An Architectural Analysis of Google's Tensor Processing Units (v1–v3)

Justin Lanfermann
Technical University of Munich
Justin.Lanfermann@tum.de

***Abstract*—Datacenter demand for deep-learning inference and training has outpaced the efficiency gains of general-purpose CPUs and even high-end GPUs. Google's response is the Tensor Processing Unit (TPU) family, a series of domain-specific ASICs engineered around dense matrix multiplication. First-generation TPU v1 (2015), designed for inference, pairs a 256 $\times$ 256 8-bit systolic array with a software-managed 24 MiB SRAM, achieving 15–30 $\times$ higher throughput and up to 80 $\times$ better TOPS/Watt than the contemporary NVIDIA K80 while meeting sub-10 ms 99th-percentile latency targets. TPU v2 (2017) retargets the architecture for training by introducing bfloat16 arithmetic, dual 128 $\times$ 128 MXUs per chip, 16 GiB of HBM2, and a 496 Gbit/s chip-to-chip mesh that scales to 256-chip "Pods" delivering 11.8 PFLOPS. TPU v3 (2018) doubles the MXU count and HBM capacity, adds liquid cooling, and pushes a 1024-chip pod beyond 126 PFLOPS. Across the v1–v3 trajectory, TPUs provide a substantial improvement in cost-performance and energy efficiency relative to leading GPUs and FPGAs [1]–[3], establishing domain-specific co-design as an effective strategy for large-scale AI systems.**

I. Introduction

Artificial intelligence (AI) has witnessed dramatic growth in recent years [4], driven in large part by advances in deep learning and the widespread adoption of large-scale language models (LLMs) [5]. The exponential growth in computational demand from these applications has spurred a surge in industry investment and a corresponding expansion of data-center infrastructure [2]. Although LLMs currently dominate public attention, the trend toward greater AI workloads predates this recent wave, with capacity requirements growing exponentially since 2012 to support a variety of neural-network tasks in vision, language, and speech processing [2]. Traditionally, Central Processing Units (CPUs) have been the primary hardware for data centers. The parallel processing capabilities of Graphics Processing Units (GPUs) have proven effective for accelerating neural-network workloads, primarily due to their massively parallel, many-core design, which is well-suited to the inherent data parallelism in deep learning tasks. The CUDA programming model, in particular, has become the de-facto standard for harnessing this power, enabling significant performance gains for both training and inference [3], [6].

Nevertheless, even modern GPUs—and especially CPUs—exhibit limitations in performance-per-watt and in providing predictable, low "tail-latency" for user-facing services [1]. These constraints have highlighted the need for a new class of hardware: domain-specific architectures (DSAs). By focusing narrowly on the operations most critical to AI workloads—primarily dense matrix multiplications—DSAs can eliminate general-purpose overhead and achieve significantly higher efficiency [1], [2]. This trend has been further accelerated by the slowdown of Dennard scaling and Moore's Law, which has reduced the energy and performance gains traditionally realized by successive process-node shrinks [2].

Recognizing these challenges, Google launched a concerted effort in 2013 to develop a custom Application-Specific Integrated Circuit (ASIC) dedicated to neural-network workloads [1]. The outcome was the Tensor Processing Unit (TPU): a hardware accelerator tailored specifically to the dense linear algebra at the heart of most deep-learning models [1]. In 2015, Google introduced TPU v1, an inference-only device that delivered a substantial improvement in throughput and energy efficiency compared to contemporary CPUs and GPUs [1]. To address the growing importance of on-premise and cloud-based model training, Google released TPU v2 in 2017 and TPU v3 in 2018. These subsequent versions added support for bfloat16 arithmetic, High-Bandwidth Memory (HBM), and scaled-out "pod" configurations to accelerate large-scale training [2], [7], [8].

This paper examines the microarchitectural details of TPU v1 and analyzes how TPU v2 and v3 extend the original design to handle training workloads and achieve greater compute density. We compare TPUs with contemporary GPUs and FPGAs, highlighting the benefits and trade-offs of each architectural approach [3], and discuss how TPUs have been integrated into Google's data centers. Finally, we offer a brief outlook on emerging trends in AI accelerator design, such as optical interconnects [5].

II. Motivation for Tensor Processing Units

By 2013, Google's capacity-planning team projected that if users performed just *three minutes of voice recognition per day*, the associated neural-network inference load would necessitate a *two-fold* increase in datacenter count, which was then based almost entirely on x86 CPUs [1]. Directly scaling CPU clusters was deemed economically and environmentally challenging, as the cost of meeting this demand with conventional processors would have been unacceptably high [1].

GPUs appeared attractive due to their massive Single-Instruction, Multiple-Thread (SIMT) throughput; however, experiments with Nvidia's Kepler-generation K80 revealed a critical limitation. Under the stringent 99th-percentile latency targets of interactive services (e.g., $\leq$7 ms), the K80's effective throughput was "just a little faster than a Haswell CPU" because the latency constraints prevented the aggregation of requests into large, efficient batches [1]. The root cause was twofold: (1) sophisticated microarchitectural features like deep cache hierarchies and out-of-order execution in CPUs and GPUs optimize for average throughput but introduce latency variance unsuitable for 99th-percentile requirements, and (2) the general-purpose design meant that many hardware resources remained idle during NN inference, leading to wasted energy [1].

These observations motivated Google to pursue a *domain-specific architecture* (DSA) designed expressly for the dense matrix operations at the core of modern neural networks. By eliminating general-purpose logic (e.g., branch predictors, large caches) and tailoring the dataflow to fixed-shape matrix multiplies, such an ASIC was expected to deliver a 10× improvement in cost-performance relative to GPUs [1]. The resulting initiative—code-named the *Tensor Processing Unit* (TPU)—was green-lit with an aggressive fifteen-month schedule from project kickoff to datacenter deployment [1]. Subsequent sections detail how this objective shaped the TPU v1 microarchitecture and how later generations extended the design to support training workloads and large-scale ”pod” configurations.

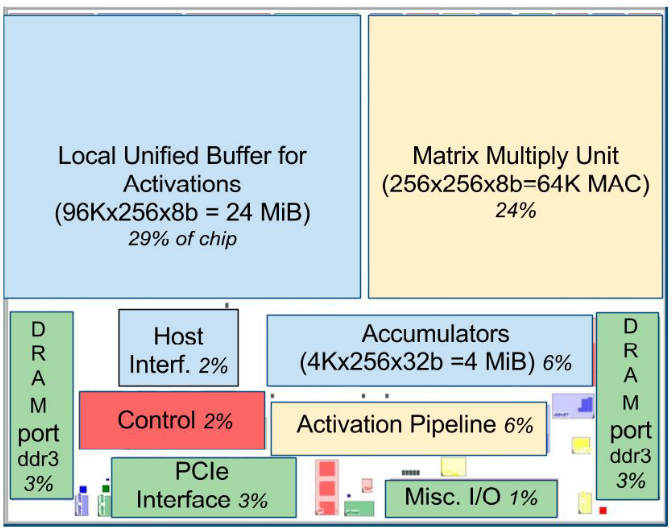
Local Unified Buffer for Activations (96Kx256x8b = 24 MiB) 29% of chip
Matrix Multiply Unit (256x256x8b=64K MAC) 24%
DRAM port ddr3 3%
Host Interf. 2%
Accumulators (4Kx256x32b =4 MiB) 6%
DRAM port ddr3 3%
Control 2%
Activation Pipeline 6%
PCIe Interface 3%
Misc. I/O 1%

Fig. 1. TPU Block Diagram. The main computation part is the yellow Matrix Multiply Unit in the upper right-hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB), and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB. Control logic is significantly smaller (2%) and less difficult to design compared to a general-purpose CPU or GPU [1, Fig. 1].

III. TPU v1 Microarchitecture and System Design

A. *Design Philosophy and Constraints*

The design of the first-generation Tensor Processing Unit (TPU v1) was significantly shaped by a condensed 15-month timeline from conception to deployment in Google's datacenters [1]. This accelerated schedule necessitated a pragmatic approach that prioritized architectural simplicity and rapid development to meet the specialized demands of neural network inference.

B. *System-Level Integration*

To minimize deployment risk and leverage existing datacenter infrastructure, TPU v1 was implemented as a coprocessor on a PCI Express (PCIe) Gen3 ×16 card [1]. This design enabled straightforward integration into existing server systems without requiring motherboard redesigns. Instructions and data are transferred from the host CPU to the TPU via this bus, with the host managing all transfers. The TPU itself does not fetch instructions from host memory, but rather executes instructions sent to it by the host [1]. While this integration strategy simplified the hardware design, the PCIe interface inherently limited communication bandwidth compared to on-chip pathways [1]. On the software side, the TPU software stack was built upon the TensorFlow framework, which compiled models into an API that could run on the TPU [1]. This choice enabled existing TensorFlow models to be retargeted to the TPU with minimal code modifications, further expediting deployment [1].

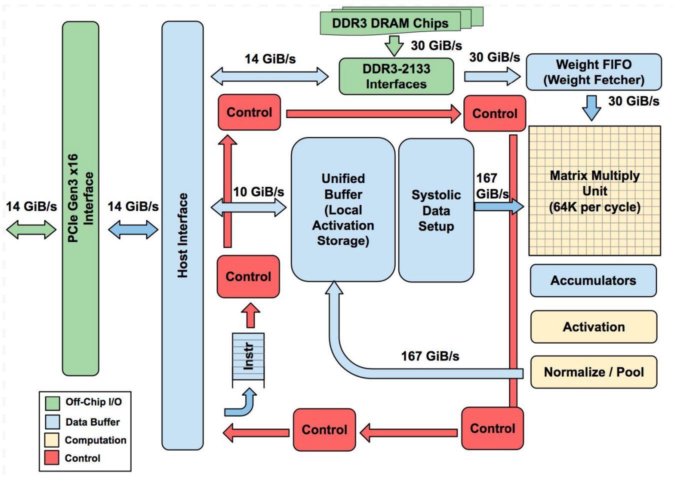

Fig. 2. Floor Plan of the TPU v1 die. The shading follows Figure 1. The light blue data buffers constitute 37% of the die, the light yellow compute blocks 30%, the medium green I/O 10%, and the dark red control logic just 2% [1, Fig. 2].

C. *Core Microarchitecture*

The core of TPU v1 comprises several specialized hardware blocks, as illustrated in Figure 1 [1]:

- **Matrix Multiply Unit (MMU):** The computational heart of the TPU, this unit features a 256×256 systolic array of 8-bit Multiply-Accumulate (MAC) units. A MAC operation multiplies two numbers and adds the product to an accumulator, forming the fundamental step for the large-scale matrix multiplications or convolutions the unit is engineered to perform using signed or unsigned 8-bit integer inputs [1].
- **Unified Buffer (UB):** A 24 MiB on-chip Static Random-Access Memory (SRAM), constituting approximately 29% of the TPU die area (Figure 2). It serves as a software-managed scratchpad for two primary functions [1]:
  - Staging input activations loaded from the host's DRAM via Direct Memory Access (DMA). During

MXU operations, 256-element activation vectors are streamed from the UB.
    - Holding intermediate partial sums generated by the MXU and final output activations before they are either written back to host memory or consumed by a subsequent layer.

  This large on-chip storage is crucial for sustaining the data rate required by the MXU, thereby mitigating stalls from off-chip memory accesses [1].
- **Accumulators:** A 4 MiB bank of 32-bit accumulators is provided, organized as 4096 units, each 256 elements wide. These units collect the 16-bit products from the MXU, allowing for higher precision during the accumulation phase of dot-product calculations before potential quantization [1].
- **Weight Memory and FIFO:** Neural network weights are stored off-chip in an 8 GiB DDR3 DRAM, termed Weight Memory. The `Read_Weights` instruction initiates a transfer of weight tiles from this memory into an on-chip weight First-In, First-Out (FIFO) buffer. This FIFO is four tiles deep and directly feeds 256 weight elements at a time to the MXU [1].
- **Activation Unit:** This dedicated hardware unit applies non-linear activation functions (e.g., ReLU, sigmoid) to values from the accumulators, a fundamental operation in neural networks [1].
- **Normalize/Pool Unit:** This unit performs normalization and pooling operations, which are common in many neural network architectures, particularly CNNs [1].

All data movement between host DRAM and the Unified Buffer is orchestrated by a programmable DMA controller, operating under the direction of specific TPU instructions like `Read_Host_Memory` and `Write_Host_Memory` [1].

D. Systolic Execution, Data Flow, and Operational Model

A defining characteristic of the TPU v1's MXU is its systolic array architecture [1]. Input activations from the UB and weights from the Weight FIFO are streamed into the MXU from orthogonal directions. These data elements propagate through the array of MAC units in synchronized, wave-like fronts. At each time step, a MAC unit receives data from its neighbors, performs a multiplication-and-addition, and passes the result along to the next MAC unit or into an accumulator. This design minimizes data movement and complex control logic, leading to high throughput and power efficiency [1]. Once the pipeline is filled, the 256×256 MXU can deliver up to 65,536 MAC operations per clock cycle. From a software perspective, this appears as 256 inputs being processed simultaneously, with each result instantly updating one location in each of the 256 accumulators [1].

The typical data flow for processing a neural network layer is as follows [1]:

1) Weights for the current layer are pre-loaded from off-chip Weight Memory into the on-chip Weight FIFO.
2) Input activations are staged in the Unified Buffer from host memory via DMA.
3) Activations and weights are streamed systematically into the MXU for computation.
4) The resulting products are aggregated in the Accumulators.
5) These accumulated values are passed through the Activation Unit and, if necessary, the Normalize/Pool unit.
6) The processed activations are written back to the Unified Buffer, ready to serve as inputs for a subsequent layer or to be transferred back to host memory.

TPU v1 operates under a model where the host CPU compiles and sends a sequence of CISC-style instructions, namely very long instruction word (VLIW), which the TPU executes autonomously. This allows the TPU to perform the entire inference computation for a model with minimal host intervention, maximizing accelerator utilization [1].

E. Key Architectural Decisions and Optimizations

Several strategic design choices were critical to TPU v1's performance and efficiency for inference workloads:

- **Absence of Caches:** A pivotal decision was the deliberate omission of hardware-managed caches. Instead, TPU v1 relies on the large, explicitly software-managed Unified Buffer [1]. This choice reduces die area and power consumption. More importantly, it eliminates the unpredictable latencies associated with cache misses, contributing to a deterministic execution model well-suited to the stringent 99th-percentile response-time requirements of online services [1]. It also obviates the overhead of cache coherence and replacement policies.
- **8-bit Integer Quantization:** Operations within TPU v1 are predominantly based on 8-bit integer arithmetic. This choice was motivated by several factors [1]:
    - **Model Compressibility:** Many deep-learning models can be quantized from 32-bit floating-point to 8-bit integers with negligible loss in predictive accuracy [1]. This integer-only approach, however, later proved to be a bottleneck for wider adoption, as some models required floating-point support [8].
    - **Hardware Efficiency:** An 8-bit MAC unit consumes significantly less energy and silicon area than a floating-point multiplier [1].
    - **Memory Footprint:** Storing weights as 8-bit values reduces their memory footprint by a factor of four compared to 32-bit floats [1].
    - **Determinism:** Integer arithmetic provides deterministic latency and outcomes, supporting the TPU's predictable execution model [1].
- **Process Node and Clock Speed:** TPU v1 was fabricated using a 28 nm process technology and operates at a core clock frequency of 700 MHz, reflecting the standard technology available at the time of its design [1].

F. Performance and Power Metrics

Google's internal benchmarks confirmed that on a per-die basis, TPU v1 provided a weighted mean throughput $29.2\times$ higher than a Haswell CPU and $15.3\times$ higher than an NVIDIA

K80 GPU on production workloads. The efficiency gains were even more pronounced, with the TPU achieving an 83× and 29× improvement in TOPS/Watt over the CPU and GPU, respectively [1].

This outperformance was not uniform across all applications, as certain workloads were limited by memory bandwidth [1]. A critical metric for its target applications was the 99th-percentile response time. Under a 7 ms latency constraint for a production service, the Haswell CPU and K80 GPU sustained only 42% and 37% of their respective peak throughputs for that application. The TPU v1, by contrast, maintained about 80% utilization under the same deadline, demonstrating its superior performance under strict tail-latency requirements [1]. In general, TPUs can handle larger batch sizes while maintaining lower 99th-percentile response times compared to CPUs and GPUs, making them well-suited for latency-sensitive applications [1].

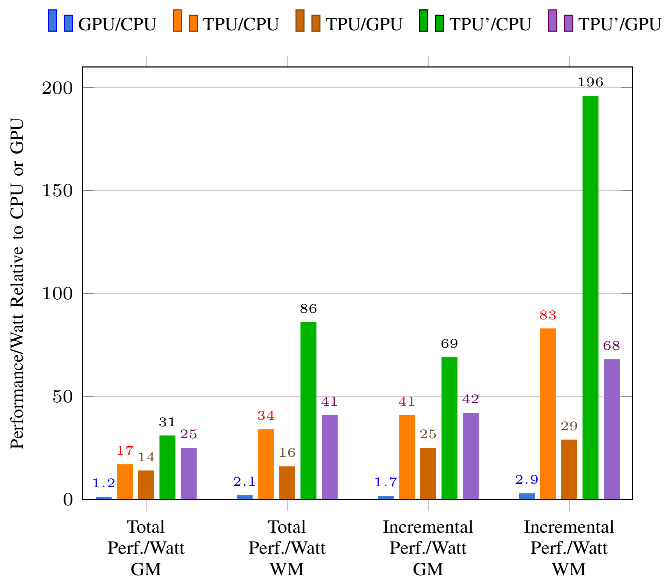

Fig. 3. Performance/Watt comparison, recreated from data in Jouppi et al. [1, Fig. 9]. The chart shows the relative performance/Watt (TDP) of a GPU server (blue) and TPU server (orange) to a CPU server, and the TPU server to a GPU server (brown). TPU′ is a hypothetical improved TPU; the lavender bar shows its ratio relative to the GPU. 'Total' includes host-server power, whereas *incremental* subtracts it. GM and WM are geometric and weighted means.

G. *Limitations and Lessons Learned*

Despite its strengths, TPU v1 had several key limitations. The principal constraint was its memory bandwidth; reliance on off-chip DDR3 memory significantly curtailed performance. Roofline models developed by Google confirmed that many applications were memory-bound [1]. A hypothetical "TPU′" design incorporating GDDR5 memory (providing over 5× the bandwidth) was modeled to quantify this bottleneck. This analysis indicated that such an enhancement would have tripled its achieved throughput and raised its TOPS/Watt to nearly 70× that of the K80 GPU and 200× that of the Haswell CPU [1].

Another notable aspect was its poor energy proportionality. At 10% utilization, TPU v1 consumed 88% of its peak power, a consequence of an accelerated development schedule that precluded the inclusion of typical power-saving features [1].

Finally, a significant limitation was that TPU v1 was engineered exclusively for integer-based calculations [1]. This design choice precluded its use for training tasks, which demand higher-precision floating-point arithmetic [2]. Even within inference, the integer-only approach proved to be a significant adoption barrier. Although many models could be quantized with minimal accuracy loss, the requirement to do so proved problematic for some datacenter applications and delayed adoption, motivating the inclusion of floating-point support in subsequent TPU generations [2], [8].

IV. Evolution to Training: TPU v2 and TPU v3

A. *TPU v2 — Enabling Large-Scale Training*

The second-generation Tensor Processing Unit (TPU v2), first deployed in 2017, represents a significant architectural evolution from its predecessor, designed to address the demanding computational requirements of training deep neural networks—a task for which TPU v1 was unequipped. This transition required fundamental changes to the arithmetic logic, memory subsystem, and system-level interconnect [2], [7].

1) Brain Float 16 (bf16) for Training: A primary objective for TPU v2 was to incorporate floating-point arithmetic, which is essential for the backpropagation algorithm. While standard 32-bit floating-point (FP32) provides sufficient precision, it is computationally expensive. Conversely, the 16-bit FP16 format presented challenges due to its limited 5-bit exponent range, which could lead to numerical instability or require algorithmic adjustments like loss scaling [2].

To resolve this, Google introduced the Brain Float 16 (bf16) format, which prioritizes dynamic range over precision. It sacrifices mantissa bits but retains the 8-bit exponent of FP32, thereby preserving the ability to represent very large and very small numbers and avoiding the need for algorithmic adjustments like loss scaling [2], [9].

2) Architectural and Memory Enhancements: The memory subsystem of TPU v2 was fundamentally redesigned to eliminate the bandwidth bottleneck that had constrained TPU v1. This shift to HBM was a direct response to the primary limitation of TPU v1, where off-chip DDR3 memory proved to be a significant bottleneck for a majority of production workloads [1]. The introduction of 16 GiB of off-chip High-Bandwidth Memory (HBM) per chip increased available memory bandwidth to 700 GB/s, a more than 20-fold improvement over its predecessor, which effectively removed this memory wall and enabled TPU v2 to sustain its compute units more effectively [2].

Architecturally, TPU v2 evolved from a single-core processor to a dual-core chip, with each core termed a "TensorCore." Each TensorCore integrates a 128×128 MXU, a large Vector Processing Unit (VPU) with 16 MiB of on-chip vector memory (Vmem), and a dedicated Transpose/Reduction/Permute unit [2]. This results in a chip-wide total of 32 MiB of on-chip Vmem, a significant increase that supports the larger intermediate values required during training.

In addition to these enhancements for dense computations, TPU v2 also incorporated specialized hardware for sparse operations, a feature later revealed as "SparseCores" in the context of the TPU v4 release [5]. These cores are designed to accelerate the performance of models that rely heavily on sparse operations, which are common in recommendation systems and other large-scale AI applications [5].

3) System-Level Scaling and Interconnect: To facilitate large-scale distributed training, TPU v2 was designed with a high-speed on-chip interconnect. Each chip features four custom Inter-Core Interconnect (ICI) links, each providing 496 Gbit/s of bandwidth [2]. These links enable chips to be assembled into a 2D torus topology.

A full TPU v2 "Pod" can scale to 256 chips arranged in a 16×16 torus, delivering a peak performance of 11.8 PetaFLOPS [2]. In this configuration, a single host CPU manages four TPU v2 chips [2]. The increased performance resulted in a higher Thermal Design Power (TDP), which rose from 75 W in TPU v1 to 280 W in TPU v2, reflecting its positioning as a high-performance training accelerator [2]. Like its predecessor, TPU v2 connects to its host via a high-speed PCIe interface, though the exact specification is not detailed in the available literature [1], [2].

4) The Role of the XLA Compiler: The hardware advancements of TPU v2 are tightly integrated with the XLA (Accelerated Linear Algebra) compiler. XLA performs whole-program analysis and optimization, compiling the entire training graph ahead of time. It manages memory transfers, schedules hardware parallelism using a VLIW execution model, and maps computations across an entire supercomputer pod. The compiler targets a high-level vocabulary of 96 optimized operations from which it constructs the final program [2]. The impact of this co-design is significant: through software and compiler optimizations alone, the performance of TPUs on the MLPerf training benchmark suite increased by a median factor of 2.1 over a six-month period [2].

B. TPU v3 — Doubling Compute and Memory

Released in 2018, TPU v3 was a strategic, incremental evolution of the TPU v2 architecture, designed to meet the continued growth in neural network model complexity [2]. Rather than a complete redesign, TPU v3 functioned as a mid-cycle update, leveraging the same underlying process technology to deliver substantial gains in compute density and system scale [2].

The most significant hardware enhancement in TPU v3 was doubling the number of MXUs per TensorCore from one to two, thereby doubling the MXU count per chip from two to four [2]. This was complemented by an increase in clock frequency to 940 MHz, a doubling of HBM2 capacity to 32 GiB, an increase in HBM2 bandwidth to 900 GB/s, and a 32% increase in ICI bandwidth to 656 Gbit/s per link [2]. Collectively, these improvements resulted in a $2.7\times$ increase in peak theoretical performance over TPU v2. On the MLPerf 0.6 benchmark suite, a single-chip TPU v3 delivered a geometric mean speedup of $1.8\times$ over a TPU v2, and a mean of $1.78\times$ on a set of six internal production applications [2]. The increased compute density raised the chip's TDP to 450 W, necessitating a shift to liquid cooling (see Table I) [2].

Despite doubling the MXUs, the chip's die area grew by only 6% over TPU v2. This efficiency was achieved through an optimized floor plan developed from the experience gained during the v2 design, which enabled a more compact layout of the major architectural blocks [2].

At the system level, TPU v3 expanded the maximum pod size fourfold to 1024 chips [2]. A fully configured TPU v3 pod delivers over 126 PetaFLOPS and demonstrates high scaling efficiency, achieving 96% to 99% of perfect linear scaling on production workloads at the full 1024-chip scale [2]. Single-chip performance was roughly equivalent to the contemporary NVIDIA Volta GPU on MLPerf benchmarks [2].

While optimized for training, TPU v3 also advanced inference capabilities. For the LSTM0 benchmark, it achieved a response time of 45 ms, comparable to the 44 ms of TPUv2 and representing a greater than $2.5\times$ reduction from the 122 ms latency of the inference-only TPUv1 [2]. Like TPU v2, it performs all operations using bf16 and does not support 8-bit integers. This reinforces the "What You Train Is What You Serve" (WYTIWYS) principle, allowing models to be deployed for inference using the same numerical representation and hardware as they were trained on, thereby eliminating the need for a separate quantization step and its associated risk of accuracy degradation [2], [8].

C. Future Outlook and Modern TPUs (v4–v7)

The evolution of Tensor Processing Units did not conclude with v3. Driven by the relentless pace of AI innovation, subsequent TPUs have introduced novel architectural concepts to address emerging challenges in model scale, efficiency, and flexibility. While detailed peer-reviewed literature is available for TPU v4, information on subsequent generations is primarily drawn from official technical announcements. This summary is based on those sources, acknowledging they have not undergone the same level of independent scientific scrutiny as the papers cited for earlier TPU versions.

1) TPU v4 and v4i — Reconfiguration and Specialization: Released in 2020, TPU v4 marked a significant evolution in ML supercomputer design, achieving a $2.1\times$ performance uplift and a $2.7\times$ improvement in performance-per-watt over its TPU v3 predecessor [5]. Each TPU v4 chip, containing two TensorCores, delivered 275 TFLOPS of peak bf16 performance and was equipped with 32 GiB of HBM providing 1.2 TB/s of bandwidth [5], [10]. This generation's key advances included the integration of Optical Circuit Switches (OCS) and the announcement of on-chip SparseCores. The OCS technology enabled the dynamic reconfiguration of the system's 3D torus interconnect, improving availability and performance across supercomputers scaling to 4096 chips. Specific slice configurations, termed *twisted tori*, could further enhance bisection bandwidth significantly for workloads with global communication patterns [5], [10]. The SparseCores, dedicated dataflow processors for embedding-heavy models, accelerated workloads like DLRMs by 5–$7\times$ while consuming only 5% of the die area and power [5].

Concurrently, insights from previous generations informed the development of the TPU v4i, a specialized, air-cooled

TABLE I
KEY PROCESSOR FEATURES OF TPU V1-V3 AND NVIDIA VOLTA. DATA IS COMPILED FROM JOUPPI ET AL. [2]. JOUPPI AND COLLEAGUES NOTE THAT WHILE SPECIFIC PROCESS NODE DETAILS FOR THE TPUS ARE PROPRIETARY, THE TPUV2'S DIE SIZE IS LESS THAN THREE-QUARTERS THAT OF THE VOLTA, AND THE TPUV3 DIE IS APPROXIMATELY 6% LARGER THAN THE TPUV2. FOR REFERENCE, TDP STANDS FOR THERMAL DESIGN POWER, AND THE VOLTA ARCHITECTURE CONTAINS 80 SYMMETRIC MULTIPROCESSORS.

Feature	**TPUv1**	**TPUv2**	**TPUv3**	**Volta**
Peak TeraFLOPS/Chip	92 (8b int)	46 (16b) 3 (32b)	123 (16b) 4 (32b)	125 (16b) 16 (32b)
Network links $\times$ Gbits/s/Chip	—	4×496	4×656	6×200
Max chips/supercomputer	—	256	1024	Varies
Peak PetaFLOPS/supercomputer	—	11.8	126	Varies
Bisection Terabits/supercomputer	—	15.9	42.0	Varies
Clock Rate (MHz)	700	700	940	1530
TDP (Watts)/Chip	75	280	450	450
TDP (Kwatts)/supercomputer	—	124	594	Varies
Die Size (mm^2)	< 331	< 611	< 648	815
Chip Technology	28 nm	$>$ 12 nm	$>$ 12 nm	12 nm
Memory size (on-/off-chip)	28MiB/8GiB	32MiB/16GiB	32MiB/32GiB	36MiB/32GiB
Memory GB/s/Chip	34	700	900	900
MXUs/Core, MXU Size	1 256x256	1 128x128	2 128x128	8 4x4
Cores/Chip	1	2	2	80
Chips/CPU Host	4	4	8	8 or 16

accelerator for inference [8]. The v4i focused on deployment efficiency, delivering 2.3× the performance-per-TDP of TPU v3 while maintaining a similar absolute performance level [8]. This bifurcation of the v4 family into a large-scale, liquid-cooled training platform (TPU v4) and a highly efficient, air-cooled inference platform (TPU v4i) illustrates the increasing need for specialized hardware tailored to the distinct phases of the machine learning lifecycle.

2) Generations v5 to v7: Accelerating Compute and Efficiency: Subsequent generations—TPU v5, v6 (Trillium), and v7 (Ironwood)—have continued a trajectory of rapid, iterative enhancement focused on increasing compute density, memory capacity, and power efficiency across distinct performance- and efficiency-focused product lines.

The TPU v5 series formalized this specialization by introducing two distinct variants. The first, TPU v5e, was designed for cost-efficient scaling, providing 197 TFLOPS of peak bf16 compute with 16 GB of HBM at a bandwidth of 0.82 TB/s per chip [11]. The second variant, the performance-focused TPU v5p, offered significant hardware advancements over TPU v4. Each v5p chip provides 459 TFLOPS of bf16 compute (a 1.7× increase over v4), 95 GB of HBM capacity (a 3× increase), and 2.8 TB/s of memory bandwidth (a 2.3× increase) [5], [12]. These chips are deployed in large-scale pods of up to 8,960 units, interconnected with a per-chip bisectional bandwidth of 4,800 Gb/s [12], [13].

The sixth-generation accelerator, TPU v6e (Trillium), succeeded the efficiency-focused v5e, delivering substantial per-chip gains. Compared to its direct predecessor, the v5e, peak bf16 compute on v6e increased 4.7× to 918 TFLOPS. Correspondingly, HBM capacity was doubled from 16 GB to 32 GB and bandwidth was doubled from 0.82 TB/s to 1.64 TB/s, while inter-chip interconnect bandwidth also more than doubled [11], [14]. These architectural improvements were coupled with a reported 67% increase in energy efficiency, highlighting a continued focus on sustainable performance scaling [15].

Announced for 2025, the seventh-generation Ironwood architecture represents a substantial leap, continuing the high-performance lineage from Trillium. Compared to its v6e predecessor, Ironwood is slated to deliver a 2× improvement in performance-per-watt. The design features a dramatic expansion of on-chip memory, increasing HBM capacity by 6× to 192 GB and HBM bandwidth by 4.5× to 7.4 TB/s per chip. These memory and interconnect improvements support a projected five-fold increase in peak computational throughput to over 4.6 PFLOPS per chip, targeting the immense demands of next-generation models [16]. This trajectory highlights a design philosophy that balances raw compute with commensurate scaling in memory access and power efficiency to avoid systemic bottlenecks.

D. Concluding Remarks on the TPU Family and Industry Impact

The TPU project has not only reshaped Google's internal AI infrastructure but has also influenced a broader industry trend toward domain-specific accelerators. Following Google's lead, other major cloud providers and silicon vendors have introduced their own specialized hardware, including AWS Inferentia, Microsoft Brainwave, Intel Gaudi, and others. This has spurred a competitive and innovative ecosystem where hardware vendors like NVIDIA, AMD, and Intel have responded by integrating dedicated tensor cores into their GPUs and optimizing their software stacks for machine learning. The success of the TPU has validated the DSA approach, demonstrating that for workloads with a clear and dominant computational pattern, specialized hardware can deliver substantial gains in performance and efficiency.

TABLE II
COMPARATIVE SUMMARY OF AI ACCELERATOR ARCHITECTURES

Attribute	GPU (e.g., NVIDIA Volta)	FPGA (e.g., MS Brainwave)	TPU (e.g., Google v1-v3)
Hardware Paradigm	General-Purpose Parallel Processor	Reconfigurable Logic (Soft Processor)	Domain-Specific ASIC
Primary Design Goal	**High-Throughput General-Purpose Parallel Computing**	**Ultra-Low Latency Inference (Batch-of-1)**	**Maximum Throughput & Efficiency at Scale (TCO)**
Primary Strength	Programmability & Generality	Flexibility & Latency (Batch-of-1)	Efficiency & Scale (Perf/Watt)
Compute Primitive	SIMT Cores (Thread Parallelism)	Matrix-Vector (Single-Request Parallelism)	Systolic Array (Matrix-Matrix Parallelism)
Numerical Precision	Standard (FP32, FP16, INT8)	Custom (e.g., Block Floating-Point)	Custom (bfloat16) & Standard (INT8)
Memory Architecture	Hardware Caches + High-Bandwidth Memory	Software-Managed On-Chip SRAM	Software-Managed SRAM + HBM
Scalability Model	Host-Managed (NVLink + InfiniBand)	Network-Attached Microservices	Custom Torus Interconnect (ICI/OCS)
Best Suited For	Wide variety of ML tasks, research	Latency-critical inference (Batch-of-1)	Large-scale training and high-volume inference

V. COMPARATIVE ANALYSIS: GPU, FPGA, AND TPU ARCHITECTURES

The landscape of AI acceleration is shaped by a fundamental trade-off between programmability and domain-specific efficiency. This spectrum is anchored by three distinct architectural paradigms, each representing a unique philosophy for solving high-performance computing challenges.

On one end, the Graphics Processing Unit (GPU), exemplified by NVIDIA's Volta architecture, represents the pinnacle of programmable parallel processing. Its thousands of SIMT cores provide immense throughput for a wide variety of ML tasks, making it a powerful and versatile tool for both research and production [6].

At the opposite extreme lies the Application-Specific Integrated Circuit (ASIC), with Google's TPU serving as the archetypal example. By foregoing general-purpose features, the TPU dedicates its silicon to a single task—dense matrix multiplication—achieving unparalleled performance-per-watt at massive scale [1], [2].

Between these two poles exists a compelling third approach: the reconfigurable Field-Programmable Gate Array (FPGA). As demonstrated by Microsoft's Project Brainwave, FPGAs offer a unique middle ground. They allow for the creation of "soft processors" with custom dataflows and numerical formats, providing hardware-level optimization without the finality of an ASIC design. This makes them exceptionally well-suited for ultra-low-latency workloads where single-request performance is paramount [3].

This section analyzes how these divergent architectural philosophies translate into tangible performance differences. By comparing GPUs, FPGAs, and TPUs, we illuminate the critical trade-offs between latency, throughput, flexibility, and scalability that define modern AI accelerator design.

A. Architectural Philosophy and the Latency-Throughput Trade-off

A central trade-off in accelerator design for interactive services is between minimizing single-request latency and maximizing aggregate throughput. TPUs and FPGAs represent two distinct and highly optimized solutions to this challenge.

1) FPGA: Optimization for Real-Time, Batch-of-1 Inference: The primary design goal of Microsoft's Brainwave NPU—and a key strength of FPGAs—is to excel at "real-time AI" by optimizing for inference on individual requests (a batch size of one) with minimal delay [3]. This philosophy is driven by the strict Service Level Agreements (SLAs) of interactive applications where batching requests is often not feasible. The architecture achieves this through several key innovations:

- **Single-Request Parallelism:** Instead of aggregating multiple requests, the Brainwave microarchitecture is engineered to extract massive fine-grained parallelism from a single inference pass. Its Hierarchical Decode and Dispatch (HDD) mechanism can expand one high-level instruction into over 7 million primitive operations, saturating spatially distributed compute units without a batch [3].
- **Matrix-Vector Engine:** The architecture centers on a matrix-vector compute primitive (Level-2 BLAS). This is more naturally suited to the dataflow of unbatched, memory-intensive models like RNNs and LSTMs, which dominate latency-sensitive language tasks, compared to matrix-matrix engines that require larger tensors for high utilization [3].

The performance results underscore this specialization. On a Stratix 10 FPGA, the Brainwave NPU sustains up to 35.9 TFLOPS on a large GRU model at a batch size of one, with latency under 4 ms [3]. In the official MLPerf Inference v0.7 results, a dual-accelerator FPGA system (Neuchips RecAccel) focused on the DLRM benchmark—a memory-intensive recommendation model—achieved a server throughput of 1,249 queries per second [17]. This result, while modest in absolute terms, is achieved under latency-constrained conditions that favor the FPGA's fine-grained dataflow architecture.

2) TPU and GPU: Optimization for Throughput and Efficiency at Scale: In contrast, TPUs and high-end GPUs are designed for maximum performance and cost-effectiveness

(TOPS/Watt) in a massive datacenter environment where requests can often be aggregated into batches [1]. While still sensitive to tail latency, their designs reflect a balance skewed heavily toward aggregate throughput.

- **Large-Scale Data Parallelism:** The heart of a TPU is a large systolic array optimized for matrix-matrix multiplication (Level-3 BLAS) [2]. Similarly, GPUs leverage thousands of cores to process large batches in parallel. These architectures achieve peak efficiency with larger tensors that keep the massive compute arrays continuously supplied with data.
- **Deterministic vs. Probabilistic Execution:** By omitting hardware-managed caches and other sources of latency variance, the TPU provides a deterministic execution model. This predictability allows it to operate closer to its peak throughput even under strict 99th-percentile latency constraints. In contrast to the time-varying optimizations of CPUs and GPUs (e.g., caches, out-of-order execution), the TPU's deterministic pipeline proved a better match for the strict latency requirements of interactive services [1].

This architectural divergence leads to starkly different benchmark results. In the same MLPerf v0.7 DLRM server benchmark, a system with eight NVIDIA T4 GPUs achieved a throughput of 250,177 queries per second—over 200× higher than the FPGA system, albeit with more accelerators [17]. A high-end 8-accelerator NVIDIA A100 system reached 2,102,386 queries per second on the same task [17]. While Google's TPU inference results[1] are not directly comparable on this specific benchmark version, their architectural similarity to GPUs in leveraging batched computation suggests a performance profile geared overwhelmingly toward throughput. The data clearly shows that for workloads that can be batched, ASICs and GPUs offer orders-of-magnitude greater throughput, a key metric for datacenter TCO.

B. Flexibility, Precision, and Future-Proofing

The choice between a fixed ASIC and a reconfigurable FPGA has profound implications for adapting to the rapid evolution of AI models.

1) FPGA: Flexibility through Reconfigurability: The Brainwave NPU's greatest strategic advantage is its implementation on an FPGA. This allows for "synthesis specialization," where the hardware datapath can be recompiled and optimized for new models or novel numerical formats. For example, Microsoft developed a custom Block Floating-Point (BFP) format with a shared exponent and variable-width mantissas, which proved highly efficient for their target models [3]. This hardware adaptability provides a powerful hedge against architectural obsolescence as DNN research uncovers new layer types or computational patterns that may not map efficiently to a fixed ASIC.

[1]Google's Cloud TPU inference results were submitted in the MLPerf v0.5 closed division; metrics are publicly summarized but full logs are restricted to MLCommons members.

2) TPU: Co-Design and Software-Driven Evolution: While a TPU's hardware is fixed at the time of fabrication, its performance is not static. Google's strategy relies on a deep hardware-software co-design with the XLA compiler. This tight integration allows for continuous performance improvements long after hardware deployment. For example, between MLPerf v0.5 and v0.6, the performance of TPUv3 systems increased by a median of 2.1 × across the benchmark suite due to compiler and software stack optimizations alone [2]. This demonstrates that an ASIC platform, when paired with a powerful, co-designed compiler, can also achieve a significant degree of "future-proofing" by better exploiting existing hardware for new workloads. The introduction of bfloat16 is a prime example of this co-design, providing the dynamic range needed for training without the full cost of FP32 [2].

C. System-Level Integration and Scalability

Finally, the architectures differ significantly in their integration into the datacenter fabric. The Brainwave model proposes FPGAs as disaggregated, network-attached "hardware microservices," offering great flexibility for resource management [3]. In contrast, the TPU Pod architecture is a purpose-built supercomputer, where up to 1024 TPU v3 chips are linked by a custom, high-bandwidth 2D torus interconnect [2]. While less flexible, this tightly-coupled design is essential for massive-scale synchronous training, enabling near-perfect linear performance scaling on a single, large workload—a critical capability for training state-of-the-art models that the more loosely-coupled FPGA model is not designed to address.

In summary, the choice between FPGA and TPU is not about which is "better," but which is better suited to a specific operational goal. The reconfigurable FPGA offers unparalleled latency for single requests and adaptability, making it ideal for specialized real-time inference. The specialized TPU ASIC, through its immense compute density, a deterministic pipeline, and a co-designed software stack, provides superior throughput and efficiency at scale, making it the dominant architecture for large-scale training and high-volume batched inference.

VI. Conclusion and Outlook

The development of Google's Tensor Processing Unit family provides a compelling case study in the application of domain-specific architecture to overcome the scaling limitations of general-purpose hardware. Faced with an exponential rise in demand for AI compute that threatened to make datacenter expansion economically and environmentally unfeasible, Google's TPU project demonstrated that a tightly co-designed, specialized approach could yield substantial improvements in performance and efficiency [1], [2].

TPU v1 was a successful proof-of-concept for the immediate challenge of neural network inference, delivering 15–30× higher throughput and up to 80× greater power efficiency than contemporary CPUs and GPUs. Its success was rooted in a disciplined architectural focus: a large systolic array for dense matrix multiplication, software-managed on-chip memory to ensure deterministic low-latency execution, and the strategic omission of general-purpose features like hardware caches

[1], [2]. However, its integer-only arithmetic and memory-bandwidth constraints highlighted the need for a more versatile architecture to tackle the demanding task of model training.

The evolution to TPU v2 and v3 represented a significant expansion of this architectural strategy. By introducing the custom bfloat16 numerical format, integrating High-Bandwidth Memory to alleviate the v1 memory bottleneck, and developing a custom high-speed chip-to-chip interconnect, Google evolved the TPU from a single-chip accelerator into a scalable supercomputing platform [2]. The TPU v3 Pod, scaling to 1024 chips with near-perfect linear efficiency on production workloads, established that a domain-specific architecture could outperform traditional supercomputers in performance-per-watt for large-scale AI training [2].

Crucially, the success of the TPU cannot be attributed to hardware alone. The close, concurrent co-design with the TensorFlow framework and the XLA compiler was fundamental. This synergy enabled the hardware's full potential to be harnessed and provided a pathway for significant performance gains through software optimization long after the silicon was fabricated, a key advantage in the fast-moving field of AI [2]. This approach stands in contrast to the hardware-first reconfigurability of FPGAs, which excel at ultra-low-latency, single-request inference but are not designed for the massive-scale synchronous training at which the TPU Pod architecture excels [3].

The trajectory from v1 to v3 laid the foundation for continued innovation. Subsequent generations like TPU v4 introduced reconfigurable optical interconnects and a split training/inference-specialized design, underscoring a continued commitment to system-level optimization [5], [8]. Ultimately, the TPU v1–v3 family did more than solve an internal scaling problem for Google; it served as a powerful proof-of-concept for the industry, influencing the trend toward specialized hardware and establishing domain-specific co-design as a key strategy for the future of artificial intelligence.

References

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," Apr. 2017.

[2] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, "A domain-specific supercomputer for training deep neural networks," *Communications of the ACM*, vol. 63, no. 7, pp. 67–78, Jun. 2020.

[3] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A Configurable Cloud-Scale DNN Processor for Real-Time AI," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2018, pp. 1–14.

[4] N. Maslej, "Artificial Intelligence Index Report 2025," *Artificial Intelligence*, 2025.

[5] N. P. Jouppi, G. Kurian, S. Li, P. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles, C. Young, X. Zhou, Z. Zhou, and D. Patterson, "TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings," Apr. 2023.

[6] J. Nickolls and W. J. Dally, "The GPU Computing Era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar. 2010.

[7] T. Norrie, N. Patil, D. H. Yoon, G. Kurian, S. Li, J. Laudon, C. Young, N. Jouppi, and D. Patterson, "The Design Process for Google's Training Chips: TPUv2 and TPUv3," *IEEE Micro*, vol. 41, no. 2, pp. 56–63, Mar. 2021.

[8] N. P. Jouppi, D. Hyun Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. Valencia, Spain: IEEE, Jun. 2021, pp. 1–14.

[9] "BFloat16: The secret to high performance on Cloud TPUs," https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus.

[10] "TPU v4 | Google Cloud," https://cloud.google.com/tpu/docs/v4.

[11] "TPU v5e | Google Cloud," https://cloud.google.com/tpu/docs/v5e.

[12] "TPU v5p | Google Cloud," https://cloud.google.com/tpu/docs/v5p.

[13] "Introducing Cloud TPU v5p and AI Hypercomputer," https://cloud.google.com/blog/products/ai-machine-learning/introducing-cloud-tpu-v5p-and-ai-hypercomputer.

[14] "TPU v6e," https://cloud.google.com/tpu/docs/v6e.

[15] "Trillium sixth-generation TPU is in preview," https://cloud.google.com/blog/products/compute/trillium-sixth-generation-tpu-is-in-preview.

[16] "Ironwood: The first Google TPU for the age of inference," https://blog.google/products/google-cloud/ironwood-tpu-age-of-inference/, Apr. 2025.

[17] MLCommons, "MLPerf Inference v0.7 results," Oct. 2020.