Quantization

Chia-Chi Tsai (蔡家齊)
cctsai@gs.ncku.edu.tw
AI System Lab
Department of Electrical Engineering
National Cheng Kung University

Outline

- Overview
- Basics of DNN Quantization
- Quantization Range Clipping
- Advanced Quantization
- Hardware Design

Outline

- Overview
- Basics of DNN Quantization
- Quantization Range Clipping
- Advanced Quantization
- Hardware Design

Approaches

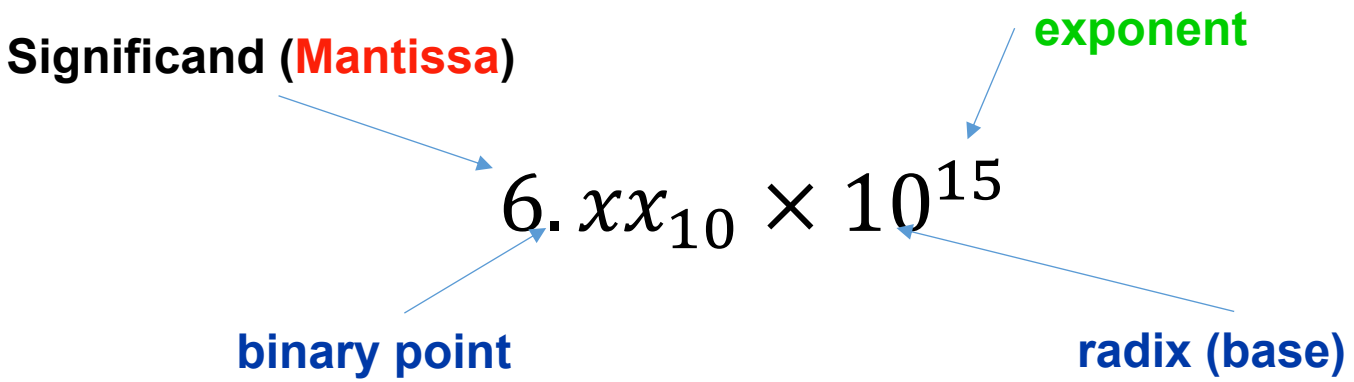
- Reduce size of operands for storage/compute
  - Floating point → Fixed point
  - Bit-width reduction
  - Non-linear quantization

- Reduce number of operations for storage/compute
  - Exploit Activation Statistics (Compression)
  - Network Pruning
  - Compact Network Architectures

Floating-Point(FP) Motivation

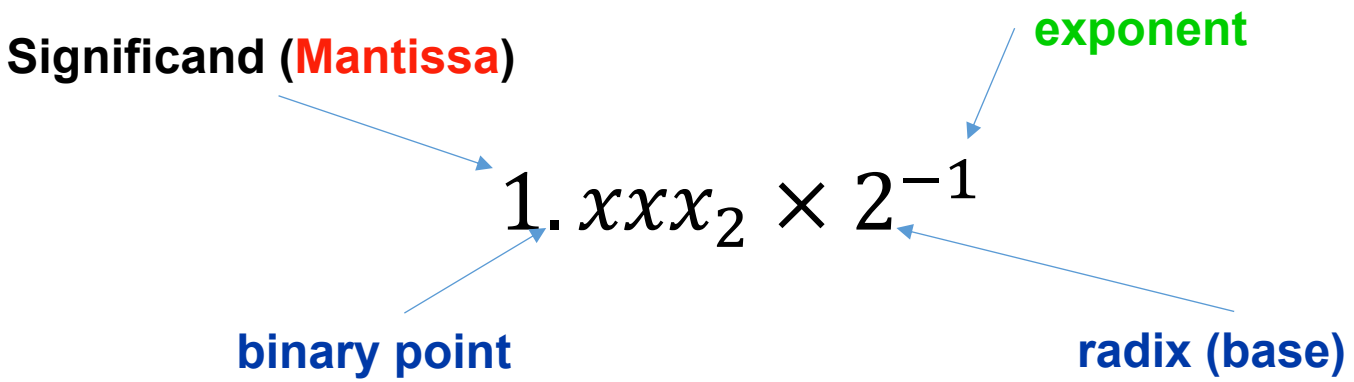
- What can be represented in n bits?
  - Unsigned 0 to 2^n - 1
  - 2's Complement -2^{n-1} to 2^{n-1} - 1
  - 1's Complement -2n-1+1 to 2^{n-1}
  - Excess M -M to 2^n - M - 1
- But, what about ...
  - Very large numbers?
    - 9,349,398,989,787,762,244,859,087,678
  - Very small number?
    - 0.00000000000000000000045691
  - Rationales 2/3
  - Irrationals $\sqrt{2}$
  - Transcendentals e, π

Scientific Notation: Decimal



- Normalized form: no leadings 0s (exactly one digit to left of decimal point)
- Alternatives to representing 1/1,000,000,000
  - Normalized: 1.0×10^{-9}
  - Not normalized: 0.1×10^{-8}, 10.0×10^{-10}

Scientific Notation: Binary



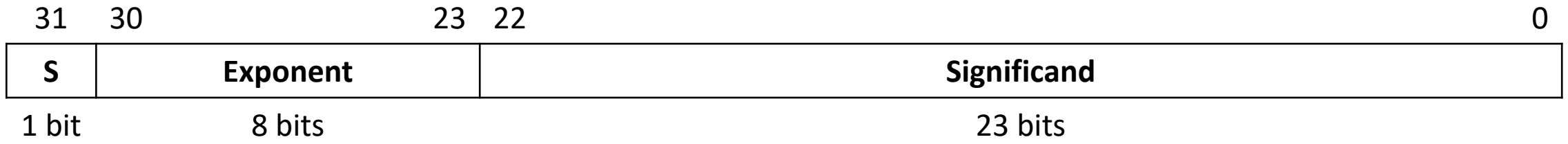
- Computer arithmetic that supports it is called **floating point**, because the binary point is not fixed, as it is for integers

Floating Point

- Representation for non-integral numbers
  - Including very small and very large numbers
- Like scientific notation
  - -2.34×10^{56} - normalized
  - 0.0002×10^{-4} - not normalized
  - 987.02×10^{9} - not normalized
- In binary
  - $\pm 1.xxxxxxx_2 \times 2^{yyyy}$
- Type float and double in C

FP Representation

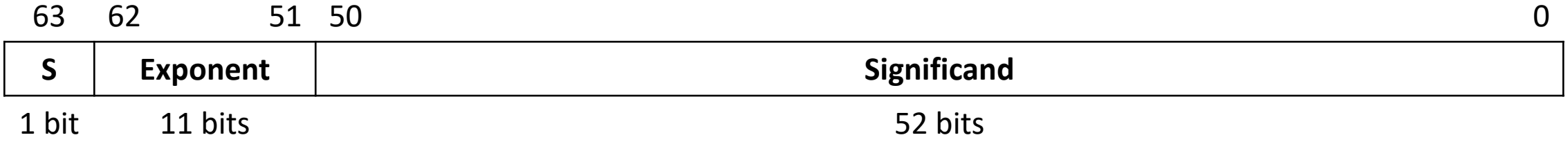
- Normal format: $1.xxxxxxx_2 \times 2^{yyyy_2}$
- Want to put it into multiple words: 32 bits for single-precision and 64 bits for double-precision
- A simple single-precision representation:



- **S** represents sign
- **Exponent** represents y's
- **Significand** represents x's
  - Represent numbers as small as $2.0\ x\ 10^{-38}$ to as large as $2.0\ x\ 10^{38}$

Double Precision Representation

- 64 bits Format



- Double precision (vs. single precision)
  - Represent numbers almost as small as $2.0 \, x \, 10^{-308}$ to almost as large as $2.0 \, x \, 10^{308}$
  - But primary advantage is greater accuracy due to larger significand

Floating Point Standard

- Defined by IEEE Std 754-1985
- Developed in response to divergence of representation
  - Portability issues for scientific code
- Now almost universally adopted
  - Two representations
  - Single precision (32-bit)
  - Double precision (64-bit)

IEEE 754 Standard (1/2)

- Regarding single precision (SP), DP is similar
- Sign bit
  - 1 means negative
  - 0 means positive
- Significand
  - To pack more bits, **leading 1** implicit for normalized numbers
  - 1 + 23 bits single, 1 + 52 bits double
  - always true: 0 ≤ Significand < 1 (for normalized numbers)
- Note
  - **0 has no leading 1, so reserve exponent value 0 just for number 0**

IEEE 754 Standard (2/2)

- Exponent
  - Need to represent positive and negative exponents
  - Also want to compare FP numbers as if they were integers, to help in value comparisons
  - If use 2's complement to represent?
    - e.g., 1.0×2^{-1} versus $1.0 \times 2^{+1}$($\frac{1}{2}$ versus 2)

1/2	**0**	**1111 1111**	**000 0000 0000 0000 0000 0000**
2	**0**	**0000 0001**	**000 0000 0000 0000 0000 0000**

If we use integer comparison for these two words, we will conclude that 1/2 > 2!!!

Biased (Excess) Notation

- Let notation 0000 be most negative, and 1111 be most positive
- Example: Biased 7

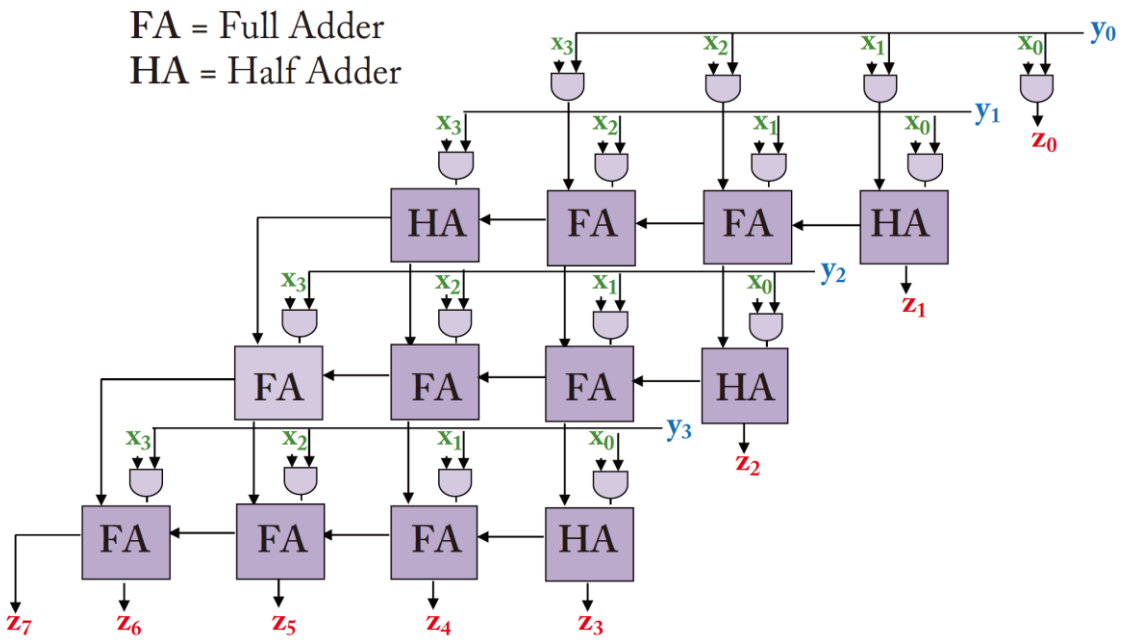
Binary	Decimal	Binary	Decimal
0000	-7	1000	1
0001	-6	1001	2
0010	-5	1010	3
0011	-4	1011	4
0100	-3	1100	5
0101	-2	1101	6
0110	-1	1110	7
0111	0	1111	8

IEEE 754 Standard

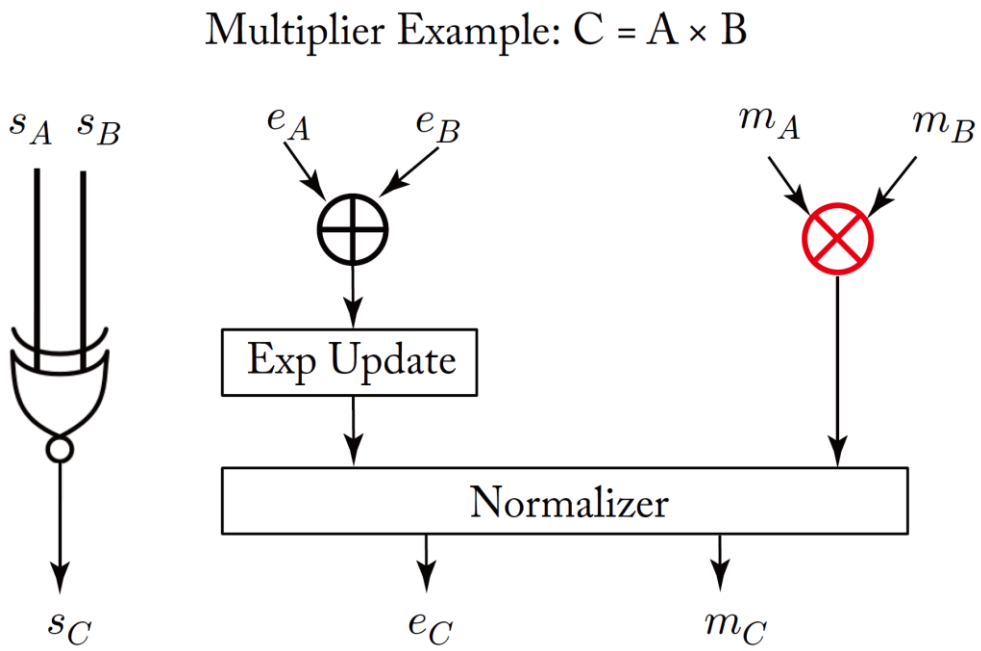
- Using biased notation
  - The bias is the number subtracted to get the real number
  - IEEE 754 uses bias of 127 for single precision
    - Subtract 127 from Exponent field to get actual value for exponent
  - 1023 is bias for double precision
  - The example becomes ….

1/2	**0**	**0111 1110**	**000 0000 0000 0000 0000 0000**
2	**0**	**1000 0000**	**000 0000 0000 0000 0000 0000**

Hardware Implications



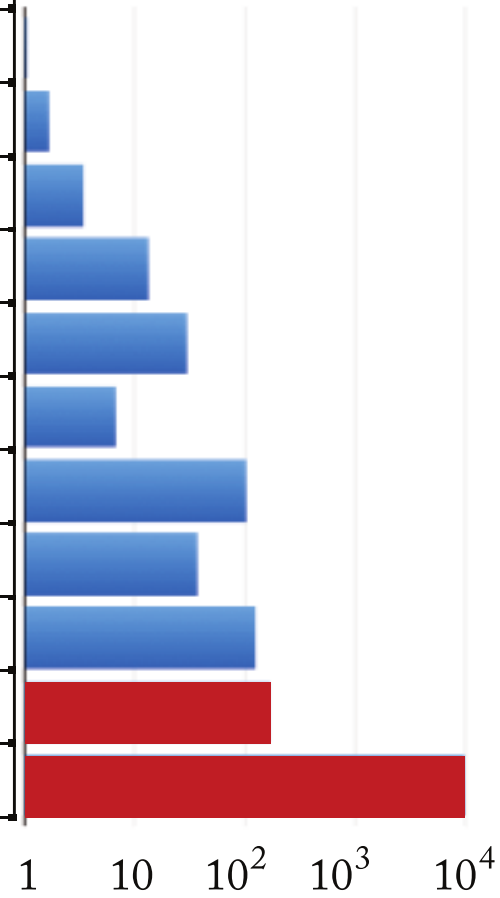
Fixed-Point Multiplier

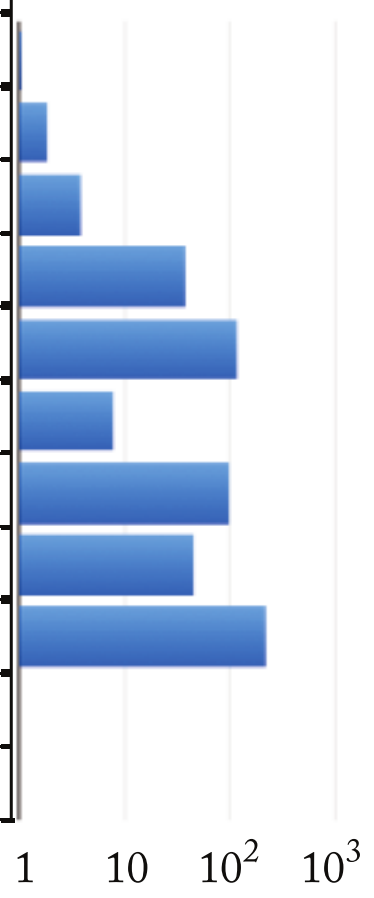


Floating-Point Multiplier

Cost of Operations

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



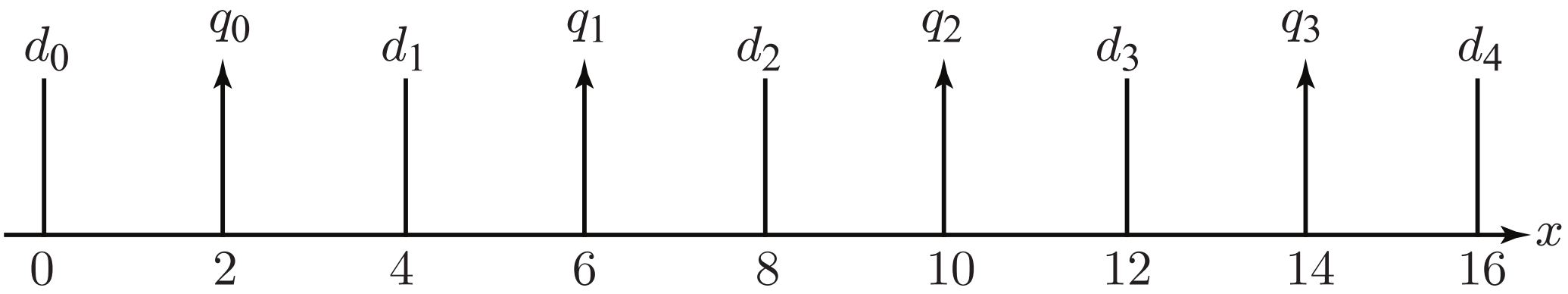


Rough Energy Numbers (45nm)" from "computing's Energy Problem, M. Horowitz, ISSCC, 2014

FP Add and Multiply

- Add two floating point numbers
  1. Find max exponent
  2. Align mantissas based on max exponent, followed by mantissa addition
- Multiplying two FP numbers
  1. Mantissa integer multiplication (quadratic cost in mantissa bits)
  2. Exponent integer add (Linear in exponent bits)
- FP arithmetic is expensive mainly due to
  - Mantissa alignment
    - FP “normalization” before and after each add
  - Mantissa multiplications
    - quadratic cost

Quantization

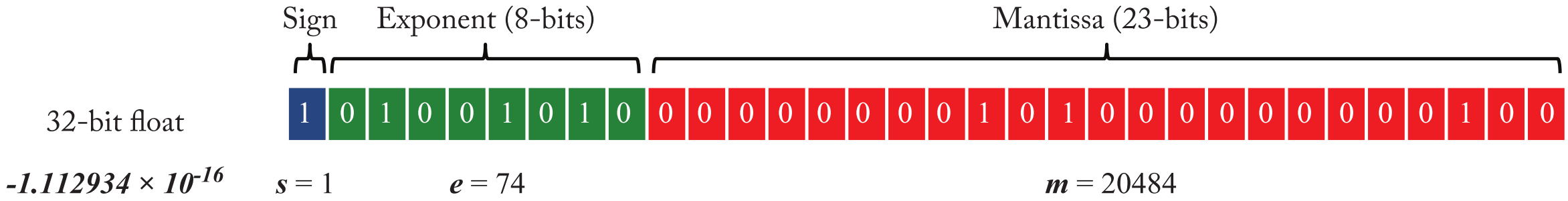


Example of values quantized using $Q(\cdot)$

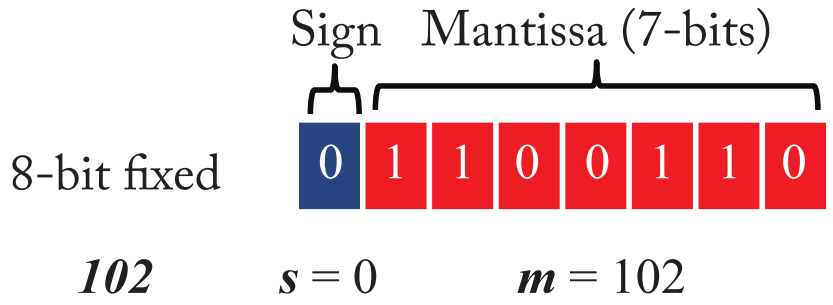
$x = 1, 3, 7, 8, 9, 15, 6, 2 \rightarrow$ Quantization $Q(\cdot)$ $\rightarrow x = 2, 2, 6, 8, 10, 14, 6, 2$

Floating Point to Fixed Point

- Floating Point

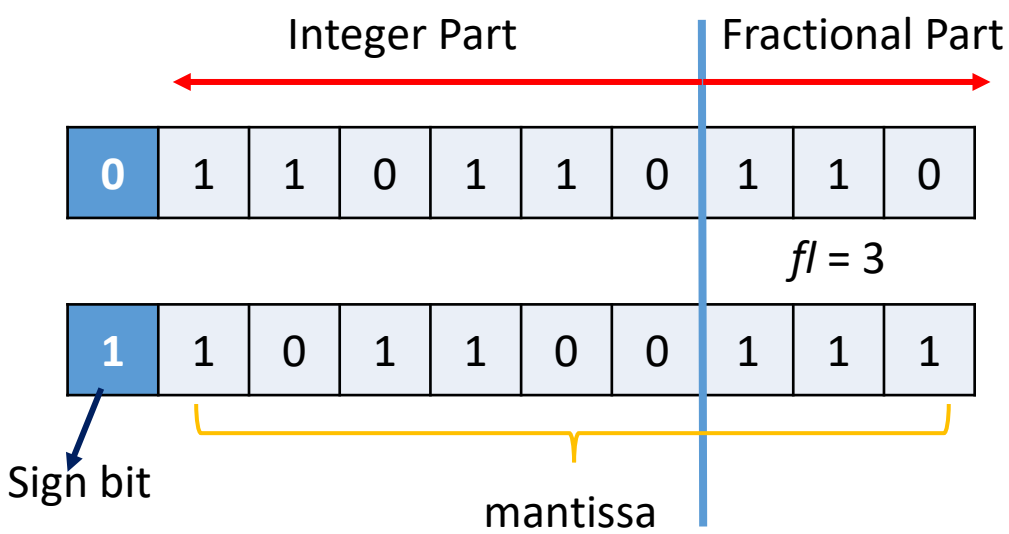


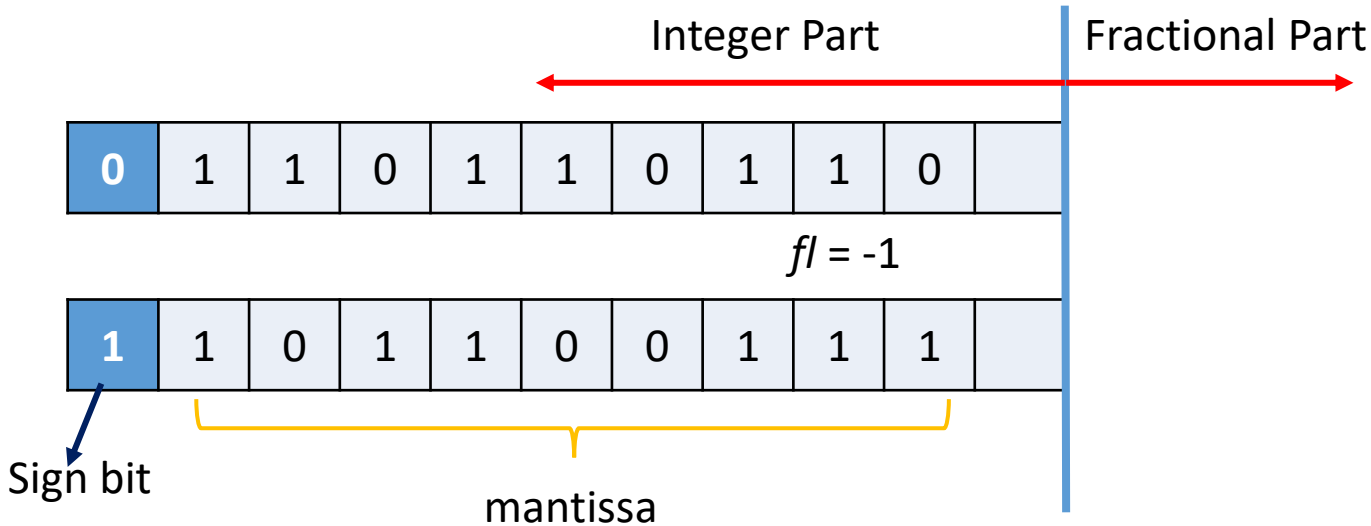
- Fixed-Point



Dynamic Fixed-Point

- Allow ***f*** to vary based on data type and layer
- Hardware friendly
  - Only requires shift and ordinary 2'complement mathematical operation



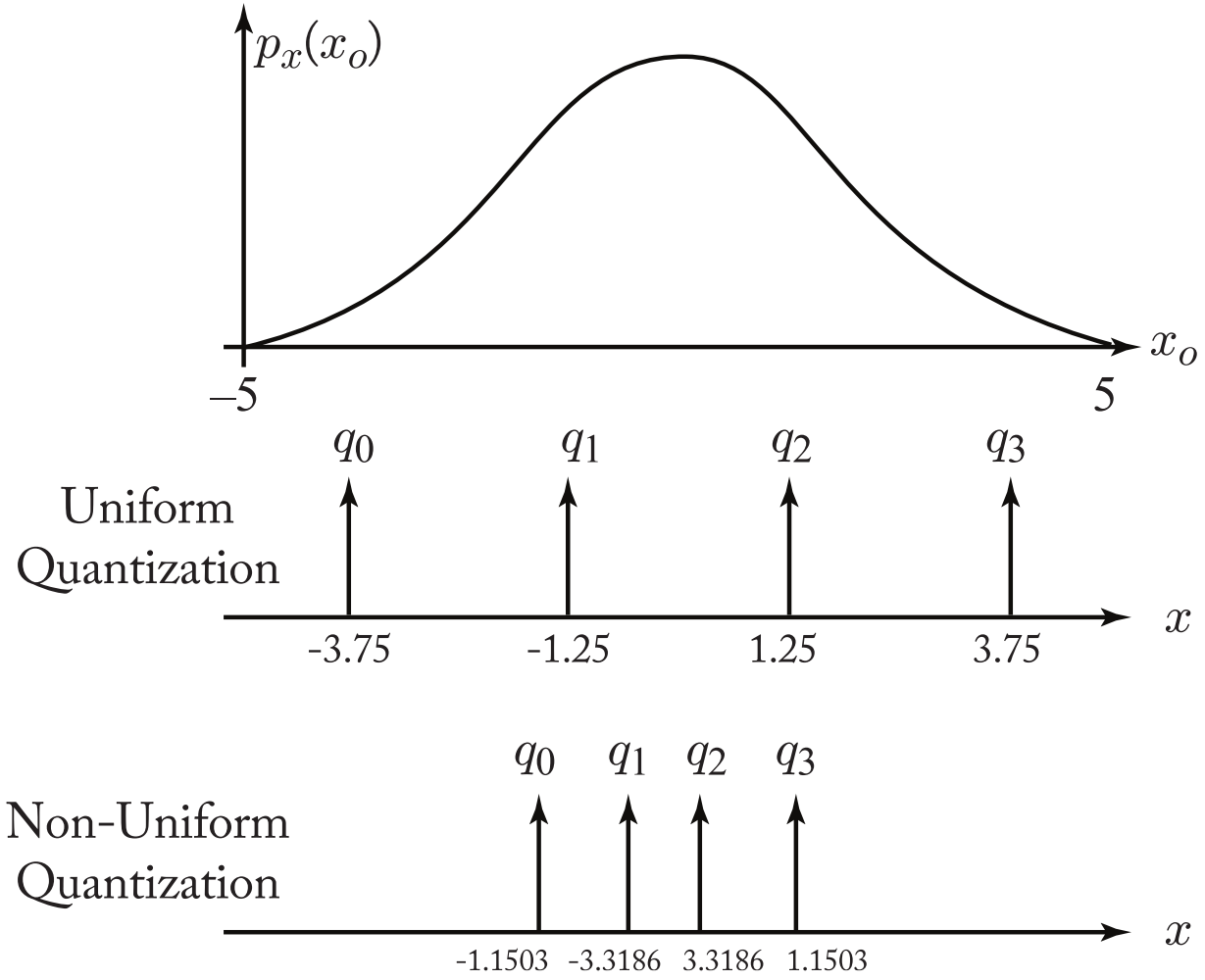


Quantization Taxonomy

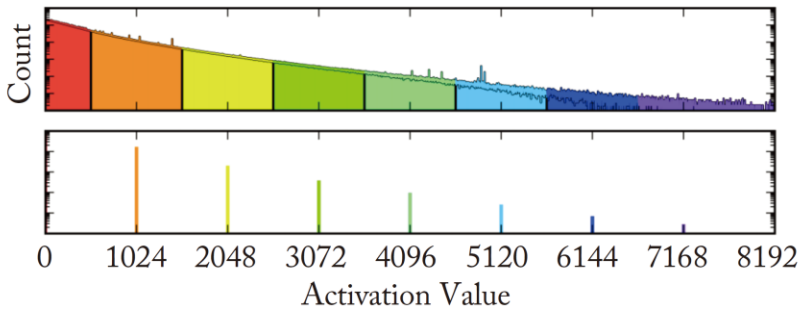
- **Precision** refers to the **number of levels**
  - Number of bits = $\log_2$ (number of levels)
- **Quantization**: mapping data to a smaller set of **levels**
  - Linear, e.g., fixed-point
  - Non-linear
    - Computed (e.g., floating point, log-domain)
    - Table lookup (e.g., learned)

Objective: Reduce size to improve speed and/or reduce energy while preserving accuracy

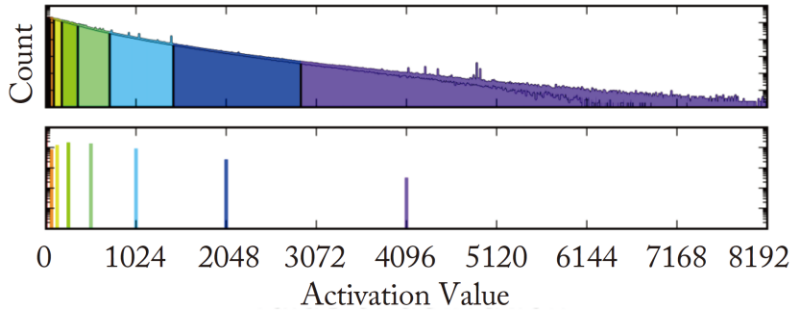
Uniform/Non-uniform Quantization



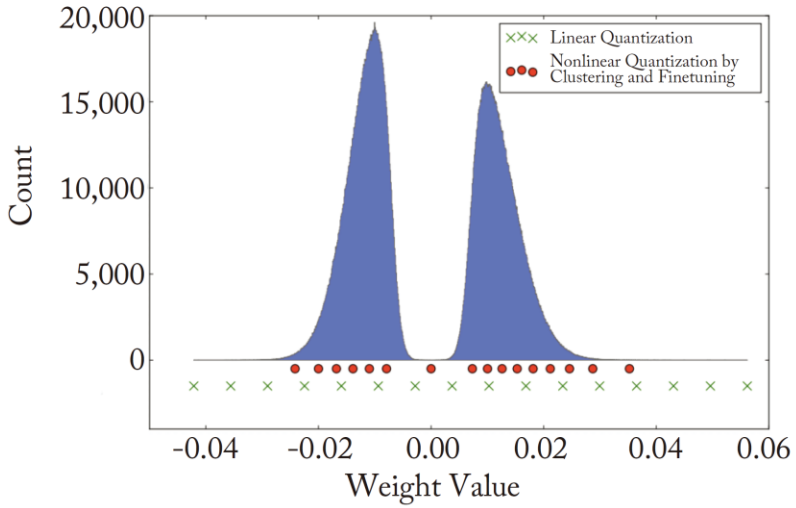
Various methods of Quantization



(a) Uniform quantization



(b) Log quantization



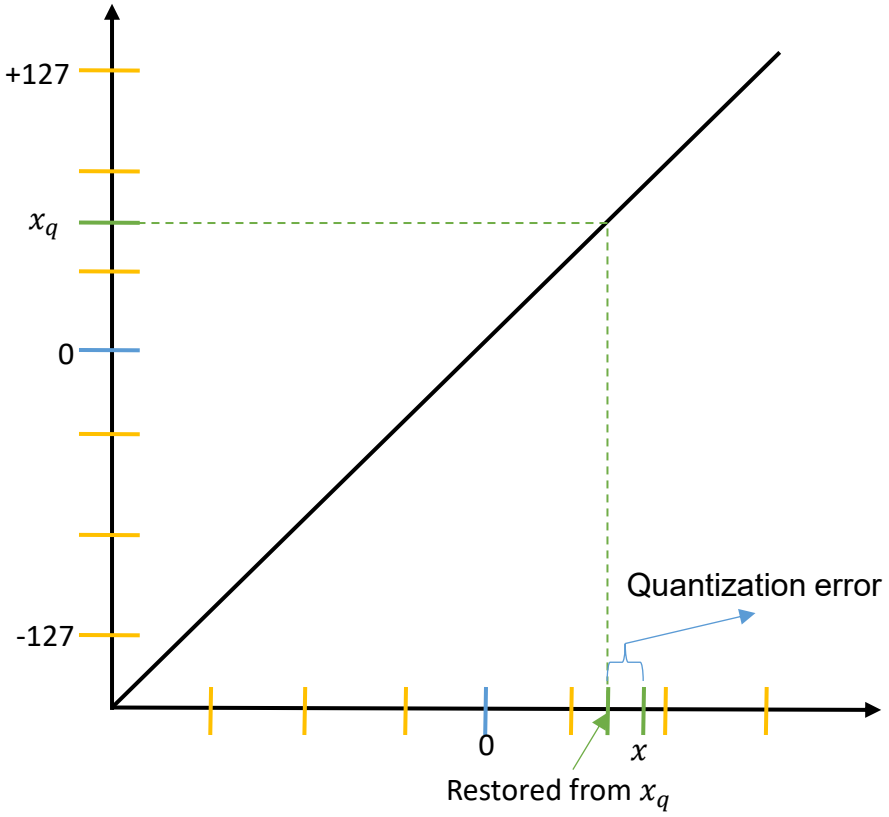
(c) Learned quantization

Outline

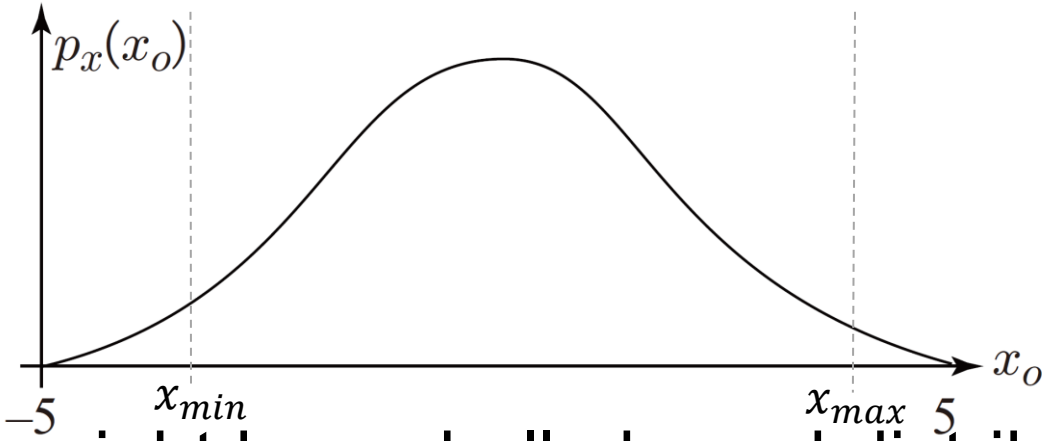
- Overview
- Basics of DNN Quantization
- Quantization Range Clipping
- Advanced Quantization
- Hardware Design

Uniform Quantization

- Consider quantizing FP32 x to INT8 x_q (Bitwidth = 8)
- Quantize a DNN network layer by layer
  1. Collecting the dynamic range of FP32 number for each layer
  2. Partition the range into uniform intervals (**uniform quantization**)
  3. The quantization function maps rounded FP32 number in these intervals onto integer, in[-127, +127], according to a linear mapping.
  - **Uniform quantization** is also called **linear quantization**
  - These intervals are called **quantization interval**
  - **Quantization error** is bounded by interval size



Clamping



- Value in activation and weight has a bell-shaped distribution
- The clamp function is defined as

$$clamp(x, -L, L) = \begin{cases} x, & if\ x\ is\ in\ [-L, L] \\ -L, & if\ x < L \\ L, & if\ x > L \end{cases}$$

- E.g. clamp(-1.5,-1,1)=1, clamp(0.5,-1,1)=0.5, clamp(-1.5,-1,1=-1)

Scaling and Clamping

- Quantizing x within [-L, L] using b bits (e.g. b=8)
  1. $sf = \frac{L}{2^{b-1}-1}$
  2. $x_c = clamp(x, -L, L)$
  3. $x_q = \left\lfloor \frac{x_c}{sf} + 0.5 \right\rfloor \times sf$

L: Boundary of quantization interval
sf: scaling factor
b: bit-width
X: floating-point value
x_c: clamped value of x
x_q: Quantized x

Quantization Error

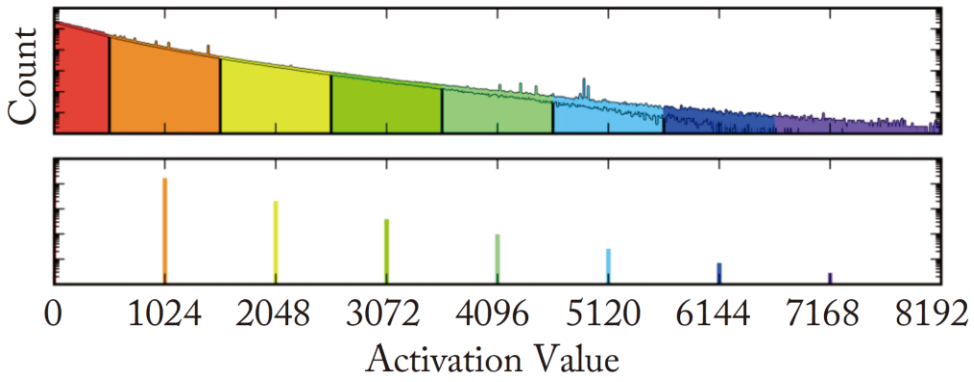
- As a **proxy** to model accuracy, we usually study **quantization error**.
- However, we note no direct relationship can easily be established between quantization error and final model accuracy, as stated in the PWLQ paper.
- Quantization errors at x need to be **weighted** by the probability of its occurrence.
- Given $x_{min} \leq x \leq x_{max}$, minimize the quantization error can be determined by solving

$$\min_{r_i, d_i} E[(x - \hat{x})^2] = \int_{x_0 = x_{min}}^{x_{max}} (\hat{x} - x_0)^2 \cdot p_x(x_0) \cdot dx_0$$

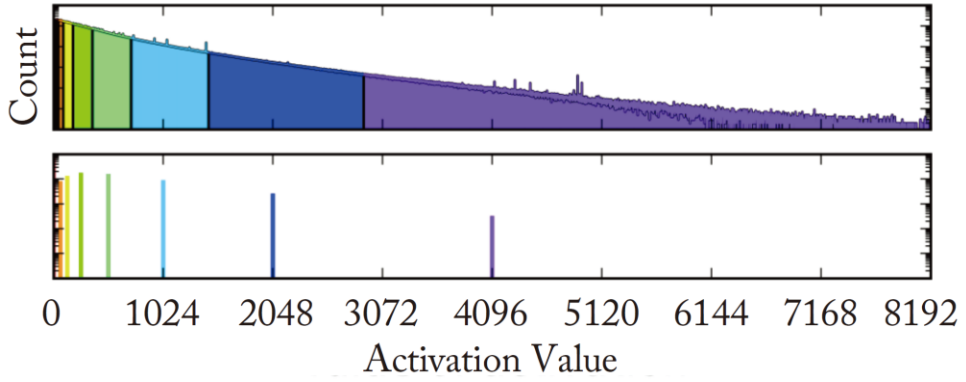
Fang, J., Shafiee, A., Abdel-Aziz, H., Thorsley, D., Georgiadis, G., & Hassoun, J. H. (2020, August). Post-training piecewise linear quantization for deep neural networks. In *European Conference on Computer Vision* (pp. 69-86). Springer, Cham.

Uniform and Non-uniform Quantization

- In DNN, we empirically observe that weight and activations for each layer has a bell-shaped distribution. This also holds for gradients during training
- Suppose that we model this data with a zero-mean Gaussian distribution
- Then in minimizing the mean quantization errors for x across all intervals, we should weight x's quantization error by the probability of occurrences of x. This leads to non-uniform quantization.



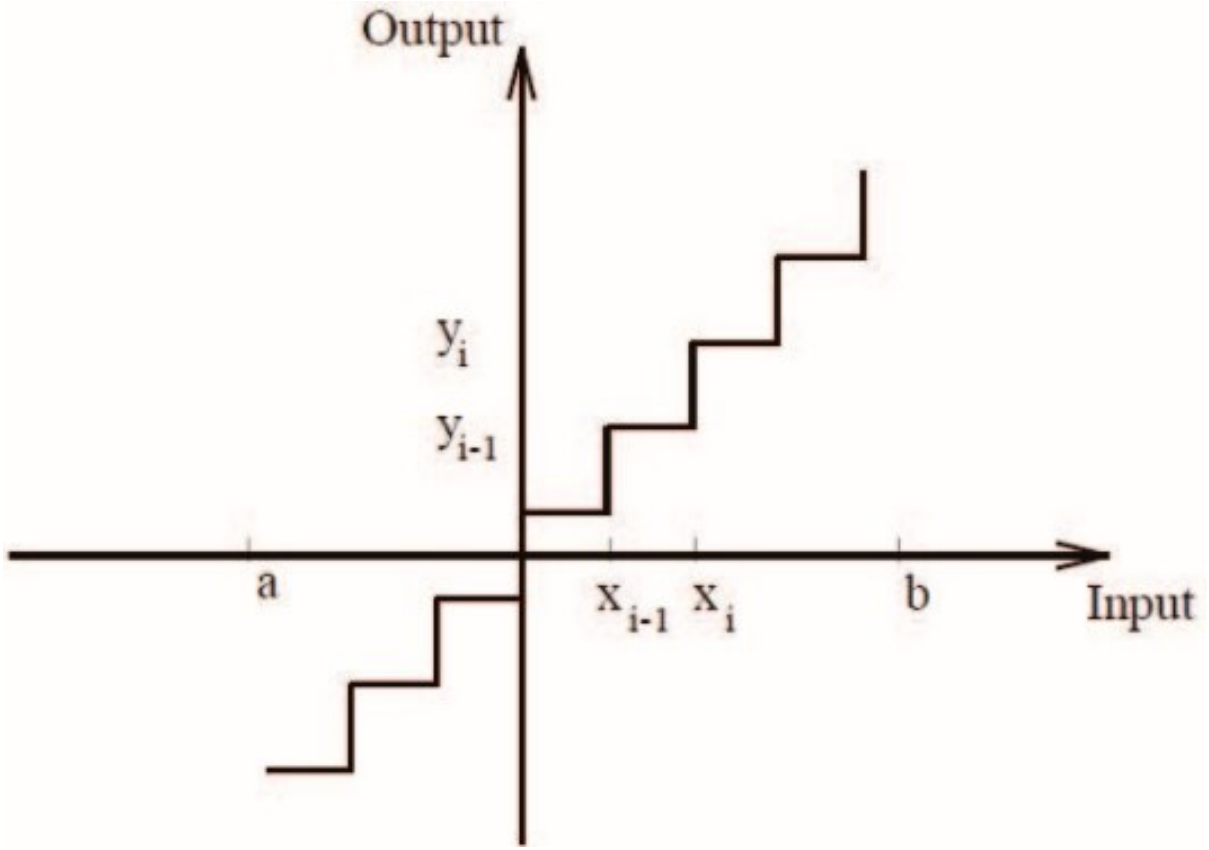
(a) Uniform quantization



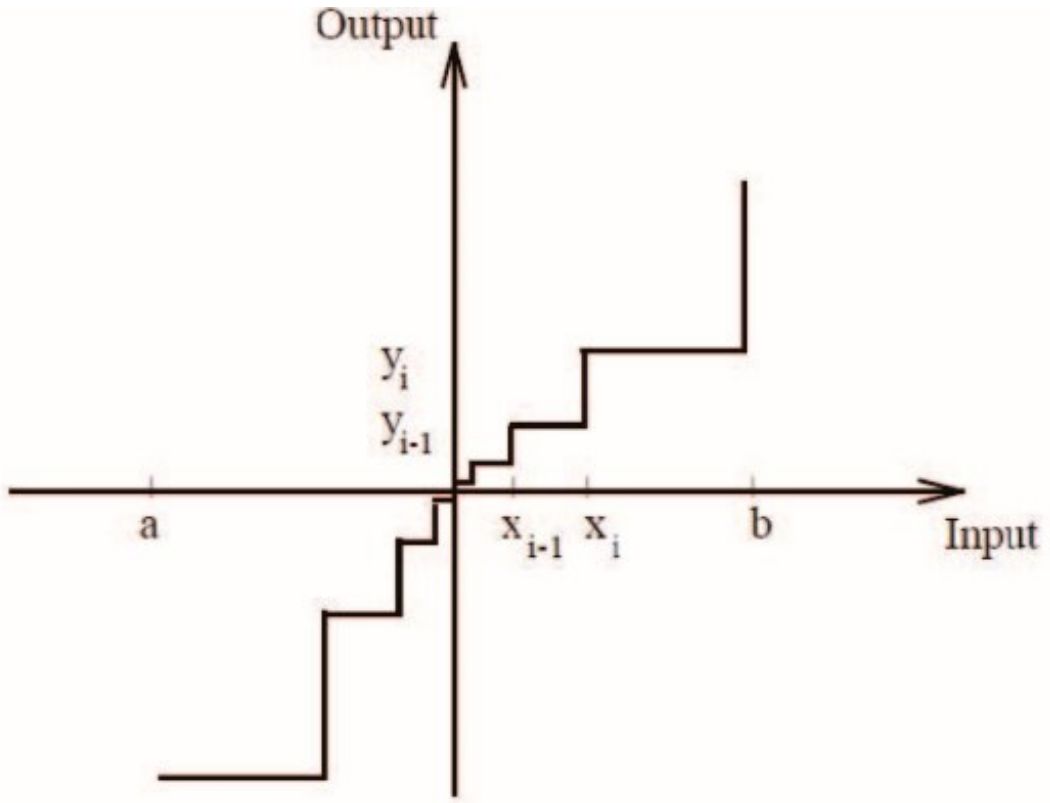
(b) Log quantization

Uniform and Non-uniform

uniform quantization function



Logarithmic quantization function



Logarithmic Quantization

- Use of power-of-two weights
- We can train a CNN using logarithmic quantization (LQ) where quantized weight values are power-of-two values, such as 2^1, 2^2, 2^3, etc.
- Then the multiplication in a MAC becomes just a simple shift
  - e.g., $2^3 \times 101_2 = 101000_2$
- This results in an efficient inference engine, for applications which require large dynamic ranges such as audio coding

McDanel, B., Zhang, S. Q., Kung, H. T., & Dong, X. (2019, June). Full-stack optimization for accelerating cnns using powers-of-two weights with fpga validation. In *Proceedings of the ACM International Conference on Supercomputing* (pp. 449-460).

Quantization

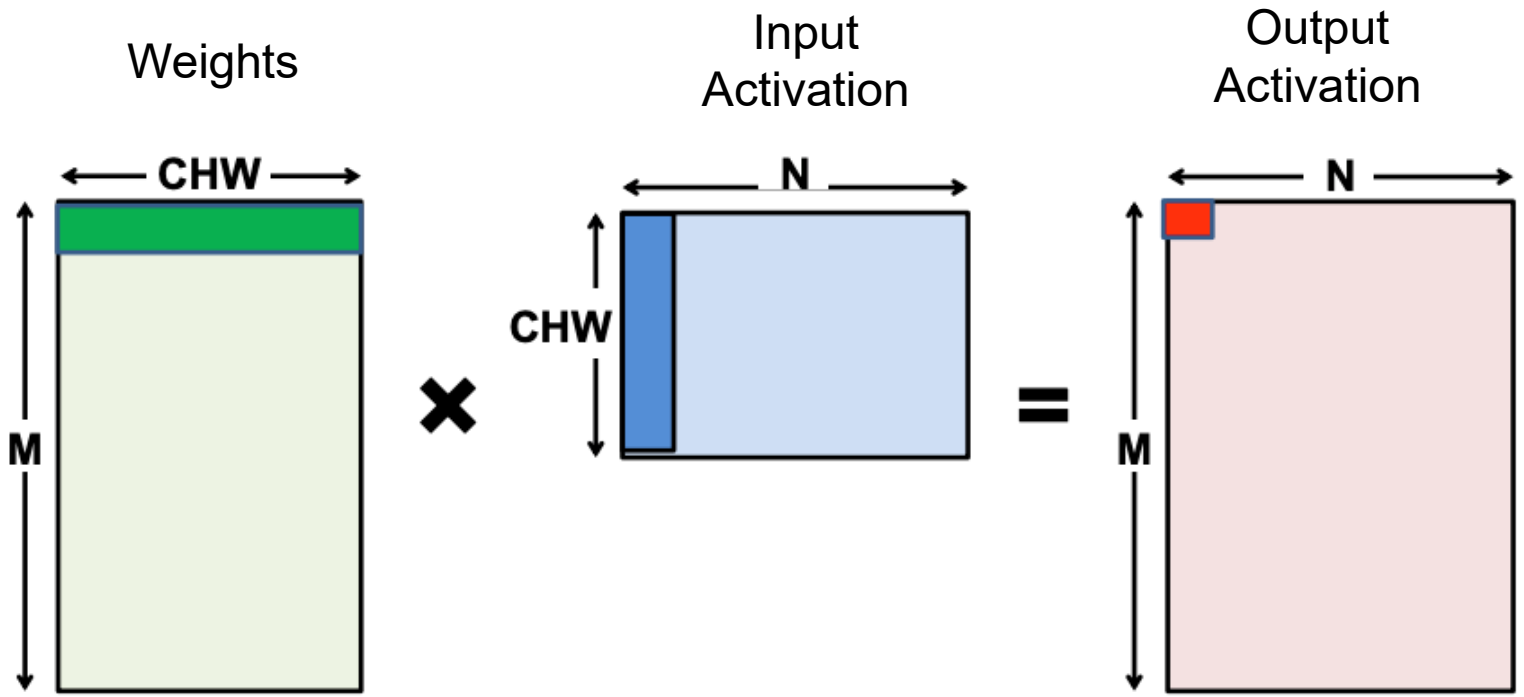
- Quantization scheme:
  - The correspondence between the fixed-point representation of values, i.e., **"q" for "quantized value"** and their floating-point value, i.e., **"r" for "real value".**
  - Recall "slope and bias" of fixed-point representation: $y = s \times x + z$

$$r = S(q - Z)$$

```
template<typename QType>  // e.g. QType=uint8
struct QuantizedBuffer {
  vector<QType> q;        // the quantized values
  float S;                // the scale
  QType Z;                // the zero-point
};
```

r: real floating-point value
q: quantized fixed-point value
S: scaling factor
Z: zero point (bias)

MAC in DNNs



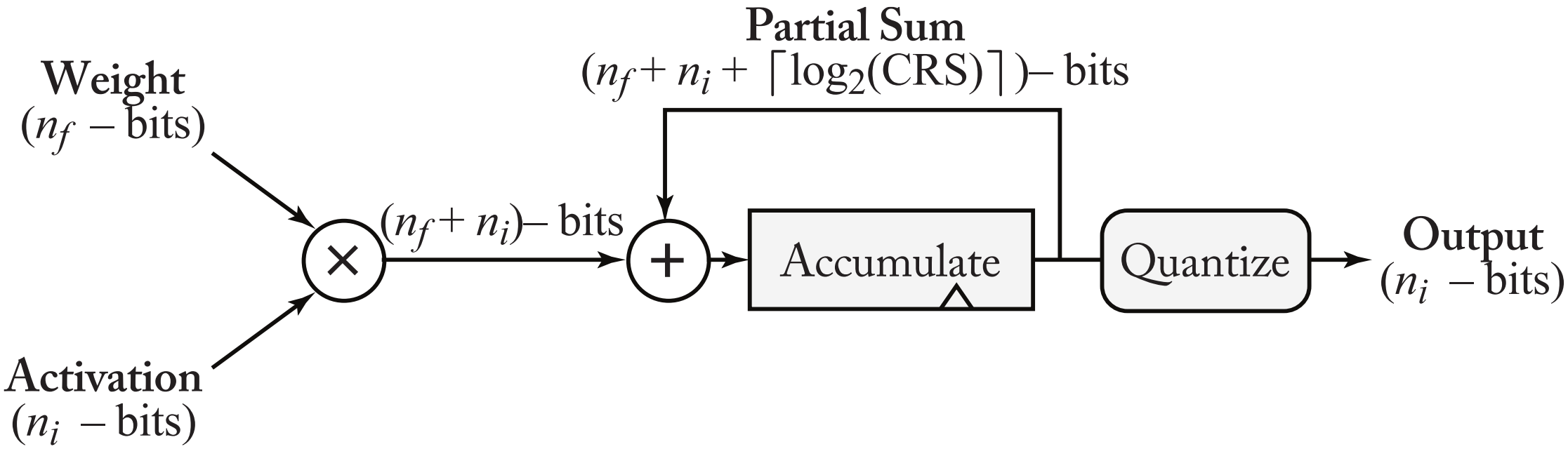
$$OA[i,k] = \sum_{j=1}^{N} (W[i,j] * IA[j,k])$$

$$OA[0,0] = \sum_{j=1}^{CHW} (W[0,j] * IA[j,0])$$

Eyeriss tutorial

MAC Precision

- For no loss in precision, **M** is determined based on largest filter size
  - In the range of 10 to 16 bits for popular DNNs



Fixed-Point Arithmetic

- Integers with a binary point and a bias
  - slope and bias: : $y = s \times x + z$
  - Qm.n: m (# of integer bits) n (# of fractional bits)

s = 1, z = 0

2^2	2^1	2^0	Val
0	0	0	**0**
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6**
1	1	1	**7**

s = 1/4, z = 0

2^0	2^{-1}	2^{-2}	Val
0	0	0	**0**
0	0	1	**1/4**
0	1	0	**2/4**
0	1	1	**3/4**
1	0	0	**1**
1	0	1	**5/4**
1	1	0	**6/4**
1	1	1	**7/4**

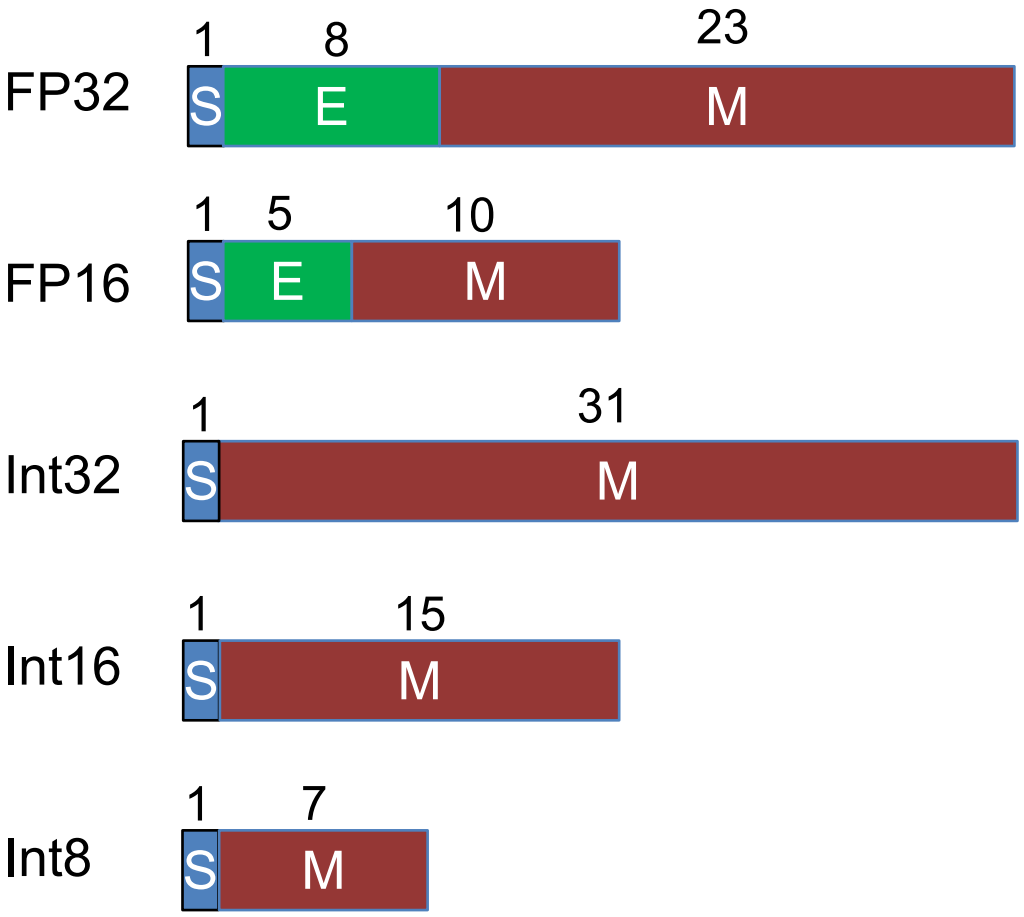
s = 4, z = 0

2^4	2^3	2^2	Val
0	0	0	**0**
0	0	1	**4**
0	1	0	**8**
0	1	1	**12**
1	0	0	**16**
1	0	1	**20**
1	1	0	**24**
1	1	1	**28**

s = 1.5, z =10

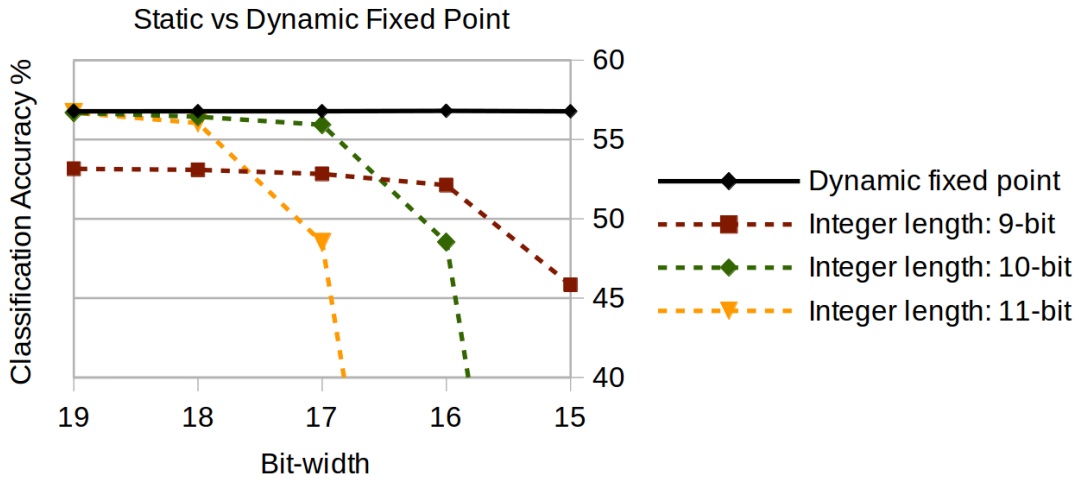
2^2	2^1	2^0	Val
0	0	0	**1.5*0 +10**
0	0	1	**1.5*1 +10**
0	1	0	**1.5*2 +10**
0	1	1	**1.5*3 +10**
1	0	0	**1.5*4 +10**
1	0	1	**1.5*5 +10**
1	1	0	**1.5*6 +10**
1	1	1	**1.5*7 +10**

Number Representation



	Range	Accuracy
FP32	$10^{-38} – 10^{38}$	.000006%
FP16	$6x10^{-5}$ - $6x10^{4}$	.05%
Int32	$0 – 2x10^{9}$	½
Int16	$0 – 6x10^{4}$	½
Int8	0 – 127	½

Impact on Accuracy



Without fine-tune

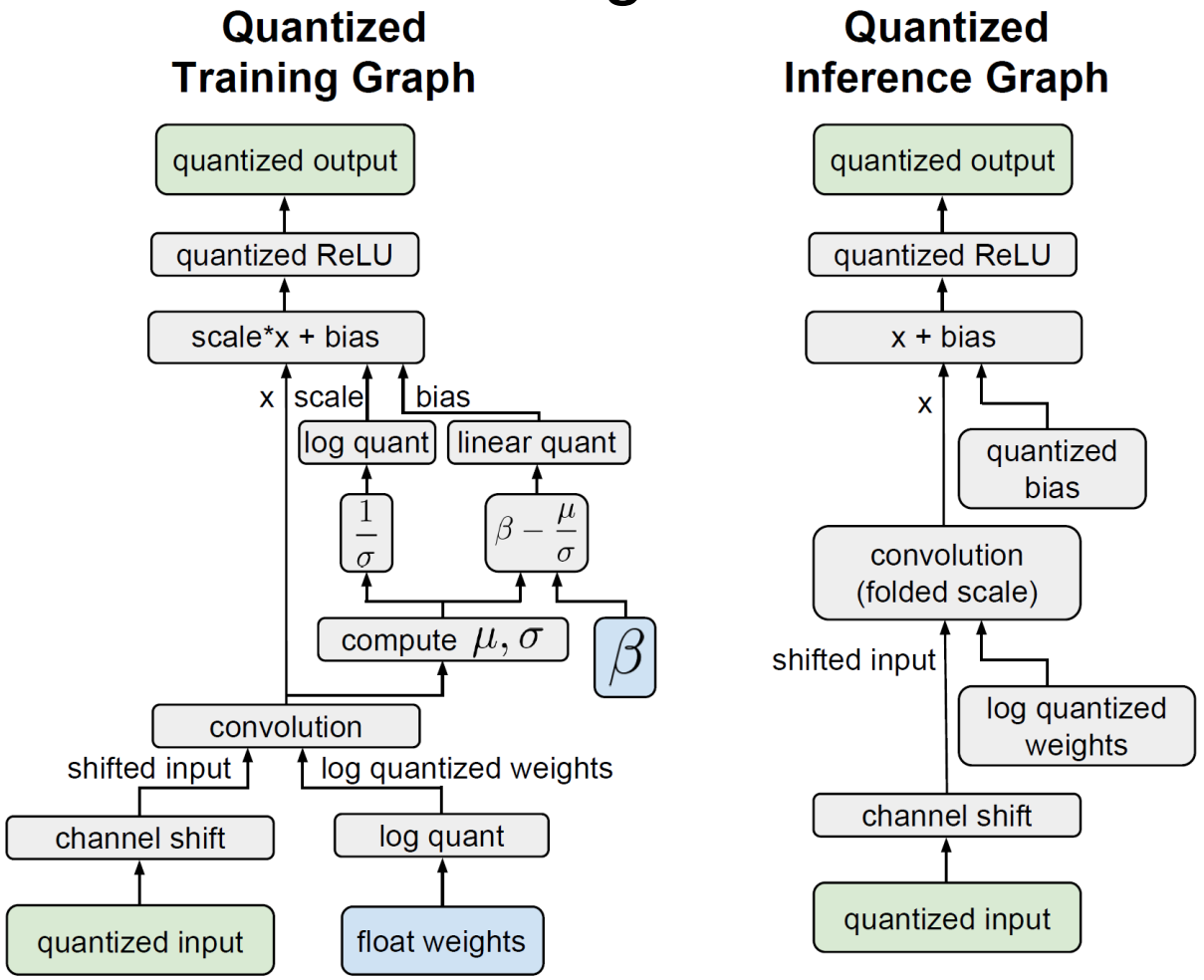
	Layer outputs	CONV parameters	FC parameters	32-bit floating point baseline	Fixed point accuracy
LeNet (Exp 1)	4-bit	4-bit	4-bit	99.1%	99.0% (98.7%)
LeNet (Exp 2)	4-bit	2-bit	2-bit	99.1%	98.8% (98.0%)
Full CIFAR-10	8-bit	8-bit	8-bit	81.7%	81.4% (80.6%)
SqueezeNet top-1	8-bit	8-bit	8-bit	57.7%	57.1% (55.2%)
CaffeNet top-1	8-bit	8-bit	8-bit	56.9%	56.0% (55.8%)
GoogLeNet top-1	8-bit	8-bit	8-bit	68.9%	66.6% (66.1%)

Two Quantization Styles

- Post-training quantization
  - Quantization applied to a trained model
  - Advantages
    - No need to access the original training data and training frameworks
    - No need to have training skills
    - Fast deployment
  - Post-training quantization has been successful.
    - E.g., FP32 7 INT8 and FP32 7 FP16 (with FP32 accumulation) are routinely used in deployment
  - Aggressive quantization schemes such as FP32 7 INT4 are still an active area of research
  - Post-training quantization still depends on a pre-trained model
    - training that model can itself be a challenge
- Quantization-aware training
  - Improving accuracy
  - Ultra-low-precision compression such as binarized or ternary models
  - Non-uniform quantization such as logarithmic quantization, we will need to retrain models by minimizing the loss of the quantized forward pass in training
  - For non-differentiable loss functions such as binary step functions, we may approximate gradients with the straight-through estimator (STE)

Quantization-aware Training

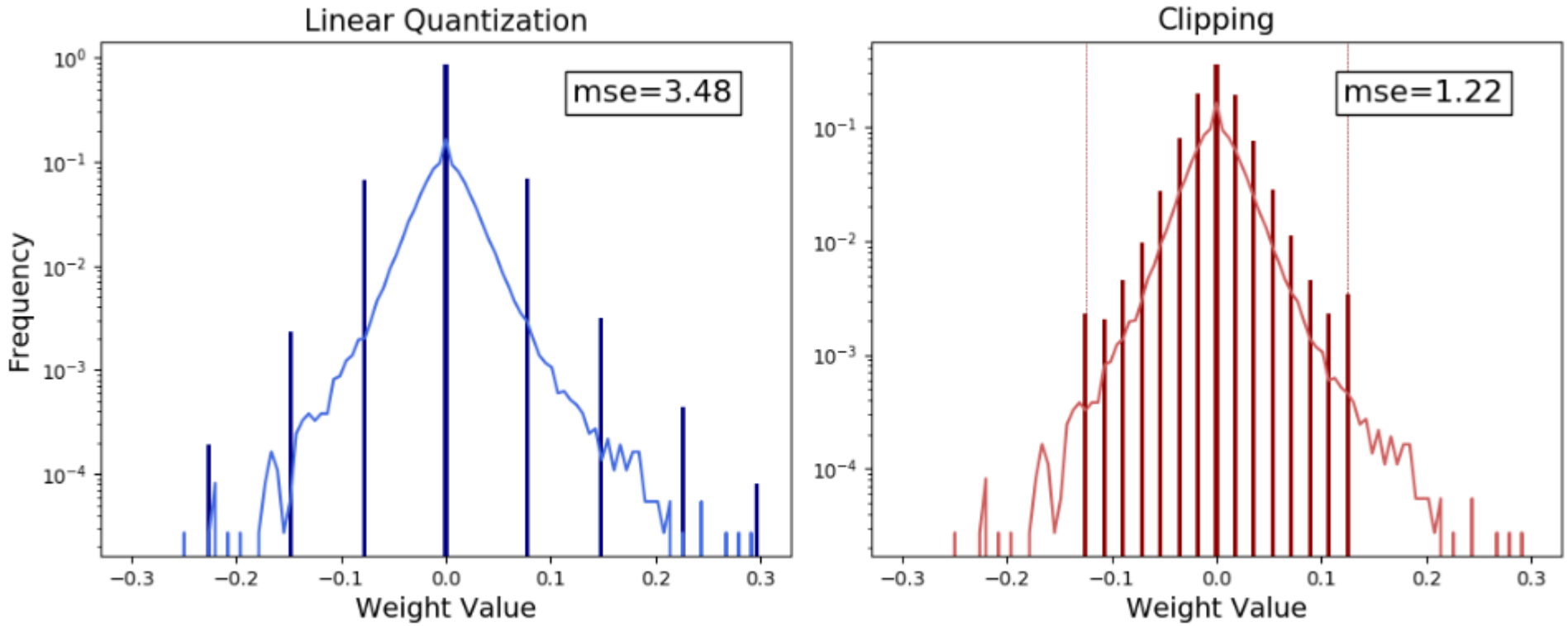
- Adding Quantization to Training to Match Quantized Inferenc



Outline

- Overview
- Basics of DNN Quantization
- Quantization Range Clipping
- Advanced Quantization
- Hardware Design

Clipping


Linear Quantization
mse=3.48
Frequency
Weight Value
Clipping
mse=1.22
Weight Value

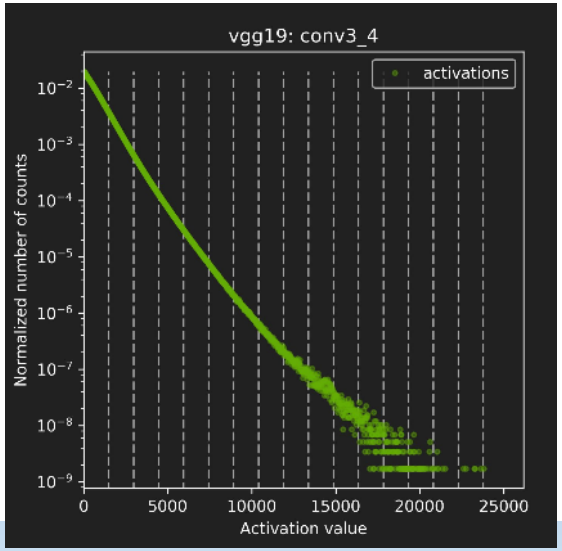
Percentile Clipping

- Running several training data batches through the un-quantized network to determine the maximum range for uniform quantization
- For each layer, determine y_{max} as the maximum across all batches of the 99.99th percentile

KL divergence

- KL divergence
  - TensorRT profiles the activation distributions using a small number (1000s) of user-provided training samples to computes a clipping threshold by minimizing the KL divergence between the original and quantized distributions

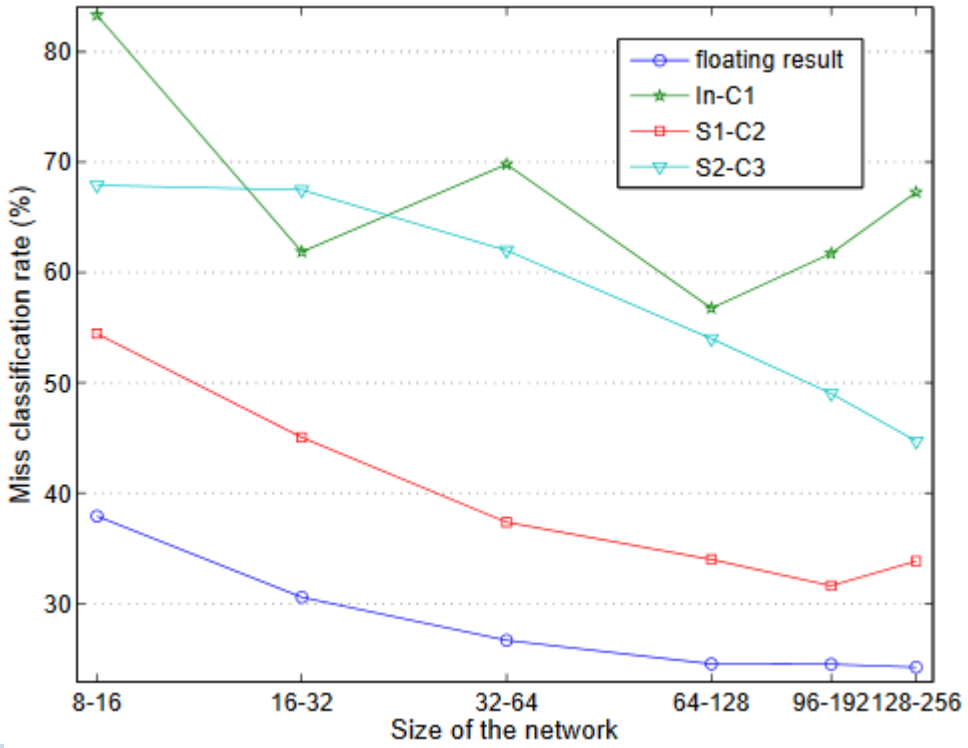
  1. Collect histograms of activations
  2. Generate many quantized distributions with different saturation thresholds
  3. Pick threshold which minimizes KL_divergence(ref_distr, quant_distr).



L2 Norm Clipping

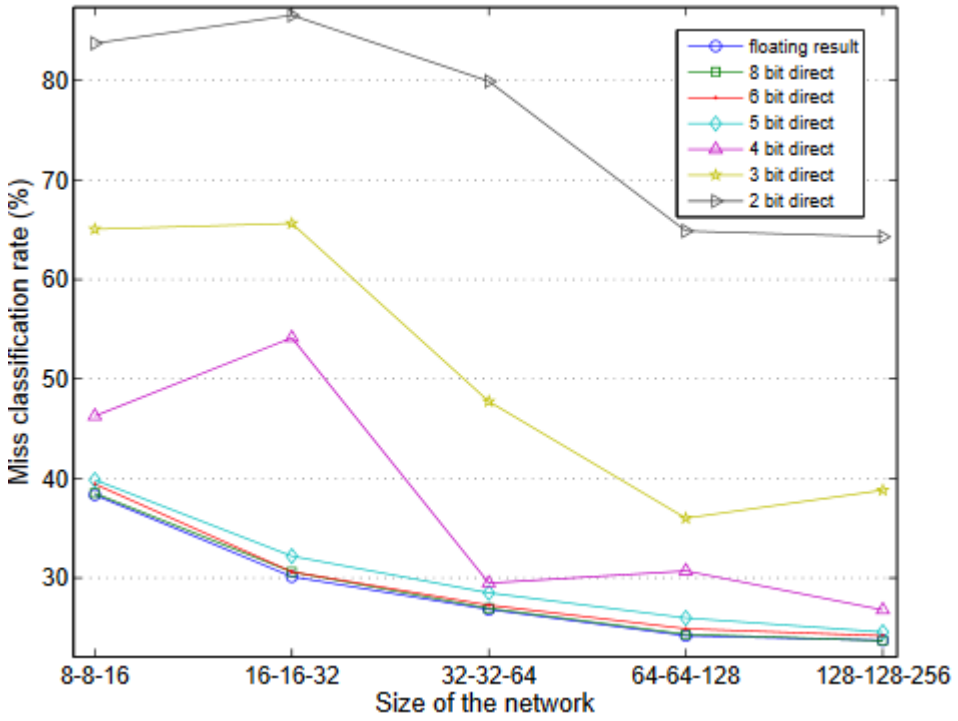
- Direct Quantization

$$Q(w) = sgn(w) \cdot \Delta \cdot min\left(\left\lfloor \frac{|(w)|}{\Delta} + 0.5 \right\rfloor, \frac{M-1}{2}\right)$$



- Minimize L2 error E to determine step size

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(Q(w_i) - w_i\right)^2$$



Analytical Clipping for Integer Quantization(ACIQ)

- ACIQ
  - Fit a Gaussian and Laplacian to the sampled distribution
  - Use the better-fitting curve to analytically compute the optimal clip threshold

$$\text{clip}(x,\alpha) = \begin{cases} x & \text{if } |x| \le \alpha \\ \text{sign}(x)\cdot\alpha & \text{if } |x| > \alpha \end{cases}$$

$$\Delta = \frac{2\alpha}{2^M}$$

$$E[(X-Q(X))^2] = \int_{-\infty}^{-\alpha} f(x)\cdot(x+\alpha)^2 dx + \sum_{i=0}^{2^M-1}\int_{-\alpha+i\Delta}^{-\alpha+(i+1)\Delta} f(x)\cdot(x-q_i)^2 dx + \int_{\alpha}^{\infty} f(x)\cdot(x-\alpha)^2 dx$$

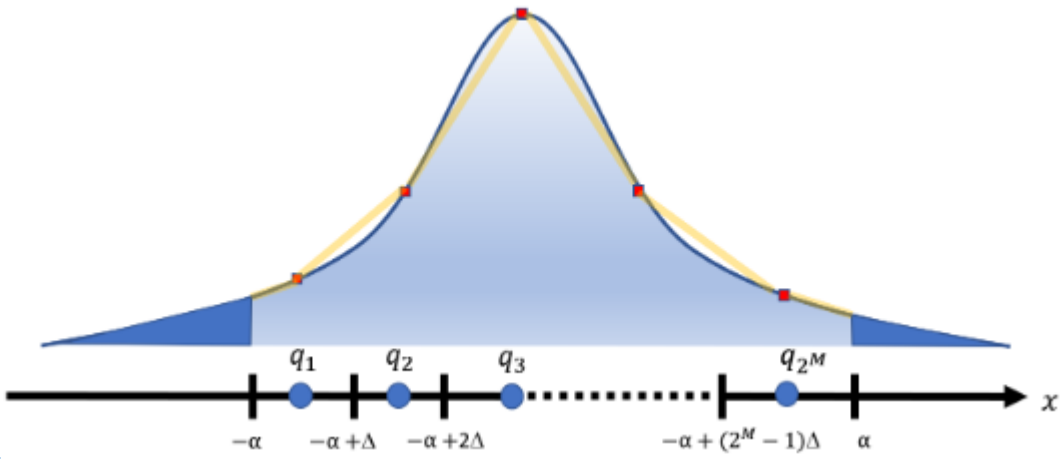
Quantify the contribution of clip(x,α) to the expected mean-square-error

quantization noise introduced when high precision values in the range[−α,α]are rounded to the nearest discrete value.

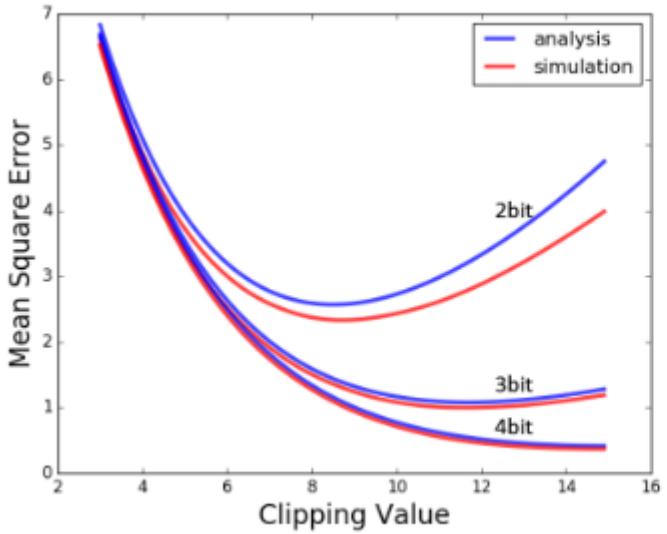
Banner, R., Nahshan, Y., Hoffer, E., & Soudry, D. (2018). ACIQ: analytical clipping for integer quantization of neural networks.

Analytical Clipping for Integer Quantization(ACIQ)

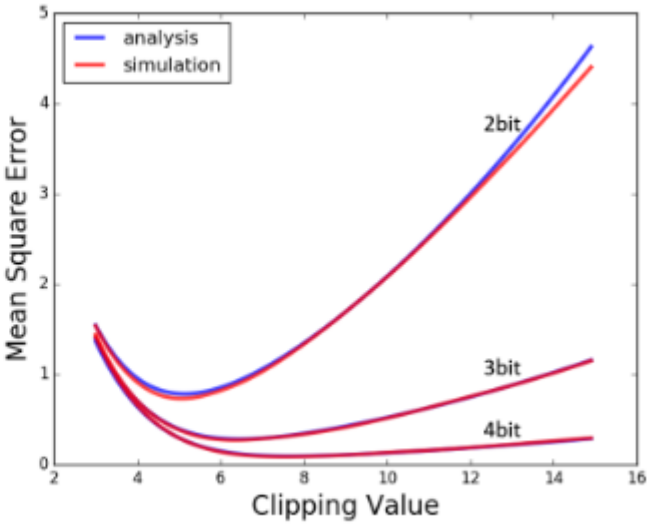
- Uniform Quantization



- Clipping Value v.s. MSE



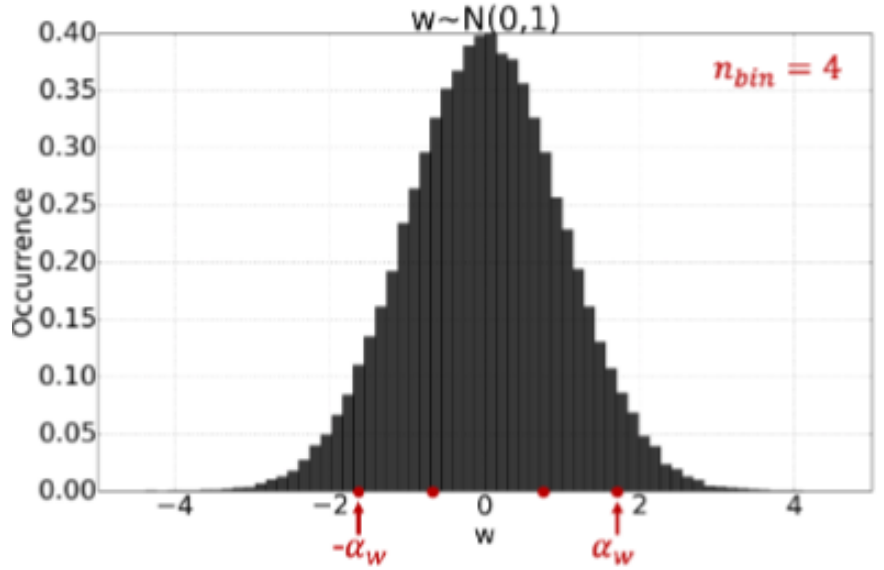
(a) Laplace ($\mu = 0$ and $b = 1$)



(b) Gaussian ($\mu = 0$ and $\sigma = 1$)

Statistics-Aware Weight Binning(SAWB)

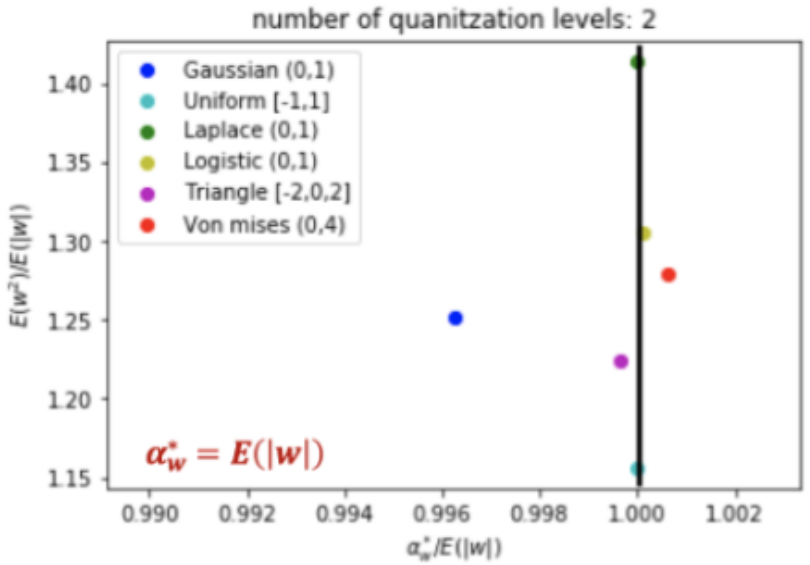
- SAWB
  - Finds the optimal scaling factor that minimizes the quantization error based on the statistical characteristics of the distribution of weights
  - α_w – Scaling factor, defines the largest quantization level
  - Find $\alpha_w^* = \arg\min_{\alpha_w} ||w - w_q||^2$
- Use 6 distributions for a wide range of quantization level
  - Gaussian with [μ:0,σ2:1]
  - Uniform distribution with μ=0 and a range of [-1:1],
  - Laplace distribution with μ=0 and decay of 1
  - Logistic distribution with μ=0 and scale 1
  - Triangle distribution with μ=0 and extreme 2
  - Von Mises distribution with center 0 and dispersion 4.
- Given a bin level n_{bin}, linear regression can be applied to derive α_w^*as a function of $\sqrt{E(w^2)}$and E(|w|)
- Compute E(|w|) and $\sqrt{E(w^2)}$
- Applied the linear coefficients -> empirically determined to calculate $\hat{\alpha}_w^*$

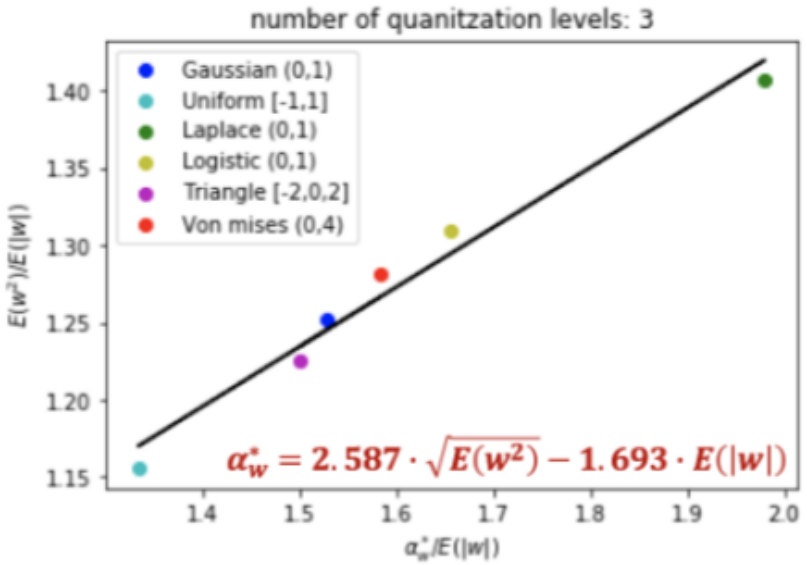


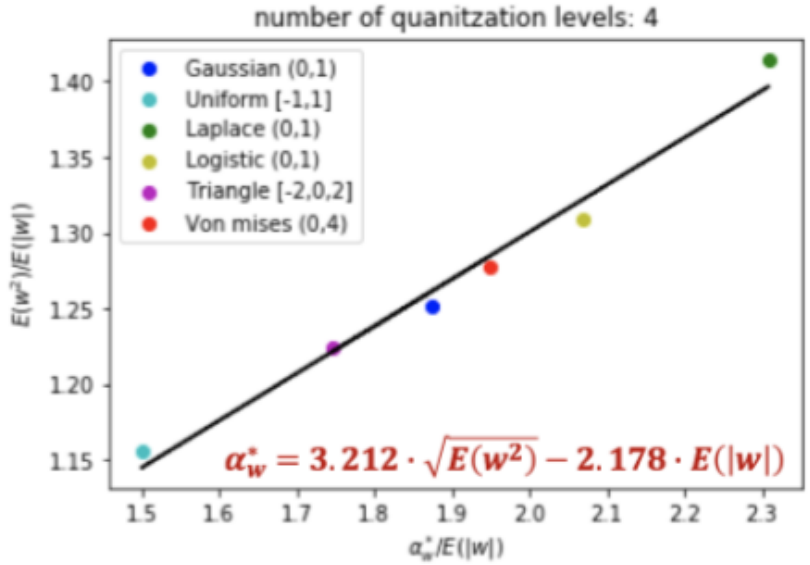
Choi, J., Chuang, P. I. J., Wang, Z., Venkataramani, S., Srinivasan, V., & Gopalakrishnan, K. (2018). Bridging the accuracy gap for 2-bit quantized neu networks (qnn). *arXiv preprint arXiv:1807.06964*.

Statistics-Aware Weight Binning(SAWB)

- Derivation of empirical equation for determining optimal quantization from statistical distribution of weights





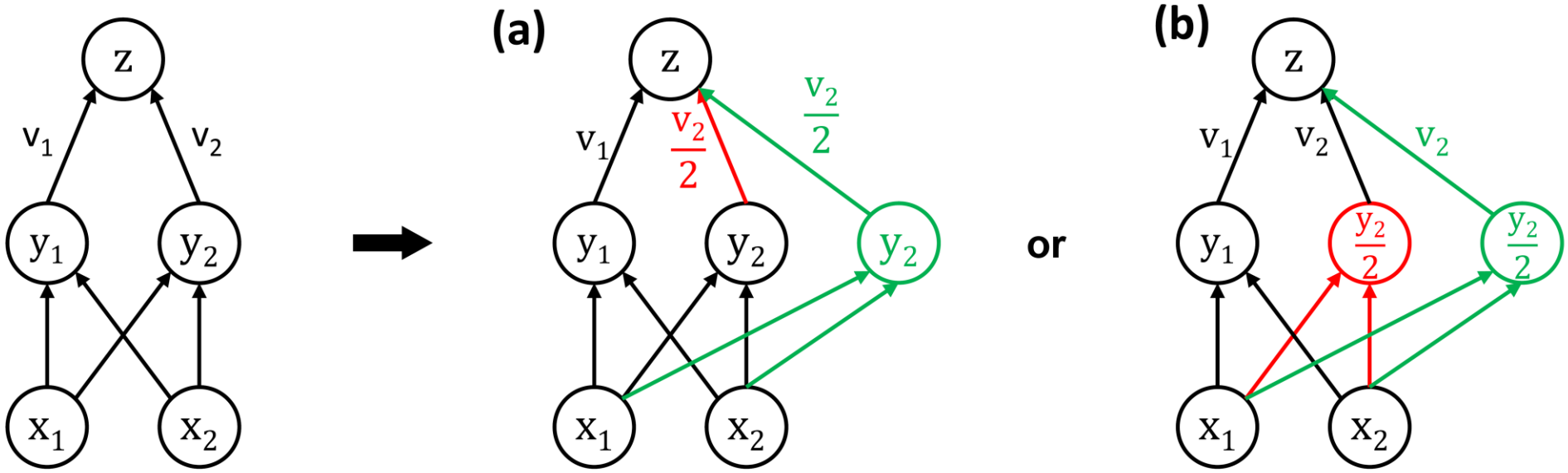


Outline

- Overview
- Basics of DNN Quantization
- Quantization Range Clipping
- Advanced Quantization
- Hardware Design

Outlier Channel Splitting

- **Split** weights or activation to have smaller absolute values



$$z = v_1 y_1 + v_2 y_2$$

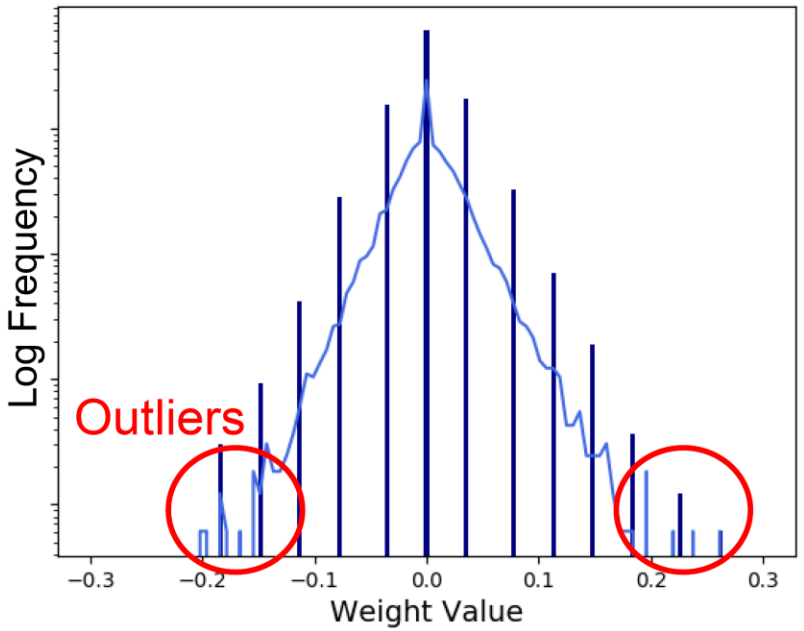
$$z = v_1 y_1 + \frac{v_2}{2} y_2 + \frac{v_2}{2} y_2$$

$$z = v_1 y_1 + v_2 \frac{y_2}{2} + v_2 \frac{y_2}{2}$$

Zhao, R., Hu, Y., Dotzel, J., De Sa, C., & Zhang, Z. (2019, May). Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning* (pp. 7543-7552). PMLR.

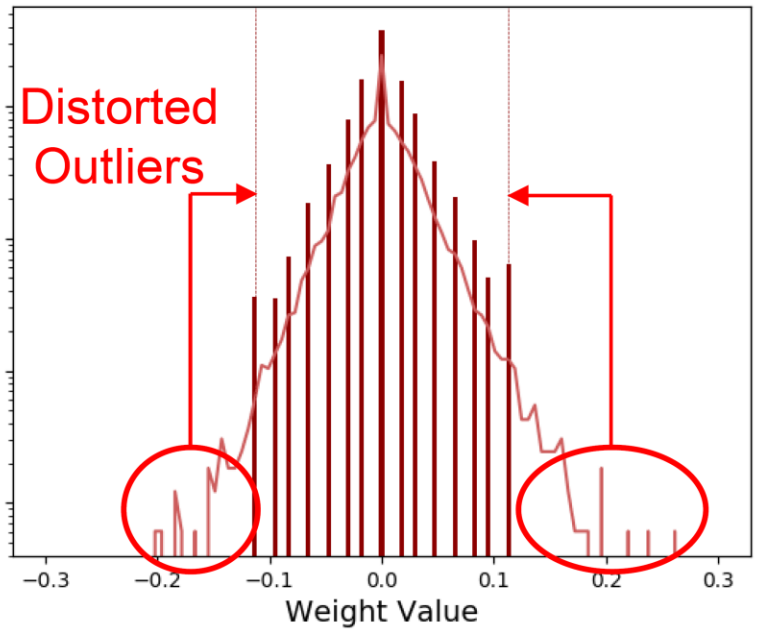
Outlier Channel Splitting

Linear Quantizer



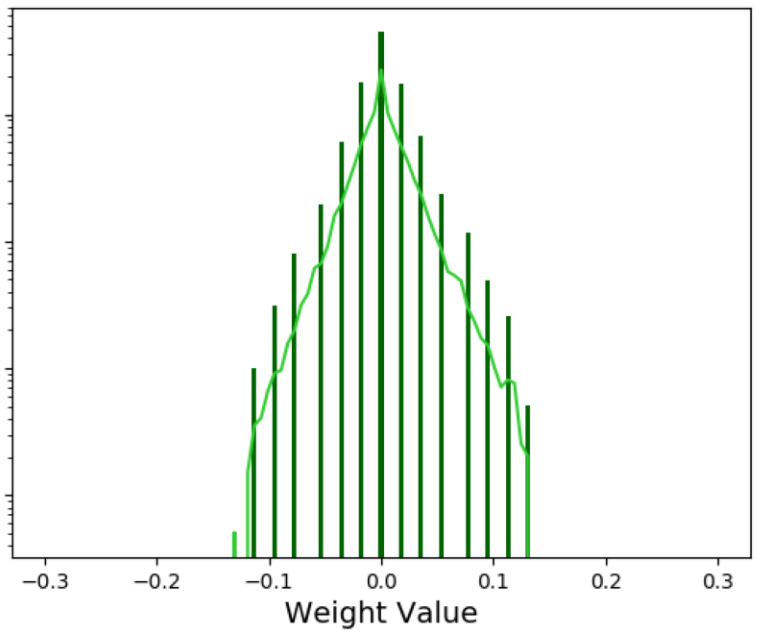
− Poor quantizer resolution due to outliers

Clipping



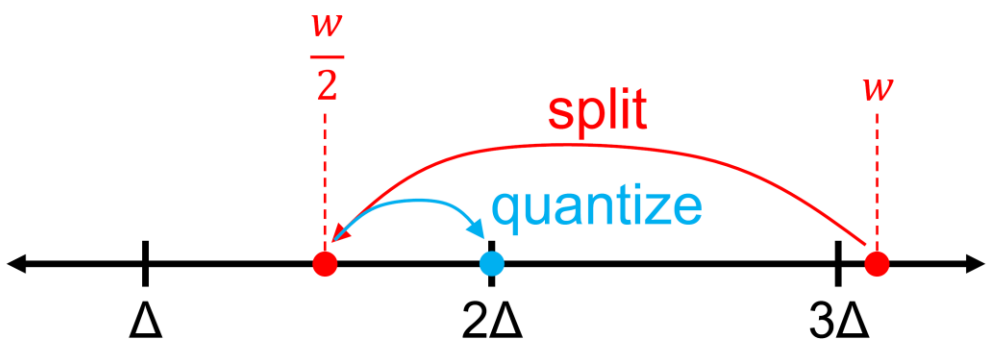
+ Reduces quantization noise
+ Used in NVIDIA TensorRT
− Distorts outliers

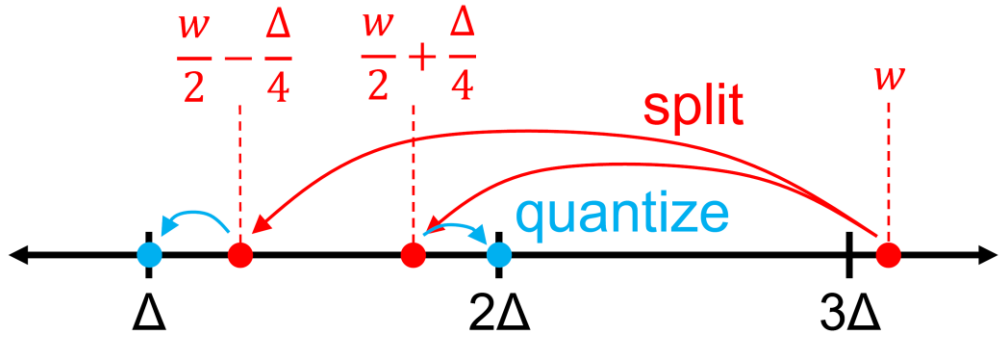
Outlier Channel Splitting



+ Reduces quantization noise
+ Removes outliers
− Model size overhead

Quantization-Aware Splitting





- **Naïve Splitting**

$$w \rightarrow (\frac{w}{2}, \frac{w}{2})$$

Halves round in the same direction

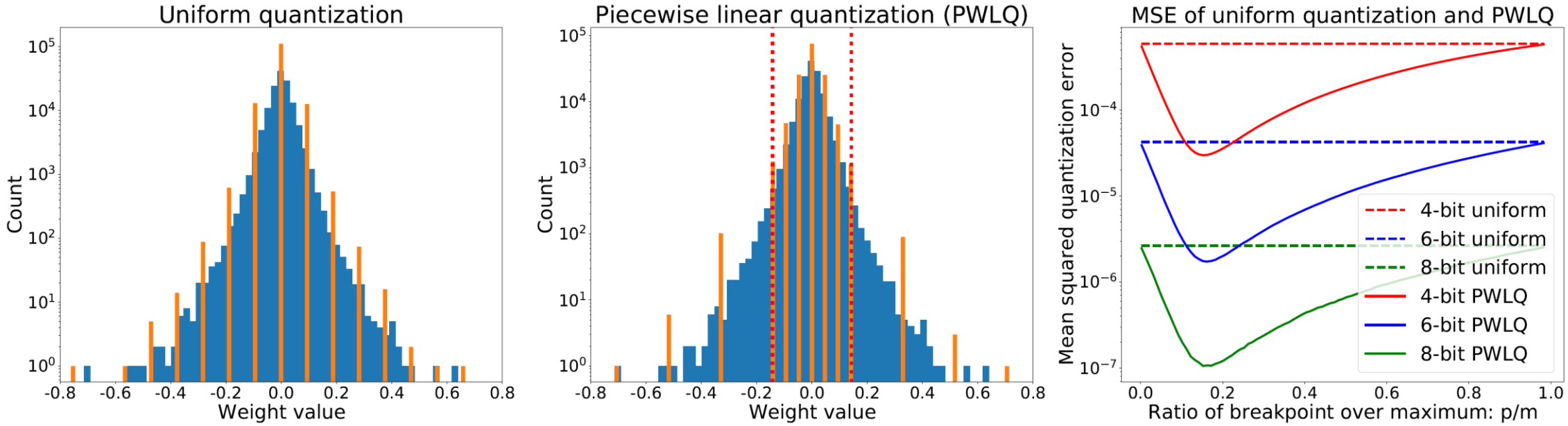
- **Quantization-Aware Splitting**

$$w \rightarrow (\frac{w}{2} - \frac{\Delta}{4}, \frac{w}{2} + \frac{\Delta}{4})$$

Halves can round in opposite directions to help reducing quantization error

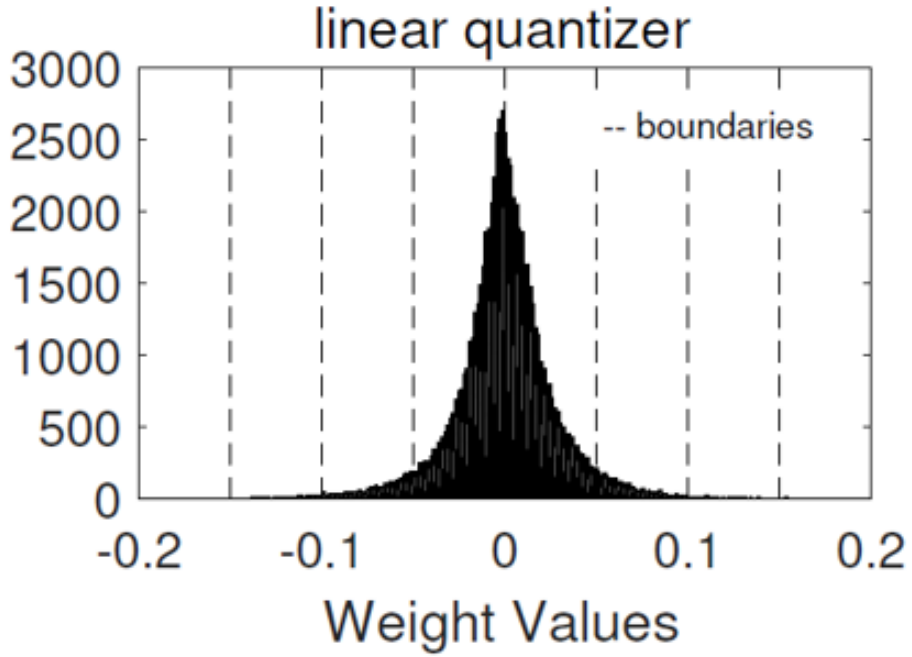
Post-Training Piecewise Linear Quantization (PWLQ)

- Center region has most of the value
  - Smaller region between each breakpoint near center
  - Reducing quantization error
- Linear quantization in each region -> piecewise linear

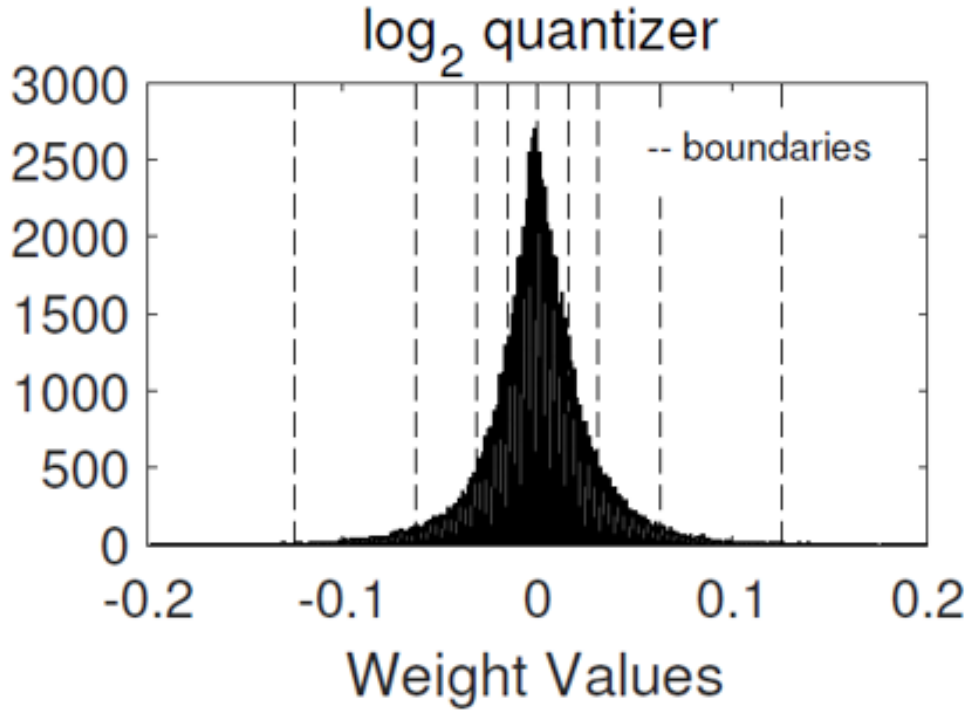


Fang, J., Shafiee, A., Abdel-Aziz, H., Thorsley, D., Georgiadis, G., & Hassoun, J. H. (2020, August). Post-training piecewise linear quantization for deep neural networks. In *European Conference on Computer Vision* (pp. 69-86). Springer, Cham.

Log Domain Quantization



Product = X * W



Product = X << W

Lee, E. H., Miyashita, D., Chai, E., Murmann, B., & Wong, S. S. (2017, March). Lognet: Energy-efficient neural networks using logarithmic computation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5900-5904). IEEE.

Per-Channel Bit-Allocation

- Dedicated scale and offset for each channel
  - Different channels have different numbers of bits for precision
  - Restrict total number of bits written to or read from memory remains unchanged
  - Allow some of the channels to have higher bit-width while limiting other channels to have a lower bit-width

Clipping Noise

$$E[(X - Q(X))^2] \approx 2 \cdot b^2 \cdot e^{-\frac{\alpha}{b}} + \frac{2 \cdot \alpha^3}{3} \cdot \sum_{i=0}^{2^M - 1} f(q_i) = 2 \cdot b^2 \cdot e^{-\frac{\alpha}{b}} + \frac{\alpha^2}{3 \cdot 2^{2M}}$$

Channel i has values in the range $[-\alpha_i, \alpha_i]$ quantized to M_i bits of precision

Lagrangian

$$\mathcal{L}(M_0, M_1, ..., M_n \lambda) = \sum_i \left(2 \cdot b^2 \cdot e^{-\frac{\alpha_i}{b}} + \frac{\alpha_i^2}{3 \cdot 2^{2M_i}} \right) + \lambda \left(\sum_i 2^{M_i} - B \right)$$

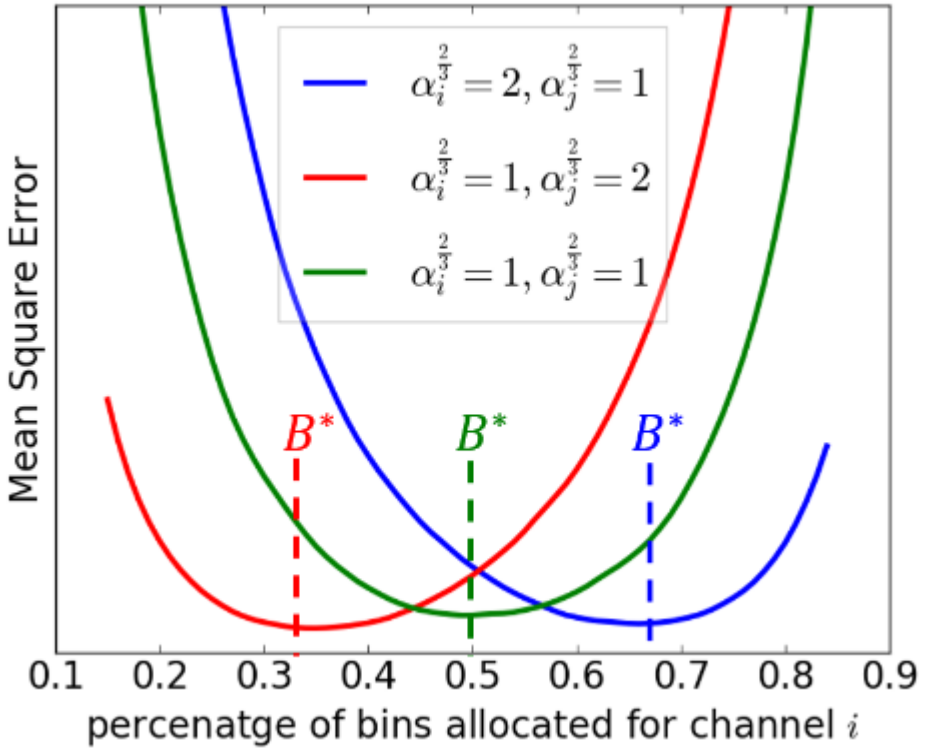
total layer quantization noise

captures the quota constraint on the total number of allowed bins B

Solve to get the width assignment for each channel I => $M_i = \left\lfloor \log_2 \left(\frac{\alpha_i^{\frac{2}{3}}}{\sum_i \alpha_i^{\frac{2}{3}}} \cdot B \right) \right\rfloor$

Per-Channel Bit-Allocation

- Optimal bin-allocation in a synthetic experiment including of a pair of channels i,j
- Each consisting of 1000 values taken from $N(0, \alpha_i^2)$ and $N(0, \alpha_j^2)$
- The overall bin quota for the layer is set to B= 32
  - Equivalent in terms of memory bandwidth to the number of bins allocated for two channels at 4-bit precision



Bias-Correction

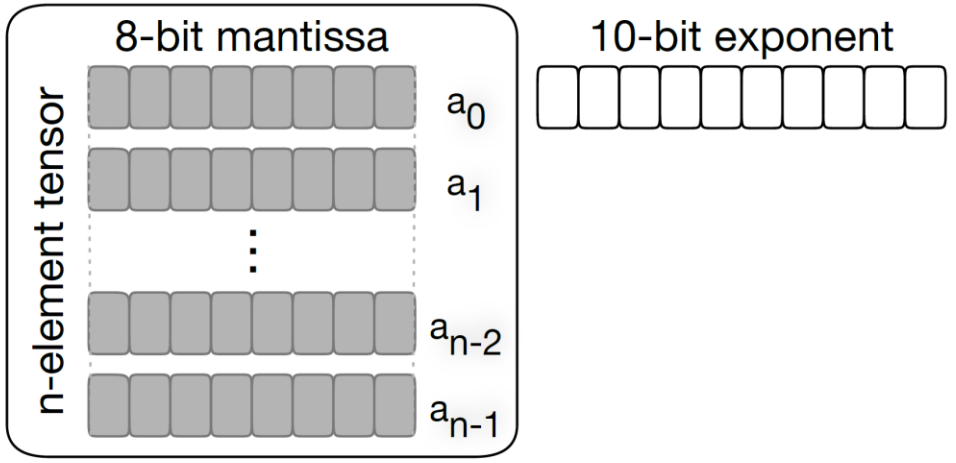
- An inherent bias in the mean and the variance of the weight values following their quantization
  - For channel c, $E(W_c) \neq E(W_c^q)$ and $\|W_c - E(W_c)\|_2 \neq \left\|W_c^q - E(W_c^q)\right\|_2$
  - Evaluate correction constants for each channel c

$$\mu_c = \mathbb{E}(W_c) - \mathbb{E}(W_c^q) \qquad \xi_c = \frac{||W_c - \mathbb{E}(W_c)||_2}{||W_c^q - \mathbb{E}(W_c^q)||_2}$$

  - Compensate for the bias in W_c^q for each channel c

$$w \longleftarrow \xi_c (w + \mu_c), \quad \forall w \in W_c^q$$

Block Floating Point (BFP)



- **FP representation with an exponent per tensor element.**
- **BFP representation with an exponent per tensor**

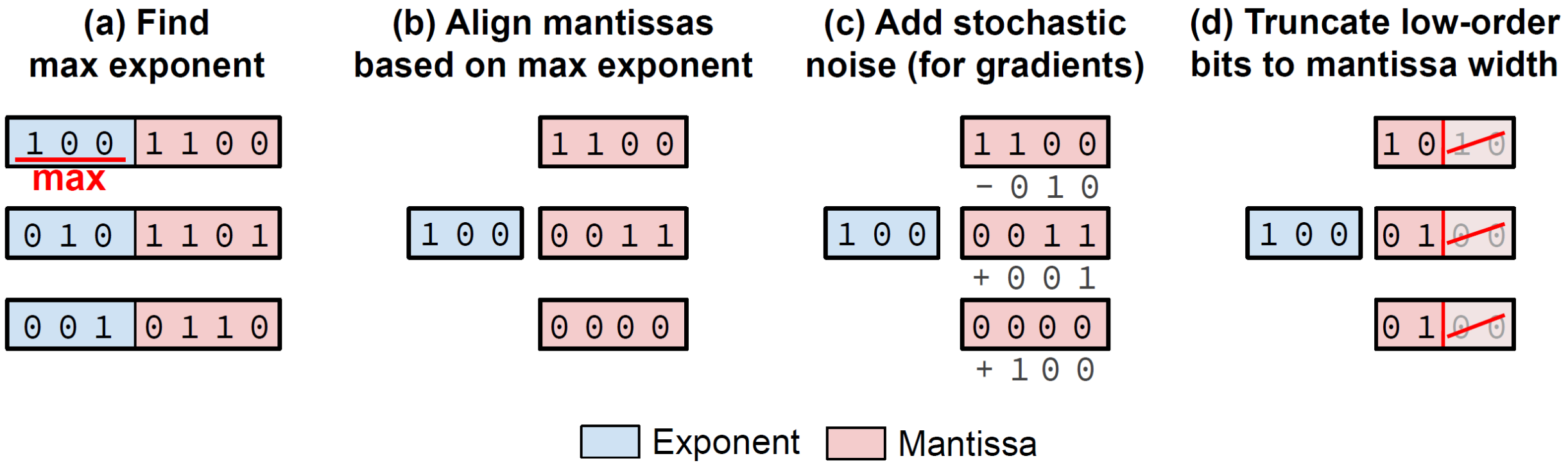
Block Floating Point Multiplication

- Floating Point with the cost of integer operations
- Consider $a_i = m_i^a \times 2^{e_a}$, $b_i = m_i^b \times 2^{e_b}$

$$a \times b = \sum_{i=1}^{N} ((m_i^a \times 2^{e_a}) \times (m_i^b \times 2^{e_b}))$$
$$= 2^{e_a + e_b} \times (m_i^a + m_i^b)$$

- BFP perform exponent additions and mantissa alignments only at the group level, rather than at the individual elements level, which FP does.
- Adjust mantissa lengths → variable precision

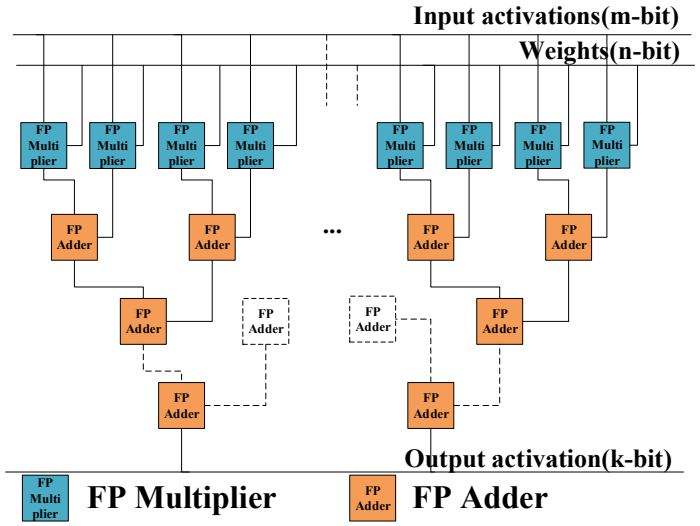
Applying Stochastic Rounding to Ultra-low-precision BFP

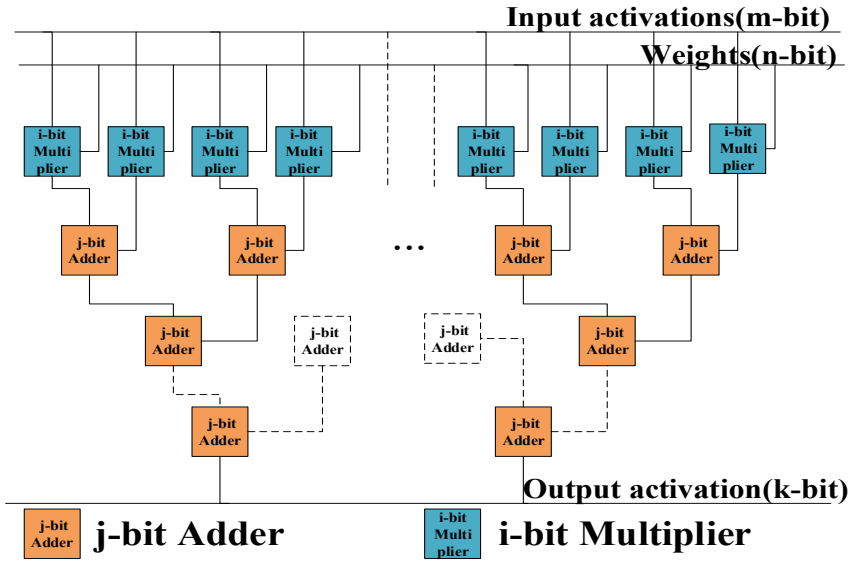


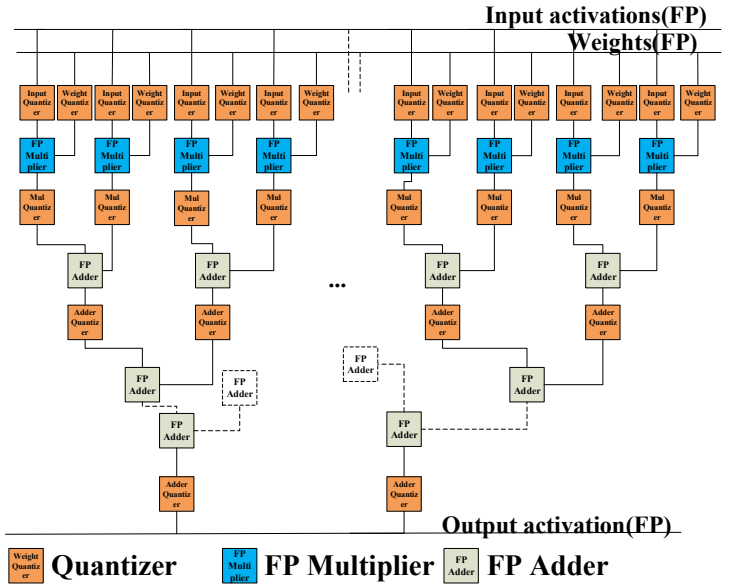
Why stochastic rounding?
Compared to the weights and activations, the gradients have a much wider exponent disparity, leading to a larger quantization error

Zhang, S. Q., McDanel, B., & Kung, H. T. (2022, April). Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 846-860). IEEE.

Bit-accurate Quantization



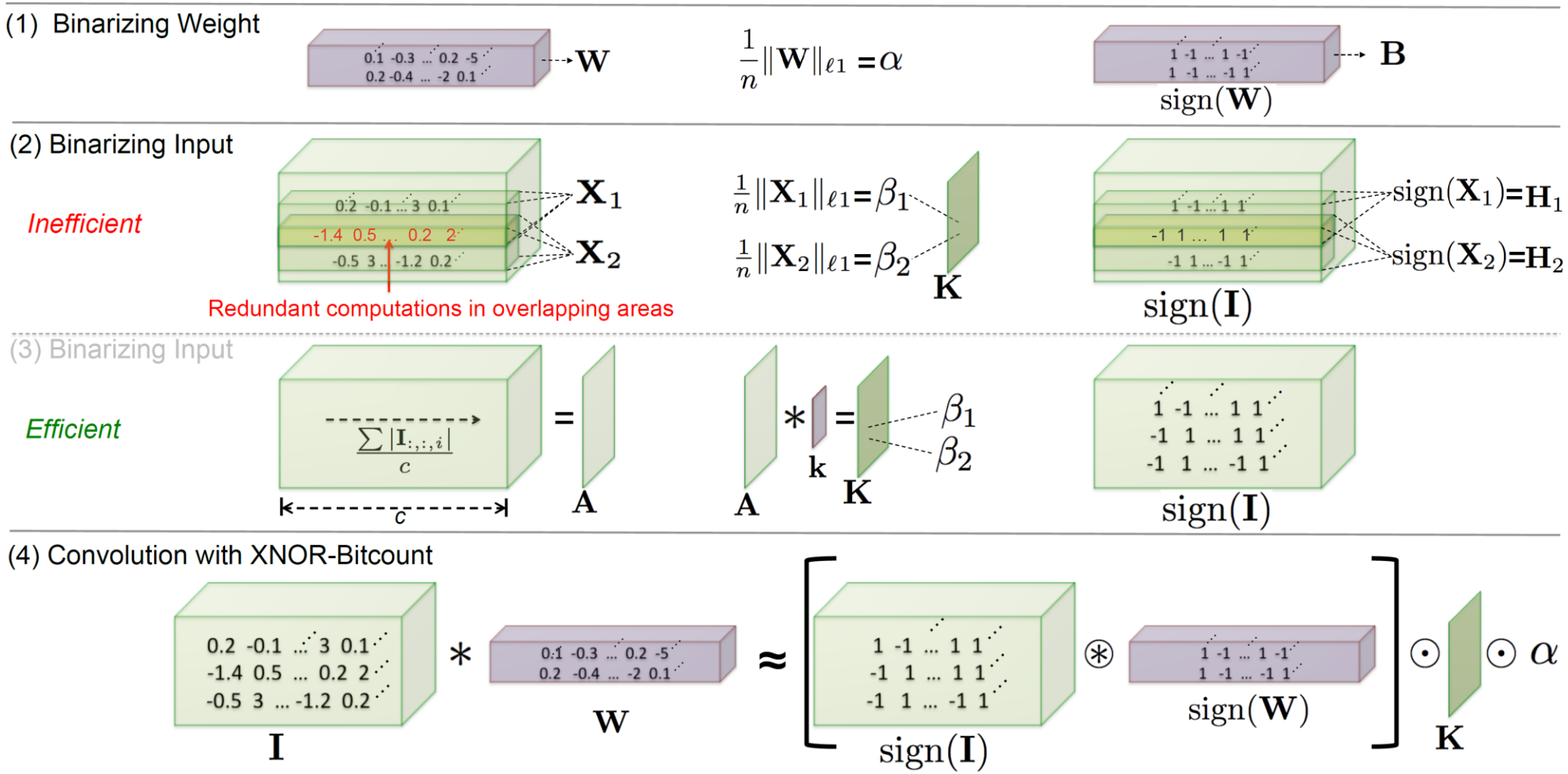




- Layer-by-layer level quantization for convolution realized with floating-point multipliers and adders
- Hardware-level quantization for convolution realized in hardware with i-bit multipliers and j-bit adders
- Bit-accurate convolution realization using FP operators followed by quantizers

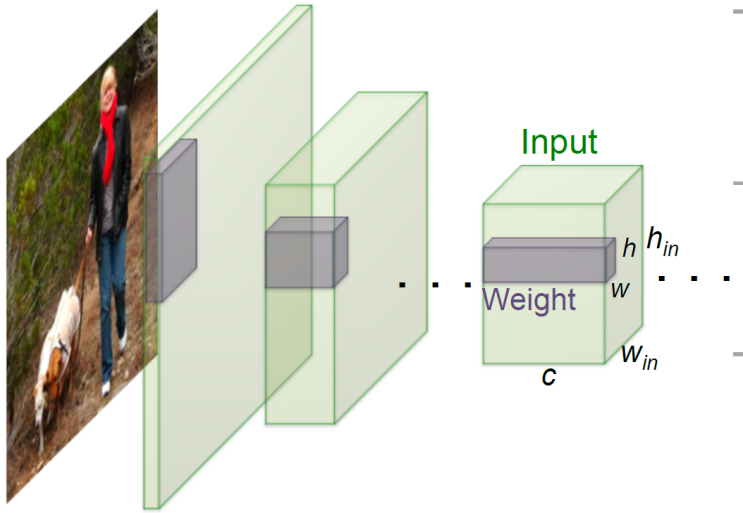
Tsai, C. C., & Guo, J. I. (2021). IVS-Caffe--Hardware-Oriented Neural Network Model Development. *IEEE Transactions on Neural Networks and Learning Systems*.

Binarized Neural Network



Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, October). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision* (pp. 525-542). Springer, Cham.

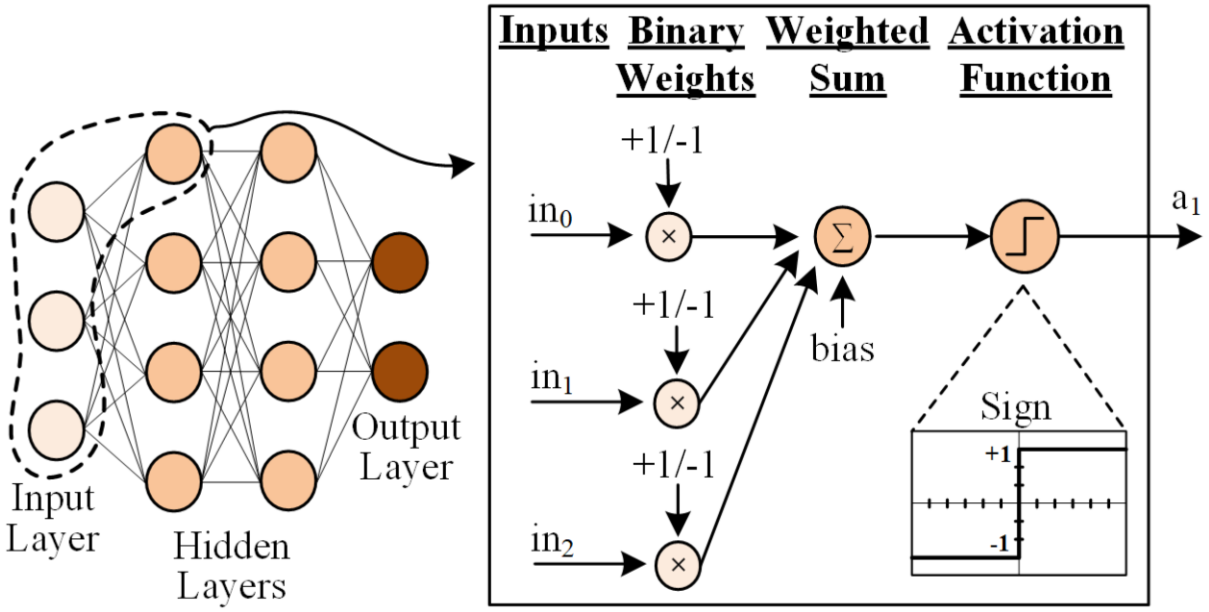
Computation and Memory Saving for BNN



	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.2 ... 0.41 / -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	+ , −	~32x	~2x	%56.8
BinaryWeight Binary Input (**XNOR-Net**)	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2

XNOR-net

- Encoding for {+1, -1} and XNOR for Multiply
  - Bit 0 means value -1
  - Bit 1 means value +1



Encoding (Value)		XNOR (Multiply)
0 (-1)	0 (-1)	1 (+1)
0 (-1)	1 (+1)	0 (-1)
1 (+1)	0 (-1)	0 (-1)
1 (+1)	1 (+1)	1 (+1)

Mixed Precision

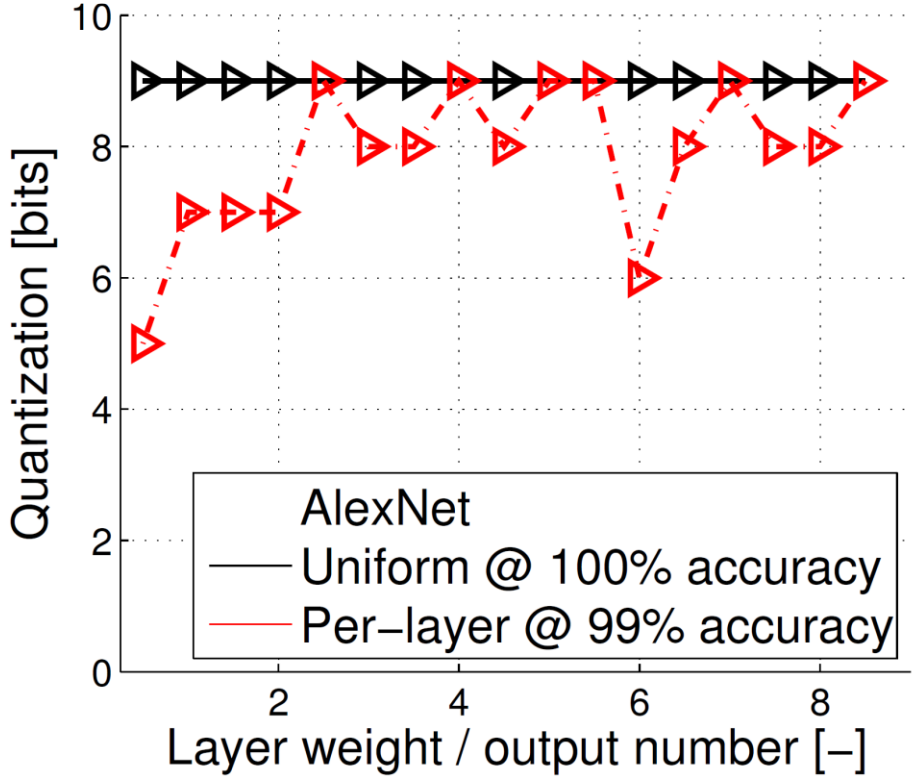
- Precision Varies from Layer to Layer

Network	Relative Accuracy			
	100%		99%	
	Per Layer Neuron Precision in Bits	Ideal Speedup	Per Layer Neuron Precision in Bits	Ideal Speedup
LeNet	3-3	5.33	2-3	7.33
Convnet	4-8-8	2.89	4-5-7	3.53
AlexNet	9-8-5-5-7	2.38	9-7-4-5-7	2.58
NiN	8-8-8-9-7-8-8-9-9-8-8-8	1.91	8-8-7-9-7-8-8-9-9-8-7-8	1.93
GoogLeNet	10-8-10-9-8-10-9-8-9-10-7	1.76	10-8-9-8-8-9-10-8-9-10-8	1.80
VGG_M	7-7-7-8-7	2.23	6-8-7-7-7	2.34
VGG_S	7-8-9-7-9	2.04	7-8-9-7-9	2.04
VGG_19	12-12-12-11-12-10-11-11-13-12-13-13-13-13-13-13	1.35	9-9-9-8-12-10-10-12-13-11-12-13-13-13-13-13	1.57

Judd, P., Albericio, J., Hetherington, T., Aamodt, T. M., & Moshovos, A. (2016, October). Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 1-12). IEEE.

Mixed Precision

- Precision Varies from Layer to Layer

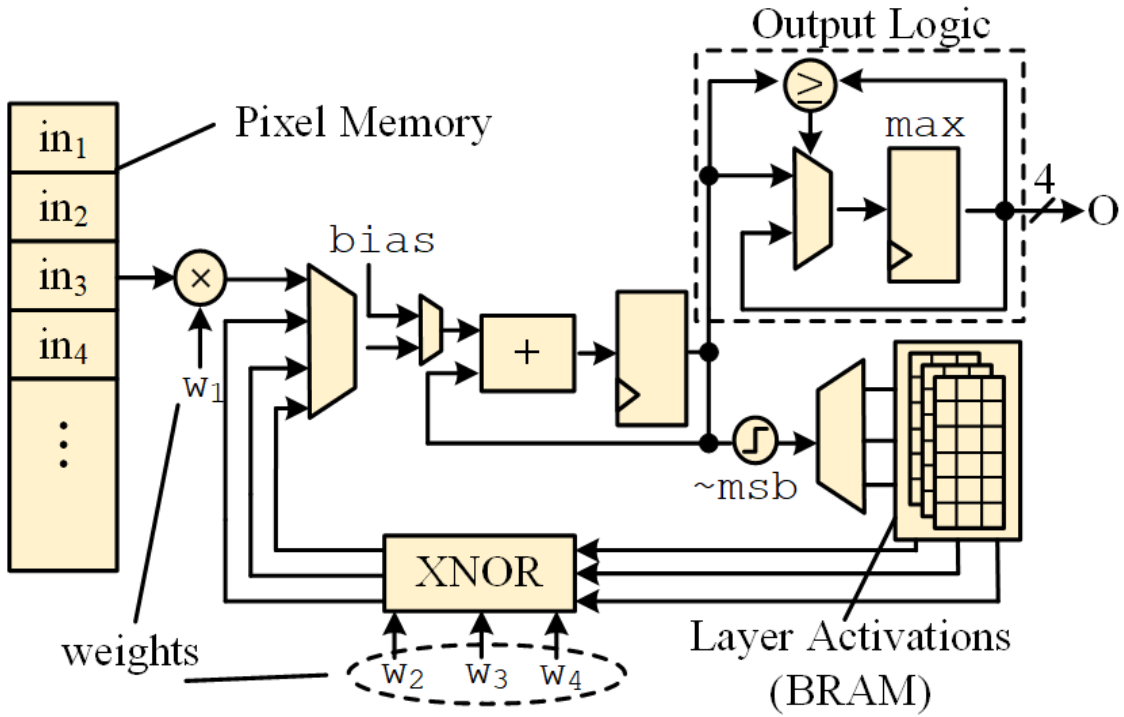


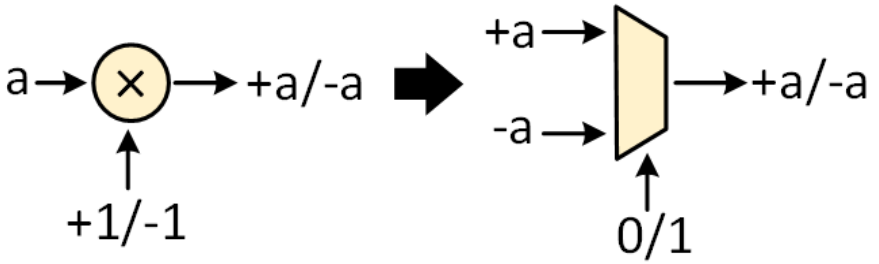
Moons, B., De Brabandere, B., Van Gool, L., & Verhelst, M. (2016, March). Energy-efficient convnets through approximate computing. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1-8). IEEE.

Outline

- Overview
- Basics of DNN Quantization
- Quantization Range Clipping
- Advanced Quantization
- Hardware Design

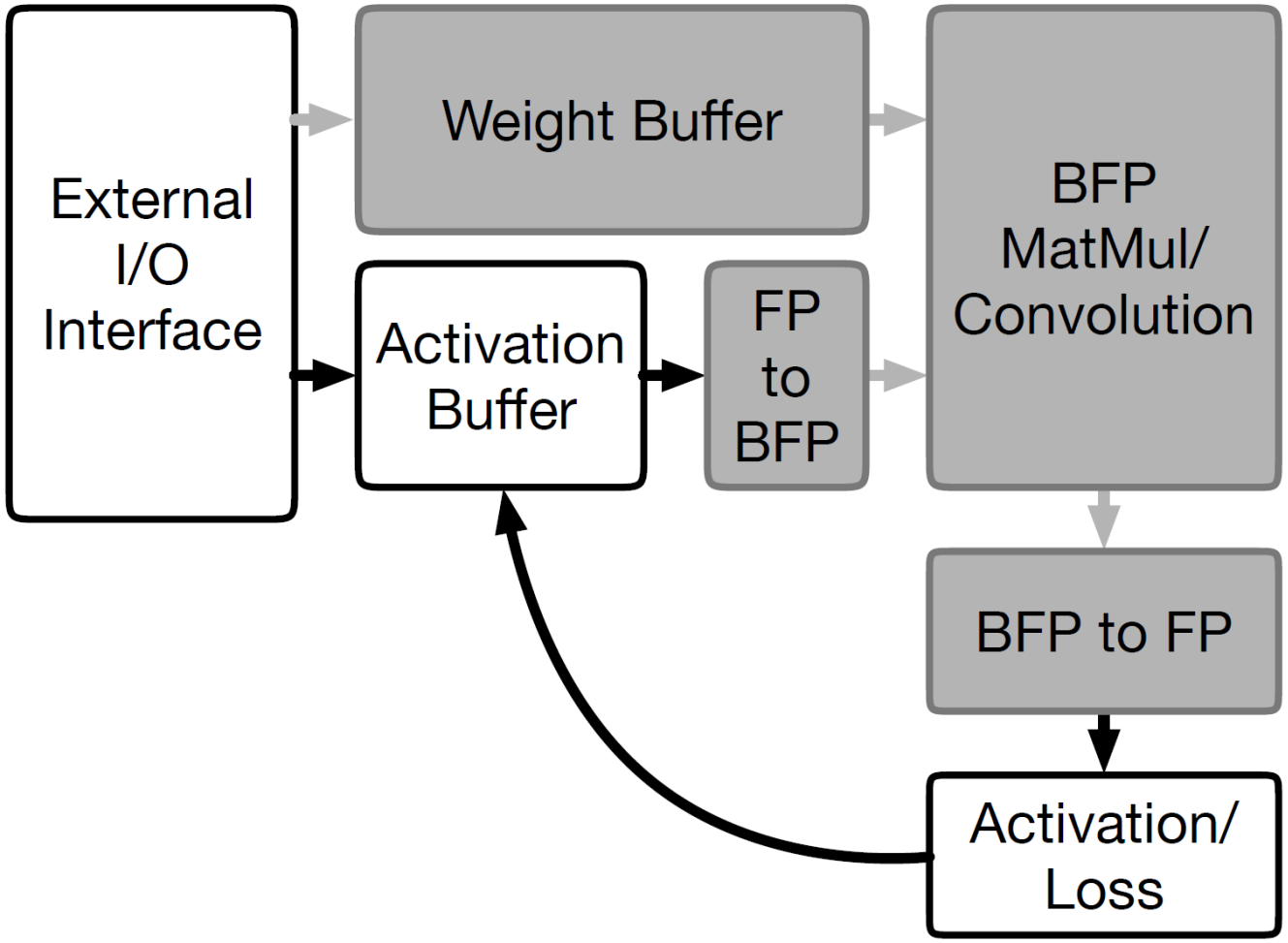
Hardware for XNOR Neural Networks





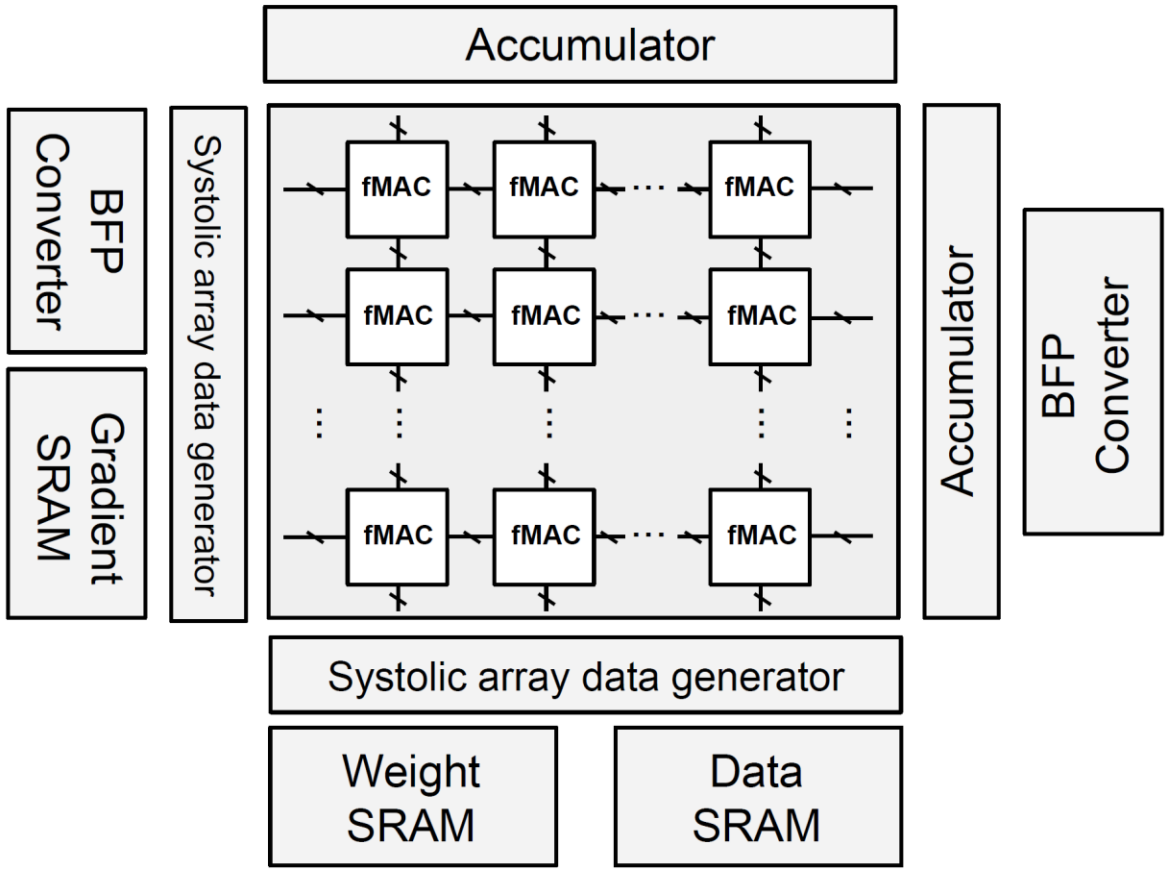
Dubey, A., Cammarota, R., & Aysu, A. (2020, November). BoMaNet: Boolean masking of an entire neural network. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (pp. 1-9). IEEE.

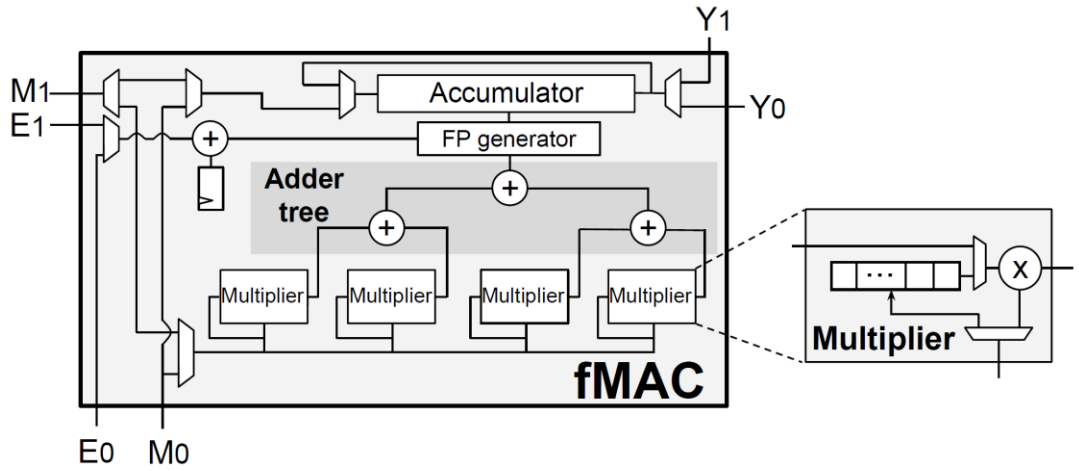
Hardware Prototype of BFP



Drumond, M., Lin, T., Jaggi, M., & Falsafi, B. (2018). Training dnns with hybrid block floating point. *Advances in Neural Information Processing Systems*, *31*.

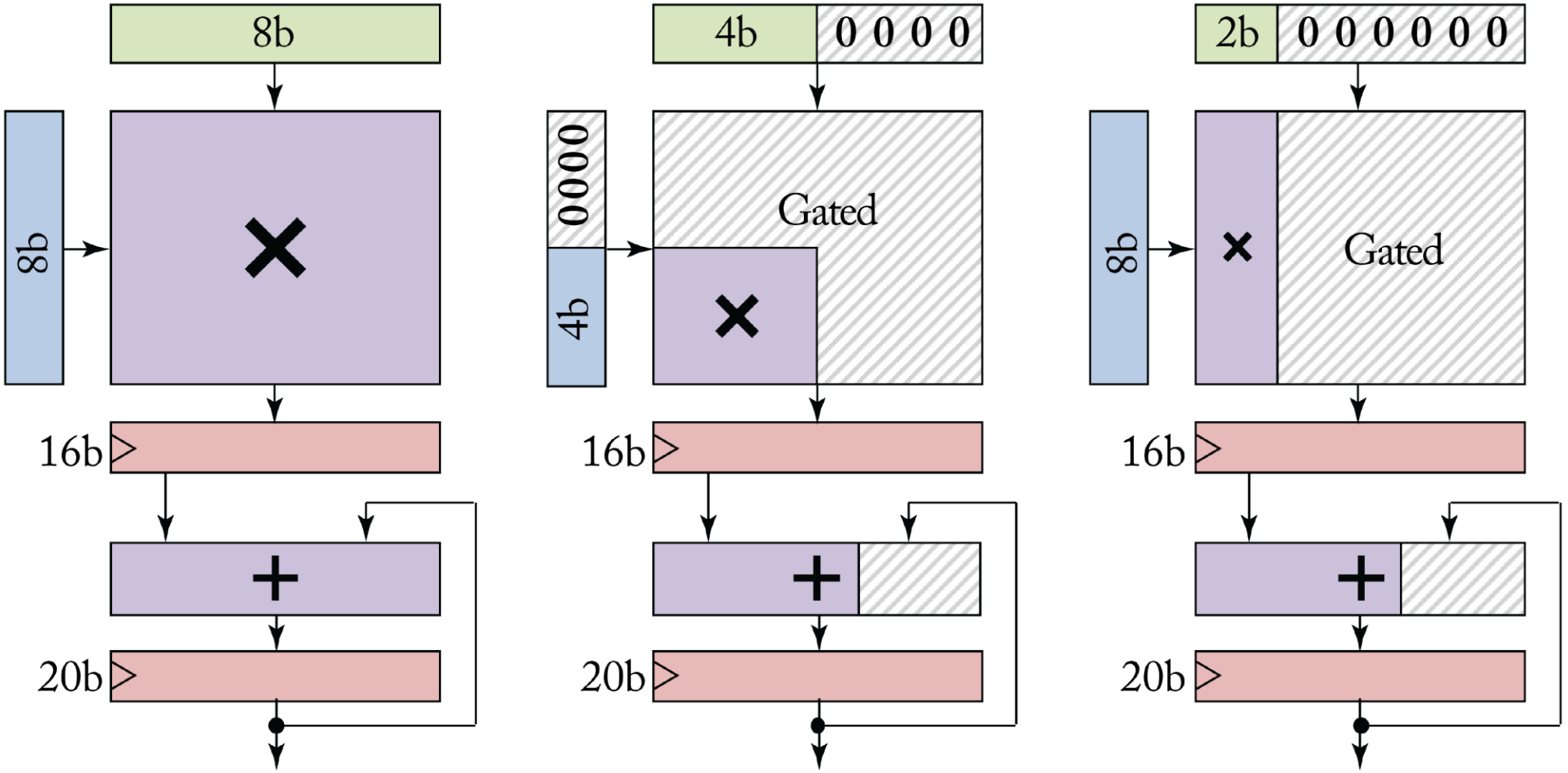
Systolic Fast MAC using BFP





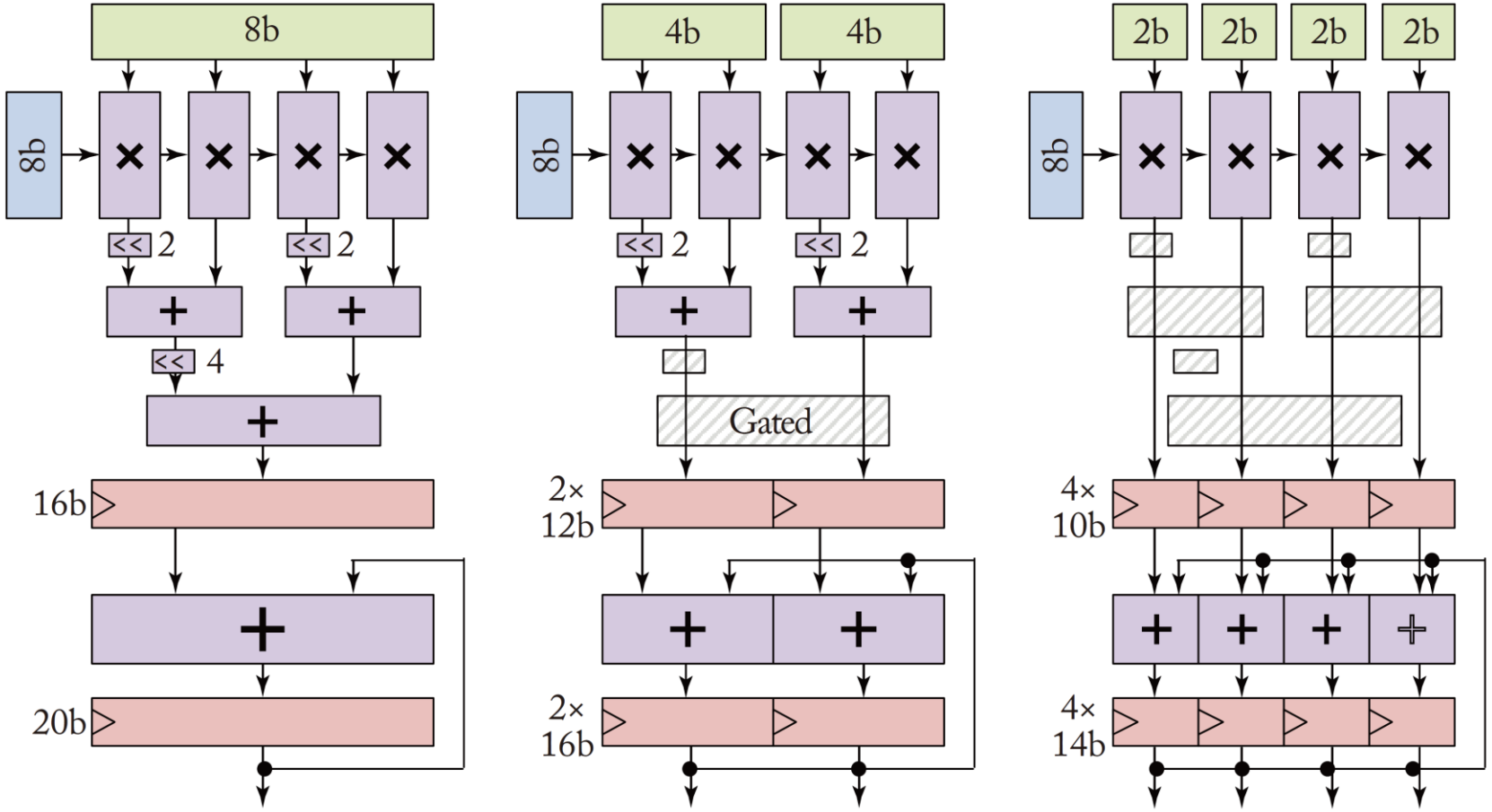
Zhang, S. Q., McDanel, B., & Kung, H. T. (2022, April). Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 846-860). IEEE.

Data-gated MAC



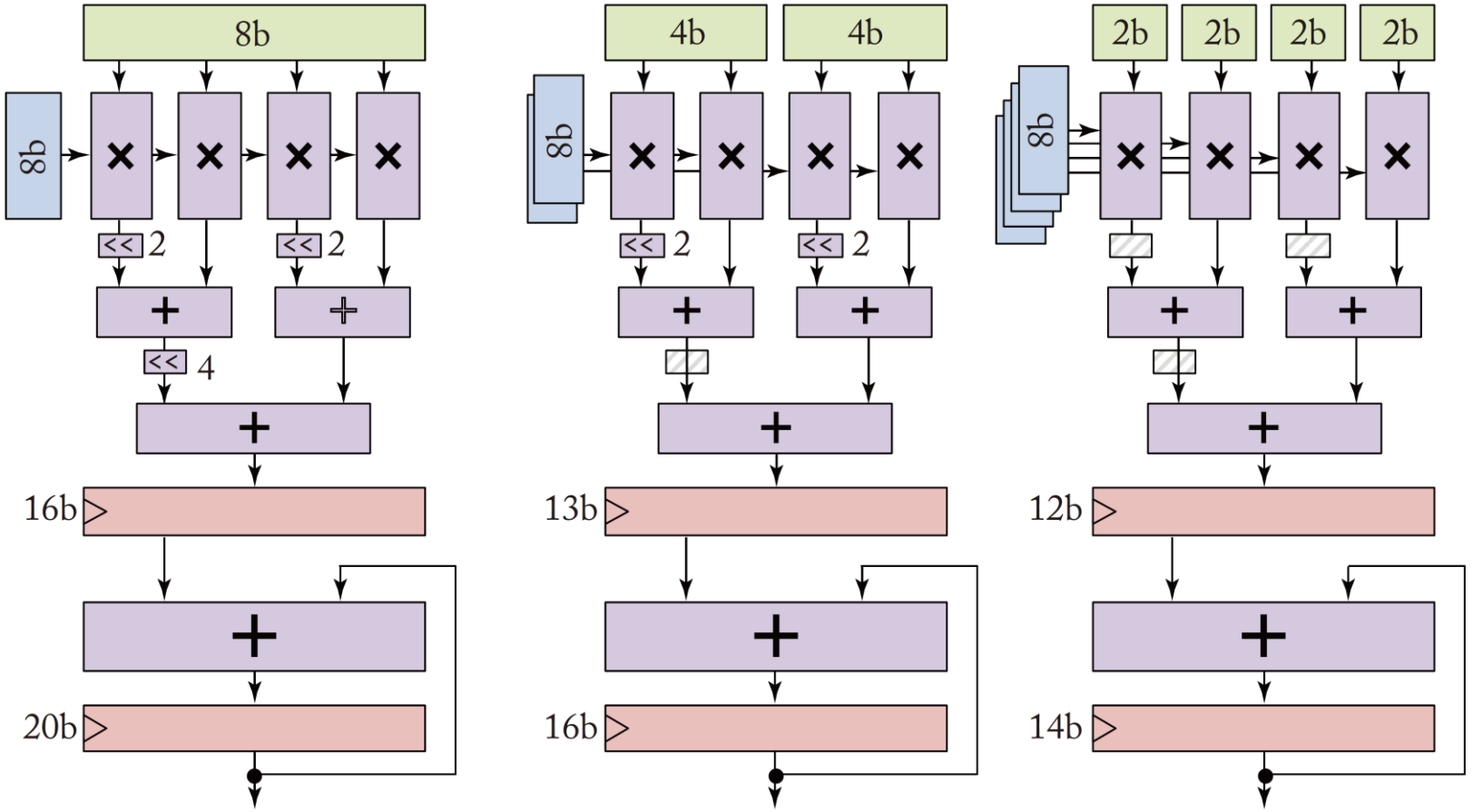
Spatial Precision-Scalable MACs

- Temporal accumulation of partial products

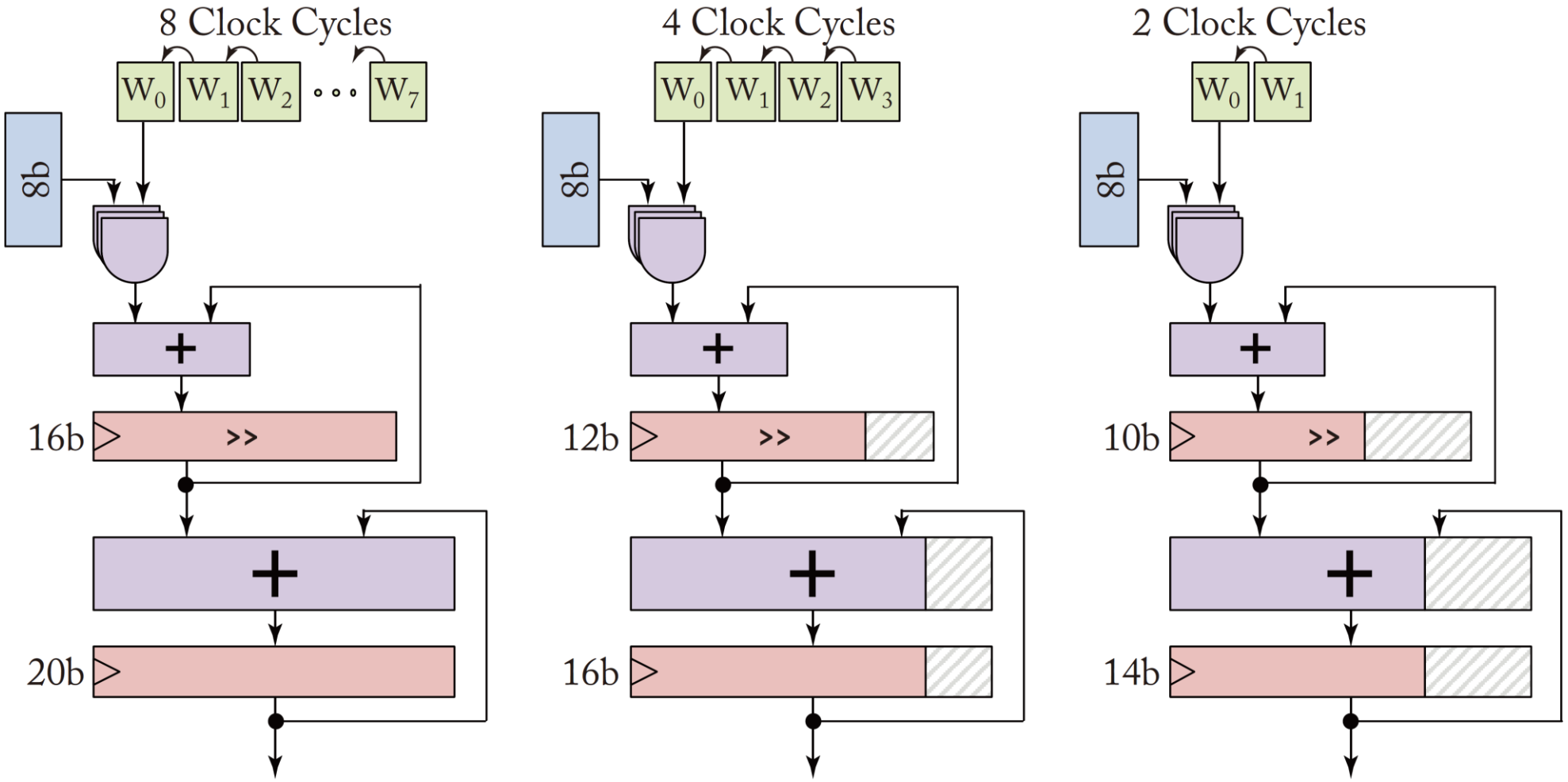


Spatial Precision-Scalable MACs

- Spatial accumulation of partial products

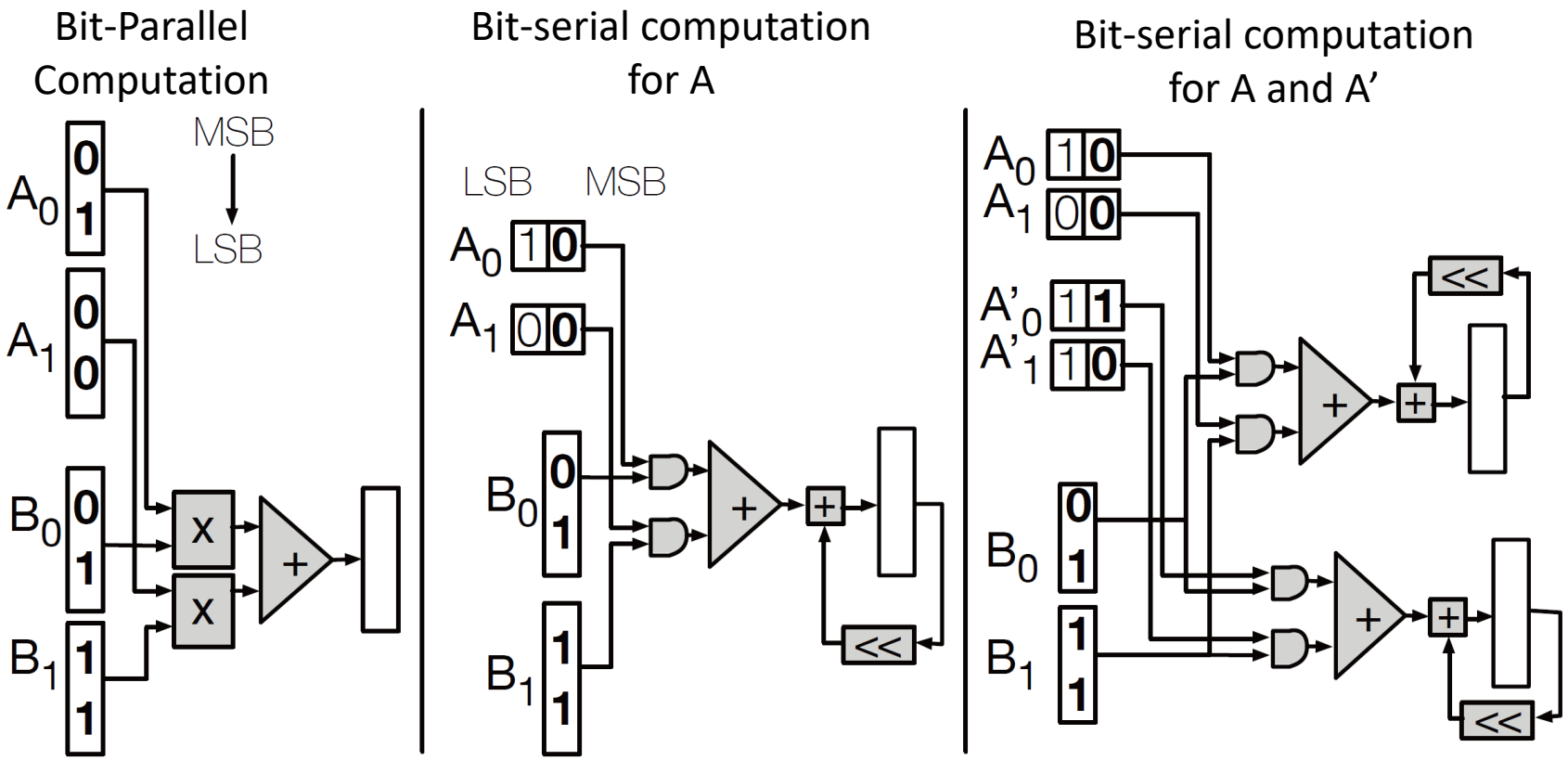


Temporal Precision-Scalable MACs



Bitwidth Scaling for Speed

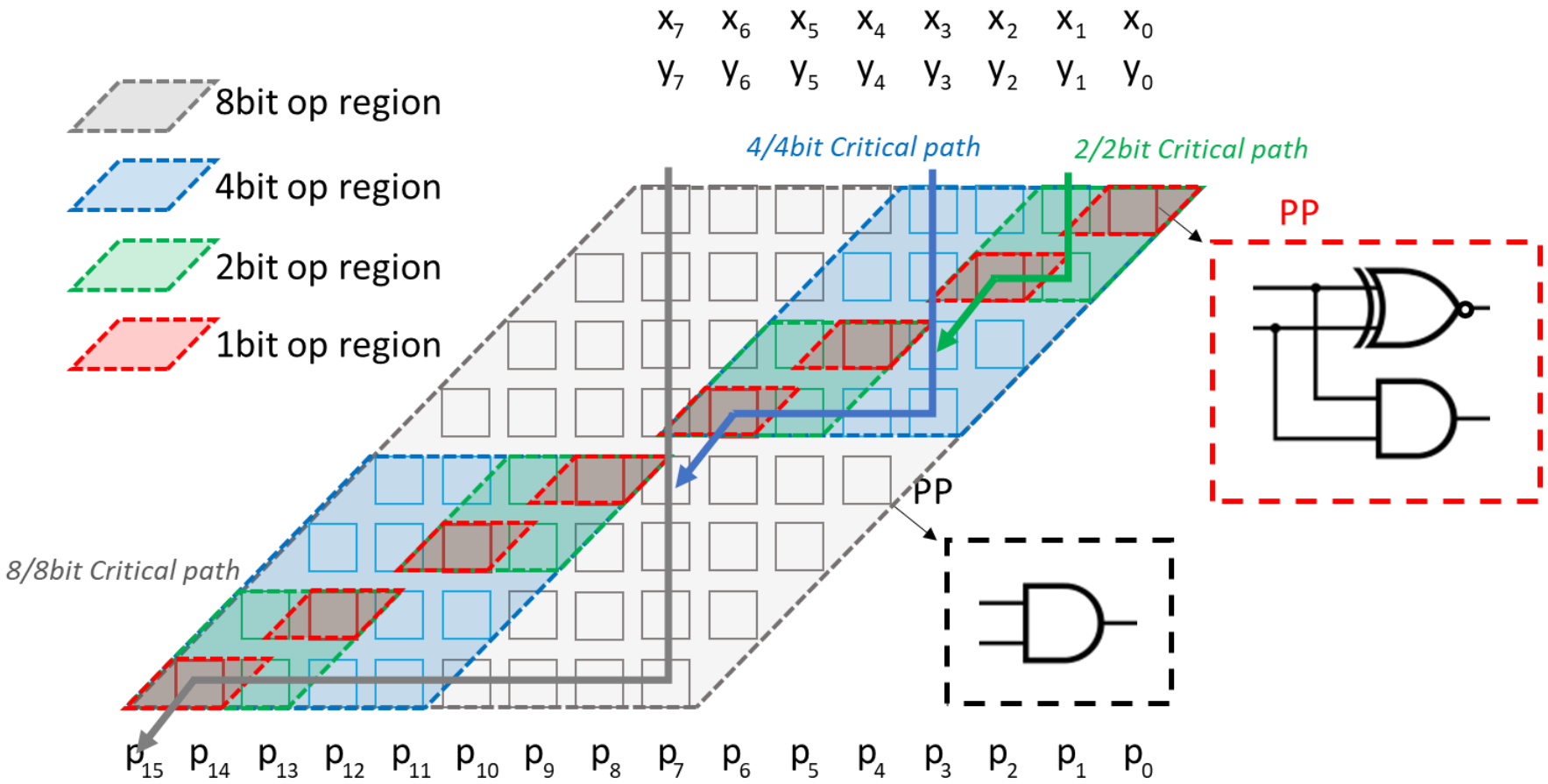
- Bit-Serial Processing
  - Reduce Bit-width → Skip Cycles Speed up of 2.24x vs. 16-bit fixed



Judd, P., Albericio, J., Hetherington, T., Aamodt, T. M., & Moshovos, A. (2016, October). Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 1-12). IEEE.

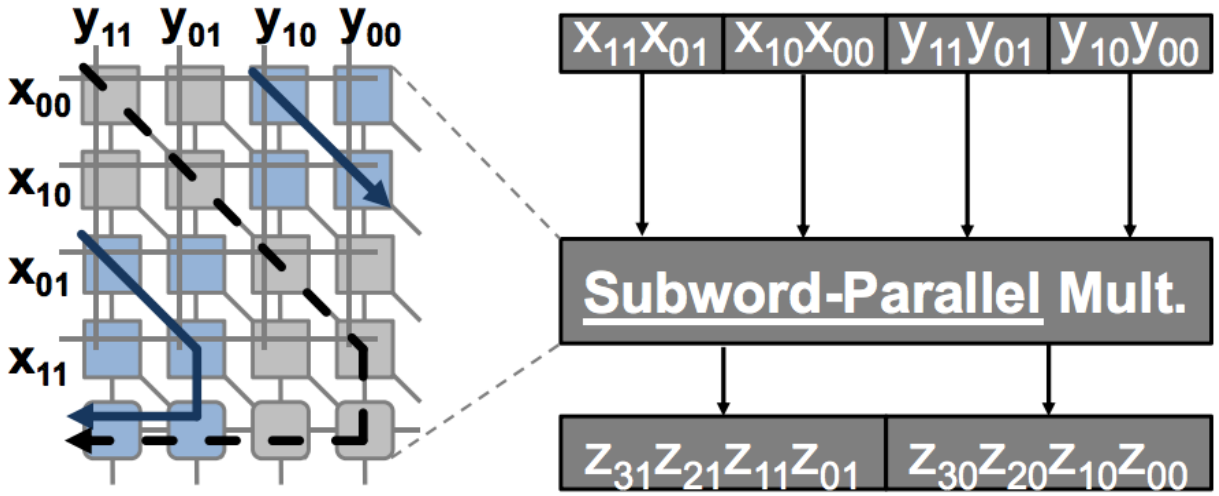
Bitwidth Scaling for Power

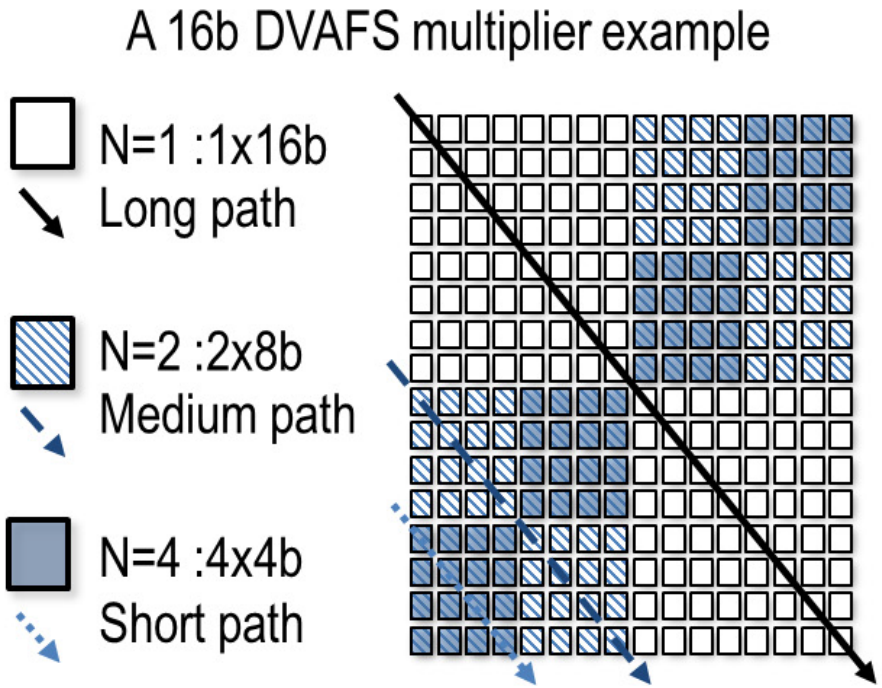
- Reduce Bit-width → Shorter Critical Path → Reduce Voltage



Reconfigure Spatial Multiply

- Configure 16bx16b multiplication into two 8x8b or four 4x4b (up to 256-64=192 adders are idle)

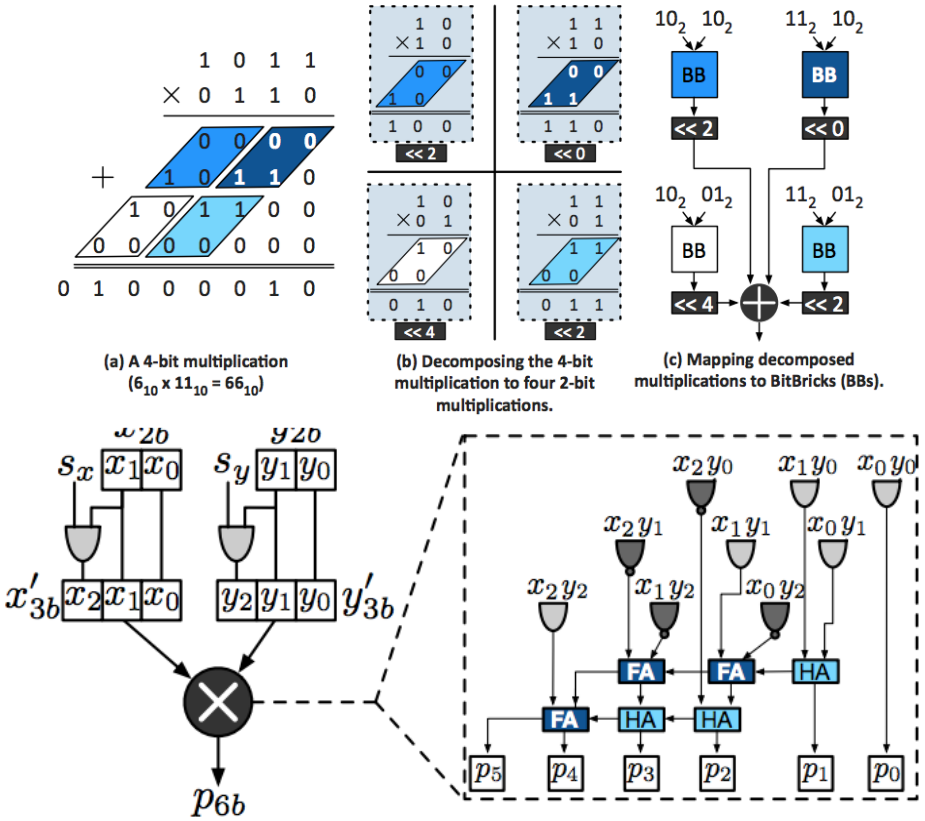


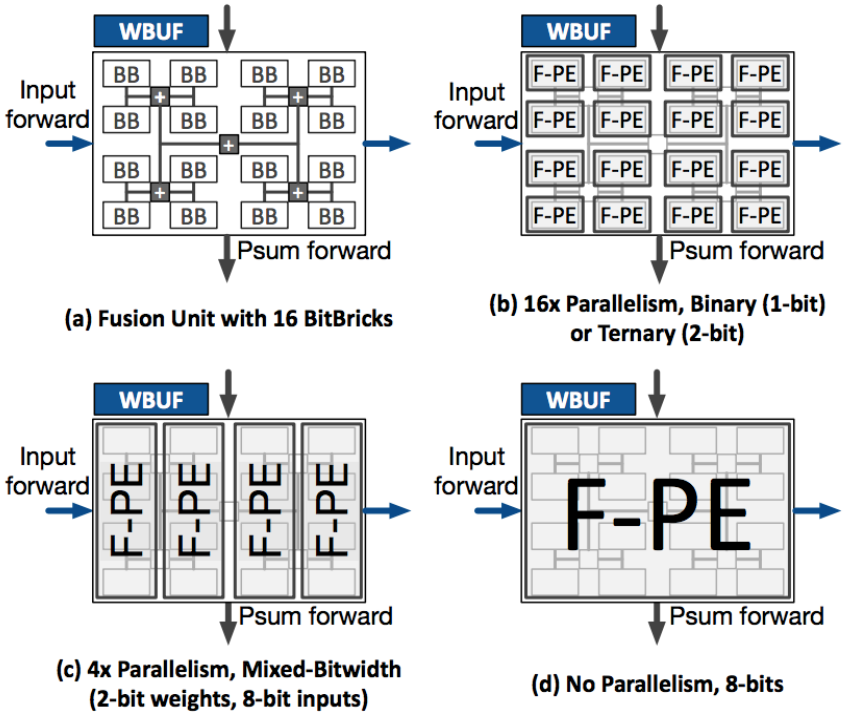


B. Moons, R. Uytterhoeven, W. Dehaene and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 246-247, doi: 10.1109/ISSCC.2017.7870353.

Reconfigure Spatial Multiply

- Build larger multipliers (Fused Unit) from small 2x2 multipliers with programmable shifters (BitBrick)

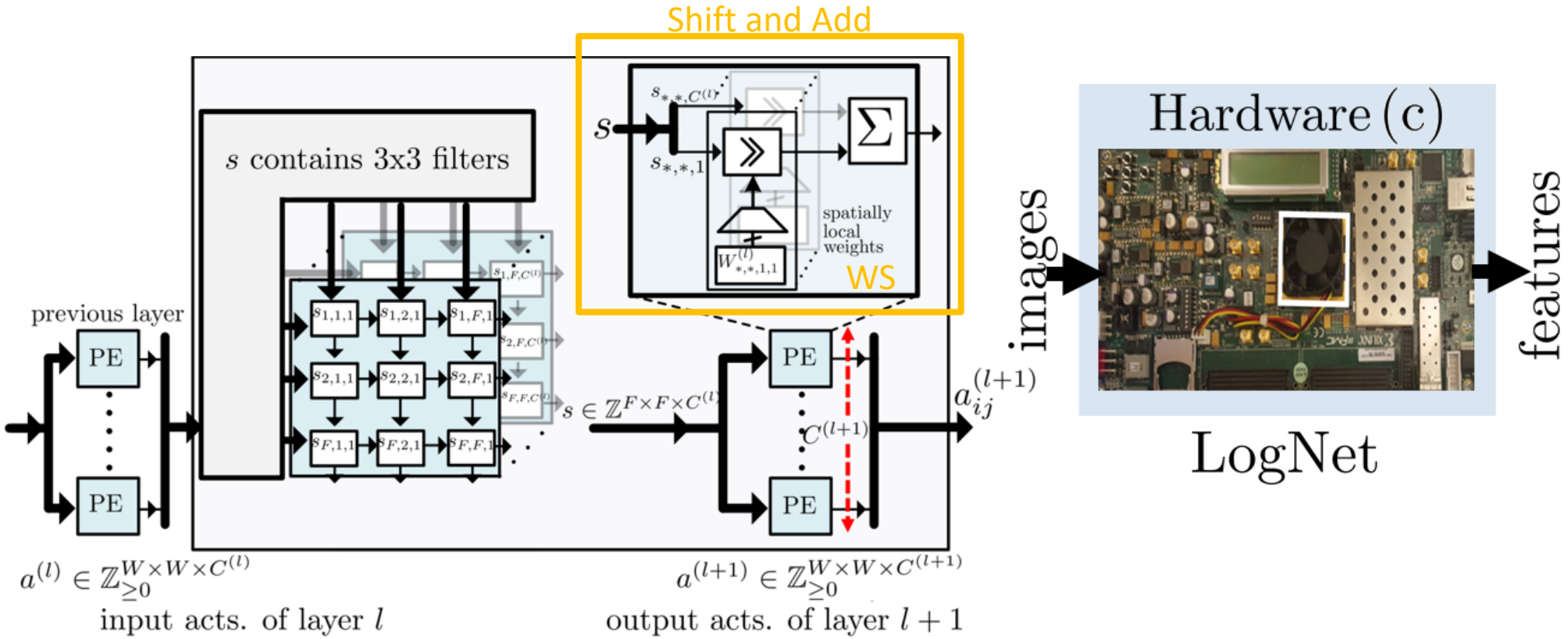




Sharma, H., Park, J., Suda, N., Lai, L., Chau, B., Chandra, V., & Esmaeilzadeh, H. (2018, June). Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)* (pp. 764-775). IEEE.

Log Domain Quantization Hardware

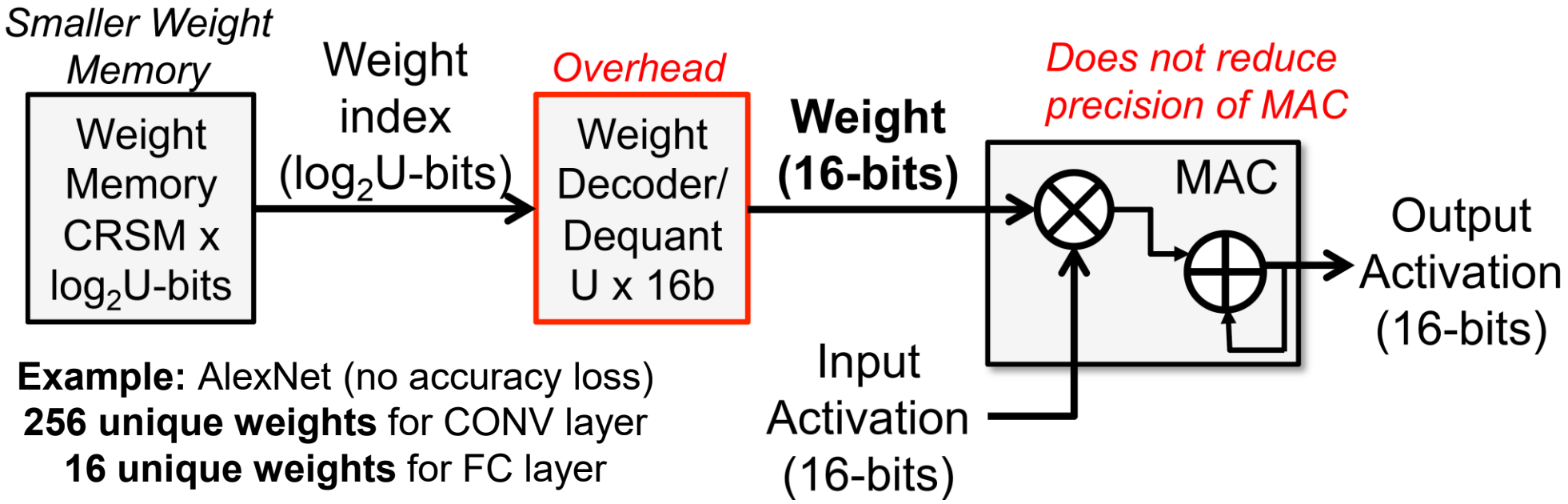
- Weights: 5-bits for CONV, 4-bit for FC; Activations: 4-bits
- Accuracy loss: 3.2% on AlexNet



Lee, E. H., Miyashita, D., Chai, E., Murmann, B., & Wong, S. S. (2017, March). Lognet: Energy-efficient neural networks using logarithmic computation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5900-5904). IEEE.

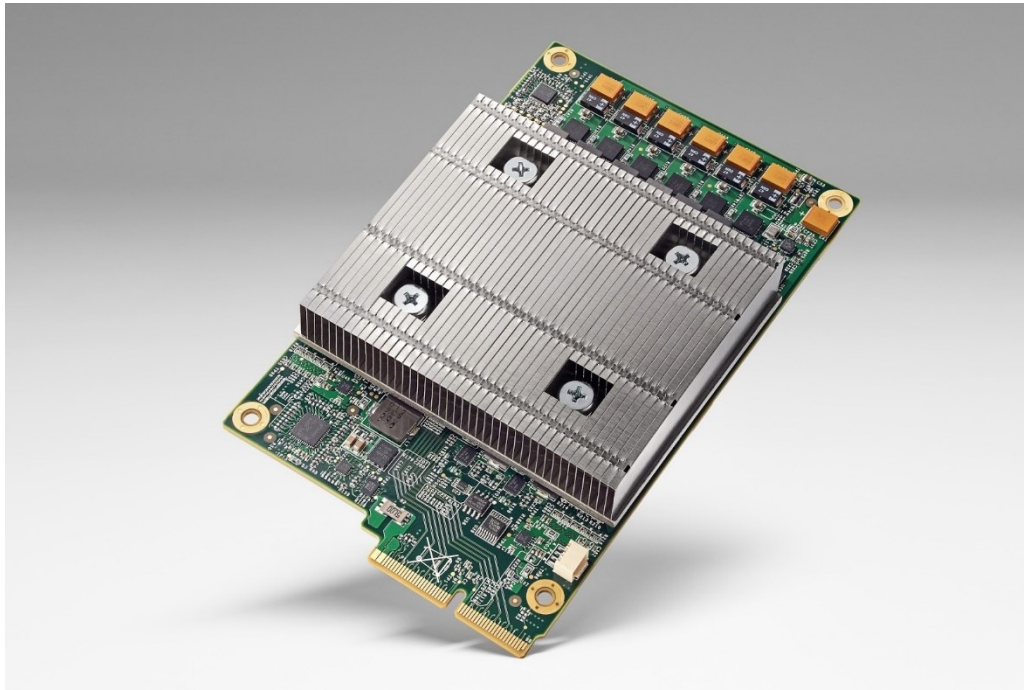
Non-Linear Quantization Table Lookup

- Trained Quantization: Find K weights via K-means clustering to reduce number of unique weights per layer (weight sharing)
- Smaller weight memory and second access from (small) table



Google's Tensor Processing Unit (TPU)

- " With its TPU Google has seemingly focused on delivering the data really quickly by cutting down on precision. Specifically, it doesn't rely on floating point precision like a GPU
  ….
  Instead the chip uses integer math…TPU used **8-bit integer**."
  - Next Platform (May 19, 2016)



NVIDIA Turing GPU

- “NVIDIA’s Turing architecture introduced **INT4** precision, which offers yet another speedup opportunity. In addition to computational speedups, using INT4 can also reduce a network’s memory footprint and conserve memory bandwidth, which makes it easier to run multiple ensemble networks on a single GPU.”
  - NVIDIA Technical Blog (Nov 06, 2019)