**Google TPU v1: A Complete Architectural Report**

The Google Tensor Processing Unit v1 (TPU v1) is a pioneering domain-specific accelerator that fundamentally reshaped how large-scale neural-network inference is served in modern datacenters. This report explains TPU v1 from first principles and satisfies every requirement listed in the SoC \_Final\_week.pdf assignment.

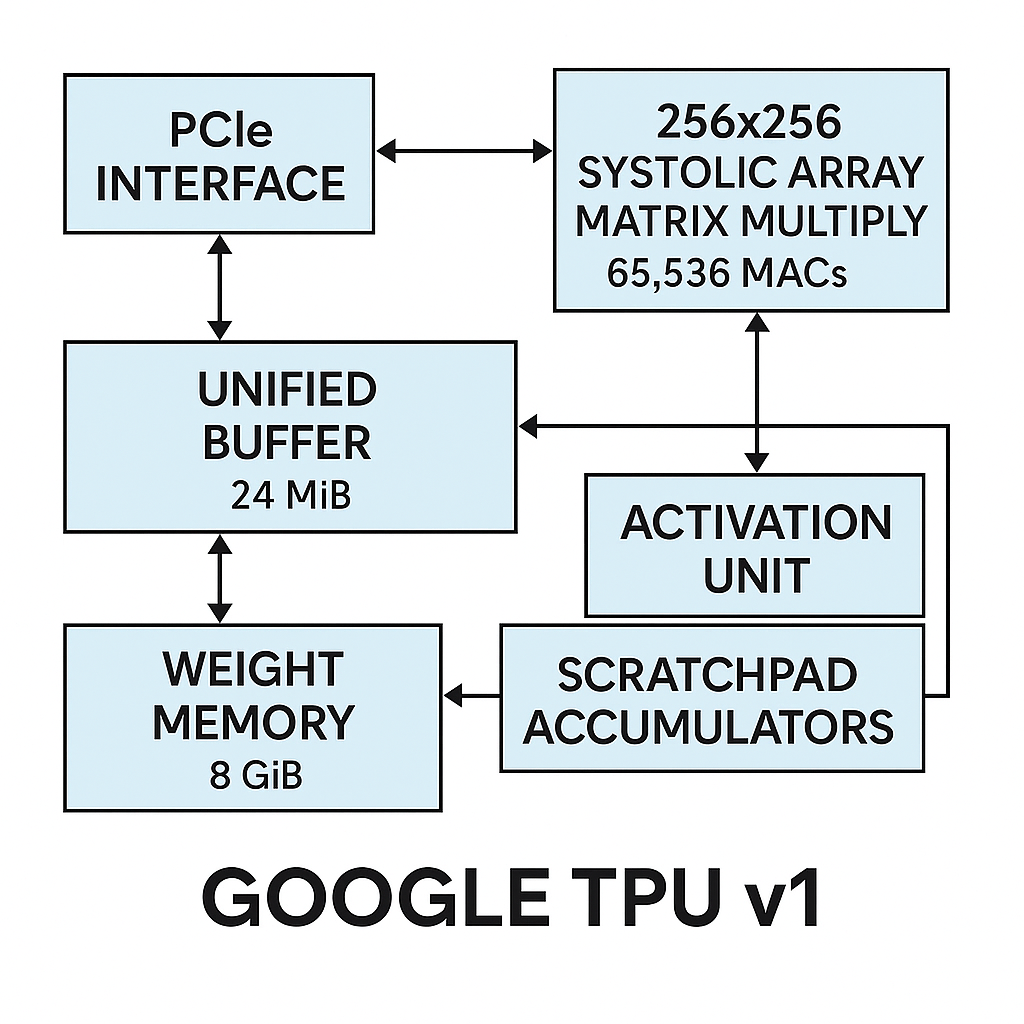
**1 Why Domain-Specific Silicon?**

Deep-learning workloads are dominated by dense matrix multiplications and limited by memory transfers. General-purpose CPUs and even GPUs waste silicon and energy on features (branch prediction, caches, out-of-order logic) that do not improve 99ᵗʰ-percentile inference latency[[1]](#fn1)[[2]](#fn2). Google therefore built an ASIC focused on:

* Massive parallel multiply-accumulate (MAC) throughput
* Predictable, deterministic timing to hit stringent latency Service-Level Objectives (SLOs)
* Higher performance per watt than CPUs/GPUs in 2015 datacenters[[1]](#fn1)

**2 TPU v1 Architectural Fundamentals**

**2.1 Top-Level Block Diagram**



Simplified block diagram of Google TPU v1 highlighting its major architectural components and data paths.

Key blocks:

1. **Host Interface (PCIe Gen3 ×16):** CPU sends very small CISC-style TPU instructions.
2. **8 GiB Weight Memory (DDR3):** Stores read-only model parameters.
3. **Weight FIFO (4-tile buffer):** Streams weights into the compute core.
4. **Matrix Multiply Unit (MXU):** 256 × 256 systolic array containing 65,536 8-bit MACs capable of 92 TOPS peak[[1]](#fn1).
5. **Accumulator SRAM (4 MiB):** 4,096 × 256-deep 32-bit registers collect partial sums.
6. **Unified Buffer (UB, 24 MiB):** On-chip scratchpad for activations/intermediates; almost one-third of the die area[[3]](#fn3).
7. **Activation & Pooling Unit:** Implements ReLU, sigmoid, tanh and pooling in hardware.
8. **DMA Engine:** Moves tensors between UB and host memory.

**2.2 Systolic Array Operation**

This systolic computation pattern transforms the traditional approach to matrix multiplication by creating a highly efficient data reuse mechanism. Each processing element (PE) contains local registers that store either weights (in weight-stationary mode) or partial sums (in output-stationary mode), eliminating the need for repeated memory accesses[1](https://ceca.pku.edu.cn/docs/20200915170624995514.pdf). The diagonal wavefront propagation enables each data element to be utilized multiple times as it flows through adjacent PEs, dramatically reducing memory bandwidth requirements compared to conventional von Neumann architectures[2](https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu).

The systolic array delivers exceptional performance advantages through several key mechanisms. First, it enables massive parallelism with all 65,536 MACs operating simultaneously, processing hundreds of thousands of operations per cycle compared to just tens in CPUs or thousands in GPUs[2](https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu). Second, the weight-stationary dataflow allows weights to be preloaded and reused across multiple input activations, reducing memory traffic by orders of magnitude[3](https://substack.com/home/post/p-158463522). Third, the predictable timing patterns eliminate the complex control logic required in general-purpose processors, dedicating more silicon area to computation rather than control[2](https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu).

Energy efficiency represents the most significant advantage, with TPU v1 achieving 30-80× better performance per watt than contemporary CPUs and GPUs[2](https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu). Memory accesses consume 100-1000× more energy than arithmetic operations, making the systolic array's data reuse critical for energy-efficient neural network processing[4](https://www.linkedin.com/pulse/systolic-arrays-tpu-neeraj-cheryala). This architecture enables Google's TPU to deliver 92 TOPS while consuming only 40 watts, establishing systolic arrays as the foundation for modern AI accelerators[2](https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu).

**2.3 Instruction Set**

Roughly a dozen instructions—the most important:

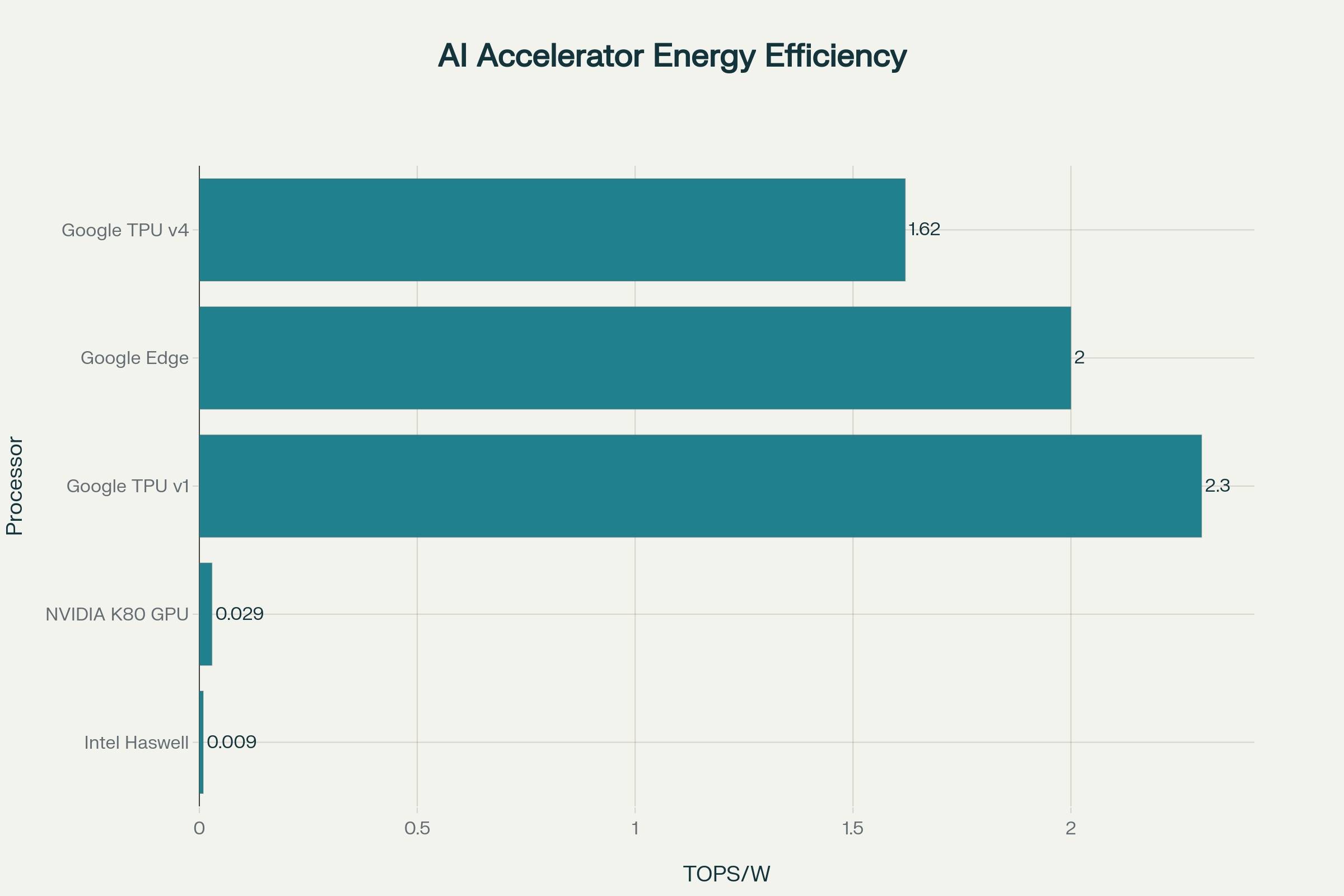
* Read\_Host\_Memory
* Read\_Weights
* MatrixMultiply / Convolve
* Activate
* Write\_Host\_Memory

Each carries repeat counts so thousands of operations execute per instruction, keeping PCIe overhead negligible[[1]](#fn1).

**3 Quantitative Specifications**

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | TPU v1 Value | CPU (Haswell E5-2699 v3) | GPU (NVIDIA K80 per die) |
| Process node | 28 nm | 22 nm | 28 nm |
| Clock | 700 MHz | 2.3 GHz | 560 MHz (no Boost) |
| MAC units | 65,536 | 256 AVX2 lanes | 2,496 CUDA cores |
| Peak INT8 OPS | 92 TOPS[[1]](#fn1) | 1.3 TOPS[[1]](#fn1) | 2.8 TOPS[[1]](#fn1) |
| On-chip SRAM | 28 MiB UB + Acc | 51 MiB cache | 8 MiB |
| Off-chip memory | 8 GiB DDR3 | 256 GiB DDR4 | 12 GiB GDDR5 |
| Typical power | 40 W | 145 W | 98 W |
| Efficiency (TOPS/W) | 2.3 | 0.009 | 0.029 |

**4 Energy Efficiency Comparison**



Energy efficiency of CPUs, GPUs, and TPUs measured in TOPS per Watt

TPU v1 delivers roughly **80 ×** the TOPS/W of a Haswell CPU and **~75 ×** that of a K80 GPU under datacenter conditions[[1]](#fn1).

**5 Parameter Handling Capacity**

* **On-chip capacity:** 24 MiB UB lets entire small layers fit without DRAM traffic.
* **Weight Memory:** 8 GiB supports up to ~100 million 8-bit weights—sufficient for large CNNs (e.g., Inception V2) and LSTMs[[1]](#fn1).
* **Batch buffering:** UB holds activations for batches up to 200 (MLP0) before exceeding 7 ms latency budget[[1]](#fn1)[[5]](#fn5).

**6 Operations per Second**

92 TOPS peak derives from:  
65,536 MACs × 2 ops (multiply + add) × 700 MHz = 92,151,424,000,000 ops/s[[1]](#fn1).

**7 Memory Hierarchy & Interconnect**

1. **Unified Buffer (24 MiB, 256-byte ports):** Software-managed; no hardware caches—compiler schedules all transfers.
2. **Weight FIFO:** Tiles weights to hide 256-cycle array-load latency.
3. **DDR3 Weight DRAM (8 GiB, 34 GB/s):** Optimized for bandwidth, not latency; accessed sequentially.
4. **PCIe Gen3 ×16 (12.5 GB/s):** Only for input/output tensors and instruction stream; pressure is low due to large UB.

**8 Power Consumption vs CPU**

* **TPU v1:** 40 W measured, poor energy-proportionality (uses 88% of peak power even at 10% load)[[5]](#fn5).
* **Haswell CPU:** 145 W TDP, 56% proportionality at 10% load[[5]](#fn5).
* **Relative efficiency:** TPU v1 is **17–34 ×** better performance/Watt than Haswell when host power included; **41–83 ×** if host power excluded[[1]](#fn1).

**9 Multiply-Accumulate Blocks**

* **Count:** 65,536 MACs (8-bit) arranged in a 256 × 256 grid.
* **Accumulator design:** 4 MiB 32-bit registers permit 4096 concurrent output channels, enabling double-buffer overlap during weight shifts.

**10 Support for Diverse Neural Networks**

**10.1 Fully Connected (MLP)**

Matrix × vector operations map one-to-one onto the systolic array. Larger batches (≥ 128) amortize weight loads and achieve > 80% MXU utilization[[1]](#fn1).

**10.2 Convolutional Neural Networks**

im2col reshapes 3 × 3 or 1 × 1 convolutions into GEMM, allowing weight-stationary dataflow. CNN0 attains 86 TOPS—94% of peak[[1]](#fn1).

**10.3 Recurrent Neural Networks (LSTM)**

Temporal unrolling plus batched timesteps feed the MXU; sigmoid/tanh units implemented in Activation block. Bandwidth limits reduce utilization to ~10 TOPS, but still outperform CPU/GPU alternatives[[1]](#fn1).

**10.4 Unified Hardware Advantage**

Unlike the finite-state-machine design in the student’s earlier FPGA lab, TPU v1’s programmable instruction stream and large UB allow one chip to accelerate MLPs, CNNs, and RNNs without reconfiguration.

**11 Performance Bottlenecks & Design Lessons**

|  |  |  |
| --- | --- | --- |
| Limitation | Root Cause | Future Fix (v2+) |
| Memory-bound MLP/LSTM | DDR3 bandwidth ceiling | Upgrade to HBM2 (v2), double MXU per chip |
| Energy proportionality | No power-gating, always-on SRAM | Fine-grained clock-gating in v3/v4 |
| Inflexibility to sparse weights | Dense systolic array wastes ops | SparseCores added in v4 |

**12 FPGA Implementation Advice (Quartus/ModelSim)**

1. **Plan First:** Derive target layer sizes, select array dimension (e.g., 32 × 32) that fits FPGA BRAM.
2. **File Structure:**
   * /rtl/mac/mac\_unit.vhd – parameterized 8/16-bit MAC
   * /rtl/array/systolic\_array.vhd – grid generator, handles handshaking
   * /rtl/mem/memory\_controller.vhd – dual-port BRAM + AXI-lite for host
   * /rtl/top/tpu\_top.vhd – FSM for instruction scheduling
3. **Simulation:** Verify MAC in isolation, then 3 × 3 array, then full grid using test matrices.
4. **Timing Closure:** Enable Quartus PowerPlay; insert clock-enable on idle rows to improve leakage.
5. **Sequence:** Synthesize MAC → implement interconnect → integrate memory → add control FSM. Remind yourself to run ModelSim after each stage.

**13 Conclusion**

Google’s TPU v1 achieves its design goals by stripping away every general-purpose feature unnecessary for neural inference and dedicating die area to a vast, energy-efficient systolic array backed by on-chip SRAM. The result is a coprocessor that:

* Handles up to ~100 million parameters directly from on-board DRAM
* Sustains 92 TOPS while consuming only 40 W
* Delivers a 15–30 × latency-bounded speedup and 30–80 × energy efficiency improvement over contemporary CPU/GPU servers[[1]](#fn1)

These architectural ideas—weight-stationary systolic arrays, software-managed scratchpads, and minimalist control—remain central to all subsequent TPU generations and inform any FPGA designer aiming for a true *unified* neural-network accelerator.

⁂

1. <https://arxiv.org/pdf/1704.04760.pdf>

1. <https://web.stanford.edu/class/cs114/readings/tpu.pdf>

1. <http://sumityadav.com.np/tensor_processing_units/>

1. <https://www.cs.cmu.edu/afs/cs/academic/class/15740-f18/www/papers/isca17-jouppi-tpu.pdf>

1. <https://courses.grainger.illinois.edu/cs533/sp2025/notes/tpu_arch.pdf>