Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks

Naveen Suda, Vikas Chandra*, Ganesh Dasika*, Abinash Mohanty, Yufei Ma, Sarma Vrudhula, Jae-sun Seo, Yu Cao.
Arizona State University. *ARM Research.

Feb 22, 2016

Outline

- **Introduction**
  - Convolutional Neural Networks (CNNs)
  - Challenges in hardware implementation
- **End-to-end FPGA accelerator for CNNs**
  - Fixed-point operations
  - Parameterized CNN layers in OpenCL
- **Optimization framework**
  - Performance and resource utilization models
  - Results: AlexNet and VGG-16 CNN models
- **Conclusion**

Convolutional Neural Networks (CNN)

- Feed-forward neural networks inspired from visual cortex
- Multi-layer feature extraction and classification
- **Applications**
  - Image/video classification, face detection, gesture detection
  - Autonomous driving, speech recognition
- **Tools:** Caffe, Torch, Theano, TensorFlow, CNTK, …

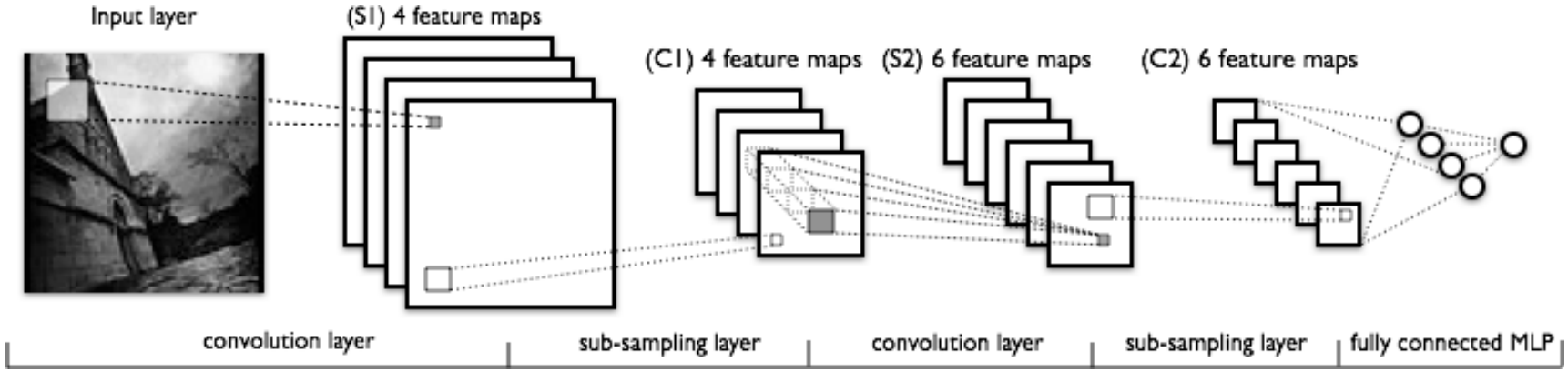
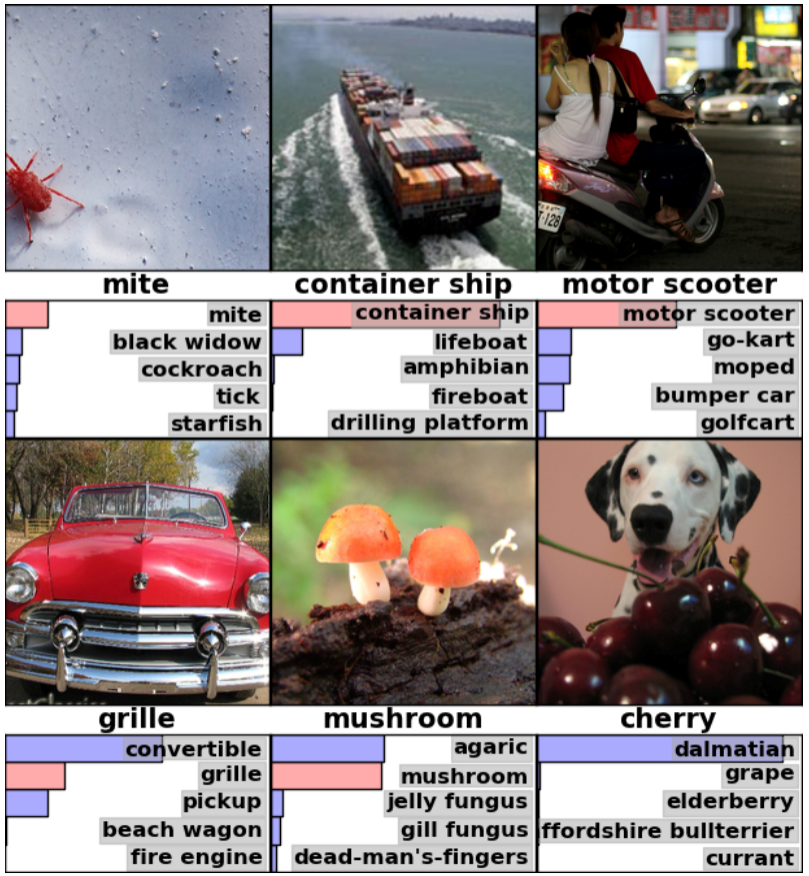


Image from: http://deeplearning.net/tutorial/lenet.html

CNNs for Image Classification

- **ImageNet Challenge**
  - 1000 output classes.
  - Dataset: 1.2M training + 50k validation + 100k test images
  - CNNs: winners since 2012
- AlexNet [1] CNN: 2012
  - 8 layers, 1.46 GOP/image
  - Top-5 Accuracy : **84.7%**
- VGG [2] CNN: 2014
  - 16-19 layers, 30.9 GOP/image
  - Top-5 Accuracy : **92.6%**



[1] A. Krizhevsky, et al. ImageNet classification with deep convolutional neural networks. *NIPS* 2012
[2] K. Simonyan, et al. Very deep convolutional networks for large-scale image recognition. *ICLR* 2015.

CNN Accelerators

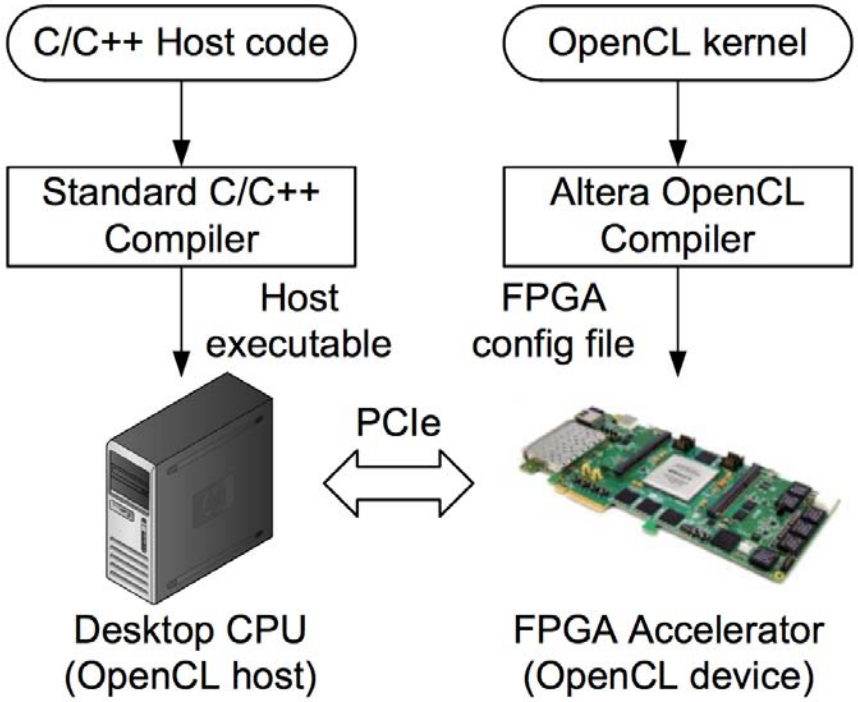
- GPUs (fast) - high power consumption (~200W)
- ASIC (energy efficient) - less flexibility
- FPGA / SoC (FPGA + embedded CPU)
  - Generic CNN accelerators: optimize compute or memory [1]-[3]
  - Balance compute + memory access: AlexNet convolution layers [4]
- **Contribution:** End-to-end large-scale CNN accelerator on FPGA
  - Design space exploration to maximize throughput

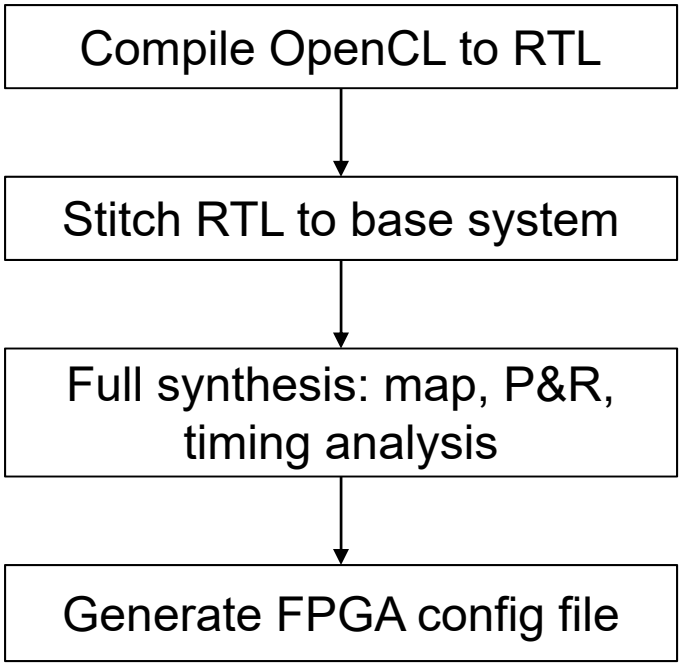
[1] Farabet, et al. Hardware accelerated convolutional neural networks for synthetic vision systems. *ISCAS* 2010
[2] M. Peemen, et al. Memory-centric accelerator design for convolutional neural networks. *ICCD* 2013
[3] V. Gokhale, et al. A 240 G-ops/s mobile coprocessor for deep neural networks. *CVPR Workshops,* 2014.
[4] C. Zhang, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks. ISFPGA 2015

Challenges in FPGA Implementation

- **Large number of computations**
  - AlexNet CNN (2012): 1.46 GOP/image
  - VGG-16 CNN (2014): 30.9 GOP/image
  - **Solution:** Parallel computational resources
- **Limited FPGA resources**
  - No. of CNN layers: 8 – 152 with different dimensions
  - **Solution:** Run-time scalable hardware modules
- **Huge models**
  - AlexNet: ~240MB: VGG-16: ~550MB
  - **Solution:** Reduced precision weights & data reuse

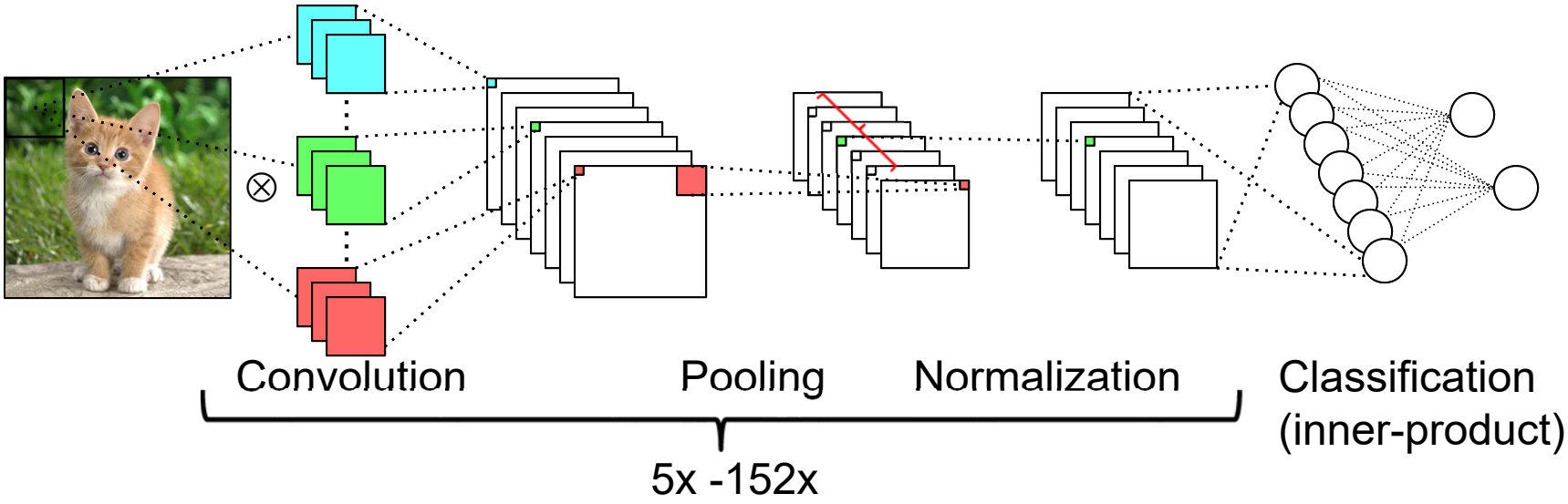
OpenCL Framework





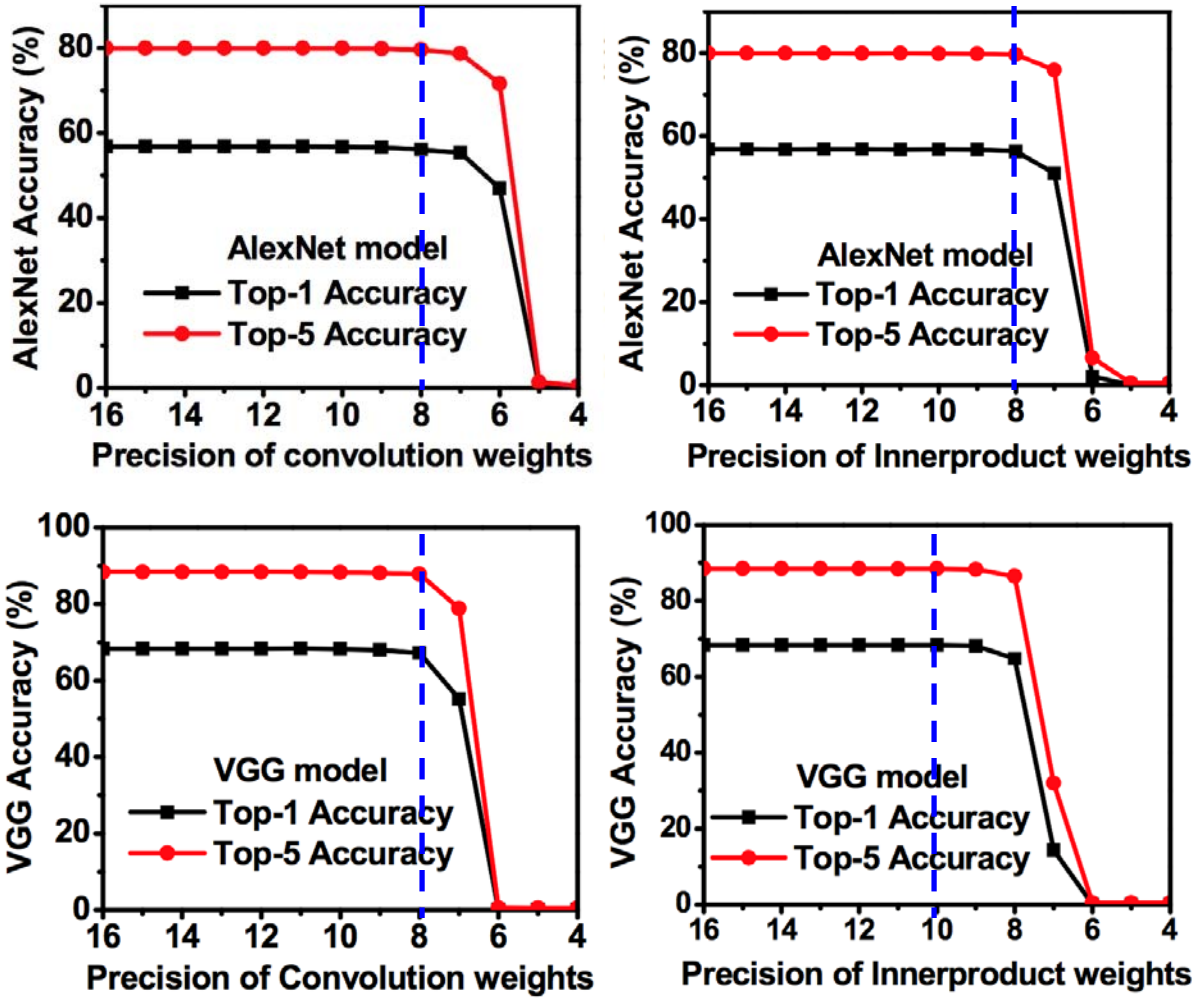
- Features enabling acceleration: loop unrolling, SIMD factor, compute unit replication, local memory usage.

CNN Operations



- Convolution: 3-D multiply and accumulate operations
- Pooling: max/average of a window
- Normalization: normalize based on neighboring neurons
- Classification: multiply and accumulate operations
- Activation function: y=max(x,0)

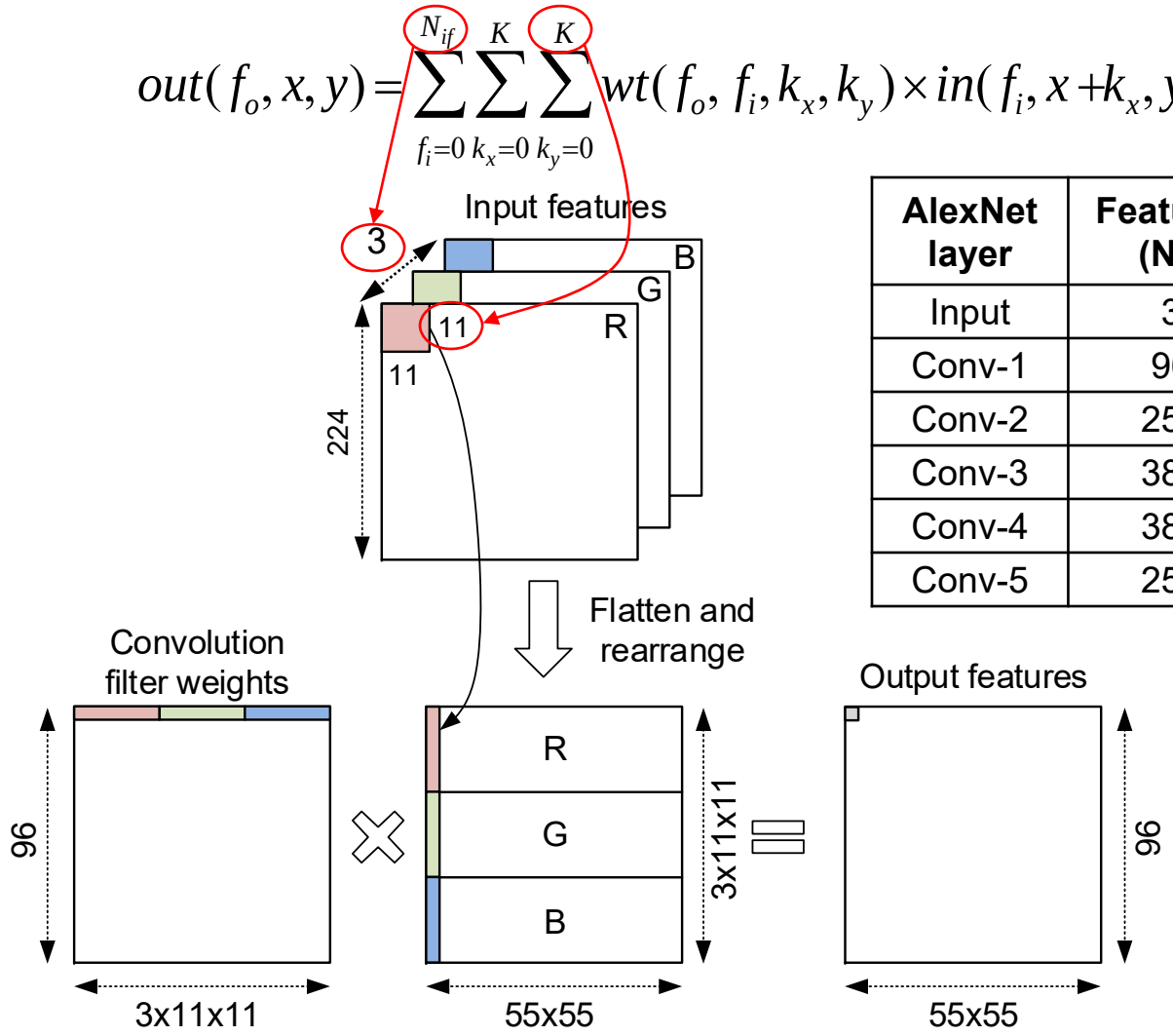
Accuracy-Precision Study



- Intermediate layer data (Neurons) : 16-bit precision

3-D Convolution as Matrix Mult.

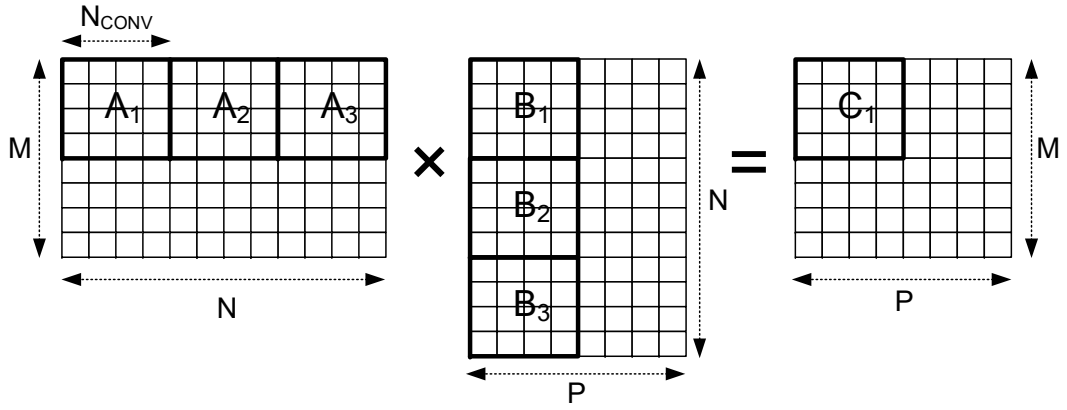
$$out(f_o, x, y) = \sum_{f_i=0}^{N_{if}} \sum_{k_x=0}^{K} \sum_{k_y=0}^{K} wt(f_o, f_i, k_x, k_y) \times in(f_i, x+k_x, y+k_y)$$



AlexNet layer	Features (N_{if})	Size	Kernel (K)	Stride
Input	3	224x224		
Conv-1	96	55x55	11x11	4
Conv-2	256	27x27	5x5	1
Conv-3	384	13x13	3x3	1
Conv-4	384	13x13	3x3	1
Conv-5	256	13x13	3x3	1

K. Chellapilla, et al. High performance convolutional neural networks for document processing. IWFHR 2006

Accelerating Convolution in OpenCL



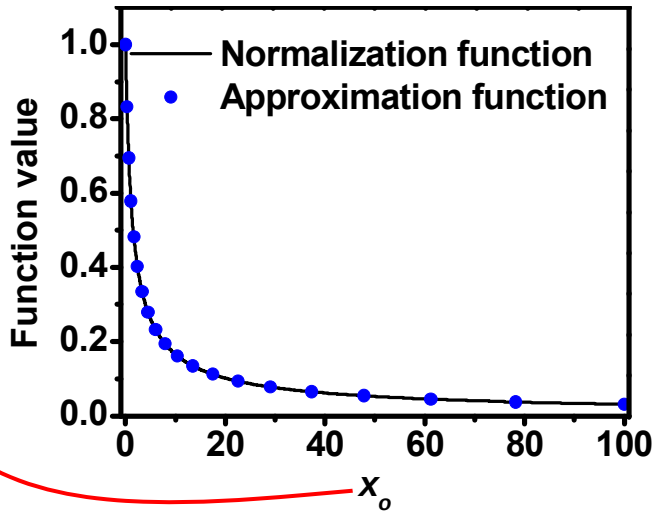
$C_1 = A_1 \times B_1 + A_2 \times B_2 + A_3 \times B_3$

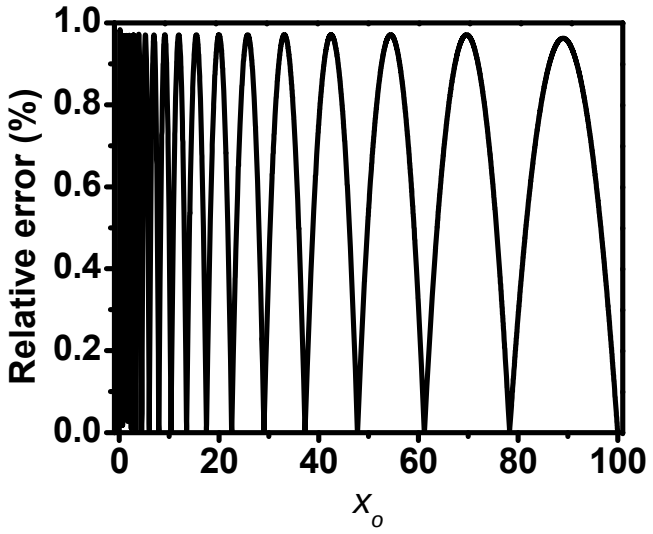
- **N_{conv}** MAC operations in parallel: (loop unroll)
- Different feature dimensions: zero-pad to multiple of **N_{conv}**
- SIMD vectorization factor: **S_{conv}**
- Acceleration factor: **N_{conv}*S_{conv}**
- Design knobs for acceleration: (**N_{conv}**, **S_{conv}**)

Normalization – PWL Approximation

$$out(f_o, x, y) = \frac{in(f_o, x, y)}{\left(1 + \frac{\alpha}{K} \sum_{f_i = f_o - K/2}^{f_o + K/2} in^2(f_i, x, y)\right)^{\beta}}$$

- Resource expensive exponent operation
- Piece-wise linear approximation
- Floating point implementation
  - Sum of squares operation only
- Design variable for acceleration: **N_{NORM}** (loop unroll factor)





Pooling & Inner Product

- Pooling:

$$out(f_o, x, y) = \max/\text{average}_{0 \le (k_x, k_y) < K} \left(in(f_o, x + k_x, y + k_y) \right)$$

  - Design variable for acceleration: **N_{POOL}** (loop unroll factor)

- Classification or inner-product or fully-connected layer

$$out(f_o) = \sum_{f_i = 0}^{N_{if}} wt(f_o, f_i) \times in(f_i)$$

  - Design variable for acceleration: **N_{FC}** (loop unroll factor)
  - Limited by external memory bandwidth

Optimization Framework

Target

- Find optimal values for (N_{CONV},S_{CONV},N_{NORM},N_{POOL},N_{FC}) that maximize throughput of FPGA accelerator.

Problem Formulation

$$\text{Minimize} \sum_{i=0}^{TL} runtime_i(N_{CONV}, S_{CONV}, N_{NORM}, N_{POOL}, N_{FC}) : \textbf{Performance Model}$$

$$\text{Subject to} \sum_{j=0}^{L} DSP_j \le DSP_{MAX}$$

$$\sum_{j=0}^{L} Memory_j \le Memory_{MAX}$$

$$\sum_{j=0}^{L} Logic_j \le Logic_{MAX}$$

Resource Utilization Model (FPGA - specific)

Performance Modeling

- Analytical models for performance

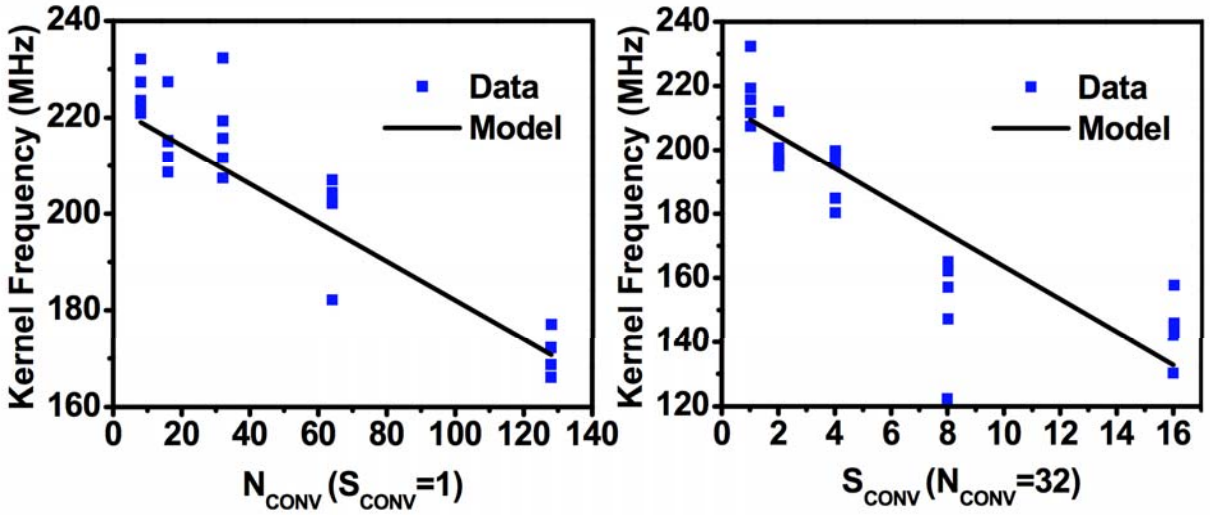
$$Convolution\ Execution\ time_i = \frac{No.\ of\ Convolution\ Ops_i}{N_{CONV} \times S_{CONV} \times Frequency}$$

$$No.\ of\ Convolution\ Ops_i$$

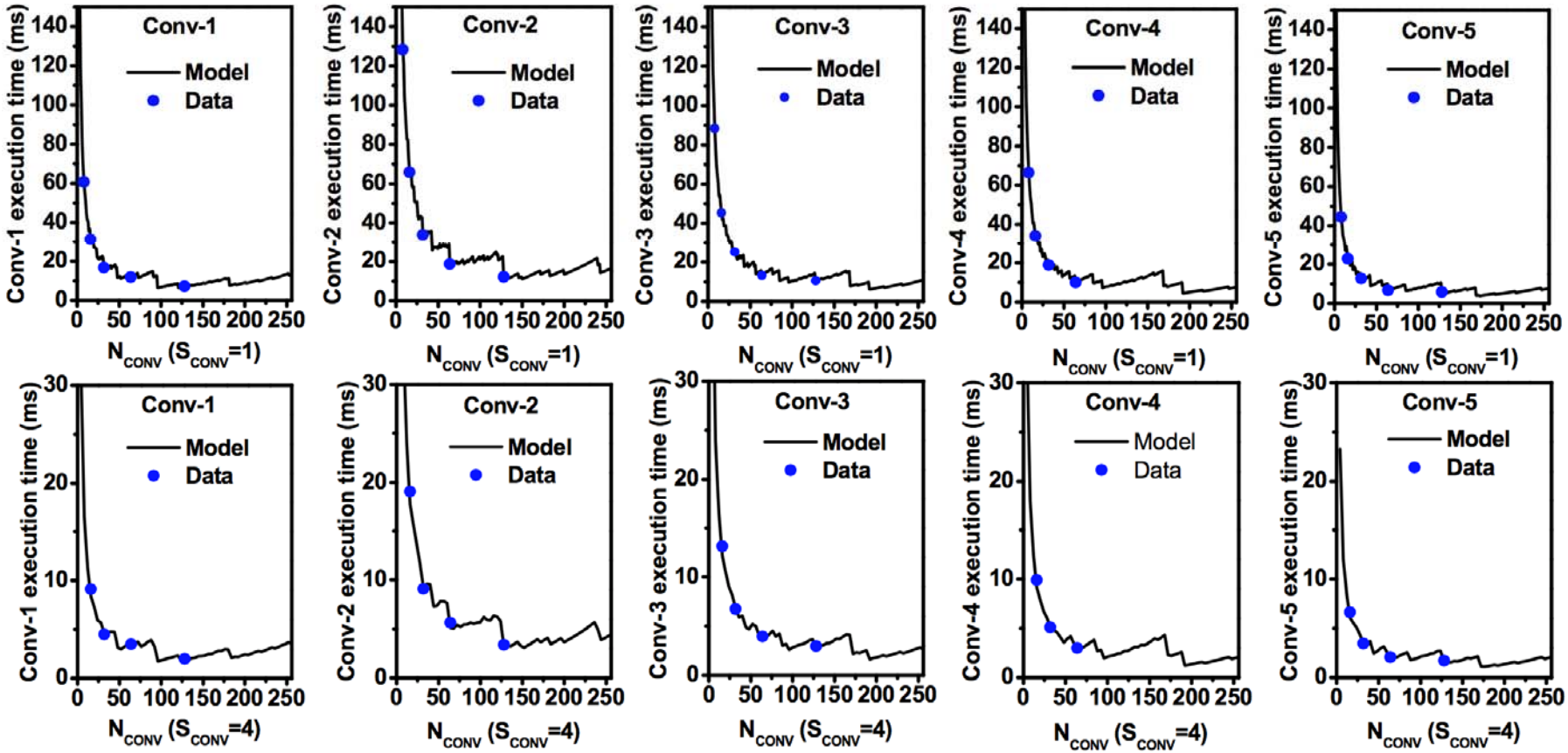
$$= PAD_{NCONV}(\text{Conv filter dimensions} \times \text{No. of input features})$$

$$\times PAD_{NCONV}(\text{No. of output features})$$

$$\times PAD_{NCONV}(\text{output feature dimensions})$$



Model vs. Measured Performance

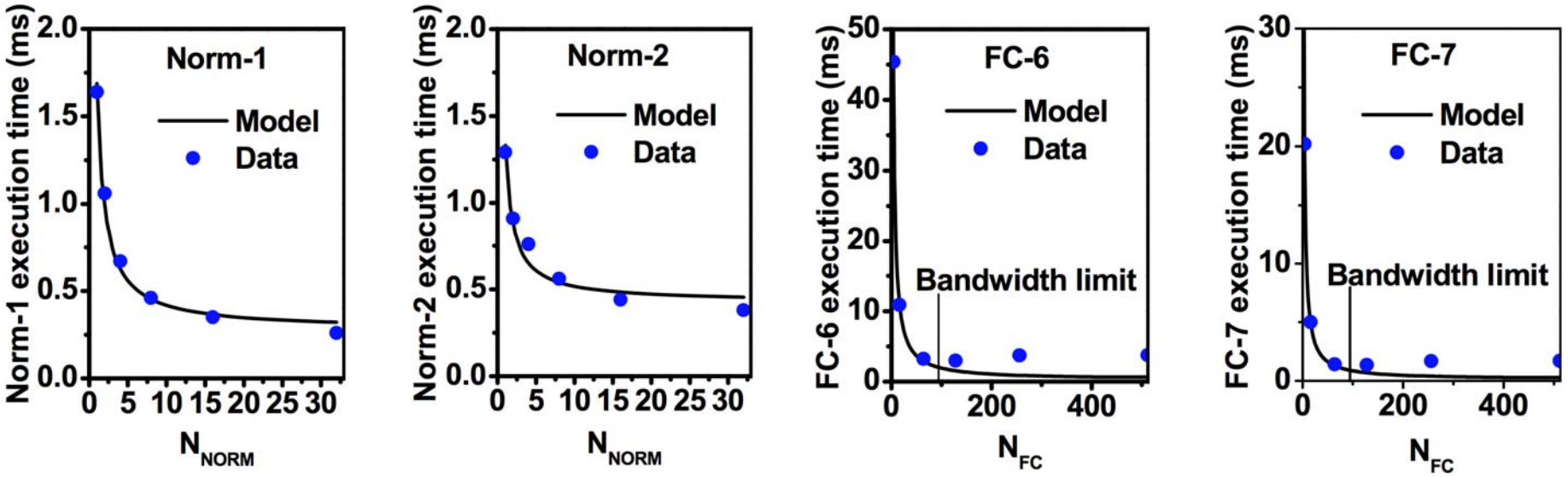


- Non-monotonic behavior because of zero-padding
  - Require global search techniques

Performance Modeling

- Other CNN layers: pooling, normalization, classification

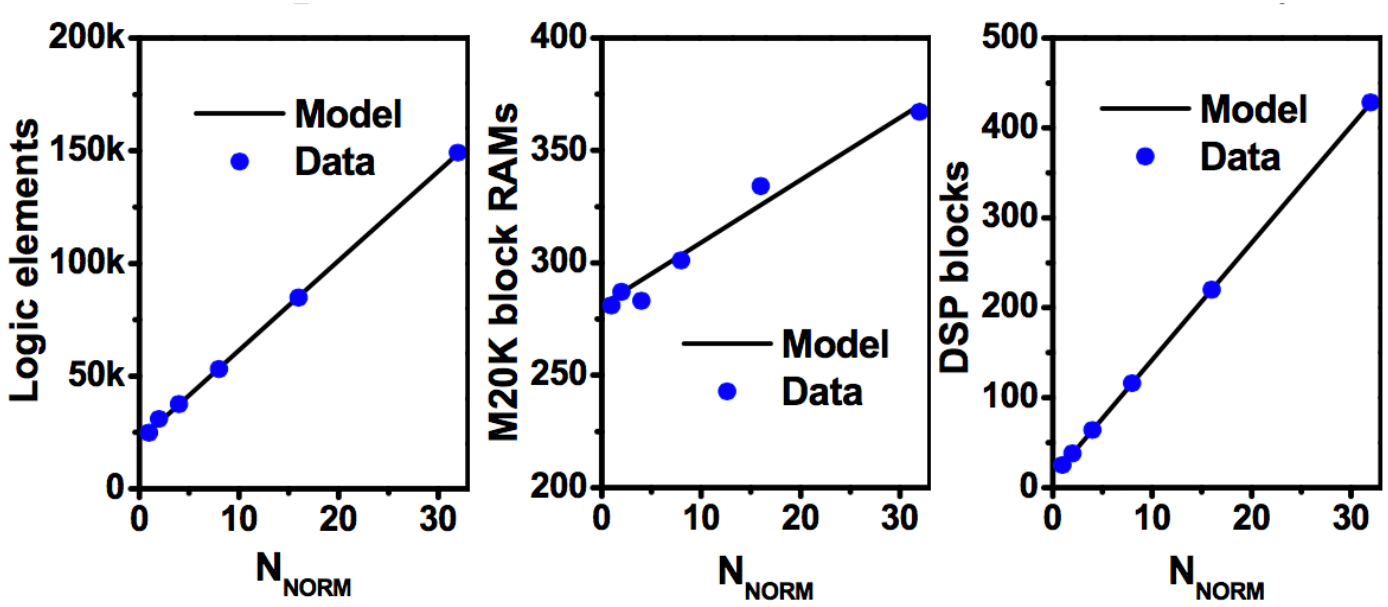
$$Execution\ time_i = \frac{\#Operations_i}{Unroll\ factor \times Frequency}$$



- Classification layers (FC): limited by memory bandwidth
  - Hardware specification defines upper limit for N_{FC}

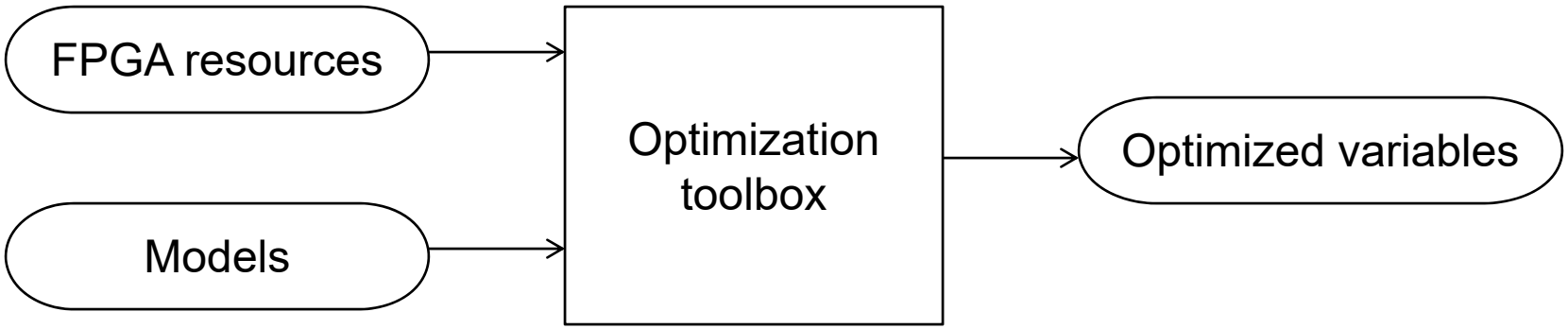
FPGA Resource Utilization Models

- Optimizations in HLS tools : difficult to analytically model
  - Loop pipelining
  - Memory coalescing
- Linear regression models: DSP blocks, logic elements and on-chip memory resources



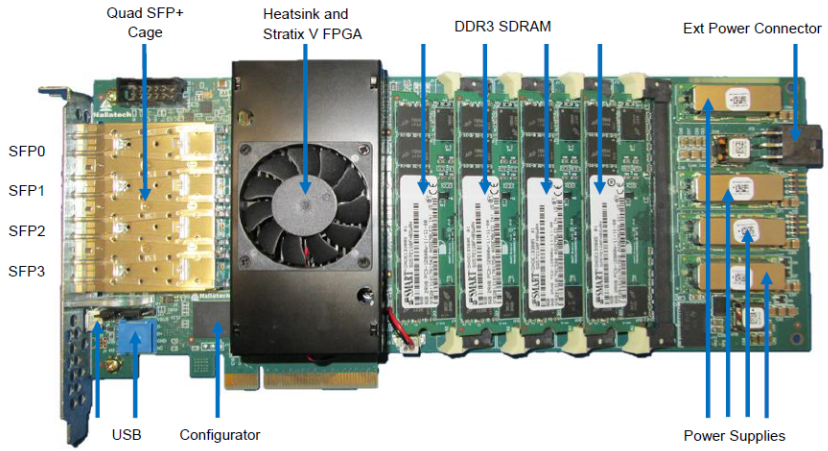
Optimization Setup

- Genetic algorithm in Matlab global optimization toolbox
  - **Objective function :** Performance models (CNN-specific)
  - Integer variables : ($\mathbf{N_{CONV}}$, $\mathbf{S_{CONV}}$, $\mathbf{N_{NORM}}$, $\mathbf{N_{POOL}}$, $\mathbf{N_{FC}}$)
  - **Constraints :** Resource utilization models (FPGA-specific)
- Altera OpenCL compiler constraints
  - $\mathbf{S_{CONV}}$: 1, 2, 4, 8 or 16
  - $\mathbf{N_{CONV}}$ = N*$\mathbf{S_{CONV}}$ (N : integer)



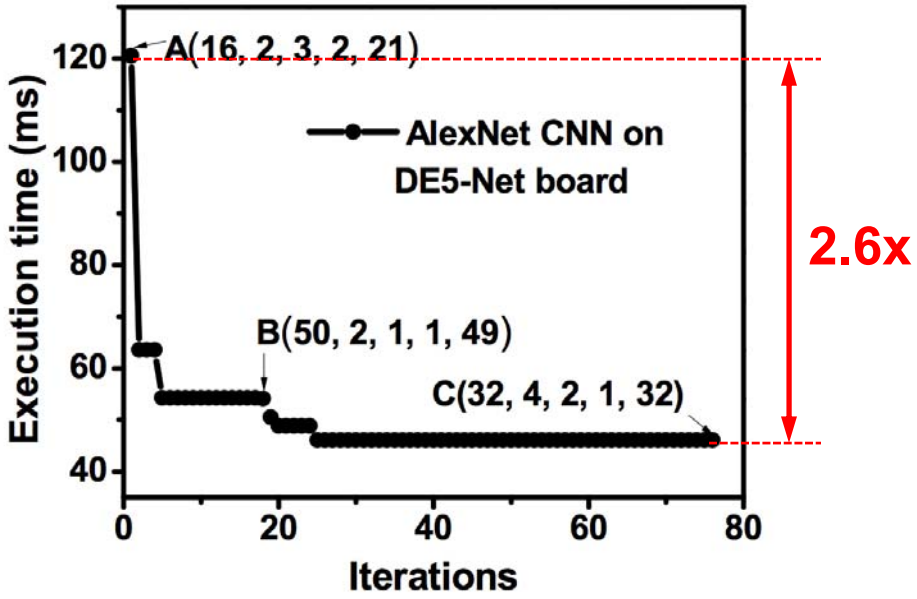
FPGA Hardware

Specification	P395-D8	DE5-Net
FPGA	Stratix-V GSD8	Stratix-V GXA7
Logic elements	695K	622K
DSP blocks	1,963	256
M20K RAMs	2,567	2,560
External memory	4× 8GB DDR3	2× 2GB DDR3



P395-D8 board with Altera Stratix-V GSD8 FPGA

Optimization Progress



AlexNet on DE5-Net	A	B	C
Exec. time (model)	120.6 ms	54.3 ms	46.1 ms
Exec. time (measured)	117.7 ms	52.6 ms	45.7 ms
Logic elements	158k	152k	153k
M20K memory blocks	1,439	1,744	1,673
DSP blocks	164	234	246

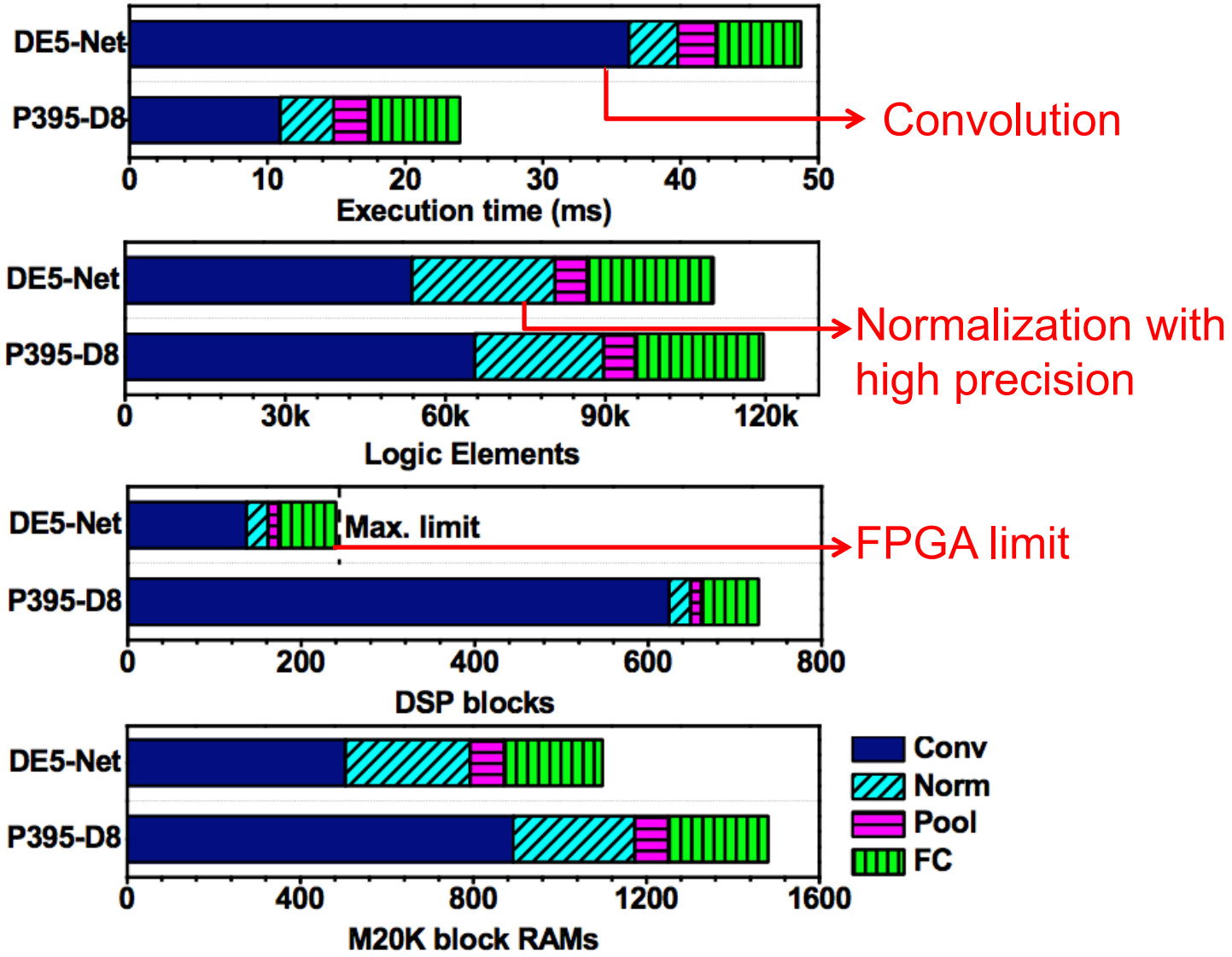
Designs with similar hardware utilizations - significant performance gap

Experimental Results

Parameter	P395-D8 board		DE5-Net board	
	AlexNet	VGG-16	AlexNet	VGG-16
N_{CONV}	64	64	32	64
S_{CONV}	8	8	4	2
N_{NORM}	2	-	2	-
N_{POOL}	1	1	1	1
N_{FC}	71	64	32	30

	Hardware	Classification time/image (ms)	Throughput (GOPS)	
AlexNet	P395-D8	20.1	72.4	
	DE5-Net	45.7	31.8	**9.5x**
	CPU (i5-4590)	191.9	7.6	
VGG-16	P395-D8	262.9	117.8	
	DE5-Net	651.2	47.5	**5.5x**
	CPU (i5-4590)	1437.2	21.5	

AlexNet Utilization & Performance



No. of DSP blocks highly impacts accelerator throughput

Accuracy Comparison

Accuracy	Full precision in Caffe tool		Fixed-point FPGA implementation	
	Top-1	Top-5	Top-1	Top-5
AlexNet	56.82%	79.95%	55.41%	78.98%
VGG	68.35%	88.44%	66.58%	87.48%

- Accuracy measured with 50K ImageNet validation images
- Accuracy degradation: <2% for top-1; <1% for top-5
  - Limited precision weights
  - Fixed point operations
  - Approximation in normalization function

Comparison

	[1] ICCD '13	[2] FPGA '15	This work	This work
FPGA	**Virtex-6 VLX240T**	**Virtex-7 VX485T**	**Stratix-V GSD8**	**Stratix-V GSD8**
Frequency	150 MHz	100 MHz	140 MHz	120 MHz
CNN size	2.74 GMAC	AlexNet 1.33 GOP	AlexNet 1.46 GOP	VGG-16 30.9 GOP
Precision	fixed	float (32b)	fixed (8-16b)	fixed (8-16b)
Throughput	17 GOPS[b]	61.6 GOPS[a]	126.6 GOPS[a] 72.4 GOPS[b]	136.5 GOPS[a] 117.8 GOPS[b]

[a] convolution operations only [b] all operations for image classification

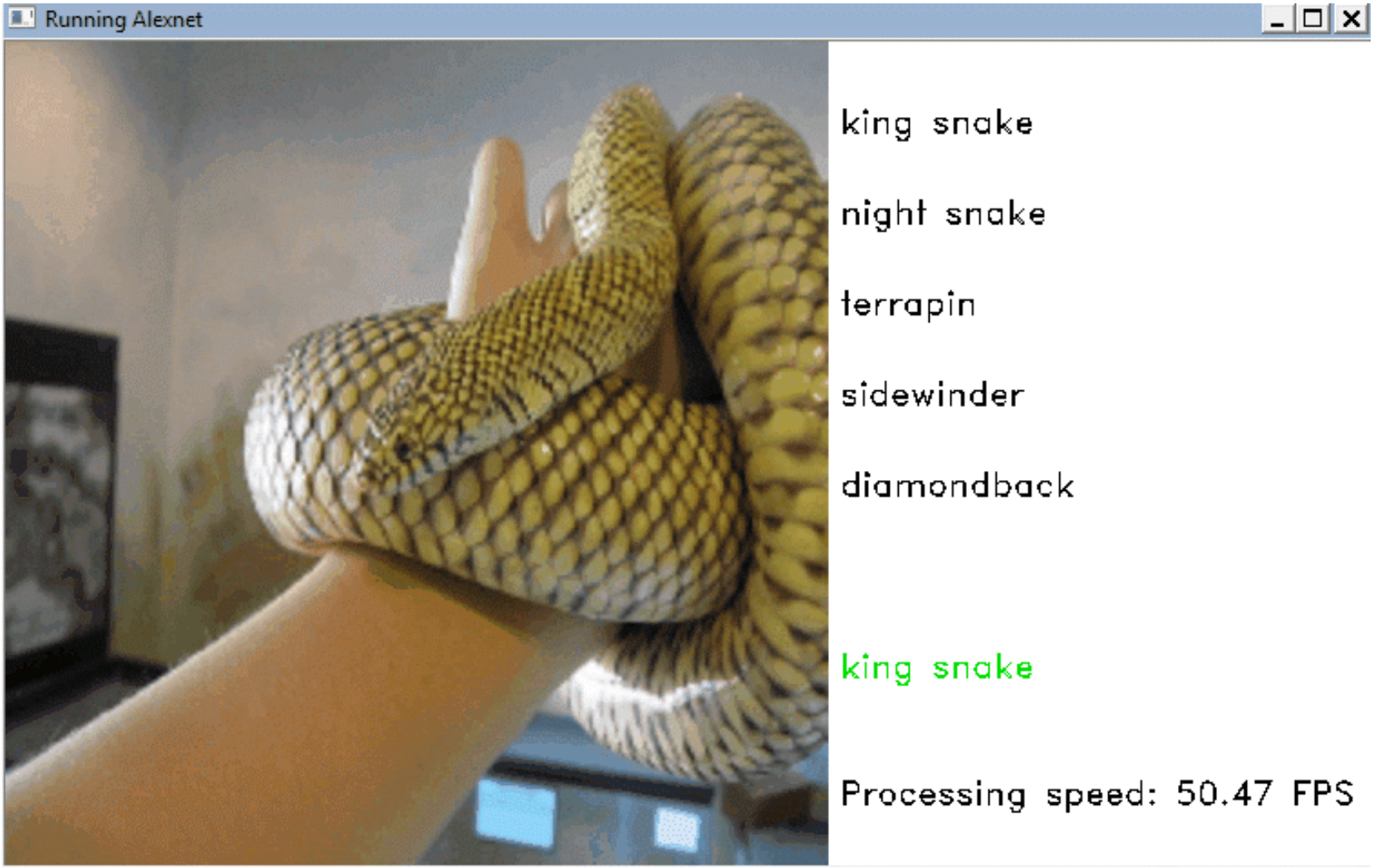
1 GMAC = 2 GOP

[1] M. Peemen, et al. Memory-centric accelerator design for convolutional neural networks. In *ICCD* 2013.
[2] C. Zhang, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *ACM ISFPGA* 2015.

Summary

- OpenCL based FPGA accelerator for CNNs
  - Fixed point operations on reduced precision weights
  - Parameterized design variables for acceleration
- Optimization strategy to maximize system throughput
  - Genetic algorithm on performance and utilization models
- End-to-end ImageNet classification demonstrated
  - AlexNet, VGG-16 on two Altera Stratix-V FPGA boards
  - AlexNet : 72.4 GOPS, VGG-16: 117.8 GOPS
- **Future directions:**
  - Optimizing pipelined CNN stages for batch processing
  - Running compressed network with sparse matrix operations

Demo: AlexNet on FPGA



Running on P395-D8 FPGA board

Thank you !

Q & A