AI Accelerator Design Automation: From Network to RTL

Team members: Wen-Cong Huang, Zih-Sing Fu

Advisor: Prof. Chia-Hsiang Yang

Technical Highlights

❖ **A flexible AI accelerator**

An engine for flexible network inference, including highly configurable descriptor, ISA and hardware co-design

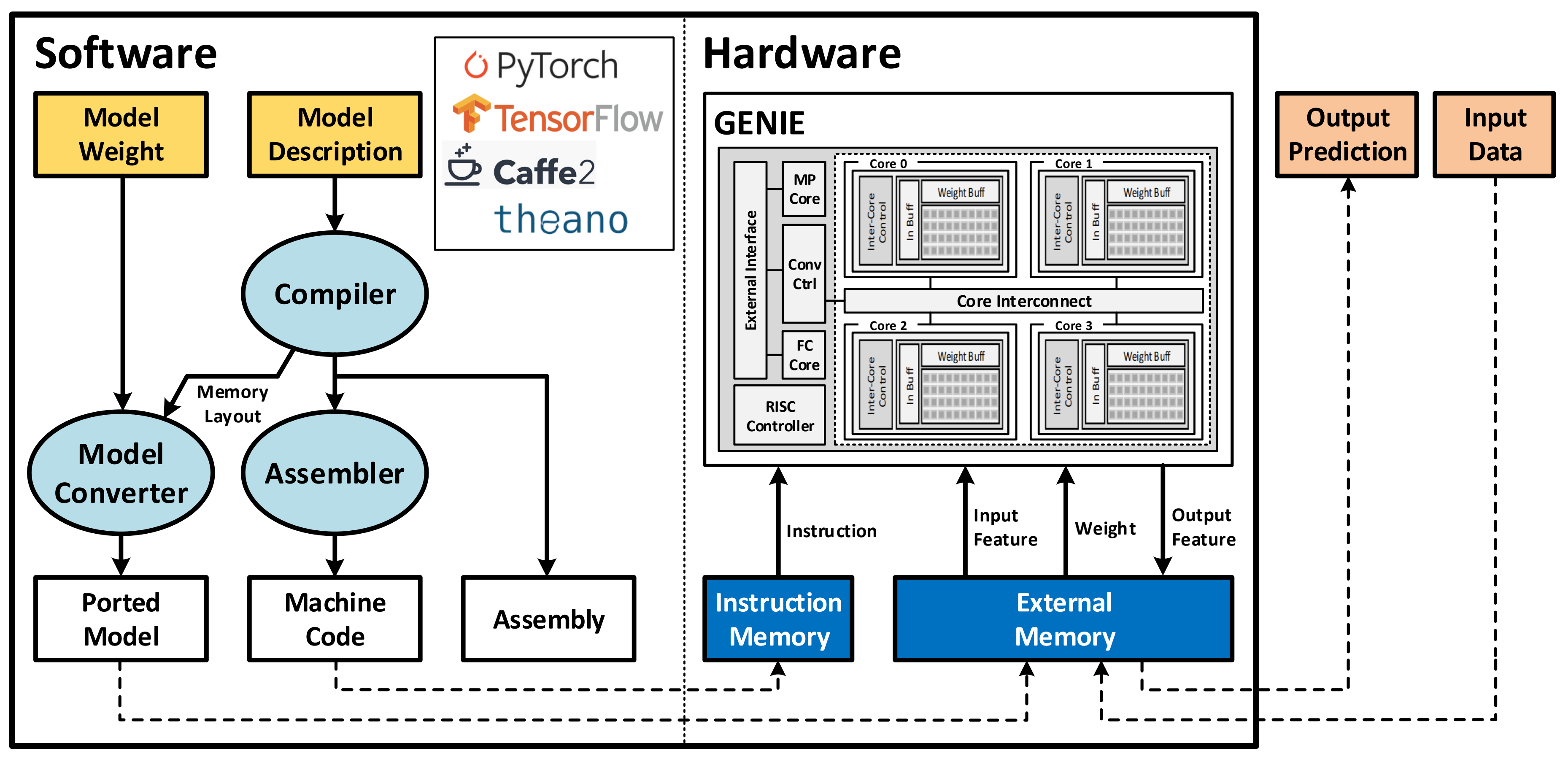
❖ **An easy-to-use AI accelerator**

Support complete inference flow of arbitrary network architecture with provided model descriptor and weight

❖ **Technical focus**

- ❑ Model compiler from custom model descriptor
- ❑ Software and hardware co-design for AI applications
- ❑ Optimal data scheduling for multi-core processing
- ❑ Optimal tensor partition strategy exploration

Design Automation and User-friendly Workflow



Flexible Model Descriptor

❖ **Sequential model descriptor**

❖ **Supported operations**

- ❑ 2D convolutional layer
  - Arbitrary input feature shape
  - Flexible activation/bias toggle
- ❑ Fully-connected layer
- ❑ Max pooling layer
- ❑ ReLU activation

```
model = nn.Sequential(
    nn.Conv2d(1, 16, 3, padding=1, bias=True),
    nn.Conv2d(16, 16, 3, padding=1, bias=True),
    nn.Conv2d(16, 16, 3, padding=1, bias=True),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2),
    nn.Conv2d(16, 32, 3, padding=1, bias=False),
    nn.Conv2d(32, 32, 3, padding=1, bias=False),
    nn.Conv2d(32, 32, 3, padding=1, bias=False),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2),
    nn.Conv2d(32, 64, 3, padding=1, bias=True),
    nn.Conv2d(64, 64, 3, padding=1, bias=True),
    nn.Conv2d(64, 64, 3, padding=1, bias=True),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2),
    Flatten(),
    nn.Linear(576, 10, bias=False),
)
```

model descriptor

```
ifdim 1,28,28
conv 1,16,3,1 noact bias
conv 16,16,3,1 noact bias
conv 16,16,3,1 relu bias
maxpool
conv 16,32,3,1 noact nobias
conv 32,32,3,1 noact nobias
conv 32,32,3,1 relu nobias
maxpool
conv 32,64,3,1 noact bias
conv 64,64,3,1 noact bias
conv 64,64,3,1 relu bias
maxpool
flatten
fc 576,10 relu nobias
```

compiled instructions

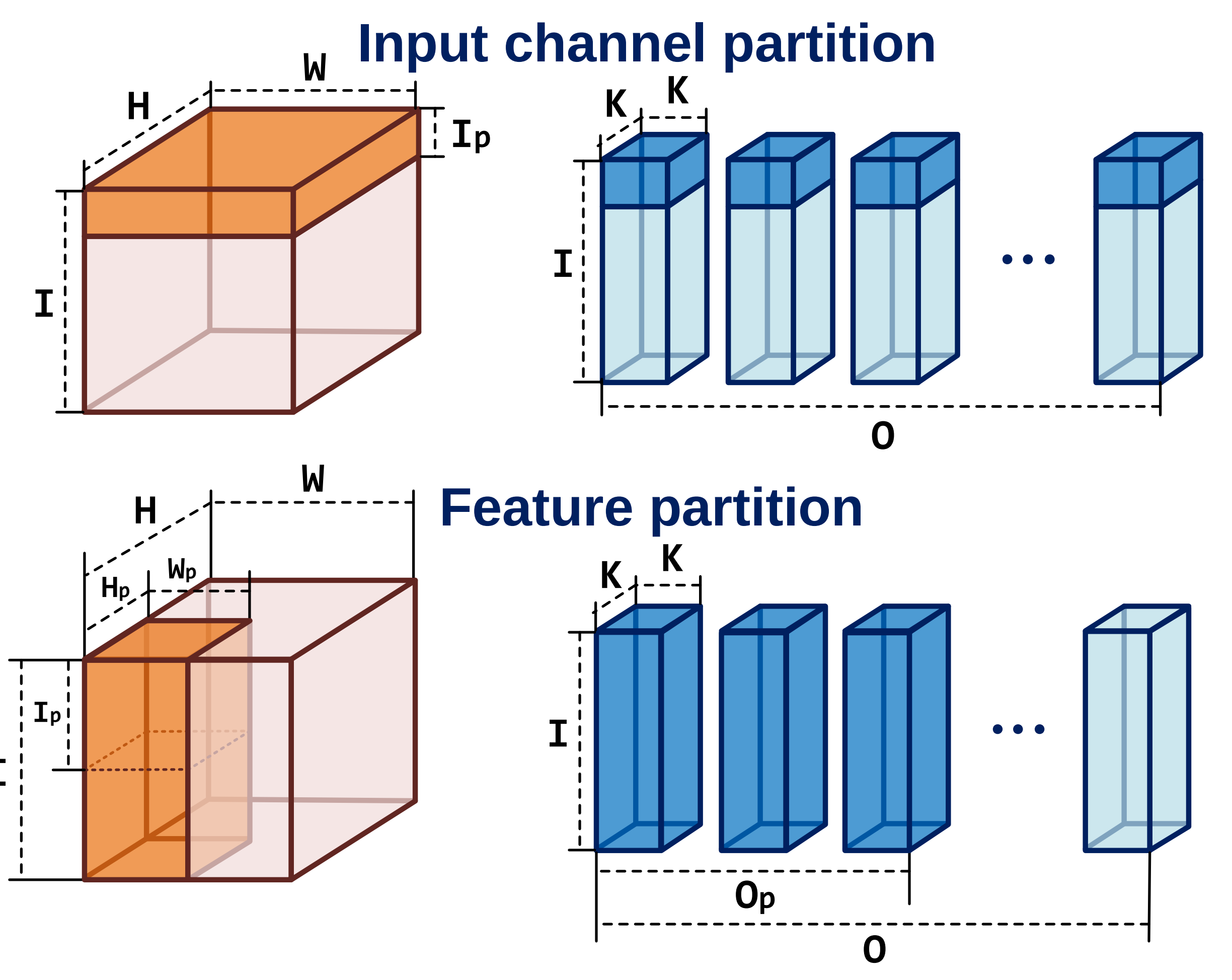
Memory Scheduling and Optimization

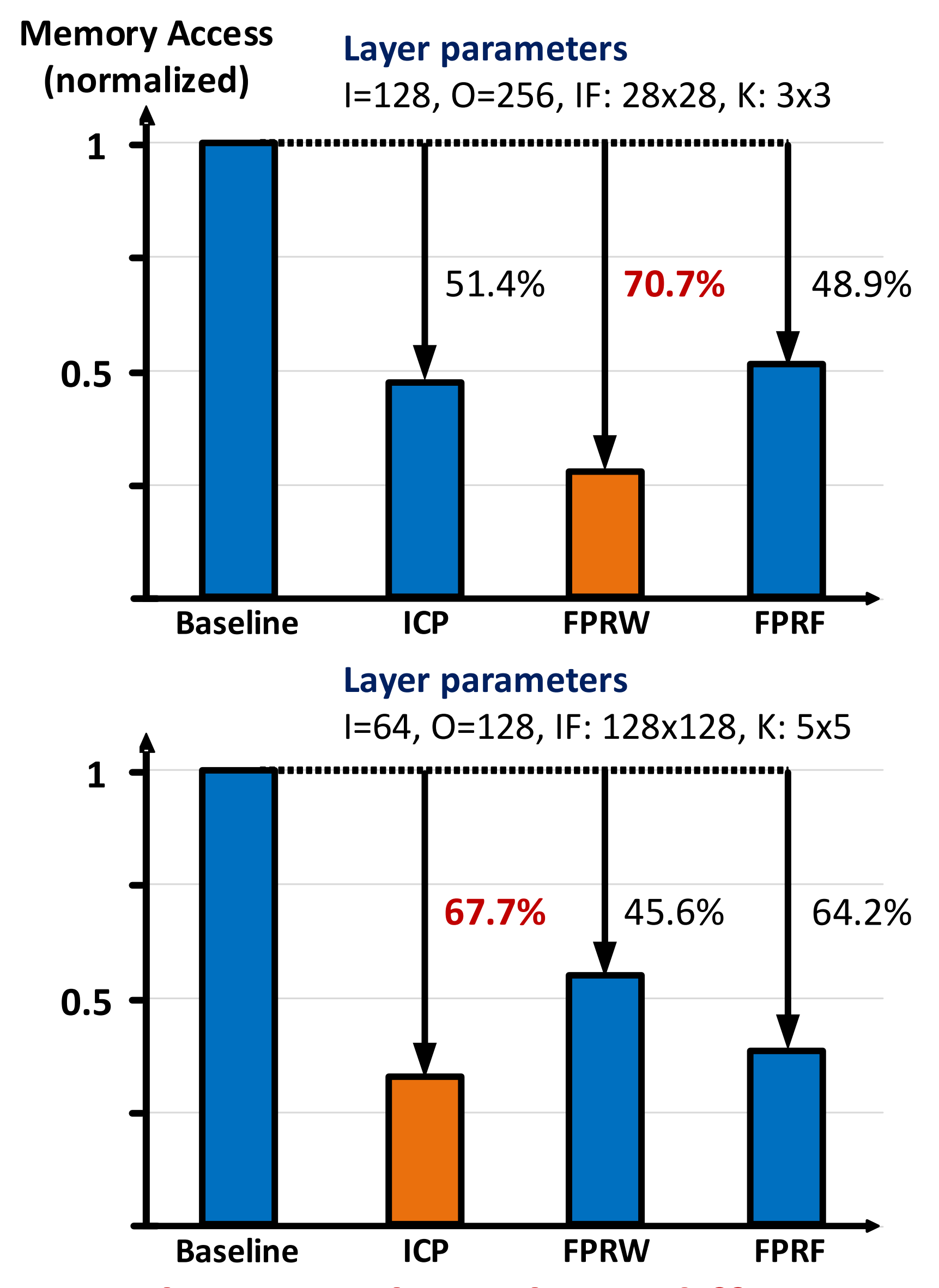
❖ **Support for arbitrary feature size require tensor partitioning**

❖ **Memory access and on-chip memory should be minimized**

❖ **Data scheduling strategies**

1. **Baseline** (no partition reuse)
2. **ICP**: Input channel partition
3. **FPRW**: Feature partition (reload weight)
4. **FPRF**: Feature partition (reload feature)





The compiler selects different strategies to optimize memory access for different layers

ISA and Instruction Generation

❖ **Instruction Set Architecture**

- ❑ 32-bit RISC ISA
- ❑ 19 instructions

❖ **Instruction generation**

Algorithm 6 Convolution Instruction Generation Routine (Single Core)

```
1: procedure ConvRoutine
2:     cfgl conv, activation, bias
3:     cfgcv I, O, K, pad
4:     cfgcvif H, W
5:     cvaif ifbase
6:     cvaw wbase
7:     cvaof ofbase
8:     for $O_{ori} \leftarrow 0$ to $O$ step $O_p$ do
9:         cvcfgori 0, $O_p$
10:        cvcfgext 0, $\min(O - O_p, O_p)$
11:        cvlwp
12:        for $H_{ori} \leftarrow -pad$ to $H + pad$ step $H_p - K + 1$ do
13:            for $W_{ori} \leftarrow -pad$ to $W + pad$ step $W_p - K + 1$ do
14:                cvcfgori $H_{ori}, W_{ori}$
15:                cvcfgext $\min(H + pad - H_{ori}, H_p), \min(W + pad - W_{ori}, W_p)$
16:                for $I_{ori} \leftarrow 0$ to $I$ step $I_p$ do
17:                    cvcfgori $I_{ori}$, hold
18:                    cvcfgext $\min(I - I_{ori}, I_p)$, hold
19:                    cvlifp
20:                cvsofp
```

- Configurations

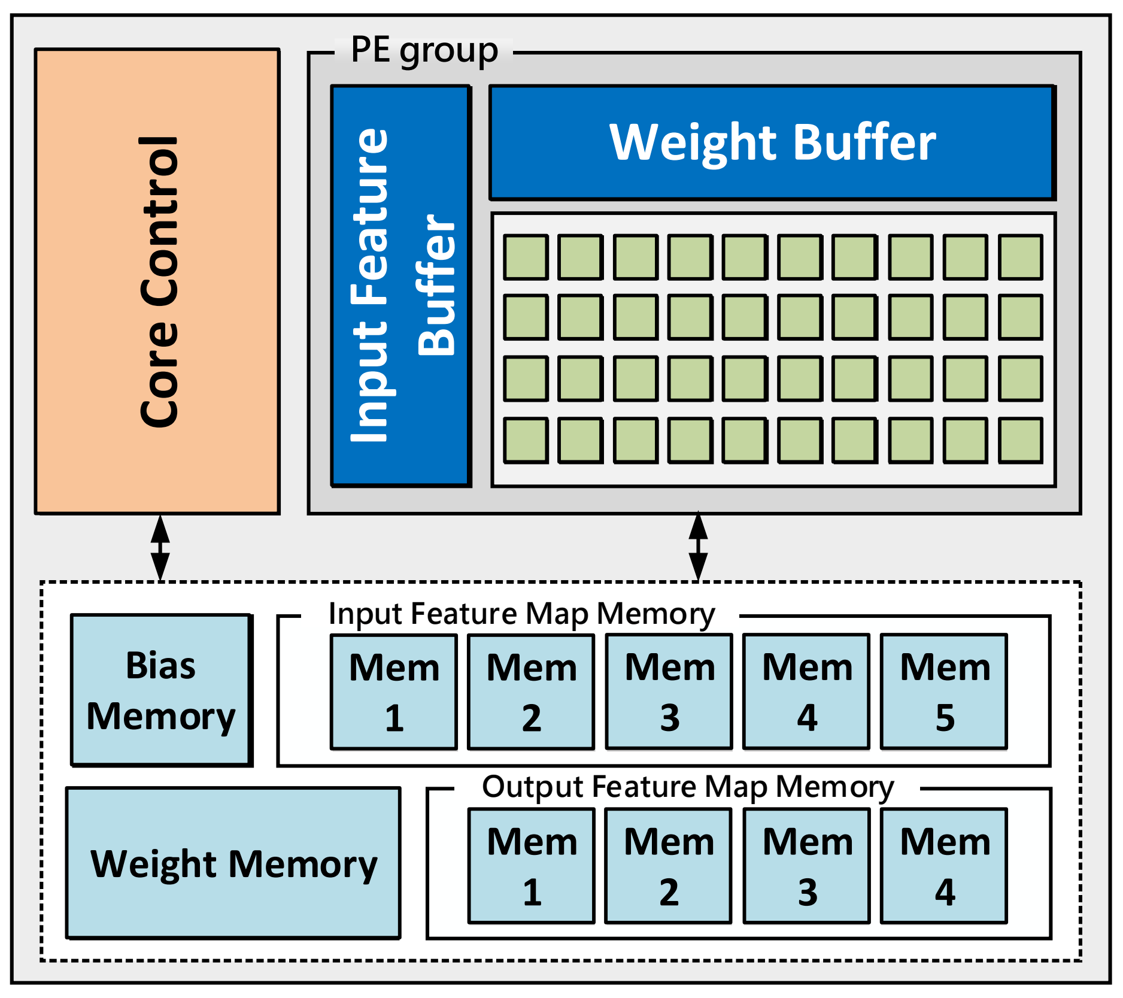
	31:27	26:16	15:5	4:3	2:0
cfgcv	11	Cin	Cout	padding	kernel

- Multi-core support

	31:27	26:9	8	7:0
cvselpe	16	reserved	broadcast	peid

AI Accelerator Design

❖ **Core Architecture**



❖ **System Specifications**

- ❑ 4-core architecture for bandwidth improvement
- ❑ Configurable memory options for performance tweaking

❖ **Memory Estimation**

Memory Type	Memory size	Estimated area
Ifmap Memory	~25 KB	$175000\mu m^2$
Ofmap Memory	~18.5 KB	$129500\mu m^2$
Weight Memory	~73.7 KB	$515900\mu m^2$
Bias Memory	~128B	$896\mu m^2$
Total Memory	~120KB	$840000\mu m^2$

Research Outcomes

❖ **Practical**: from network model to RTL in minutes

❖ **Efficient**: more than 60% less external memory access

❖ **Reconfigurable**: different memory size for different scenarios

❖ **Ongoing tasks**: AXI-compatibility, higher external throughput

1. Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *JSSC*, Jan. 2017.
2. Lee, Jinmook, et al. "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," *ISSCC*, Feb. 2018.
3. Shin, Dongjoo, et al. "14.2 DNPU: An 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," *JSSC*, Feb. 2017.
4. Cheng-Hsun Lu et al., "A Fully-Programmable Deep Learning Processor with Adaptable Intelligence"