White Paper

Artificial Intelligence
FPGA

Real Performance of FPGAs Tops GPUs in the Race to Accelerate AI

Authors

Eriko Nurvitadhi
Sr. Research Scientist
Programmable Solutions Group
Intel Corporation

Rohit D'Souza
AI Product Marketing Manager
Programmable Solutions Group
Intel Corporation

Martin Won
Senior Member of Technical Staff
Programmable Solutions Group
Intel Corporation

Table of Contents

Introduction

As artificial intelligence (AI) models increase in size and complexity at a relentless clip of approximately 10X per year [2], providers of AI solutions face great pressure to reduce time to market, improve performance, and quickly adapt to a changing landscape. Such increasing model complexity has led to AI-optimized hardware. For example, in recent years, graphics processing units (GPUs) have integrated AI-optimized arithmetic units to increase their AI compute throughput. Nevertheless, as AI algorithms and workloads evolve, they can exhibit properties that make it challenging to fully utilize the available AI compute throughput, unless the hardware offers extensive flexibility to accommodate such algorithmic changes. Recent papers (e.g., [1], [3]) have shown that for many AI workloads, it can be challenging to achieve the full compute capacity reported by GPU vendors. Even for highly parallel computation such as general matrix multiplication (GEMM), GPUs can only achieve high utilization at certain large matrix sizes [4]. Therefore, even though GPUs offer high AI compute throughput in theory (often called 'peak throughput'), when running AI applications, the real performance achieved could be much lower.

FPGAs offer a different approach to AI-optimized hardware. Unlike GPUs, FPGAs offer unique fine-grained spatial reconfigurability. This means that FPGA resources can be configured to perform the exact mathematical functions in precisely the correct order to implement the desired operation. The output of each function can be routed directly to the input of the function that needs it. This approach allows greater flexibility to accommodate specific AI algorithms and application characteristics that enable improved utilization of available FPGA compute capabilities. Furthermore, while FPGAs require hardware expertise to program (through a hardware description language), specialized soft processors (a.k.a. overlays) [1], [3] allow FPGA programming in a similar fashion as processors. The FPGA programming is done purely via software toolchains and abstracts away any FPGA-specific hardware complexity.

The discussion in this white paper is based on results that have been published in the 2020 IEEE International Conference on Field Programmable Technology (FPT) [1]. It presents the first performance evaluation of the Intel® Stratix® 10 NX FPGA in comparison to the NVIDIA T4 and V100 GPUs. This performance evaluation is done over a suite of real-time inference workloads. For the FPGA, the workloads are deployed using an implementation of a soft AI processor overlay called the Neural Processing Unit (NPU) with a software toolchain that allows you to program the FPGA without invoking any FPGA-specific Electronic Design Automation (EDA) tools. Results show that for these workloads, the Intel Stratix 10 NX FPGA achieves far better utilization and performance than the NVIDIA T4 and V100 GPUs.

Background on FPGA and GPU Architectures

In 2020, Intel announced its first AI-optimized FPGAs, the Intel Stratix 10 NX devices. The Intel Stratix 10 NX FPGA includes AI Tensor Blocks, which enable the FPGA to deliver up to 143 INT8 and 286 INT4 peak AI compute TOPS or 143 Block Floating Point 16 (BFP16) and 286 Block Floating Point 12 (BFP12) TFLOPS [2]. Recent publications (e.g., [6] [7]) have shown that Block Floating Point precision provides greater accuracy and lower cost for many AI workloads. Similarly, NVIDIA GPUs offer Tensor Cores. But GPU Tensor Cores and FPGA AI Tensor Blocks are very different from an architecture perspective as illustrated in Figures 1 and 2. A few key differences are described in this section, since understanding the basic architectural variances can help provide valuable context and background for the results discussed later in this white paper.

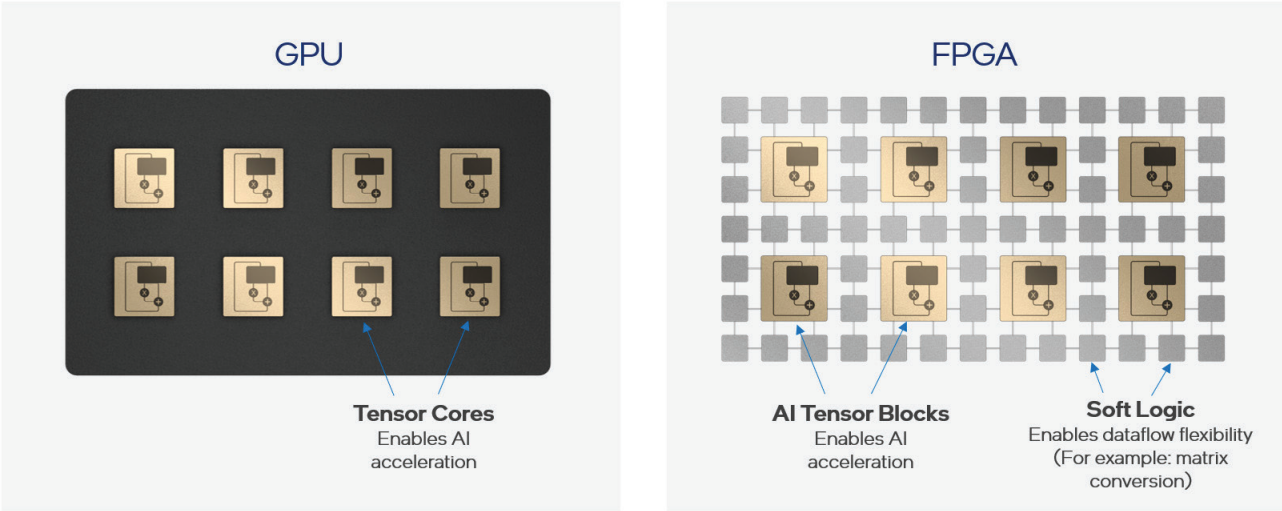

Figure 1. Both GPUs and FPGAs have Tensor Cores. But FPGAs have soft logic that can be woven in and out of the data flow [5]

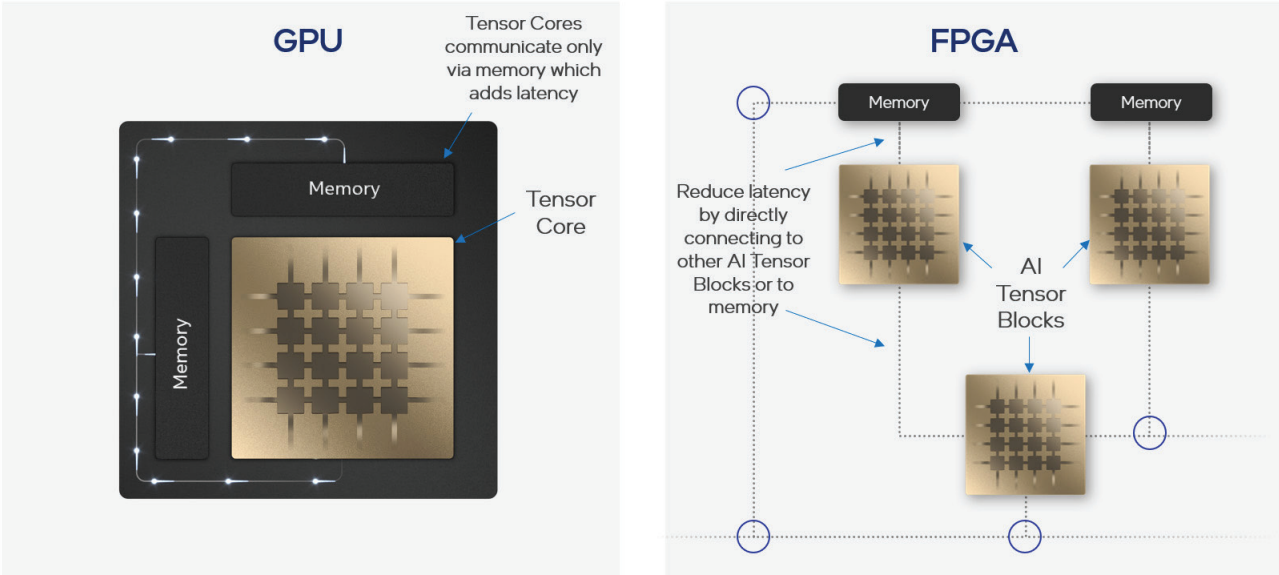

Figure 2. (Left) GPU data is read from the memory system processed by the Tensor Core and written back to the memory system. (Right) FPGA data can be read from memory, but data flow can be arranged to one or more Tensor Cores in parallel. The output can be consumed by any number of Tensor Cores with minimal transfer overhead. Data can be written back to memory or routed anywhere else [5].

Communication efficiency: GPU Tensor Cores communicate via memory systems and do not have direct connections among them. On the other hand, FPGA AI Tensor Blocks can be directly connected to each other using the flexible FPGA spatial fabric providing options to scale optimized compute resources that are not available in GPUs (see Figures 1 and 2). Thus, FPGAs provide efficient data coordination across tensor units.

Consolidation of results: The partial result of each tensor unit often needs to be combined to form the final output, such as concatenating or reducing output matrix blocks produced by multiple tensor units to arrive at a final output matrix. In GPUs, Tensor Core outputs must be synchronized and combined via the memory system that can introduce latency. In FPGAs, the fabric can directly gather the results in a way optimized for the target computation. The direct links across the FPGA AI Tensor Blocks can also be used to improve efficiency further (see Figures 1 and 2). All of this can happen on-chip with optimized number of cycles.

Consumption of results: Outputs of tensor units are often consumed by the next part of the application such as an activation function. In GPUs, consolidated tensor results must be stored in memory system (hierarchy of on-chip and off-chip memories). Additionally, due to synchronization that was done during the tensor operation, the GPU core pipelines may need to ramp-up to execute the next part of the application that takes the tensor results as its inputs. These delays and inefficiencies can be avoided in FPGAs since their fine-grained nature enables the next part of the application to be implemented in the fabric to directly receive the tensor result.

Sharing across tensor units: In AI computation, input tensors may be shared across multiple tensor units. In GPUs, this sharing is inefficient because no direct broadcast path is available across multiple Tensor Cores. Inputs are first loaded into a local register file via the memory system before a Tensor Core can access it. In FPGAs, a "broadcast network" can be implemented in the fabric, and the direct link across a set of Tensor Blocks also facilitate input broadcast.

Matrix conversion: GPU Tensor Cores operate on a specific, predefined matrix layout. Hence, if a different matrix layout is required by the target application, it must first be converted into the predefined layout present in the GPU. The overheads for conversion could be non-trivial as it can require multiple read/write operations to the memory subsystem. For example, the GPU may need to read data from memory, shuffle some bytes, write back to memory and repeat for the subsequent blocks until the new format is ready. Then the GPU reads the new matrix format from memory, does computation and then writes the result back to memory.

In FPGAs, any needed matrix format conversion logic can be implemented efficiently in-line on the FPGA fabric by programming the soft logic to convert from the desired input matrix format into vectors that go into the AI Tensor Blocks. This method greatly reduces the latency.

Programming flow: GPUs can be programmed via software application programming interfaces (APIs) and libraries. But FPGAs offer multiple options. Using conventional FPGA developer flows, AI-optimized FPGAs can be programmed at cycle- and bit-level via register transfer level (RTL). Additionally, there are FPGA developer flows that offer higher levels of abstraction and software programmability. A soft processor overlay is one such example that we will discuss briefly later in this white paper, but more details can be found in [1].

Beyond Peak Performance

The introduction of tensor compute in AI-optimized hardware has given rise to peak TOPS as a key metric when comparing acceleration solutions. However, peak TOPs metrics can be misleading, as these levels of performance can only be attained when tensor compute is utilized 100%. This is not the case in real applications. Utilization of tensor compute is typically affected by – (1) mapping of a workload to available tensor compute units; (2) end-to-end system-level overheads of bringing the data in/out of the hardware. The work presented in [1] studies the actual achievable performance of AI-optimized FPGAs and GPUs through evaluation of both compute and system-level performance on key AI workloads.

AI Soft Processor

Intel researchers have developed an AI soft processor called the Neural Processing Unit (NPU). This AI soft processor is designed for low-latency, low-batch inference. It maintains model persistence by keeping all model-weights on one or multiple connected FPGAs to improve latency.

As shown in Figure 3, a complete NPU software toolchain has been implemented. The toolchain allows an application developer to write NPU programs in a higher-level programming language such as Python. This approach abstracts away the FPGA hardware complexity and allows an application developer to rapidly experiment with various AI models [1]. Low level details of the NPU architecture and the software toolchain are outside the scope of this white paper and are described in [1].

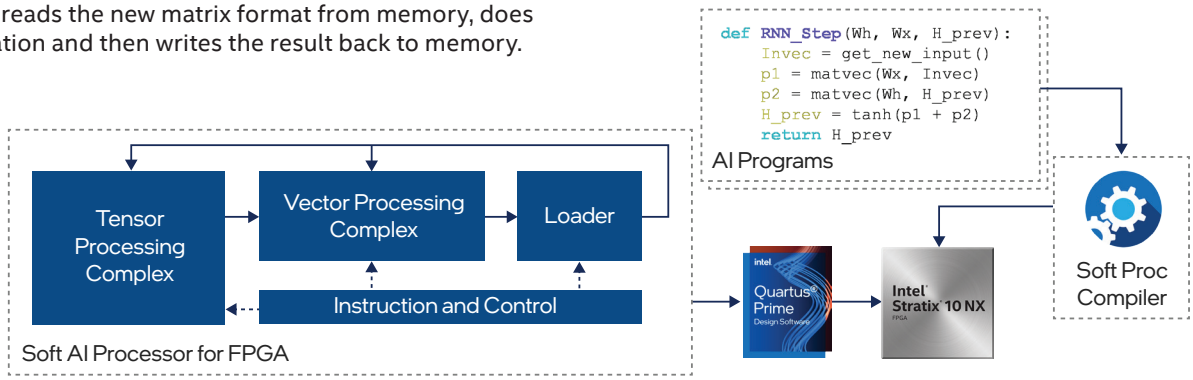

Figure 3. High-level overview of the NPU overlay architecture and the front-end tool chain for programming the NPU soft processor

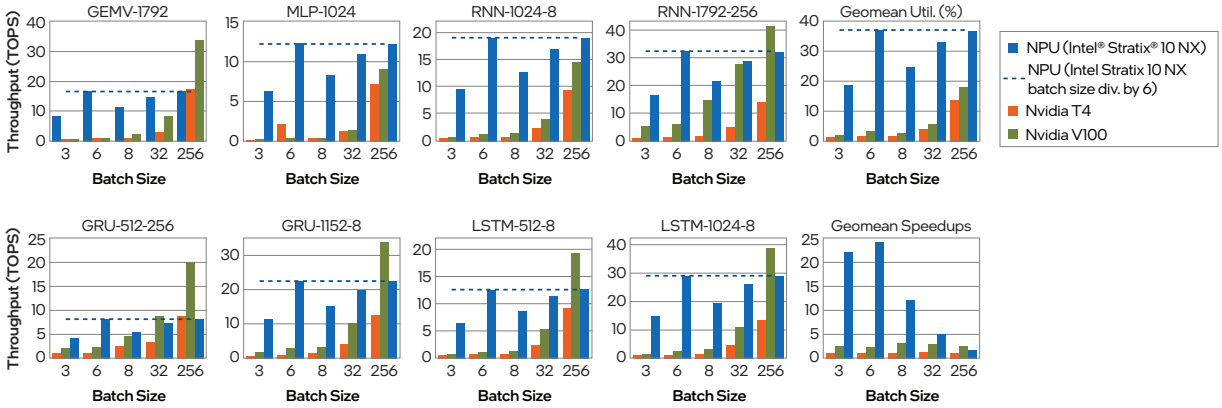

Figure 4. Performance of NVIDIA V100 and NVIDIA T4 vs. NPU on the Intel Stratix 10 NX FPGA at various batch sizes. Dashed lines show NPU performance for batch sizes divisible by 6 [1].

FPGA vs. GPU Compute Performance Comparison

The key area of focus in this study is the compute performance. Figure 4 compares the performance of the NPU on the Intel Stratix 10 NX FPGA to the NVIDIA T4 and V100 GPUs across a wide range of deep learning workloads including Multilayer Perceptron (MLP), General Matrix Vector Multiplication (GEMV), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). GEMV and MLP are specified by their matrix sizes, while RNNs, LSTMs, and GRUs are specified with their sizes and number of time steps. For example, LSTM-1024-16 workload describes an LSTM with 1024x1024 matrices and 16 time steps.

As shown in Figure 4, the Intel Stratix 10 NX FPGA implementing an NPU delivers significantly higher TOPS performance than both GPUs for batch sizes of 3 and 6, and similar or better TOPS up to batch sizes of 32. The best speedup is achieved at batch size 6 that the NPU is optimized for, where the Intel Stratix 10 NX device shows approximately 24X better performance compared to the NVIDIA T4, and 12X better performance compared to the NVIDIA V100. Even at batch size 3, where the Intel NPU is only 50% utilized, the Intel Stratix 10 NX FPGA displays approximately 22X and 9X average performance advantage over NVIDIA T4 and NVIDIA V100 GPUs respectively. For batch sizes divisible by 6, as shown by dotted lines in Figure 4, the Intel NPU is fully utilized to achieve 100% efficiency. At medium batch size of 32, the Intel Stratix 10 NX FPGA has better performance than the NVIDIA T4 and comparable performance to the much larger NVIDIA V100. The bottom right histogram summarizes the geometric mean (geomean) speedups of all studied workloads relative to the NVIDIA T4 performance at various batch sizes.

In Figure 4, the top right histogram shows the geomean utilization of the Intel NPU compared to both NVIDIA T4 and NVIDIA V100 GPUs at all the studied batch sizes. The Intel NPU achieves a geomean utilization of 37.1% at batch size 6 compared to 1.5% and 3% for the NVIDIA T4 and NVIDIA V100, respectively. FPGAs achieve significantly higher utilization of their compute resources compared to GPUs due to architectural differences described earlier in this paper. These differences enable FPGAs to apply their compute resources much more efficiently than GPUs, which result in much higher performance in real-world applications.

It is abundantly clear from these results that Intel Stratix 10 NX FPGAs can not only achieve an order of magnitude better performance than GPUs at low-batch real-time inference, but also compete effectively with high-batch inference.

End-to-End Performance Comparison

Results published in [1] highlight another key value proposition of the Intel Stratix 10 NX FPGA, end-to-end performance on Recurrent Neural Network (RNN) workloads for short and long sequences. Figure 5 shows system-level execution time of RNN workloads at batch size 6 and sequence lengths of 8 and 256. After accounting for system overheads (e.g. data transfers in and out of the device, and initialization), the Intel FPGA system achieves 16X-19X higher performance than the NVIDIA T4 GPU and 15X-25X higher performance than the NVIDIA V100 GPU for short sequences. Additionally, for long sequences, the Intel Stratix 10 NX FPGA system achieves 11X-16X higher performance than the NVIDIA T4 GPU and 5X-6X higher performance than the NVIDIA V100 GPU. This is because FPGA's finer grained programmability allows efficient overlapping of compute with data transfers and reduces initialization overhead [1].

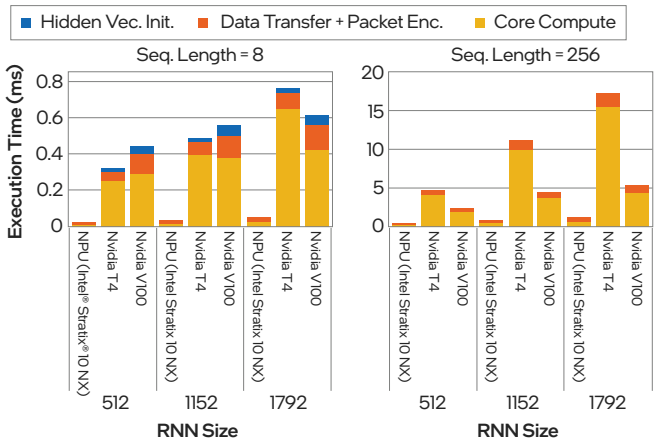

Figure 4. System-level execution time of RNN workloads for short and long sequences (lower is better) [1]

The Intel Stratix 10 NX FPGA delivers much better end-to-end performance numbers due to its architectural differences and flexible programming model. It does not suffer from the same overheads that GPUs have.

Conclusion

The Intel Stratix 10 NX FPGA, with its highly flexible architecture, delivers 24X higher average performance than an NVIDIA T4 GPU and 12X higher average performance than an NVIDIA V100 GPU (as summarized in tables 1 and 2). This order of magnitude higher performance over the NVIDIA GPU makes the Intel Stratix 10 NX FPGA a better choice for the following applications requiring real-time inference:

- Natural language processing
- Financial fraud prevention
- Real-time video analytics

Batch Size	Intel® Stratix® 10 NX Performance Advantage over NVIDIA T4	Intel Stratix 10 NX Performance Advantage over NVIDIA V100
3	22.3X	9.3X
6	24.2X	11.7X

Table 1. Summary of average performance comparison of AI workloads across several combinations of configurations [1]

Batch Size	Sequence Length	Intel Stratix 10 NX Performance Advantage over NVIDIA T4	Intel Stratix 10 NX Performance Advantage over NVIDIA V100
6	8	16X - 19X	15X - 25X
6	256	11X - 16X	5X - 6X

Table 2. End-to-end system level comparison of RNN workloads for batch size 6 [1]

With its high arithmetic density, the Intel Stratix 10 NX FPGA delivers features that are critical for high-performance, latency-sensitive AI systems where real achievable performance is a key differentiator.

For Additional Information

For more details, visit the Intel Stratix 10 NX FPGA website.

References

[1] E. Nurvitadhi et al., "Beyond Peak Performance: Comparing the Real Performance of AI-Optimized FPGAs and GPUs", 2020 IEEE International Conference on Field Programmable Technology (FPT). Available (Author's version of conference paper): Intel® Stratix® 10 NX AI-Optimized FPGA Vs GPUs

[2] R. D'Souza et al., "Pushing AI Boundaries with Scalable Compute-Focused FPGAs", 2020. [Online]. Available: www.intel.com/scalable-compute-fpga

[3] J. Fowers et al., "A Configurable Cloud-Scale DNN Processor for RealTime AI," in International Symposium on Computer Architecture (ISCA), 2018, pp. 1–14

[4] Z. Jia et al., "Dissecting the NVidia Turing T4 GPU via Microbenchmarking," arXiv preprint arXiv:1903.07486, 2019

[5] M. Langhammer, "A new FPGA Architecture Optimized for AI Acceleration", Linley Fall Processor Conference 2020 – The Linley Group. [Online]

Available: Intel: A New FPGA Architecture Optimized for AI Acceleration - YouTube

[6] B. Rouhani et al., "Pushing the Limits of Narrow Precision Inferencing at Cloud Scale with Microsoft Floating Point", in Neural Information Processing Systems (NeurIPS), 2020

Available: https://proceedings.neurips.cc/paper/2020/file/747e32ab0fea7fbd2ad9ec03daa3f840-Paper.pdf

[7] M. Drumond, et. Al., "Training DNNs with hybrid block floating point", in Neural Information Processing Systems (NIPS), 2018

Available: https://papers.nips.cc/paper/2018/file/6a9aeddfc689c1d0e3b9ccc3ab651bc5-Paper.pdf