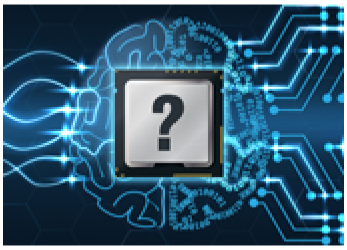
FPGA vs GPU for Machine Learning Applications: Which one is better?

Can FPGAs beat GPUs?

Farhad Fallahlalehzari, Applications Engineer

Like (2) Comments (0)

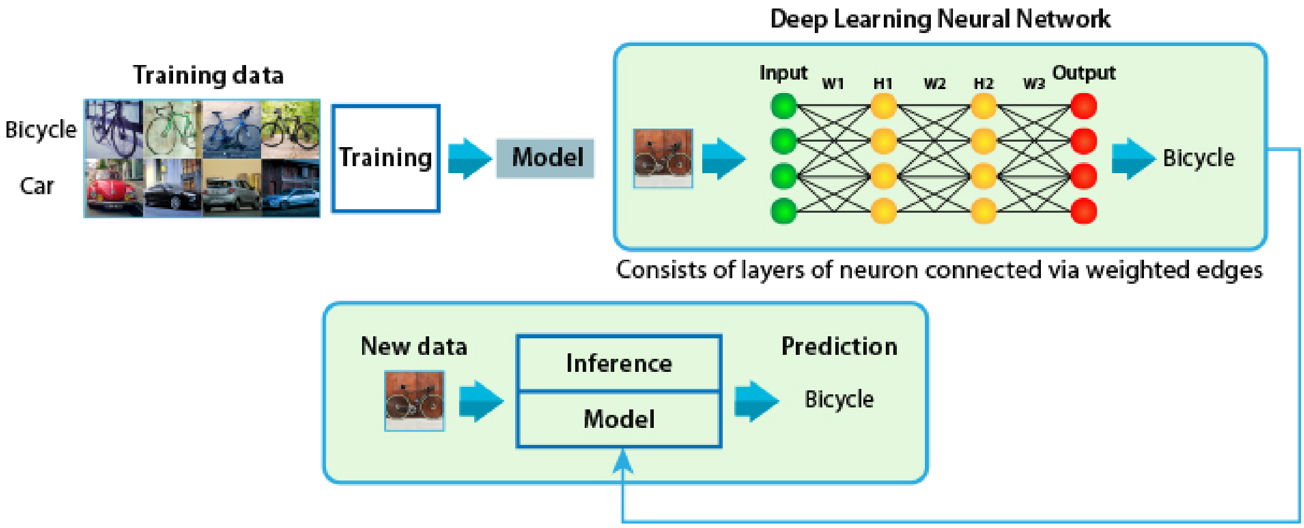


FPGAs or GPUs, that is the question.

Since the popularity of using machine learning algorithms to extract and process the information from raw data, it has been a race between FPGA and GPU vendors to offer a HW platform that runs computationally intensive machine learning algorithms fast and efficiently. As Deep Learning has driven most of the advanced machine learning applications, it is regarded as the main comparison point.

Even though GPU vendors have aggressively positioned their hardware as the most efficient platform for this new era, FPGAs have shown a great improvement in both power consumption and performance in Deep Neural Networks (DNNs) applications, which offer high accuracies for important image classification tasks and are therefore becoming widely adopted [1]. As there are various tradeoffs to consider, it is hard to answer with just a “Yes” or “No”. To form your own opinion, I invite you to read this blog, which first touches on the benefits and barriers of using FPGAs and GPU, anc consider the studies performed by the main players in this field; namely Xilinx, Intel, Microsoft and UCLA research labs.

DNNs are widely used as learning models because of their inference accuracies. They can be formulated as graphs similai to the one shown in Figure 1.



the network is derived by going through each layer. Each neuron's value is calculated by multiplying and accumulating all the values of the previous layer's neurons with the corresponding edge weights. This shows that the computation relies on the multiplication and accumulation operations. To predict a given sample, passing forward through the layers is enough. For training, the prediction error is passed back to the model to update the network weights for accuracy. Users of DNNs have been using different data types which has challenged GPUs and brought home the benefits of FPGAs for machine learning application. These capabilities will now be explained in more details.

FPGA vs GPU - Advantages and Disadvantages

To summarize these, I have provided four main categories: Raw compute power, Efficiency and power, Flexibility and ease of use, and Functional Safety. The content of this section is derived from researches published by Xilinx [2], Intel [1], Microsoft [3] and UCLA [4].

1. **Raw Compute Power:** Xilinx research shows that the Tesla P40 (40 INT8 TOP/s) with Ultrascale+TM XCVU13P FPGA (38.3 INT8 TOP/s) has almost the same compute power. When it comes to on-chip memory, which is essential to reduce the latency in deep learning applications, FPGAs result in significantly higher computer capability. The high amount of on-chip cache memory reduces the memory bottlenecks associated with external memory access as well as the power and costs of a high memory bandwidth solution. In addition, the flexibility of FPGAs in supporting the full range of data types precisions, e.g., INT8, FTP32, binary and any other custom data type, is one of the strong arguments for FPGAs for Deep Neural Network applications. The reason behind this is because deep learning applications are evolving at a fast pace and users are using different data types such as binary, ternary and even custom data types. To catch up with this demand, GPU vendors must tweak the existing architectures to stay up-to-date. So, GPU users must halt their project until the new architecture becomes available. Therefore, the re-configurability of FPGAs comes in handy because users can implement any custom data type into the design.
2. **Efficiency and Power:** FPGAs are well-known for their power efficiency. A research project done by Microsoft on an image classification project showed that Arria 10 FPGA performs almost 10 times better in power consumption. In other research, Xilinx showed that the Xilinx Virtex Ultrascale+ performs almost four times better than NVidia Tesla V100 in general purpose compute efficiency. The main reason for GPUs being power- hungry is that they require additional complexity around their compute resources to facilitate software programmability. Although the NVidia V100 provides a comparable efficiency to the Xilinx FPGAs (almost the same Giga operations per second per watt GOP/s/W) due to the hardened Tensor Cores for tensor operations for today's deep learning workloads, it is unpredictable for how long NVidia's Tensor Cores remain efficient for deep learning applications, as this field is evolving quite fast. For other general-purpose workloads, i.e. other than deep learning, the NVidia V100 is challenged from the performance and efficiency perspective. The re-configurability of FPGAs in addition to the software development stack of main vendors such as Xilinx (SDAccel) and Intel (FPGA SDK for OpenCL) provides much higher efficiency for a large number of end applications and workloads.
3. **Flexibility and Ease-of-Use:** Data flow in GPUs is defined by software and is directed by the GPU's complex memory hierarchy (as is the case with CPUs). The latency and power associated with memory access and memory

workload can be mapped efficiently into the vastly parallel architecture, and if enough parallelism cannot be found within the threads, this results in lower performance efficiency. FPGAs can deliver more flexible architectures, which are a mix of hardware programmable resources, DPS and BRAM blocks. User can address all the needs of a desired workload by the resources provided by FPGAs. This flexibility enables the user to reconfigure the datapath easily, even during run time, using partial reconfiguration. This unique re-configurability means the user is free from certain restrictions, like SIMT or a fixed datapath, yet massively parallel computations are possible. The flexible architecture of FPGAs has shown great potential in sparse networks, which is one of the hot trends in current machine learning applications.Another important feature of FPGAs, and one that makes them even more flexible, is the any-to-any I/O connection. This enables FPGAs to connect to any device, network, or storage devices without the need for a host CPU. Regarding ease-of-use, GPUs are more ‘easy going’ than FPGAs. This is one of the main reasons that GPUs are widely being used these days. CUDA is very easy to use for SW developers, who don’t need an in-depth understanding of the underlying HW. However, to do a machine learning project using FPGAs, the developer should have the knowledge of both FPGAs and machine learning algorithms. This is the main challenge for FPGA vendors; to provide an easy development platform for users. Xilinx has put considerable effort into this by providing tools such as SDSoC, SDAccel and Vivado HLS. These tools have made the process of FPGA design flow much easier for SW engineers, as they can easily convert their C/C++ code to the HDL.

4. **Functional Safety:** GPUs are originally designed for graphics and high-performance computing systems where safety is not a necessity. Some applications, such as ADAS, do require functional safety. In such a case, GPUs should be designed in a way to meet the functional safety requirements. This could be a time-consuming challenge for GPU vendors. On the other hand, FPGAs have been used in industries where functional safety plays a very important role such as automation, avionics and defense. Therefore, FPGAs have been designed in way to meet the safety requirement of wide range of applications including ADAS. In this respect, Xilinx Zynq®-7000 and Ultrascale+TM MPSoC devices are designed to support safety-critical applications such as ADAS.

It is clear that the application and also the project goal are very important to choose the right HW platform. Based on the mentioned features, FPGAs have shown stronger potential over GPUs for the new generation of machine learning algorithms where DNN comes to play massively. Based on the studies alluded to in this blog, I would say the main winning points of FPGAs over GPUs would be the flexibility provided by FPGAs to play with different data types - such as binary, ternary and even custom ones – as well as the power efficiency and adaptability to irregular parallelism of sparse DNN algorithms. However, the challenge for FPGA vendors is to provide an easy-to-use platform.

Building any type of advanced FPGA designs such as for machine learning require advanced FPGA design and verification tools. Simulation is the de-facto verification methodology for verifying FPGA designs using mixed-language HDL with SystemC/C/C+ testbenches. Compilation and simulation speed are the key factors - the faster simulations you can do the more test scenarios you can check within a given timeframe. Majority of the time that you will spend during verification is debugging so you would need advanced debugging tools in your arsenal such as Waveform Viewer, Advanced Dataflow, State Machine Coverage, Memory Visualization and Breakpoints. You can check out our High-Performance Simulator, Riviera-PRO and evaluate it for free.

Once you are ready for machine learning inference, having a robust and high-capacity FPGA board with rich set of

In the next blog, I will cover FPGA advantages in other AI applications.

[1] Nurvitadhi, Eriko, et al. "Can FPGAs beat GPUs in accelerating next-generation deep neural networks?" Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2017.
[2] Cathal Murphy and Yao Fu, "Xilinx All Programmable Devices: A Superior Platform for Compute-Intensive Systems".
[3] Ovtcharov, Kalin, et al. "Accelerating deep convolutional neural networks using specialized hardware." Microsoft Research Whitepaper 2.11 (2015).
[4] Cong, Jason, et al. "Understanding Performance Differences of FPGAs and GPUs" Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2018.

Tags: Aceleration, Embedded, FPGA, Hardware, HDL, Validation, Verilog, VHDL, Xilinx

Farhad Fallah works as an Application Engineer focusing on Aldec's Embedded Systems and Hardware Prototyping solutions. As a technical support engineer, Farhad has a deep understanding of developing and debugging embedded system designs using Aldec's TySOM boards (Xilinx Zynq based embedded development boards). He is also proficient in FPGA/ASIC digital system design and verification. He received his master's degree in Electrical and Computer Engineering, concentrating on Embedded Systems and Digital Systems Design from University of Nevada, Las Vegas in year 2016.

RELATED RESOURCES

Products: TySOM™ EDK Embedded

Comments

Have a comment?

Log In or Register to comment

Ask Us a Question

x

Legal | Privacy | Site Map | RSS Feeds | Feedback