WHITE PAPER

FPGA

Accelerating Deep Learning with the OpenCL™ Platform and Intel® Stratix® 10 FPGAs

Intel® FPGAs leverage the OpenCL™ platform to meet the image processing and classification needs of today's image-centric world.

Authors

Andrew Ling, Ph.D.
Machine Learning Engineering Manager
Intel Programmable Solutions Group

Davor Capalija, Ph.D.
Machine Learning Engineering Manager
Intel Programmable Solutions Group

Gordon Chiu
Director, Software Engineering
Intel Programmable Solutions Group

Table of Contents

Introduction

Internet video traffic will grow fourfold from 2015 to 2020.[1] With this explosion of visual data, it is critical to find effective methods for sorting, classifying, and identifying imagery. Convolutional neural networks (CNNs), a machine learning methodology based on the function of the human brain, are commonly used to analyze images. Software separates the image into sections, often overlapping, and then analyzes them to form an overall map of the visual space. This process involves several complex mathematical steps to analyze, compare, and identify the image with a low error rate.

Developers create CNNs using computationally intensive algorithms and implement them on a variety of platforms. This white paper discusses a CNN implementation on an Intel® Stratix® 10 FPGA that processes 14,000 images/second at 70 images/second/watt for large batches and 3,015 images/second at 18 images/second/watt for batch sizes of 1.† As these numbers show, Intel Stratix 10 FPGAa are competitive with other high-performance computing (HPC) devices such as GPUs for large batch sizes, and are significantly faster than other devices at low batch sizes.

CNN benchmarks

The Stanford Vision Lab has hosted the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) since 2010. Competitors are challenged to develop a CNN algorithm that can analyze and classify objects in data sets comprising millions of images or video clips. The 2012 contest-winning algorithm, dubbed AlexNet,* [2] provided a huge leap forward in reducing the classification error rates compared to previous algorithms.[3] In 2014, the winning algorithm (GoogLeNet*) used an improved algorithm to reduce the error rate even further.[4] Intel has developed a novel design that implements these benchmark algorithms with modifications to boost the performance on Intel FPGAs.

CNN algorithms consist of a series of operations or layers. For example, the AlexNet algorithm has:

- Convolution layers that perform a convolution operation on a 3-dimensional (3D) data array (called a feature map) and a 3D filter. The operation uses a rectified linear unit (ReLU) as an activation function.
- Cross-channel local response normalization layers that scale the feature map elements by a factor that is a function of the elements at the same location in adjacent channels as the element being normalized.
- Max pooling layers that read the data in 2-dimensional (2D) windows and output the maximum values.

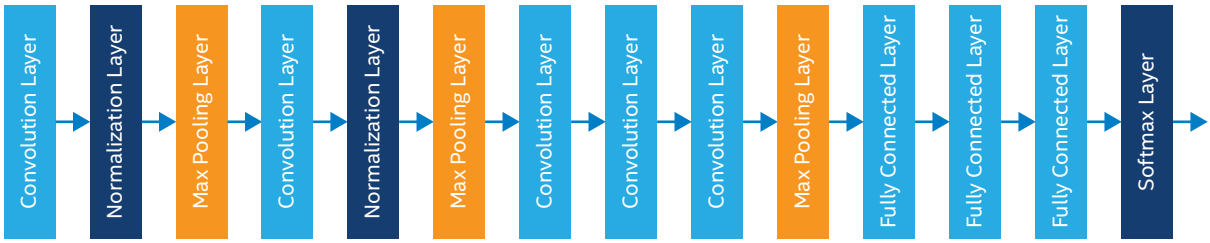

Figure 1. Alexnet Algorithm Layers

- Fully connected layers that perform a special convolution function in which each node is connected to every node in the previous layer. The output can be represented as the matrix dot product of the flattened (2D) version of the input feature map with a flattened filter.
- A softmax layer that normalizes the input feature map values with a softmax function, which is a normalized exponential function. This layer outputs a 1,000-element vector containing the probability that the original image belongs to one of the 1,000 possible classes in the ILSVRC image set.

GoogLeNet contains convolution, pooling, and softmax layers combined in repeating "Inception" blocks.

Using Intel Stratix 10 FPGAs for CNNs

CNN algorithms are computationally intensive, and developers typically use matrix multiplication to obtain the results. This multiplication can require significant external memory bandwidth to pass large data sets back and forth. Additionally, most FPGA CNN designs experience various inefficiencies:

- Many designs implement only the convolution layers causing the remaining layers to become a bottleneck.
- Transferring large amounts of data between the FPGA and external memory can become a bottleneck.
- Many designs do not take advantage of the FPGA's peak operational performance, leading to low performance.

Using the OpenCL™ platform, Intel has created a novel deep learning accelerator (DLA) architecture that is optimized for high performance. In most CNNs, the convolution layers use most of the total number of floating-point calculations. The DLA implements parallel computations to maximize the convolution layer throughput and use as many FPGA DSP blocks in parallel as possible. The DLA building blocks are written as OpenCL kernels that execute independently and concurrently.

Exploiting parallelism

Using parallelism improves overall throughput. The convolution layers have four dimensions that the DLA vectorizes:

- Output feature columns (Q)
- Output feature maps (K)
- Input feature maps (C)
- Input feature columns (W)

The DLA uses dot products to perform calculations with the vectors. Breaking up the convolution operations into separate dot products uses the FPGA's DSP blocks more effectively. Additionally, converting the input and output feature columns to vectors enables the DLA to simplify the algorithm using Winograd* transformations as discussed later.

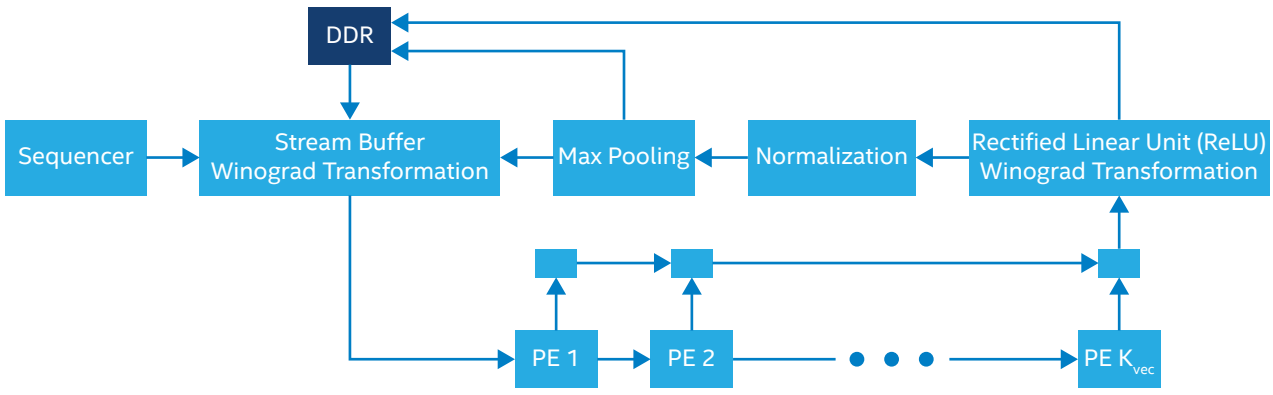

Figure 2. DLA Architecture

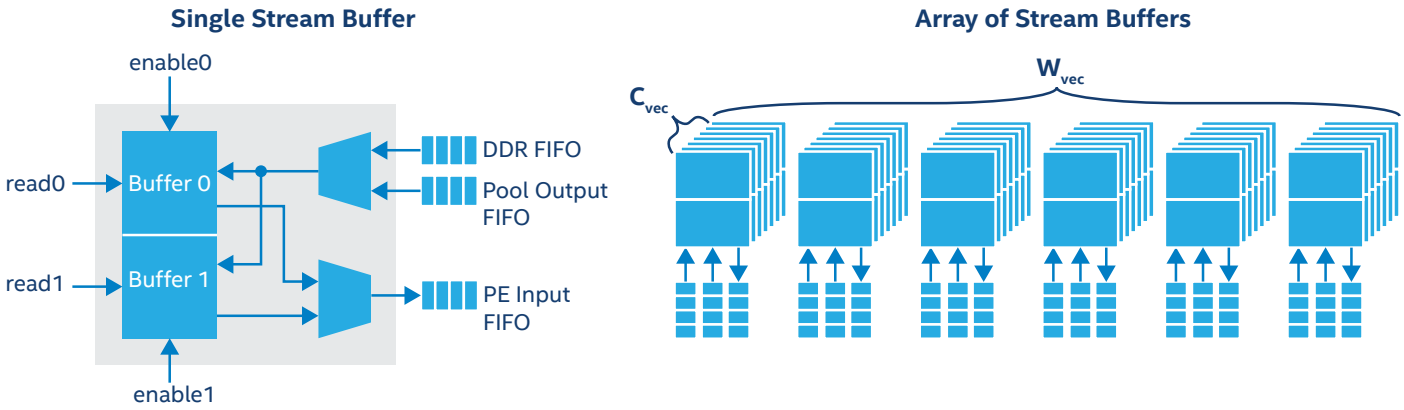

Figure 3. Stream Buffers

Caching data on chip

The DLA caches feature data in on-chip RAM and streams it to a daisy chain of parallel processing elements (PEs) to compute the convolutional and fully connected layers. To avoid idle computation cycles, the DLA double buffers the data and overlaps convolutions with PE cache updates. While the convolution layer executes, the DLA streams feature data into the PEs and simultaneously stores the outputs back into the RAM buffers. This architecture reduces unnecessary external memory accesses and eliminates them as a potential bandwidth bottleneck. Additionally, the caching allows the DLA to re-use input feature maps and filter weights.

Simplifying calculations using the Winograd* transformation

Shmuel Winograd developed innovative minimal filtering algorithms that reduce the complexity of a CNN layer by factor of four compared to traditional convolution operations.[5] These algorithms work particularly well with small filters. The AlexNet and GoogLeNet topologies use small 3x3 filters in most of their convolution layers, and other currently modeled CNN architectures also use small filters. By incorporating the Winograd transformation, the DLA reduces the number of multiply-accumulate operations needed to calculate the convolutions. For example, the DLA uses only six multiplications and additions each clock cycle instead of the 12 required for standard convolution operations.

Modeling for the optimal architecture

The developer can model the DLA performance using the input/output feature maps and columns vectorization factors (Qvec, Kvec, Cvec, and Wvec). For example, the following equation models the number of DSP blocks required (without the Winograd transformation):

$$N_{DSPblocks} = (W_{vec} - Q_{vec} + 1) \times Q_{vec} \times K_{vec} \times C_{vec} \times 0.5$$

Similarly, the developer can model:

- The number of M20K RAM blocks required to store the largest input and output feature map for any given layer.

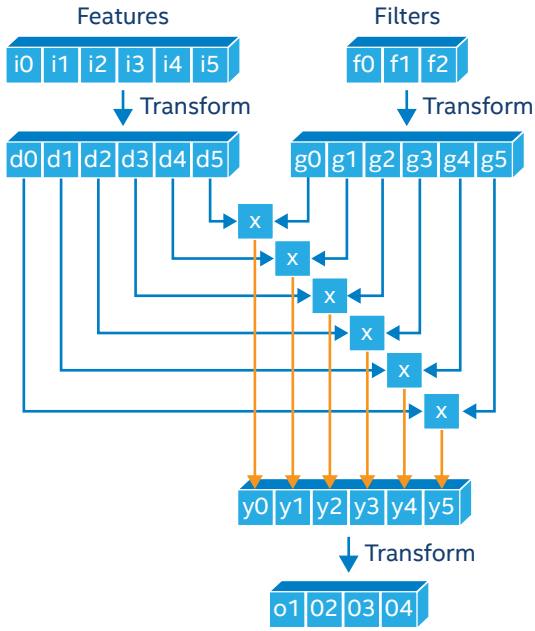

Figure 4. Using the Winograd Transformations

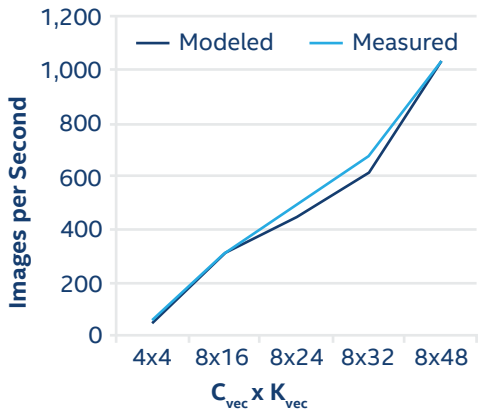

Figure 5. Comparing Modeling and Empirical Testing

- The number of cycles needed to process an image for one convolution layer.
- The throughput in images per second.

This modeling allows the developer to determine the optimal C_{vec} and K_{vec} values for a target FPGA for a specified f_{MAX}, W_{vec}, and Q_{vec}. The modeling and empirical testing results align closely as shown in Figure 5.

DLA performance results

Figures 6 - 9 provide projected AlexNet and GoogLeNet data benchmark results. It compares the DLA running on Intel Arria® 10 and Intel Stratix 10 FPGAs to GPU performance using the nVidia* Tesla* P4 and P40 GPUs (which is based on nVidia's Pascal* architecture[6]). The benchmarks were implemented using the following hardware and software:

- Intel Arria 10 GX1150 FPGA
- Intel Stratix 10 GX2800 FPGA
- Intel Quartus® Prime Design Suite v16.1 with OpenCL
- nVidia P40 and P4 GPUs[7]
- nVidia TensorRT* neural network inference engine[8]

The nVidia P4 and P40 results are available on the nVidia developer web site, were presented at the 2016 GPU Technology Conference, as well as calculated by Intel during the experiment.

As Figures 6-9 show, Intel Arria 10 and Intel Stratix 10 FPGAs are competitive versus the latest generation nVidia GPU. †

Conclusion

In an increasingly image-centric world, there is a growing need for powerful image processing and classification. Intel's novel DLA architecture leverages the OpenCL language to compute CNNs on Intel FPGAs. This DLA reduces memory bandwidth by storing input and output feature maps on chip. Additionally, by vectorizing the input/output feature maps and columns, the DLA takes advantage of Winograd transformations to simplify the operations required to calculate results. Combining these efficiencies with Intel Stratix 10 FPGAs produces a power/performance efficiency of 70 images/second/watt for the AlexNet benchmark (large batches) and 18 images/second/watt for the GoogLeNet benchmark (batch = 1).

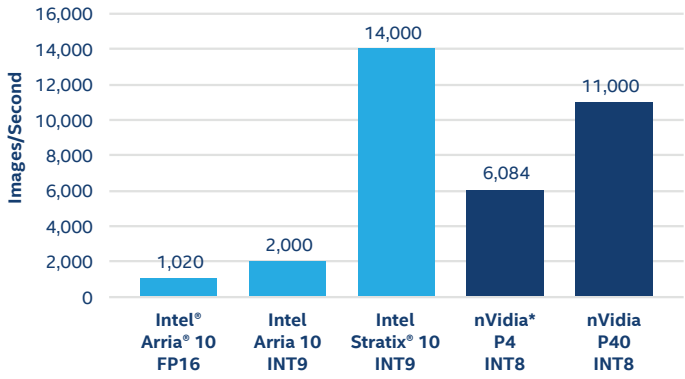

Figure 6. Comparing AlexNet Performance, Large Batch ≥32†

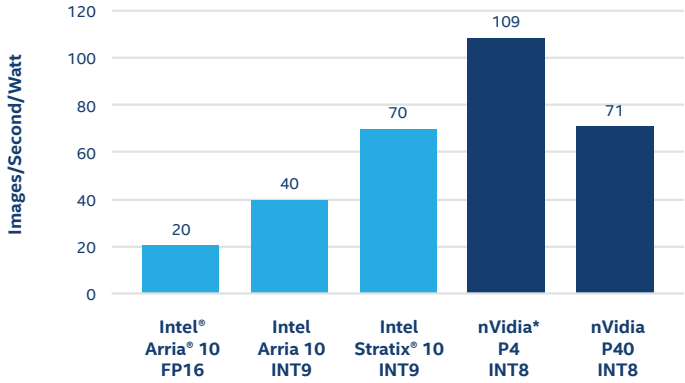

Figure 7. Comparing AlexNet Efficiency, Large Batch ≥32†

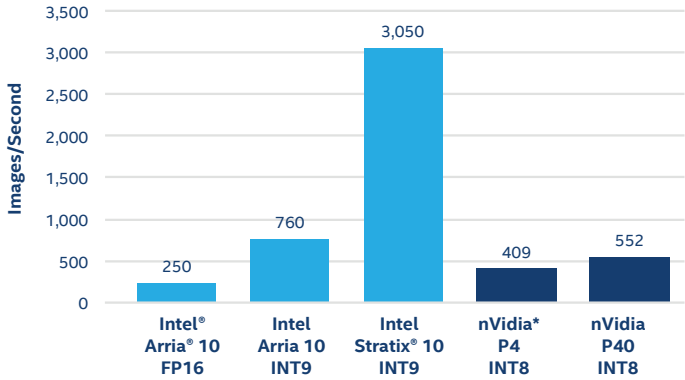

Figure 8. Comparing GoogLeNet Performance, Batch = 1†

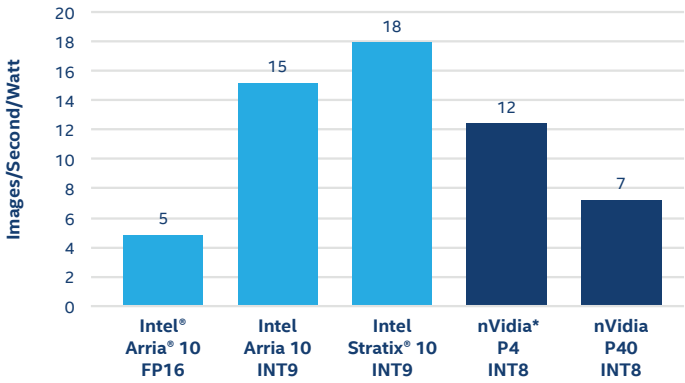

Figure 9. Comparing GoogLeNet Efficiency, Batch = 1†

References

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
[2] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[3] http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf
[4] http://deeplearning.net/2014/09/19/googles-entry-to-imagenet-2014-challenge/
[5] https://arxiv.org/pdf/1509.09308v2.pdf
[6] https://devblogs.nvidia.com/parallelforall/new-pascal-gpus-accelerate-inference-in-the-data-center/
[7] http://hpcadvisorycouncil.com/events/2016/china-conference/wp-content/uploads/2016/09/10_NVIDIA-HPC-Advisory.pdf
[8] https://developer.nvidia.com/tensorrt

Where to get more information

For more information about Intel and Intel Stratix 10 FPGAs, visit **https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html**

For more information about Intel and Intel Arria 10 FPGAs, visit **https://www.altera.com/products/fpga/arria-series/arria-10/overview.html**

For technical information about the DLA, read the conference paper at **http://dl.acm.org/citation.cfm?id=3021738**

§ OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

† Projected processing capability for Intel Stratix 10 FPGA using the AlexNet and GoogLeNet benchmarks. Projected processing capability for Intel Arria 10 FPGA using INT9 algorithm enhancement. Tests measure performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.