USC University of Southern California

Frequency Domain Acceleration of Convolutional Neural Networks on CPU-FPGA Shared Memory System

Chi Zhang, Viktor K Prasanna
University of Southern California
{zhan527, prasanna}@usc.edu
fpga.usc.edu

ACM FPGA, Feb. 2017

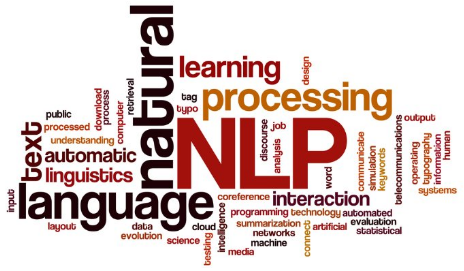
Outline

- Motivation
- Target platform and problem definition
- Main approach
- Overall system design
- Experiments and Results

Outline

- Motivation
- Target platform and problem definition
- Main approach
- Overall system design
- Experiments and Results

Motivation



Natural Language Processing

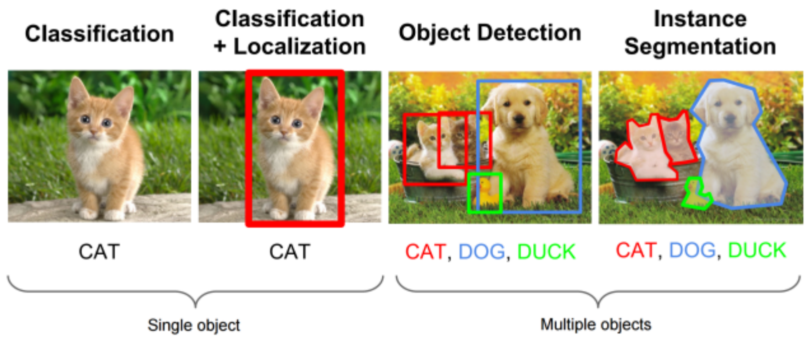


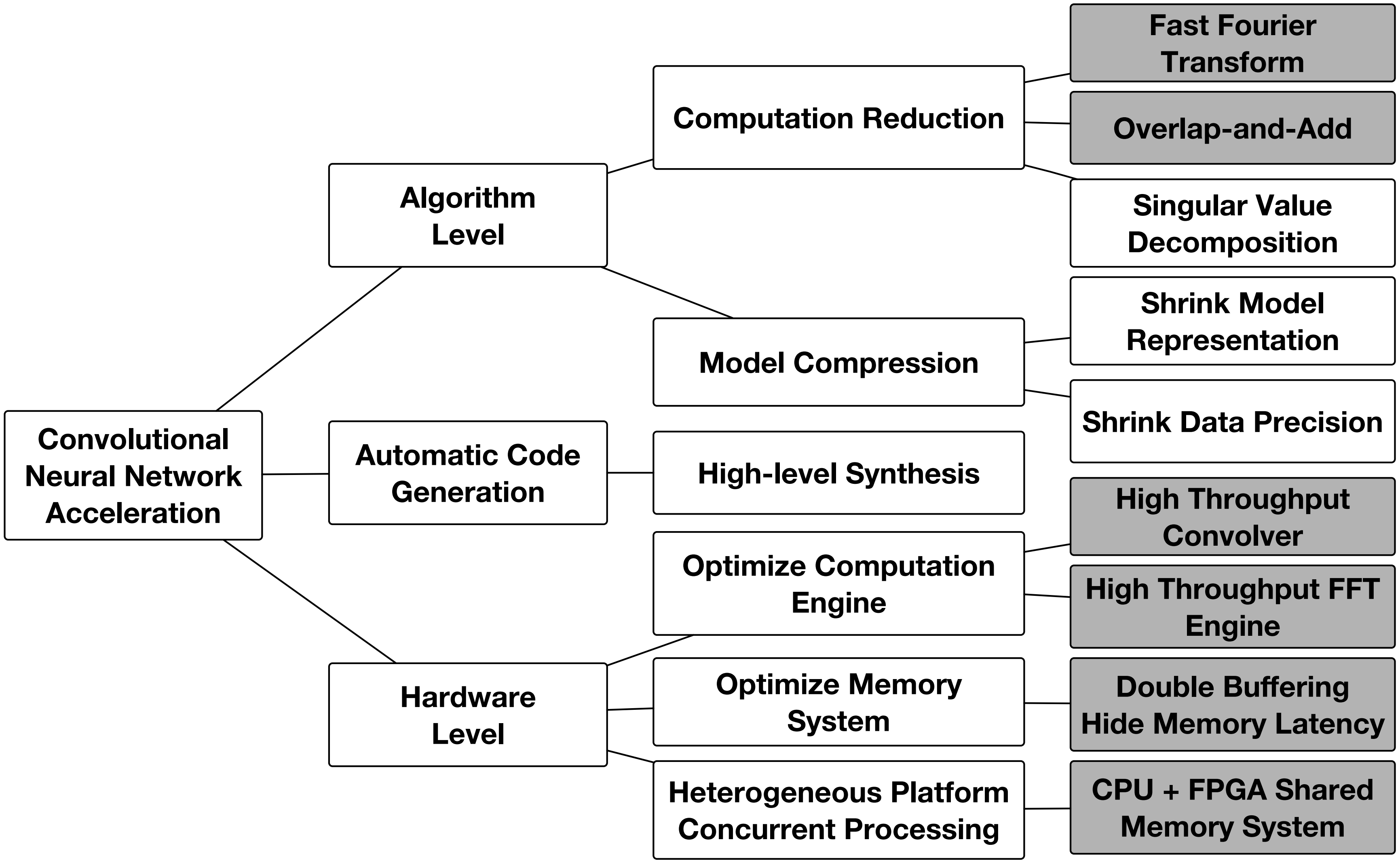
Image Processing and Computer Vision

Speech Recognition

Self-driving Cars

Playing Games

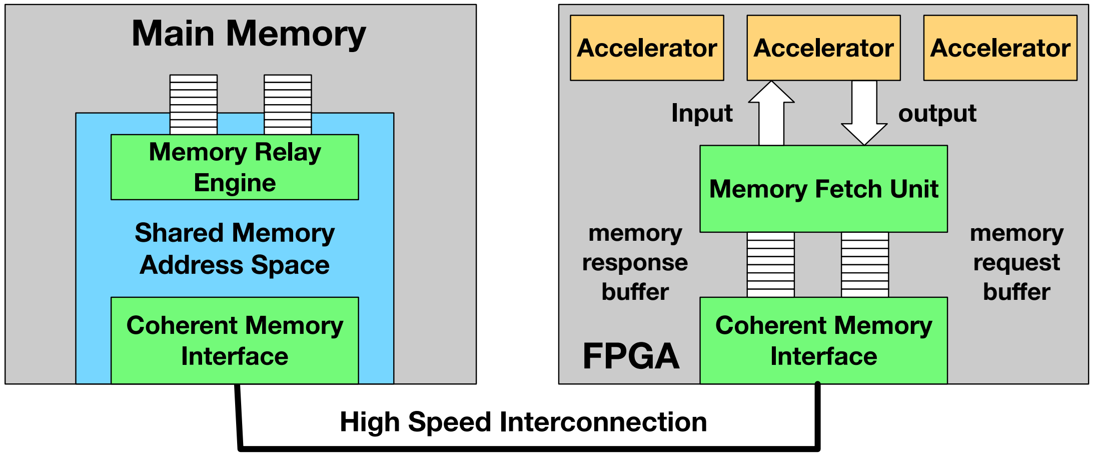
Related Work


Convolutional Neural Network Acceleration
Algorithm Level
Automatic Code Generation
Hardware Level
Computation Reduction
Model Compression
High-level Synthesis
Optimize Computation Engine
Optimize Memory System
Heterogeneous Platform Concurrent Processing
Fast Fourier Transform
Overlap-and-Add
Singular Value Decomposition
Shrink Model Representation
Shrink Data Precision
High Throughput Convolver
High Throughput FFT Engine
Double Buffering Hide Memory Latency
CPU + FPGA Shared Memory System

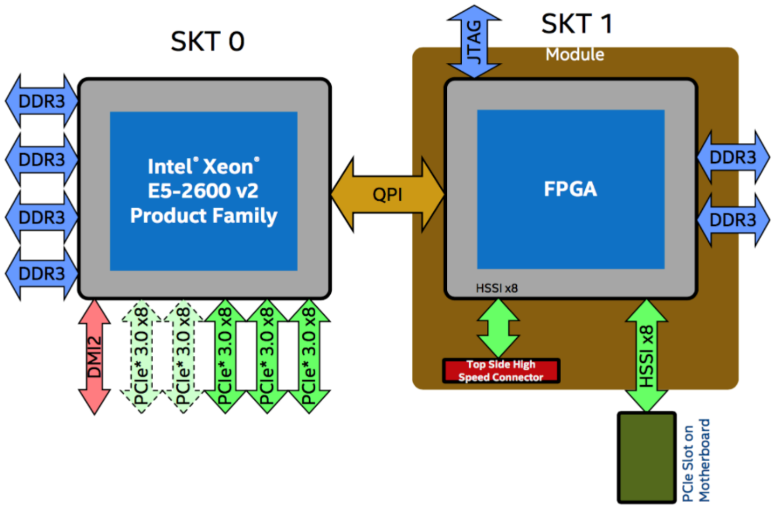
Outline

- Motivation
- Target platform and problem definition
- Main approach
- Overall system design
- Experiments and Results

Target Platform: CPU+FPGA+ Shared Memory



CPU-FPGA Shared Memory Model



Intel QuickAssist QPI-FPGA Platform

Key Attributes:

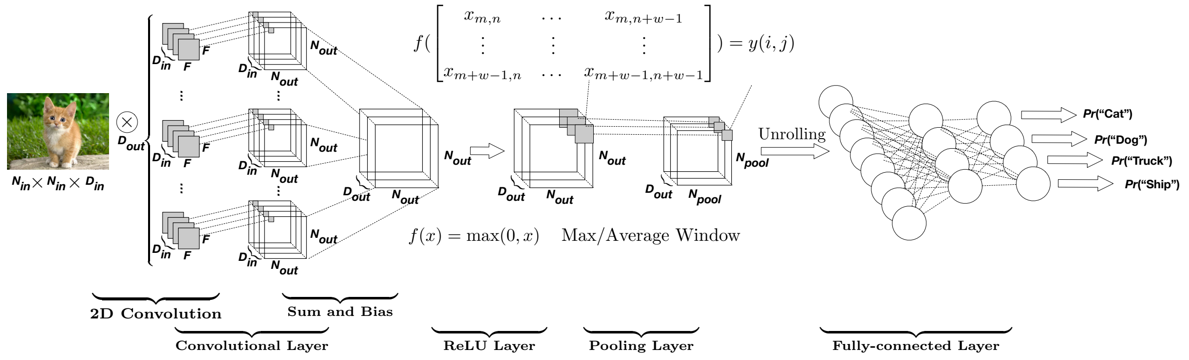
1. CPU threads and FPGA share ***large*** virtual address space.
2. Cache line accessible through high speed interconnection.

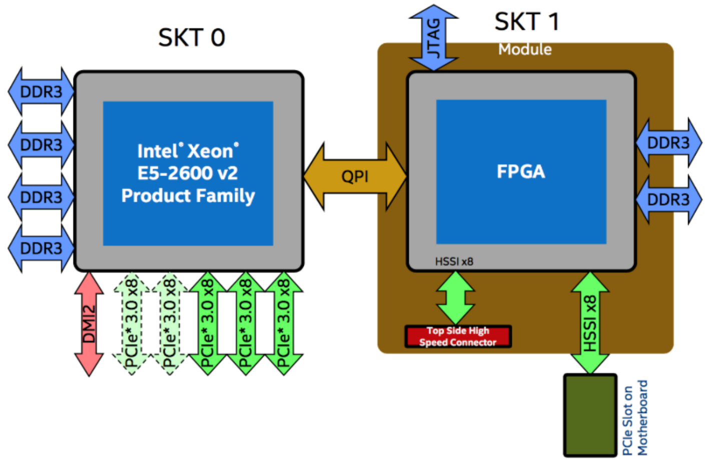
Big advantage for streaming applications; overlap data transfer and processing times => **P-M software pipeline**

USC School of Engineering

Problem Definition

- Given an ***arbitrary trained*** CNN architecture and resource constrains, map the ***forward path*** (inference) onto the CPU-FPGA shared memory system, such that the ***total execution time*** is minimized.



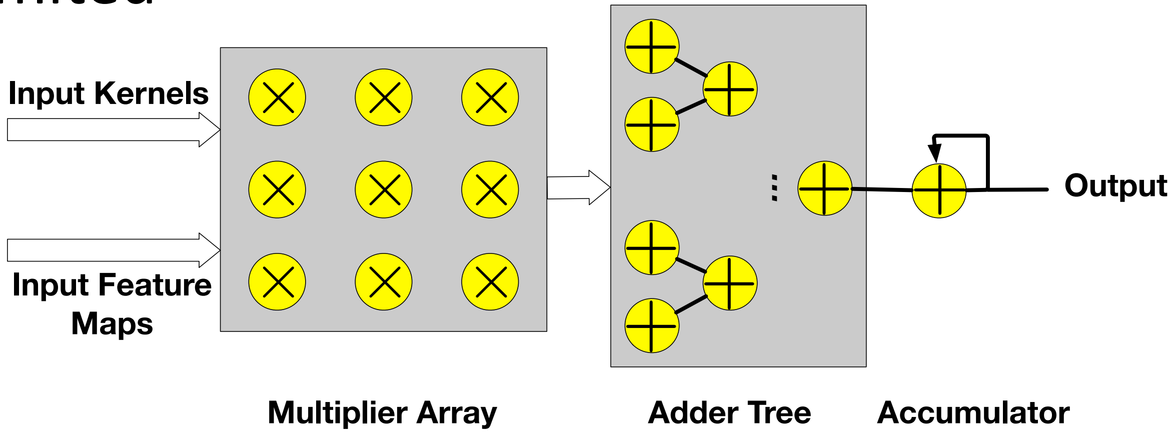


Outline

- Motivation
- Target platform and problem definition
- Main approach
- Overall system design
- Experiments and Results

State of the art on FPGA

- Convolutional layer occupies over 90% of the total computation time. Most previous work focus on using highly optimized hardware to accelerate the convolutional layer.
- Direct convolution: Adder Tree based convolver
  - Data parallelism is limited by the tree structure
  - Throughput is limited

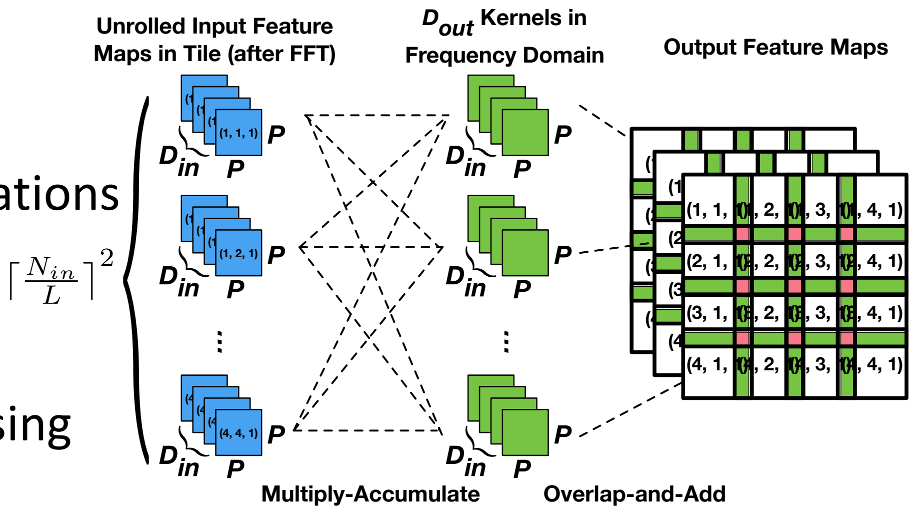


FFT based approach

- 2D convolution can be computed efficiently using FFT.
- In signal processing, Overlap-and-Add (OaA) is an efficient way to compute the discrete convolution of a long signal with a small kernel.
- Parameter definition
  - Input feature map: $N_{in} \times N_{in} \times D_{in}$
  - Output feature map: $N_{out} \times N_{out} \times D_{out}$
  - Kernel: $F \times F \times D_{in} \times D_{out}$
  - FFT size: P, tile size: L

Convolutional Layer in Frequency Domain

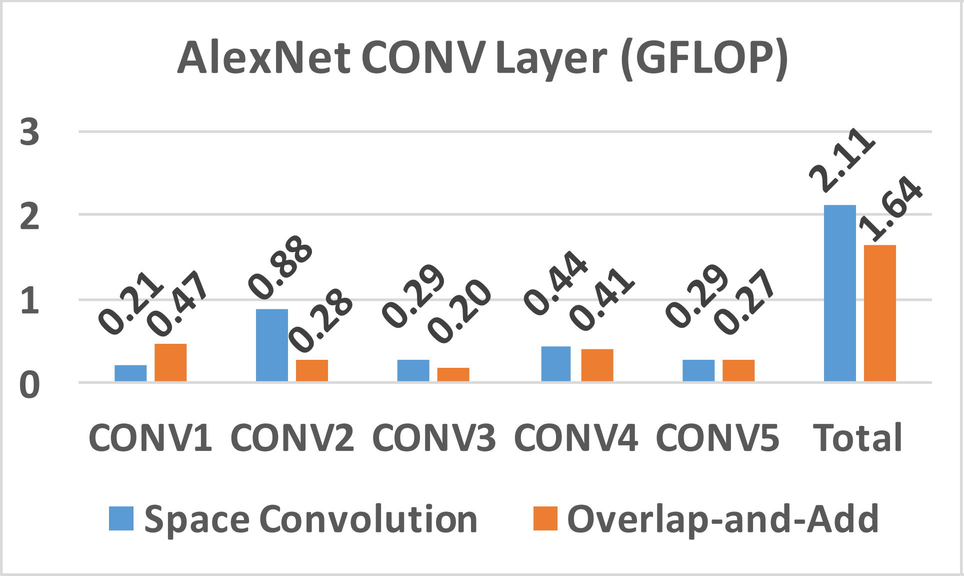
- Overlap-and-Add (OaA)
  - Partition the input feature map to $L \times L$ tiles
  - Perform 2D FFT on each $L \times L$ tile of input feature maps and kernels
  - Perform Hadamard (Dot) Product of each input feature map tile and corresponding kernel map
  - Sum the result and perform 2D IFFT on each output feature map tile
  - Perform Overlap-and-Add on contiguous tiles with stride $F - 1$, where F is the kernel size
- Advantages
  - Reduces the total number of operations
  - Increases the data parallelism
  - Easy to store each tile on FPGA BRAM for high throughput processing

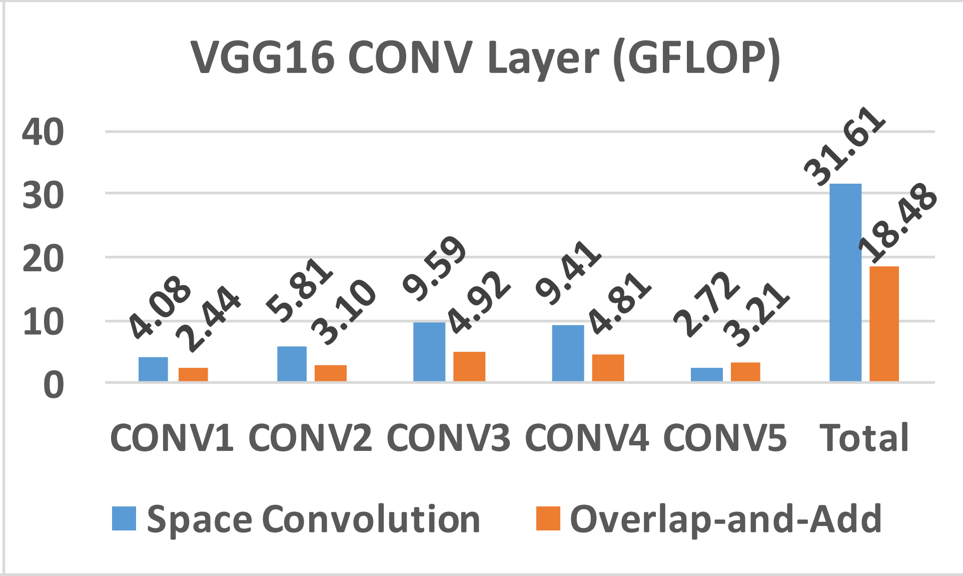


Algorithmic Analysis

- Theoretical computational complexity
  - Direct convolution: $O(N^2F^2)$
  - OaA-based convolution: $O(N^2 \log F)$
- Reduced computation complexity leads to
  - ***less execution time***
  - ***less energy consumption***
  - FFT parallelization => further ***reduction in the overall execution time.***

Computational Complexity Comparison





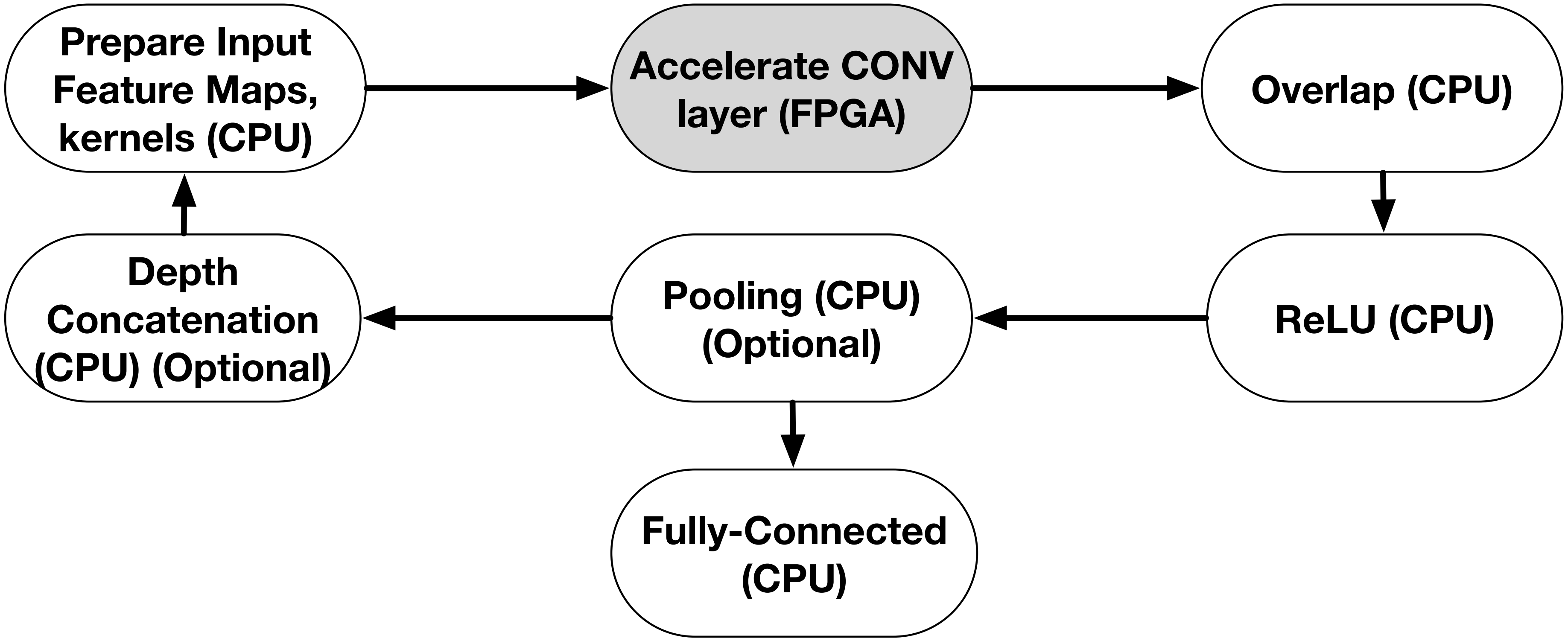
- GFLOP Reduction: AlexNet: 22.12%, VGG16: 41.5%
- The FFT size to achieve the minimum computational complexity for each layer is different.

USC School of Engineering

Mapping Choices on CPU and FPGA

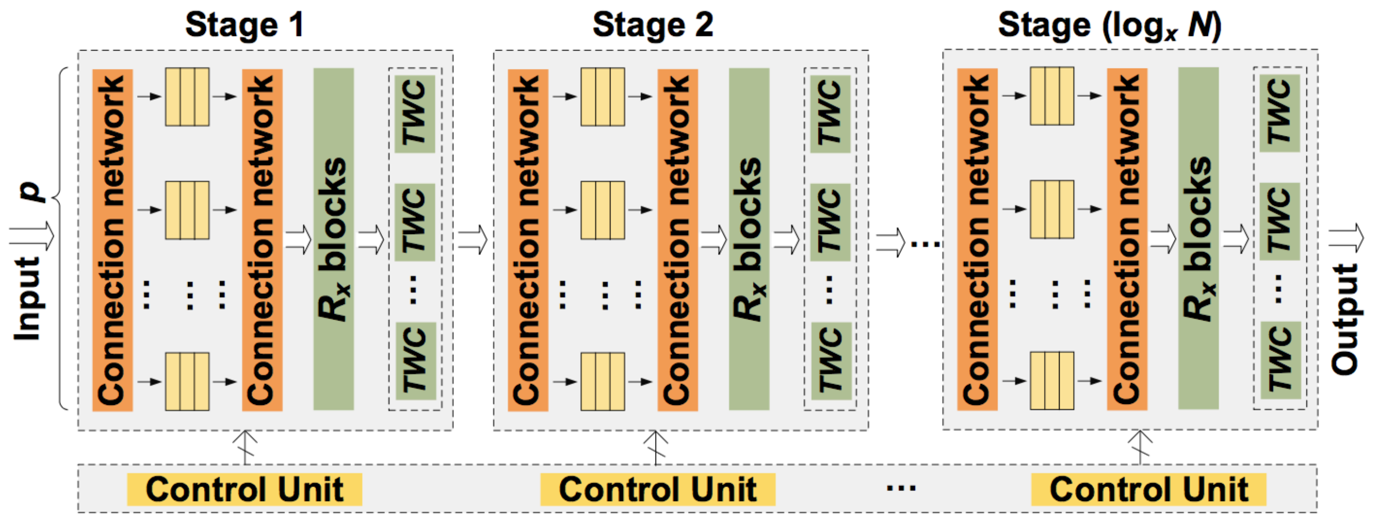
- Convolutional layer occupies over 90% of the total computation time.
- The 2D convolution operation can be fully pipelined and parallelized when transformed into frequency domain.
- Convolutional layer is computation bound (instead of memory bandwidth bound).
- The data transfer overhead for ReLU and Pooling layer overwhelm the acceleration using FPGA.
- If the kernel size across all CONV layers is the same, we can put ReLU and pooling layer right after CONV layer. However, the hardware complexity to build a ***generic*** accelerator for all layers on FPGA is too high.

Mapping Choices on CPU and FPGA


Prepare Input Feature Maps, kernels (CPU)
Accelerate CONV layer (FPGA)
Overlap (CPU)
ReLU (CPU)
Pooling (CPU) (Optional)
Depth Concatenation (CPU) (Optional)
Fully-Connected (CPU)

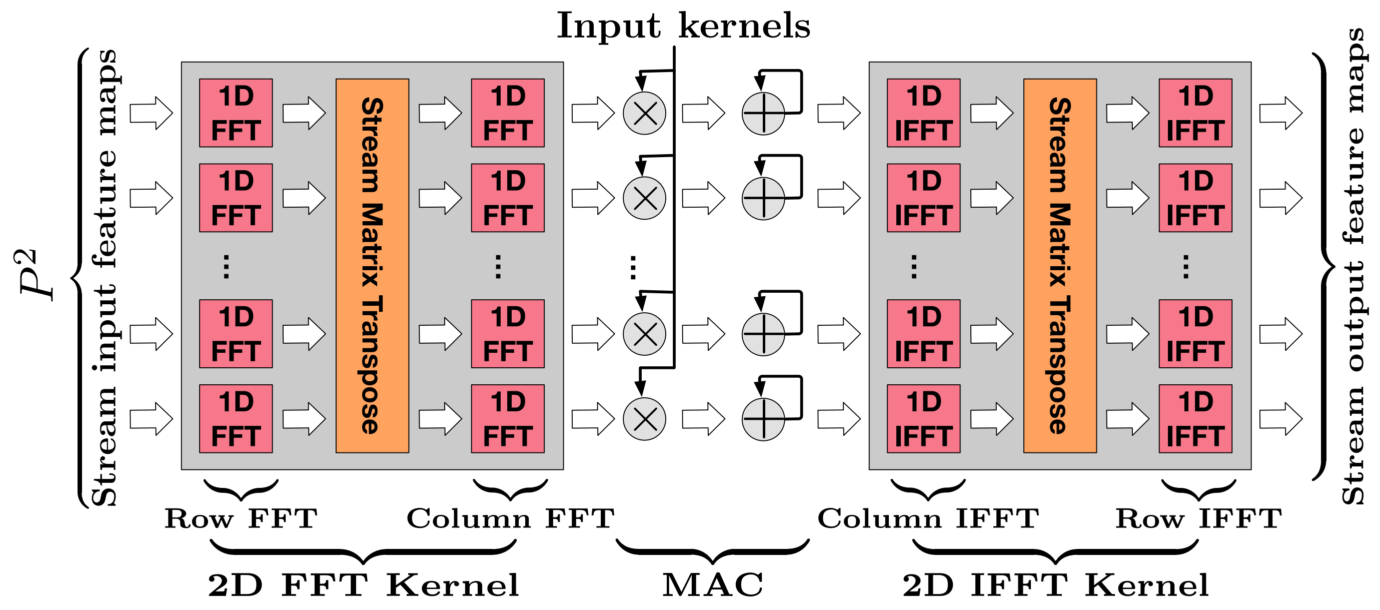
Variable Length FFT Module

- Example of Radix-4 FFT module:



- To do 64 point FFT, we need $\log_4 64 = 3$ stages.
- To do 16 point FFT, we use the first 2 stages of the 64 point architecture, and bypass the third stage.

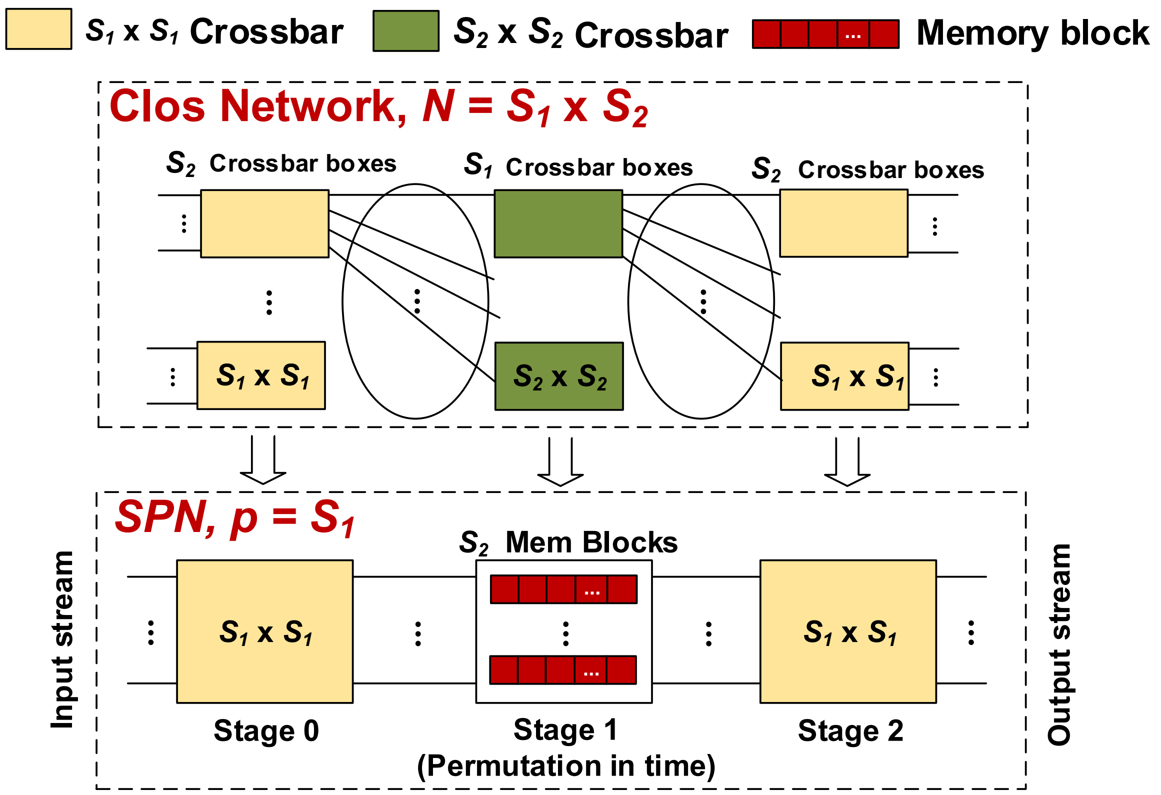
2D Convolver Accelerator



- Architecture: 2D FFT + Multiply-Accumulate + 2D IFFT
- 2D FFT = Row FFT + Column FFT
- Matrix Transpose is achieved by direct wiring or **streaming permutation** on streaming data from memory

Key Idea in our Streaming Permutation Hardware Design

- Fold the Clos network into a 3-stage Streaming Permutation Network (*SPN*)
- *SPN*
  - Stage 0
    - S_1-*to*-S_1 connection
  - Stage 1
    - S_1 single-port memory blocks, each of size S_2
  - Stage 2
    - S_1-*to*-S_1 connection
- Permutation in time
  - Permuting temporal order of data elements



Supports any given arbitrary permutation on streaming data!

Fast Fourier Transform

- Parameterized architecture:
  - Algorithm-mapping parameters
    - Vertical parallelism (V_p)
    - Number of FFT processor (N_p)
  - Architecture-binding parameters
    - Type of memory, pipeline depth
- Optimizations:
  - Memory activation scheduling
    - Deactivate periodically when not accessed
  - Energy efficient permutation
    - Permute streaming data using memories
    - Use the proposed data remapping technique
  - Clock gating
    - Disable the floating point unit when one of its inputs is 0, 1, j, or -j
- Comparison: **4.5x** improvement

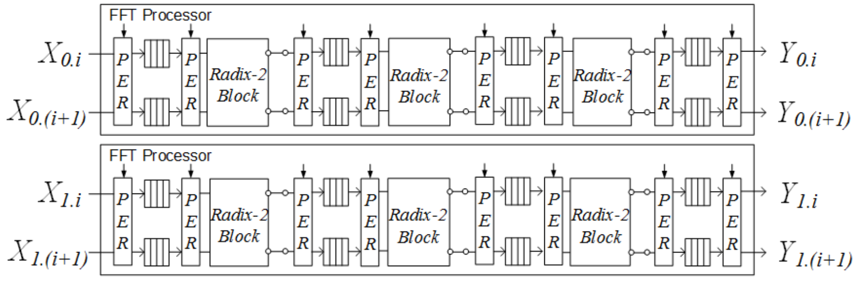
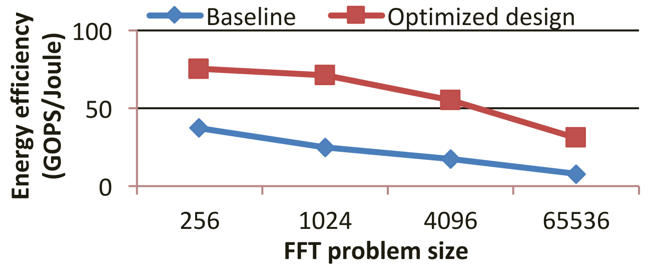
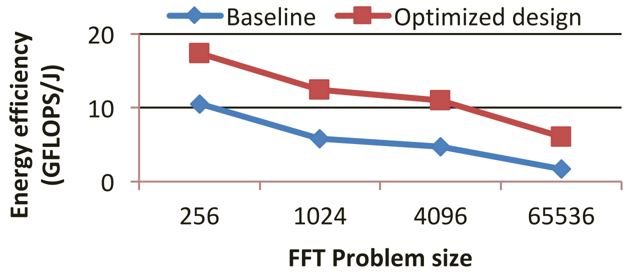


Fig. 8-point FFT architecture (V_p = 2, N_p = 2)





Performance Evaluation (1)

- Performance metric: ***Delay-Multiplier Product***
  - It is computed as the number of ***cycles*** to process a complete convolutional layer times the number of ***multipliers*** consumed by the design.
  - Power consumption of the convolver is proportional to the number of multipliers.
  - Area of the convolver is proportional to the number of multipliers.
  - Approximation to ***Energy dissipation*** and ***Area-Delay Product.***

Performance Evaluation (2)

- ***DM ratio***: $\dfrac{DM\ Product\ of\ Space\ Convolver}{DM\ Product\ of\ OaA\text{−}based\ Convolver}$
- Design Choice: Choose the FFT size such that the ***DM* ratio** is maximized for given resource constraints.

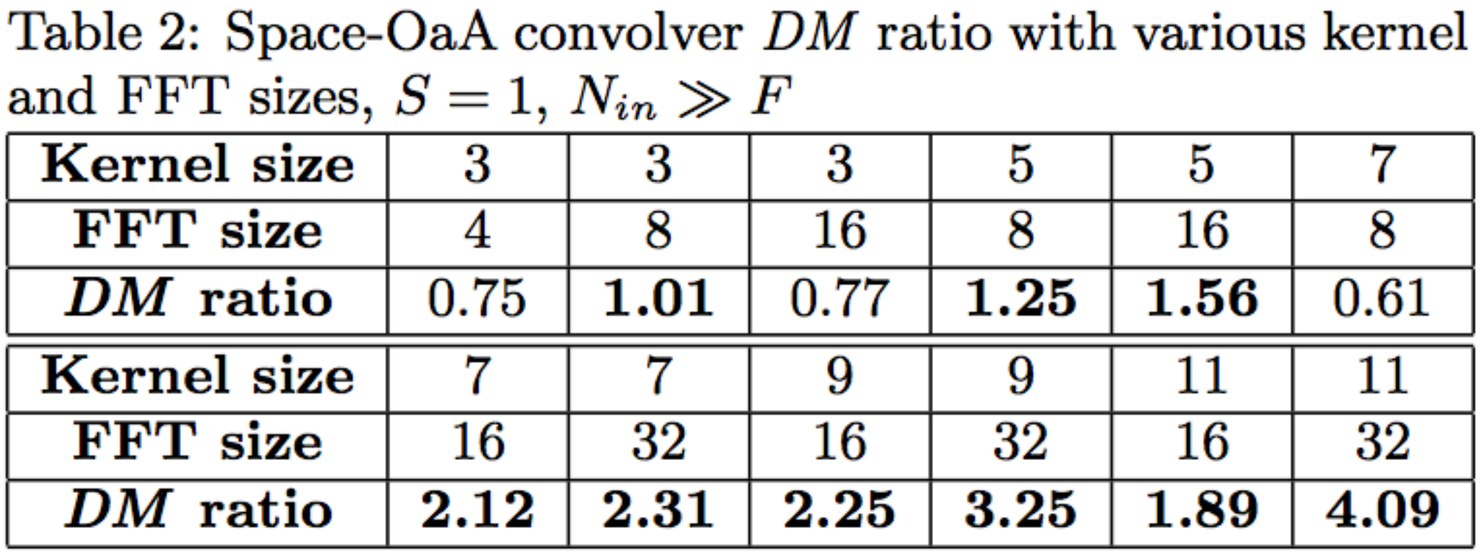


Table 2: Space-OaA convolver *DM* ratio with various kernel and FFT sizes, $S = 1$, $N_{in} \gg F$

Kernel size	3	3	3	5	5	7
FFT size	4	8	16	8	16	8
***DM* ratio**	0.75	**1.01**	0.77	**1.25**	**1.56**	0.61
Kernel size	7	7	9	9	11	11
FFT size	16	32	16	32	16	32
***DM* ratio**	**2.12**	**2.31**	**2.25**	**3.25**	**1.89**	**4.09**

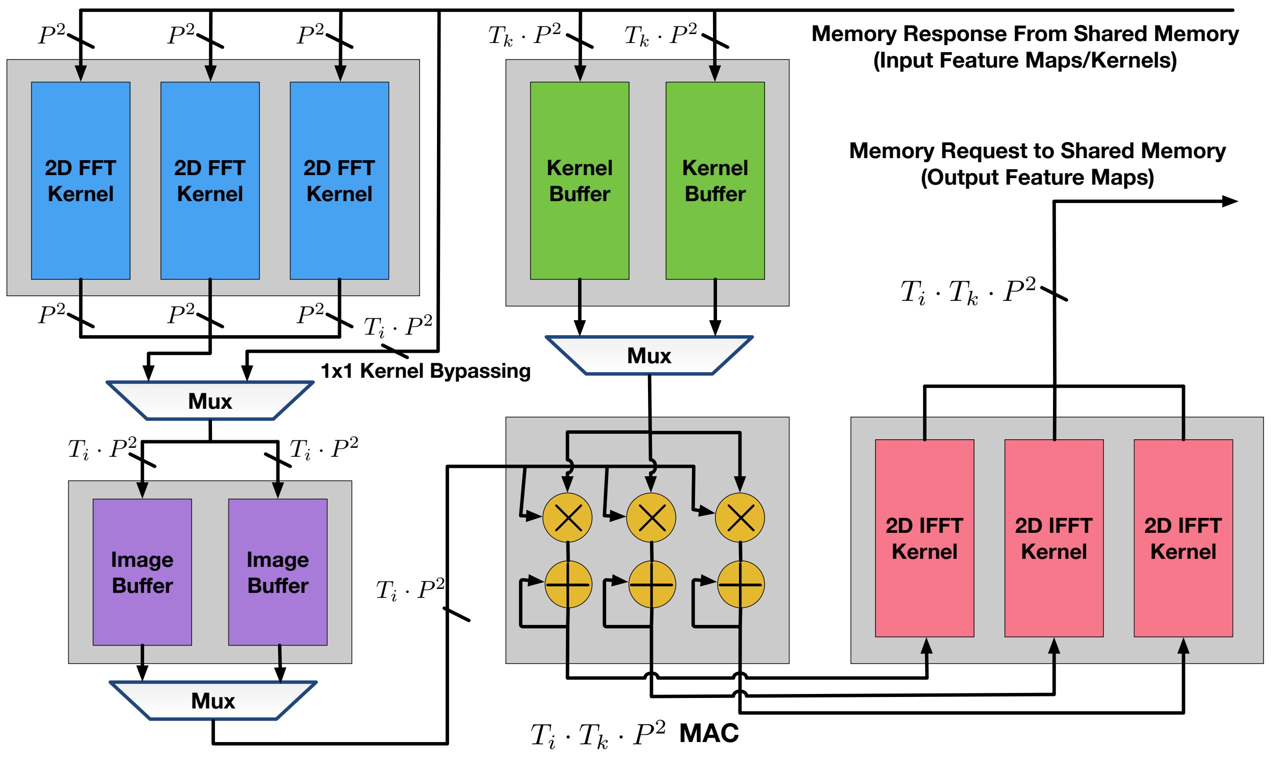
Outline

- Motivation
- Target platform and problem definition
- Main approach
- Overall system design
- Experiments and Results

Overall System Design

- CPU-FPGA concurrent processing through shared memory
- Overlap memory access latency and data processing latency by buffering data on FPGA
- By using CPU for light-weight but general tasks, using FPGA for high computational intensive but dedicated tasks, our framework can apply to a wide range of CNN models.

Overall System Design



- Buffer FFT processed data for **reuse** across different kernels.
- Image buffer task parallelism T_i and kernel buffer task parallelism T_k. The system parallelism is T_iT_k.

USC School of Engineering

Outline

- Motivation
- Target platform and problem definition
- Main approach
- Overall system design
- Experiments and Results

Experimental Setup

- Intel QuickAssist QPI FPGA Platform (Intel Heterogeneous Architecture Research Platform)
- 10 Core Intel Xeon E5-2600 v2 processor
- Altera Stratix V FPGA
- 6.25 MB BRAM + 939 K registers + 234720 ALM + 256 DSP
- CPU and FPGA share 2 GB address space.
- FPGA can access data through QPI from the last level cache in the memory system of CPU.
- FPGA is only cacheline addressable (64 Bytes).
- We implemented a generic accelerator for convolutional layer on FPGA. The remaining work is done on CPU.

Experimental Results

	Zhang (2015)	Qiu (2016)	Our Work	
Data Precision	32-bit float point	16-bit fixed	32-bit float point	
Platform	Virtex-7	Zynq ZC706	Intel Xeon + Stratix V	
Frequency (MHz)	100	150	200	
CNN Model	AlexNet	VGG16-SVD	AlexNet	VGG16
Multiplier	747	780	224	224
Memory	4.5 MB	2.13 MB	4.0 MB	4.0 MB
Throughput (GFLOPs/sec)	61.62	187.80 (CONV)	83.00 (CONV)	123.48 (CONV)
Delay (CONV) (ms)	21.61	163.42	40.81	263.27
Delay * Multiplier	16142	127467	9141	**58972**
Resource Efficiency (GFLOPs/sec/Multiplier)	0.028	0.24	0.37	**0.55**
Power Efficiency (GFLOPs/sec/W)	3.31	19.50	6.30	9.37

Conclusion

- High throughput OaA-based 2D convolver to accelerate convolutional layer
- Implemented AlexNet and VGG16 on Intel QuickAssist QPI-FPGA Platform and compared with state-of-art designs
  - ***AlexNet***: 1.35x throughput improvement using 3.33x less multipliers and 1.1x less memory
  - ***VGG16***: 0.66x throughput using 3.48x less multipliers with 32-bit floating point. Potential performance improvement with compressed data representation
- Future work
  - Using compressed data representation
  - Build automatic code generation tool
  - Explore other algorithms including Winograd algorithm

Thanks

fpga.usc.edu

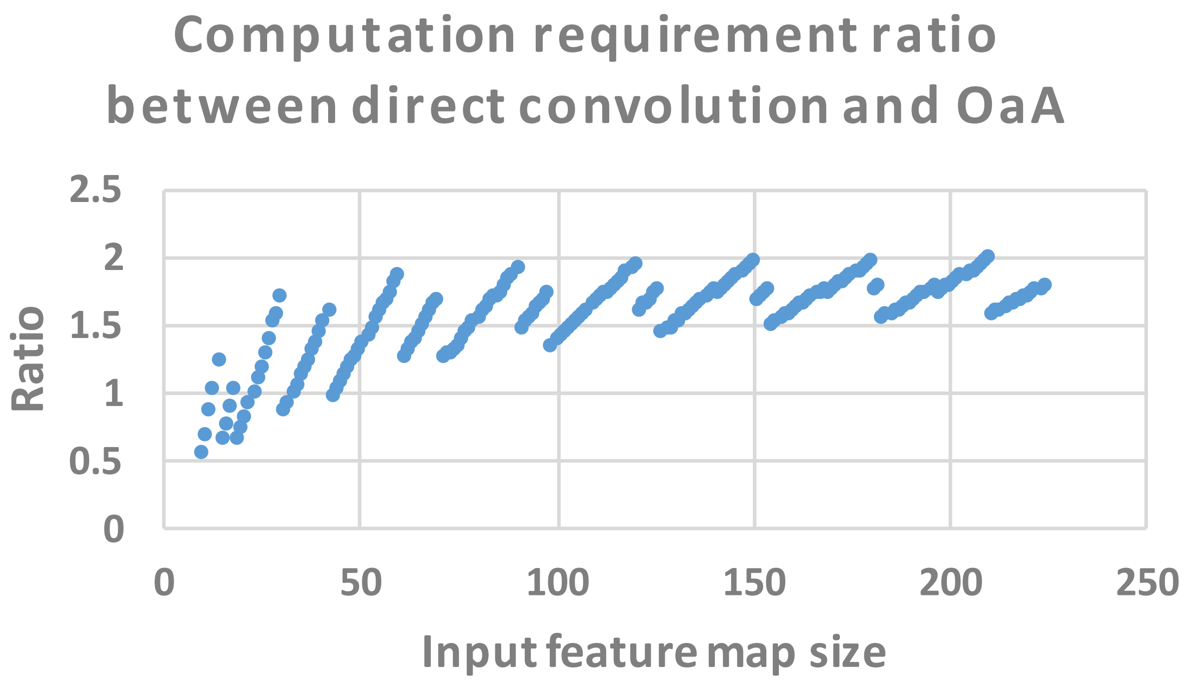
Backup Slides

Deeper understanding of Overlap-and-Add

- Basic observation
  - Large kernel sizes benefit more from frequency domain approach than small kernel sizes
  - Large stride makes frequency domain approach less advantageous
  - Deeper input feature map channels make frequency domain approach more advantageous due to the linearity of FFT.

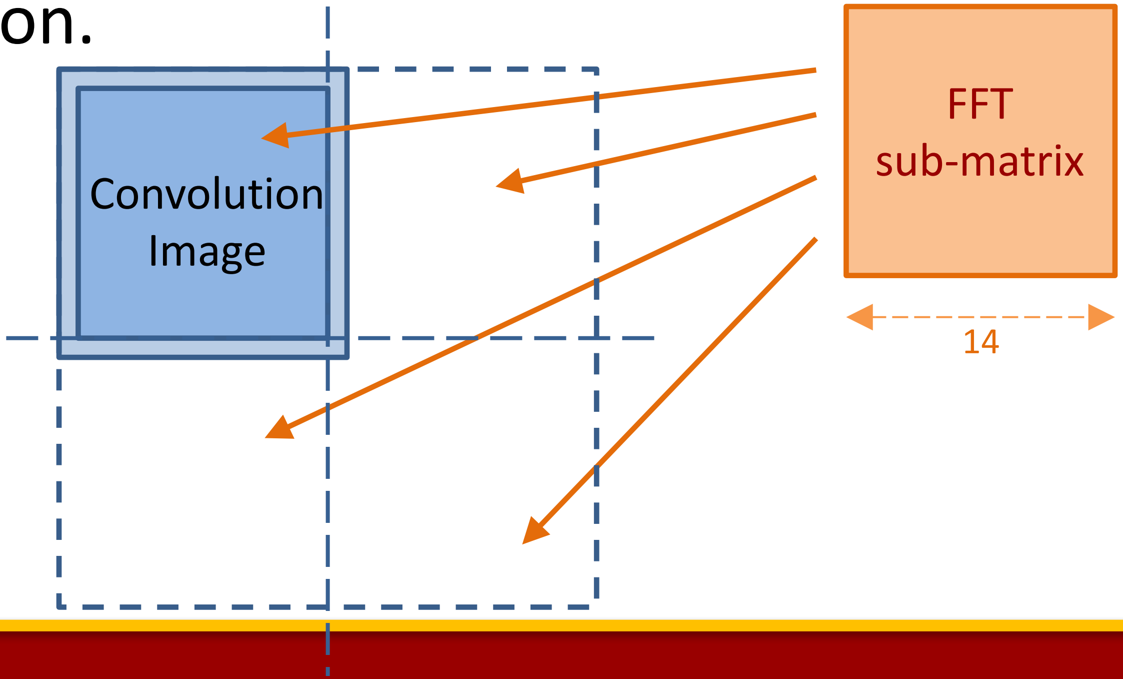
Deeper understanding of Overlap-and-Add (2)

- We fix D_in = 64, D_out = 64, kernel size = 3, stride = 1. We vary the input feature map size and calculate the computation requirement ratio between direct convolution and OaA.



Deeper understanding of Overlap-and-Add (3)

- Large input feature map size leads to more advantage for OaA-based convolution.
- Corner case: input feature map size is slightly greater than FFT sub-matrix. This causes approximately 4x waste of computation.



Reference

- [1] C. Zhang, et al. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '15.
- [2] Ren Chen, Hoang Le and Viktor K. Prasanna, Energy Efficient Parameterized FFT Architecture, IEEE International Conference on Field Programmable Logic and Applications (FPL), August 2013.