Accelerating Tiny YOLOv3 Using FPGA-Based Hardware/Software Co-Design

Afzal Ahmad, Muhammad Adeel Pasha, and Ghulam Jilani Raza

Department of Electrical Engineering,
Lahore University of Management Sciences (LUMS), Pakistan

2020 IEEE International Symposium on Circuits & Systems
Virtual, October 10-21, 2020

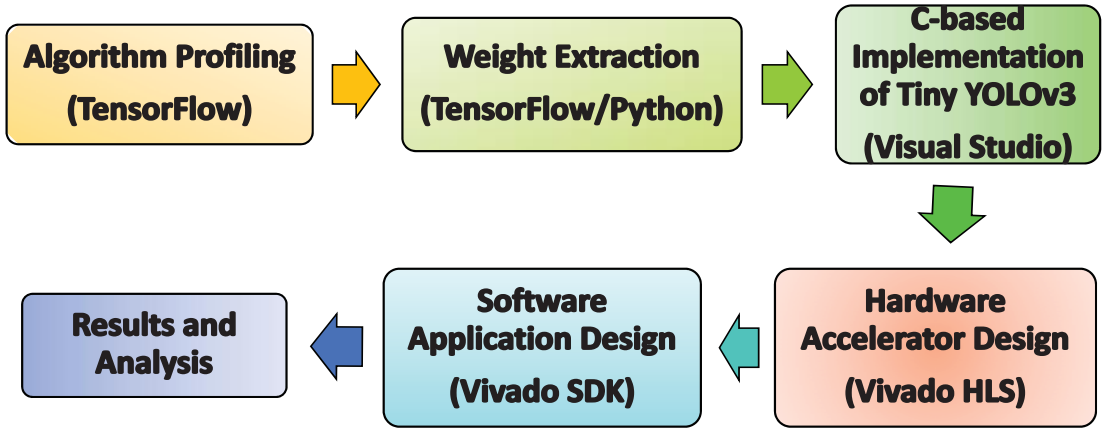
Agenda

- Convolutional Neural Networks (CNNs)
- Design Flow
- Profiling of Tiny-YOLO v3
- Hardware Accelerator Design and Analysis
- System Level Design
- Performance and Energy Results

Convolutional Neural Networks (ConvNets)

- ConvNets are state-of-the-art algorithms in several machine vision tasks.
- Employ billions of Multiply-Accumulate (MACC) operations.
- Graphical Processing Units (GPUs) are considered as the standard platform for their deployment.
- Mobile vision applications require reduced computational complexity, hardware portability, and energy efficiency.
- Specialized hardware for convnets to meet real-time and energy constraints for embedded vision applications.
- Tiny-YOLOv3 is an efficient and lightweight convnet for object detection.

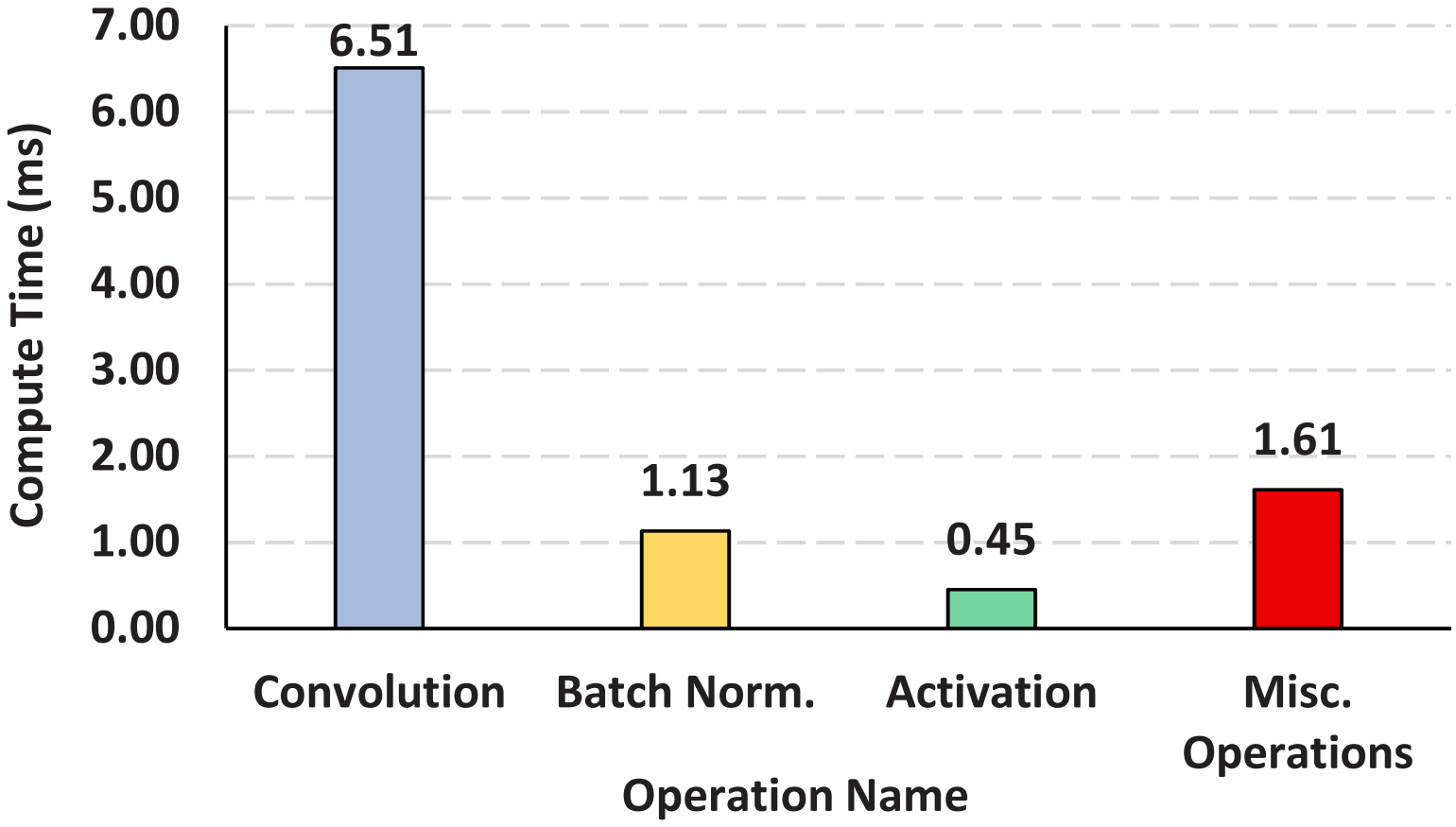
Design Flow



Flowchart highlighting Design Flow Steps for our accelerator

- Design of our system follows a conventional HW/SW partitioning design-flow.
- Profiling enables quantification of computation workloads.
- A C-based implementation running on the PS is used as integrity verification of the algorithm.
- Computationally complex portions of the C-based implementation are then moved to the hardware accelerator block.
- Glue logic and data movement command and control are adjusted into the PS and PL codes when adding the accelerator.

Profiling of Tiny YOLOv3



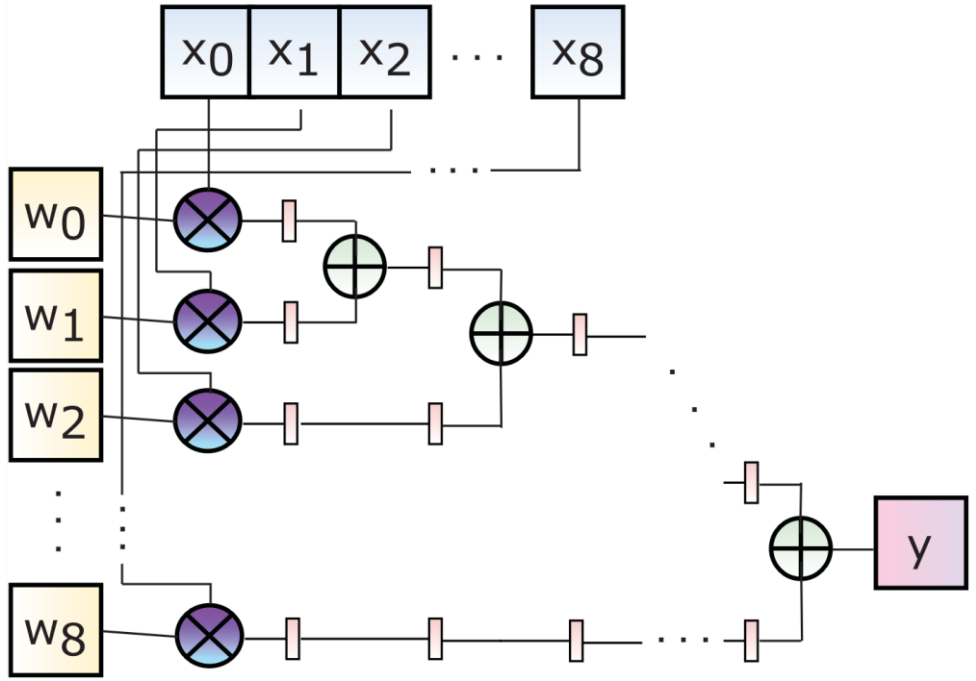
Profiling Results of Tiny YOLOv3 on a Tesla K40m GPU

Hardware Accelerator Design

- We design HW block to offload the convolution and batch-norm operations.
- The architecture follows a bottom-up design approach where a simpler processing element (PE) is initially designed, called PE_t.
- Many simper PEs are utilized to design another more complex PE (PE_v) for processing a much larger quantity of data.
- The top-level module utilizes many PE_v units in parallel to make a larger hardware accelerator block.

Hardware Accelerator Design Contd.

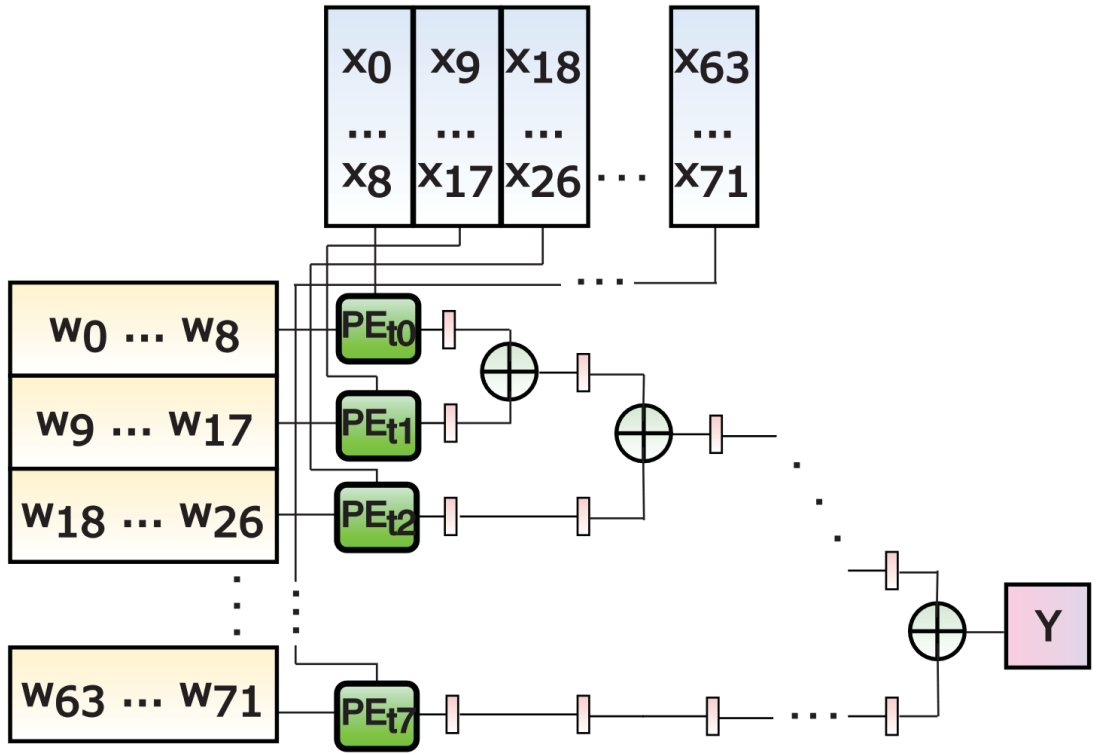
- PE_t, a tile processing element, performs MACC of a 3×3 feature map tile with a kernel map tile.



PE_t, Tile Processing Element for computing MACC of 3×3 elements

Hardware Accelerator Design Contd.

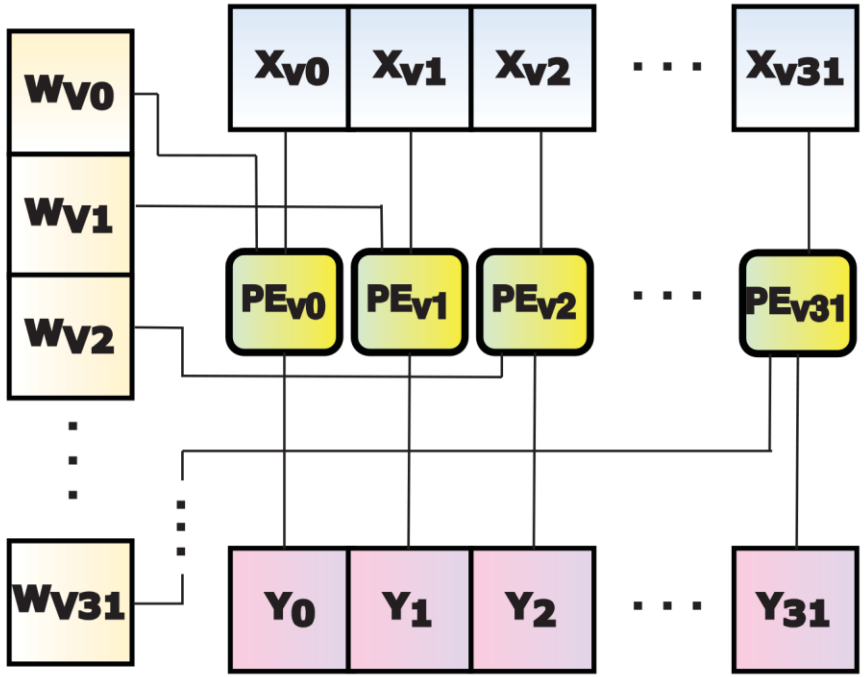
- A PE_v performs MACC of a $3 \times 3 \times 8$ volume of feature map with a kernel map.



PE_v, Volume Processing Element for computing MACC of $3 \times 3 \times 8$ volume

Hardware Accelerator Design Contd.

- Top-level module utilizes 32 PE_v units in parallel to compute convolution outputs of 32 volumes of size $3 \times 3 \times 8$.



Top-level module instantiating 32 PE_v modules

Complexity Analysis

- Time for convolution between a feature map of size $H \times W \times C$ and kernel map of size $3 \times 3 \times C \times K$ given by

$$t_s = \left(\left\lceil \frac{H - 2P}{3s} \right\rceil \left\lceil \frac{W - 2P}{3s} \right\rceil \left\lceil \frac{C}{8} \right\rceil \left\lceil \frac{K}{32} \right\rceil + D_t + D_v - 1 \right) t_c$$

- Multiplication complexity given by

$$O_c = \left(\left\lfloor \frac{H + 2P - k}{s} \right\rfloor + 1 \right) \left(\left\lfloor \frac{W + 2P - k}{s} \right\rfloor + 1 \right) k^2 CK$$

Elimination of Batch Normalization

- Batch Normalization (BN) is the second most complex operation in Tiny-YOLOv3 that is also composed of MACC.
- We utilize an optimization that allows fusing BatchNorm operation into the preceding convolution layer.

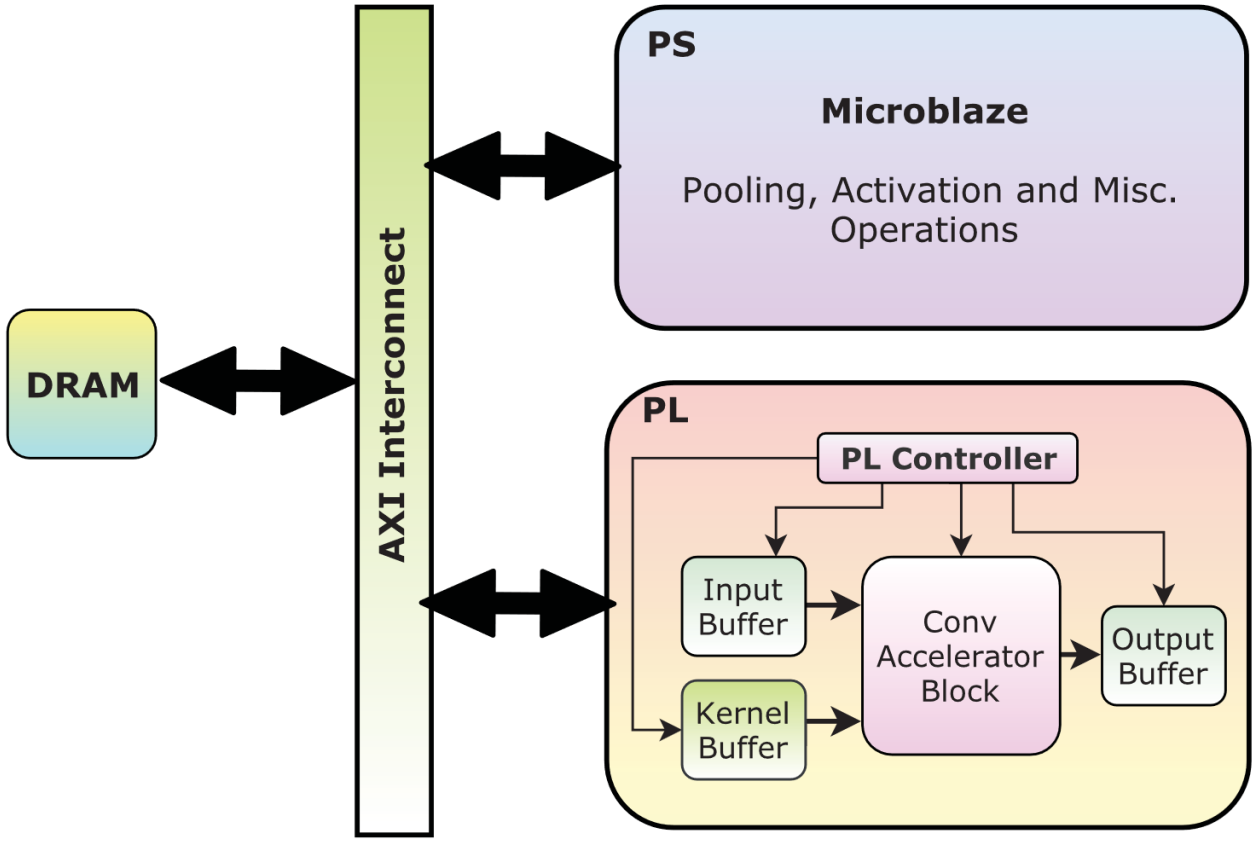
$$BN(x_i) = \omega x_i + \tau$$

$$Conv(X) = W \odot X - b$$

$$Y_{BN+Conv} = BN\big(Conv(X)\big) = W_{new} \odot X - b_{new}$$

$$W_{new} = W \times \omega \text{ and } b_{new} = b\omega - \tau$$

System Level Design



Block Diagram of Complete System Showing PS and PL portions and their communication interfaces.

Performance Comparison

	[1]	[2]	[3]	This Work
Device	Stratix-V	Cyclone-V	Virtex-7 480t	Virtex-7 VC707
Network	AlexNet/VGG	Tiny YOLOv2	Tiny YOLOv1	Tiny YOLOv3
Design Tool	OpenCL	OpenCL	Vivado	Vivado
Design Scheme	HW	HW	HW	HW/SW
Precision (bits)	16	16	16	18
Frequency (MHz)	120	117	143	200
Throughput (GOPS/s)	117.8	21.6	48.0	460.8
DSP Utilization	246	122	112	2304
DSP Efficiency (GOPS/s/DSP)	0.29	0.18	0.43	0.20

Resource Utilization

	DSPs	FF	LUTS	BRAM18K	Power (W)
Utilized	2304	93225	48583	141	4.81 W
Available	3600	607200	303600	2060	-
Percentage	82.3 %	15.4 %	16.0 %	6.8 %	-

References

[1] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J. -s. Seo, and Y. Cao, “Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks,” in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2016, pp. 16–25.

[2] Y. J. Wai, Z. bin Mohd Yussof, S. I. bin Salim, and L. K. Chuan, “Fixed Point Implementation of Tiny-YOLOv2 using OpenCL on FPGA,” International Journal of Advanced Computer Science and Applications, vol. 9, no. 10, 2018. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2018.091062

[3] J. Ma, L. Chen, and Z. Gao, “Hardware Implementation and Optimization of Tiny-YOLO Network,” in Digital TV and Wireless Multimedia Communication, G. Zhai, J. Zhou, and X. Yang, Eds.Singapore:Springer Singapore, 2018, pp. 224–234.

Thank You