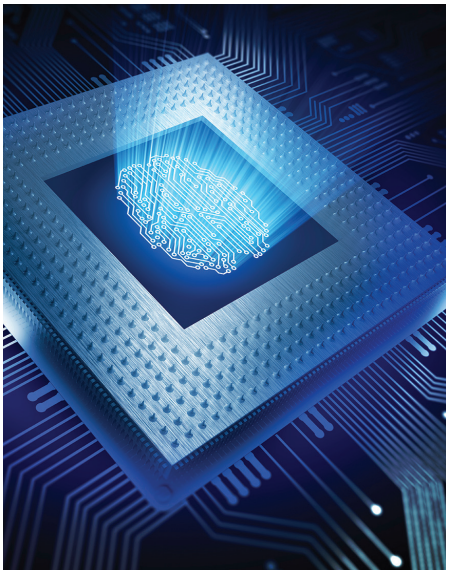
SOLUTION BRIEF

OpenCL Efficient Neural Networks

Efficient Implementation of Neural Network Systems Built on FPGAs, and Programmed with OpenCL™



Deep learning neural network systems currently provide the best solution to many large computing problems for image recognition and natural language processing. Neural networks are inspired by biological systems, in particular the human brain; they use conventional processing to mimic the neural network and create a system that can learn by observing. While large strides have recently been made in the development of high-performance systems for neural networks based on multi-core technology, significant challenges in power, cost, and performance scaling remain.

The Context

The most widely used deep learning systems are convolutional neural networks (CNNs). These systems use a feed forward artificial network of neurons to execute image identification or recognition. They use a reverse feed system for learning, and produce a set of weights to calibrate the execution system. CNNs are composed of layers. These layers include a convolution layer that extracts low-level features from the input to identify lines or edges within an image.

The pooling layers reduce variations with maxing or value averaging, pooling common features over a particular region of an image. The result can be passed on to further convolutions and pooling layers. The number of CNN layers correlates to the accuracy of the image recognition; more layers require more system performance. These layers can operate independently; therefore, they are implemented most efficiently in a data pipeline where data is passed from one layer to another. Multi-core processing systems use external memory to buffer the data between each layer, which requires significant amounts of memory bandwidth.

By far the most performance intense functionality in the neural network are the convolutions themselves. These complex sigmoidal functions are typically best implemented in single-precision floating-point math for image classification. Conventional processor cores must execute a large set of instructions for each convolution, which requires significant processing bandwidth.

$$I_{new}[x][y] = \sum_{x'=-1}^{1} \sum_{y'=-1}^{1} I_{old}[x+x'][y+y']$$

Figure 1. 2D Convolution Layer

There are two main challenges to achieve efficient implementation of CNNs. The first is the ability to execute functions in a pipeline, passing data from one layer to the next. The second is to execute the convolution functions efficiently. Additionally, these functions should be constructed with a methodology that allows easy reprogramming for different types of hardware and for porting to future advanced hardware, otherwise, each new implementation requires extensive re-optimization.

The Design

Field-programmable gate arrays (FPGAs) are a natural choice for implementing neural networks because they can combine computing, logic, and memory resources in a single device. However, FPGAs have been impractical for wide spread use in complex algorithmic-based systems due to the traditional low-level hardware programming environments. Intel® FPGA SDK for OpenCL™ solves this problem, making FPGAs useful for a wide array of acceleration applications using complex algorithms. Software developers can use the OpenCL C level programming standard to target FPGAs as accelerators to standard CPUs without having to deal with hardware level design. Additionally, Intel has developed a scalable convolutional neural network reference design for deep learning systems using the OpenCL programming language built with our SDK for OpenCL. This design was first implemented on the Stratix® V device series, and is now available for Arria® 10 devices. The design performance is being benchmarked using two popular CNN benchmarks: CIFAR-10 and ImageNet.

The design uses OpenCL kernels to implement each CNN layer. Data is passed from one layer to the next through a mechanism called OpenCL Channels or Pipes, which allow data to move from one kernel to the next without sending data to the external memory. This FPGA implementation is a data buffer that uses the internal memory structures next to the kernel itself, resulting in very efficient data movement through the neural network. In addition, the design can be adapted to systems that use cameras, sensors, or other input and output devices directly attached to the FPGA. The CNN system can interface directly with these devices through the same channel mechanism without the data going to the external memory. A typical GPU implementation batches images and requires significant external memory bandwidth. In contrast, the FPGA can process one image at a time with significantly greater data reuse on chip and less external memory bandwidth.

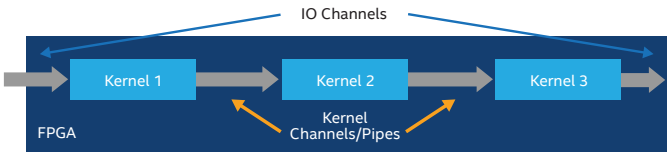


Figure 2. Neural Network Datapath

The convolutions are implemented using DSP blocks and logic in the FPGA. In the past, FPGA DSP blocks were efficient for multiplication, however, floating-point addition required significant FPGA logic. Arria 10 and Stratix 10 FPGAs have innovative DSP blocks that also deliver IEEE-754 floating-point addition, which results in less FPGA logic and higher clock speeds for better performance and lower power. At the 2015 Intel Developers Forum, Intel presented the Arria 10 AlexNet estimates shown below. A week later, a Microsoft* white paper at the Hot Chips Conference projected similar performance efficiency versus peak TFLOPs for Arria 10 GX1150 compared to the Titan X GPU, but with 2X better performance to power ratio. Stratix 10 FPGAs are expected to deliver over 5X better performance compared to Arria 10 FPGAs with more than 3X the DSP blocks at ~2X the clock speed, providing continued improvement in performance to power.

PROJECTED ALEXNET PERFORMANCE AND POWER[1]

CNN CLASSIFICATION PLATFORM	POWER (W)	PERFORMANCE (IMAGE/S)	EFFICIENCY (IMAGES/SEC/W)
E52699 Dual Xeon® CPU (18 core per Xeon)	321	1,320	4.11
PCIe* w/Dual Arria 10 1150	130[2]	1,200	9.27

Note:
1. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.
2. If the CPU is active with other tasks while the two FPGAs are used, use 65 W instead of 130 W.

[1] 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.

SS-1078-1.2