High Performance Binary Neural Networks on the Xeon+FPGA™ Platform

Duncan J.M. Moss*‡, Eriko Nurvitadhi†, Jaewoong Sim†, Asit Mishra†,
Debbie Marr†, Suchit Subhaschandra‡, Philip H.W. Leong*

*The University of Sydney, †Accelerator Architecture Lab and ‡FPGA Product Team, Intel Corporation

***Abstract*—Convolutional neural networks (CNNs) are deployed in a wide range of image recognition, scene segmentation and object detection applications. Achieving state of the art accuracy in CNNs often results in large models and complex topologies that require significant compute resources to complete in a timely manner. Binarised neural networks (BNNs) have been proposed as an optimised variant of CNNs, which constrain the weights and activations to $+1$ or -1 and thus offer compact models and lower computational complexity per operation.**

This paper presents a high performance BNN accelerator on the Intel®Xeon+FPGA™ platform. The proposed accelerator is designed to take advantage of the Xeon+FPGA system in a way that a specialised FPGA architecture can be targeted for the most compute intensive parts of the BNN whilst other parts of the topology can be handled by the Xeon™ CPU. The implementation is evaluated by comparing the raw compute performance and energy efficiency for key layers in standard CNN topologies against an Nvidia Titan X Pascal GPU and other published FPGA BNN accelerators. The results show that our single-package integrated Arria™ 10 FPGA accelerator coupled with a high-end Xeon CPU can offer comparable performance and better energy efficiency than a high-end discrete Titan X GPU card. In addition, our solution delivers the best performance compared to previous BNN FPGA implementations.

I. Introduction

Convolutional Neural Networks (CNNs) have become one of the most popular forms of machine learning with state-of-the-art performance in applications such as computer vision and artificial intelligence. Since AlexNet [4] in 2012, the computational power needed to run newer CNNs has increased significantly resulting in the pursuit of high performance linear algebra accelerators. Because the fundamental math operations are matrix multiplications, GPUs or the Xeon Phi excelled due to their high memory bandwidth and large amount of parallel processing units. Traditionally, CNNs have been implemented using single precision floating point math, however recent work [5], [6], [1], [9], [10] has shown that often a reduced precision representation can be sufficient for inference. As such, field programmable gate arrays (FPGA) have presented themselves as a flexible alternative to GPUs and CPUs since the architecture can be specialised to accelerate the particular precision needed by the algorithm.

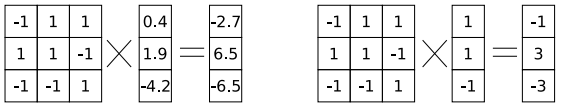
In the past, discrete FPGAs have been used to accelerate the algorithms in CNNs, but this limits the FPGA as it needs to handle all parts of the algorithm. This often means that the FPGA area which could be used to significantly accelerate a particular section of the algorithm needs to be partitioned to implement other functions that do not offer the same speedup. Recently, heterogeneous CPU+FPGA platforms have been proposed as an alternative to discrete. By coupling with a CPU, the FPGA's area can be freed to optimise only the most profitable parts of the application while the CPU handles functions that are less FPGA friendly. The advantage of the Xeon+FPGA is that the CPU is part of the standard x86 ecosystem and offers high performance computing. In contrast to other CPU+FPGA platforms, the Xeon CPU can take on significant workloads allowing the FPGA implementation to truly specialise and achieve close to its peak theoretical performance.

Binarised Neural Networks (BNNs) [2], [3], [7] have been proposed to address two challenges. First, the number of multiply-accumulates (MACs) needed by newer topologies is growing significantly and is thus becoming intractable without significant compute resources. Secondly, as the number of layers in the network increases, the storage for the weights and activations can quickly surpass the available system memory when stored as single precision floating point numbers. BNNs address these issues by first changing the representation from single precision floating point to a single bit, resulting in a 32x reduction in storage, and by fundamentally changing the math from MACs to XNORs and population counts.

This paper investigates high performance BNNs on Intel's Xeon+FPGA platform. As opposed to prior studies which aim to implement all parts of the algorithm on the FPGA, our implementation focuses on mapping only the most profitable portion of the algorithm onto the FPGA and relies on the high-performance Xeon CPU to handle the other parts of the algorithm. Given that state-the-art neural networks are trending to use a wider variety of operations, an all-FPGA implementation may not be optimal and potentially intractable since supporting the various operations on FPGA would add to the design and development complexity. As an example, a state-of-the-art BNN [7], which is studied in this paper, contains matrix multiply operations of various precisions among their operands (e.g., binary vs. binary, FP32 vs. binary, FP32 vs. FP32), as well as other operations (e.g., ReLU, normalization, scaling).

Specifically, the contributions of this work are as follows:

- The first heterogeneous implementation of BNNs on Xeon+FPGA.
- To our knowledge, the fastest implementation of a BNN accelerator on an FPGA to date.
- Evaluation on popular topologies used for ImageNet.



(a) Binary Weights and Real Activations (b) Binary Weights and Activations

Fig. 1: Two types of Binary Neural Network implementations: (a) binary weights and real activations (b) both binary weights and binary activations. We focus on (b) in this paper.

II. Background

A. Convolutional Neural Networks

Neural networks (NNs) are a class of machine learning algorithms that are described as a connected graph of basic compute nodes called neurons. The fundamental compute in each neuron is a dot product of the inputs, called activations, and a set of weights that is unique to that particular neuron. In addition to the dot product, an activation function (e.g., tanh, ReLU, sigmoid) is applied to the output of each neuron, thereby introducing non-linearity into the network. CNNs are a sub-class of NNs in which the main layer is the convolutional (CONV) layer. The CONV layer can be represented as a 3 dimensional block of neurons, often called a filter. The input into a convolutional layer is a collection of 2 dimensional input images (or channels) called input feature maps (IFMs), and these IFMs are fed into a four dimensional filter to produce a collection of 2 dimensional output images (or channels) called output feature maps (OFMs).

B. Binarised Neural Networks

BNNs are gaining significant traction in the CNN and FPGA communities [5], [6], [2], [3], [7], [10], [9]. BNNs provide two optimisations that are attractive to CNNs: (1) a smaller memory footprint compared to the single precision floating point and (2) the ability to replace the conventional multiply-accumulate with more area efficient implementations. Fig. 1 illustrates two types of binarisation: (a) binarised weights and full precision activations and (b) binarised weights and binarised activations. These are generally applied to the Fully Connected (FC) and CONV layers since they stand to benefit the most. Binarised weights and full precision activations generally reduce the storage requirements of the weight matrix by 32x and replace multiplication with a conditional negation. When performing the dot product within the neuron, the sign ($+$ for 1 and $-$ for 0) of the binarised weights is applied to the activations, and the results are accumulated as normal. In the case of binarised weights and binarised activations, both the weight and activation matrices are reduced down to a single bit representation, and the standard multiply-accumulates are replaced by XNOR and a signed bitcount.

C. Operation Types in Binarised Neural Networks

While BNNs allow a substantial number of matrix multiply operations to be converted to binary operations (XNOR and bitcount), there are still other types of operations that are needed by the algorithm. A state-of-the-art BNN [7] typically uses full precision (FP32) for the first and last layer of the network, in order to achieve acceptable accuracy. Prior FPGA accelerators propose modifying the algorithm (e.g., [9], [10]) to use fixed precision narrow data types that are more FPGA-friendly (i.e., does not require FP32 operations). However, their experiments target only smaller datasets (MNIST, CIFAR) instead of large scale ImageNet, where accuracy is often more sensitive to such data type changes.

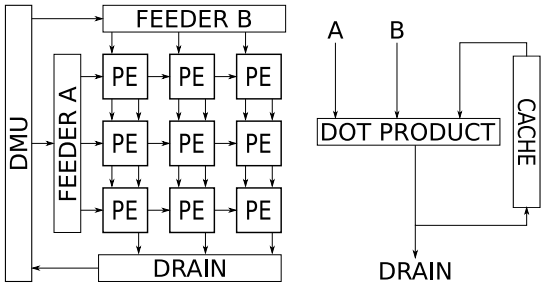


Fig. 2: Systolic Array and Processing Element (PE)

III. FPGA Architecture

As shown in Fig. 2, the architecture used in this paper is a systolic array with each PE constructed as a dot product engine. In the case of the binarised activations and weights, the dot product engine performs XNOR and bitcount operations, as in [6], [5]. Within the systolic array, input vectors fed into each PE are interleaved to take advantage of data reuse and to help meet the bandwidth requirement of the system. Due to the interleaving, a small cache within each PE is necessary to store the partial results for accumulation later in the computation. The design is fully pipelined so that each cycle new data is fed into the grid and the data from the previous cycle is propagated along the appropriate rows and columns. Along two boundaries of the systolic array are memory blocks that feed the grid with new data to process, which are called feeders. The feeders are double buffered to ensure that multiple read requests are in-flight and the maximum bandwidth of the system can be achieved. The data management unit (DMU) is responsible for requesting the input data, filling the feeders, draining out completed sections of the compute, and generating write requests to the system memory.

IV. Implementation

The architecture presented in Sec. III is implemented on the Intel Xeon+FPGA platform. The Xeon+FPGA system gives the FPGA access to the Xeon's cache memory and makes the system memory addressable via a high bandwidth and low latency interconnect. This capability allows for two unique opportunities. First, since there is a high performance CPU next to the FPGA, the architecture can be handcrafted to accelerate only the important parts of the algorithm where the FPGA can truly shine. Second, it is possible to perform fine-grained task partitioning and offloading since both the FPGA and CPU can work in the same address space.

The entire FPGA design is written in RTL to ensure maximum performance and correct mapping to the FPGA

resources. The FPGA in the Xeon+FPGA is an Arria 10 GX1150 with 427.2K ALMs (1.150M logic elements), 1518 hard DSP blocks and 2K M20K memory blocks. Synthesis, place and route were performed using the Quartus Prime Pro version 15.1.

A. Training and Mixed Precision

Training NNs on FPGAs has generally been avoided since the gradient update step during back-propagation poses additional difficulties to NN architectures that implement the entire algorithm on the FPGA. However, the Xeon+FPGA provides a unique opportunity to perform fine-grained partitioning over the two devices. Instead of sacrificing FPGA resources to handle the additional routing and configuration requirements, the CPU can be sequestered to perform these costly back-propagation steps. Each layer within the computation can be staged such that memory transfer overhead is avoided. Additionally, inefficiency usually introduced by maintaining different address spaces is circumvented since both the FPGA and CPU share the same address space.

B. GPU Implementation

The GPU implementation is based off the one presented in [3]. The population count instructions are supported in the GPU via the $\_\_popc()$ for 32bit and $\_\_popcll()$ for 64bit. A streaming multiprocessor (SM) in an Nvidia Titan X can issue 32 $\_\_popc()$ instructions every cycle, resulting in 1024 binary ops per cycle.

V. Results

This section presents the results for the FPGA BNN accelerator. First, we provide a baseline comparison of a binarised matrix-matrix multiplication against the GPU. We then evaluate the performance for two different networks (AlexNet [4] and VGGNet [8]) using the XNOR-Net scheme in [7].

A. Baseline

Performing a baseline analysis shows how close each implementation is to achieving the theoretical performance of the device. The operation used to perform the analysis is a matrix-matrix multiplication with 10240×10240×10240 (M×N×K) matrix parameters. Theoretically, the FPGA is capable of performing 131072 binary ops per cycle. Running at 312.5Mhz the peak FPGA performance is 40.96 tera-operations per second (TOPs). In comparison, each SM in the Titan X Pascal GP102 can perform 32 $\_\_popc()$ operations per cycle resulting in 1024 binary ops per cycle per SM. With 28 SMs and a core boost frequency of 1.531GHz, the peak GPU performance is $1024 * 28 * 1.531GHz = 43.89TOPs$.

The FPGA achieves near peak performance (40.77 TOPs) consuming 48 Watts with an efficiency of 99.5%. Similarly, The GPU achieves 93.43% (41.01 TOPs) efficiency consuming 70 Watts. The FPGA implementation shows that it is within 99.4% of the measured GPU performance. Additionally, the FPGA achieves 849.38 GOPs per Watt which is a 1.44x improvement in energy efficiency over the GPU (585.86 GOPs per Watt).

TABLE I: AlexNet Estimated Topology Performance

	CPU(I)	FPGA(I)	CPU(T)	FPGA(T)
IPS	607.8	1610.4	286.9	406.3
IPS/W	3.68	33.5	1.74	8.46

This shows results for both Inference (I) and Training (T)

B. Network Evaluation

For each network the evaluation was performed on the CONV and FC layers. The mini-batch sizes used were 256 and 32 for AlexNet and VGGNet respectively. Each measurement is based off the wall-time for performing that particular layer. Since the input formats to both the FPGA and GPU are identical for each layer, memory transformations incur a near-constant cost and have been factored out. For the topologies presented the FPGA remains competitive with the GPU for most layers and provides a sizable energy efficiency improvement. When extrapolating the FPGA results into network performance, each layers performance is weighted based on its contribution to the total number of networks OPs. The training and inference results for the FPGA consider the case where only the binary layers are implemented on the accelerator and all others are performed on the CPU. For both topologies the estimated images per second (IPS) and IPS per watt are provided to give a network performance overview. A runtime breakdown of each topology was collected using the Xeon Broadwell CPU running Caffe with Intel MKL2017. The CPU IPS is provided as baseline, and each layer is performed at full precision. The Xeon is a 14 core machine with 64 GB of main memory, and the power is assumed to be the thermal design power (TDP) of the socket.

1) AlexNet: Fig. 3 shows the performance of the binarised layers in AlexNet for FPGA and GPU. On average, the FPGA is within 83% of the GPU performance for the binarised layers and provides a 1.1x improvement in energy efficiency over the GPU. As illustrated in Tab. I, taking into account the distribution of the compute and the estimated IPS, the FPGA provides a 2.6x and 9.1x improvement in speed and energy efficiency over the CPU for inference. In the case of training, the FPGA provides a 1.4x improvement in IPS and a 4.8x improvement in IPS per Watt.

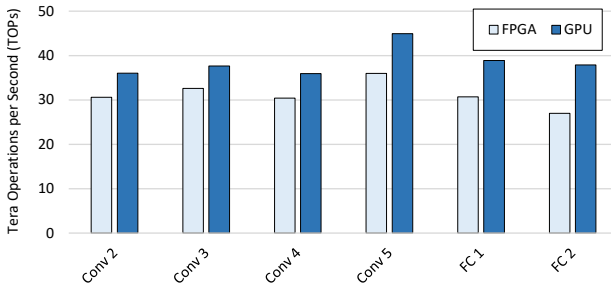


Fig. 3: AlexNet Layer Performance

2) VGGNet: For VGGNet, presented in Fig. 4, the FPGA is within 87% of the GPU performance on average. Note that for the majority of the middle layers, the FPGA remains within

TABLE II: VGGNet Estimated Topology Performance

	CPU(I)	FPGA(I)	CPU(T)	FPGA(T)
IPS	14.2	114.8	6.03	9.89
IPS/W	0.09	2.39	0.04	0.21

This shows results for both Inference (I) and Training (T)

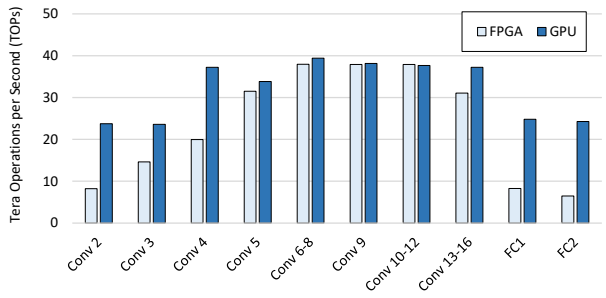


Fig. 4: VGGNet Layer Performance

99% of the GPUs performance. When power is considered, the FPGA provides a 1.05x improvement in energy efficiency over the GPU. As illustrated in Tab. II, the FPGA is 8.5x faster and 20x times more energy efficient over the CPU for inference. However in training the difference is less pronounced at 1.6x and 3.8x for IPS and IPS/W respectively, since the backward step accounts for 90% of the total computation time.

C. FPGA Comparison

Tab. III compares the current state-of-the-art FPGA performance for XNOR BNN accelerators. As shown in the table, the implementation presented in this paper reports the highest Peak TOPs of any FPGA implementation. Additionally, even though the device is significantly larger than some of the other designs, it still shows the second highest energy efficiency (i.e., GOPs/Watt). Note that other designs often change the fundamental algorithm to allow for scaling at fixed-point or low-precision data types. One of the key advantages of the Xeon+FPGA implementation is that with a Xeon CPU able to provide significant compute floating point operations, the underlying BNN algorithm does not need to be modified to suit the platform.

VI. Related Work

Given the growing importance of BNNs, there have been several recent studies on accelerating BNNs using FPGAs [5], [6], [10], [9]. Our work is different from these studies as follows. First, we focus on a Xeon+FPGA server platform, where we study synergistic collaboration between the high-end CPU and FPGA in the platform. On the other hand, prior work has focused on implementing the entire BNN on FPGAs, while we focus on implementing only the most profitable portion of the algorithm onto the FPGA and rely on the high-performance Xeon CPU to handle the rest. Second, we offer evaluation across two well-known state-of-the-art neural networks (AlexNet, VGG) for the ImageNet dataset, while prior studies either target fewer networks and/or smaller less complex image datasets (e.g., MNIST, CIFAR, SVHN). Third,

TABLE III: Comparison Against Previous FPGA Studies

	[5]	[10]	[9]	[6]	This Paper
Platform	Arria 10 GX1155	Zynq 7Z020	Zynq Z7045	GX1155	GX1155
Logic Elements	1,150K	85K	350K	1,150K	1,150K
Clock (MHz)	150	143	200	270	312.5
Power (W)	-	4.7	11.3	-	48
TOPs (Peak)	9.8	0.3189	11.612	25	40.77
GOPs / Watt	-	67.85	1027.68	-	849.38
Alg. Changed?	No	Yes	Yes	Yes	No

'-' indicates that the particular numbers were not provided.

we offer analyses of end-to-end usages in both inference and training, while all prior work focuses on inference.

VII. Conclusion

A heterogeneous BNN accelerator was presented and implemented on the Intel Xeon+FPGA. The evaluation of the architecture and implementation presented in this paper shows that the in-package integrated FPGA can remain competitive with a high-end discrete GPU on raw performance. By utilising the Xeon+FPGA a flexible usage model capable of performing mixed precision methods was presented and evaluated. By coupling the FPGA with a high performance Xeon CPU, architectures targeted at very specific compute, such as the one presented in this paper, can be augmented with the Xeon allowing for fine-grain sharing of the compute whilst still providing the flexibility to handle many different problems.

Acknowledgment

This research was partially supported under the Australian Research Councils Linkage Projects funding scheme (project number LP130101034).

References

[1] Utku Aydonat, Shane O'Connell, Davor Capalija, Andrew C Ling, and Gordon R Chiu. An OpenCL (TM) Deep Learning Accelerator on Arria 10. In *ISFPGA*, 2017.

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations. In *NIPS*, 2015.

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv:1602.02830*, 2016.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[5] Eriko Nurvitadhi, David Sheffield, Jaewoong Sim, Asit Mishra, Ganesh Venkatesh, and Debbie Marr. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. In *FPT*, 2016.

[6] Eriko Nurvitadhi, Ganesh Venkatesh, Jaewoong Sim, Debbie Marr, Randy Huang, Jason Ong Gee Hock, Yeong Tat Liew, Krishnan Srivatsan, Duncan Moss, Suchit Subhaschandra, and Guy Boudoukh. Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In *ISFPGA*, 2017.

[7] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 2016.

[8] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. *arXiv:1409.1556*, 2014.

[9] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In *ISFGPA*, 2017.

[10] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs. In *ISFPGA*, 2017.