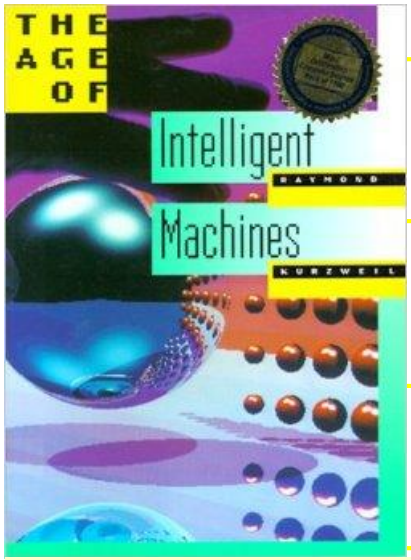
Exploration of FPGA based Accelerators for Convolutional Neural Networks

Guangyu Sun
Peking University

gsun@pku.edu.cn

Computer vs. Intelligence



The Age of Intelligent Machines
MIT Press, 1990
ISBN 0-262-11121-7

Dr. Raymond Kurzweil

Human
100G Neurons

Monkey
3G Neurons

Lizard
2M Neurons

Worm
300 Neurons

Bacteria
1 “Neuron”

“A sufficiently advanced computer program could exhibit human-level intelligence”

Convolutional Neural Network

Automotive Safety

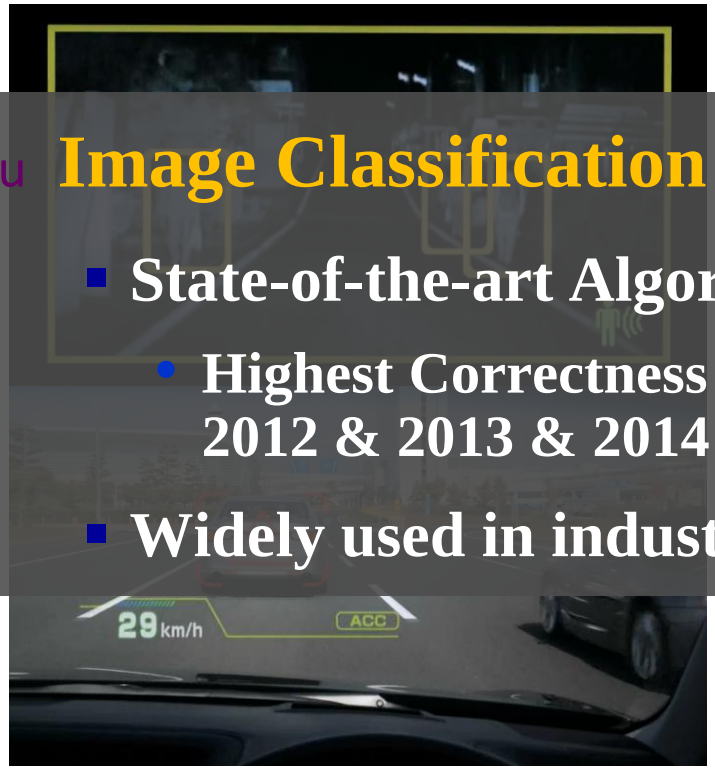


Image Descriptions @ Stanford

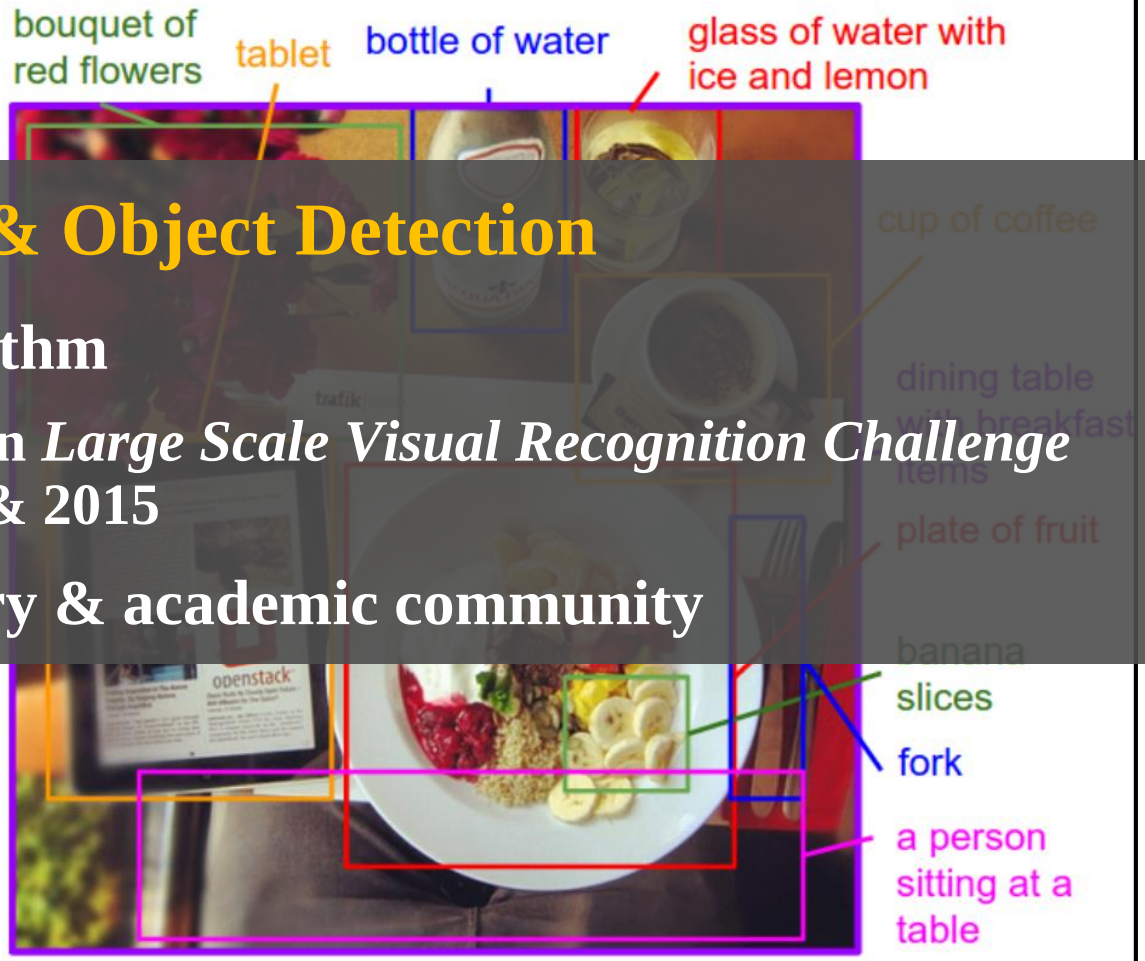
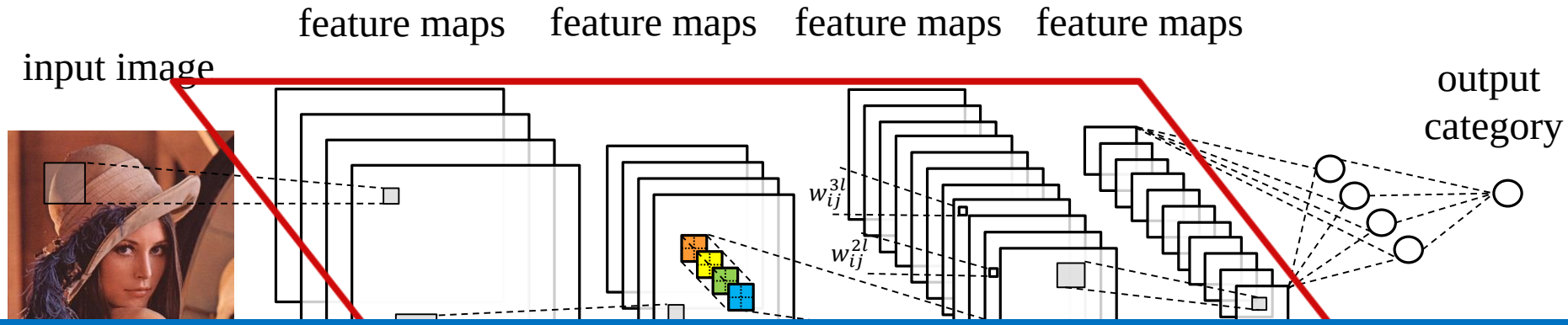


Image Classification & Object Detection

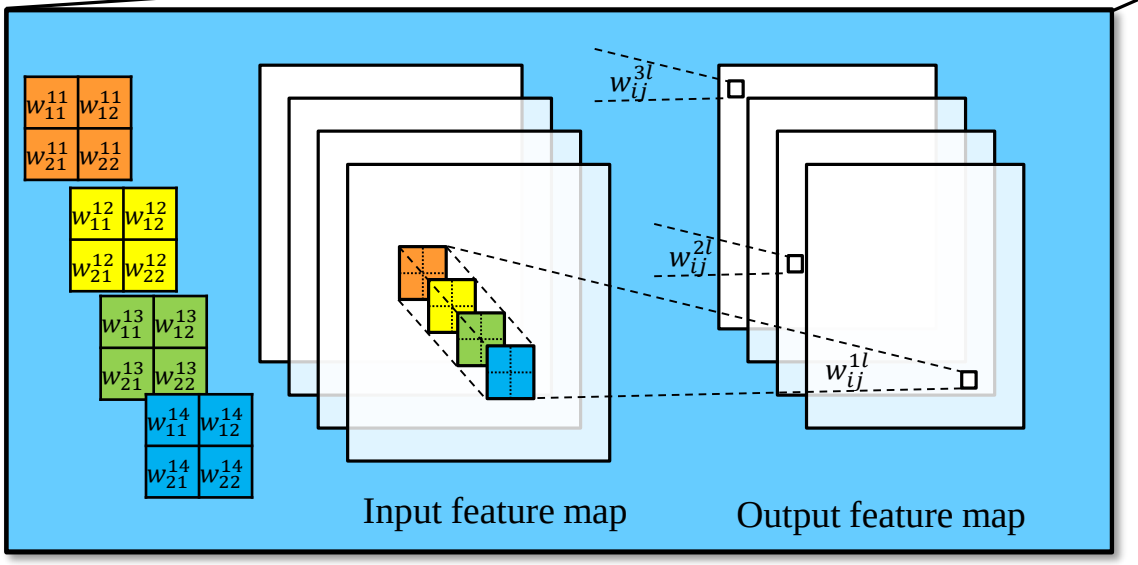
- State-of-the-art Algorithm
  - Highest Correctness in *Large Scale Visual Recognition Challenge* 2012 & 2013 & 2014 & 2015
- Widely used in industry & academic community

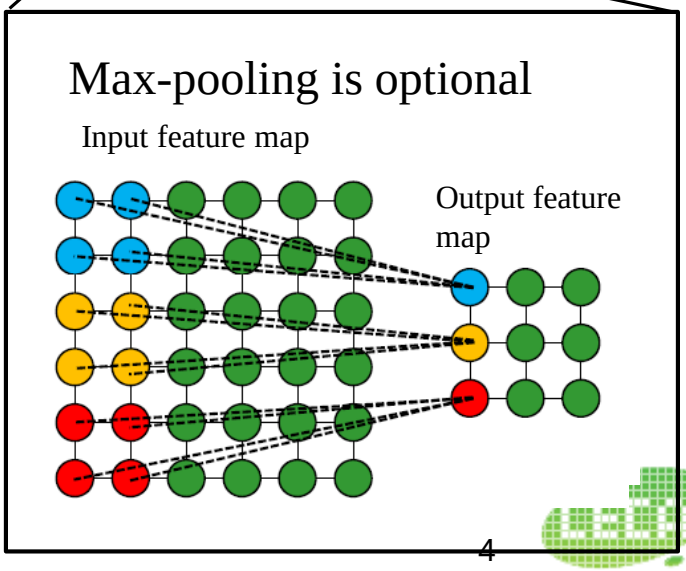
Convolutional Neural Network



Convolutional layers account for over 90% computation

[1] A. Krizhevsky, etc. Imagenet classification with deep convolutional neural networks. NIPS 2012.
[2] J. Cong and B. Xiao. Minimizing computation in convolutional neural networks. ICANN 2014





Related Work

FPGA-based:

1. **A programmable parallel accelerator for learning and classification. PACT 2010;**
2. **A Dynamically Configurable Coprocessor for Convolutional Neural Networks. ISCA 2010;**
3. **Memory-Centric Accelerator Design for Convolutional Neural Networks. ICCD 2013;**
4. **Going deeper with embedded FPGA platform for convolutional neural network. FPGA 2016;**
5. **Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. FPGA 2016;**
6. **Deepburning: automatic generation of FPGA-based learning accelerators for the neural network family. DAC 2016;**

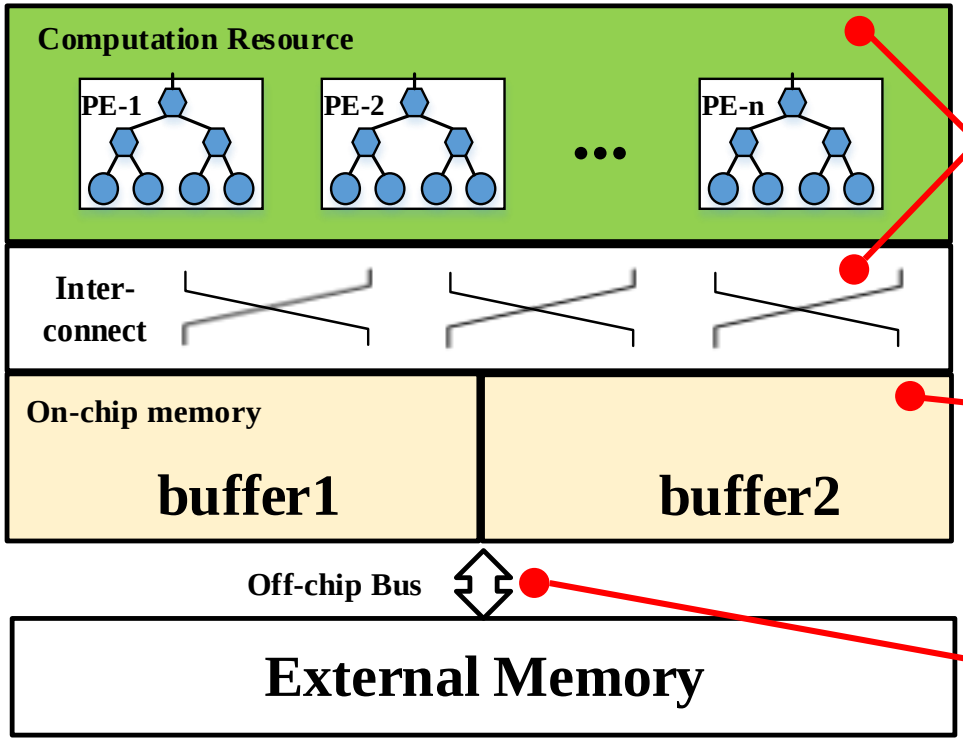
ASIC-based:

1. **DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. ASPLOS 2014;**
2. **ShiDianNao: Shifting Vision Processing Closer to the Sensor. ISCA 2015;**
3. **ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. ISCA 2016 …**

Our Work

- **Single-board Acceleration**
  - **Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks (FPGA 2015)**
- **Multi-board Acceleration**
  - **Energy-Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster (ISLPED 2016)**
- **Automated Design Flow**
  - **Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks (ICCAD 2016)**

Convolution Layer Computing on FPGA



Challenge : Resource

Solution: Unroll & Pipeline

Challenge: BRAM

Solution: Loop Tiling

Challenge: Bandwidth

Solution: Data reuse

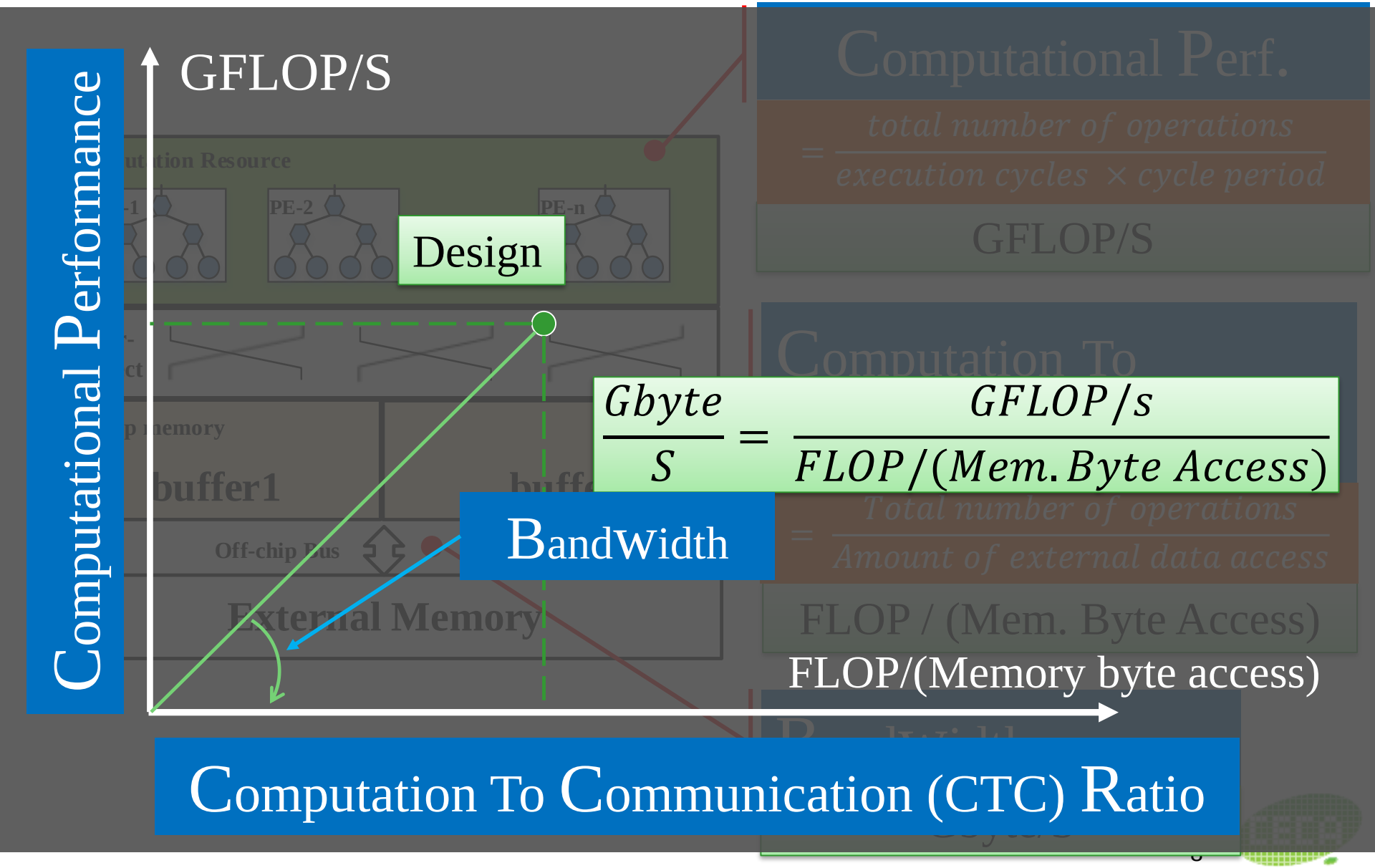
Challenge: Co-optimization,
-- match 'computation' & 'communication'

Solution: Roofline Model

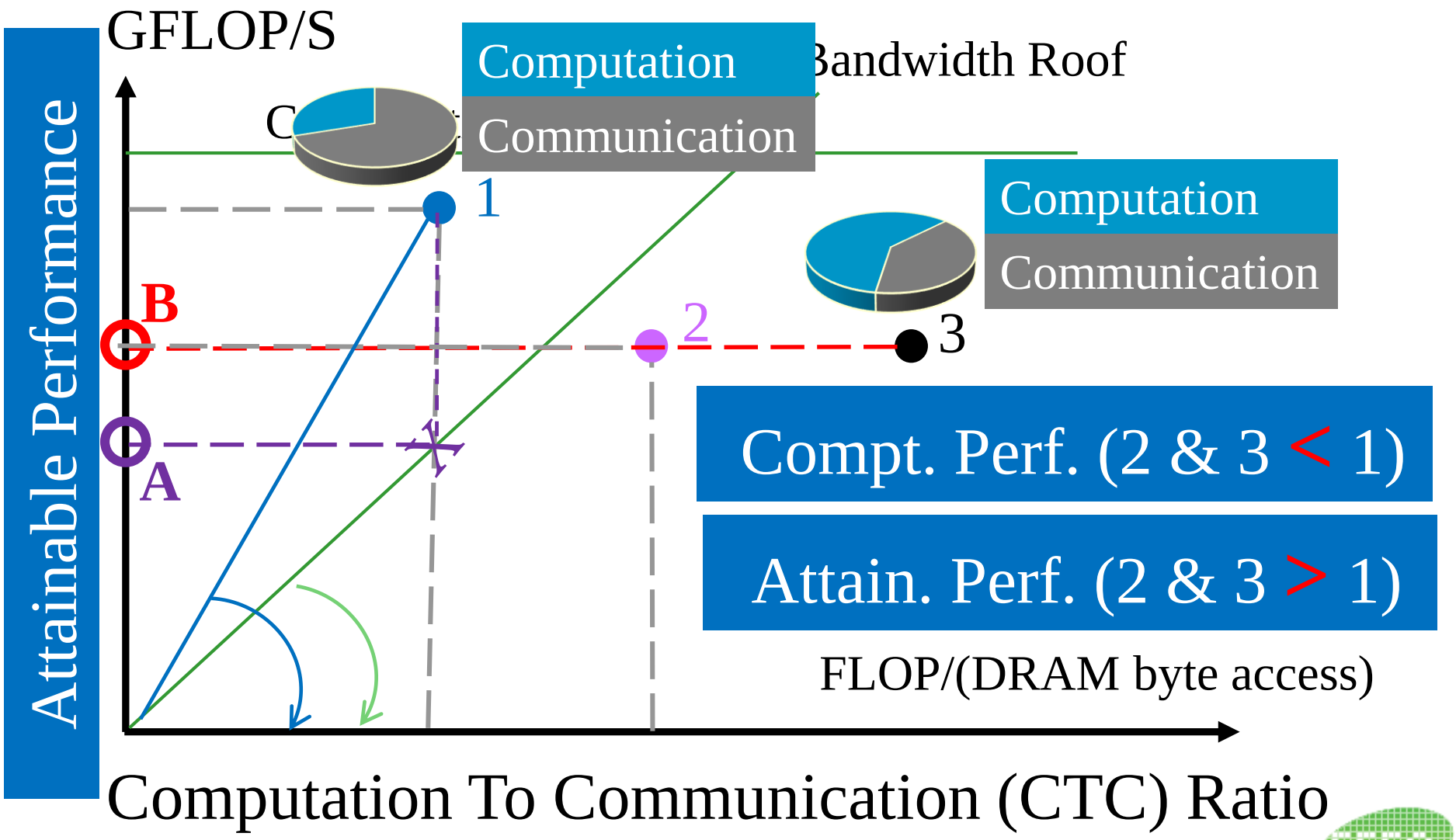
Main Contribution

S. Williams, et al. Roofline: An insightful visual performance model for multicore architectures. Commun. ACM, 52(4):65~76, Apr. 2009

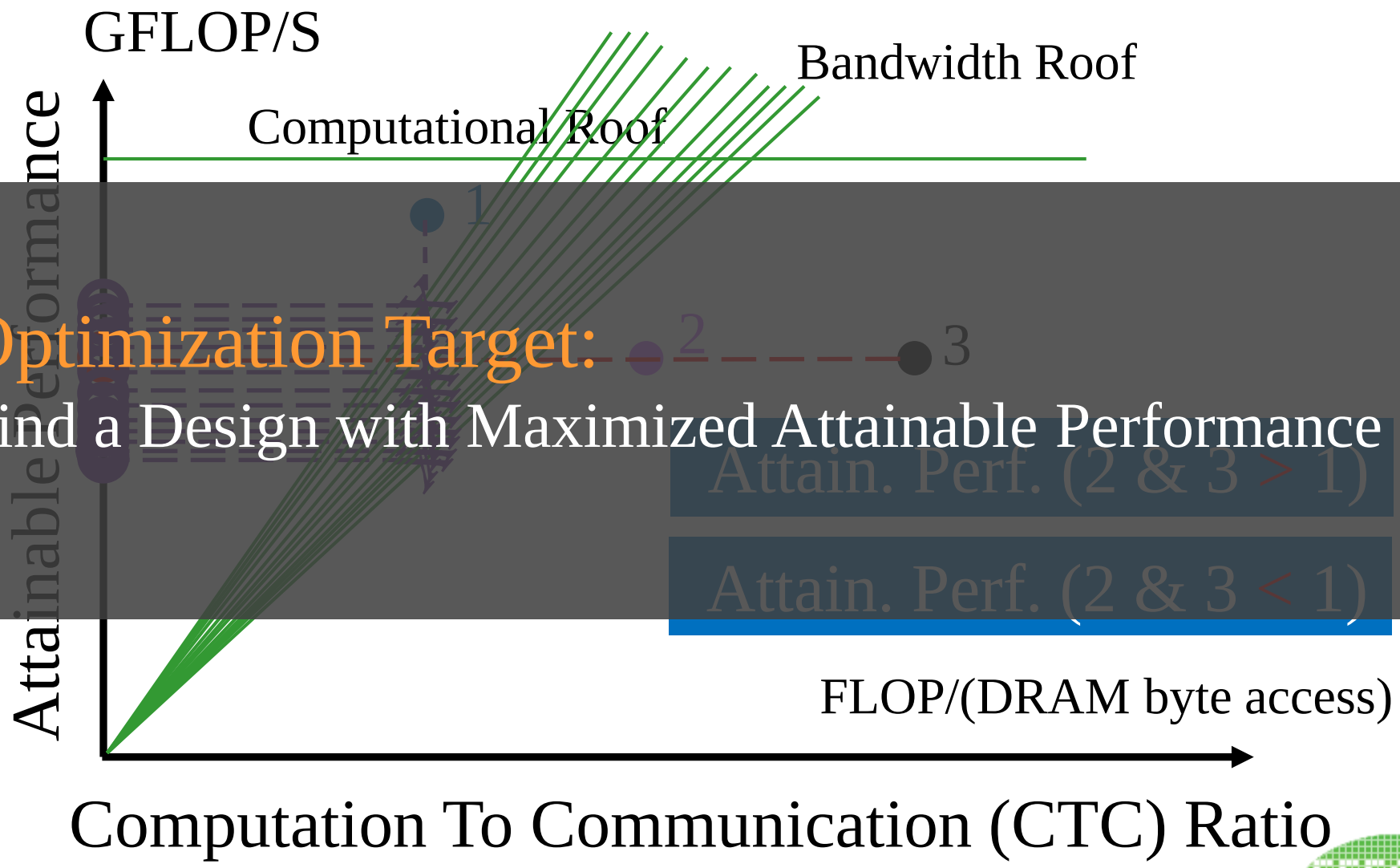
Definitions in Roofline Model



Attainable Performance



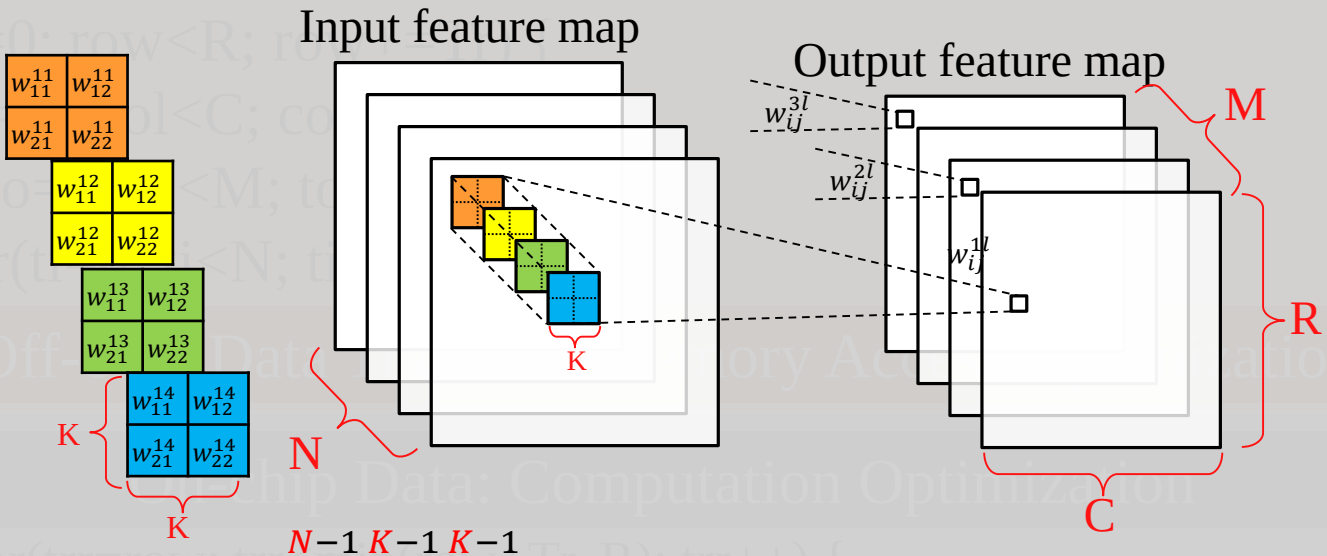
Attainable Performance



Optimization Target:

Find a Design with Maximized Attainable Performance

Loop tiling

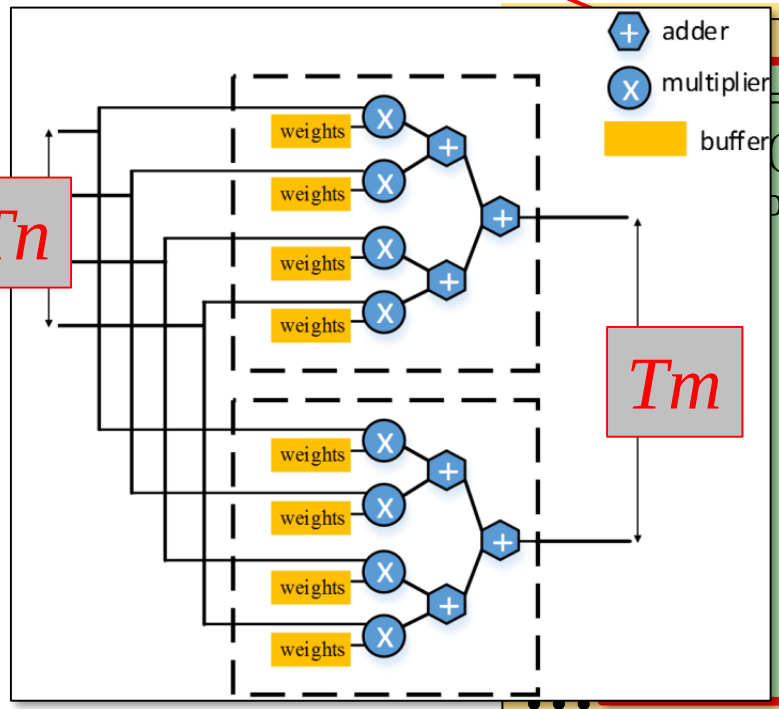


$$Y[m][r][c] = \sum_{n=0}^{N-1} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} W[m][n][i][j] \times X[n][S \times r + i][S \times c + j]$$

```
1  for(row=0; row<R; row++) {
2    for(col=0; col<C; col++) {
3      for(to=0; to<M; to++) {
4        for(ti=0; ti<N; ti++) {
5          for(i=0; i<K; i++) {
6            for(j=0; j<K; j++) {
                output_fm[to][row][col] +=
                weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
}}}}}}
```

R, C, M, N, K, S are all configuration parameters of the convolutional layer

Computation Optimization




```
(Tile loops)
=0; i<K; i++) {
(j=0; j<K; j++) {
or(trr=row; trr<min(row+Tr, R); trr++) {
 for(tcc=col; tcc<min(tcc+Tc, C); tcc++) {
 HLS pipeline
   for(too=to; too<min(to+Tm, M); too++) {
 HLS UNROLL
      for(tii=ti; tii<(ti+Tn, N); tii++) {
 HLS UNROLL
           output_fm[to][row][col] +=
           weights[to][ti][i][j]*input_fm[ti][S*row+i][S*col+j];
    }}}}}}
```

$$\text{Computational Performance} = \frac{total\ number\ of\ operations}{execution\ cycles}$$

$$\text{execution cycles} = \frac{R}{Tr} \times \frac{C}{Tc} \times \left\lceil \frac{M}{Tm} \right\rceil \times \left\lceil \frac{N}{Tn} \right\rceil \times (K \times K \times Tc \times Tr + P)$$

$$\approx \left\lceil \frac{M}{Tm} \right\rceil \times \left\lceil \frac{N}{Tn} \right\rceil \times R \times C \times K \times K$$

Memory Access Optimization

```
1  for(row=0; row<R; row+=Tr) {
2    for(col=0; col<C; col+=Tc) {
3      for(to=0; to<M; to+=Tm) {
4        for(ti=0; ti<N; ti+=Tn) {
         load output feature map
         S: foo(output_fm(to, row, col));
         store output feature map
       }
}}}
```

```
5  for(trr=row; trr<min(row+Tr, R); trr++)
6    for(tcc=col; tcc<min(tcc+Tc, C); tcc++)
7      for(too=to; too<min(to+Tn, M); too++)
8        for(tii=ti; tii<(ti+Tn, N); tii++)
9          for(i=0; i<K; i++) {
10           for(j=0; j<K; j++) {
               output_fm[to][row][col] +=
                 weights[to][ti][i][j]*
                 input_fm[ti][S*row+i][S*col+j];
   }}}}}}
```

Before local memory promotion

'*output_fm*' memory access

$$= 2 \times \frac{R}{Tr} \times \frac{C}{Tc} \times \frac{M}{Tm} \times \frac{N}{Tn} \times Size_{array}$$

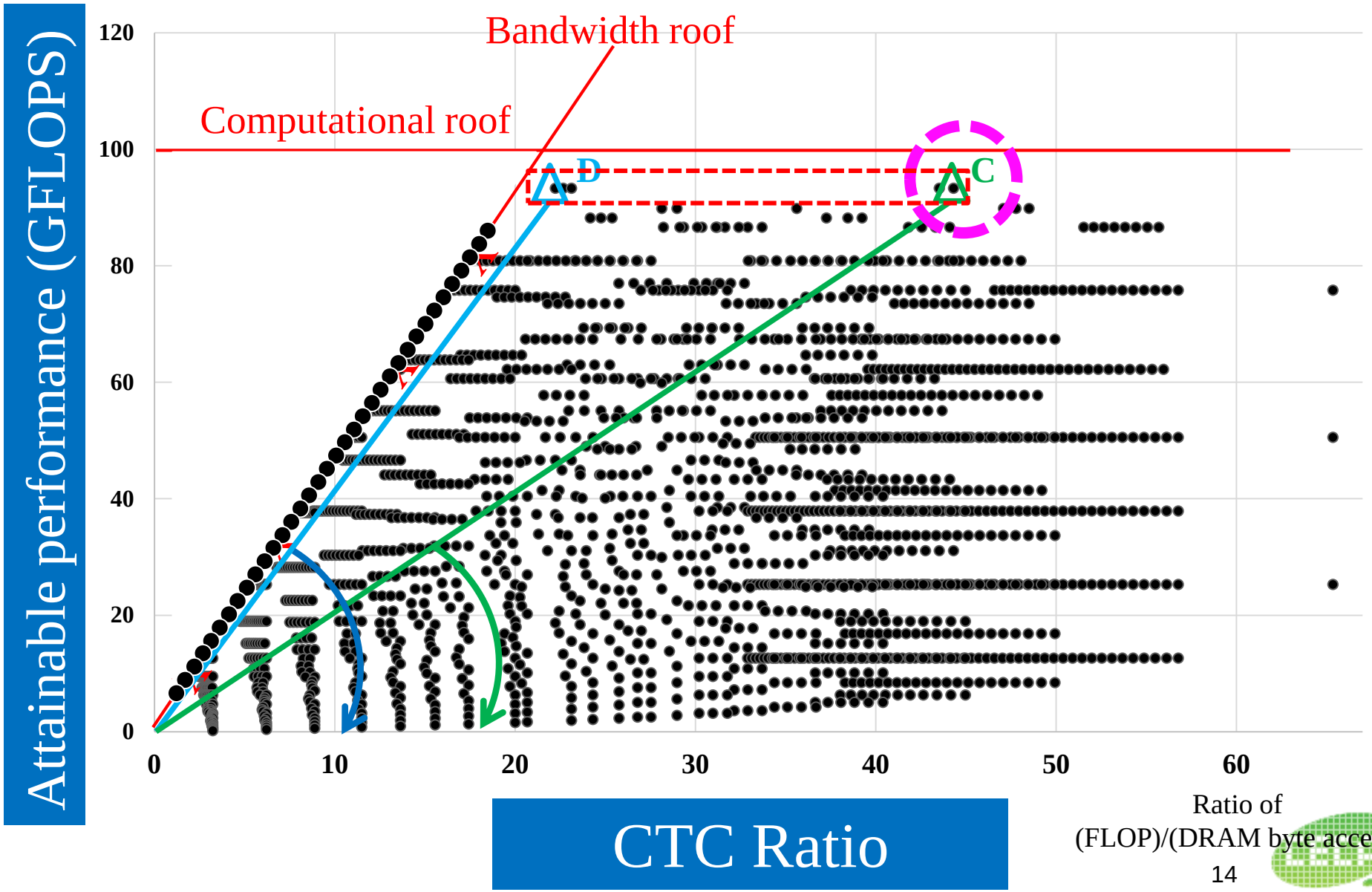
After local memory promotion

'*output_fm*' memory access

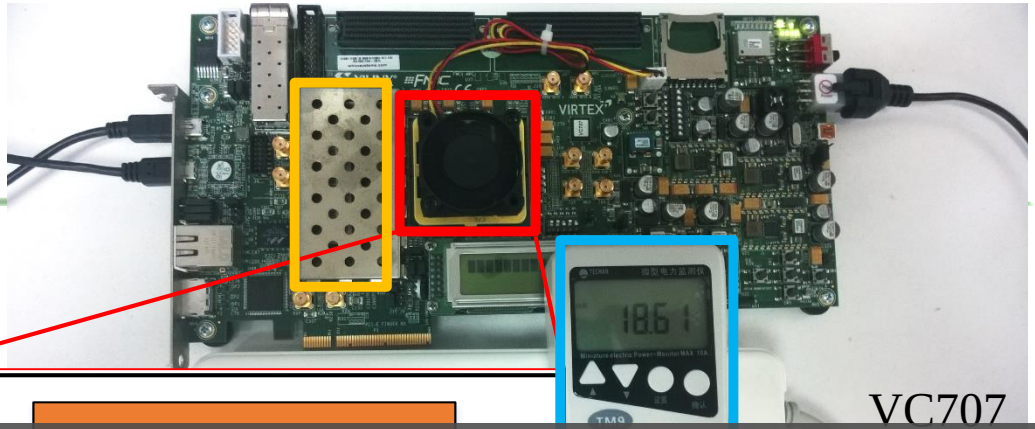
$$= 2 \times \frac{R}{Tr} \times \frac{C}{Tc} \times \frac{M}{Tm} \times Size_{array}$$

$$\textbf{Computation to Communication Ratio} = \frac{\textbf{Total number of operations}}{\textbf{Total amount of external data access}}$$

Design Space Exploration



System Overview

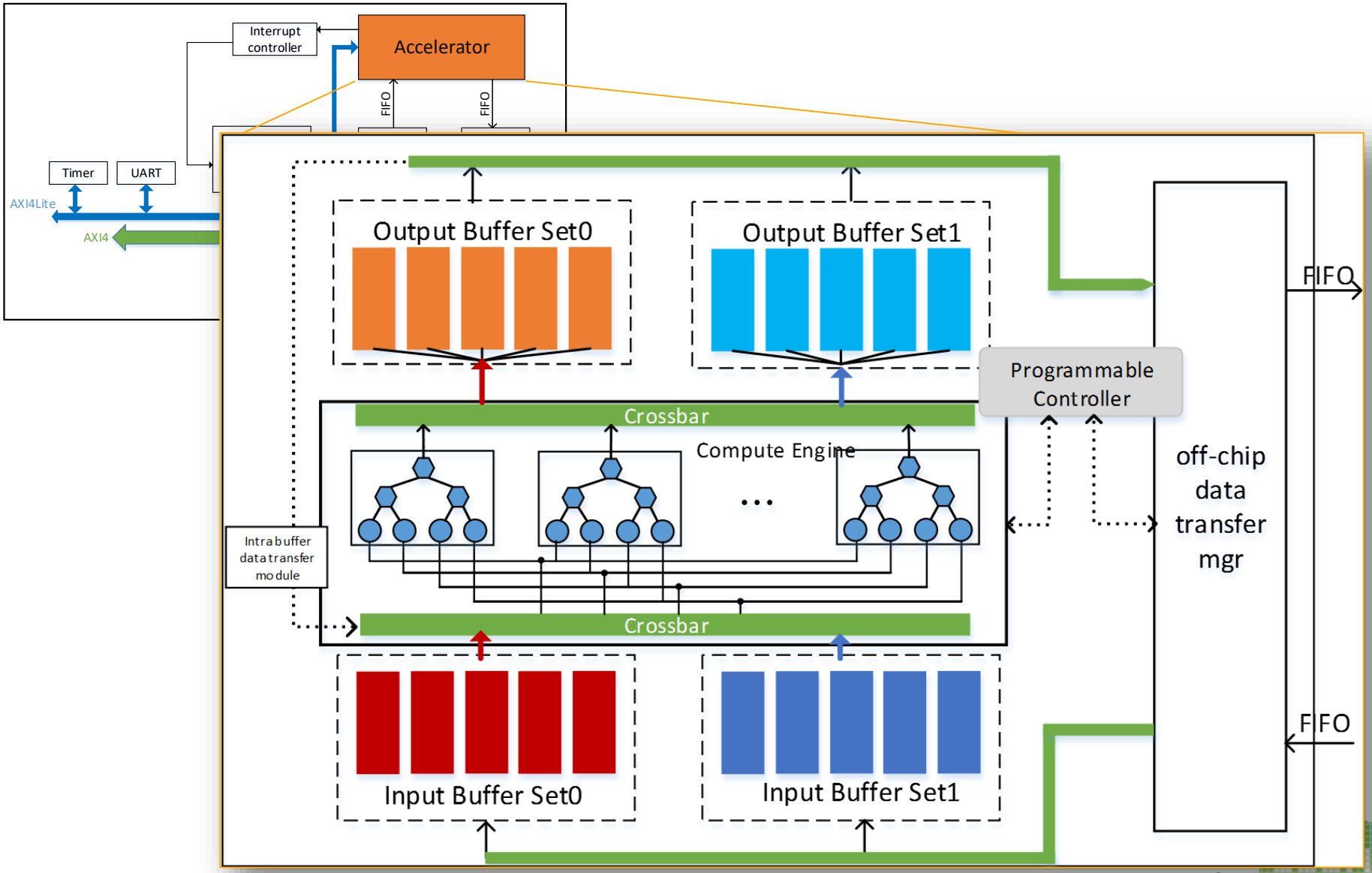


VC707

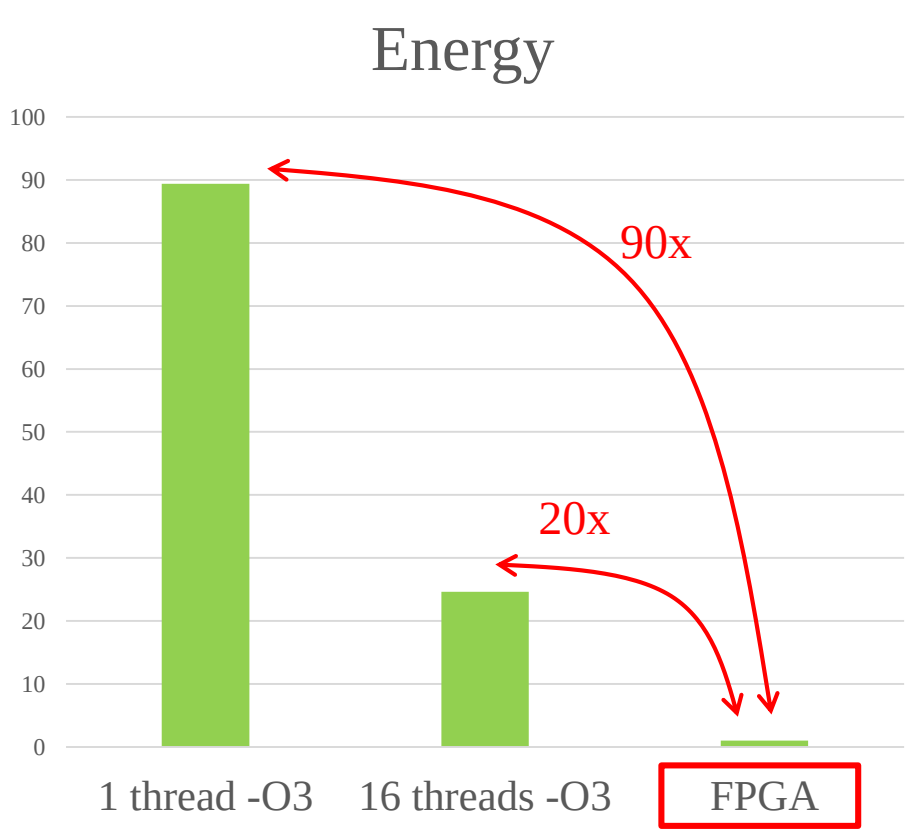
u

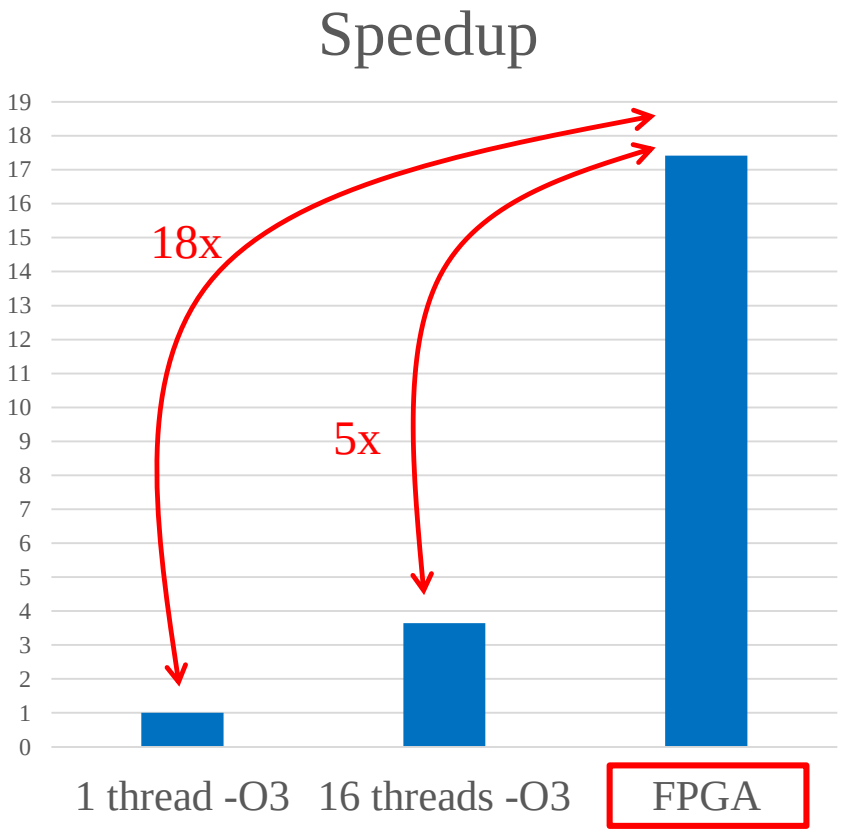
	Estimated	On-board run	Difference
Layer 1	7.32 ms	7.67 ms	~5%
Layer 2	5.11 ms	5.35 ms	~5%
Layer 3	3.42 ms	3.79 ms	~9%
Layer 4	2.59 ms	2.88 ms	~10%
Layer 5	1.73 ms	1.93 ms	~10%
Overall	20.17 ms	21.61 ms	~7%

Accelerator



Experimental Results: vs. CPU



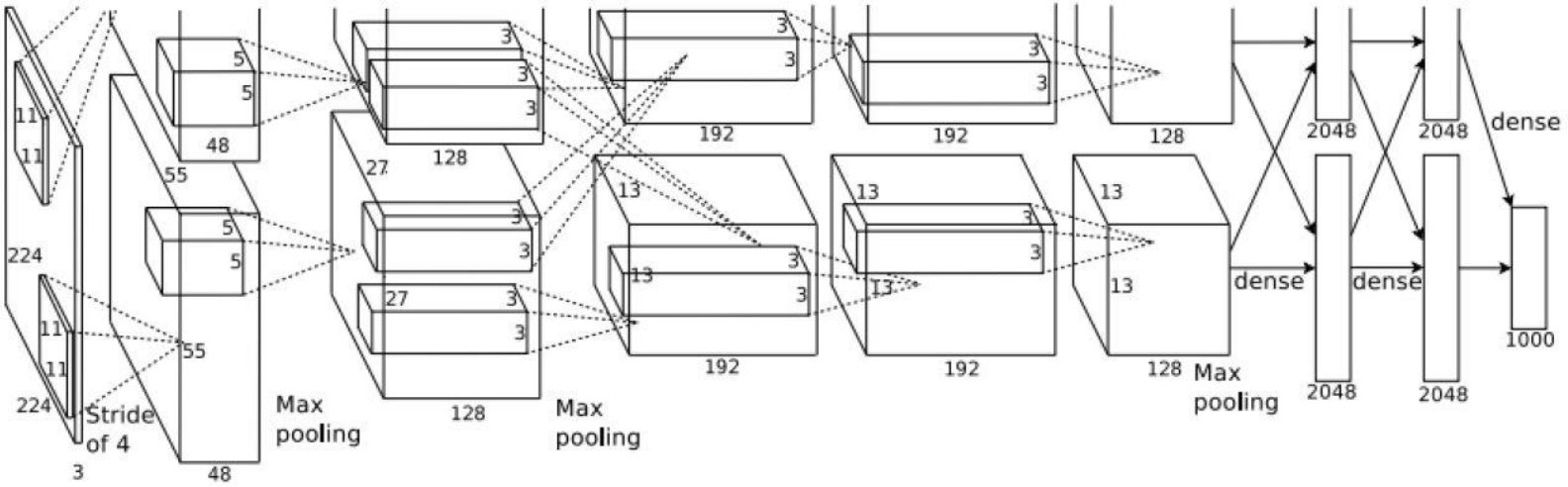


CPU	Xeon E5-2430 (32nm)	16 cores	2.2 GHz	gcc 4.7.2 –O3 OpenMP 3.0
FPGA	Virtex7-485t (28nm)	448 PEs	100MHz	Vivado 2013.4 Vivado HLS 2013.4

Mapping CNN on an FPGA-cluster

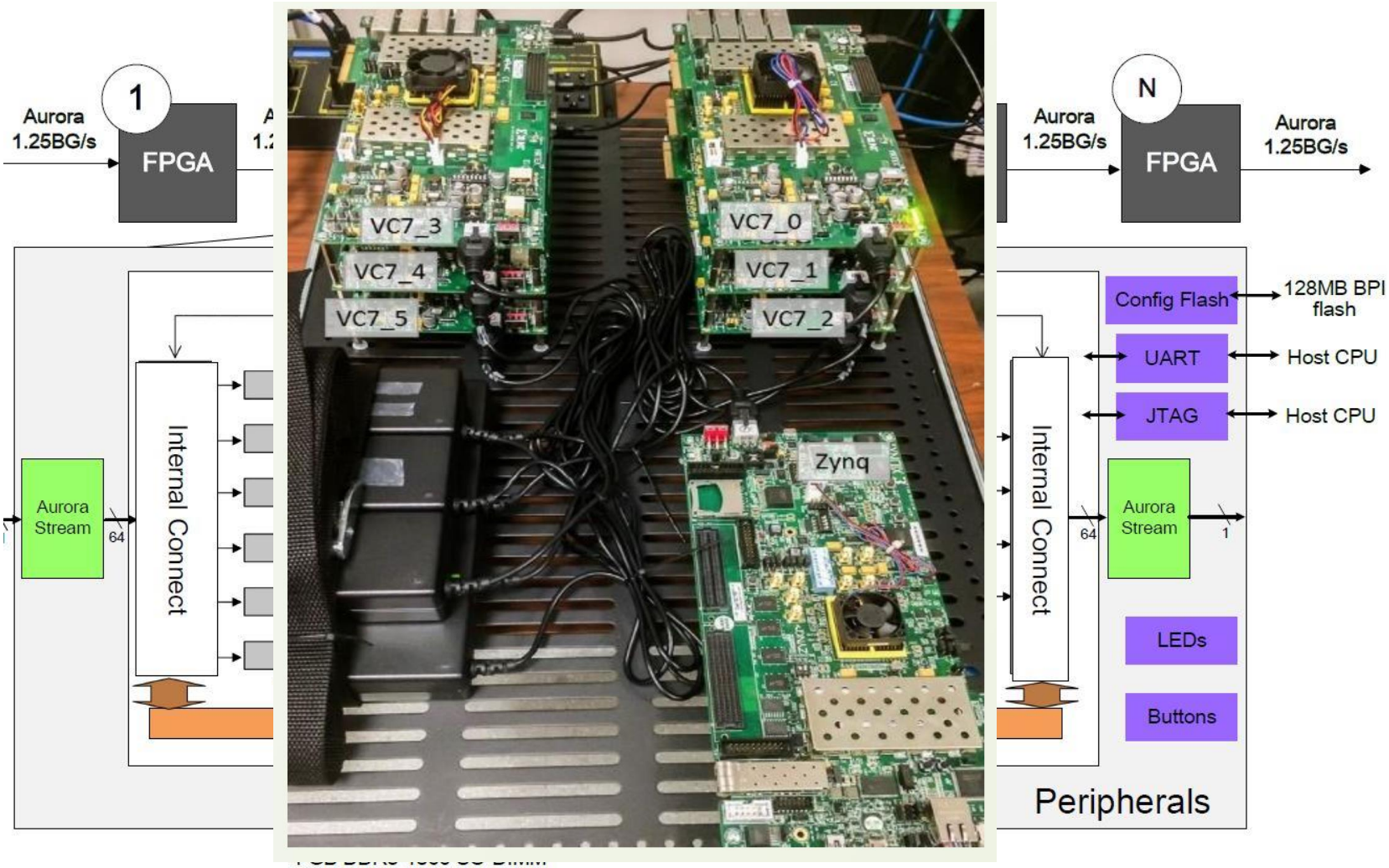
- **Problem 1**
  - **Single FPGA board has limited number of floating-point units and memory buffers**
- **Problem 2**
  - **CNN computing kernels have diverse requirements for compute resource, memory bandwidth, and on-chip buffers**
- **Solution: Multi-board Acceleration**

An Example of Real-life CNN



	Convolution	LR Norm	POOL	Non Linear	Fully Connect
Total OP (10^6)	591	5.5	0.5	0.0	37.7
Percentage(%)	93.2%	0.8%	0.1%	0.0%	6.0%
	Convolution	**LR Norm**	**POOL**	**Non Linear**	**Fully Connect**
Weight Size(MB)	10	0	0	0	224
Percentage(%)	4.31%	0.0%	0.0%	0.0%	95.7%

Overview


Aurora
1.25BG/s
1
FPGA
N
Aurora
1.25BG/s
FPGA
Aurora
1.25BG/s
VC7_3
VC7_4
VC7_5
VC7_0
VC7_1
VC7_2
Zynq
Aurora
Stream
64
Internal Connect
Internal Connect
64
Aurora
Stream
1
Config Flash
128MB BPI
flash
UART
Host CPU
JTAG
Host CPU
LEDs
Buttons
Peripherals

Mapping Problem

Objective 1: Minimize Latency

$$L(i,j,k) = \begin{cases} L(i,j,1), & k = 1 \\ \min_{r=i}^{j-1} \begin{pmatrix} L(i,r,k-1) + \\ L(r+1,j,1) + \\ T_{ext}(r) \end{pmatrix}, & k \geq 2 \end{cases}$$

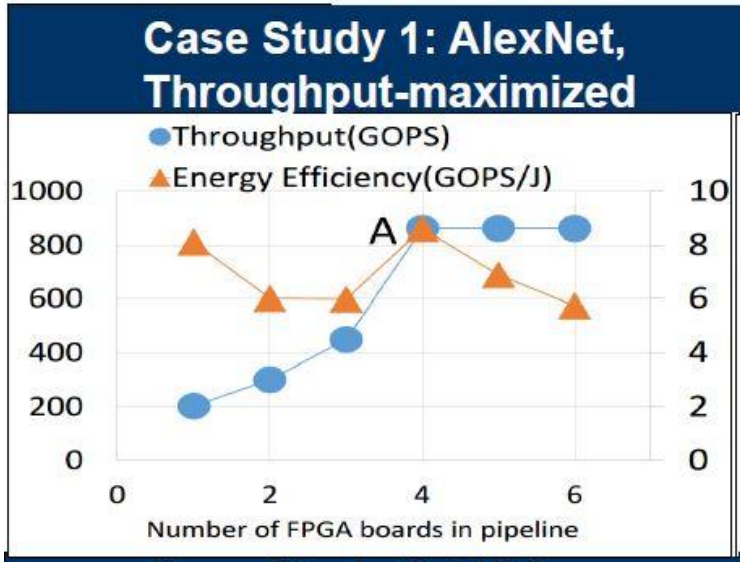
$L(i, j, 1)$ is the minimum latency of implementing layer i to layer j on one FPGA, which can be obtained using existing work.

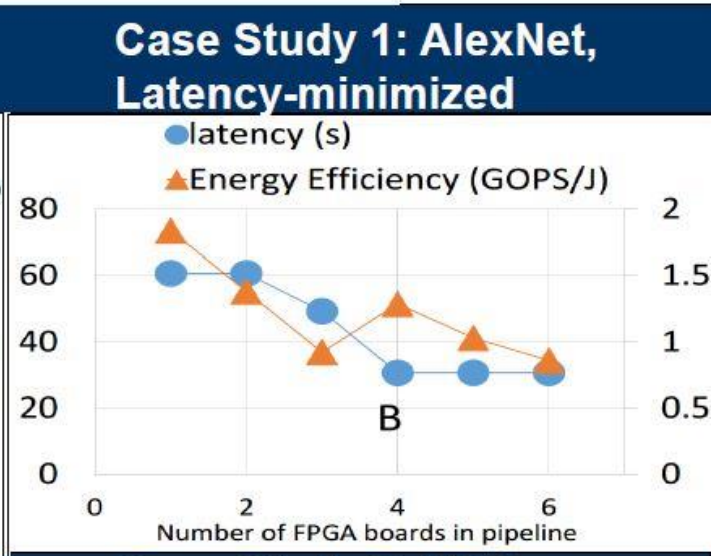
Objective 2: Maximize Throughput

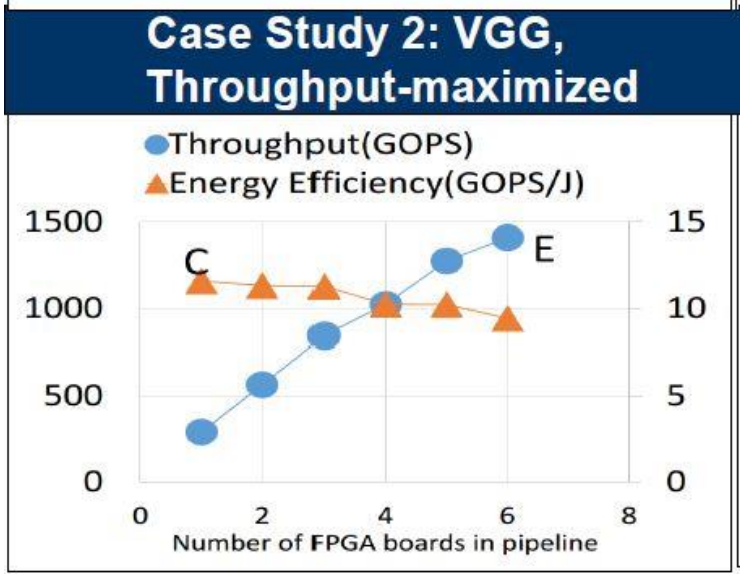
$$T(i,j,k) = \begin{cases} T(i,j,1), & k = 1 \\ \max_{r=i}^{j-1} \min \begin{pmatrix} T(i,r,k-1), \\ T(r+1,j,1), \\ \frac{1}{T_{ext}(r)} \end{pmatrix}, & k \geq 2 \end{cases}$$

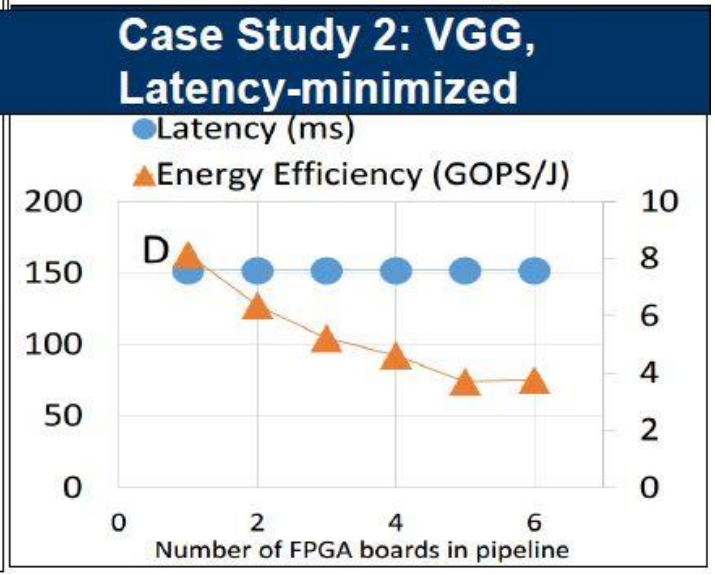
T(i, j, 1) is the maximum throughput achieved by implementing layer i to layer j on one FPGA, which can be obtained using existing work.

Case Study


Case Study 1: AlexNet, Throughput-maximized
Throughput(GOPS)
Energy Efficiency(GOPS/J)
1000
800
600
400
200
0
10
8
6
4
2
0
A
0
2
4
6
Number of FPGA boards in pipeline


Case Study 1: AlexNet, Latency-minimized
latency (s)
Energy Efficiency (GOPS/J)
80
60
40
20
0
2
1.5
1
0.5
0
B
0
2
4
6
Number of FPGA boards in pipeline

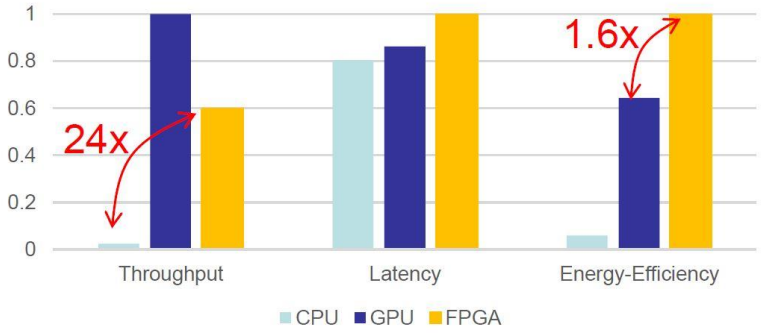

Case Study 2: VGG, Throughput-maximized
Throughput(GOPS)
Energy Efficiency(GOPS/J)
1500
1000
500
0
15
10
5
0
C
E
0
2
4
6
8
Number of FPGA boards in pipeline


Case Study 2: VGG, Latency-minimized
Latency (ms)
Energy Efficiency (GOPS/J)
200
150
100
50
0
10
8
6
4
2
0
D
0
2
4
6
Number of FPGA boards in pipeline

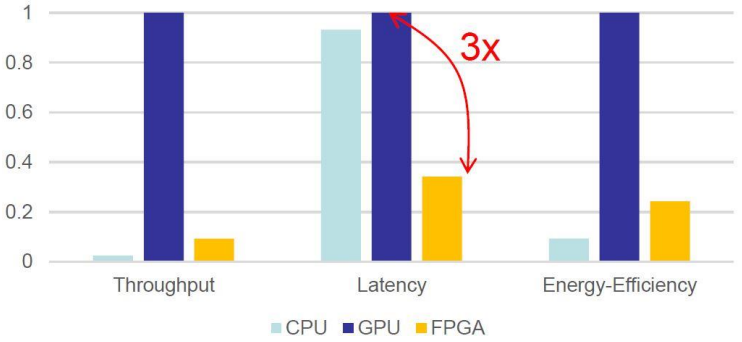
Cross-Platform Comparison

AlexNet

Solution A: throughput optimized

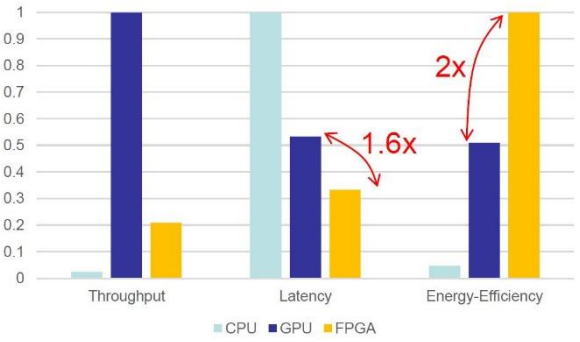


Solution B : latency optimized

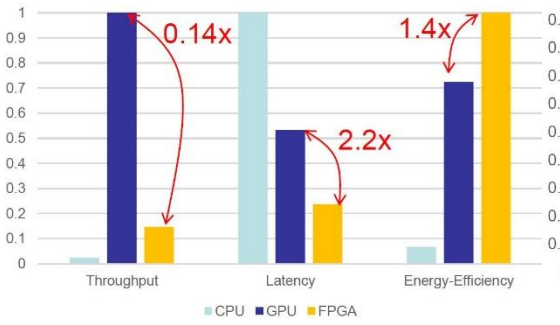


VGG

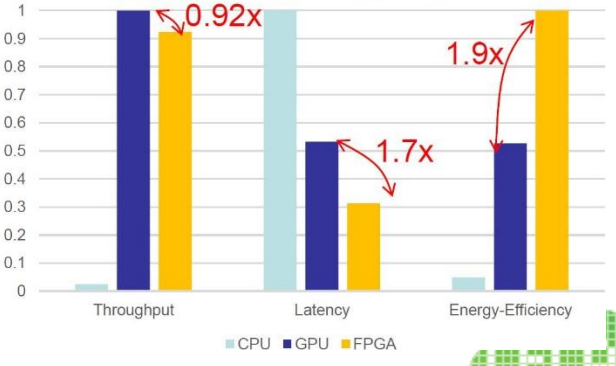
Solution C: energy-efficiency optimized



Solution D: latency & energy optimized



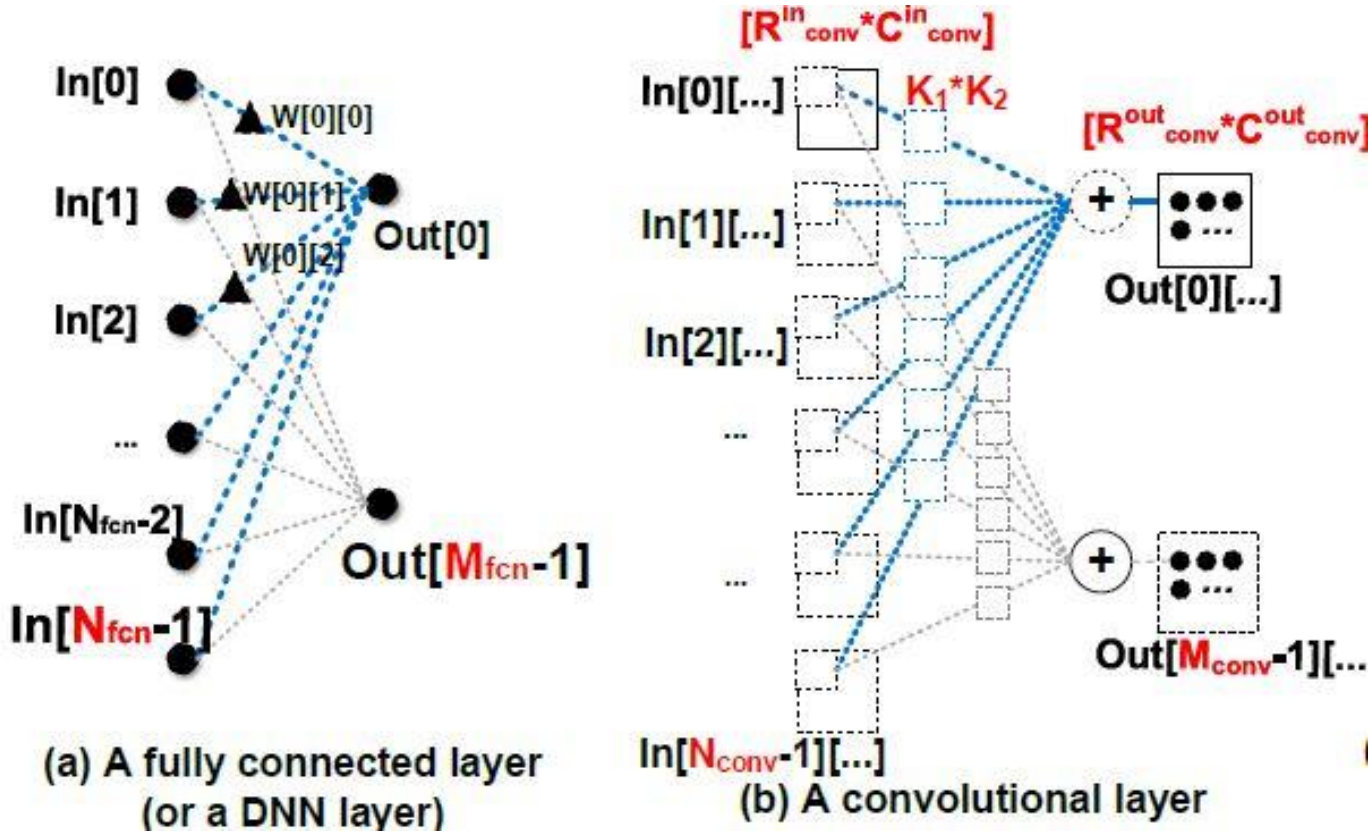
Solution E: throughput & energy optimized



Automated Design Flow

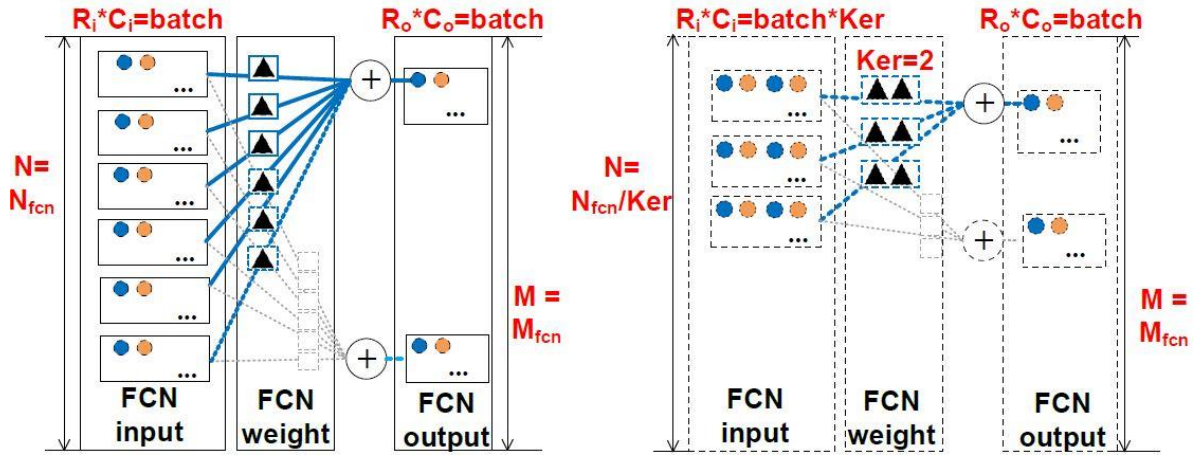
- **Deep CNN Framework**
  - Caffe, Theano, TensorFlow, etc.
  - Software-based
  - Support CPU/GPGPU only
- **FPGA-based Accelerator**
  - Energy-efficiency (GOP/J)
  - Performance (GOP/S)
  - Flexibility
- **Automated Flow**
  - From Model Descriptions(Caffe) to FPGA Accelerators

Layer Representation (FC and Conv)

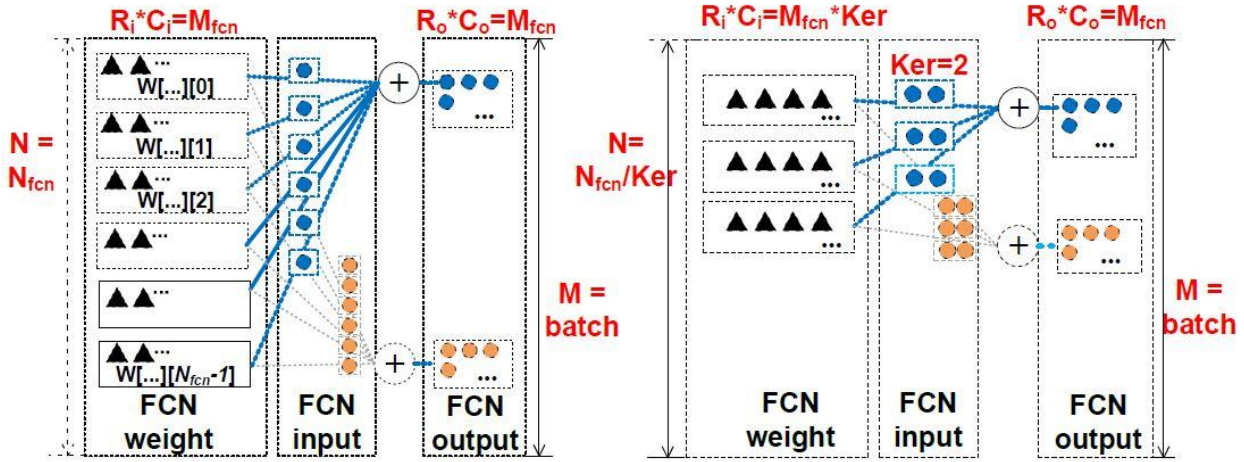


Uniformed Representation

Input-major Mapping

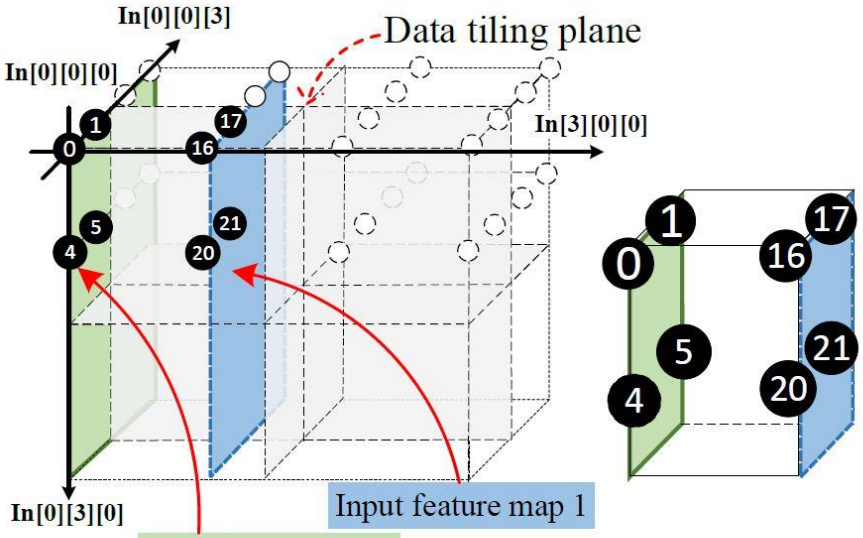


Weight-major Mapping



For small batch sizes, Weight-major mapping is better.

Data Layout Optimization



a) A logical 3D data layout

b) A data tile of input feature maps

Data tile of input feature map 0 is buffered in BRAM Bank 0: 0 1 4 5

Data tile of input feature map 1 is buffered in BRAM Bank 1: 16 17 20 21

c) Physical data layout in on-chip buffer per BRAM bank

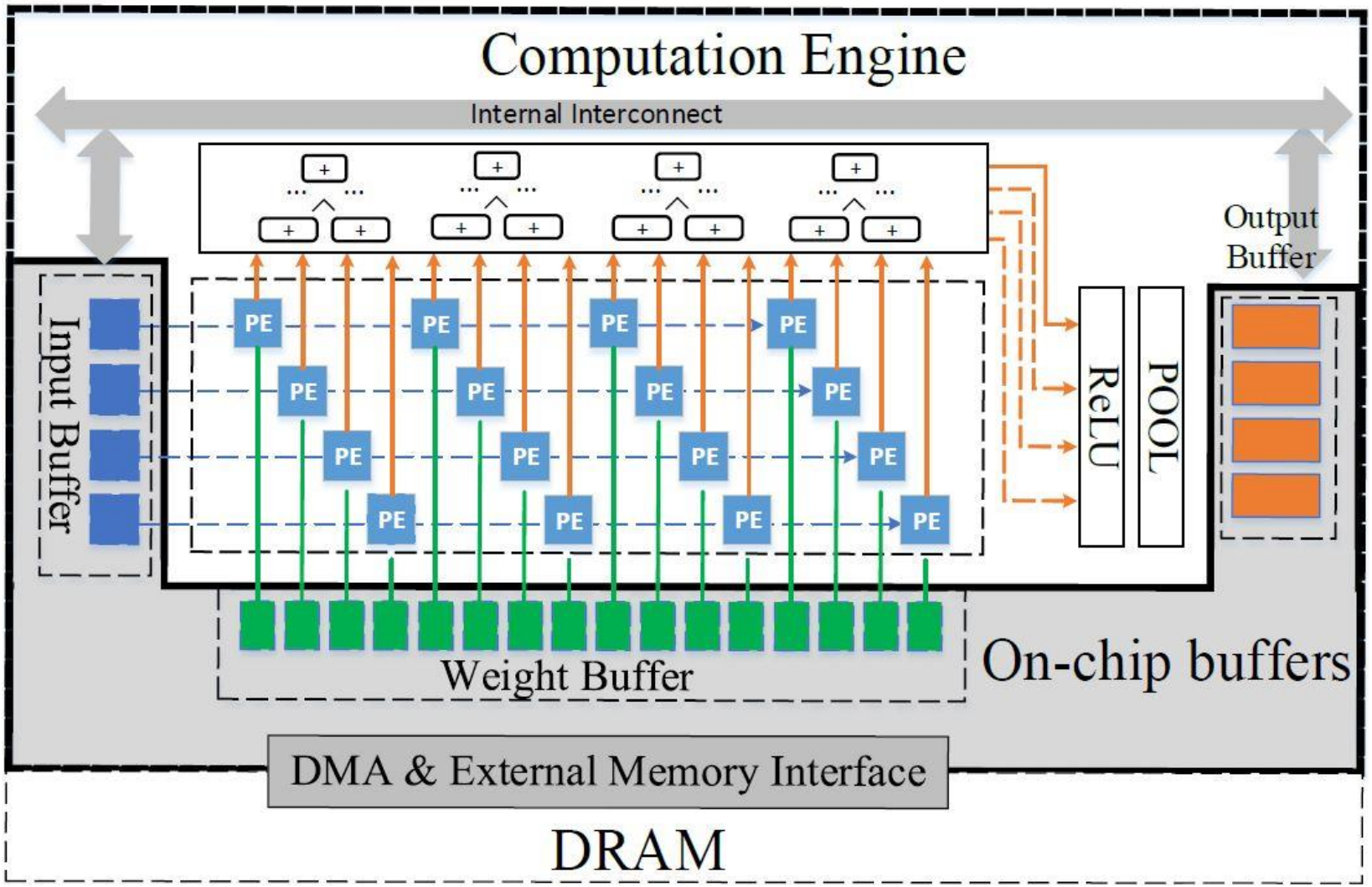
Data	0	1	...	4	5	...	...	16	17	...	20	21
DRAM Addr.	x0	x4	...	x10	x14	...	...	x40	x44	...	x50	x54

d) Row-major data layout in DRAM space

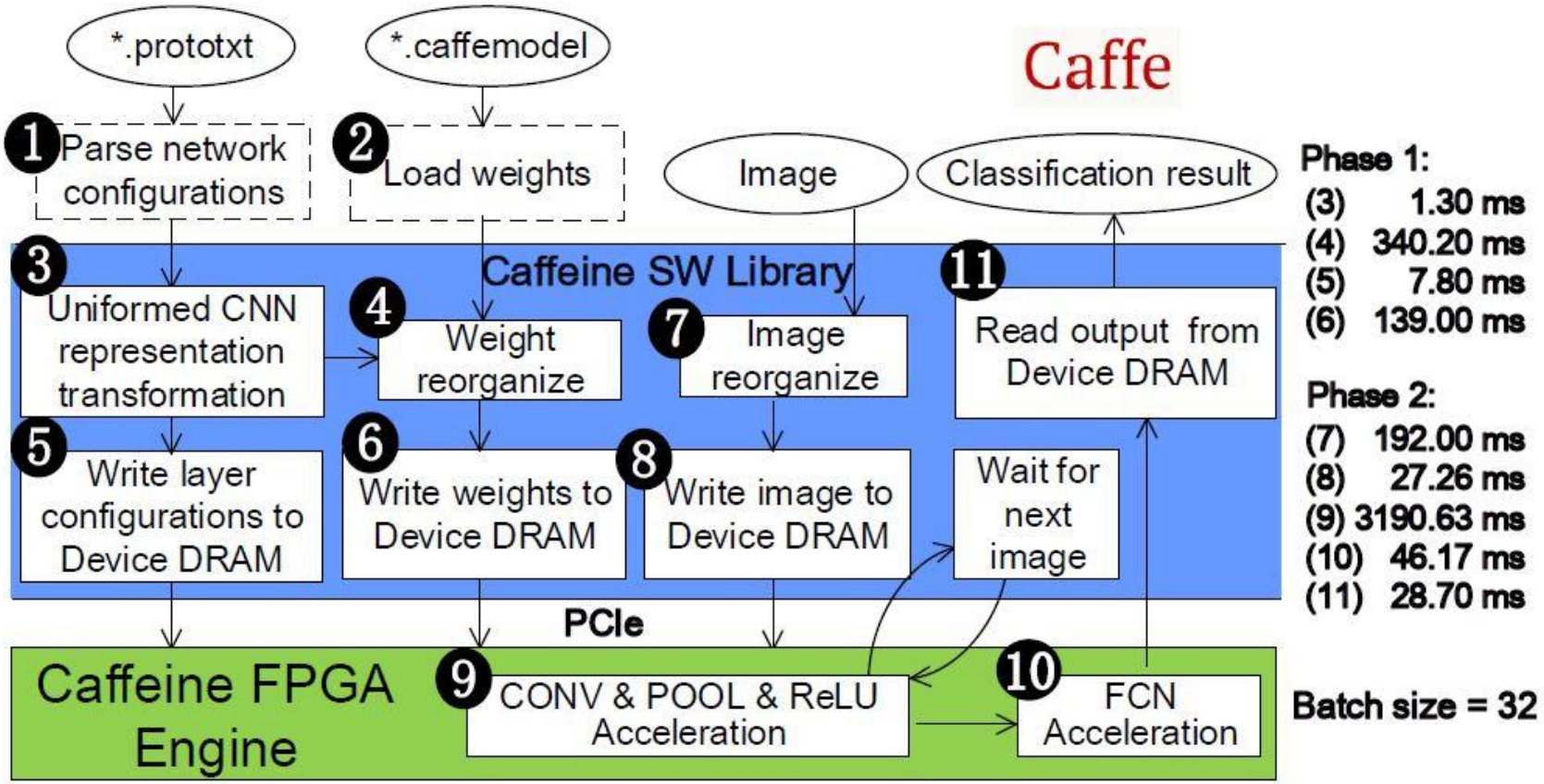
Data	0	16	1	17	4	20	5	21	...	...	...	...
DRAM Addr.	x0	x4	x8	xc	x10	x14	x18	x1c	...	...	...	...

e) Proposed data layout in DRAM space

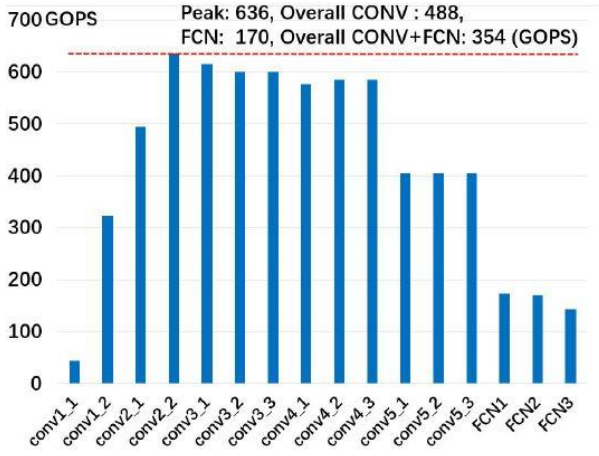
System Overview



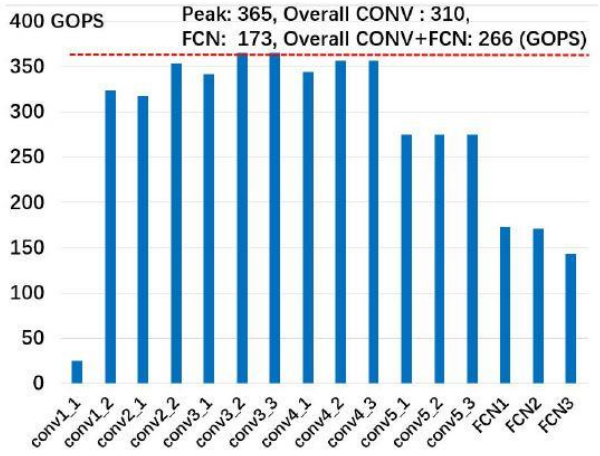
Caffe-Caffeine integration



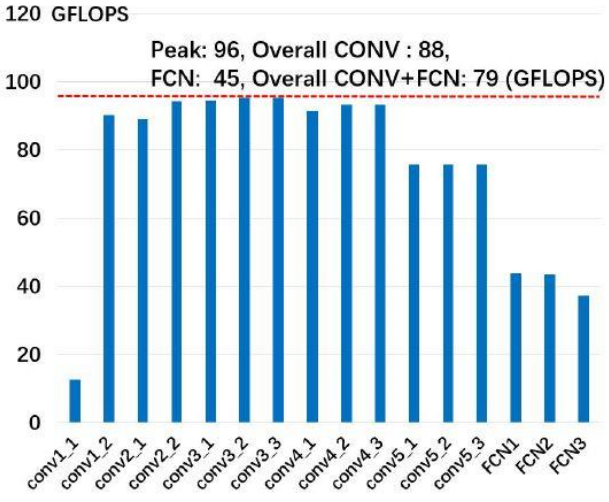
Experimental Results



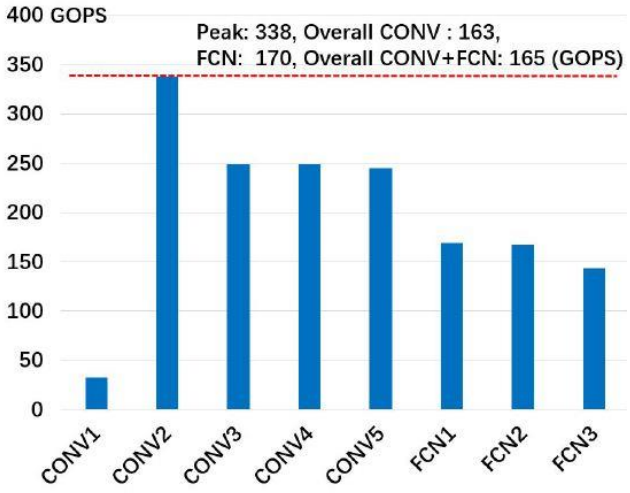
(a) VC709 VGG 16-bit fixed-point



(b) KU VGG 16-bit fixed-point



(c) KU VGG 32-bit floating-point



(d) KU AlexNet 16-bit fixed-point

Cross-platform Comparison

platforms	CPU	CPU+GPU	CPU+FPGA	
Device	E5-2609	K40	KU060	VX 690t
Technology	22nm	28nm	20nm	28nm
Freq.	1.9GHz	1GHz	200MHz	150MHz
Power(Watt)	150	250	25	26
Latency (ms) per img.	733.7	15.3	101.15	65.13
Speedup	1x	48x	7.3x	9.7x
J per image	110	3.8	2.5	1.69
Energy Efficiency	1x	28.7x	43.5x	65x

Conclusions

- **FPGA based CNN accelerators are promising**
- **Design space should be carefully explored on single/multiple FPGA boards**
- **Design automation flow is helpful**

Acknowledgement

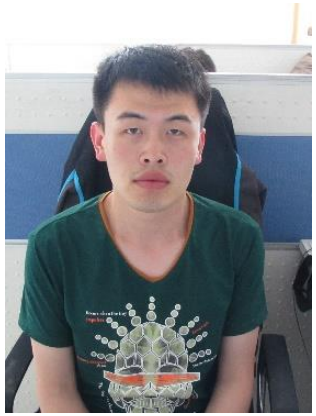
Faculty Members:

Jason Cong
(UCLA)

Guangyu Sun
(PKU)

PhD Students:

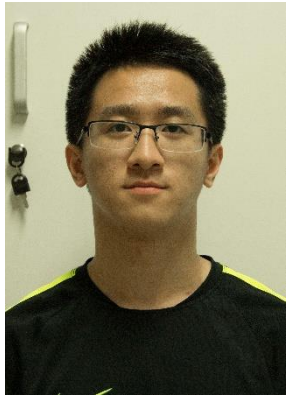
Chen Zhang
(PKU)



Yijin Guan
(PKU)

Bingzhe Wu
(PKU)

Undergraduate Students:



Zhihang Yuan
(PKU)

Yao Fu
(PKU)

Sixue Wang
(PKU)

Thanks!