How to Implement high performance power efficient Deep Learning on FPGAs

V. Rayapeta, D. Nandi
Microchip Technology Inc
Chandler, USA

R. Green, H. Richter
ASIC Design Services
Midrand, South Africa

Abstract – Field Programmable Gate Arrays (FPGAs) are a compelling choice for hardware acceleration on the edge especially when adding newer capabilities for machine learning inference. Specialized neural networks called Convolutional Neural Networks (CNN) are being deployed on the edge in embedded vision systems to perform tasks such as object detection, face and gesture recognition and pose estimation. FPGA architecture provides a unique set of features to satisfy the high computational complexity requirements along with sufficient memory access (via local and external memories) to realize CNNs efficiently. Their inherent programmability provides the flexibility to integrate and upgrade customized functions on a single device.

In this paper we share details about how Microchip's programmable hardware along with the Core Deep Learning (CDL) framework from ASIC Design Services enable a power efficient imaging and video solution platform for embedded and edge computing applications. The techniques include quantization of the CNN at 8-bit integer precision, neural network optimization based on the underlying FPGA architecture and the INT8 dot product mode of the Math block to efficiently deploy Microchip FPGAs for machine learning inference.

Keywords— Embedded Vision, Machine Learning, Edge Computing, Hardware accelerators

I. INTRODUCTION

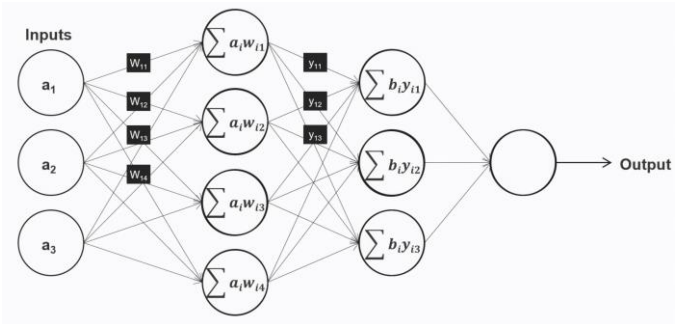
Breakthroughs in Deep Neural Networks (DNNs) have paved the way for realizing an increasing number of image and speech recognition applications. Improvements in the state of the art performance of network models have led to efficient architectures like MobileNetV2 that is tailored for resource constrained environments [1]. Image and video processing applications stand to benefit by including features such as object detection, face and gesture recognition and pose estimation into their existing pipelines.

For applications such as measuring wait times in stores and traffic patterns it would be desirable to use computer vision to extract information at the image sensor rather than in the cloud [2]. Other applications such as autonomous vehicles, drone navigation and robotics require local processing to minimize both process latency and security risk.

The evolution of neural network architectures seems certain to continue with end-to-end approaches for classical computer vision problems [3], stereo vision [4] and face reconstruction [5]. Deploying a programmable hardware platform ensures the flexibility to continuously provide algorithm upgrades and other feature customizations.

In this paper we share details about how Microchip's programmable hardware along with the Core Deep Learning (CDL) framework from ASIC Design Services enable a power efficient imaging and video solution platform for embedded and edge computing applications. The techniques include quantization of the CNN at 8-bit integer precision, neural network optimization based on the underlying FPGA architecture and the INT8 dot product mode of the Math block to efficiently deploy Microchip FPGAs for machine learning inference. Specifically, we can demonstrate an object classification application using the popular Tiny YOLO v2.0 algorithm running at 102GOPS/s/W at 8-bit integer precision.

II. INT8 DOT PRODUCT MODE IN MATH BLOCK



Inputs: a_i Weights: w_{ii}, y_{ii}

Fig. 1. Fully Connected Neural Network Example

The dot product operation lies at the heart of the CNN [6]. These dot products are usually implemented using multiply-accumulate units with several degrees of parallelism. The basic computation for both the fully connected and convolutional layers (Figure. 1) is of the form:

$$o_j = f\left(\sum_i a_i w_{i,j}\right),\ i \in [1,n] \quad (1)$$

where $f(x)$ is a non-linear activation function like ReLU.

The equation in expanded form is:

$$o_j = f(a_1 * w_{11} + a_2 * w_{21} + a_3 * w_{31} + \cdots) \quad (2)$$

Traditionally, the compute and memory intensive nature of CNNs have limited their deployment on embedded systems. Multiple approaches have recently been developed to successfully prune and quantize the network [7, 8].

Microchip FPGAs (SmartFusion2, RTG4, PolarFire) have a dot product (DOTP) mode that is especially suited for 8-bit arithmetic operations (Figure. 2).

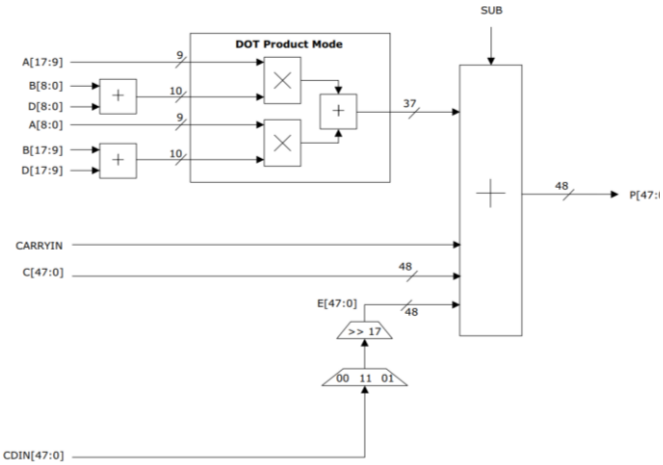


Fig. 2. DOTP Mode available in PolarFire FPGA's Math block.

In the DOTP mode, a single math block can accommodate four useful operations:

- DOT Product Mode block can perform two multiplies and one add:

$$a_1 * w_{11} + a_2 * w_{21}$$

- The accumulator block can perform one add needed to accumulate the result from the previous math block in cascade.

Unlike the Math block in Microchip FPGAs, alternative FPGAs do not have a DOTP mode for INT8 arithmetic. Due to this limitation, each Digital Signal Processing (DSP) block can only accommodate two useful operations (one multiply and one add).

To alleviate this inefficiency, a weight sharing architecture has been proposed that computes the DOTP for two different inputs using the same weights. However, since the computation is for two different inputs, the results need to be separated in the accumulator. This limits the length of the DSP cascade before the upper and lower words result become unrecoverable. An additional DSP must be used to handle the separation and summing of the lower and upper words after each cascade. This additional DSP reduces the effective useful operations per DSP. Also, additional control logic needed to shift input data and unpack results leads to a complex data flow with additional limitations and reduced operational efficiency.

The presence of the in-built DOTP mode in Microchip FPGAs enables higher efficiency for INT8 computations used by the fully connected and convolutional layers (Table I). Microchip FPGAs can deliver 100% higher efficiency compared to competing FPGA and 14% higher efficiency compared to competing FPGA with weight sharing.

TABLE I. MICROCHIP VS COMPETING FPGA COMPARISON

Family	INT8 Operations per DSP / Math Block
Microchip FPGA[a]	4
Alternative FPGA	2
Alternative FPGA (Weight Sharing)	Up to 3.5[b]

[a.] PolarFire, SmartFusion2, and RTG4.

[b.] Requires overhead logic to shift input data and unpack results

III. CORE DEEP LEARNING FRAMEWORK

The Core Deep Learning (CDL) framework was developed to accelerate CNNs on Microchip FPGAs. In addition to utilizing the architectural efficiency of the INT8 DOTP mode, the design of the framework attempts to achieve the following goals:

1. Identify and utilize the opportunity to parallelize computations in a CNN (kernel, input and output feature maps).
2. Use data quantization methods to reduce compute and memory requirements while preserving the network accuracy.
3. Optimize the implementation considering the targeted FPGA architecture and specific the CNN being deployed. This allows the framework to be both scalable and generative.

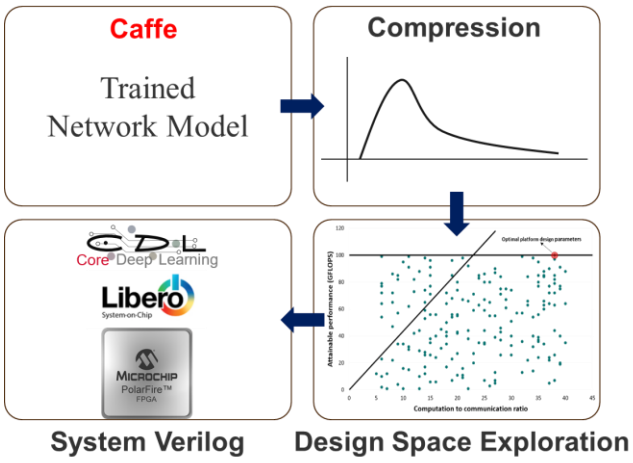


Fig. 3. Core Deep Learning Framework for Microchip FPGAs.

The framework currently supports the standard Caffe CNN description model. The input model along with some training data is used to quantize the network from 32-bit floating to 8-bit dynamic fixed-point. The design space exploration phase determines the optimal design parameters for the specific network and FPGA. The output from the design space exploration configures the synthesizable System Verilog core.

Network models use 32-bit floating point implementations during training and inference. Inference when running in servers use GPUs or massively parallel FPGA arrays. However, on-the-edge devices are constrained by space, compute capability and power. Compressing the network from 32-bit floating point to 8-bit dynamic fixed point is a key step in ensuring the models can be implemented on FPGAs. Moving from a 32-bit implementation to an 8-bit implementation is done without much loss in accuracy of the model. In fact, several studies have shown that the loss in accuracy in the model to be less than 1% [9, 10] (Table II). Similar low accuracy loss is observed while using the CDL framework. This phase yields a quantized trained network that is bit-accurate to an FPGA design.

TABLE II: ACCURACY ON VARIETY OF DATASETS

	Gysel, 2016		Guo et al, 2016		CDL	
Network	32b FP[a]	8bit Int[b]	32b FP[a]	8bit Int[b]	32b FP[a]	8bit Int[b]
LeNet	99.1%	99.1%			99.12%	99.13%
CIFAR-10	81.7%	81.4%				
CaffeNet	56.9%	56.0%	77.12%	76.64%		
VGG16			88.10%	87.60%	88.84%	87.54%
GoogLeNet			88.82%	88.64%		
SqueezeNet			79.92%	79.16%		
Scene Labelling					73.89%	73.31%

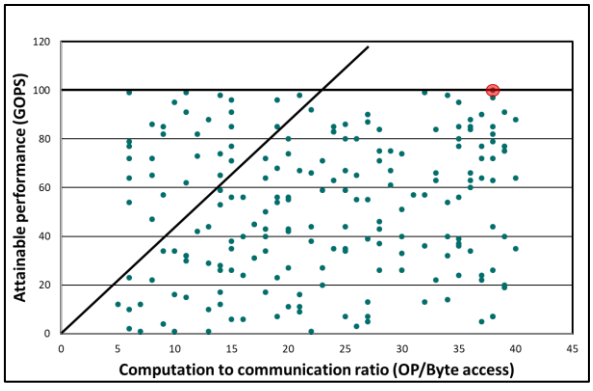
a: Floating Point, b: Integer

Inference on an FPGA is constrained by the number of math blocks, logic elements, available memory bandwidth and the fabric speed. In other words, the implementation can be either computation bounded or memory bounded.

$$Attainable\ perf. = \min\{Computational\ roof\ CTC\ ration\ x\ BW\}$$

$$Computational\ roof = \frac{\#\ of\ operations}{\#\ of\ execution\ cycles}$$

$$CTC = \frac{\#\ of\ operations}{\#\ of\ external\ data\ access}$$

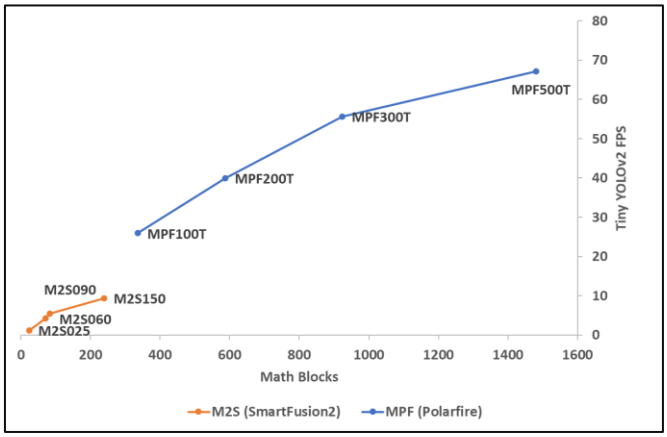


Red Dot: Optimal platform design parameter for performance

Fig. 4. CDL Design Space Exploration

Design Space Exploration enables tweaking inputs to the model and user constraints like availability of logic elements, DDR bandwidth and fabric frequency that translates to real world performance targets like frames per second performance, accuracy and power consumption. It also enables scalable designs spanning different FPGA sizes that share similar fabric architectures (Figure 5).

The Design Space Exploration phase yields accurate performance and usage report together with a network and platform specific optimized SystemVerilog CDL core memory map. The SystemVerilog CDL FPGA core interface is imported to the Libero design environment and allows an easy integration of a trained, quantized and optimized machine learning inference core.



M2Sxxx: SmartFusion2, MPFxxxT: PolarFire, xxx- LUT4 in thousands

Fig 5: TinyYolov2 fps chart with SmartFusion2 and PolarFire FPGAs

IV. EVALUATING FRAMEWORK PERFORMANCE (POWER EFFICIENCY)

YOLO (You Only Look Once) is a single pass through neural network where an input image is processed to simultaneously predict multiple bounding boxes and their class probabilities [11]. Tiny YOLO is a reduced network based on YOLO that detects 20 different object classes compared to more than thousands object classes supported by YOLO.

Parameters	Value
Input Image Shape	416 x 416
Number of Convolutional Layers	9
GOPs (MULACC)	7
Performance	43.6 fps
GOPs	304
Power Consumption (Core)	2.98W
GOPs/W	102

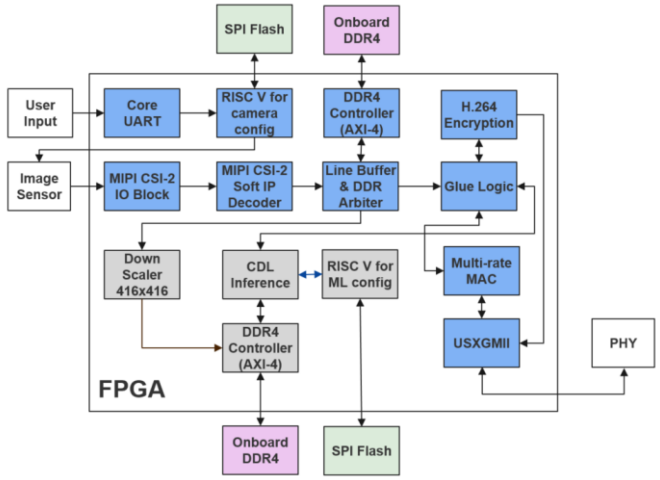
Fig. 6: Tiny YOLOv2 benchmark demo on PolarFire MPF300

Tiny YOLO v2.0 when implemented using CDL into PolarFire FPGA, with 300K logic elements and 924 (18x18) math blocks, processes images in real-time at 43.5 frames per second while

consuming 2.98W of power, thereby offering a 102GOPs/W performance (Fig. 6).

V. IMAGE PROCESSING PIPELINE

FPGAs bring unmatched value for machine learning inference in edge-devices. They have the requisite compute power (DSP math blocks) compared to MCUs, MPUs and CPUs, and low power consumption compared to GPUs. FPGAs integrate functions like video, communication and security, and allow firmware upgrade of the system that includes newer trained network models.



Blue: Video and Imaging pipeline, Grey: CDL inference pipeline

Fig. 8. Integrating Video and Machine Learning on-chip

An embedded designer can upgrade existing video signal chain to add machine learning by simply upgrading to a device with more logic elements and adding the Core CDL framework interface.

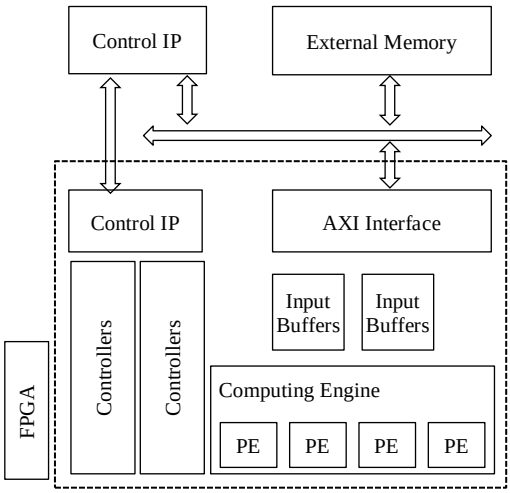


Fig. 7: CDL FPGA Core Interface

In a networking camera example (Fig. 8), the FPGA hardware interfaces an image sensor using MIPI-CSI2, runs the CDL inference engine and uses a USXGMII multi-rate PHY to transport a H.264 compressed video as well as intelligence pertaining to object detection and classification (Table 3).

TABLE III: RESOURCE UTILIZATION

IP	LUT4s (K)	DFFs (K)	DSP (18x18)
MIPI-CSI2	1.3	1.6	0
DDR4 (Imaging)	24.1	19	0
TinyYolov2 @40fps	60	77	512
DDR4 (Inference)	17	15	0
H.264 Encode	90	90	59
USXGMII	15.5	18	0
Dual RISC V	27.2	14	4
Total	235.1	234.6	575
MPF300	300	300	924
Utilization	78.4%	78.2%	62.2%

Compressing video reduces the bandwidth requirement on the Ethernet network and enables moving intelligence from the gateway or cloud towards the edge. Inference on the edge eliminates the need to send high quality uncompressed video upstream, while reducing latency in decision making. This also increases the capacity of an existing network allowing more cameras to be supported.

VI. BENCHMARK DATA

The Core Deep Learning FPGA framework has been developed on Microchip's PolarFire FPGA that offer high-efficiency INT8 Dot Product Mode math blocks. The CDL framework supports multiple frameworks like Convolutional, Pooling, Depthwise Separable Convolutional, Fully Connected, Non-Linear Activation, Concatenation and Residual layers etc. and has been tested on various networks supported by the Caffe framework (Table IV).

TABLE IV: CDL BENCHMARKS ON POLARFIRE FPGA

Network	**Lenet**	**Squeezenet**	**Pose**	**VGG**	**VGG (SVD)**	**Tiny-Yolo**
Network GOPs	0.005	0.782	3.578	30.941	30.764	6.970
Fabric 4LUT	59469	78268	73000	85531	68484	59991
Fabric DFF	80968	84048	84782	96103	77554	76674
uSRAM Blocks (64x12)	1440	1024	1024	1008	1024	1024
LSRAM Blocks (20k bit)	126	578	578	550	434	306
Total RAM (Mbits)	3.52	12.04	12.04	11.48	9.23	6.73
Math Blocks (18x18 MACC)	720	512	512	504	512	512
FPS	2747.25	155.02	23.14	5.67	8.51	39.47
Power (mW)	3653.53	4522.60	4422.43	4181.42	3803.46	3456.16
Performance (GOPs/s)	12.60	121.19	82.79	175.43	261.80	275.11
Power Efficiency (GOPs/s/W)	3.45	26.80	18.72	41.96	68.83	79.60
DSP Efficiency (GOPs/s/DSP)	0.02	0.24	0.16	0.35	0.51	0.54

VII. CONCLUSION

In this paper we have shown how FPGAs offer a suitable architecture for running Inference algorithms in edge devices. FPGAs offer high compute capabilities to devices that are power and space constrained and need low-latency in decision making. They are suitable for designing platforms, offering the flexibility to add or upgrade new features, including security functions. The Core Deep Learning framework along with Microchip's highly efficient INT8 dot product mode FPGAs makes it very easy for embedded designers to add optimized machine learning core for a single-chip design.

REFERENCES

[1] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[2] V. Sze, Y.H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in IEEE Custom Integrated Circuits Conference (CICC), 2017.

[3] C. Chen, Q. Chen, J. Xu, and V. Koltun, "Learning to see in the dark," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[4] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in International Conference on Computer Vision (ICCV), 2017.

[5] P. Dou, S.K. Shah, and I.A. Kakadiaris, "End-to-end 3D face reconstruction with deep neural networks," in Computer Vision and Pattern Recognition (CVPR), 2017.

[6] M. Vestias, R. Duarte, J.T. de Sousa, and H. Neto, "Parallel Dot-Products for Deep Learning on FPGA," in 27th International Conference on Field Programmable Logic and Applications (FPL), 2017.

[7] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-Oriented Approximation of Convolutional Neural Networks," in ICLR Workshop, 2016.

[8] S. Han, H. Mao, and W.J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in International Conference on Learning Representations (ICLR), 2016.

[9] Philipp Gysel. Ristretto: Hardware-oriented approximation of convolutional neural networks. arXiv:1605.06402, 2016

[10] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In Advances In Neural Information Processing Systems, pp. 1379–1387, 2016.

[11] Joseph Redmon, Santosh Divala, Ross Girshick, Ali Farhad. You Only Look Once: Unified, Real-Time Obect Detection.arXiv: 1506,0264v5