Scalable and Modularized RTL Compilation of Convolutional Neural Networks onto FPGA

Yufei Ma, Naveen Suda, Yu Cao, Jae-sun Seo, Sarma Vrudhula†

School of Electrical, Computer and Energy Engineering

†School of Computing, Informatics, Decision Systems Engineering

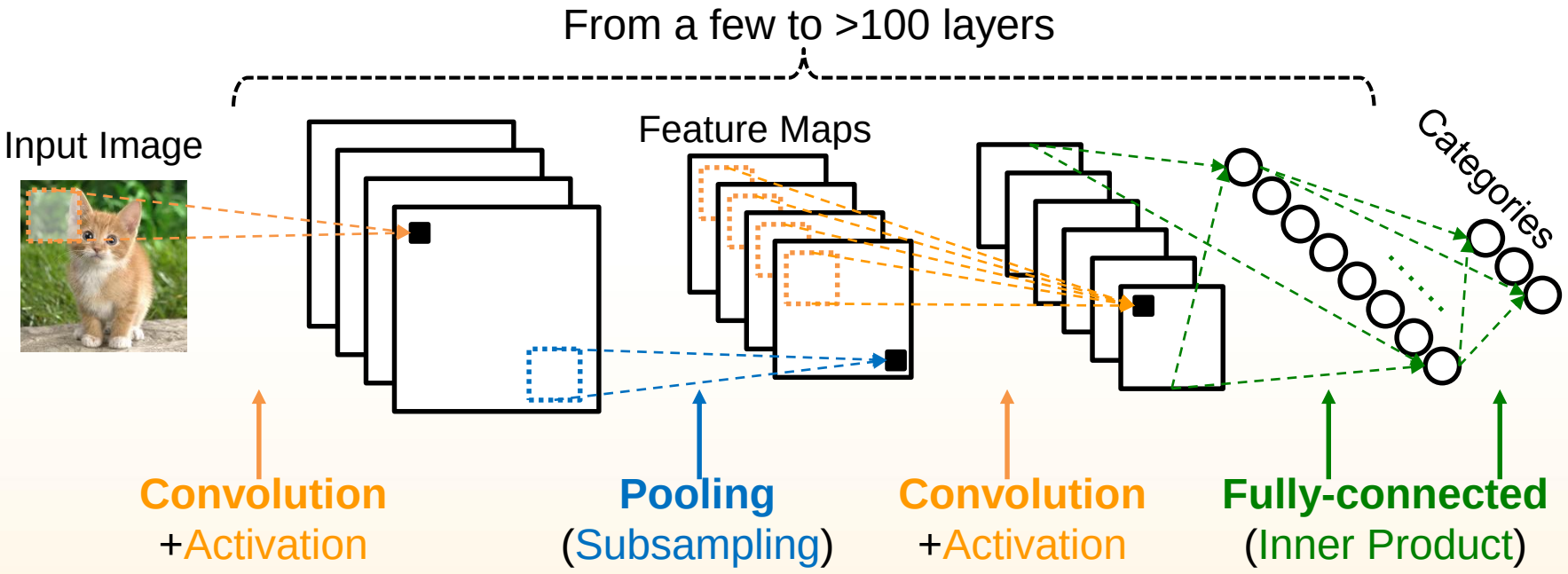
Arizona State University, Tempe, USA

Outline

- Overview of CNN Algorithms
- Current CNN Accelerators & Motivation
- Proposed Modular CNN RTL Compiler
- Experimental Results
- Conclusion

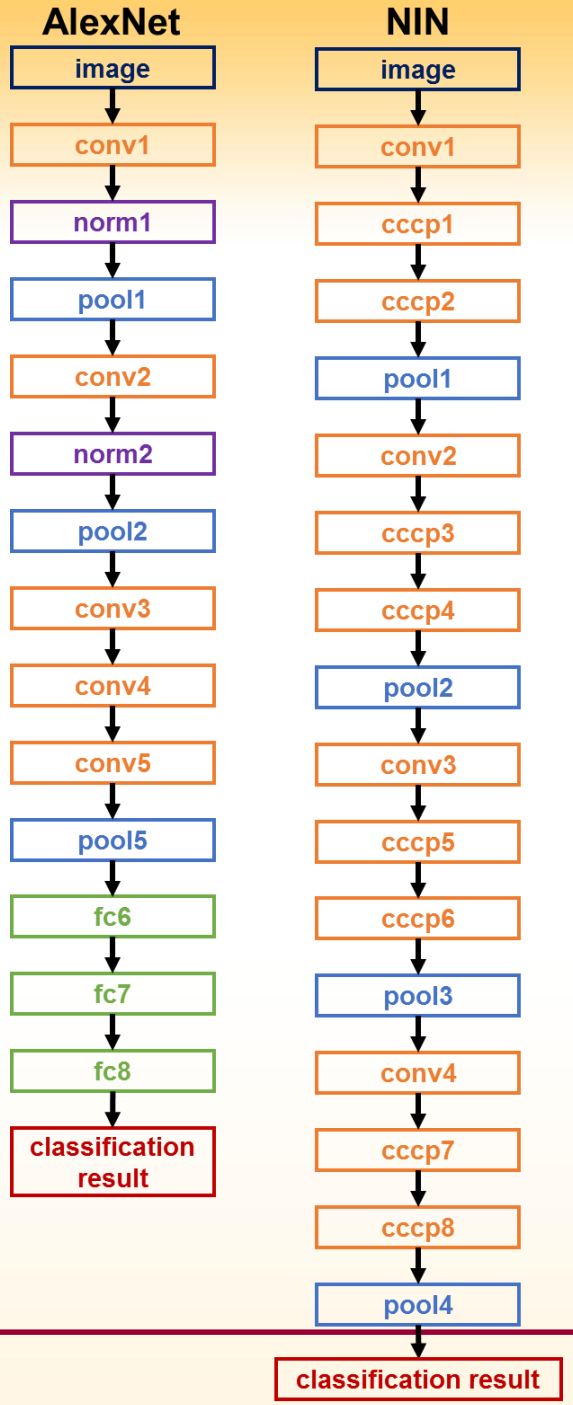
Convolutional Neural Networks (CNN)

- Dominant approach for recognition and detection tasks
- Highly iterative with a few computing primitives
- Composed of multiple types of layers
- Evolving rapidly with more layers to achieve higher accuracy



CNN Layers and Structure

- **Convolution** (conv or cccp)
  - 3D MAC operations
  - Constitute >90% of the total operations
- **Pooling** (pool)
  - Keep the maximum or average value of pixels
- **LRN** (norm)
  - Local response normalization : non-linear
- **Fully-connected** (fc)
  - Matrix-vector multiplication
  - Require large volume of weights
- **CNN Structure** for image classification
  - **AlexNet** [A. Krizhevsky, *NIPS2012*]
  - **NIN** [M. Lin, *ICLR2014*]



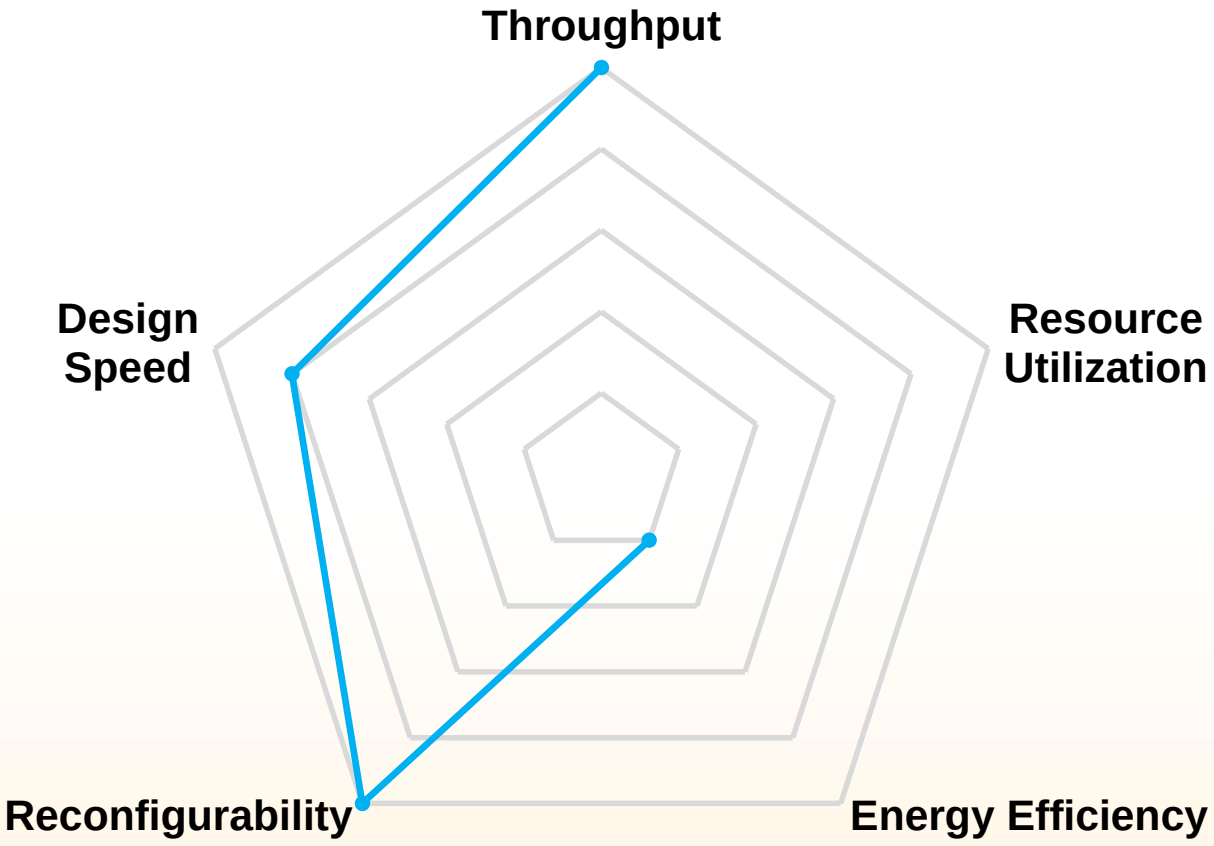
Outline

- Overview of CNN Algorithms
- Current CNN Accelerators & Motivation
- Proposed Modular CNN RTL Compiler
- Experimental Results
- Conclusion

Comparison of CNN Accelerators

Software, GPU [Y. Jia, *Caffe*; M. Abadi, *TensorFlow*]

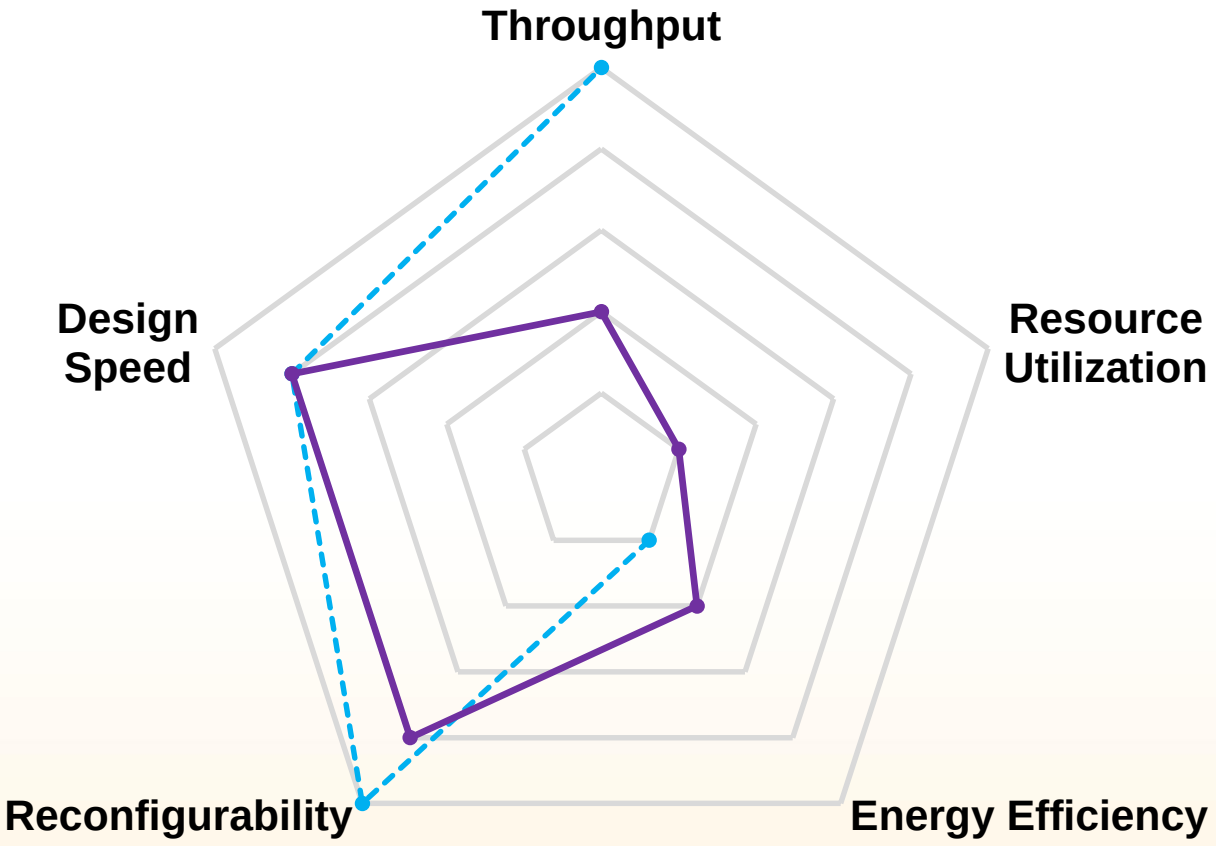
- Flexible deep learning framework with modularity
- Accelerated on GPU with thousands of parallel cores
- High power consumption (>100W)



Comparison of CNN Accelerators

HLS, FPGA [C. Zhang, *FPGA2015*; N. Suda, *FPGA2016*]

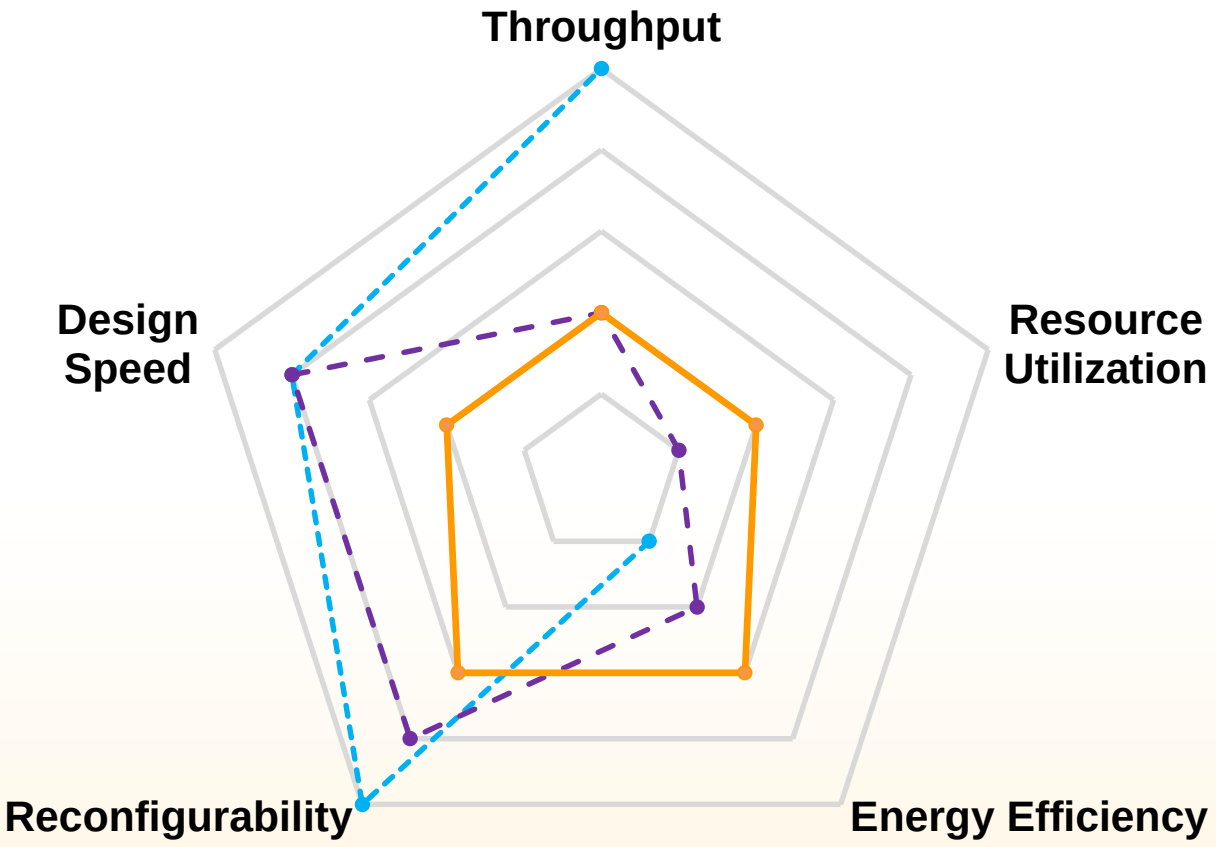
- High-level synthesis (e.g. OpenCL) based FPGA accelerator
- Short turnaround time and fast design optimization
- Cannot exploit low-level hardware structures



Comparison of CNN Accelerators

RTL, generic CNN accelerator [C. Farabet, *CVPR2011*]

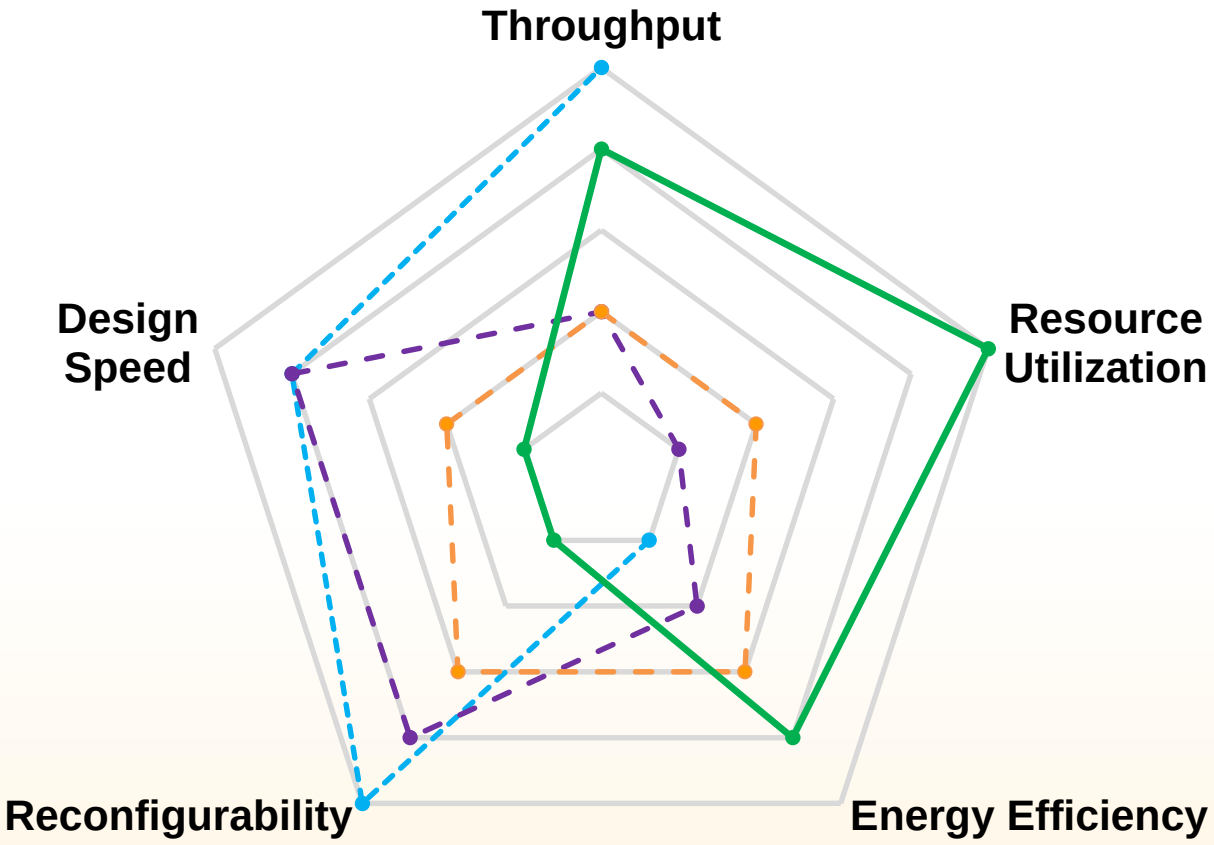
- Agnostic to the CNN model configuration
- Inefficient hardware resource usage



Comparison of CNN Accelerators

RTL, optimized for a specific CNN [J. Qiu, *FPGA2016*]

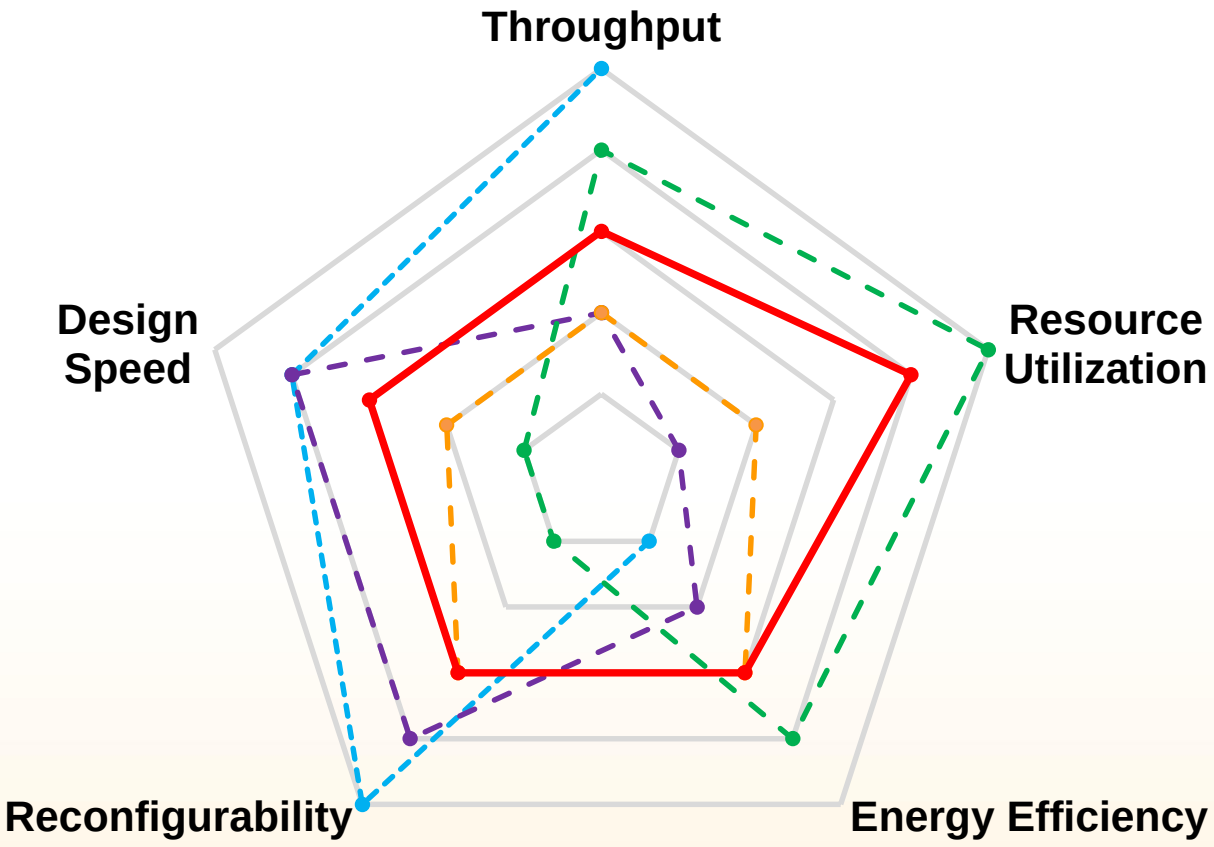
- High efficiency with greater acceleration
- Poor flexibility, long turnaround time
- Require in-depth understanding of FPGA/ASIC



Comparison of CNN Accelerators

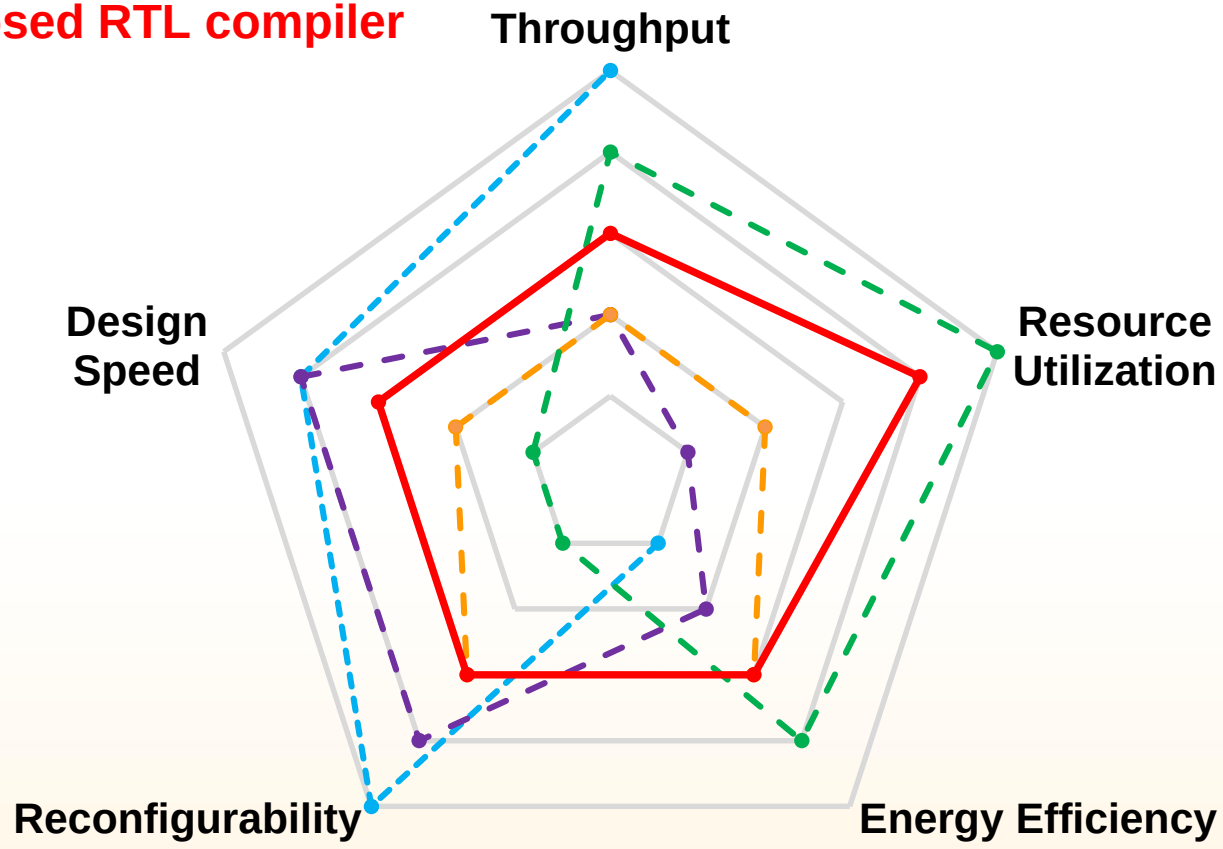
Proposed RTL compiler

- Modular and scalable hardware design framework
- **Integrate the flexibility of HLS and the finer level optimization of RTL**



Comparison of CNN Accelerators

- Software, GPU
- HLS, FPGA
- RTL, generic CNN accelerator
- RTL, optimized for a specific CNN
- Proposed RTL compiler

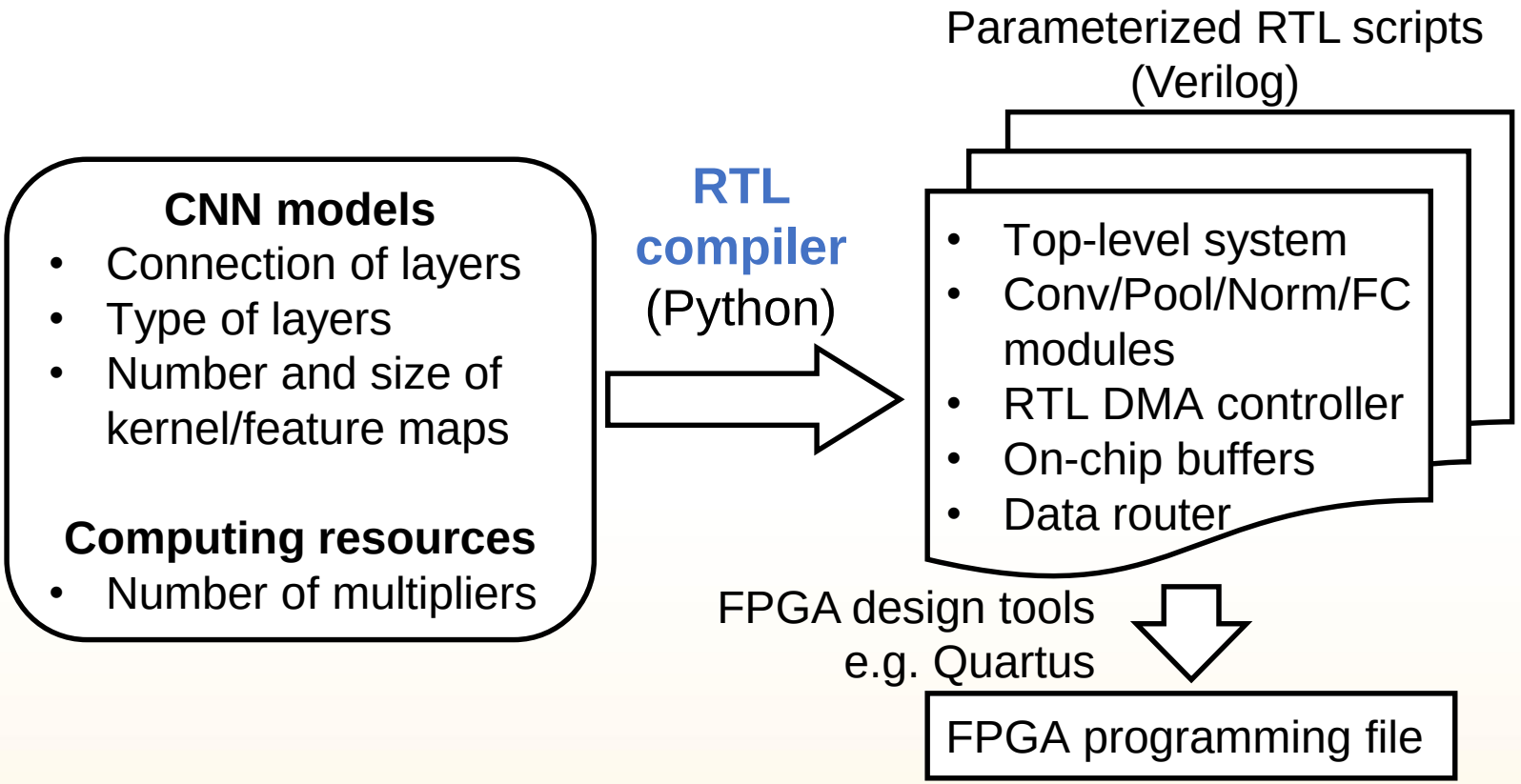


Outline

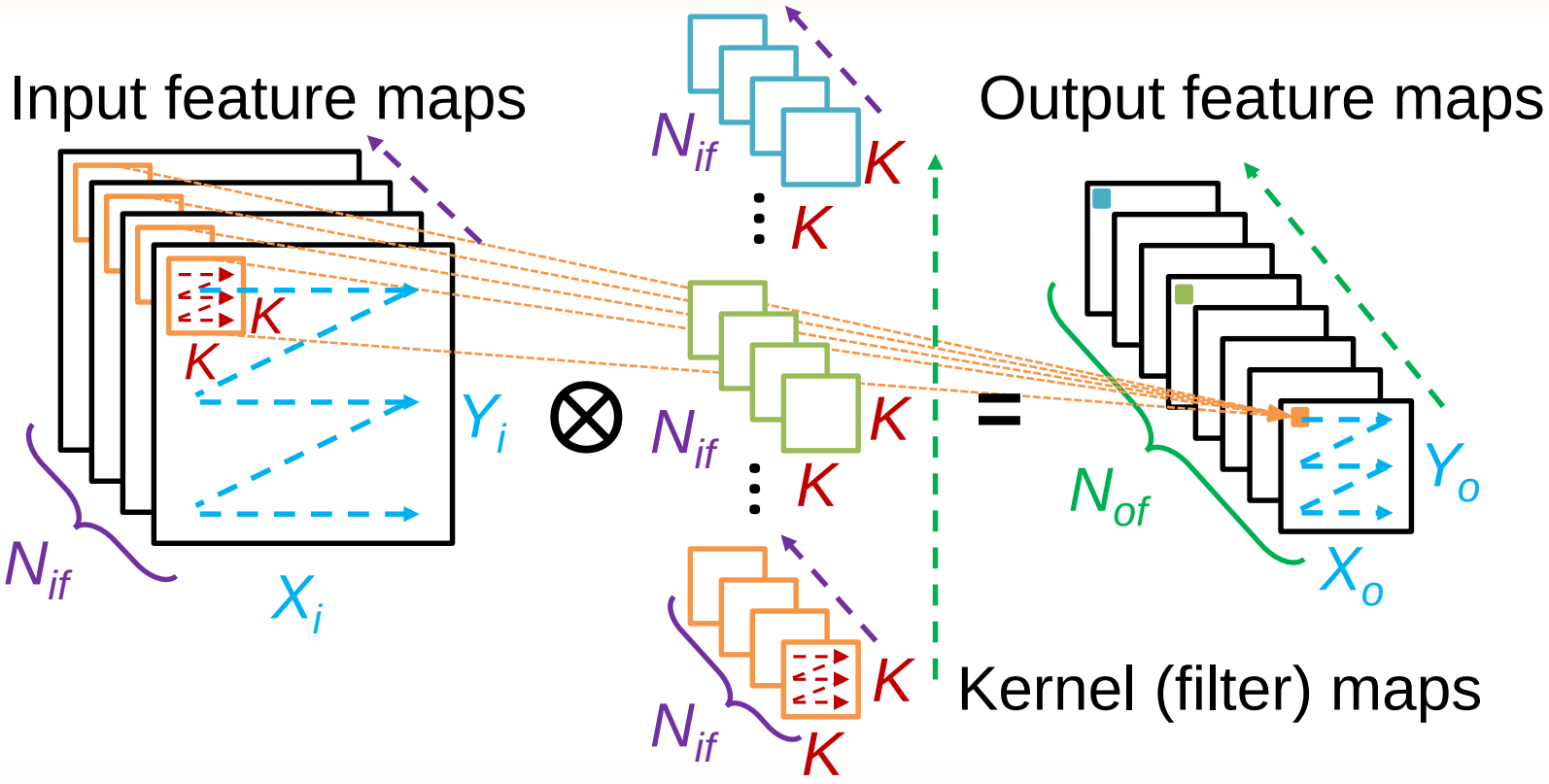
- Overview of CNN Algorithms
- Current CNN Accelerators & Motivation
- Proposed Modular CNN RTL Compiler
- Experimental Results
- Conclusion

Proposed CNN RTL Compiler

- Modular and scalable hardware design framework
- Compile end-to-end CNNs into efficient RTL codes for FPGA/ASIC



Convolution Parameters and Loops



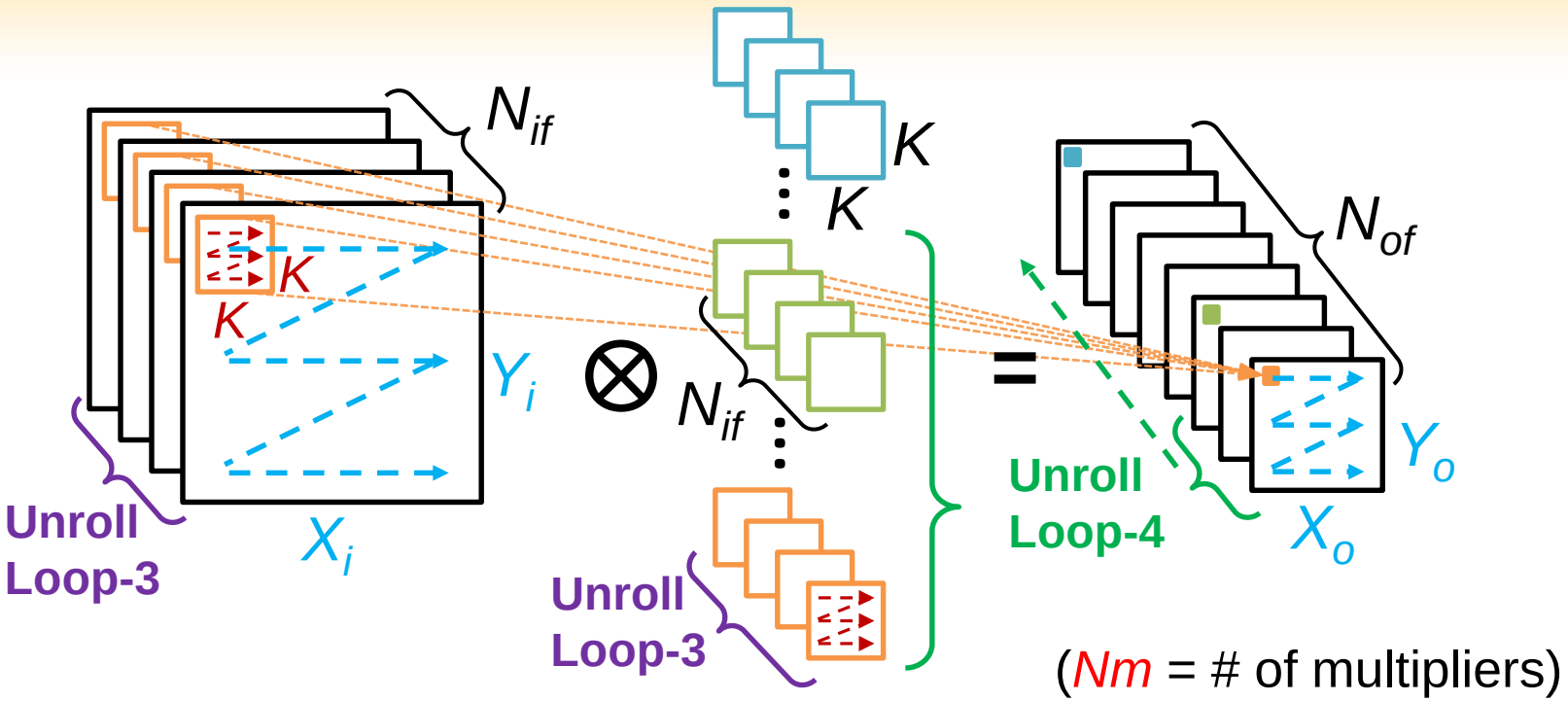
Loop-4 Across the output feature maps of *Nof*

Loop-3 Across the input feature maps of *Nif*

Loop-2 Scan within one input feature map along *X*×*Y*

Loop-1 MAC within a kernel window of *K*×*K*

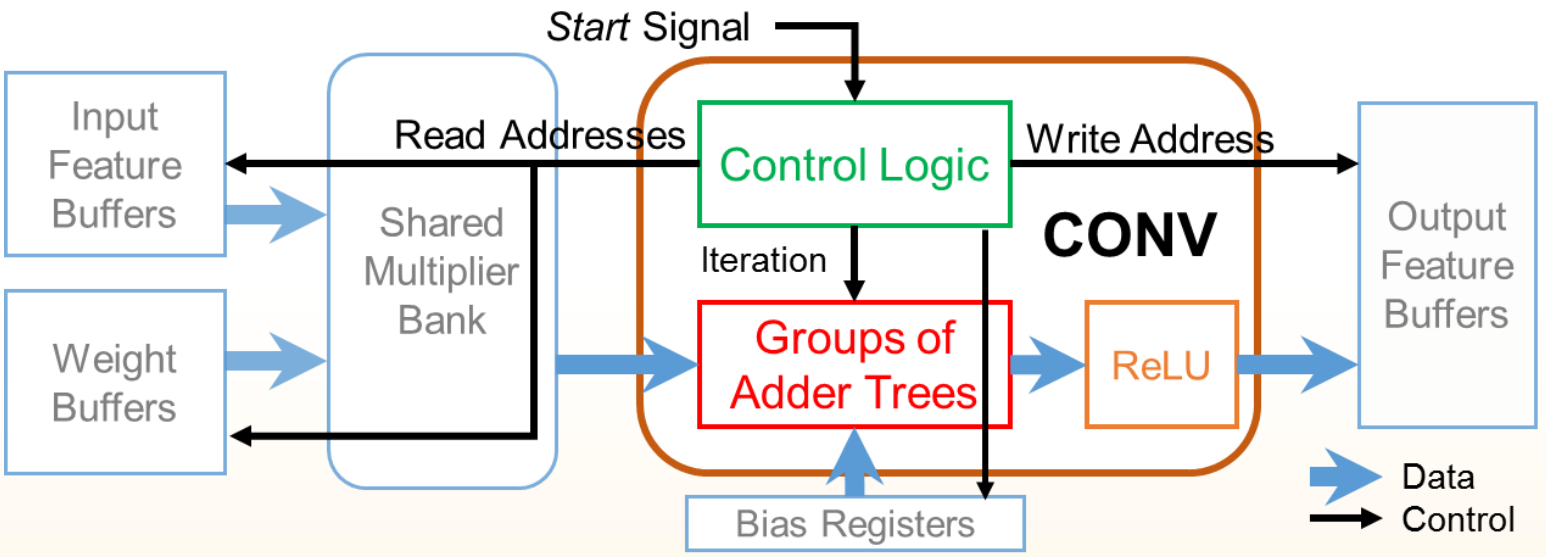
Strategy to Accelerate Convolution



- If *Nm*>*Nif* : fully unroll **Loop-3** and further unroll **Loop-4**
  - *Nm*/*Nif* output feature maps with shared features
- If *Nm*<*Nif* : partially unroll **Loop-3**
  - Repeat kernel window sliding by *Nif*/*Nm* times
- Serially compute **Loop-1** before **Loop-2** : reduce # of partial sums

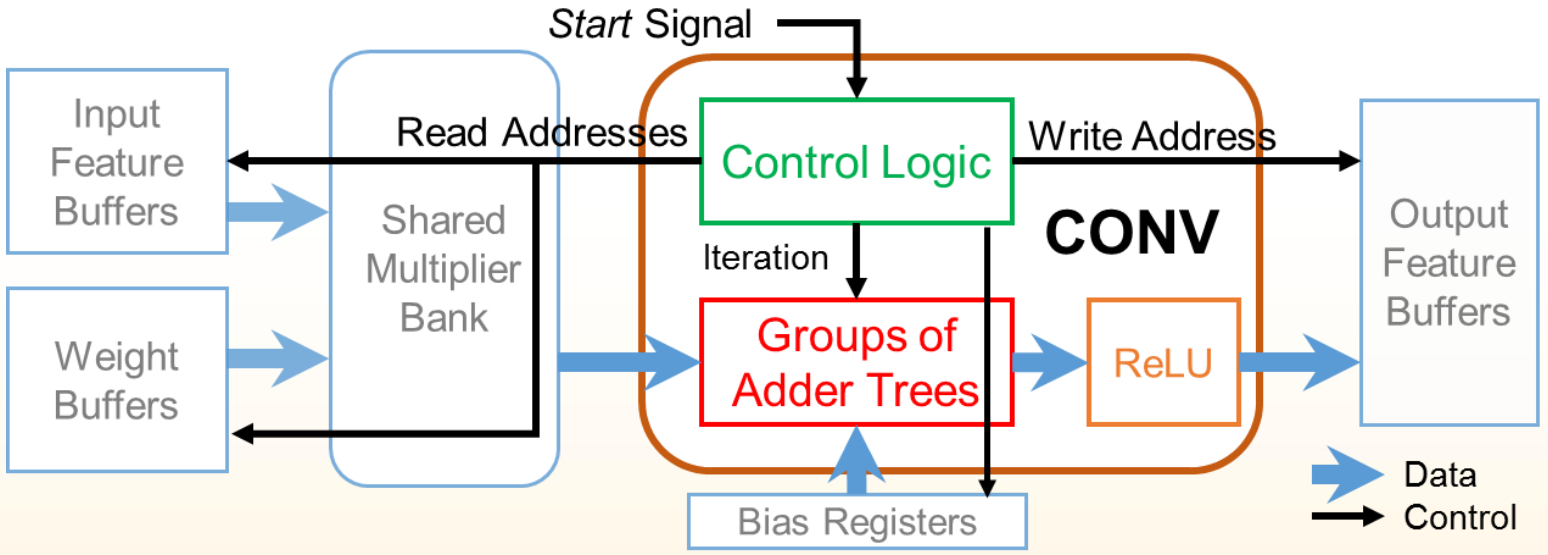
CONV Module and Components

- Control logic
  - Control the sliding of four loops by counters
  - Counters are parameterized to *K*, *X*, *Y*, *Nif* and *Nof* of each layer
  - Generate buffer addresses



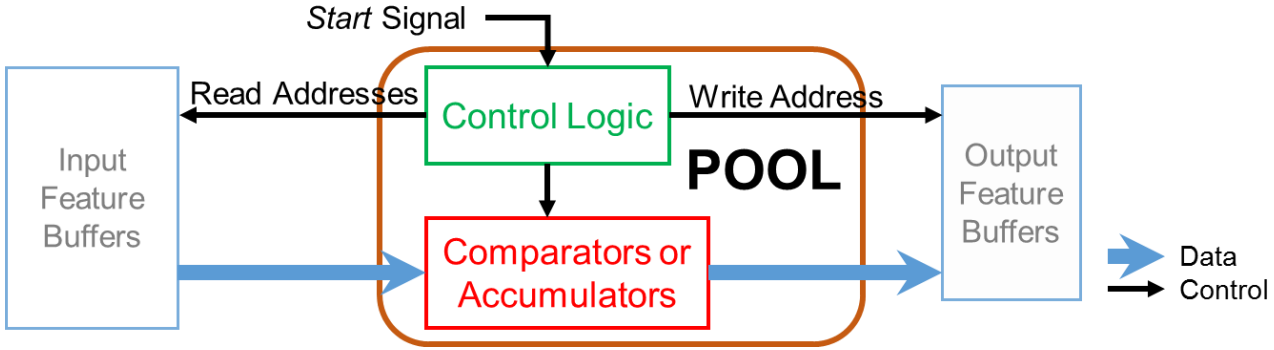
CONV Module and Components

- Adder Trees
  - # of fan-in = *Nif*, # of adders = *Nm*/*Nif*
  - Sum results from *Nif* parallel multipliers
  - Accumulate within one kernel window (*K*×*K*)
  - Shared by convolution layers with identical *Nif*.
- ReLU = max(pixel, 0)
  - Check the sign bit

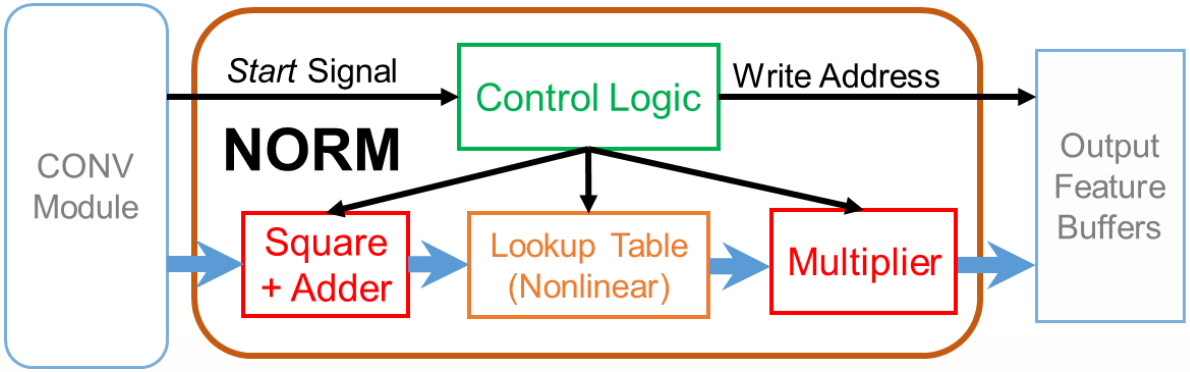


POOL, NORM, and FC Modules

- **POOL** (MAX or AVE) Module



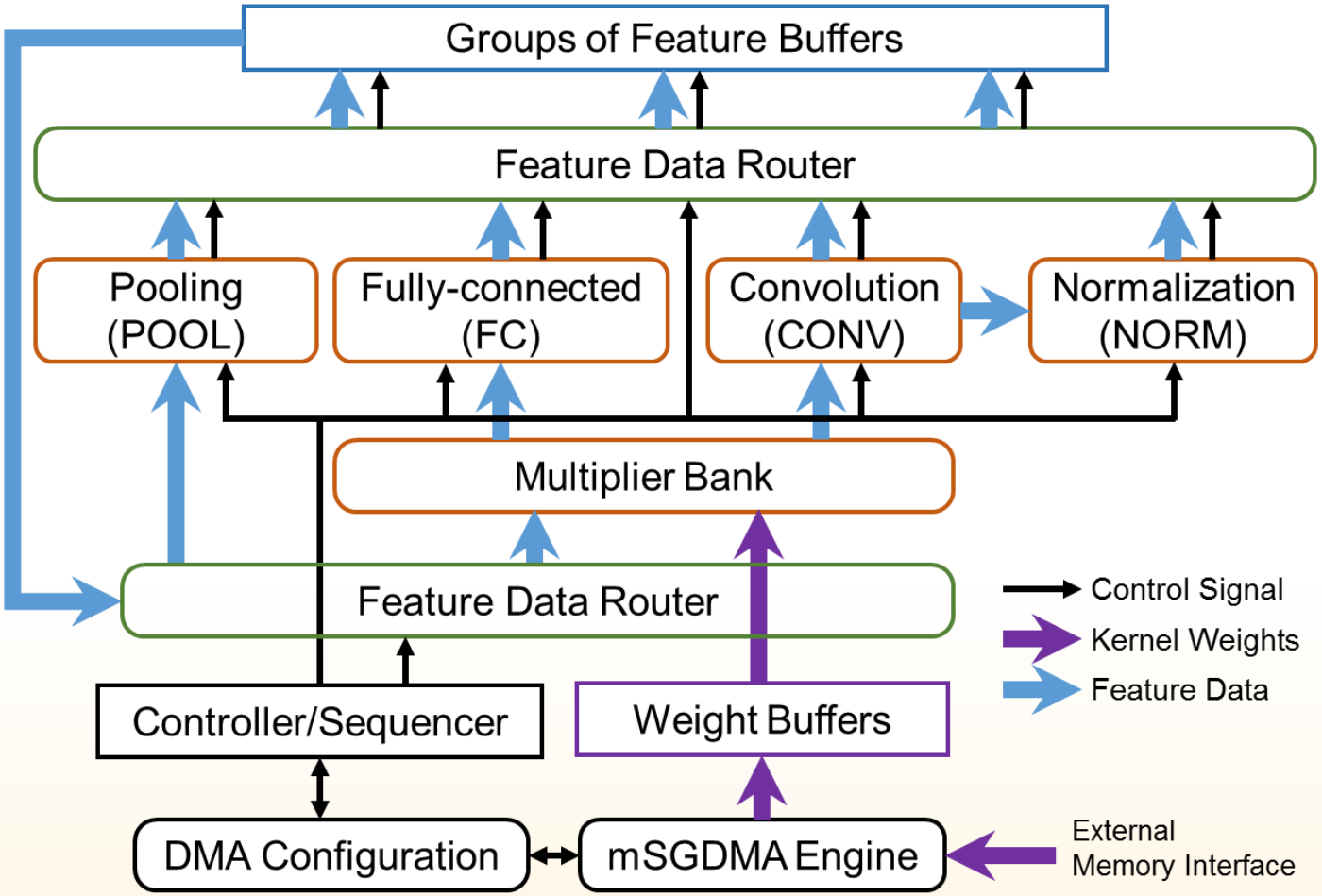
- **NORM** Module



- **FC** Module
  - Perform matrix-vector multiplication (special form of convolution)
  - Share multipliers with CONV
  - Adders are shared across all FC layers

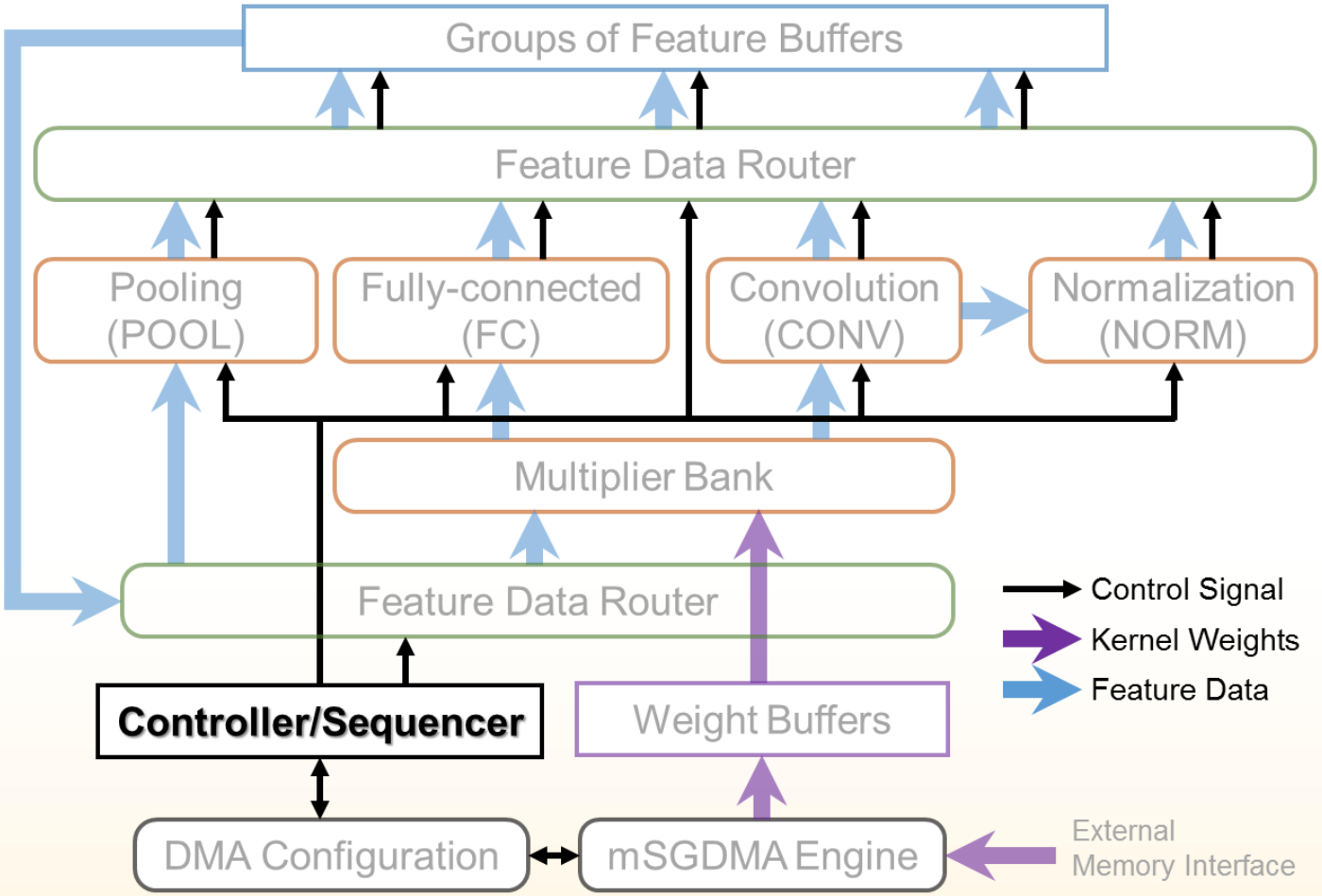
Integration of Modules

- Overall CNN Accelerator



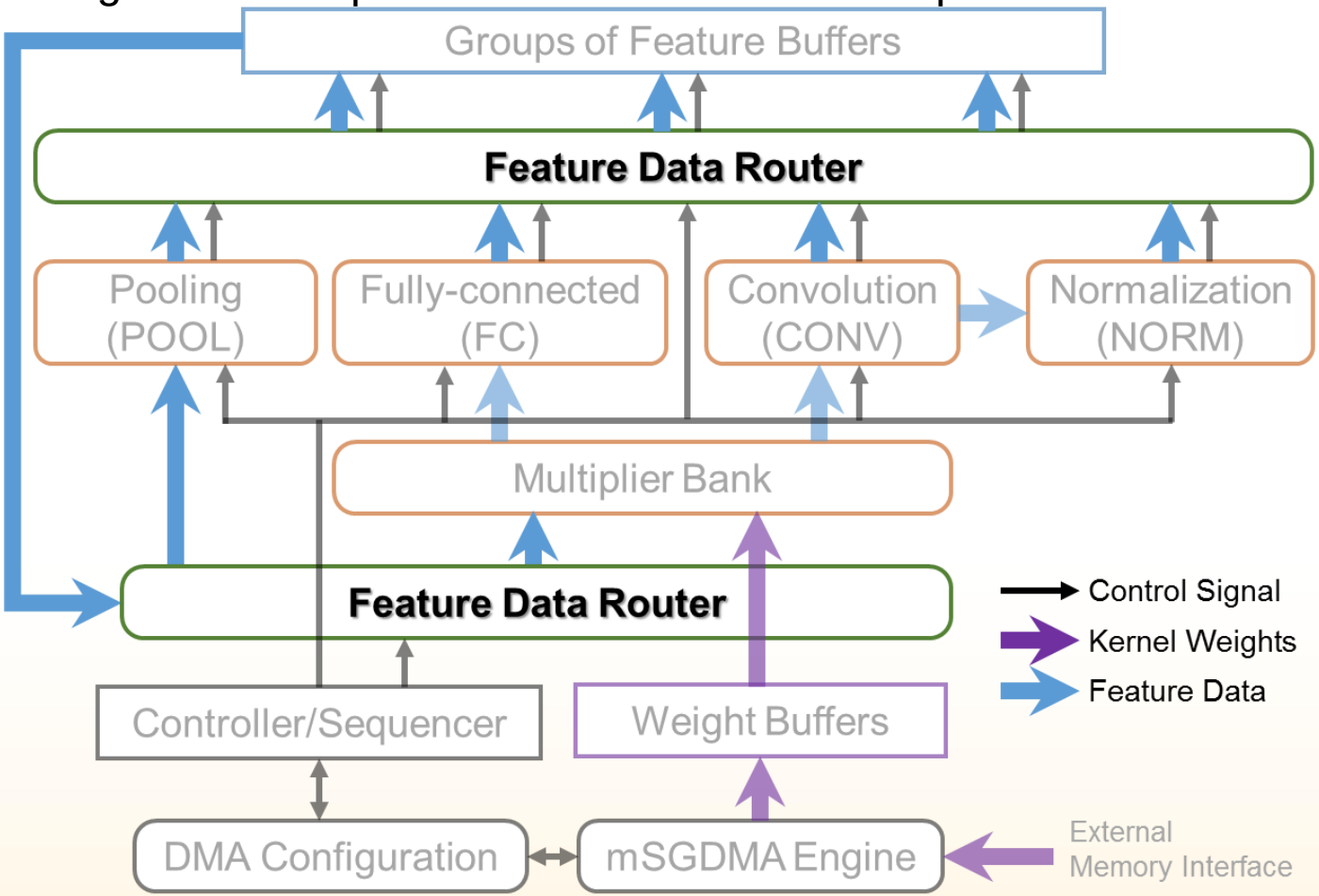
Integration of Modules (Controller)

- Controller
  - Direct the layer by layer serial computation of modules



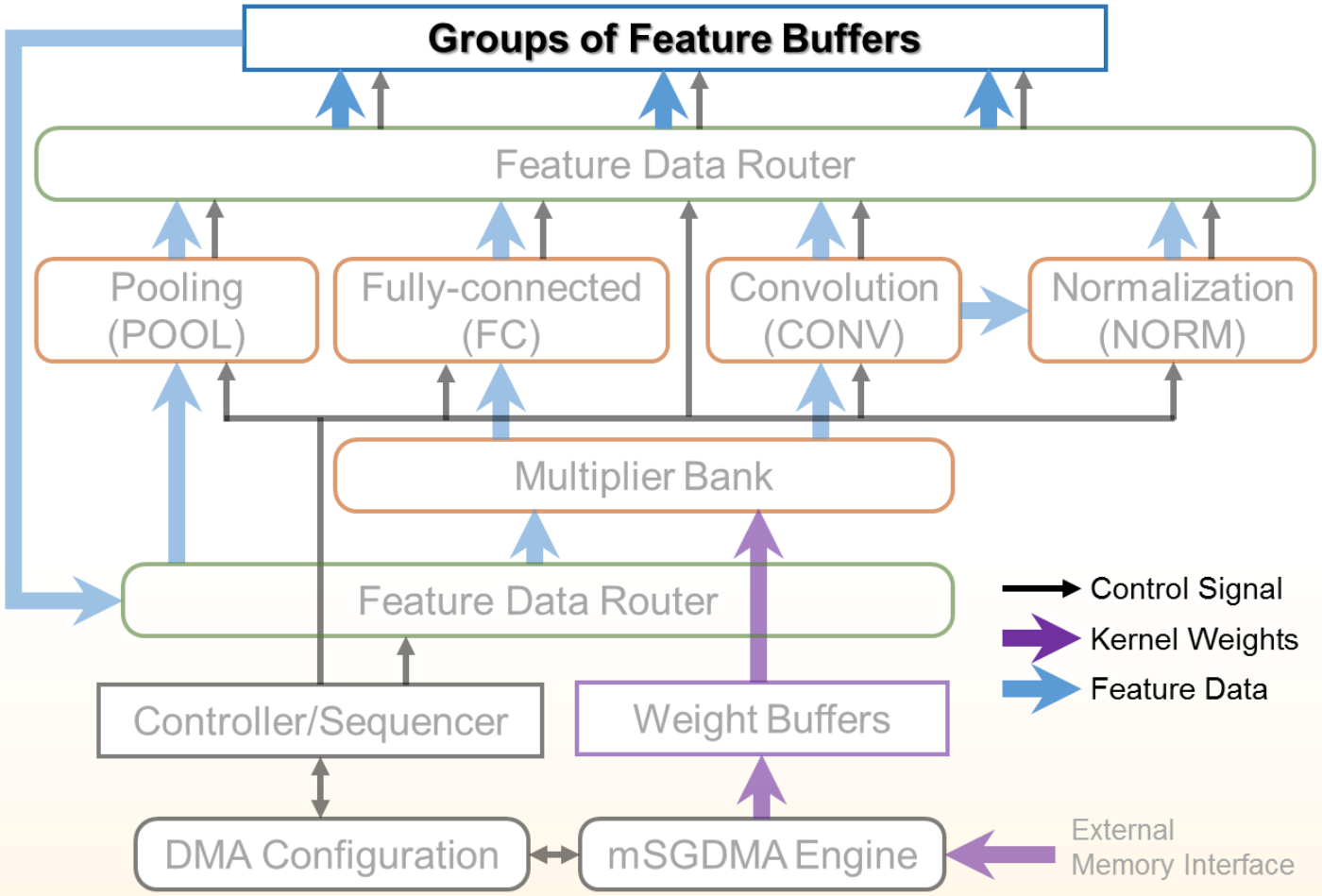
Integration of Modules (Data Router)

- Feature Data Router
  - Select write and read data of two adjacent modules
  - Assign buffer outputs to POOL or shared multipliers



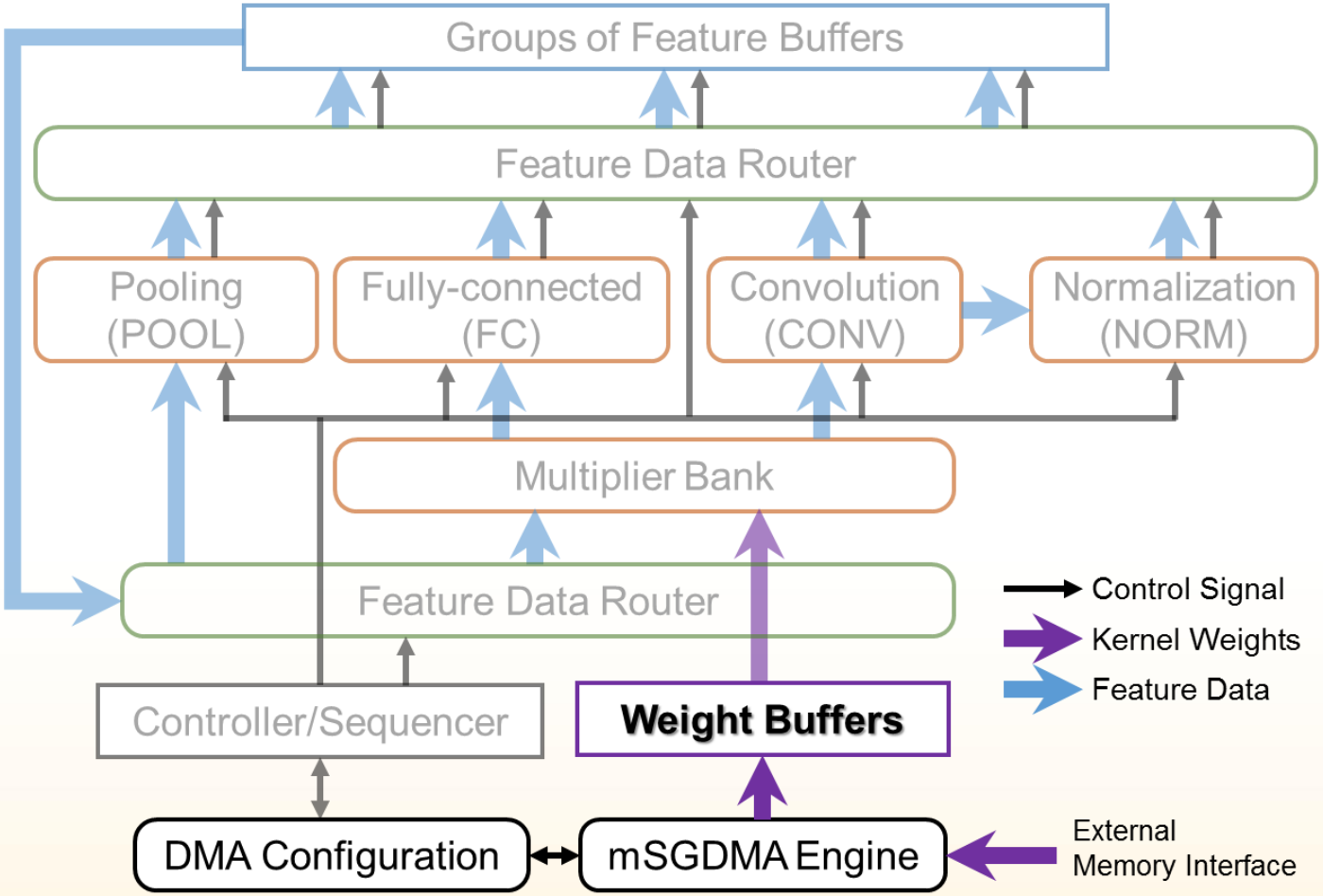
Integration of Modules (Memory)

- Feature Buffers
  - Feature maps are stored in separate on-chip RAMs



Integration of Modules (Memory)

- Weight Buffers
  - FC weights transfer is overlapped with its computation
  - CONV weights transfer is before its computation

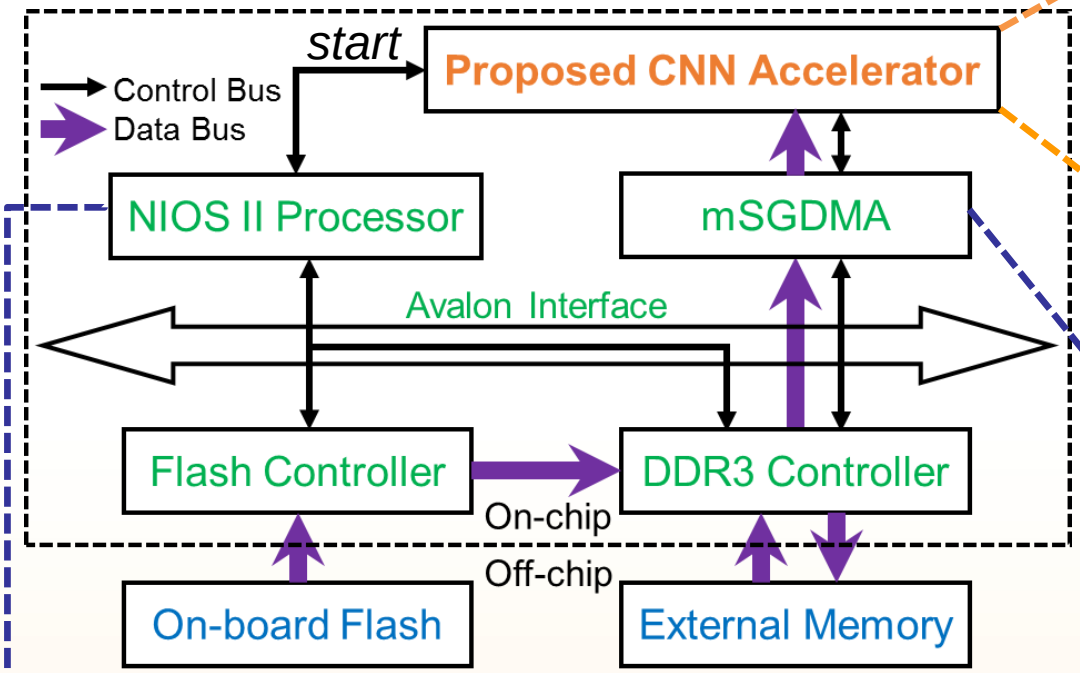


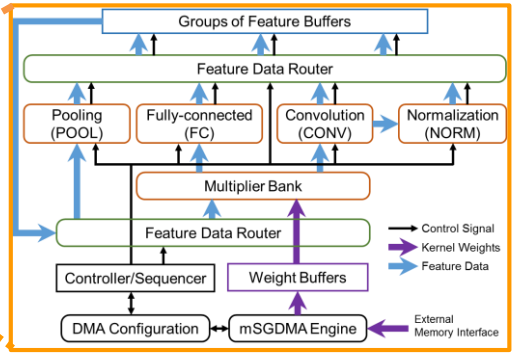
Outline

- Overview of CNN Algorithms
- Current CNN Accelerators & Motivation
- Proposed Modular CNN RTL Compiler
- Experimental Results
- Conclusion

Experimental Setup & FPGA System

- AlexNet and NIN CNN models
- Stand-alone **DE5-Net** board with Altera Stratix-V GXA7 FPGA chip
  - 622K logic elements, 256 DSP blocks, 2560 M20K RAMs.
- Synthesized by Altera Quartus tool.





Transfer data from SDRAM to on-chip RAMs.

Standard Altera IP

Control the transfer of data from flash memory to SDRAM, and then start the CNN acceleration process.

Experimental Results

	J. Qiu *FPGA2016*	C. Zhang *FPGA2015*	N. Suda *FPGA2016*	This work	This work
FPGA	Zynq XC7Z045	Virtex-7 VX485T	**Stratix-V GXA7**	**Stratix-V GXA7**	Stratix-V GXA7
Design Entry	RTL	**C-language**	**OpenCL**	**RTL Compiler**	RTL Compiler
CNN Model	VGG - 16	**AlexNet**	**AlexNet**	**AlexNet**	NIN
# of op. per image	30.76 GOP	1.33 GOP	1.46 GOP	1.46 GOP	2.2 GOP
DSP Utilization	780 (89%)	2,240 (80%)	**256 (100%)**	**256 (100%)**	256 (100%)
Logic Utilization[a]	183K (84%)	186K (61%)	**114K (49%)**	**121K (52%)**	112K (48%)
On-chip RAM[b]	486 (87%)	1,024 (50%)	**1,893 (74%)**	**1,552 (61%)**	2,330 (91%)
Convolution throughput	187.80 GOPS	**61.6 GFOPS**	67.5 GOPS	**134.1 GOPS**	117.3 GOPS
Overall throughput	136.97 GOPS	N/A	**60.2 GOPS**	**114.5 GOPS**	117.3 GOPS

- Compared to OpenCL design, 1.9X overall throughput improvement
  - On the same FPGA board
  - Using similar hardware resources
- Compared to HLS design, 2X convolution throughput improvement

a. Xilinx FPGAs in LUTs and Altera FPGAs in ALMs
b. Xilinx FPGAs in BRAMs (36 Kb) and Altera FPGAs in M20K RAMs (20 Kb)

Experimental Results

	J. Qiu *FPGA2016*	C. Zhang *FPGA2015*	N. Suda *FPGA2016*	This work	This work
FPGA	Zynq XC7Z045	Virtex-7 VX485T	Stratix-V GXA7	Stratix-V GXA7	Stratix-V GXA7
Design Entry	**RTL**	C-language	OpenCL	**RTL Compiler**	RTL Compiler
CNN Model	**VGG - 16**	AlexNet	AlexNet	**AlexNet**	NIN
# of op. per image	30.76 GOP	1.33 GOP	1.46 GOP	1.46 GOP	2.2 GOP
DSP Utilization	**780 (89%)**	2,240 (80%)	256 (100%)	**256 (100%)**	256 (100%)
Logic Utilization[a]	183K (84%)	186K (61%)	114K (49%)	121K (52%)	112K (48%)
On-chip RAM[b]	486 (87%)	1,024 (50%)	1,893 (74%)	1,552 (61%)	2,330 (91%)
Convolution throughput	**187.80 GOPS**	61.6 GFOPS	67.5 GOPS	**134.1 GOPS**	117.3 GOPS
Overall throughput	**136.97 GOPS**	N/A	60.2 GOPS	**114.5 GOPS**	117.3 GOPS

- Model customized RTL and more DSPs improve throughput
- More regular structure of VGG benefits the performance
  - Uniform kernel map size, *Nif* in power of two, no norm

a. Xilinx FPGAs in LUTs and Altera FPGAs in ALMs
b. Xilinx FPGAs in BRAMs (36 Kb) and Altera FPGAs in M20K RAMs (20 Kb)

Experimental Results

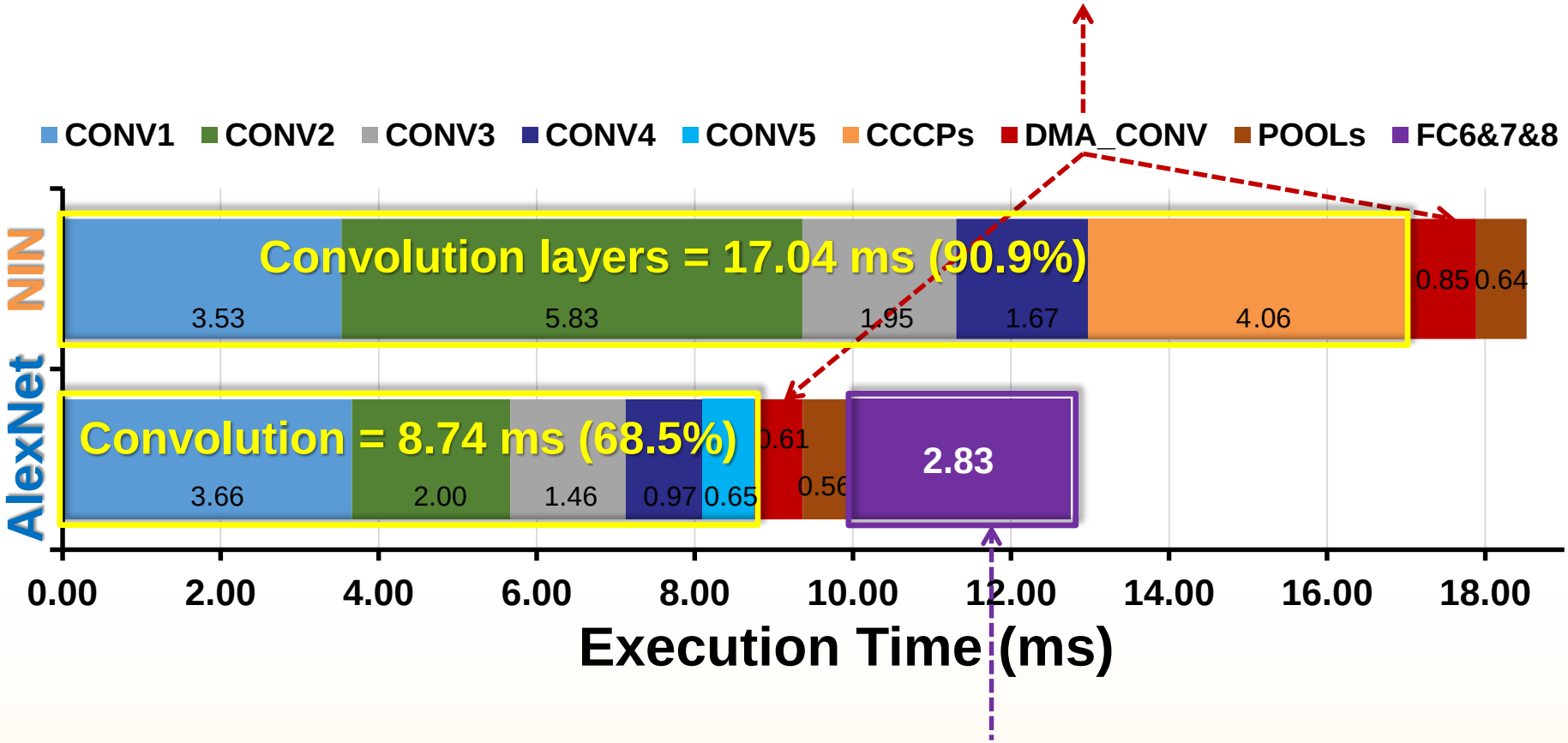
	J. Qiu *FPGA2016*	C. Zhang *FPGA2015*	N. Suda *FPGA2016*	This work	This work
FPGA	Zynq XC7Z045	Virtex-7 VX485T	Stratix-V GXA7	**Stratix-V GXA7**	**Stratix-V GXA7**
Design Entry	RTL	C-language	OpenCL	**RTL Compiler**	**RTL Compiler**
CNN Model	VGG - 16	AlexNet	AlexNet	**AlexNet**	**NIN**
# of op. per image	30.76 GOP	1.33 GOP	1.46 GOP	**1.46 GOP**	**2.2 GOP**
DSP Utilization	780 (89%)	2,240 (80%)	256 (100%)	256 (100%)	256 (100%)
Logic Utilization[a]	183K (84%)	186K (61%)	114K (49%)	121K (52%)	112K (48%)
On-chip RAM[b]	486 (87%)	1,024 (50%)	1,893 (74%)	1,552 (61%)	2,330 (91%)
Convolution throughput	187.80 GOPS	61.6 GFOPS	67.5 GOPS	134.1 GOPS	117.3 GOPS
Overall throughput	136.97 GOPS	N/A	60.2 GOPS	**114.5 GOPS**	**117.3 GOPS**

- NIN has more convolution layers and more operations
- Similar throughput can be achieved for both models

a. Xilinx FPGAs in LUTs and Altera FPGAs in ALMs
b. Xilinx FPGAs in BRAMs (36 Kb) and Altera FPGAs in M20K RAMs (20 Kb)

Timing Breakdown of Layers

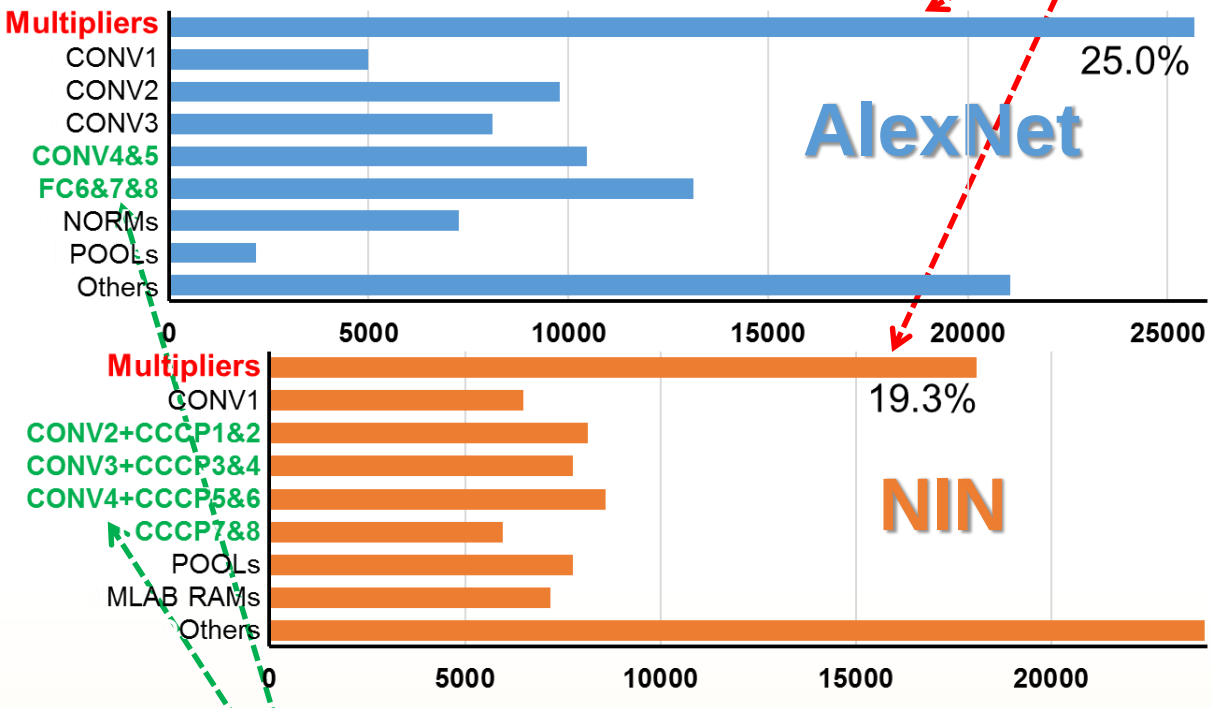
- DMA transfer latency of **CONV weights** is NOT hidden.



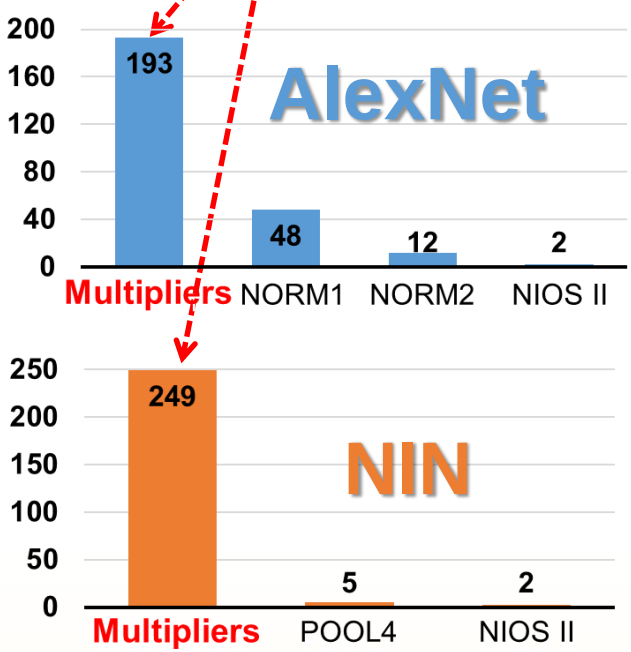
- **FC latency** is determined by the DMA transfer delay that covers the computation latency.

Logic and DSP Block Utilization

- Stratix-V GXA7 has only 256 DSP blocks.
- **Multipliers** are implemented by both **logic** elements and **DSP** blocks.



Logic Utilization in ALMs

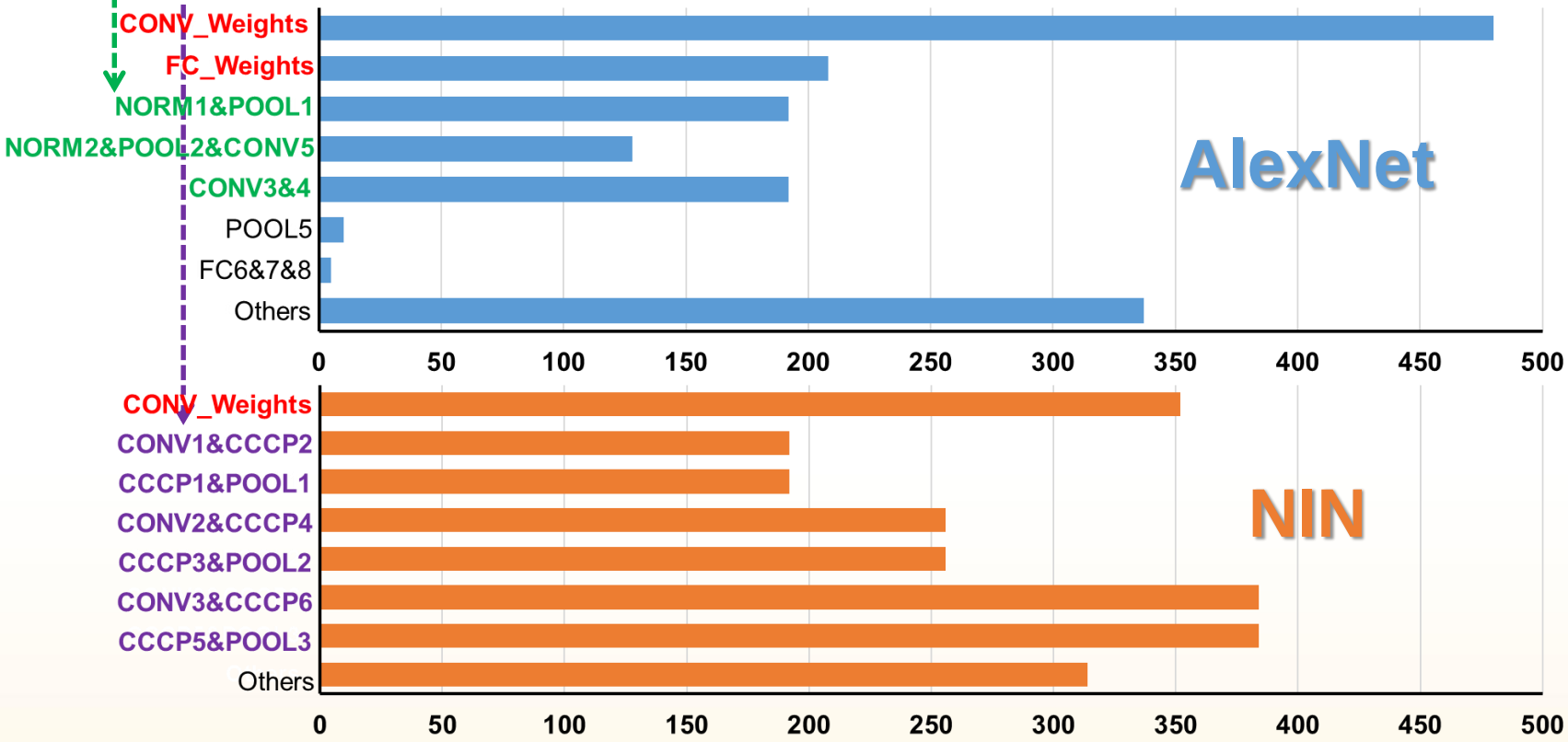


of **DSP** Blocks

- Layers with same *Nif* are **combined** to be one module with shared adder tree.

On-chip RAM Breakdown

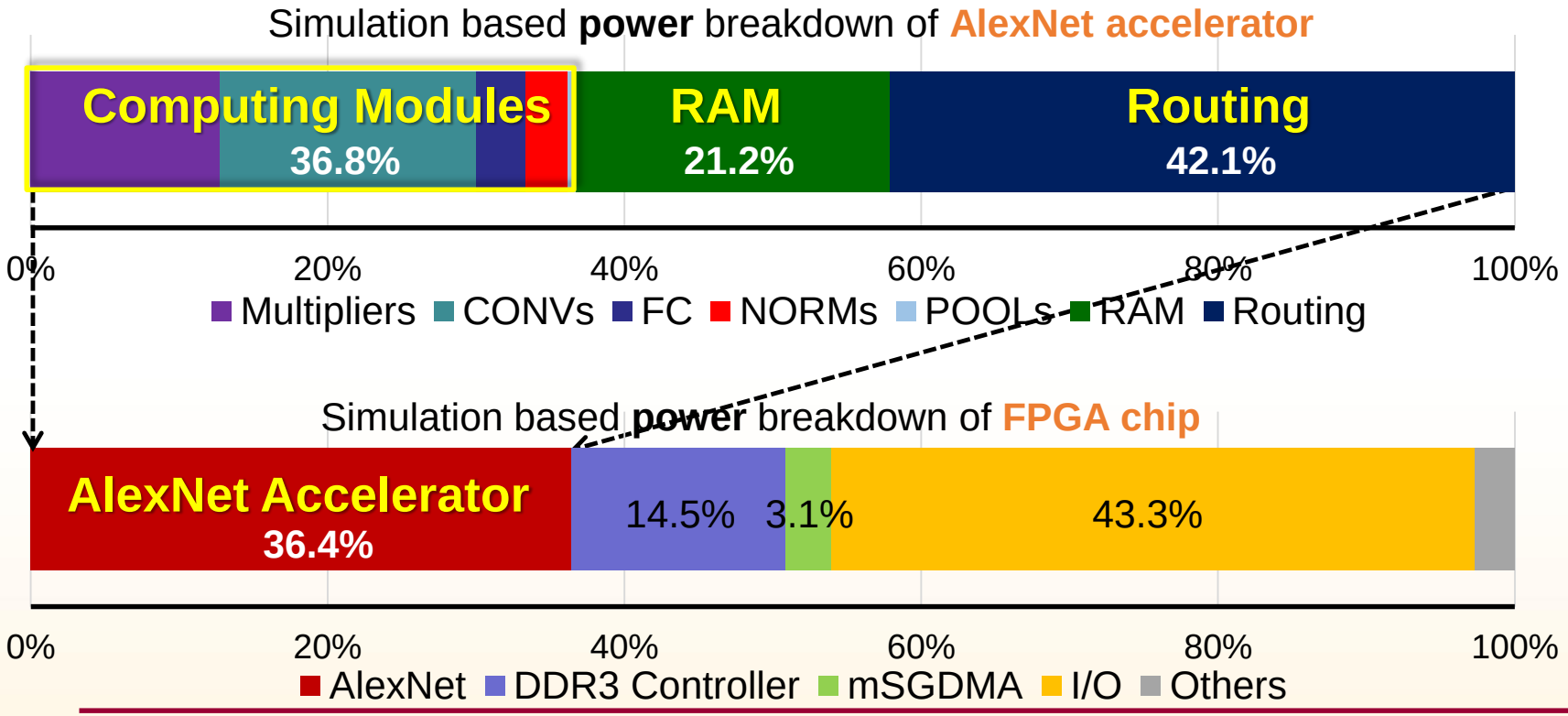
- Weight buffers to receive weights from external memory
- RAMs are stacked for modules with shallow word depths
- RAMs are shared by non-consecutive modules



On-chip **RAMs** Utilization in M20K blocks

Power Measurement and Breakdown

- **Measured power** of DE5-Net
  - running **nothing** is **16.5W**
  - running **AlexNet** is **19.5W**
  - running **NIN** is **19.1W**
- **Simulated power** of Stratix V
  - running AlexNet is **12.8W**



ImageNet Accuracy

- Data width: Features = 10-bit, Weights = 8-bit
- The portion of integer and fractional bits are adjusted according to the range of values for **different layers**.

Reduce data width requirement
while retaining the same accuracy level

Model accuracy comparison	Software implementation (Caffe tool, 32-bit)		FPGA implementation (Our work)	
CNN model	Top-1	Top-5	Top-1	Top-5
AlexNet	56.78 %	79.72 %	55.64 %	79.32 %
NIN	56.14 %	79.32 %	55.74 %	78.96 %

Tested on 5K images from ImageNet 2012 validation database.

Outline

- Overview of CNN Algorithms
- Current CNN Accelerators & Motivation
- Proposed Modular CNN RTL Compiler
- Experimental Results
- Conclusion

Conclusion

- Modularized and scalable RTL design for CNN
- Demonstrated on Altera Stratix-V GXA7 FPGA
- End-to-end implementation of deep CNNs
- 114.5 GOPS for AlexNet and 117.3 GOPS for NIN
- 1.9X performance improvement compared to OpenCL design on the same FPGA
- Future work : increase generality and efficiency for larger state-of-the-art CNNs.

Thanks!

Questions?