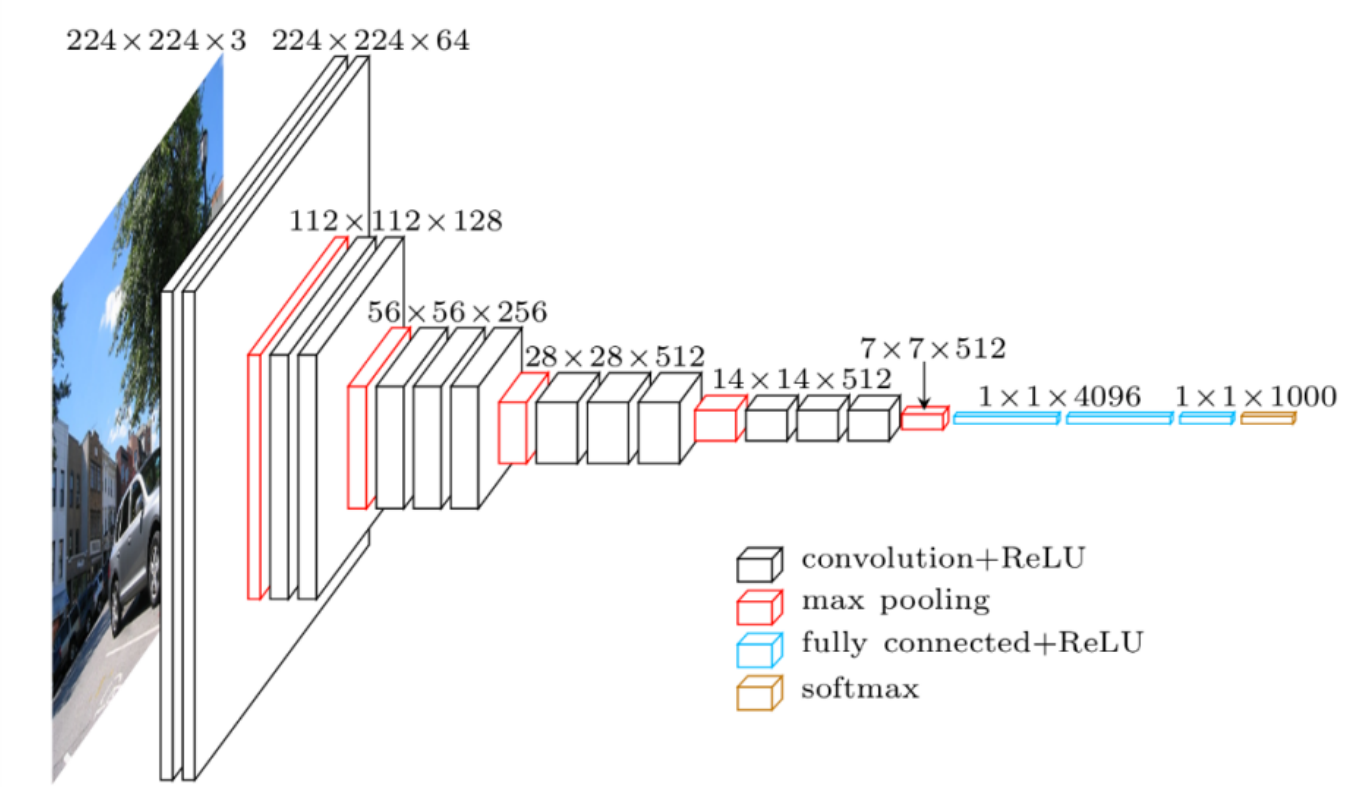
Improving the Performance of OpenCL-based FPGA Accelerator for Convolutional Neural Network

Jialiang Zhang and Jing Li

Outline

- Background
- Motivation
- Balance analysis model
- Proposed design
- Performance
- Conclusion

Convolutional Neural Network

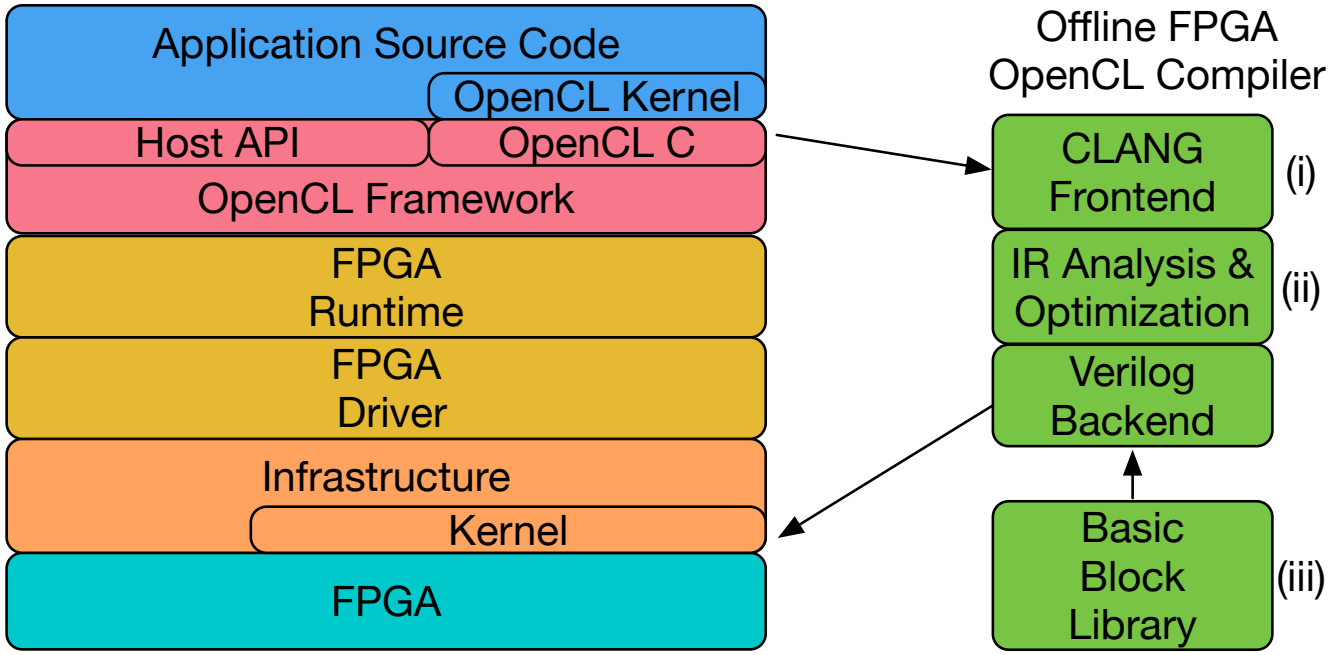


Marco Architecture of VGG

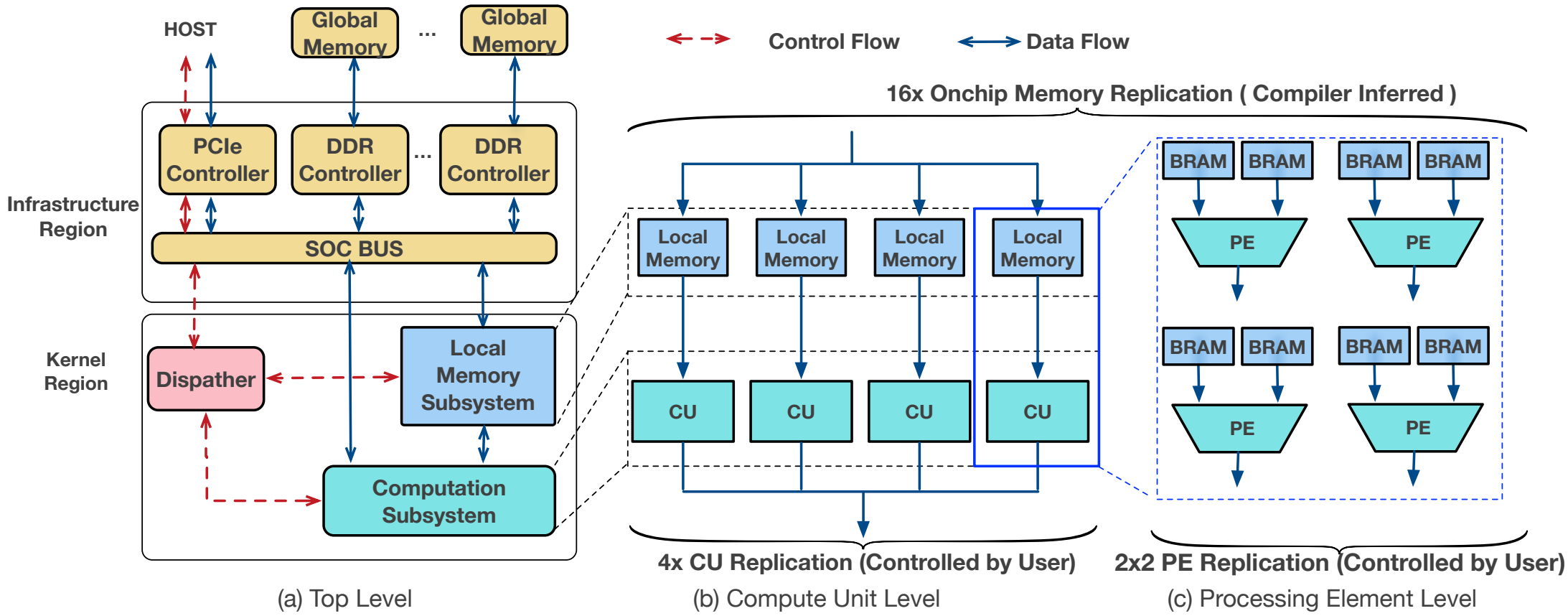
Image from: https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/l

OpenCL Framework

- OpenCL provides a good FPGA abstration
- Unlike OpenCL on GPU or CPU, OpenCL FPGA describes both hardware and software
- Use #prgma to guide hardware generation:
  - loop unrolling; SIMD factor; compute unit replication



OpenCL FPGA Framework



Related works

- Generic CNN accelerators: [1]-[3]
- Balance computation and external memory access: [4] -[7]
- Hardware Abstraction: [7][8]
- **Contribution:**
  - Identify the performance bottleneck in large scale FPGA CNN accelerator is on-chip memory bandwidth
  - A CNN accelerator achieved optimized balance among computation, on-chip memory and external memory access

[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
[5] N. Suda, et al. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks. ISFPGA 2016
[6] J. Qiu, et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. ISFPGA 2016
[7] C. Zhang Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks. ICCAD 2016
[8] H. Sharma, et al. From High-Level Deep Neural Models to FPGAs. MICRO 2016

Motivation

- New FPGA devices have more and faster DSP resources

	28nm	20nm	16/14nm
Amount of DSP	3600	5520	12288
DSP frequency (MHz)	741	741	891
Arith. Perf. (GFLOP/sec)	5335	8180	21897

- We should use all DSP resources every cycle to obtain the GFLOPS number in datasheet

- The existing design cannot utilize the DSP resources well

Balance Analysis Model

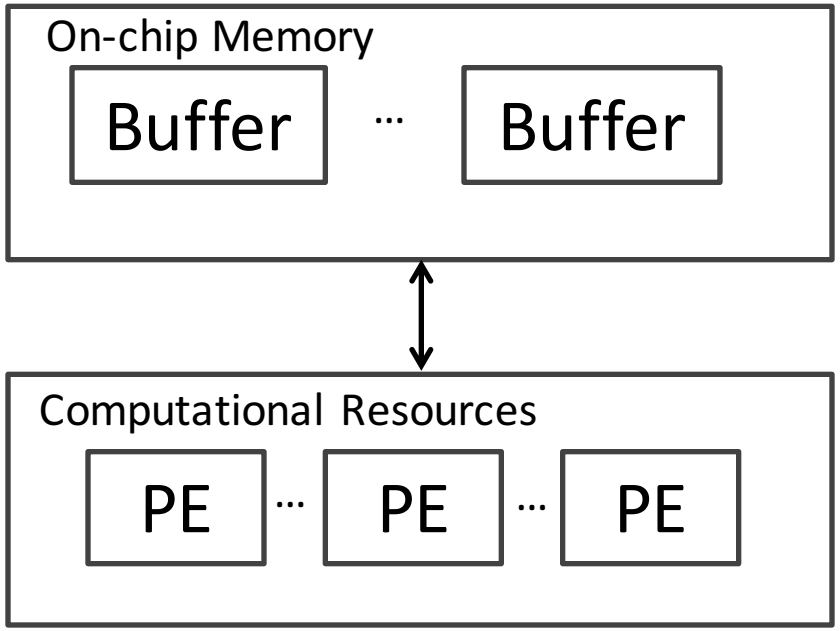
- Balance is the relationship between storage and computation resources
- Assumption:
  - Only one bottleneck in the system
  - Bandwidth bounded
  - Computation and memory access overlap perfectly.

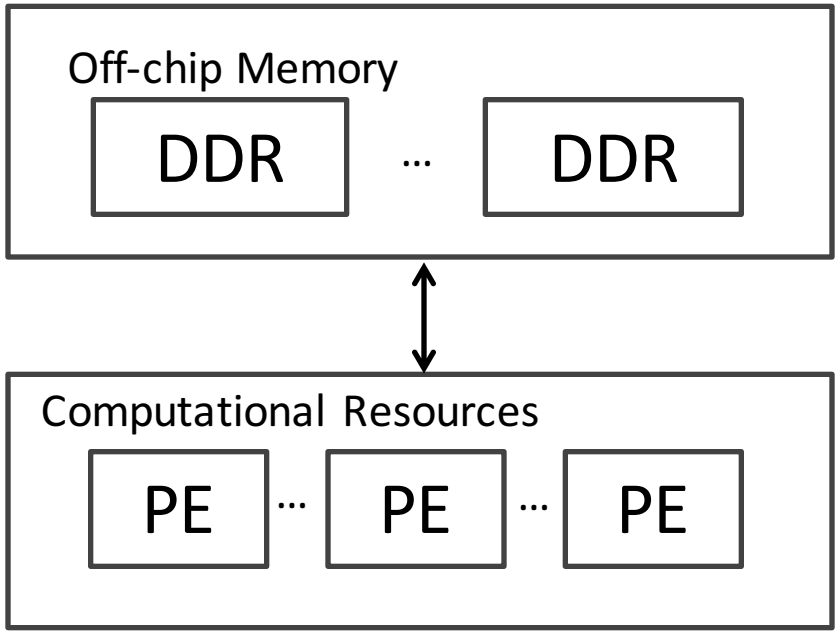
	Hardware determined	Algorithm determined
	Machine Balance	**Code Balance**
On-chip	B_m^{On}	B_m^{On}
Off-chip	B_m^{Off}	B_C^{Off}

Refer to on-chip Mem BW

Refer to off-chip Mem BW

Machine Balance





$$B_m^{on} = \frac{N_{BRAM} * WIDTH_{BRAM} * f_{BRAM}}{N_{DSP} * OPS/DSP * f_{DSP}} = \frac{bw_{max}}{P_{max}}$$

$$B_m^{off} = \frac{N_{DDR} * BW_{DDR}}{N_{DSP} * OPS/DSP * f_{DSP}} = \frac{bw_{max}}{P_{max}}$$

Code Balance

$$B_c^{on} = \frac{\text{Instruction Input } (Words)}{\text{No. of Operations } (Ops)},$$

- On-chip Code Balance is determined by input and output data size of instructions
- For example, MAC operation has a $B_c^{on} = 1$

$$B_c^{off} = \frac{\text{Data Size}}{\text{Total No. of Operation}};$$

- Off-chip Code Balance is determined by the input and output data size of the whole program
- Assume unlimited on-chip memory size

Break the memory Wall

$$P^{\{on,off\}} = \begin{cases} P_{max}, & \frac{B_m^{\{on,off\}}}{B_c^{\{on,off\}}} \geq 1, \quad \text{(comp. bound)}, \\ \frac{bw_{max}^{\{on,off\}}}{B_c^{\{on,off\}}}, & \frac{B_m^{\{on,off\}}}{B_c^{\{on,off\}}} < 1, \quad \text{(mem. bound)}, \end{cases}$$

$$P = \min\left(P^{on}, P^{off}\right) = \min\left(\frac{bw_{max}^{on}}{B_c^{on}}, \frac{bw_{max}^{off}}{B_c^{off}}, P_{max}\right),$$

On-chip Mem BW Bounded

Off-chip Mem BW Bounded

Comp. Bounded

To achieve P_{max} , we need to satisfy $B_m^{Off} > B_c^{Off}$ and $B_m^{On} > B_c^{Off}$

Off-chip balance

Layer	B_C^{Off}
CONV1	0.0021
CONV2	0.0010
CONV3	0.00062
CONV4	0.00087
CONV5	0.00277
FC	0.5

Technology Node	B_m^{Off}
28nm	0.168
20nm	0.217
14/16nm	0.177
14/16nm w/ HBM	0.177

- $\boldsymbol{B_C^{Off} < B_m^{Off}}$ for convolution layers
- $\boldsymbol{B_C^{Off} > B_m^{Off}}$ for FC layer, but only contribute a small portion of computation

On-chip balance

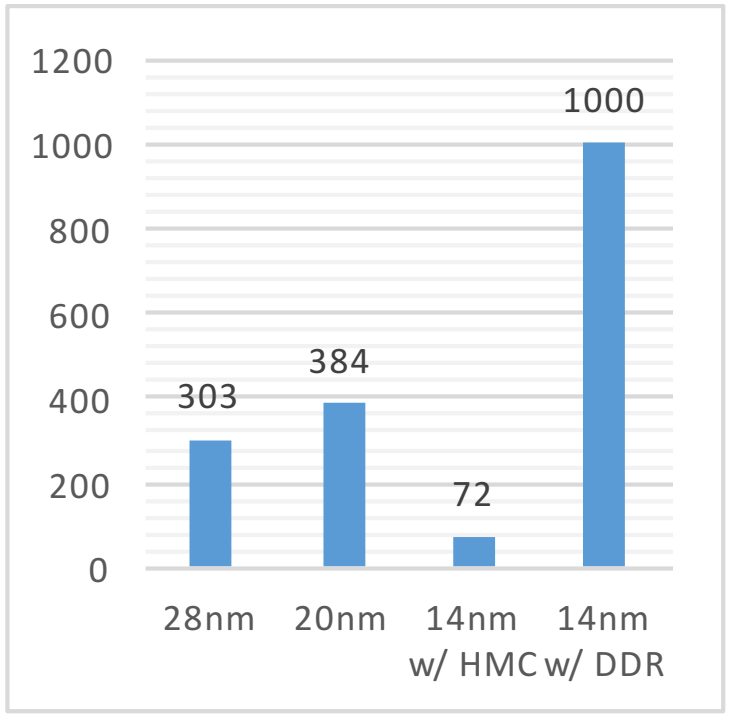
Layer	B_C^{On}
CONV1	1
CONV2	1
CONV3	1
CONV4	1
CONV5	1
FC	1

Technology Node	B_m^{On}
28nm	0.0033
20nm	0.0026
14/16nm	0.0010

- $\boldsymbol{B_C^{On} > B_m^{On}}$ for all layer
- On-chip memory bandwidth becomes the bottle neck
- We need to increase $\boldsymbol{B_m^{On}}$

Data Reuse Requirement

$\frac{B_c^{on}}{B_m^{off}}$



- The balance analysis assumes unlimited on-chip memory size (load data once)
- Under limited on-chip memory capacity, we need to load data more than once.
- To satisfied external memory bandwidth requirement, we need to reuse data at least $\frac{B_c^{on}}{B_m^{off}}$.

Optimization Direction

- Increase B_m^{on} to utilize all DSP resources

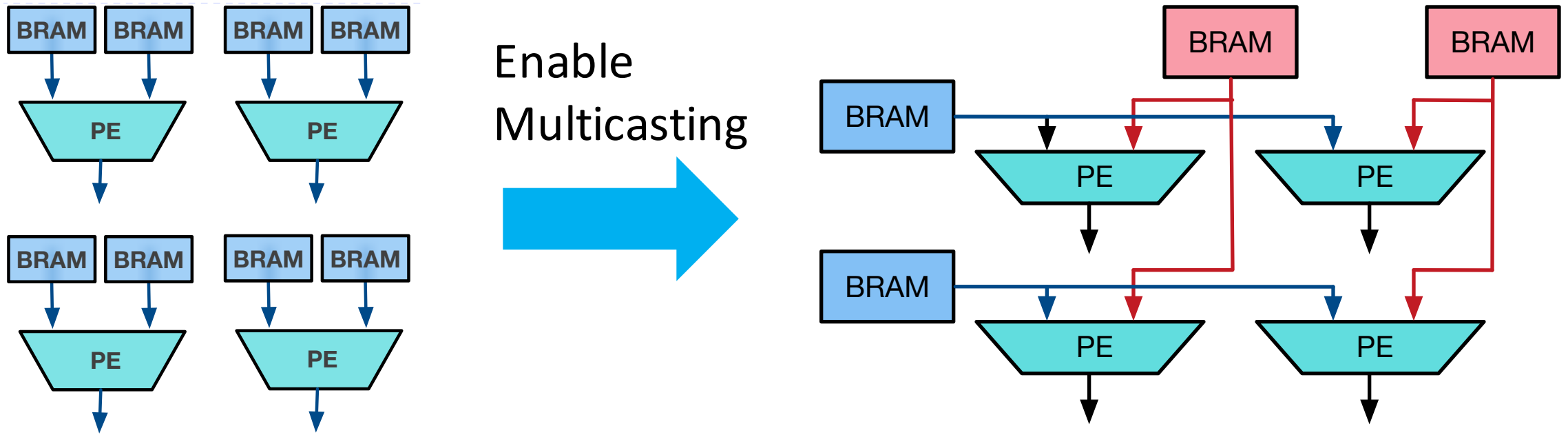
- Satisfy the data reuse requirement with limited on-chip memory size.

Optimization Direction

- Increase B_m^{on} to utilize all DSP resources

- Satisfy the data reuse requirement with limited on-chip memory size.

Matrix Multiplication Kernel



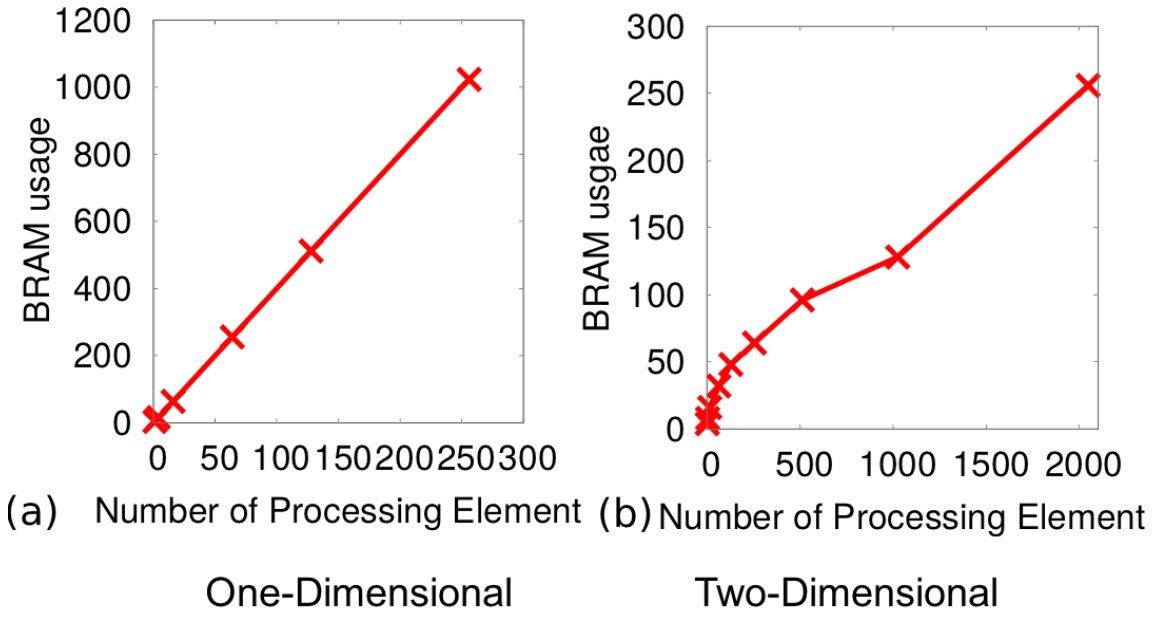
$$(AB)_{ij} = \sum_k A_{ik} B_{kj} \qquad \forall i < m, j < n$$

BRAM usage reduction

$$B_m^{on\prime} = \sqrt{N_{DSP}} \cdot \frac{N_{BRAM} \cdot BW_{BRAM}}{N_{DSP} \cdot f_{BRAM}} = \sqrt{N_{DSP}} B_m^{on}$$

By multicasting, we can increase on-chip machine balance by $\sqrt{N_{DSP}}$

Experiment on Arria10 GX1150 FPGA (w/ 1518 DSP and 2713 M20k Memory):

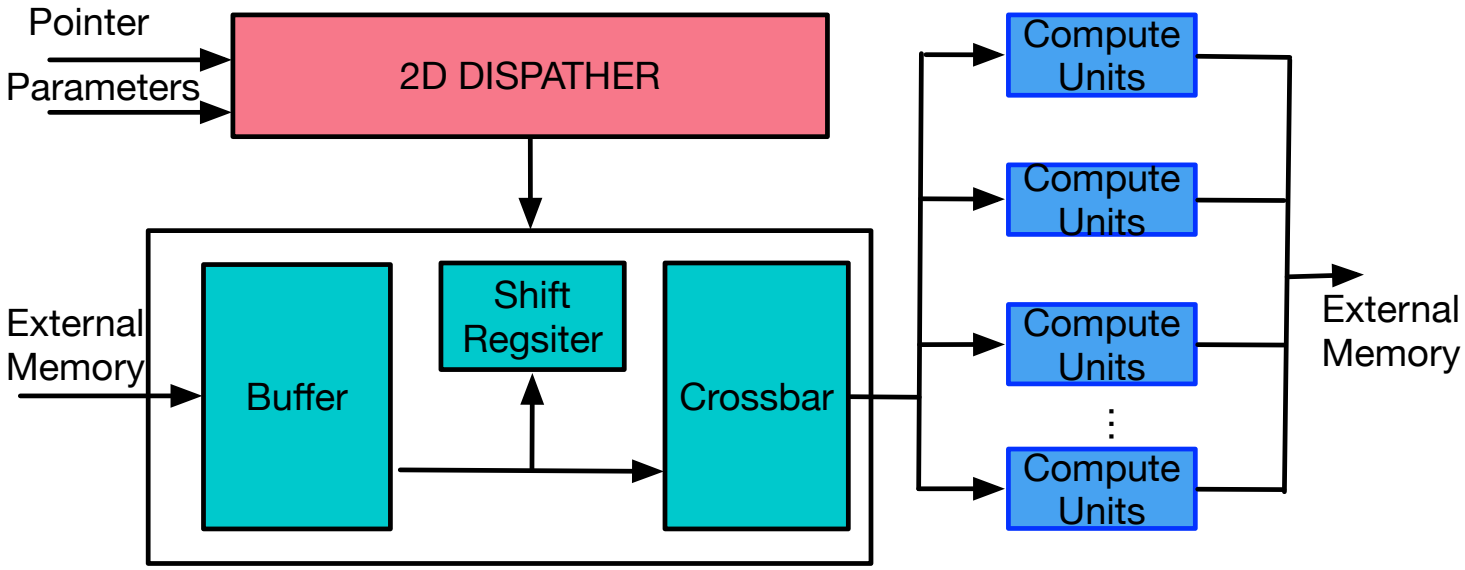


Optimization Direction

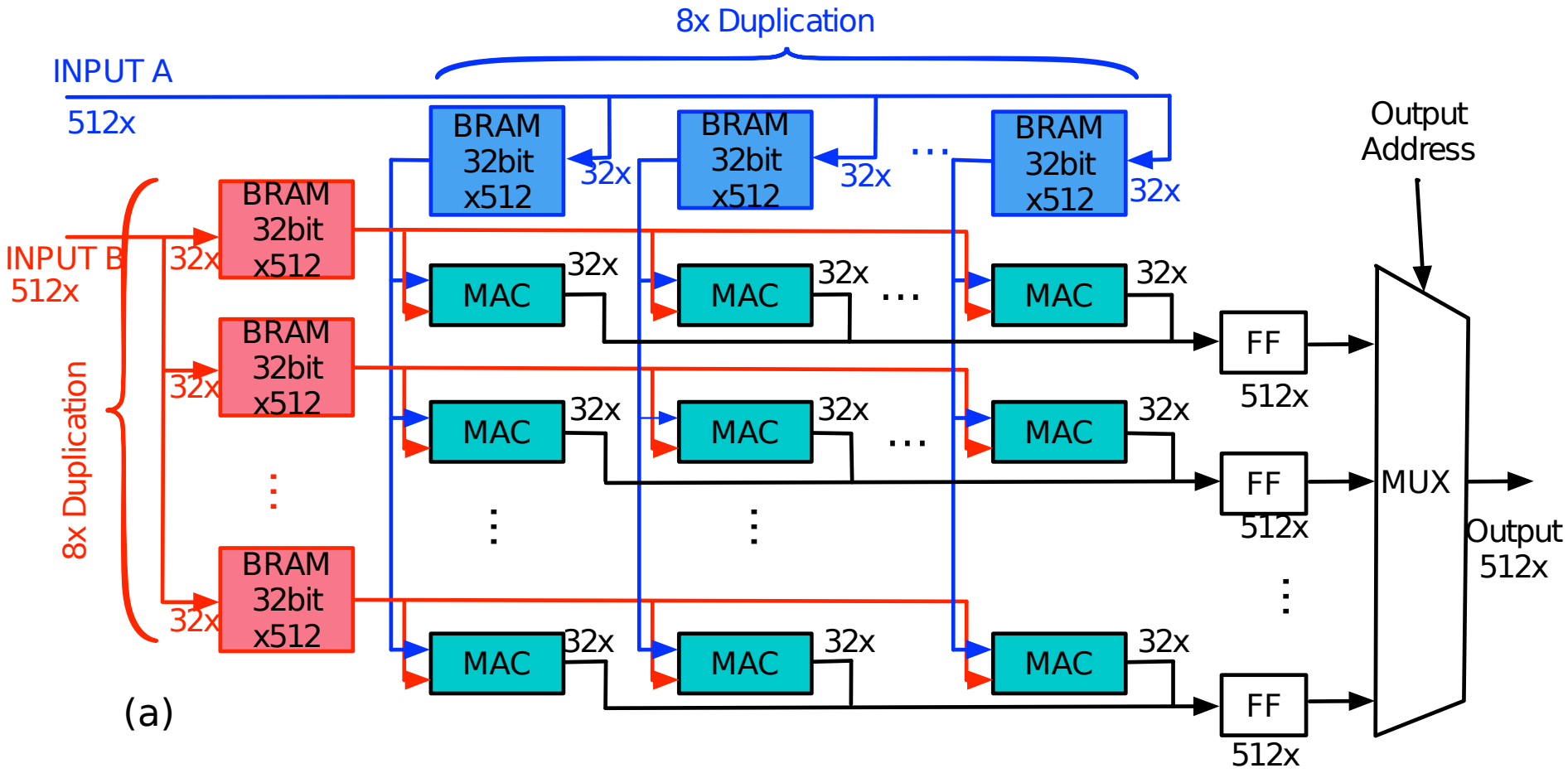
- Increase B_m^{on} to utilize all DSP resources

- Satisfy the data reuse requirement with limited on-chip memory size.

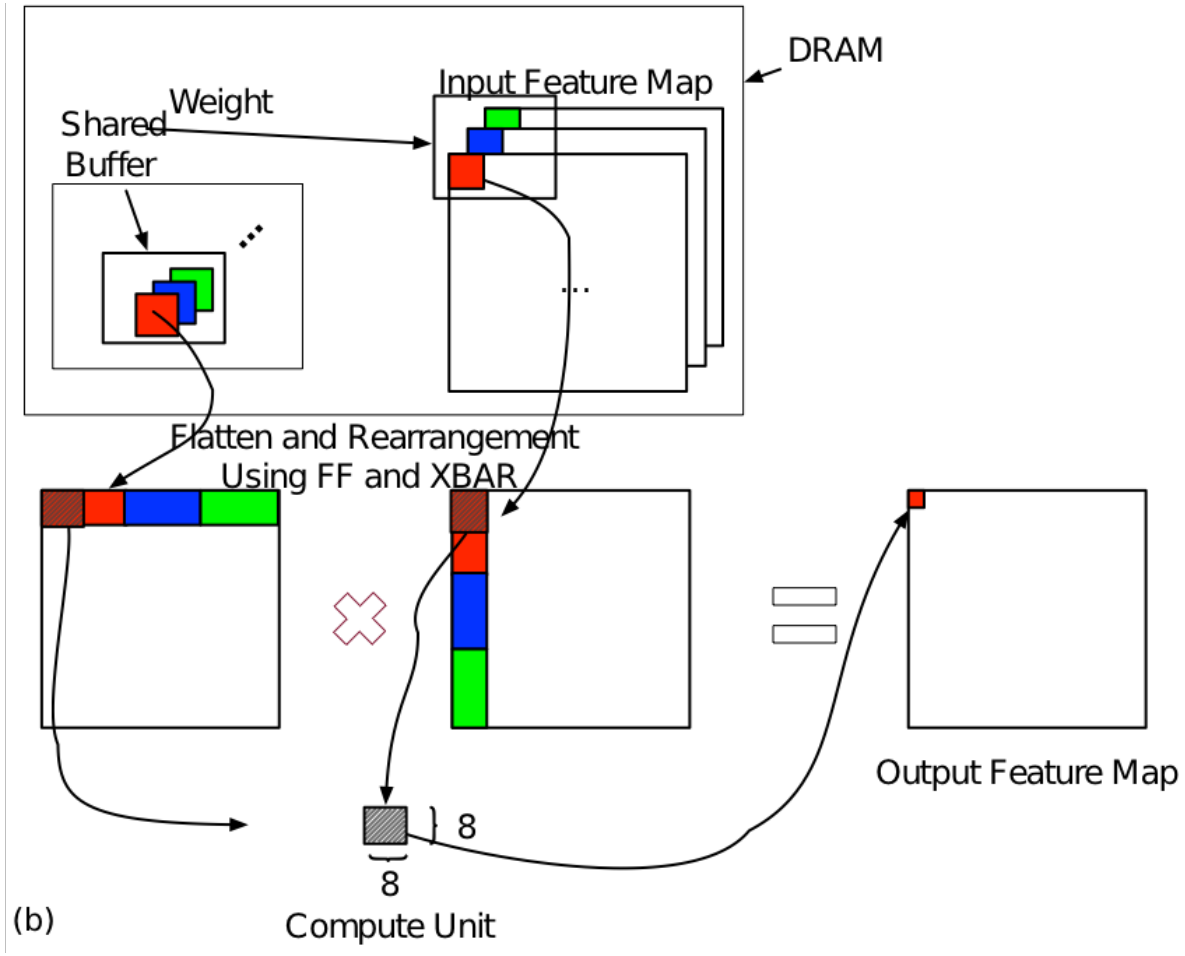
Kernel Design

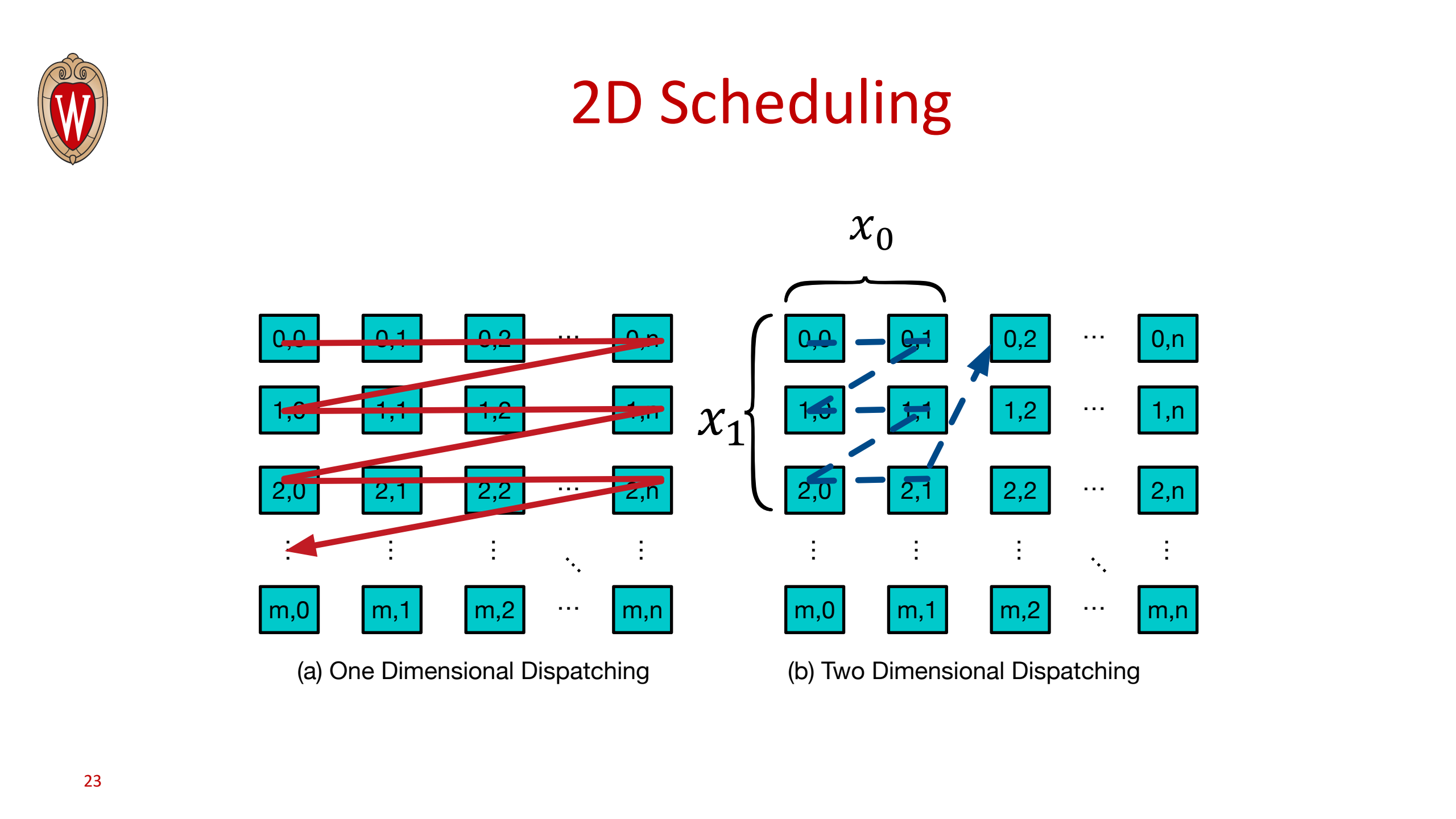


CU Design


8x Duplication
INPUT A
512x
BRAM
32bit
x512
32x
INPUT B
512x
8x Duplication
MAC
FF
512x
Output
Address
MUX
Output
512x
(a)

Work-item Scheduling





2D Scheduling

(a) One Dimensional Dispatching

(b) Two Dimensional Dispatching

Minimizing external memory bandwidth

$$\min_{x_0,x_1,x_2} \quad a\frac{x_0}{x_1} + b\frac{x_0}{x_2} + c\frac{1}{x0},$$

$$s.t. \quad x_0x_1 + x_1x_2 < size_{on-chip},$$

$$a = \frac{N_{of} \cdot size_{if}^2 \cdot}{size_k^2},$$

$$b = N_{of} \cdot size_{if}^2,$$

$$c = N_{if} \cdot size_k^2 \cdot size_{of}^2 \cdot N_{of},$$

- Objective:
  - Find the optimized scheduling policy
  - To minimize external memory bandwidth
- Constraint:
  - On-chip memory size
- Based on:
  - CNN layer parameters

Optimization result

Layer	Optimal $<x_1, x_2>$	Req. of DRAM BW with optimal 2-D scheduling	Req. of DRAM BW with 1-D scheduling
CONV1	<6,13>	14.5GB/s	123.9GB/s
CONV2	<6,4>	10.8GB/s	89.8GB/s
CONV3	<5,3>	11.5GB/s	92.6GB/s
CONV4	<7,9>	10.7GB/s	82.1GB/s
CONV5	<4,5>	11.6GB/s	94.6GB/s

The optimization can effectively reduce the requirement of off-chip memory bandwidth

Experiment Setup

Arria 10 GX FPGA Development Kit

- FPGA: Arria10 GX1150
- 1518 DSPs
- 2713 M20k memory
- 1GB DDR4 with 17GB/s BW

- Kernel implemented as an OpenCL IP library package using Verilog

Implementation Result

	Total	Ours	Percentage
DSP	1518	1320	86
BRAM	2713	1250	46
Logic	1506k	437k	43
Frequency		370 MHz	

VGG Performance

Layers	Number of Ops	Duration (ms)	Performance GOP/s
Conv	30.69G	11.9	2568
FC	0.073G	5.5	13
Total	30.76G	17.4	1790

Overall VGG Classification Throughput: 57 image/s

Performance Comparison

	Suda et. al [1]	Qiu et. al [2]	Ours
Platform	Stratix V GSD8	Zynq XC7Z045	Arria 10 GX1150
Performance(GOP/s)	117.8	136.9	1790
Performance Density (OPs/DSP/Cycle)	0.36	1.17	3.06
Power efficiency (GOP/J)	1.84	14.22	47.88

[1] N. Suda, et al. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks. ISFPGA 2016
[2] J. Qiu, et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. ISFPGA 2016

Summary

- Motivation
- Balance analysis model
- Our Work:
  - Multicasting among PEs
  - 2D Scheduling
- Performance comparison

Thanks!

Q&A