calibrate predicted cycles and traffic against RTL simulation, then test predictions on configurations outside the calibration set. Evaluate fused attention directly in RTL where Timeloop cannot represent it faithfully.

2. **Implement with Chipyard and verify with Verilator.** Generate parameterized RTL from Chisel and check it against an independent golden model. Arithmetic-preserving changes must match bit for bit; numerical changes must meet explicit error budgets and model-quality criteria. Iterate on small configurations and confirm selected larger designs.
3. **Model energy from measured counts.** Apply cited per-event energy coefficients with their process, voltage, precision, and scope stated. Cross-check against published accelerator measurements, report discrepancies, and identify omitted costs—especially interconnect across the full chip.
4. **Validate a representative design on FPGA.** Reuse **Chipyard's existing FPGA shells and Vivado build flow** on a supported board. These supply clocking, resets, external-memory connections, and basic I/O; we integrate the accelerator through a custom configuration. Test longer workloads and memory behavior, reporting the FPGA's actual clock, utilization, and traffic. These validate the execution model; FPGA watts and frequency do not directly establish ASIC performance. [Chipyard FPGA documentation](#)

This phase uses **Timeloop, Chipyard, Verilator, and FPGA verification**. Physical implementation will later establish achievable ASIC frequency, die area, and power.

6. Addressing Jensen's Law

Jensen's Law asserts that the incumbent's hardware improves by roughly **2× a year** (which is way too generous as an expectation, but valid as an upper bound). To address it, we first have to ask where those gains come from. Most of them are advances that benefit everyone: better silicon, denser SRAM, faster DRAM, improved packaging, and lower-voltage transistors. We ride the same wave as the incumbents.

Consider the TPU. Since 2013, GPUs have improved by roughly **100×**, but the TPU never needed to target 100× up front. It only had to be consistently a few times better at every stage — a multiplicative architectural factor on top of the same technology curve — which it achieves purely through the topology of the chip.

In our view the right framing is two questions: given access to the same physical primitives, what is our chip's performance multiplier? And what multiplier makes people willing to switch? The TPU shows it can be as low as **3×**; Groq's LPU and Etched suggest it may be closer to **10×**.

Our goal is the same: to stay consistently some multiple better than the incumbent at every stage, riding the same wave of general improvement. We target a modest **3×** for our first silicon and **5×** as the compiler matures; the long-term vision is to co-design the RPU once model architectures converge, reaching **>15×** in energy efficiency.

We believe these multipliers can emerge purely from removing the generality tax of GPUs, which spend **less than 10%** of their energy on arithmetic and more energy than necessary on each operation. Energy efficiency is the product of these two factors, and it is the right metric because edge inference is bottlenecked by compute, not memory bandwidth — and diffusion transformers are more compute-dense than autoregressive LLMs.

7. The Market Opportunity: Win the transition to edge inference

As discussed previously, edge inference will require models and hardware to improve together. The emergence of scaling laws suggests the generalist robotics policies will continue to grow in size, and robots that can fit more of that intelligence onboard will always have a competitive advantage. As the industry scales edge inference, the RPU architecture will remain more competitive than the GPU, because it is a specialized ASIC. Nvidia is too deeply invested in general purpose accelerators. It is the same reason they did not develop Groq's LPU in-house. It is also the same reason that google sat on the transformer and did nothing with it — they could not sabotage their bet on page-rank-search.

In the near term, we hope to win deployments, form customer relationships, and attain the workload knowledge needed for the larger opportunity. When the model architectures finally converge, we will be positioned to co-design the model and chip as a single system — creating an accelerator that NVIDIA's general-purpose architecture cannot match.

We can move fast because an emerging ecosystem is lowering the barriers to chip development. Reusable hardware, open-source simulation tools, and FPGA prototyping let small teams test an architecture before committing to silicon. Chipyard and Gemmini provide an existing foundation, allowing us to focus on the design choices that matter for robotics. [Gemmini documentation](#)

There is a precedent for what we are doing. A few years ago, datacenter LLM architecture largely converged, and a market opportunity opened up for Etched's transformer ASIC and Taalas's approach of hardening a model directly into the fabric.

We are focused entirely on robotics. We see robotics following the same architectural trend, a few years behind LLMs. As robot policies stabilize, we expect a similar market opportunity for specialization at the edge, where none of the datacenter incumbents are looking. Robotics has a set of unique constraints, and we can design around those from day zero.

Our long-term conviction is that **robotics compute can become a larger market than datacenter compute**. If general-purpose robots become commonplace in factories, warehouses, and homes, each will need reliable local intelligence. Elon claims there will be more robots than humans. We hope that each one will contain an RPU brain.

8. About The Author

I majored in Physics at Caltech. Then, I helped build *and operate* the world's first satellite mega constellation, scaling it to 10 thousand satellites and 10 million users. After 4 years at SpaceX, I left to work at a frontier robotics lab. I was responsible for running world models in real time for robotic control. We optimized the model architecture extensively, and pushed GPU utilization to the limits. But we still could not run on the edge, so I had to build a complex remote inference solution. I realized that this is the wrong solution, and that it cannot scale beyond the demo. Now, I'm on a mission to build towards a future of true, embodied intelligence.

I am building a team around this vision. Our angel investors are co-founders and current/former execs of well known robotics companies. We are advised by current/former execs of well known chips companies. Next spring, I will onboard an additional cofounder who is currently completing his PhD at Harvard on ASIC design.

Lessons Learned At SpaceX

In Starlink, we also factored physical intelligence into System 2 (deliberate planning) and System 1 (autopilot). We ran System 2 remotely, on the ground, but it was essential that we ran System 1 onboard the satellite, and we sized the hardware accordingly. I see many similarities in robotics, but arguably households and workplaces are much more safety critical environments than low earth orbit, and I don't believe the industry is sizing the hardware correctly.

There is no such thing as pure software. Fundamentally, software is constrained by hardware. We must dare to imagine the system as a whole, and improve the hardware as well.

9. The Most Inspiring Future: True Embodied Intelligence

"Then you do not remember a world without robots. There was a time when humanity faced the universe alone and without a friend. Now he has creatures to help him, strong creatures than himself, more faithful, more useful, and absolutely devoted to him. Mankind is no longer alone."

— Isaac Asimov

I imagine living in a world filled with robots, like in Star Wars. In such a world, I don't want robots to be puppets controlled by datacenters hundreds or thousands of miles away, at the mercy of network transients. This would be a deeply unsatisfying vision of the future and falls far short of the scifi dream of true embodied intelligence.

Looking toward the horizon, we will need to deploy embodied intelligence to achieve humanity's most ambitious projects. Robots will build and operate the moon base. Remote inference is intractable at those distances, unless we put datacenters on the moon, but then it becomes a chicken-and-egg problem. How do we set up all that infrastructure without robots?

10. References

- [1] NVIDIA, "Jetson Thor Series Modules Datasheet" (T5000: dense FP4 1,035 TFLOP/s at MAXN; 273 GB/s LPDDR5X).
- [2] DreamZero authors, "World Action Models are Zero-shot Policies," [arXiv:2602.15922](https://arxiv.org/abs/2602.15922), 2026.
- [3] Q. Shen et al., "World Action Models: A Survey," [arXiv:2606.20781](https://arxiv.org/abs/2606.20781), 2026.
- [4] N. Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit," ISCA 2017.
- [5] N. Jouppi et al., "TPU v4: An Optically Reconfigurable Supercomputer," ISCA 2023.
- [6] R. Hameed et al., "Understanding Sources of Inefficiency in General-Purpose Chips," ISCA 2010.
- [7] FLASH-D, [arXiv:2505.14201](https://arxiv.org/abs/2505.14201); ExpMul, [arXiv:2505.14314](https://arxiv.org/abs/2505.14314).
- [8] WorldCache (content-aware caching for video world models), [arXiv:2603.22286](https://arxiv.org/abs/2603.22286); Chorus (inter-request caching reuse), [arXiv:2604.04451](https://arxiv.org/abs/2604.04451); Sliding Tile Attention, [arXiv:2502.04507](https://arxiv.org/abs/2502.04507).
- [9] K. Black, M. Y. Galliker, and S. Levine (Physical Intelligence), "Real-Time Execution of Action Chunking Flow Policies," [arXiv:2506.07339](https://arxiv.org/abs/2506.07339), 2025.
- [10] Efficient-WAM (1 B world-action model), [arXiv:2606.10040](https://arxiv.org/abs/2606.10040); LaWAM (latent world action models), [arXiv:2606.15768](https://arxiv.org/abs/2606.15768); NanoVLA, [arXiv:2510.25122](https://arxiv.org/abs/2510.25122).
- [11] Epoch AI, "Trends in GPU price-performance" — measured FLOP/s per watt doubling time.