DNPU: An Energy-Efficient Deep Neural Network Processor with On-Chip Stereo Matching

Dongjoo Shin, **Jinmook Lee, Jinsu Lee,**
Juhyoung Lee, and Hoi-Jun Yoo

Semiconductor System Laboratory
School of EE, KAIST

About DNPU v1 *– Designed in 2015*

- **D**eep **N**eural network **P**rocessing **U**nit
- Embedded Deep Neural Network Processing in Mobile Platforms

- **Heterogeneous Architecture** for Convolutional Layers vs MLP-RNN
- **Convolution Processor**
  - Mixed workload division method
  - Layer-by-layer dynamic-fixed-point operation with on-line adaptation
- **MLP-RNN Processor**
  - LUT-based multiplication with weight quantization (Q-table)

- **Stereo Matching Processor** for Depth Map Generation
- **RGB-D 4-ch** Processing Support

MLP: Multi-layer Perceptron

RNN: Recurrent Neural Network

Targets of DNPU

- Embedded Deep Neural Network Processing in Mobile Platforms

- **Platform: Mobile**
  → Low-power and high energy efficiency
- **Task: Vision**
  – 4.7GB/min (HD 30fps) → Bottleneck for cloud computing
- **Operation: Real-time & low-latency**
  → High throughput and embedded computing

DNN-dedicated SoC

- **Smart Machines & Intelligence-on-Things (IoT)**
  – Robot, drone, smartphone, wearable devices, home appliances

Why both CNN & RNN?

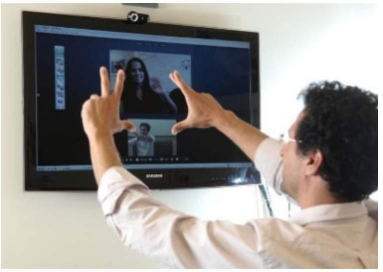
CNN: Convolutional Neural Network
RNN: Recurrent Neural Network

- **CNN: Visual feature extraction and recognition**
  - **Face recognition, image classification…**
- **RNN: Sequential data recognition and generation**
  - **Translation, speech recognition…**
- **CNN + RNN: CNN-extracted features → RNN input**

Image Captioning

CNN: Visual feature extraction

RNN: Sentence generation



Action Recognition

CNN: Visual feature extraction

RNN: Temporal information recognition

- **Previous works**
  - **Optimized for convolution layers only: [1], [2], [3], [4]**
  - **Optimized for MLP and RNN only: [5]**

[1] Y. Chen, ISSCC 2016 [2] B. Moons, SOVC 2016 [3] B. Moons, ISSSC 2017 [4] G. Desoli, ISSCC 2017 [5] M. Price, ISSCC 2017

Hardware for Deep Neural Networks

High

Dedication

Low

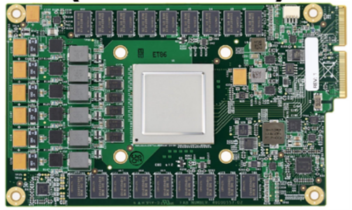
CPU

Intel Xeon
E7-8894 v4
14 nm, 2.4 GHz
11 GOPS/W (FP 32)

GPU

Nvidia P100
16 nm, 1.3 GHz
64 GOPS/W (FP 16)

ASIC
(Server)



Google TPU
28 nm, 0.7 GHz
2.3 TOPS/W (INT 8)

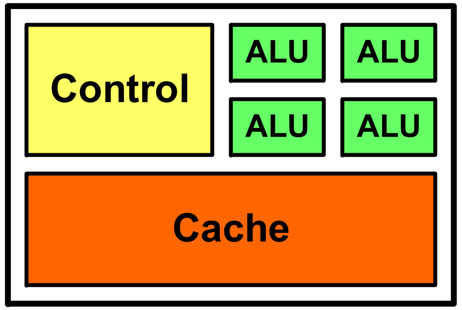
ASIC
(Mobile)

KAIST DNPU
65 nm, 0.1 GHz
4.2 TOPS/W (INT 8)

Low Energy Efficiency High

DNPU: DNN-dedicated SoC

CPU

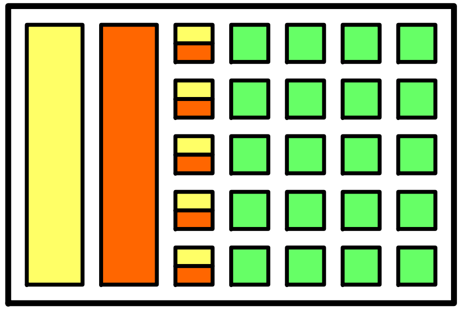


Complex control logic

Low compute density

High programmability

GPU

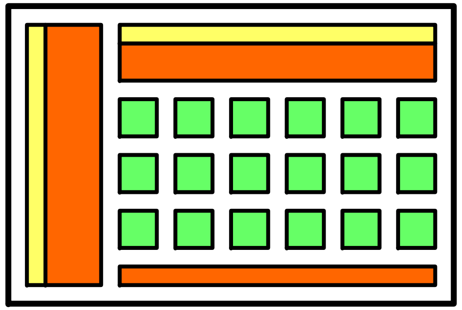


Parallel computation

High compute density

Floating point data type

ASIC (server)

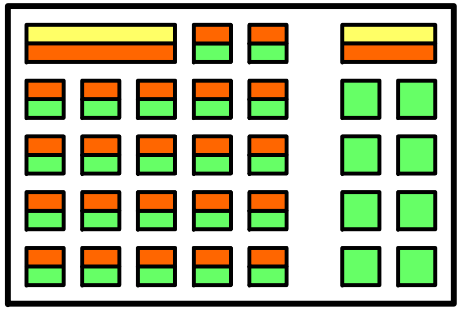


Matrix multiply arch.

Dedicated memory arch.

Fixed point data type

DNPU (mobile)



Heterogeneous ***arch.***

Fixed computation pattern

Dedicated data type

Parallel Processing

Dedicated Architecture

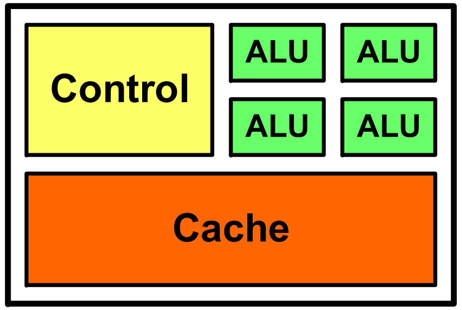
Heterogeneous Architecture

Higher Energy Efficiency (Operations/W) →

← **Higher Programmability**

DNPU: DNN-dedicated SoC

CPU

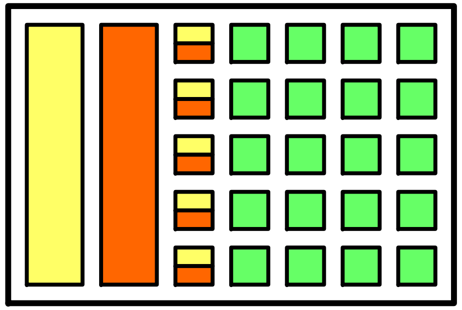


Complex control logic

Low compute density

High programmability

GPU

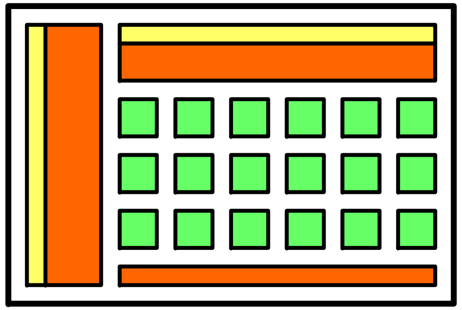


Parallel computation

High compute density

Floating point data type

ASIC (server)

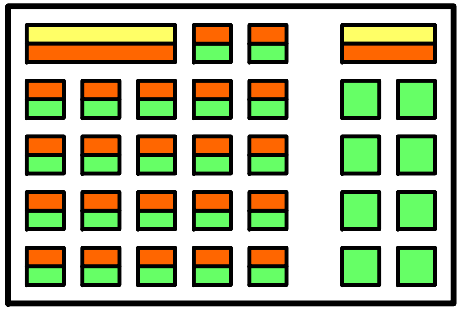


Matrix multiply arch.

Dedicated memory arch.

Fixed point data type

DNPU (mobile)



Heterogeneous arch.

Fixed computation pattern

Dedicated data type

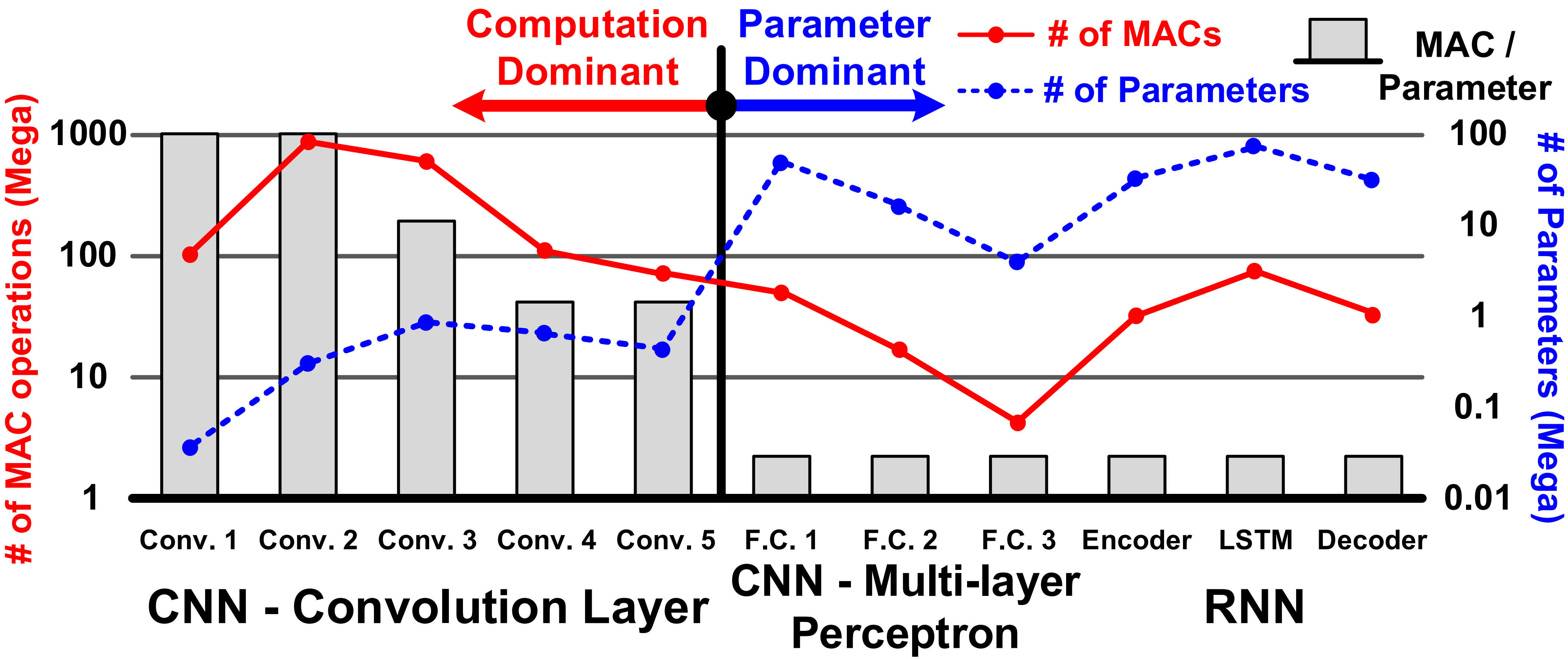
Parallel Processing

Dedicated Architecture

Heterogeneous Architecture

DNN itself has high adaptability for various applications

Heterogeneous Characteristics



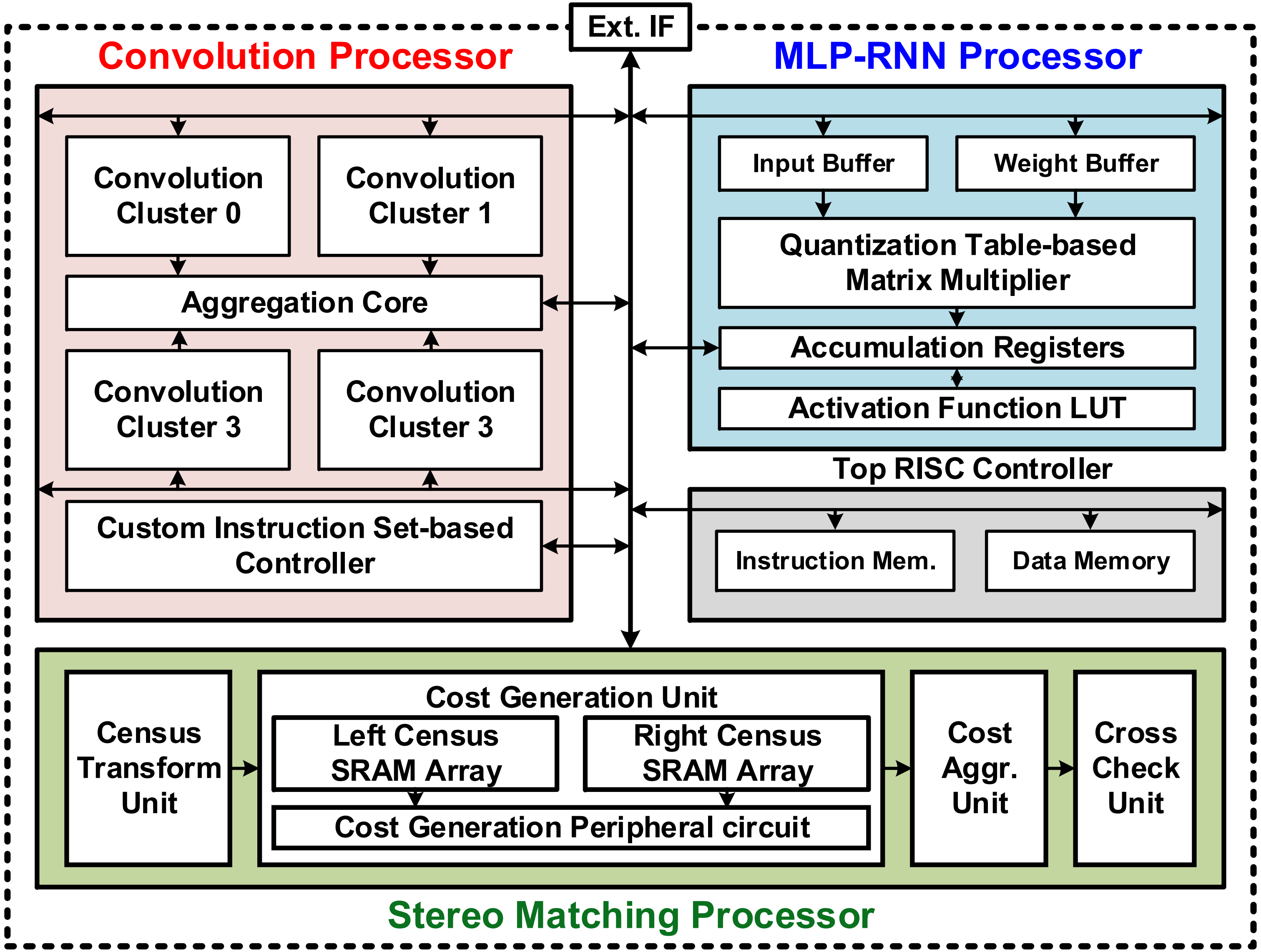
- Convolution Layer (CNN): Computation >> Parameter
- MLP (of CNN), RNN: Computation ≈ Parameter

Heterogeneous Architecture

- **Architecture for Convolution Processor**
  - Image, kernel, convolution reuse architecture
  - Dynamic fixed-point with on-line adaptation (Feature map reduction)
  - Distributed memory
  - On-chip memory portion: Input > weight > output

- **Architecture for MLP-RNN Processor**
  - Matrix multiply architecture
  - Weight quantization (Weight reduction)
  - LUT-based multiplication (Quantization-table or Q-table)
  - On-chip memory portion: Output > weight > input

Overall Architecture



Convolution Processor

- **Kernel**: Support any size
  - Maximum utilization @ 3xn, 6xn, 9xn, …: 100%,
  - Minimum utilization @ 1x1: 33%
- **Stride**: 1, 2, 4
- **Channel**: Support any size
  - Optimized for 16, 32, 64, 128, 256, 512, 1024
- **Pooling**: 2 x 2
- **Activation**: ReLU

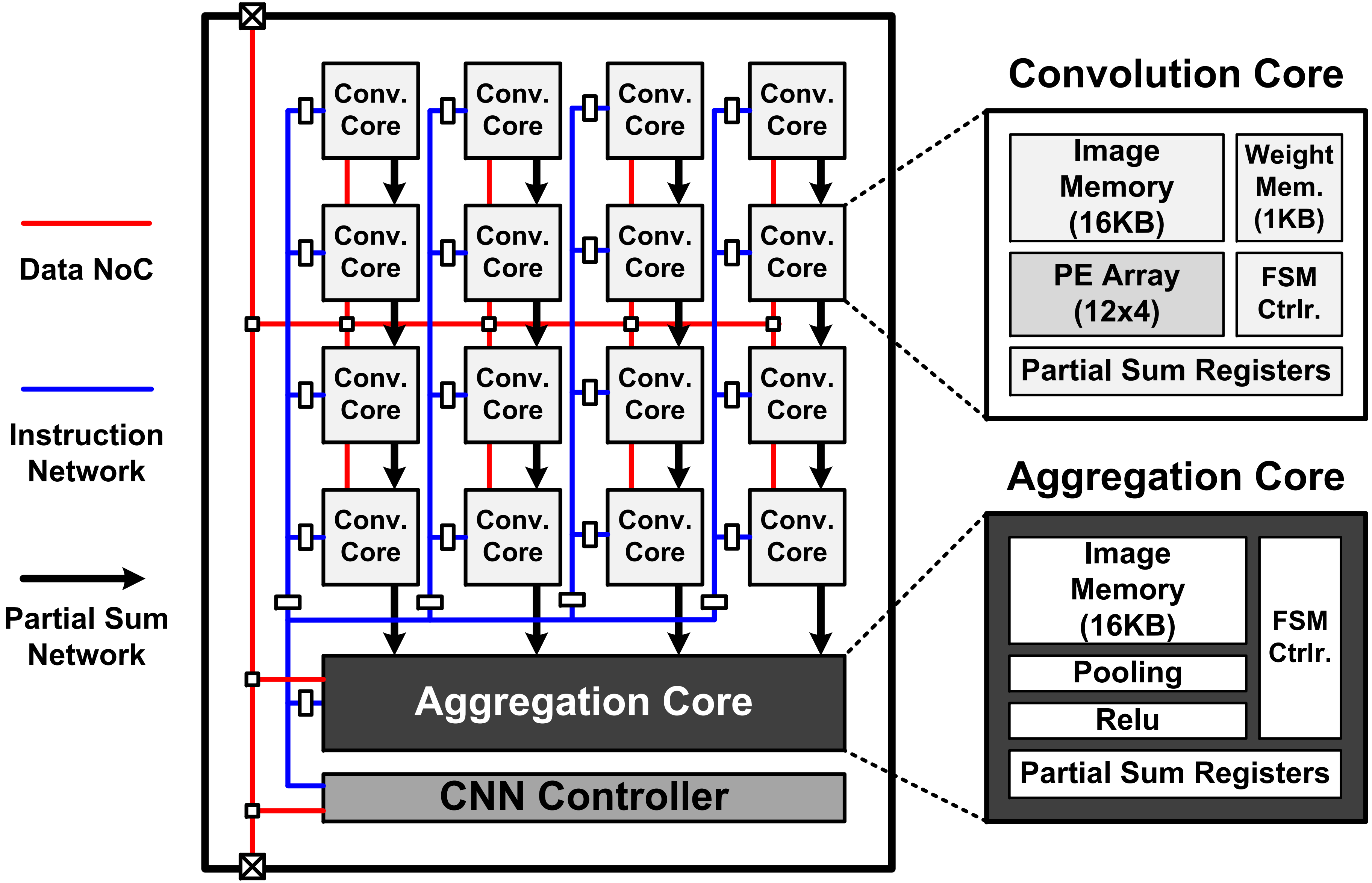
MLP-RNN Processor

- **Channel**: Support any size
- **Activation**: ReLU, sigmoid, tanh

Stereo Matching Processor

- **Depth level**: 64
- **Input image**: QVGA

Convolution Processor



Mixed Workload Division Method

- Limited on-chip memory size → Workloads should be divided

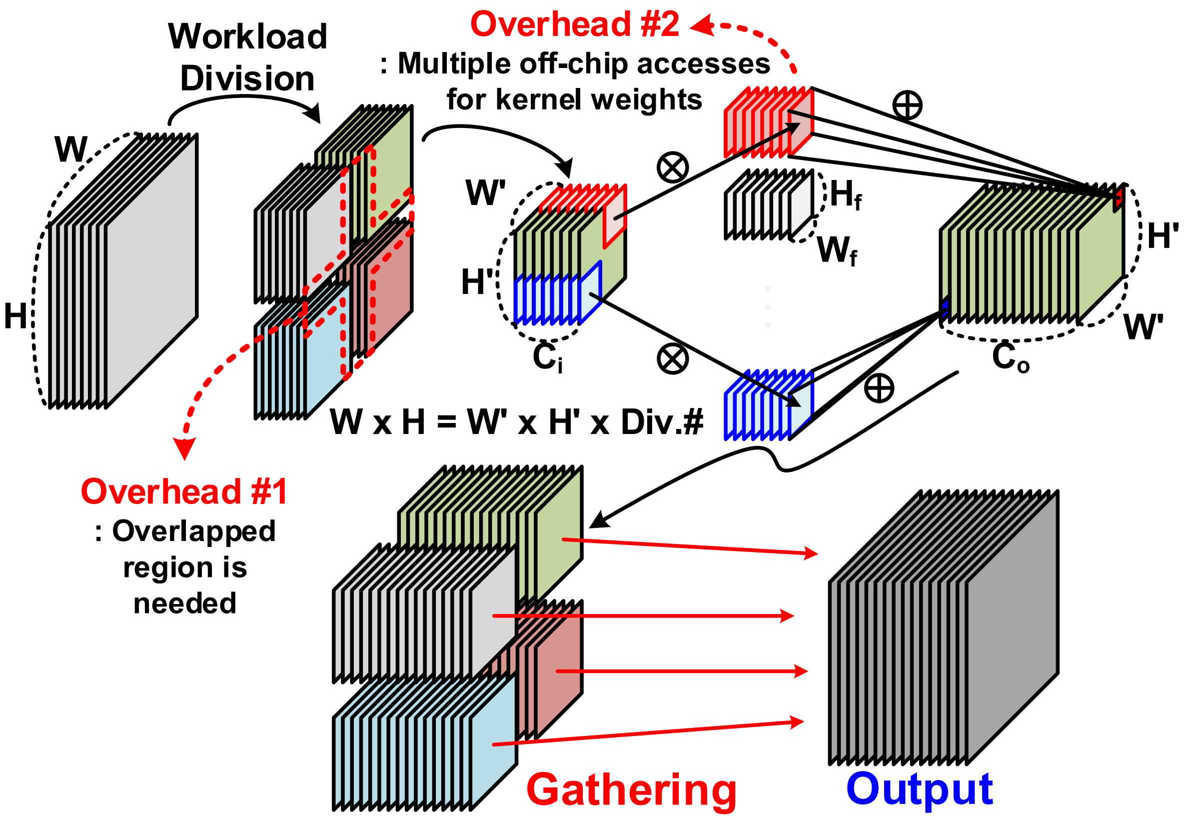
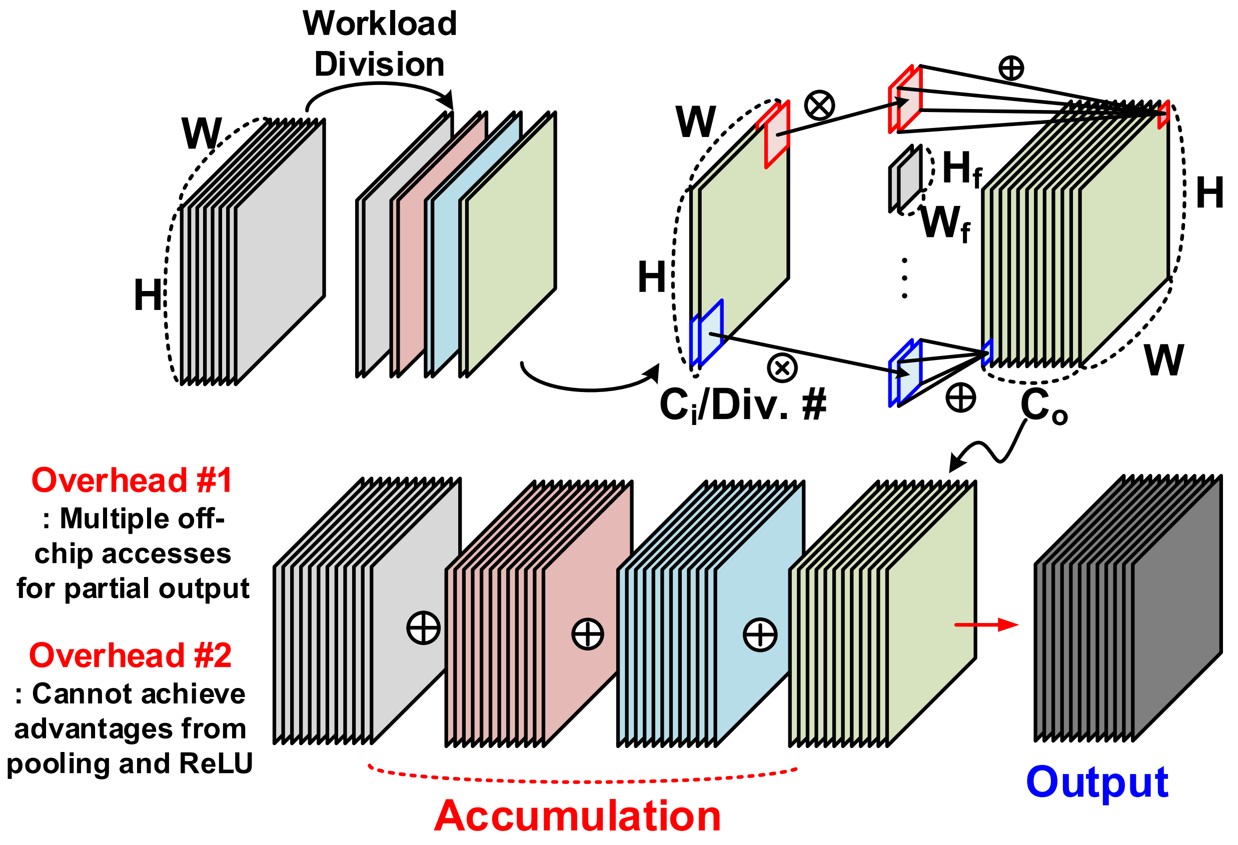
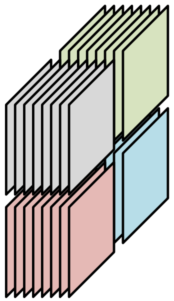
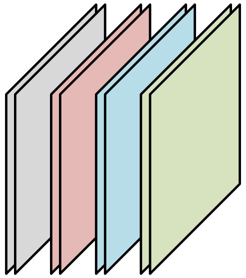
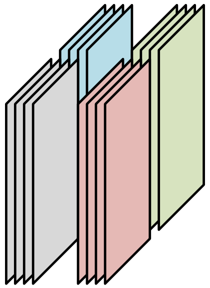


Image Division

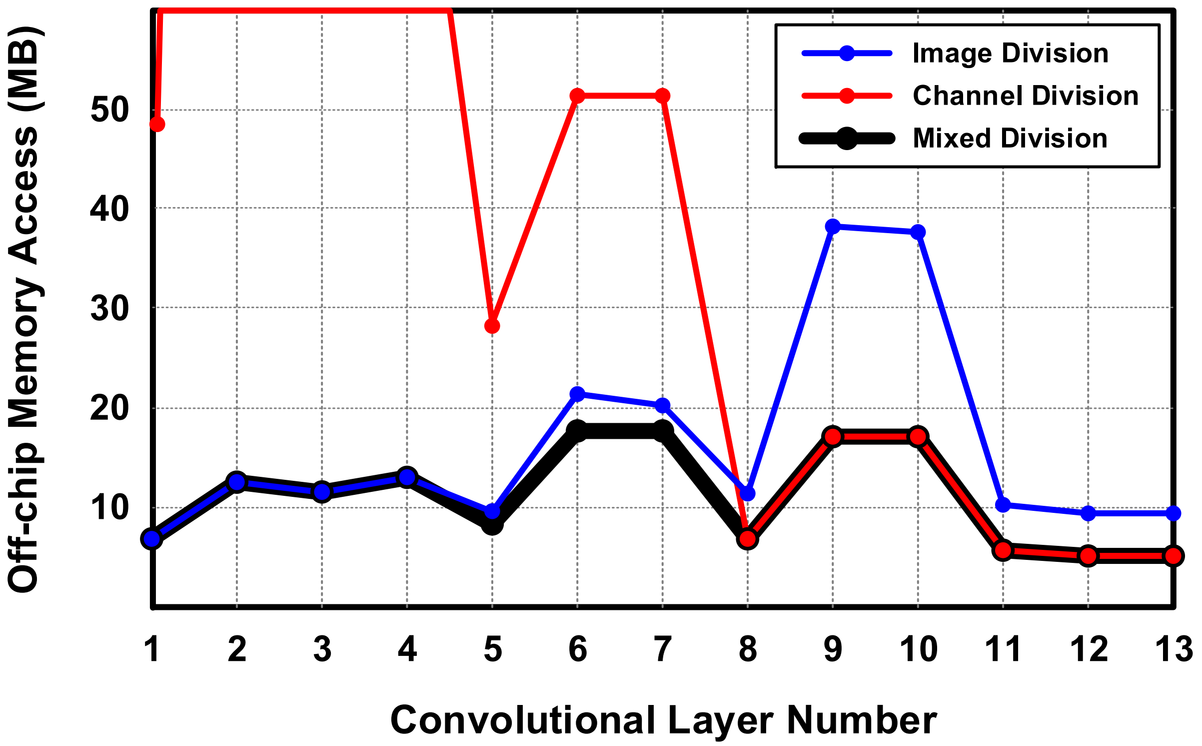


Channel Division

Mixed Workload Division Method

	Input Layer Division Method		
	Image Division	Channel Division	Mixed Division
			
	Multiple off-chip accesses for **weight**	Multiple off-chip accesses for **partial output**	Use both divisions
	Having advantage: image >> **weight**	Having advantage: **image** << weight	
Off-chip Access (W/O Compression Scheme)			
Input Image	$W_i \times H_i \times C_i$	$W_i \times H_i \times C_i$	$W_i \times H_i \times C_i$
Weight	$W_f \times H_f \times C_i \times C_o$ $\times$ Img. Div. #	$W_f \times H_f \times C_i \times C_o$	$W_f \times H_f \times C_i \times C_o$ $\times$ Img. Div. #
Output Image	$\frac{W_o \times H_o \times C_o}{\text{Pooling Size}}$	$W_o \times H_o \times C_o$ $\times$ Ch. Div. # $\times$ 2	$W_o \times H_o \times C_o$ $\times$ Ch. Div. # $\times$ 2

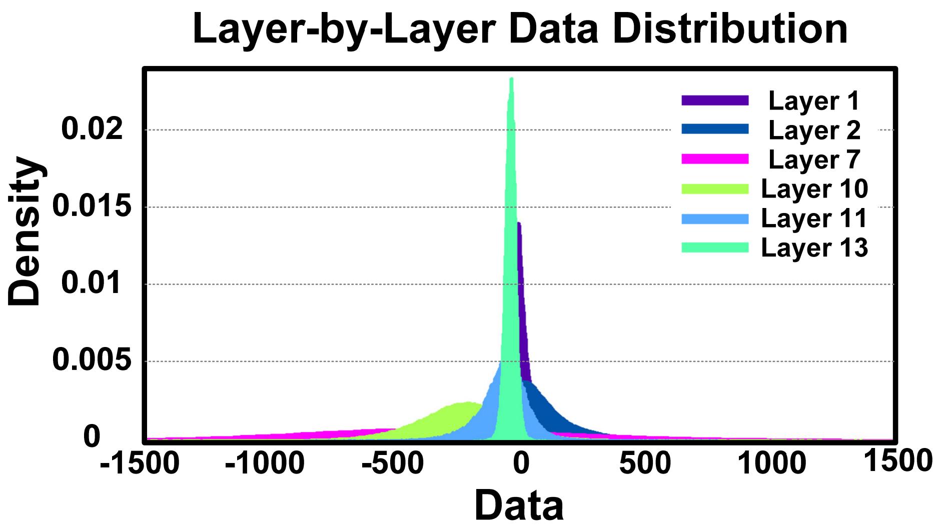
VGG-16 Off-chip Memory Access Analysis

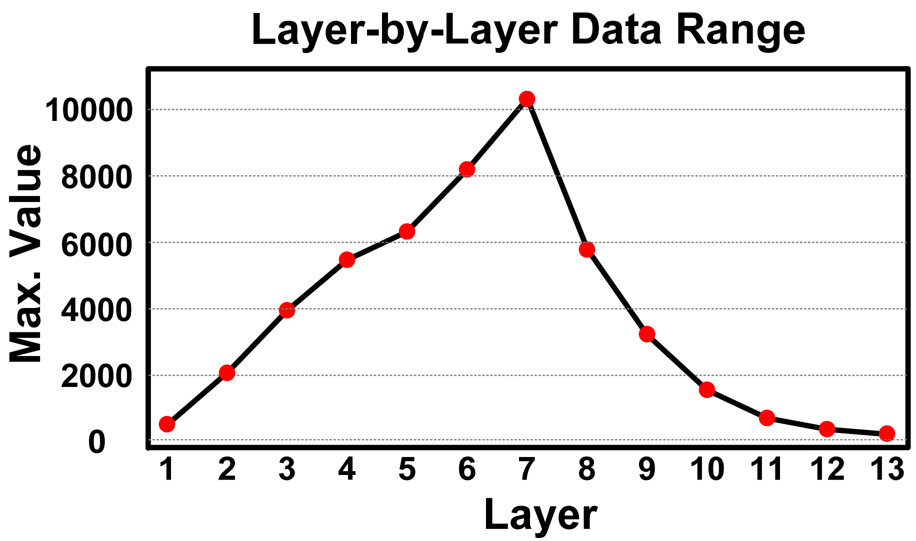


→ ***Mixed division can take lower points***

Layer-by-Layer Dynamic Fixed-point

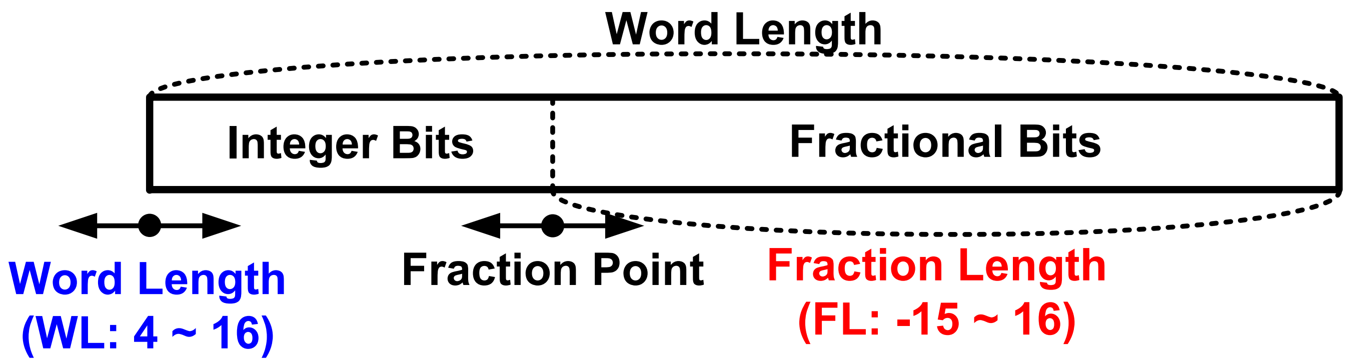
- **Data distribution in each layer**





- **Floating-point implementation**
  - **Large data range, but high cost**
- **Fixed-point implementation**
  - **Low cost, but limited data range**

Layer-by-Layer Dynamic Fixed-point



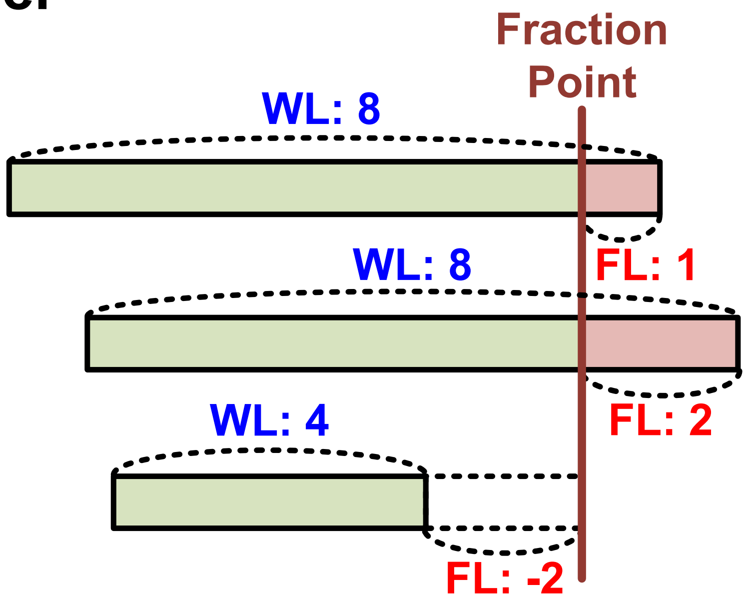
- **Each layer has different WL and FL**
  → Having floating-point characteristic via layers
- **WL and FL are fixed in a same layer**
  → Enabling fixed-point operation

Example)

***Conv. Layer 1* – WL: 8, FL: 1**

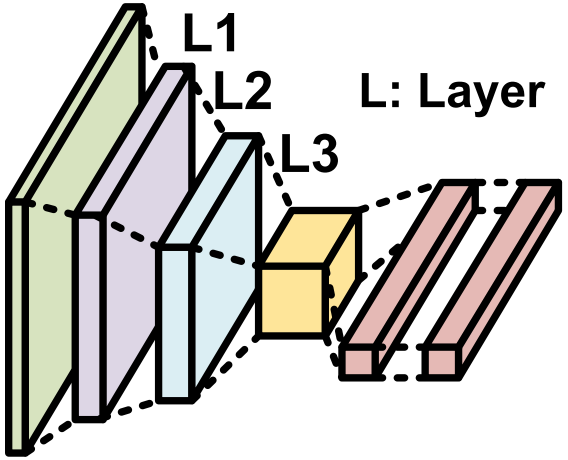
***Conv. Layer 2* – WL: 8, FL: 2**

***Conv. Layer 3* – WL: 4, FL: -2**



Off-line Learning-based Approach (Previous)

Image Data Set



Fraction Length (FL) Sets

Set1 – L1: 0, L2: 0, L3:-4 …

Set2 – L1: 1, L2: -1, L3: -5 …

Set3 – L1: 1, L2: -3, L4: -6 …

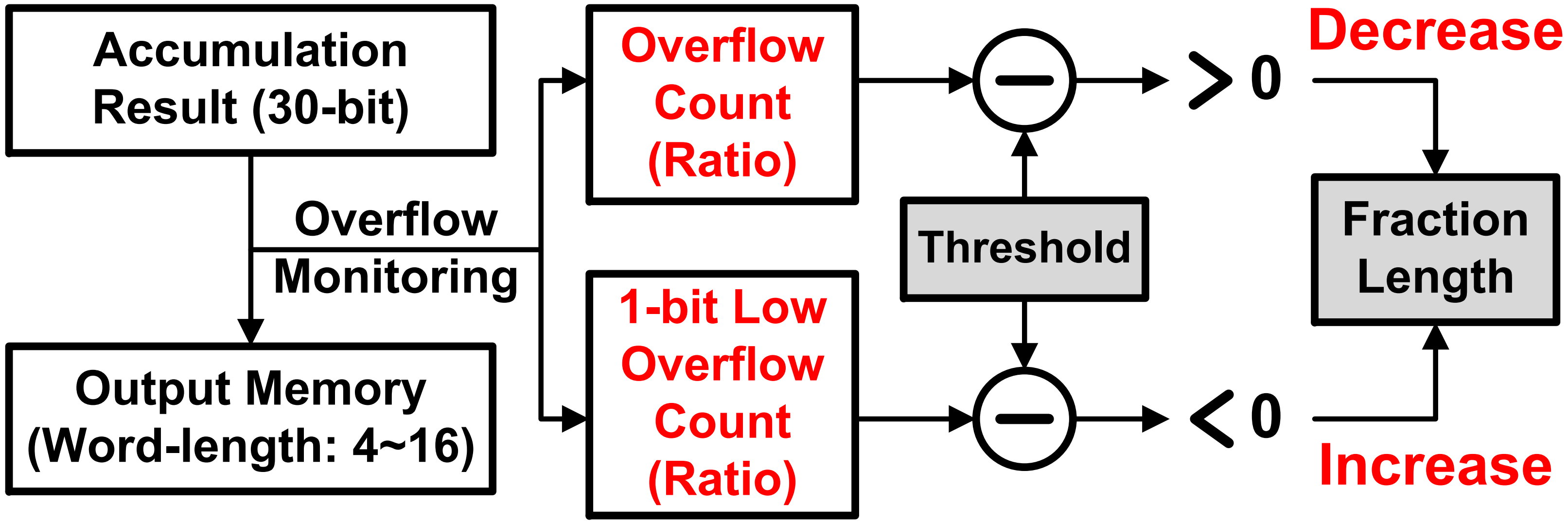
⋮

Finding **FL set** which shows the **minimum error** with given **image data set**

- **Off-line (off-chip)** learning-based FL selection
- FL is ***trained to fit*** with ***given image data set***
- Selected FL is used for every image at run time

Proposed On-line Adaptation

- **On-line adaptation-based FL selection**



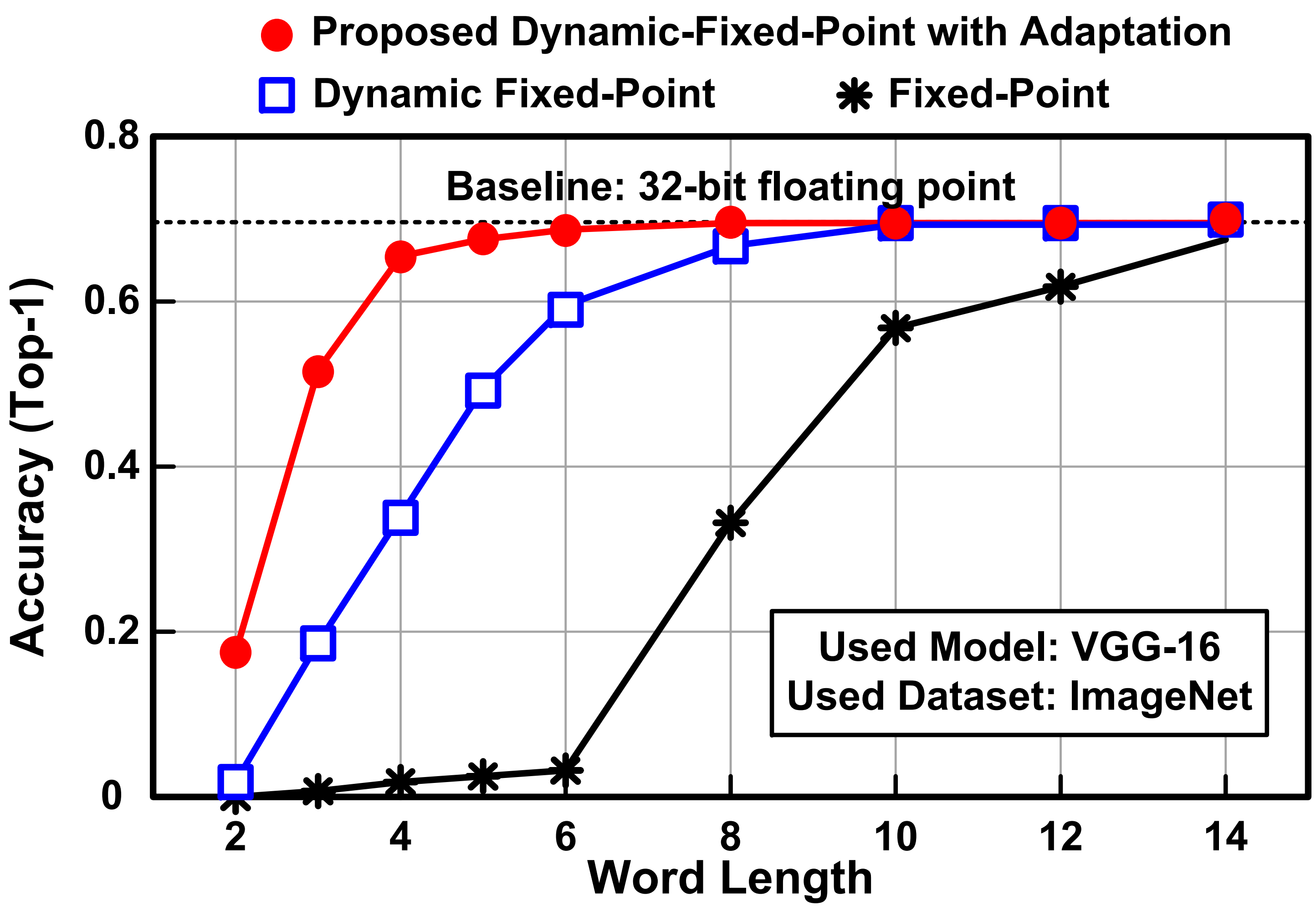
– **On-line (on-chip)** learning-based FL selection

– FL is ***dynamically fit*** to ***current input image***

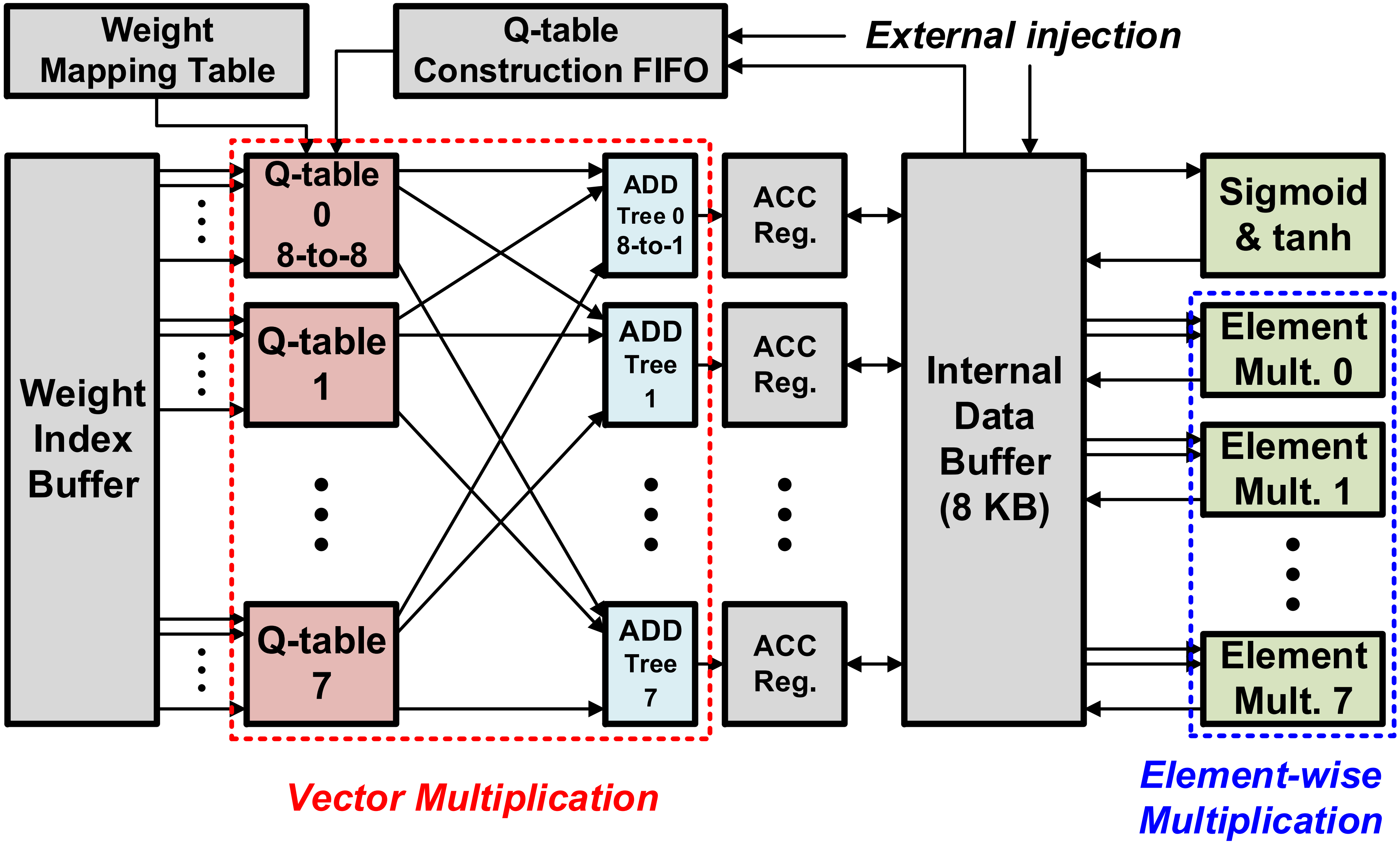
→ No off-chip learning, lower required WL

Performance Comparisons

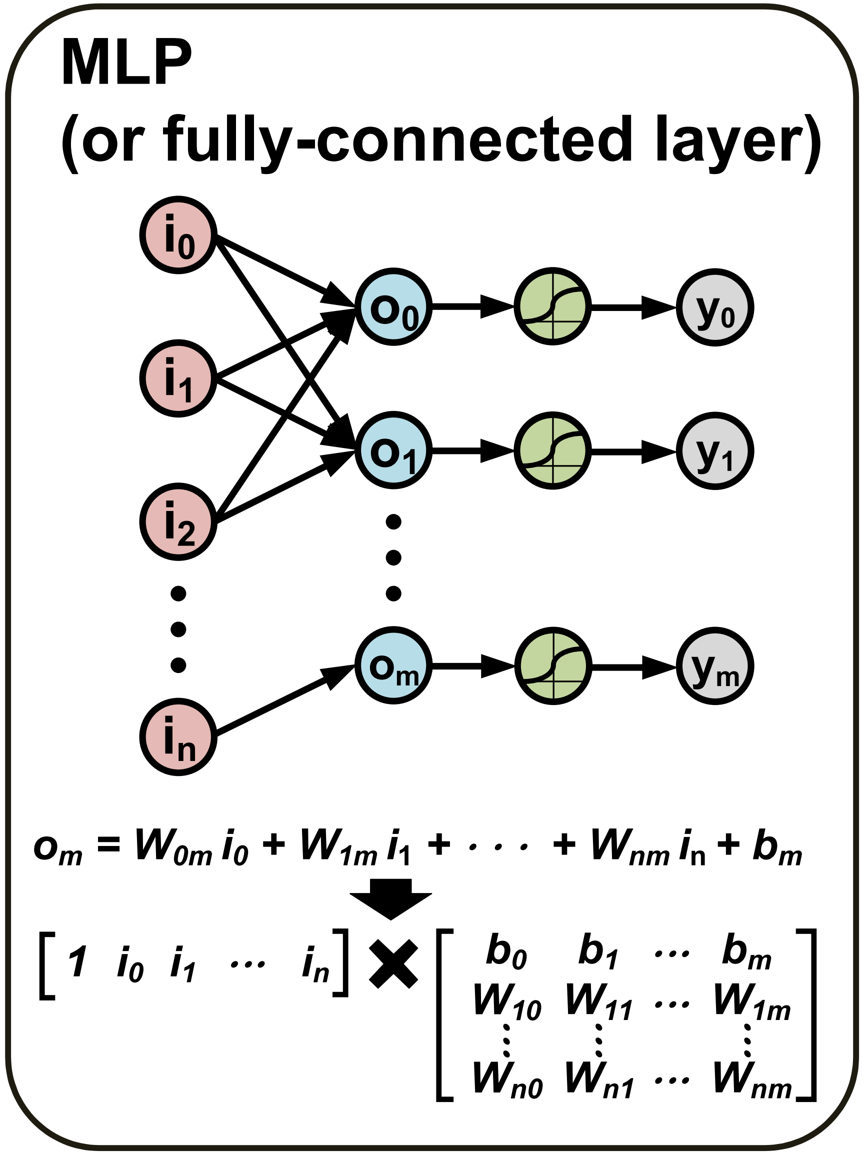
- **Image classification results**

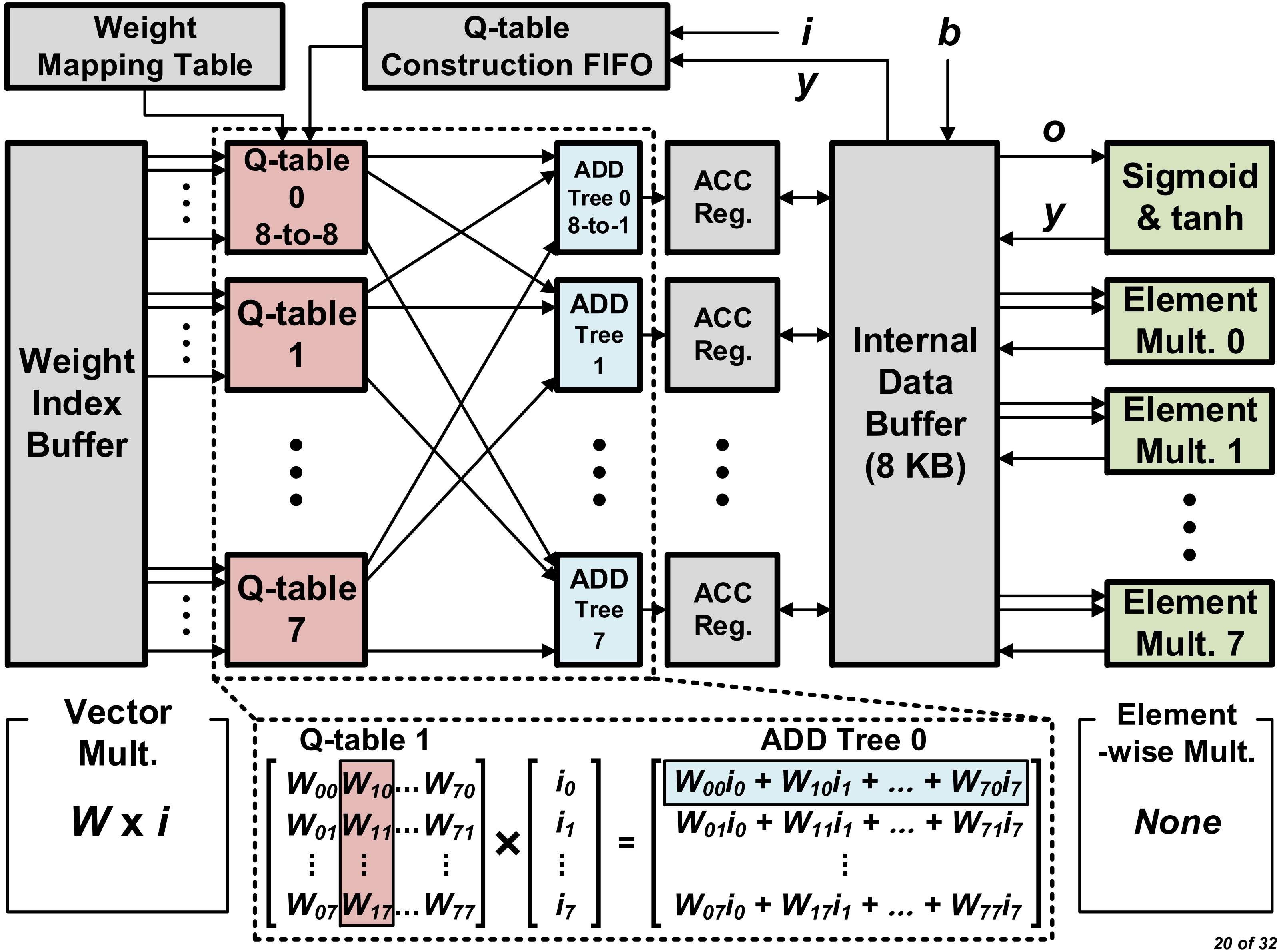


MLP-RNN Processor



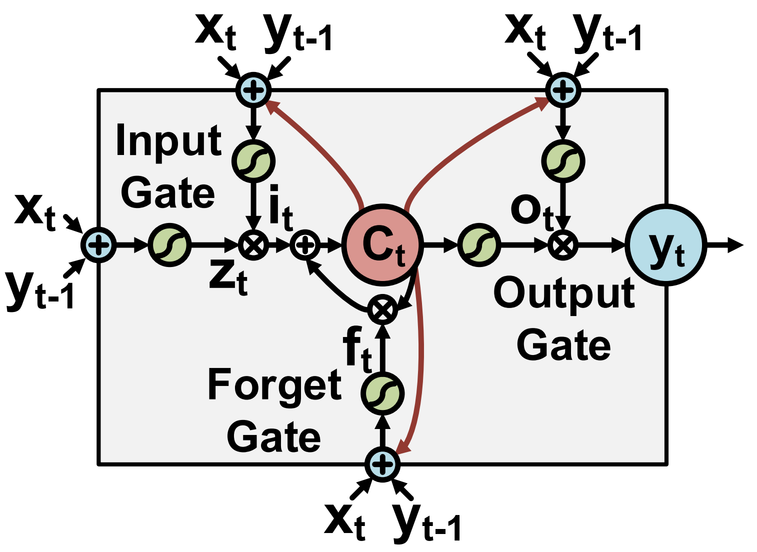
MLP-RNN Processor





MLP-RNN Processor

RNN - LSTM



$$i_t = \sigma(W_{xi}x_t + W_{yi}y_{t-1} + W_{ci}c_{t-1} + b_i)$$

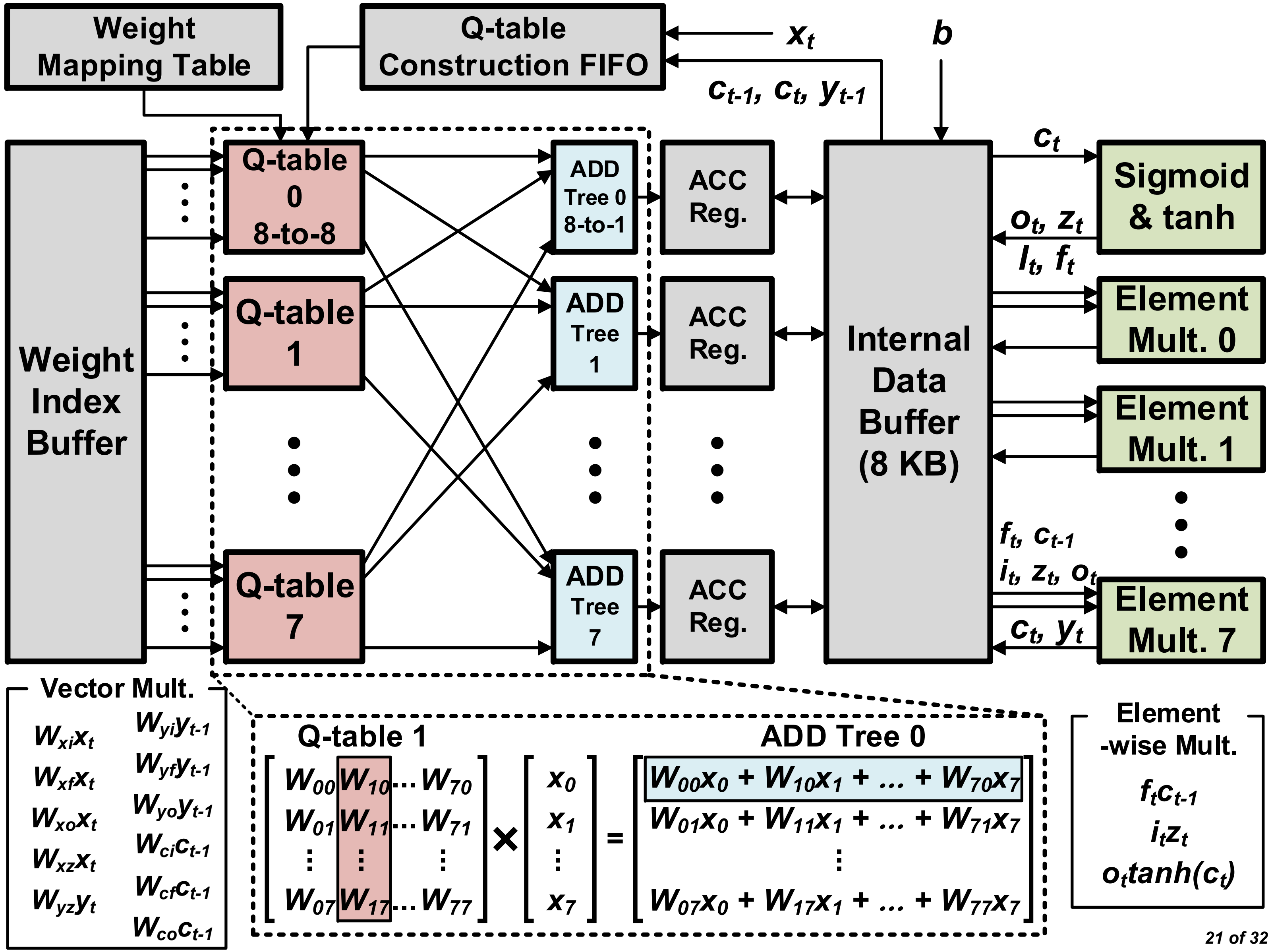
$$f_t = \sigma(W_{xf}x_t + W_{yf}y_{t-1} + W_{cf}c_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{yo}y_{t-1} + W_{co}c_t + b_o)$$

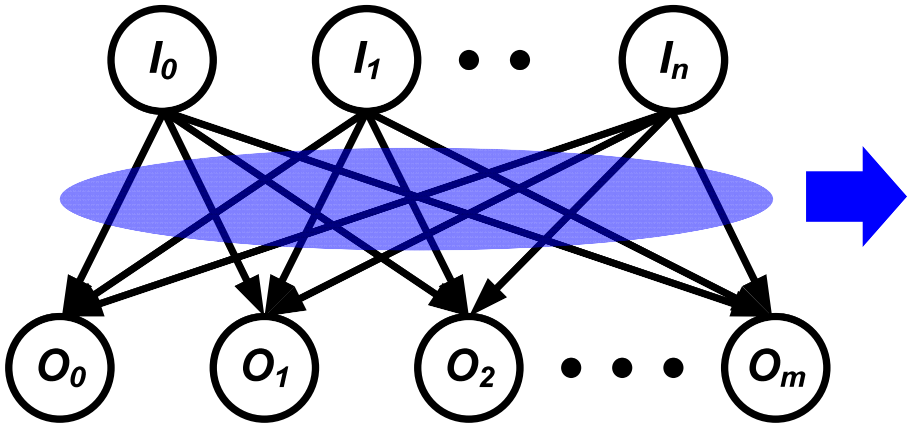
$$z_t = tanh(W_{xz}x_t + W_{yz}y_{t-1} + b_z)$$

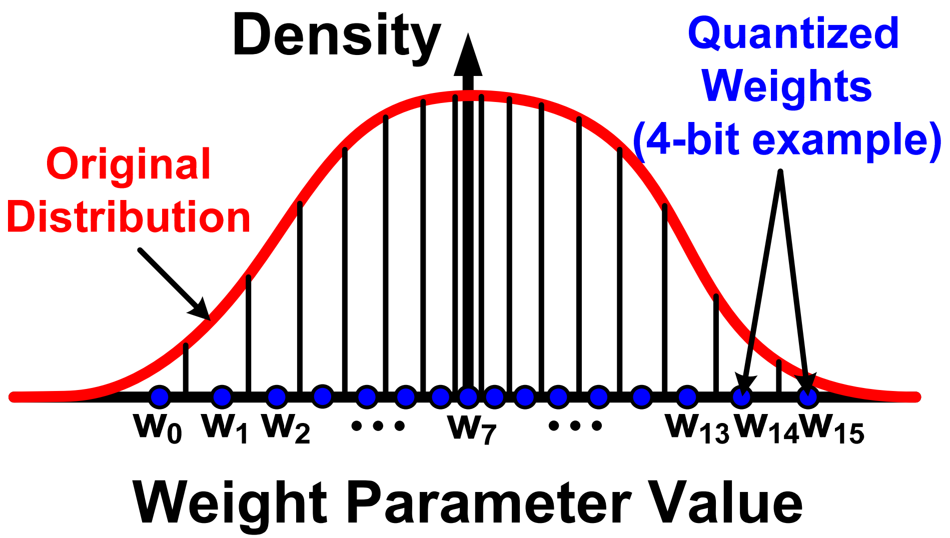
$$c_t = f_t c_{t-1} + i_t z_t$$

$$y_t = o_t tanh(c_t)$$



MLP-RNN Processor: Weight Quantization





FC Layer Weight Quantization

	Top-1 Error	Top-5 Error
32bits	42.78%	19.73%
4bits	42.79%	19.73%
2bits	44.77%	22.33%

ImageNet Classification Test

LSTM Layer Weight Quantization

	Perplexity (Lower is beter)	BLEU (Higher is better)
32bits	15.680	55.7 / 37.4 / 24 / 15.7
4bits	15.829	56.7 / 38 / 24.4 / 15.7
2bits	19.298	58.4 / 38.6 / 24 / 14.8

Flickr 8K Image Captioning Test

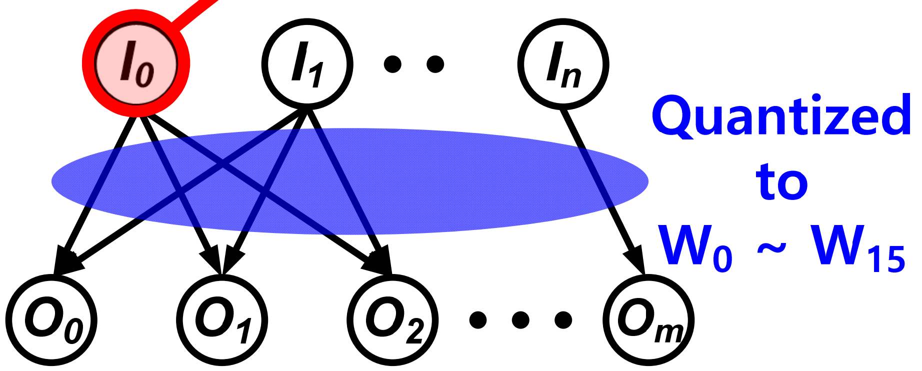
MLP-RNN Processor: Q-table Construction

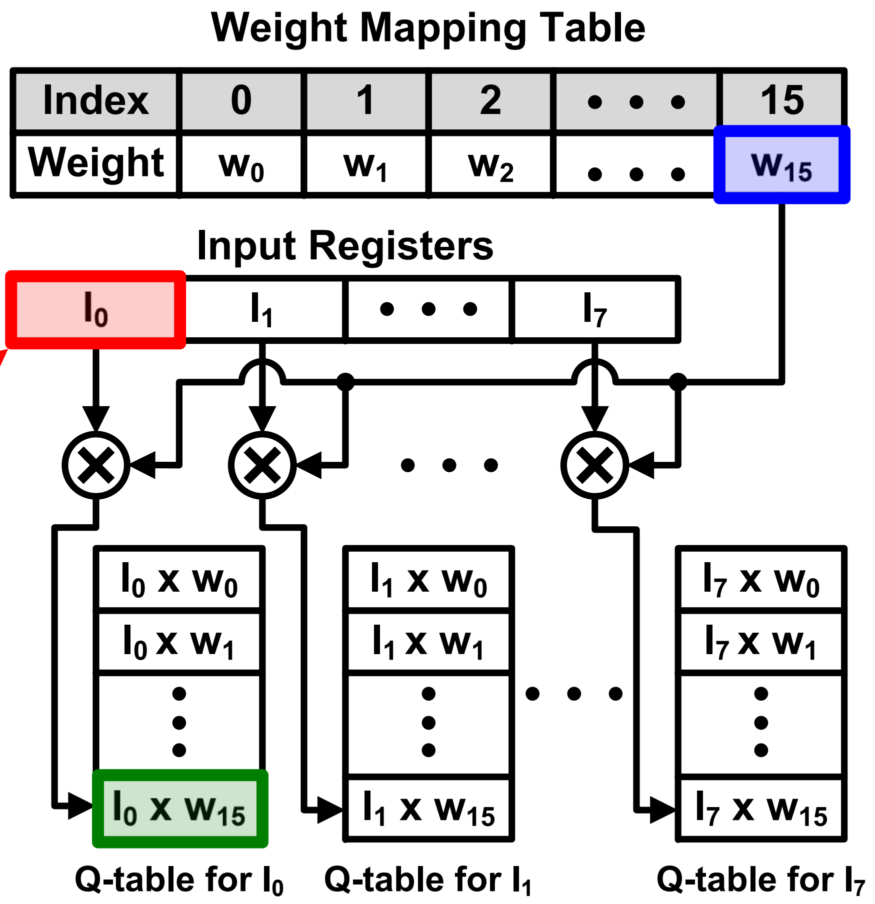
- **Pre-computation with each quantized weight**

Multiplications between
input* and quantized *weights

⬇

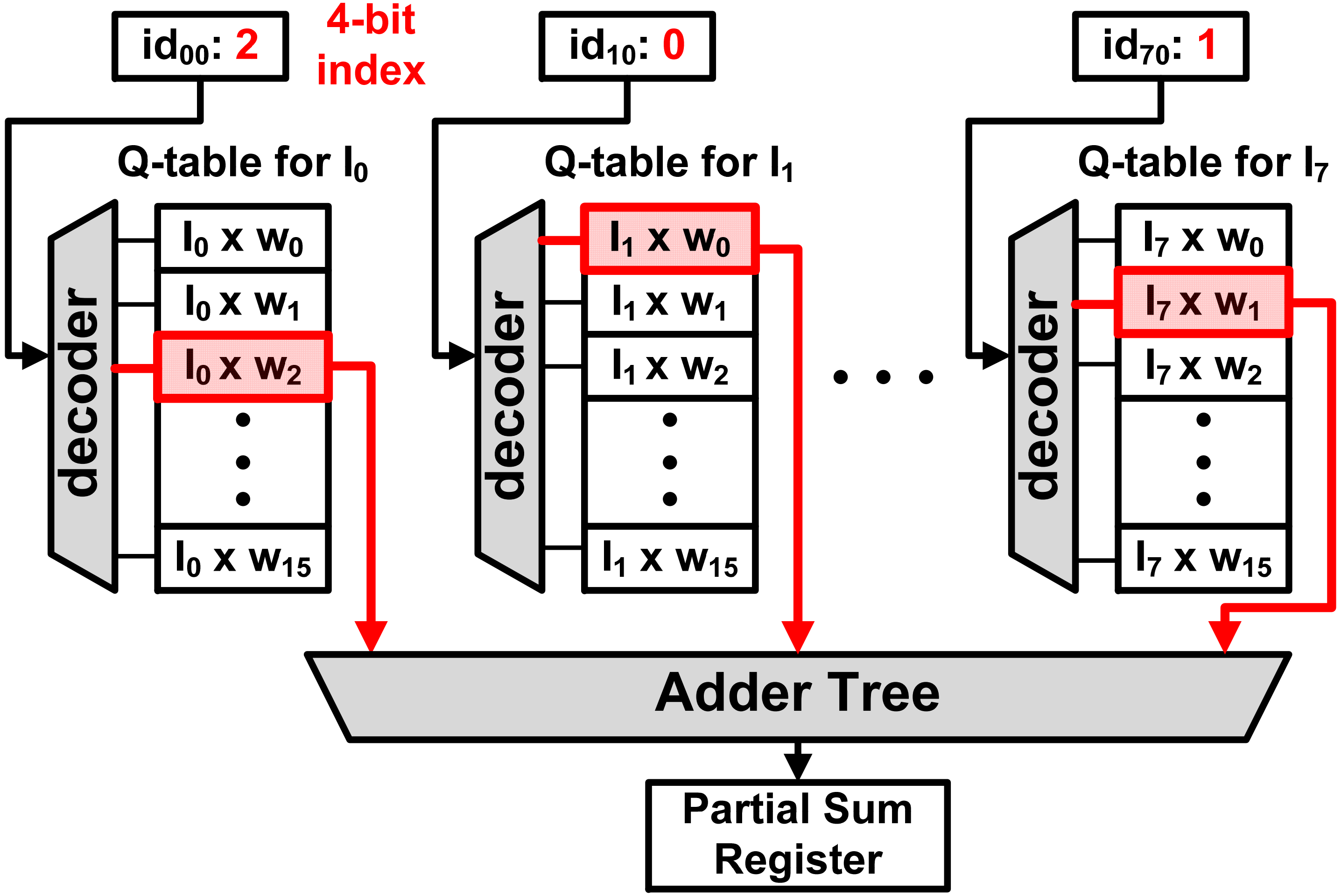
Multiplication results are also quantized to *16 values.*





MLP-RNN Processor: Multiplication with Q-Table

- Decode index to load the pre-computed result



Quantization

: 16-bit weight → 4-bit index

Off-chip Access

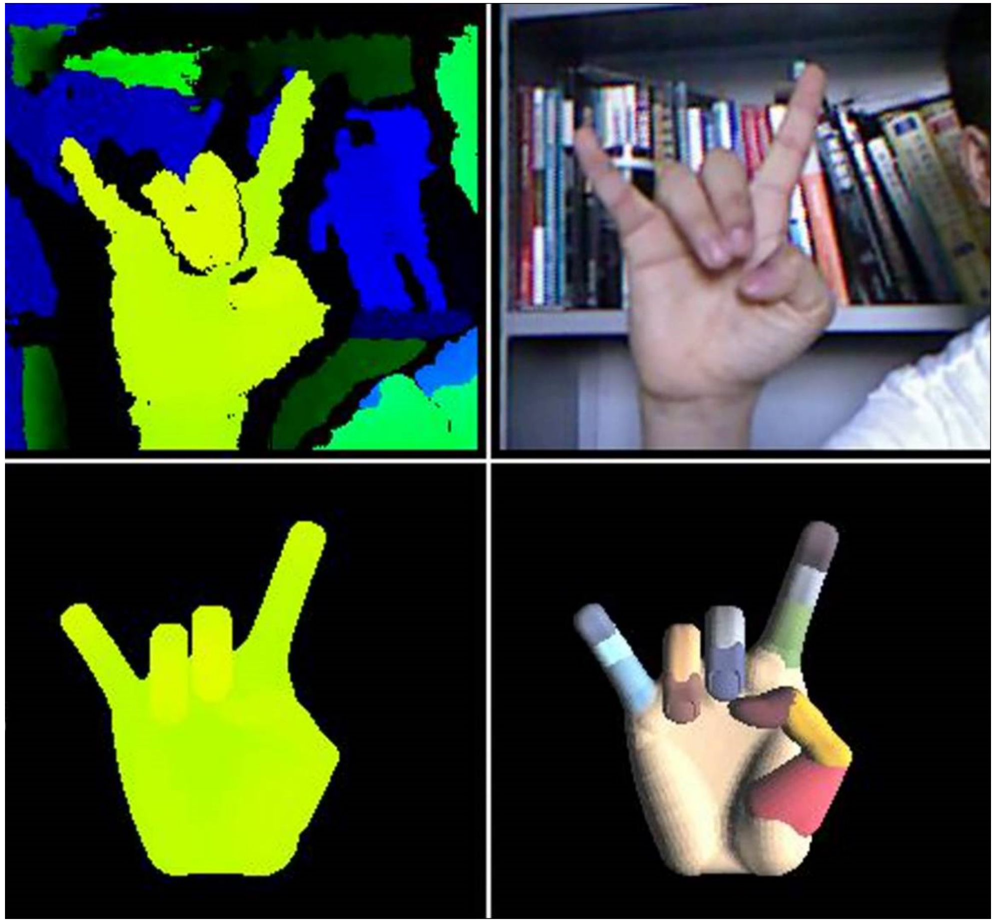
: 75% reduction for weight

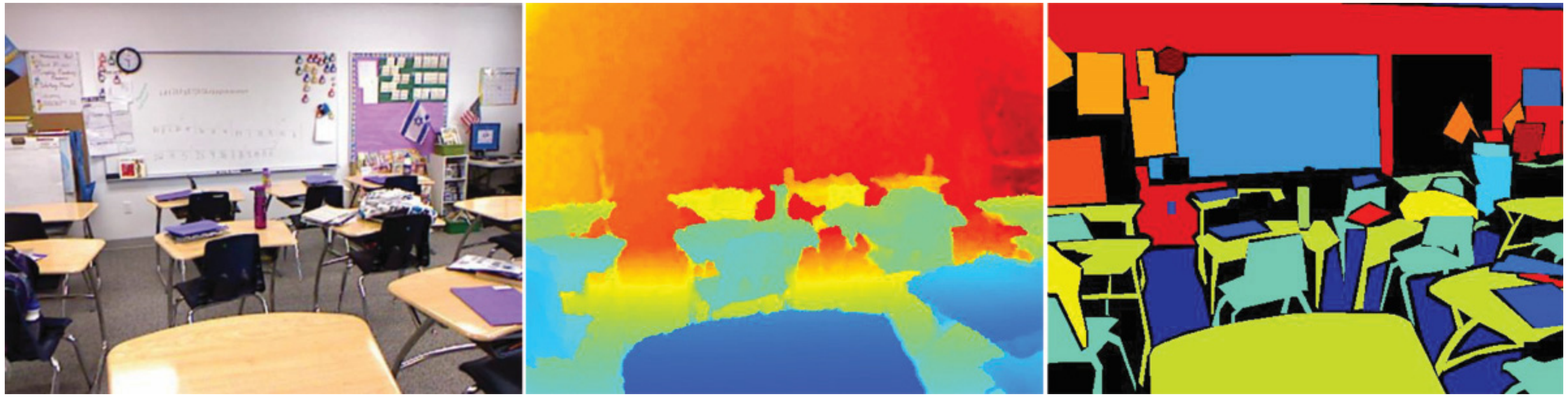
	16-bit Fixed-point	Q-table
Area	1890	1380 (per 1 Mult.)
Power	0.32mW	0.068mW (per 1 Mult.)
Latency (Unconstraied)	7.1ns	0.49ns

Simulation results on 65nm @ 200MHz, 1.2V

Motivation: Why RGB-D?

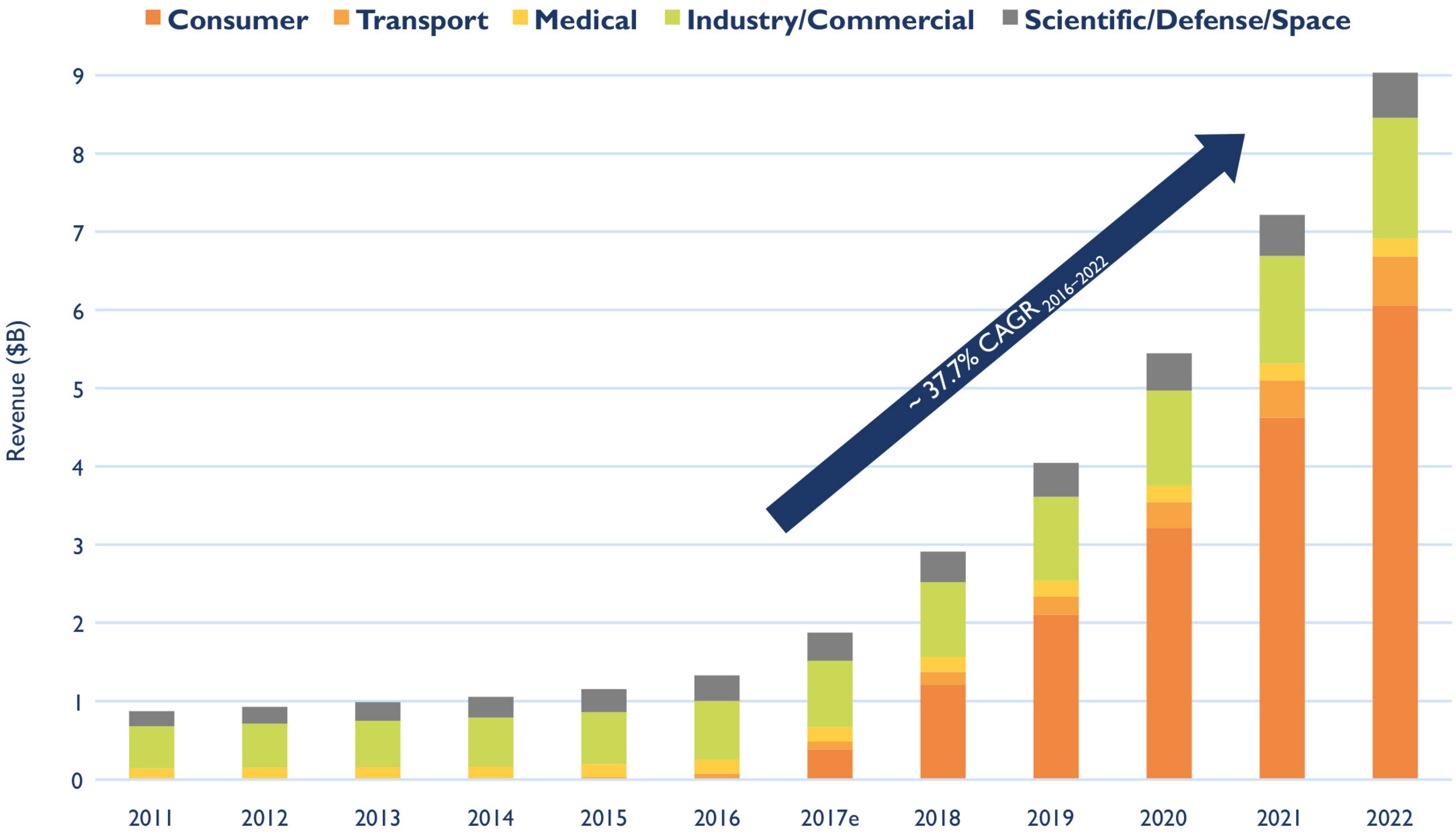
- **Higher Accuracy then RGB**
  - – Object detection
  - – Image segmentation
  - – 3D face recognition
  - – Car detection
  - – Gesture recognition
  - – Visual attention





2011-2022 Market Forecast for 3D Imaging & Sensing Devices

(Source: 3D Imaging & Sensing 2017 report, April 2017, Yole Developpement)



2016
US $1.3 billion

2022
US $9 billion

Big Data and Transfer Learning

- ImageNet DB: 14,197,122 images, 21,841 synsets indexed

- Rich visual features can be trained with ImageNet

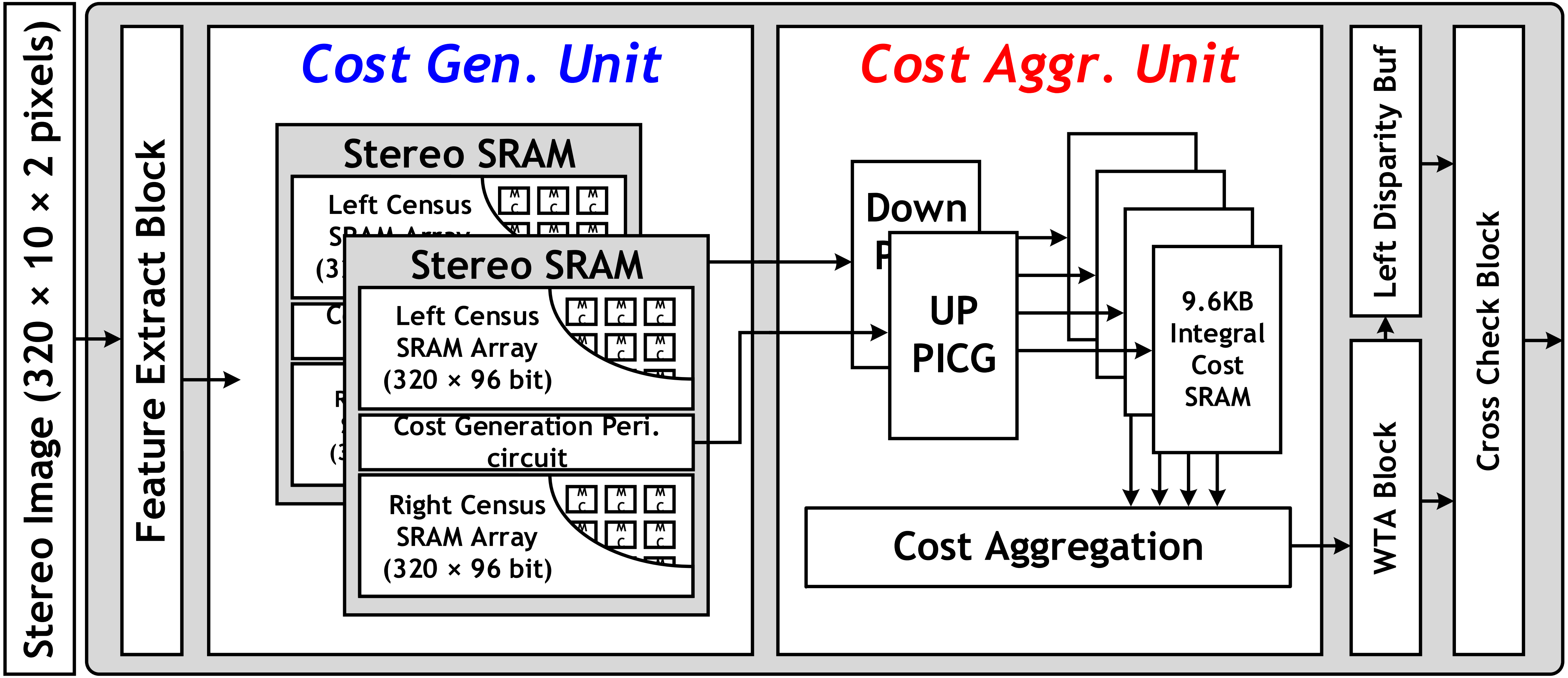
Transfer Learning ⬇

- Enabling learning for other vision tasks with small dataset

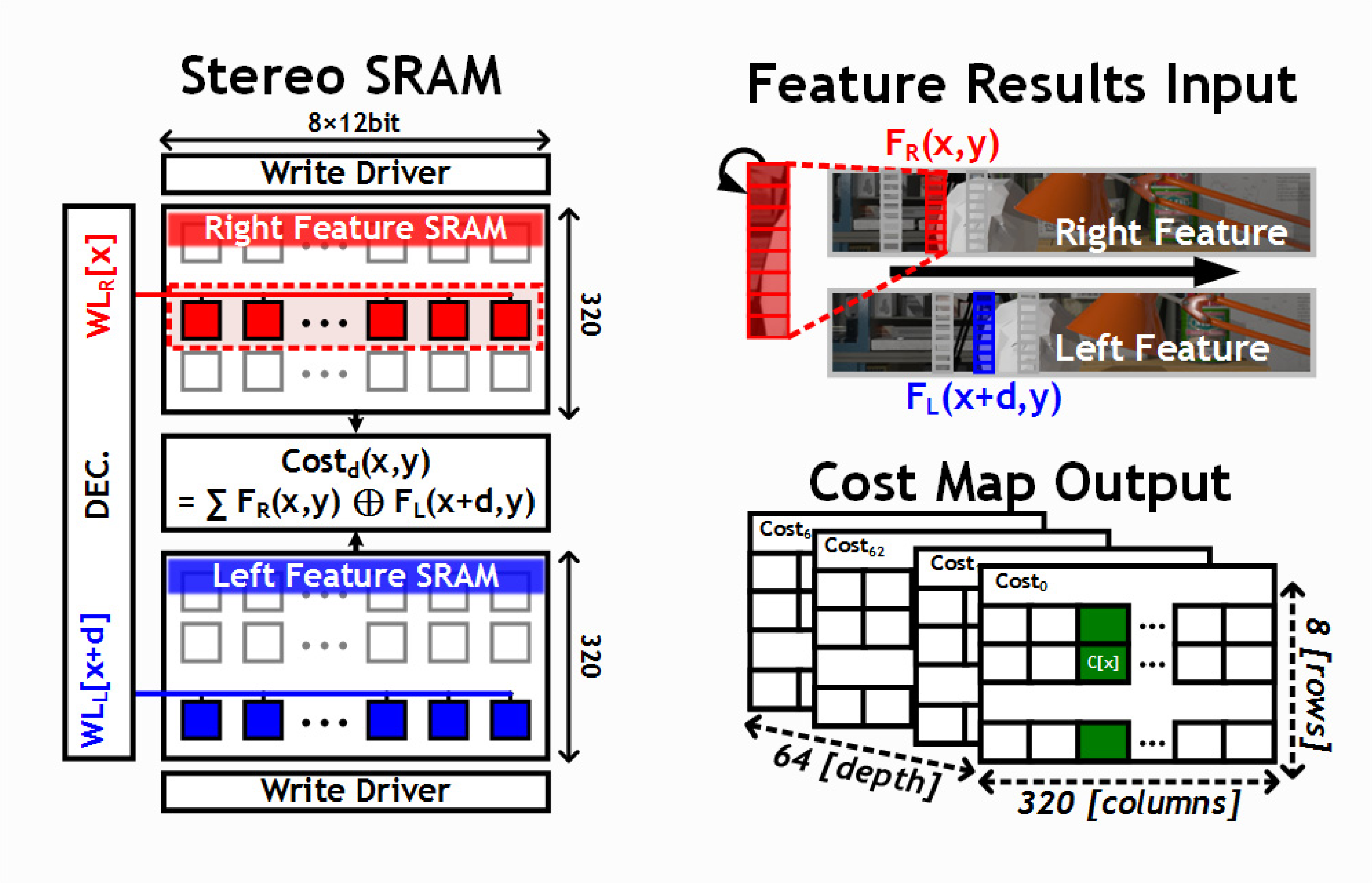
- Consumer 3D imaging devices (Smartphone): Big data
- Transfer learning on RGB-D dataset

→ Amplification on RGB-D DNNs will be coming soon

Stereo Matching Processor



Cost Generation with Computational SRAM



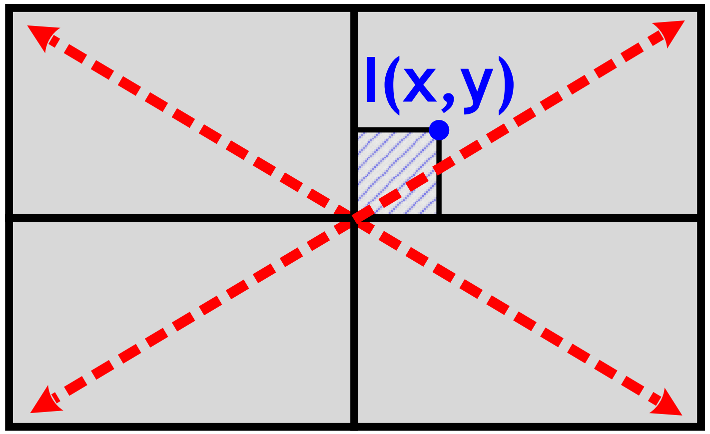
- **Hierarchical Bit-line** is used to Minimize the Bit-line Switching

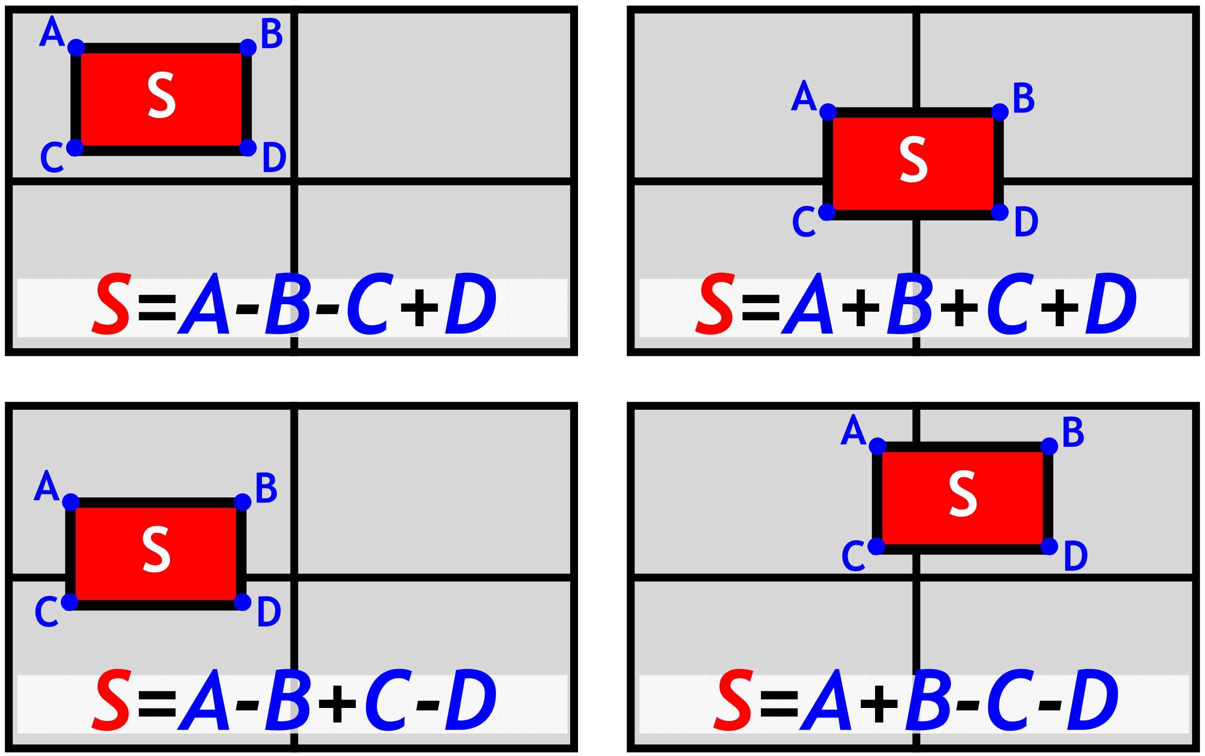
Proposed 4-Square Integral Image

- **4 independent region ➔ 4-way *parallel processing***
- **¼ Maximum range ➔ data bit width *2-bit reduction***
- ***No additional computation & Mem access cost***

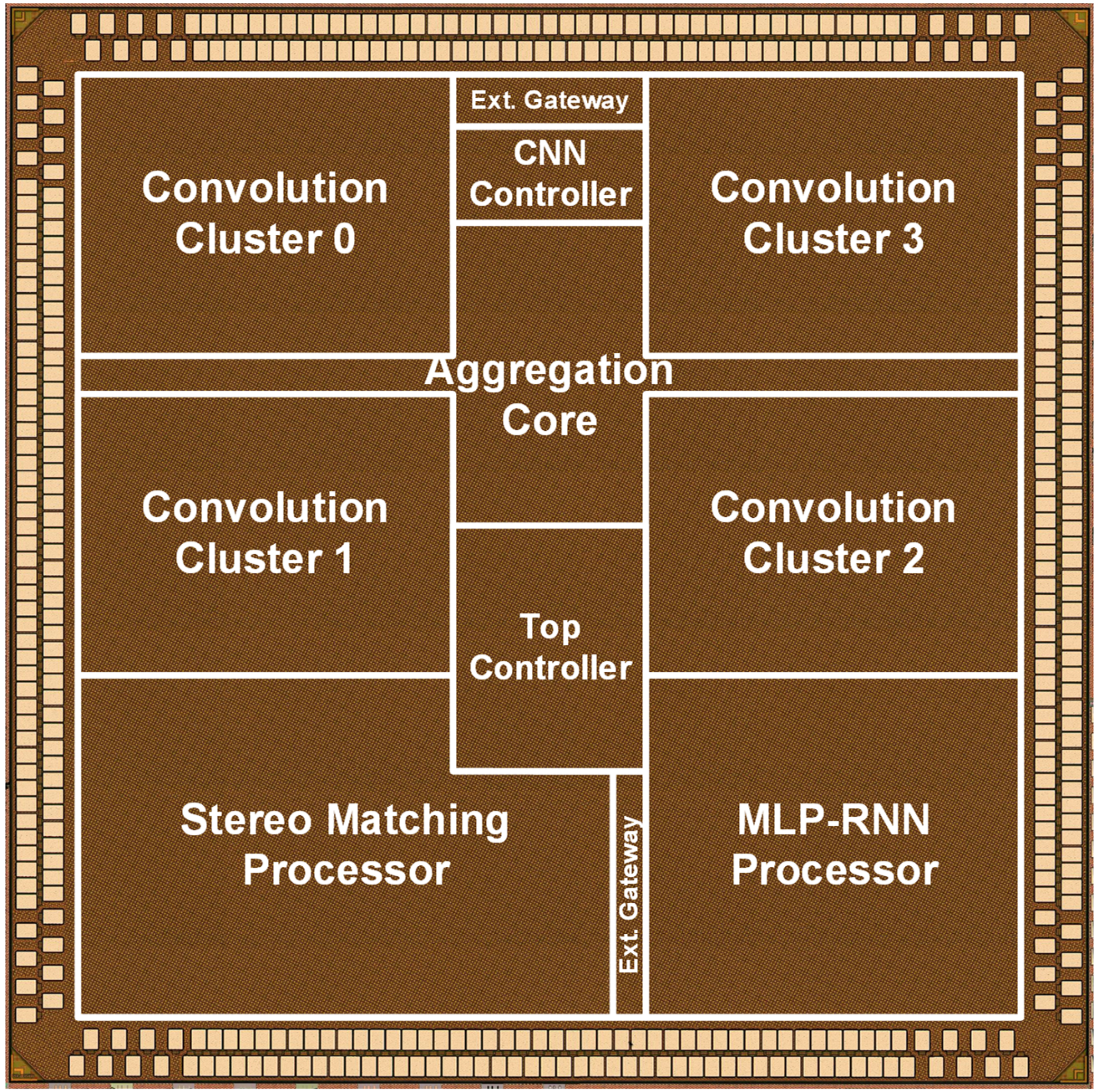
Aggregation

4-way Integral Image Generation





Chip Photograph and Summary



	Specifications				
Technology	65nm 1P8M CMOS				
Die Area	4mm x 4mm (16mm^2)				
SRAM Size	Convolution		MLP-RNN		
	280 KB		10 KB		
Supply Voltage	0.77V ~ 1.1V				
Operating Frequency	Conv.	MLP-RNN	NoC	ETC	
	~200MHz	~200MHz	~400MHz	~200MHz	
Power Consumption		Conv.	MLR-RNN	ETC	Total
	50MHz @ 0.77V	29mW	2.6mW	3mW	35mW
	200MHz @ 1.1V	235mW	21mW	23mW	279mW
Energy Efficiency		Word Length: 16-bit	Word Length: 4-bit		
	50MHz @ 0.77V	2.1 TOPS/W	8.1 TOPS/W		
	200MHz @ 1.1V	1.0 TOPS/W	3.9 TOPS/W		

Conclusion

- **DNPU**: Enabling **embedded DNNs** in mobile platforms

- **Heterogeneous Architecture**
  - – Convolution Processor
  - – MLP-RNN Processor

- Stereo Matching Processor
- **RGB-D 4-ch** Operation

- Next version DNPU for **large input image** & **large kernel**