Computer Arithmetic for Machine Learning

Philip Leong (philip.leong@sydney.edu.au)

http://phwl.org/talks

THE UNIVERSITY OF SYDNEY

Computer Engineering Lab

Acknowledgment of Country

I acknowledge the Gadigal people of the Eora Nation as the traditional custodians of the land on which the university stands and pay respects to Elders past, present and emerging.

Australia vs Taiwan

- More than 200x larger than Taiwan
- Population about the same
  - 27.0M vs 23.4M in 2023

Computer Engineering Lab

Computer Engineering Lab

- CEL focuses on how to utilise parallelism to solve computing problems
    - Novel architectures, applications and design techniques

Motivation

- CPUs/GPUs designed to support datatypes of fixed wordlength
  - Double, float, long, short, char
- FPGA and ASICs can provide custom datapaths of arbitrary wordlength
- So how can we utilize low-precision for inference and training?

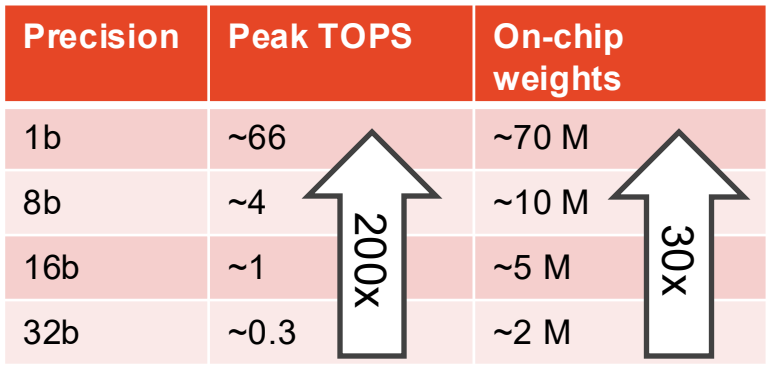


Figure: Xilinx

Overview

1. Number systems
2. Block Minifloat / Microscaling
3. Applications of BM

1. *Number systems*
2. *Block Minifloat / Microscaling*
3. *Applications*

THE UNIVERSITY OF SYDNEY

The University of Sydney

Number Systems – Unsigned

Unsigned integers are used to represent the nonnegative integers. An N-bit unsigned integer has a range $[0, 2^N - 1]$ and can be described in binary form, with u_i being the i'th binary digit:

$$U = (u_{N-1}u_{N-2}\ldots 0),\ u_i \in \{0, 1\}.$$

This represents the number

$$U = \sum_{i=0}^{N-1} u_i 2^i.$$

Number Systems – Two's Complement

$$X = (x_{N-1}x_{N-2}\ldots 0),\ x_i \in \{0, 1\}.$$

X has a range of $[-2^{N-1}, 2^{N-1} - 1]$ and represents

$$X = -x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i$$

Number Systems – Two's Complement Fractions

The most significant $N-F$ bits of the number represent the integer part and the remaining F bits are the fractional part of the number

$$Y = (\overbrace{a_{N-1} \ldots a_F}^{\text{integer}} \overbrace{a_{F-1} \ldots a_0}^{\text{fraction}}).$$

This corresponds to a scaling of the two's complement integer representation by the factor $S = 2^{-F}$ and the two's complement fraction number Y represents

$$Y = 2^{-F} \times (-x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i)$$

Note that the two's complement fraction $(N,0)_{\mathcal{I}}$ corresponds to the two's complement integer case and $(N,N)_{\mathcal{I}}$ has a range of $[-1,1)$.

Arithmetic Operations on Two's Complement Fractions

- If we wish to perform arithmetic on two (N,F) format 2's complement fractions
- Addition and subtraction
  - Normal addition
- Multiplication
  - An (N,F) multiplication gives a (2N, 2F) result so you need to do an arithmetic right shift by F bits from the 2N multiplier output
  - E.g. for (4,3) 0.75*0.75 = 0.110*0.110=00.100100 >> 3=0.100=0.5

Number Systems – Floating Point (1)

$$Z = (\overbrace{a_0}^{\text{A}} \overbrace{b_{J-1} \ldots b_0}^{\text{B}} \overbrace{c_{F-1} \ldots b_0}^{\text{C}}).$$

A represents the sign S where

$$S = \begin{cases} +1 & \text{if } a_0 = 0 \\ -1 & \text{if } a_0 = 1 \end{cases}$$

The unsigned integers B and C are encoded representations of the exponent and mantissa respectively. The exponent E, is stored in a biased representation with $E = B - (2^{J-1} - 1)$. For normalized numbers, $B \neq 0$ and the significand is represented by $M = 1 + C \times 2^{-F}$. This is a two's complement fraction $(F+1, F)_{\mathcal{I}}$ with the most significant bit being implicitly set to 1. If $B = 0$, it is called a denormalized number, and there is no implicit 1 in the $(F, F)_{\mathcal{I}}$ fraction.

Number Systems – Floating Point (2)

- Standard is IEEE 754-2019 https://standards.ieee.org/ieee/754/6210/

$$Z = \begin{cases} S \times 2^E \times M & \text{if } (0 < B < 2^J - 1) \\ S \times 2^E \times (M - 1) & \text{if } (B = 0) \\ S \times \infty & \text{if } (B = 2^J - 1 \text{ and } C = 0) \\ NaN & \text{if } B = 2^J - 1 \text{ and } C \neq 0). \end{cases}$$

Normalised

Subnormal (or Denormal)

Infinity

NaN

Number systems – Logarithmic Number System

The logarithmic number system (LNS) is a special case of floating point in which the mantissa is always 1 (i.e. only the sign and exponent fields are used). It has the advantages of simplified implementation at the expense of reduced precision. For an N bit LNS number, $(N, F)_{\mathcal{L}}$, the most significant bit is a zero flag, Z. Z is zero if the number is zero (since there is no log of zero), otherwise set. The next most significant bit is used for a sign bit and the rest of the number is the base 2 logarithm of the magnitude of the number to be represented in $(N-2, F)_{\mathcal{I}}$ two's complement fraction format. If E is the value of this two's complement fraction and S is defined as for floating point, then

$$L = \begin{cases} 0 & \text{if } Z = 0 \\ L = S \times 2^{E} & \text{if } Z = 1 \end{cases}$$

IEEE 754 Rounding [1]

- IEEE 754 has 4 rounding modes
  - Round down
  - Round up
  - Round towards zero
  - Round to nearest (this is what everyone means when they don’t specify)
    - If tie then the one with LSB=0 is chosen

Theorem 5.1 (Error Bound)

Theorem 5.1 *Let x be any real number in the normalized range of a binary floating point system with precision p. Then*

$$\text{round}(x) = x(1+\delta)$$

for some δ satisfying

$$|\delta| < \epsilon,$$

where ϵ, machine epsilon, is the gap between 1 and the next larger floating point number, i.e.,

$$\epsilon = 2^{-(p-1)}.$$

Furthermore, if the rounding mode in effect is round to nearest,

$$|\delta| < \frac{1}{2}\epsilon = 2^{-p}.$$

Rounding Error of Operators

- For $\oplus,\ominus,\otimes,\oslash$ operators (op), then

$$x \oplus y = \text{round}(x + y),$$

$$x \ominus y = \text{round}(x - y),$$

$$x \otimes y = \text{round}(x \times y),$$

$$x \oslash y = \text{round}(x / y),$$

i.e. IEEE computed result **must be the rounded value of the exact result**

- For x, y normal numbers, (x op y) error has same bounds as Theorem 5.1

Addition/Subtraction

- Let's consider adding $x = m \times 2^E$ and $y = p \times 2^F$
  1. Make E>=F by optionally swapping x and y
  2. If E>F align significands by shifting p to the right E-F positions;
  3. Add significands
  4. Correctly round result

- Example 3+2

$$
\begin{aligned}
 & (\ 1.10000000000000000000000\ )_2 \times 2^1 \\
+ & (\ 1.00000000000000000000000\ )_2 \times 2^1 \\
= & (10.10000000000000000000000\ )_2 \times 2^1 \\
\textbf{Normalize}: & (\ 1.01000000000000000000000\ )_2 \times 2^2
\end{aligned}
$$

Additional Bits for Rounding

Now consider adding 3 to 3×2^{-23}. We get

$$
\begin{array}{rll}
 & (\ 1.10000000000000000000000 &)_2 \times 2^1 \\
+ & (\ 0.00000000000000000000001|1 &)_2 \times 2^1 \\
= & (\ 1.10000000000000000000001|1 &)_2 \times 2^1 \\
\text{Round Down}: & (\ 1.10000000000000000000001 &)_2 \times 2^1 \\
\text{or Round Up}: & (\ 1.10000000000000000000010 &)_2 \times 2^1
\end{array}
$$

The "|" indicates bits beyond the LSB

An additional bit called the Guard Bit is needed to get correctly rounded result

How Many Additional Bits do we Need?

$$
\begin{array}{rll}
 & (\ 1.00000000000000000000000 &)_2 \times 2^0 \\
- & (\ 0.00000000000000000000000|01000000000000000000001 &)_2 \times 2^0 \\
= & (\ 0.11111111111111111111111|10111111111111111111111 &)_2 \times 2^0 \\
\text{Normalize :} & (\ 1.11111111111111111111111|01111111111111111111110 &)_2 \times 2^{-1} \\
\text{Round to} & & \\
\text{Nearest :} & (\ 1.11111111111111111111111 &)_2 \times 2^{-1}.
\end{array}
$$

1-24 guard bits will get wrong result (need 25 in this case!)

- If the additional bits are
  - 10…0 then exactly half way
  - 0x…x less than half way (round down)
  - Anything else = more than half way (round up)

Sticky Bit

- To determine which of the 3 cases keep Guard, Round and Sticky Bit (GRS)
  - Sticky bit informs whether there is a 1 at or beyond that position

$$
\begin{aligned}
 & (\ 1.00000000000000000000000| \ )_2 \times 2^0 \\
- & (\ 0.00000000000000000000000|011 \ )_2 \times 2^0 \\
= & (\ 0.11111111111111111111111|101 \ )_2 \times 2^0 \\
\text{Normalize}: & (\ 1.11111111111111111111111|01 \ )_2 \times 2^{-1} \\
\text{Round to Nearest}: & (\ 1.11111111111111111111111 \ )_2 \times 2^{-1}.
\end{aligned}
$$

Multiplication

- Let's consider multiplying $\mathrm{x} = m \times 2^E$ and $\mathrm{y} = p \times 2^F$
    1. Multiply the significands
    2. Add the exponents
    3. Noramlise and correctly round the result

Catastrophic Cancellation (1)

- Catastrophic cancellation can occur when subtracting similar values
- Consider $\hat{x} = \hat{a} - \hat{b}$ where $\hat{a} = a(1 + \delta_a)$ and $\hat{b} = b(1 + \delta_b)$

$$
\begin{aligned}
\left|\frac{x - \hat{x}}{x}\right| &\leq \left|\frac{(a - b) - (\hat{a} - \hat{b})}{a - b}\right| \\
&\leq \left|\frac{[a - a(1 + \delta_a)] - [b - b(1 + \delta_b)]}{a - b}\right| \\
&\leq \left|\frac{-a\delta_a + b\delta_b}{a - b}\right| \\
&\leq \max(|\delta_a|, |\delta_b|)\frac{|a| + |b|}{|a - b|}
\end{aligned}
$$

- Relative error is highest when $|a - b| \ll |a| + |b|$

Catastrophic Cancellation (2)

- Catastrophic cancellation can occur when subtracting similar values
- Consider $\hat{x} = \hat{a} - \hat{b}$ where $\hat{a} = a(1 + \delta_a)$ and $\hat{b} = b(1 + \delta_b)$

$$
\begin{aligned}
\left|\frac{x - \hat{x}}{x}\right| &\leq \left|\frac{(a - b) - (\hat{a} - \hat{b})}{a - b}\right| \\
&\leq \left|\frac{[a - a(1 + \delta_a)] - [b - b(1 + \delta_b)]}{a - b}\right| \\
&\leq \left|\frac{-a\delta_a + b\delta_b}{a - b}\right| \\
&\leq \max(|\delta_a|, |\delta_b|)\frac{|a| + |b|}{|a - b|}
\end{aligned}
$$

- Relative error is highest when $|a - b| \ll |a| + |b|$

Other Fun Facts about Floating Point Numbers

- (a+b)+c ≠ a+(b+c)
- There are representations for +/- 0
- There are 5 types of exceptions: *invalid operation, division by zero, overflow, underflow, and inexact*
- Required arithmetic operations (add, subtract, multiply, divide, square root, fused multiply-add, remainder, minimum, maximum)

```
#include <stdio.h>
int main()
{
        int cnt = 0;
        for (double t=0.0; t < 1.0; t += 0.1)
                printf("%d %f\n", cnt++, t);
}
```

Lower Precision: Single Precision and Bfloat16

(a) fp32: Single-precision IEEE Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

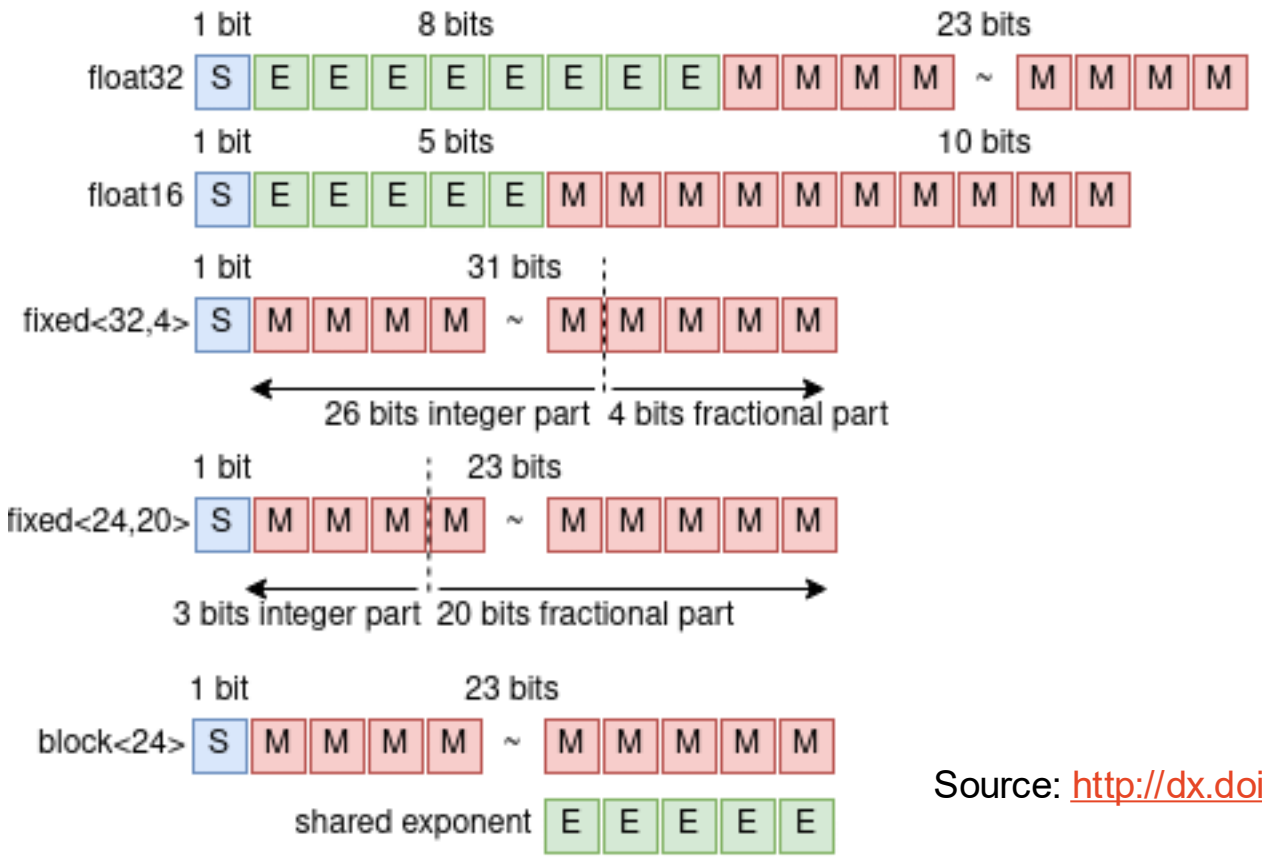
(b) fp16: Half-precision IEEE Floating Point Format

Range: ~5.96e^{-8} to 65504

(c) bfloat16: Brain Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

Formats used for ML including Block Floating Point



Source: http://dx.doi.org/10.48550/arXiv.2107.13490

References (available at https://phwl.org/assets/papers/papers)

1. Michael Overton, “Numerical Computing with IEEE Floating Point Arithmetic”, SIAM, 2001 https://doi.org/10.1137/1.9780898718072.fm

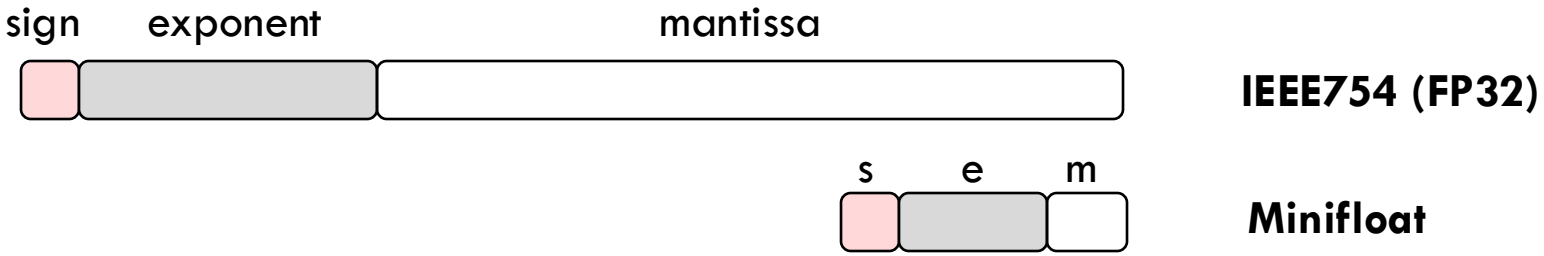
1. *Number systems*
2. *Block Minifloat / Microscaling*
3. *Applications*

THE UNIVERSITY OF SYDNEY

The University of Sydney

Minifloat circa 2021 [1]

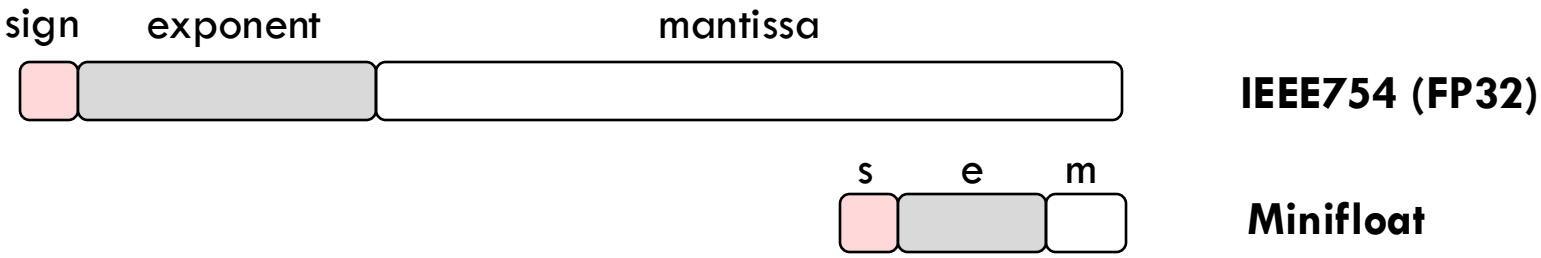
- Narrow floating-point representation
  - Our range between 4-8 bits
  - NaN/Infinity NOT supported



- Pros:
  - Memory (fewer bits)
  - Smaller hardware
- Cons:
  - Dynamic Range (exponent bits)

Block Minifloat (also known as Microscaling or MX)

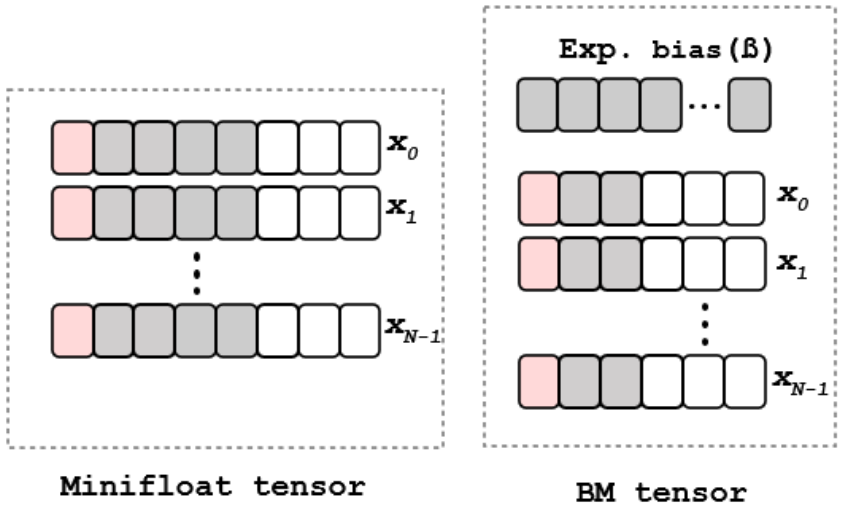
- Narrow floating-point representation
  - Our range between 4-8 bits
  - NaN/Infinity NOT supported

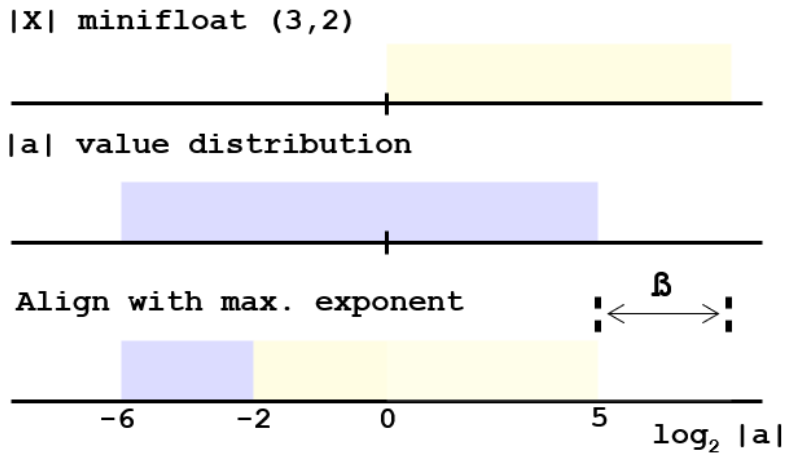


- Pros:
  - Memory (fewer bits)
  - Smaller hardware
- Cons:
  - Dynamic Range (exponent bits)

Block Minifloat

- Share exponent bias across **blocks** of NxN minifloat numbers





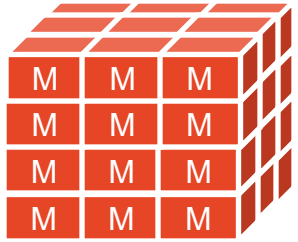
- Dynamic range (with fewer bits)
- Denser dot-products in hardware

- Align wtih **max** exponent
- Underflow is tolerated

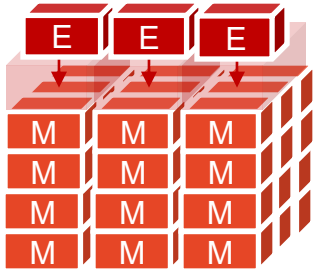
Block Minifloat (BM)

- BM = minifloat + shared exponent bias β

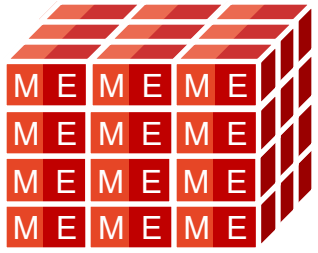
› minifloat: small floating-point format (same idea as Microscaling MX)



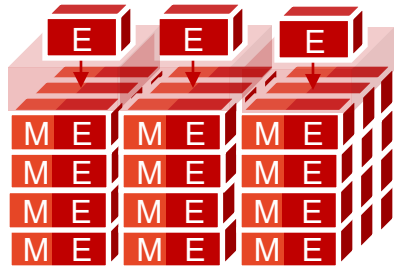
Fixed



BFP



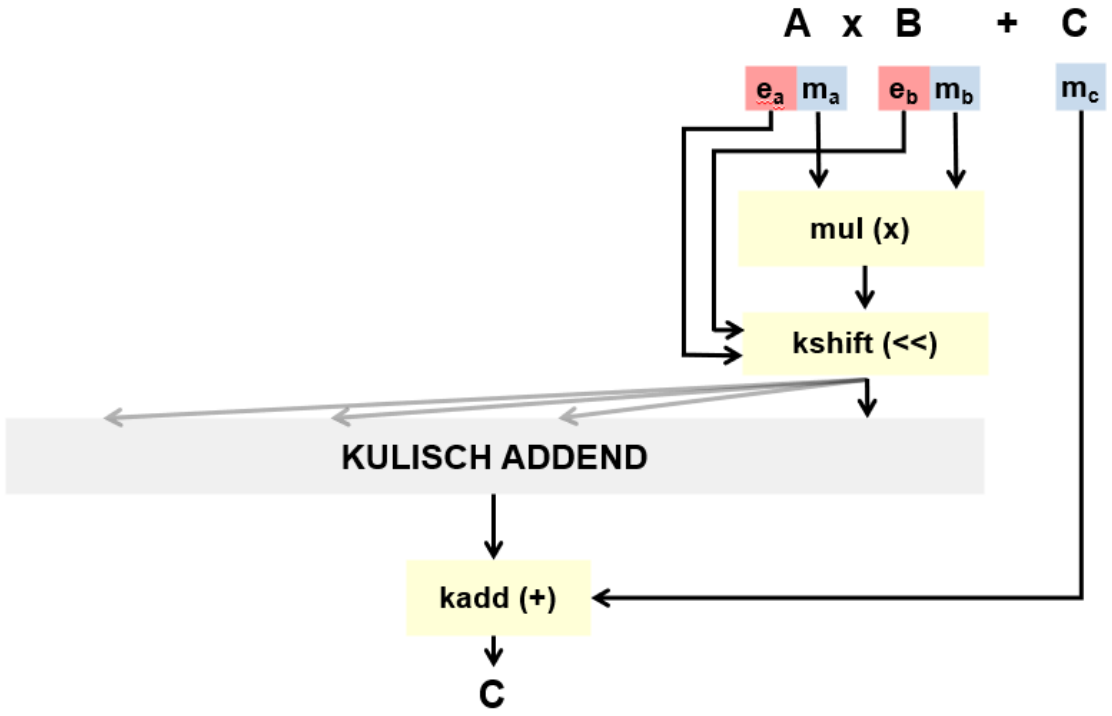
Minifloat



Block Minifloat and Microscaling (MX)

Fused Multiply-Add with Kulisch Accumulation

- **Kulisch Accumulator:** Fixed point accumulator wide enough to compute error-free sum of floating-point products
- Integer-like hardware complexity for **exponent <=4 bits**

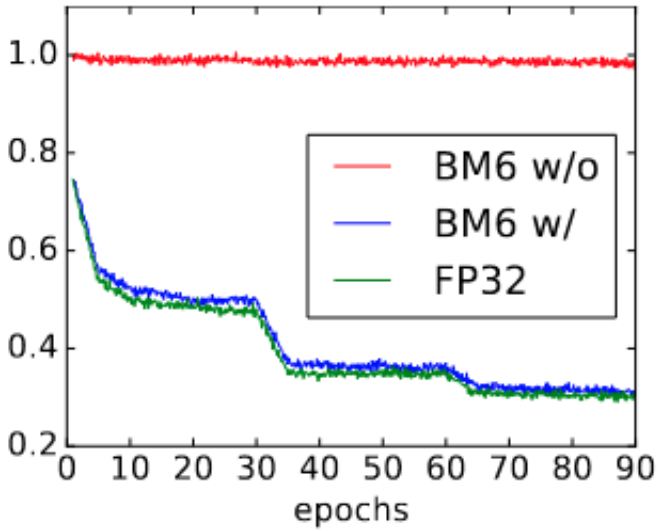


Implementation Details

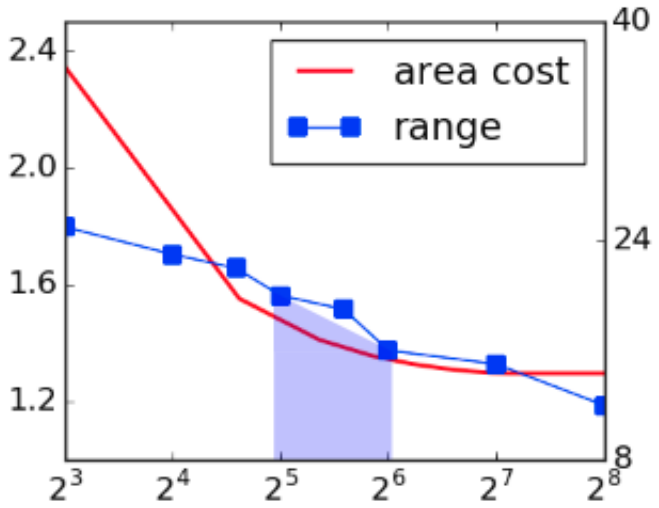
- Three techniques to reduce data loss:
  - Gradual underflow, Block Design, Hybrid Formats
- Simulate specialized BM hardware on GPU (with FP32)
  - Apply Block Minifloat to all weights, acts, grads
- Our Spectrum of Block Minifloats

BM8 (ours)	(2,5)/(4,3)
BM7 (ours)	(2,4)/(4,2)
BM6 (ours)	(2,3)/(3,2)
BM5 (ours)	(2,2)/(3,1)
BM5-log (ours)	(4,0)/(4,0)
BM4 (ours)	(2,1)/(3,0)
BM4-log (ours)	(3,0)/(3,0)

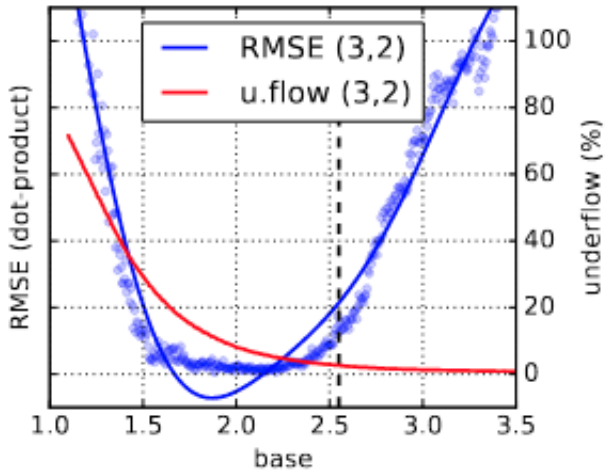
Data Loss Experiments



(a) Validation Accuracy: Training with denormal numbers on ImageNet

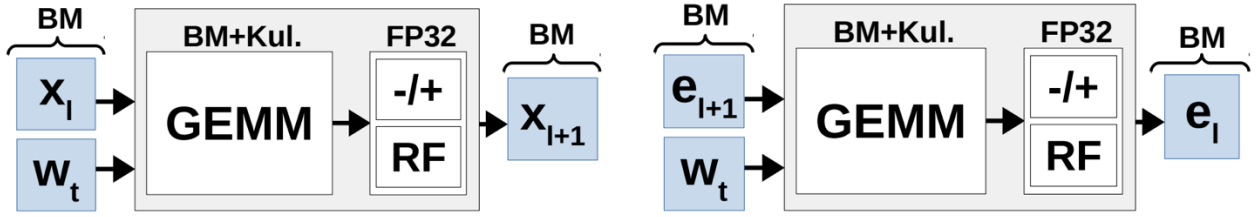


(b) HW (left axis) vs Range (right axis): Selecting the block size

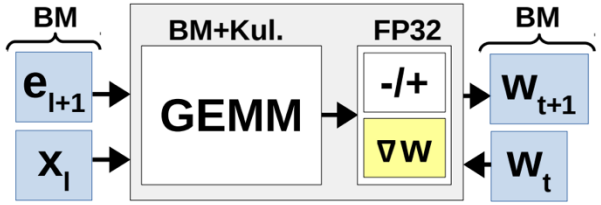


(c) Minifloat scaling by varying the exponent base

GPU Acceleration with BM



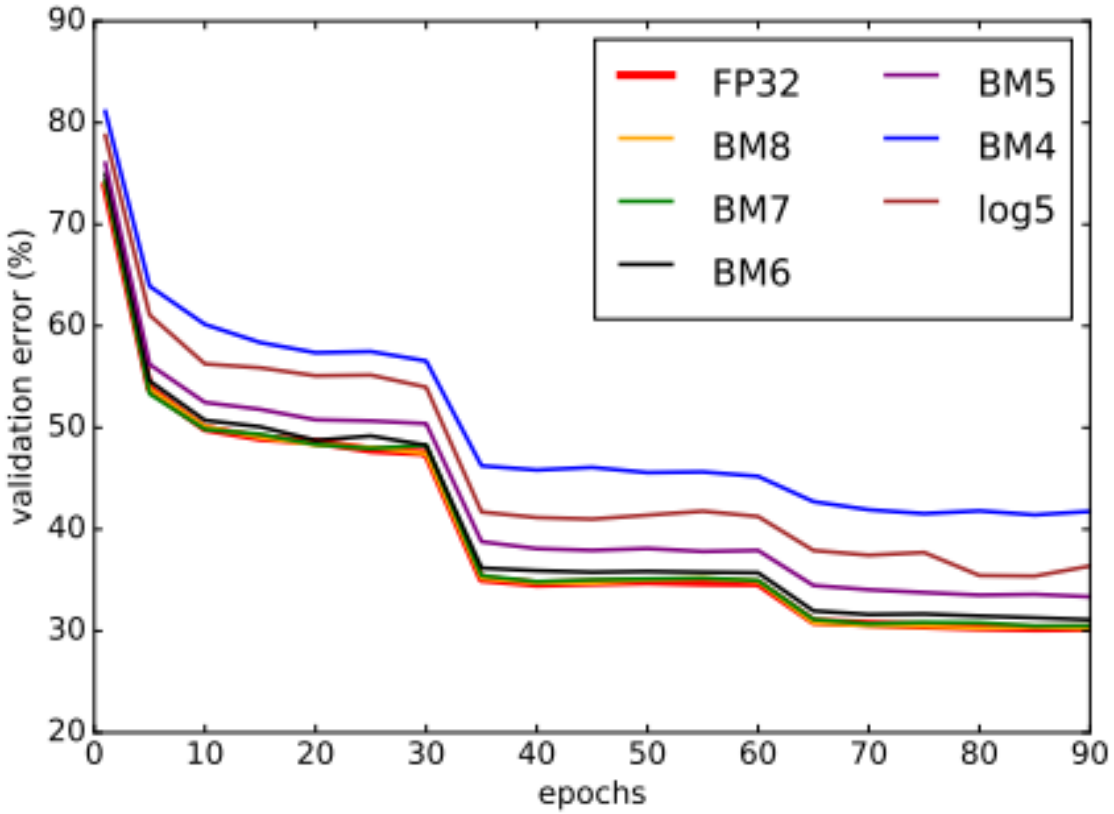
(a) Fwd activation (b) Bwd activation grad.



(c) Bwd weight grad. and update

- Weight, activation and gradient tensors quantized to BM with stochastic rounding
- Kulisch accumulator ensures our dot products are exact (can use FP CUDA lib directly)
- FP32 used for Kulisch to floating-point conversion, block minifloat alignments, quantization etc.
- Approx 1x floating point operation every N MACs, 5x slowdown

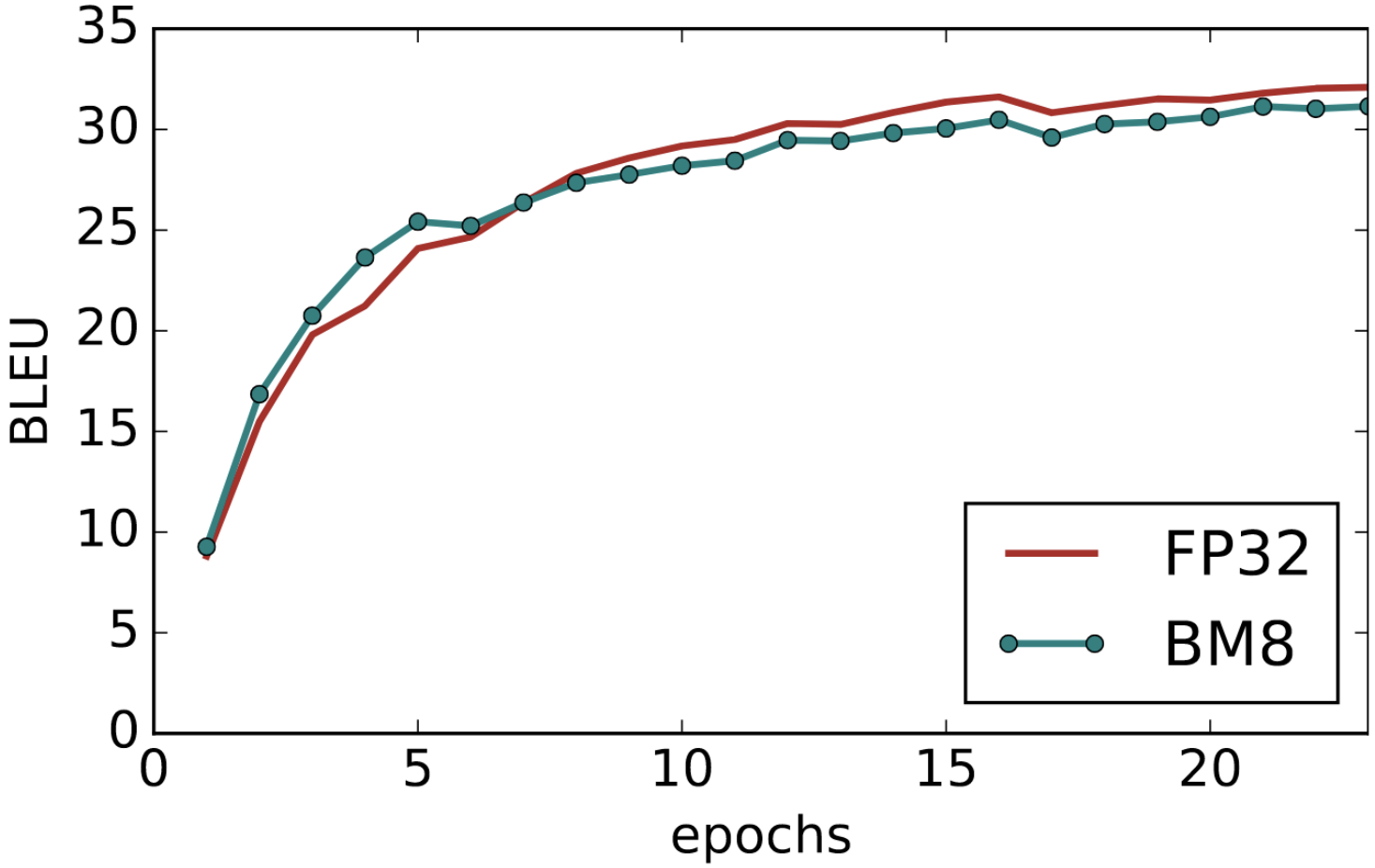
Training Experiments



ResNet18 on ImageNet Validation

Scheme	BFP (ours)	BM (ours)	∇
6-bit	67.0	69.0	+2.0
8-bit	69.2	69.8	+0.6

Transformer



Transformer on IWSLT'14 DE-En dataset

Training Summary

Model (Dataset) [Metric]	FP32	BM8
AlexNet (ImageNet)	56.0	56.2
EfficientNet-b0 (small ImageNet)	62.6	61.8
LSTM (PTB)[Val ppl.]	84.7	87.33
Transformer-base (IWSLT)[BLEU]	32.3	31.8
SSD-Lite (MbNetV2) (VOC)[mAP]	68.6	68.0

Training Accuracy with BM ≈ FP32

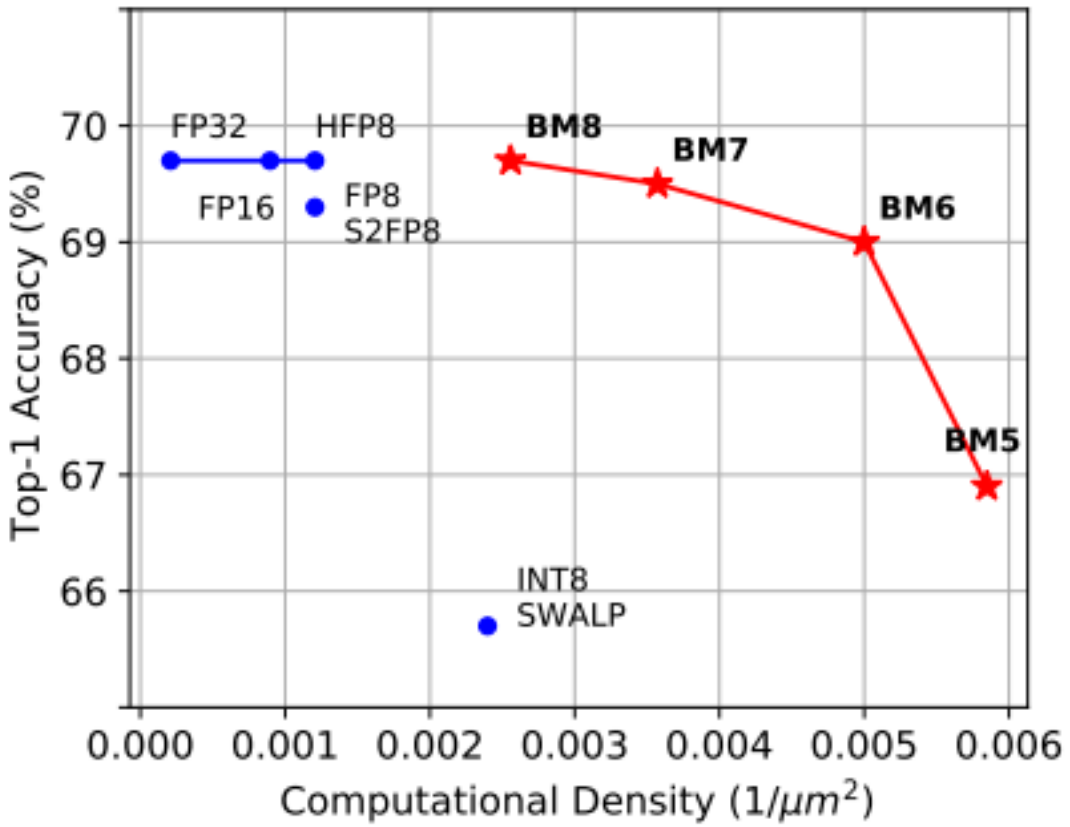
RTL Synthesis

- Designs synthesized at 750MHz with Cadence RTL Compiler and 28nm cell library
  - Fused multiply-add (FMA)
  - 4x4 systolic matrix mutlipliers

Component	Area (μm^2)	Power (μW)
FP32	4782	10051
FP8 (w/ FP16 add)	829	1429
INT8 (w/ INT32 add)	417	1269
BM8	391	1141
BM6	**200**	**624**
INT8 (4x4 systolic)	7005	20253
FP8 (4x4 systolic)	18201	56202
BM8 (4x4 systolic)	**6976**	**18765**

BM8 area and power comparable to INT8

Imagenet



Model: ResNet-18 Dataset: ImageNet

BM units are:

- **Smaller**
- **Consume less Power**

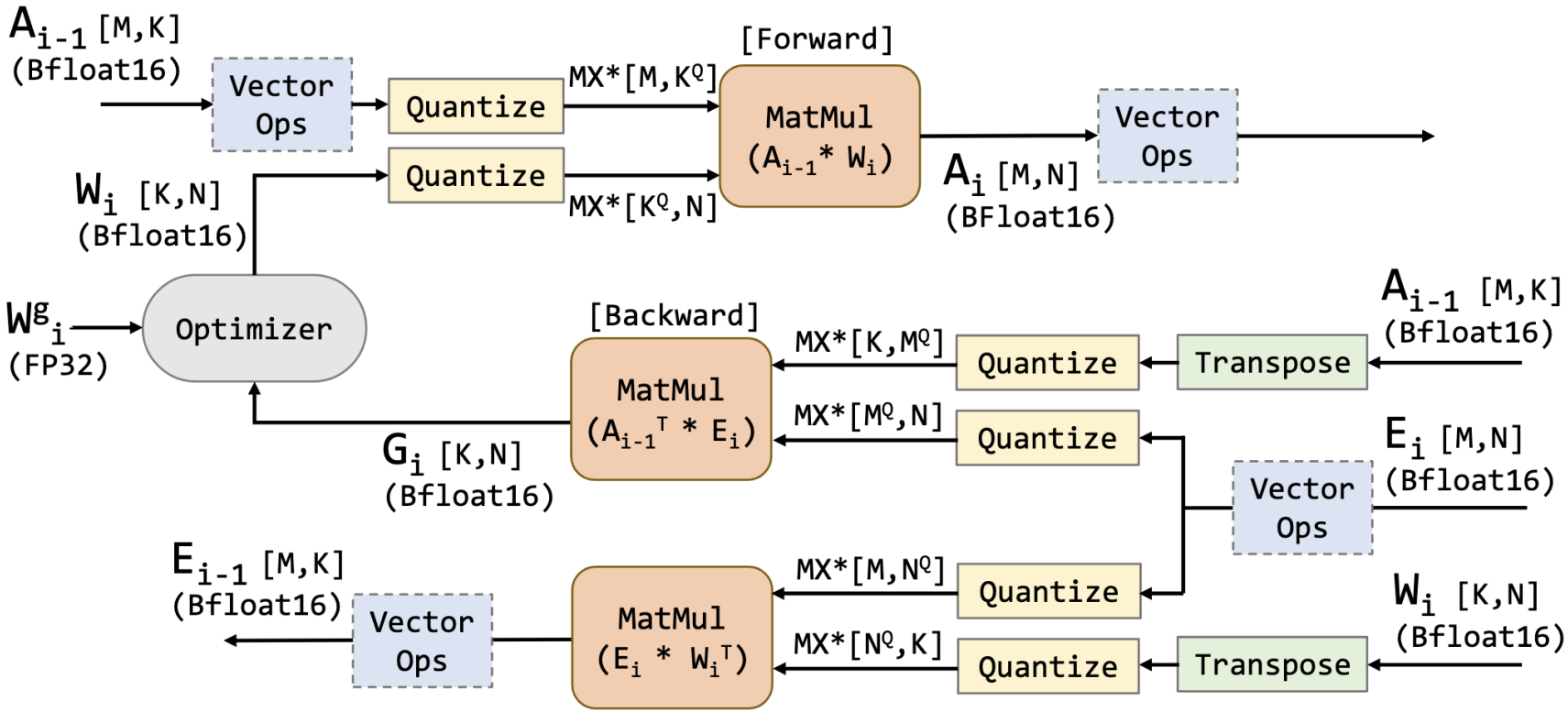
Microscaling Data Formats circa 2023 [2]

- Microsoft, AMD, Intel, Meta, NVIDIA, Qualcomm introduced Microscaling (MX) which is similar idea to BM

Table 1: Concrete MX-compliant data formats and their parameters.

Format Name	Block Size	Scale Data Format	Scale Bits	Element Data Format	Element Bit-width
MXFP8	32	E8M0	8	FP8 (E4M3 / E5M2)	8
MXFP6	32	E8M0	8	FP6 (E2M3 / E3M2)	6
MXFP4	32	E8M0	8	FP4 (E2M1)	4
MXINT8	32	E8M0	8	INT8	8

Compute flow with MX Formats



Blackwell Supported 4-bit Formats

Feature	FP4 (E2M1)	MXFP4	NVFP4
Format Structure	4 bits (1 sign, 2 exponent, 1 mantissa) plus software scaling factor	4 bits (1 sign, 2 exponent, 1 mantissa) plus 1 shared power-of-two scale per 32 value block	4 bits (1 sign, 2 exponent, 1 mantissa) plus 1 shared FP8 scale per 16 value block
Accelerated Hardware Scaling	No	Yes	Yes
Memory	~25% of FP16		
Accuracy	Risk of noticeable accuracy drop compared to FP8	Risk of noticeable accuracy drop compared to FP8	Lower risk of noticeable accuracy drop particularly for larger models

Table 1. Comparison of Blackwell-supported 4-bit floating point formats

Nvidia NVFP4

- High-precision scale encoding plus a two-level micro-block scaling strategy

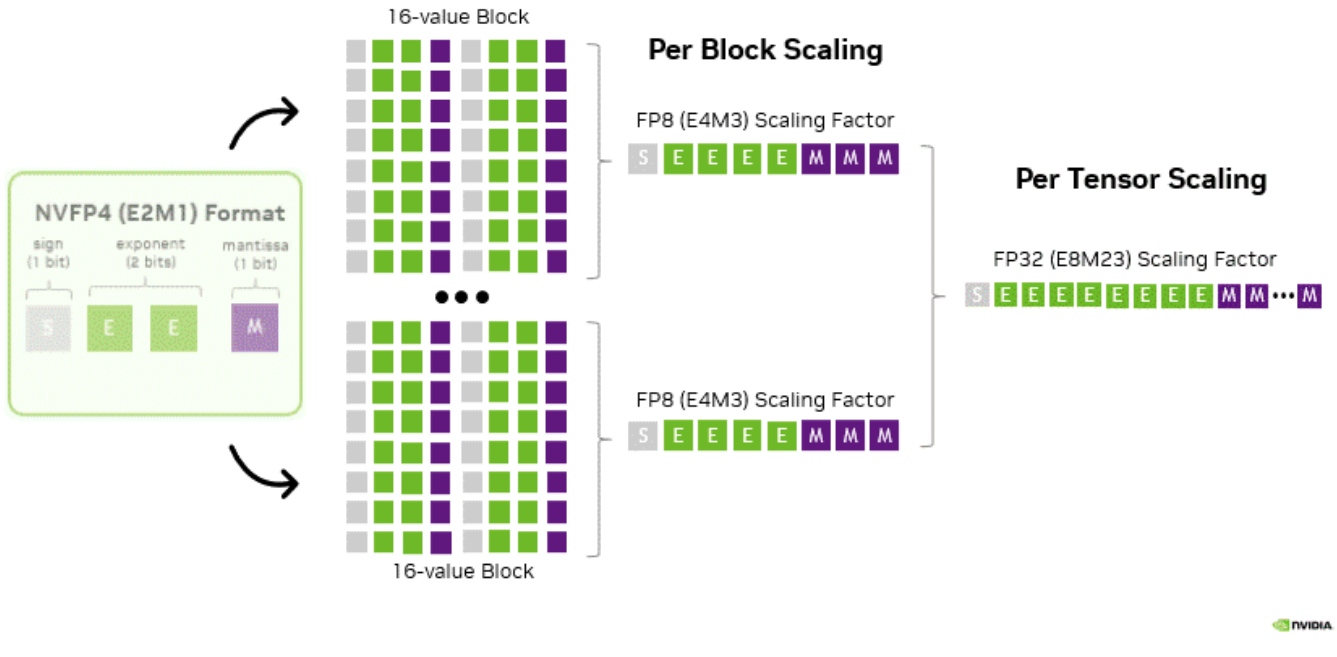
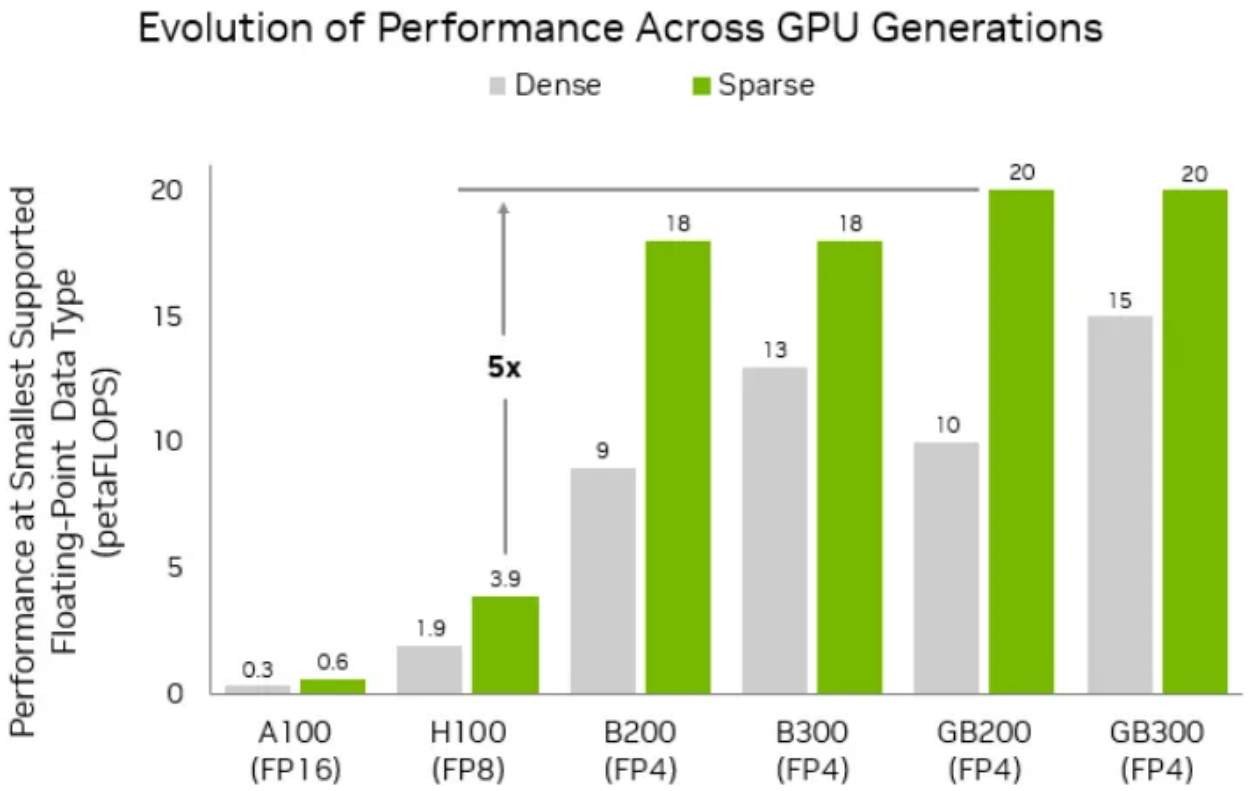


Figure 2. NVFP4 two-level scaling per-block and per-tensor precision structure

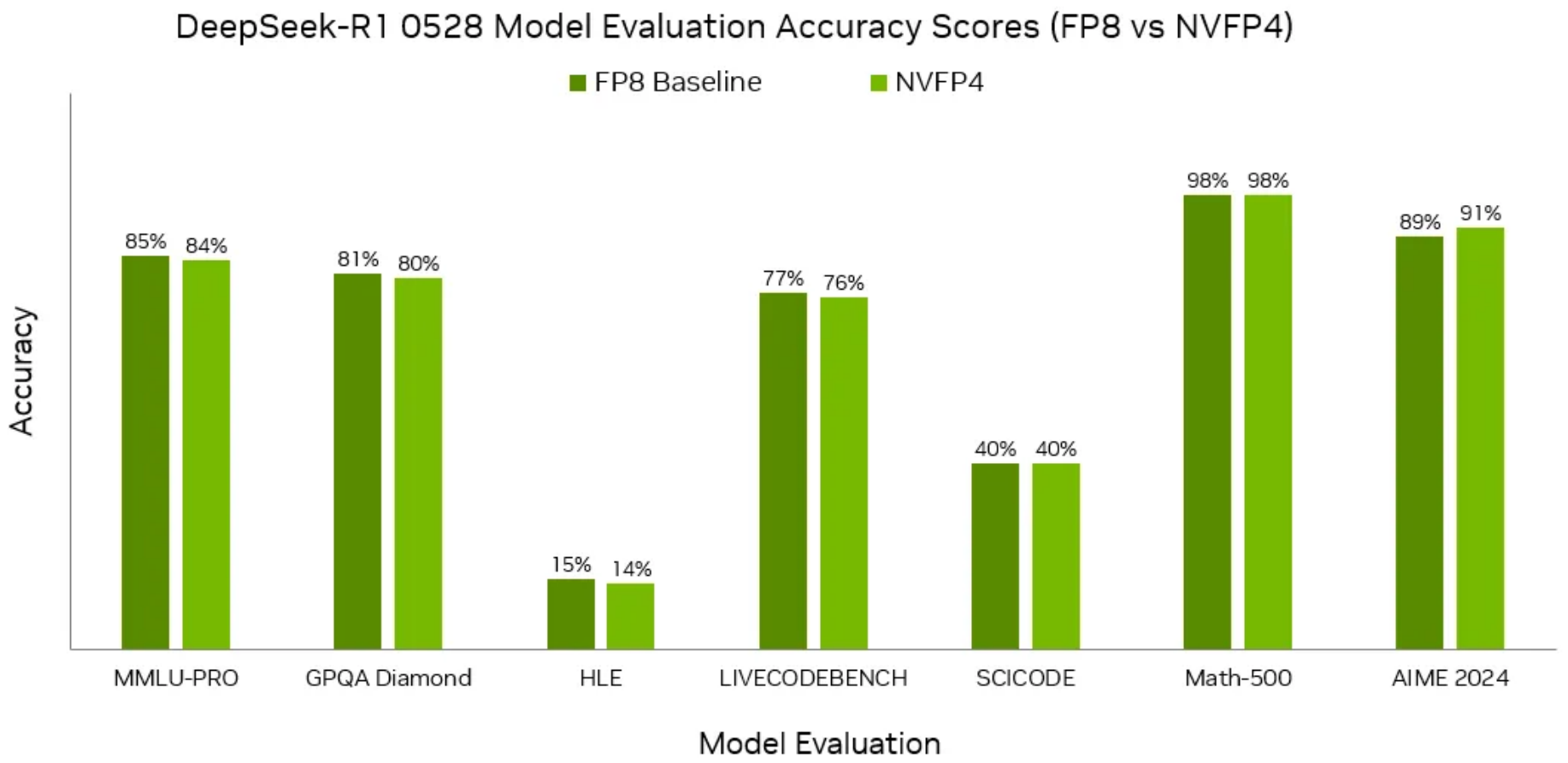
GPU Performance for Different Generations circa 2024 [3]



Source: NVidia

https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/

Inference Accuracy vs FP8



Generative Inference Results

Task	Family	Model	Dataset	Metric	FP32 Baseline	MXFP6		MXFP4
						E2M3	E3M2	
Language Translation	Transformers (Enc-Dec)	Transformer-Base [9]	WMT-17	BLEU Score ↑	26.85	26.98	27.01	25.97
		Transformer-Large [9]			27.63	27.60	27.62	27.33
	LSTM	GNMT [10]	WMT-16		24.44	N/A	N/A	24.56
Image Classification	Vision Transformer	DeiT-Tiny [12]	ImageNet ILSVRC12	Top-1 Acc. ↑	72.16	72.09	70.86	66.41
		DeiT-Small [12]			80.54	80.43	79.76	77.61
	CNN	ResNet-18 [13]			70.79	70.6	69.85	67.19
		ResNet-50 [13]			77.40	77.27	76.54	74.86
		MobileNet v2 [14]			72.14	71.49	70.27	65.41
Speech Recognition	Transformer	Wav2Vec 2.0 [15]	LibriSpeech	WER ↓	18.90	N/A	21.46	29.64

Energy Efficiency (GPT-MoE 1.8T)

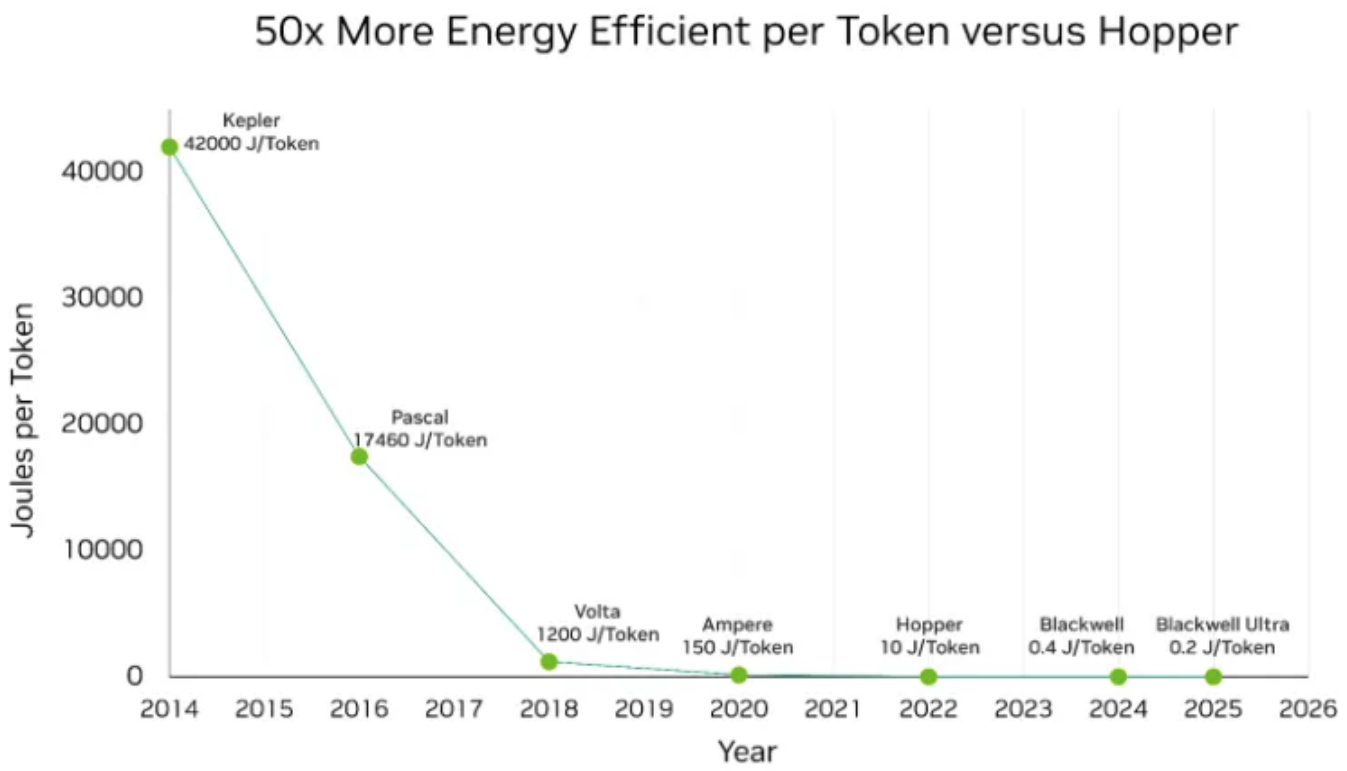


Figure 7. NVFP4 enables up to 50x energy efficiency per token for Blackwell Ultra versus Hopper for GPT-MoE 1.8T

References (available at https://phwl.org/assets/papers/papers)

1. Sean Fox, Seyedramin Rasoulinezhad, Julian Faraone, and David Boland Philip H.W. Leong. A block minifloat representation for training deep neural networks. In *Proc. of The International Conference on Learning Representations (ICLR)*. 2021.
2. BD Rouhani et al., “Microscaling data formats for deep learning”, arXiv:2310.10537, 2023
3. https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/

1. *Number systems*
2. *Block Minifloat / Microscaling*
3. *Applications*

THE UNIVERSITY OF SYDNEY

The University of Sydney

Motivation

- Target problem: Training on Edge
  - Potential: adapt to local conditions
  - Problem: ~3x memory and compute vs. inference
  - Solution: Use custom number system

Overview

- CNN training
- NBeats training

CNN Training Workflow

- Back-propagation using SGD
  - •3X workload of inference

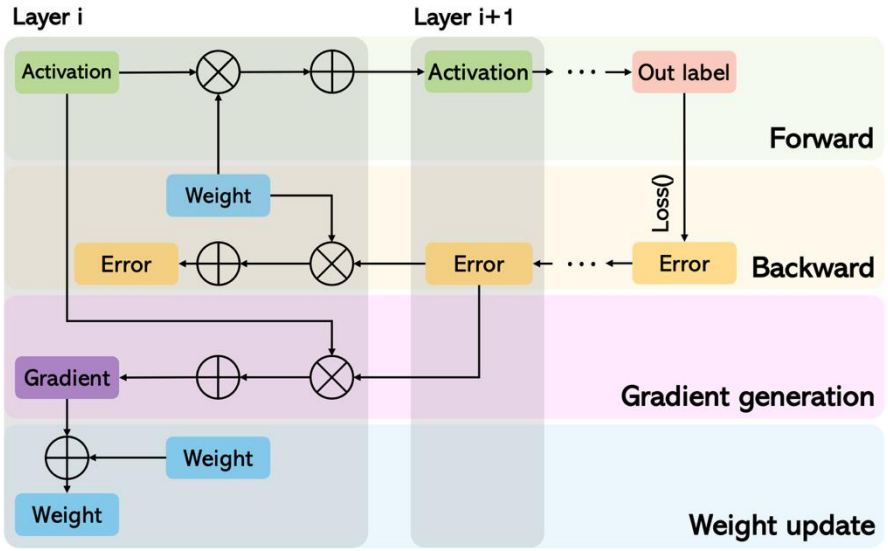


Fig. 1 CNN training workflow: (1) Conv in forward path, (2) transposed Conv in backward path, (3) dilated Conv in gradient generation, and (4) weight update.

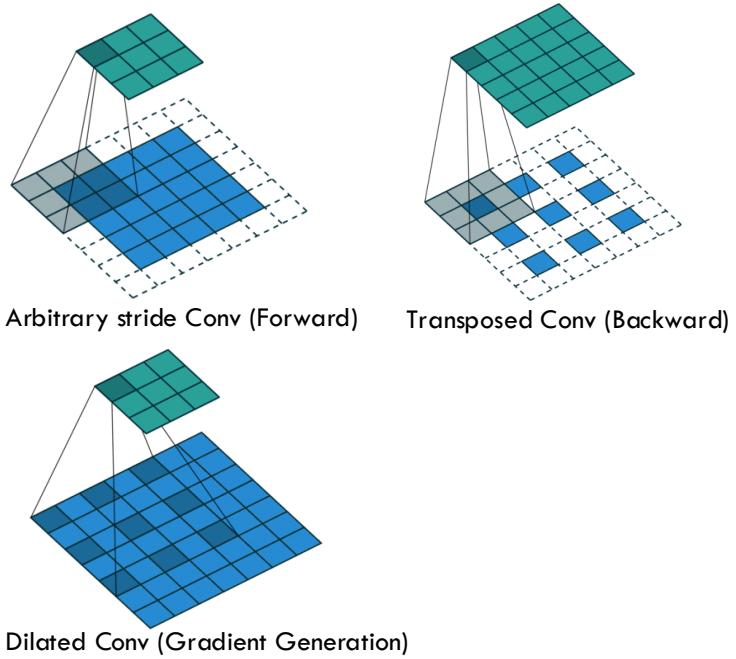


Fig. 2 Non-unit stride Conv, transposed Conv, and dilated Conv [1].

[1] Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

CNN ResNet20/VGG-like Accelerator

- Layer-wise CNN blocks
  - •Unified bm(2,5) representation
    - •Non-unit stride Conv support
    - •Simplified mult/add/MAC
    - •Fused BN&ReLU
- Main blocks
  - •Unified Conv
    - •Conv & transposed Conv
  - •Dilated Conv
    - •Weight kernel partition

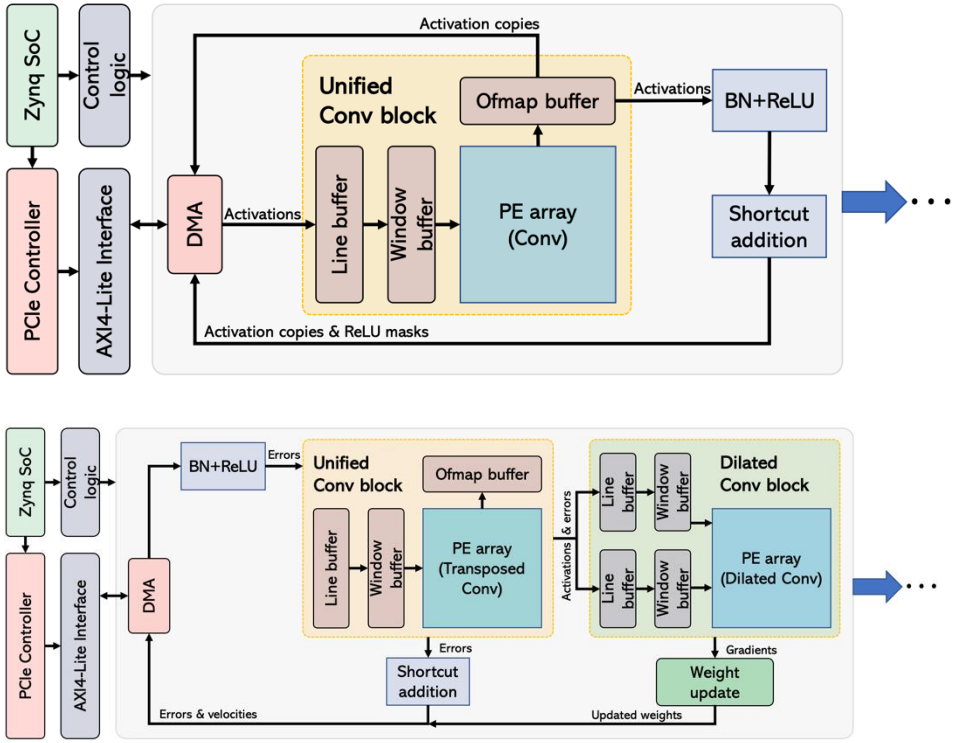
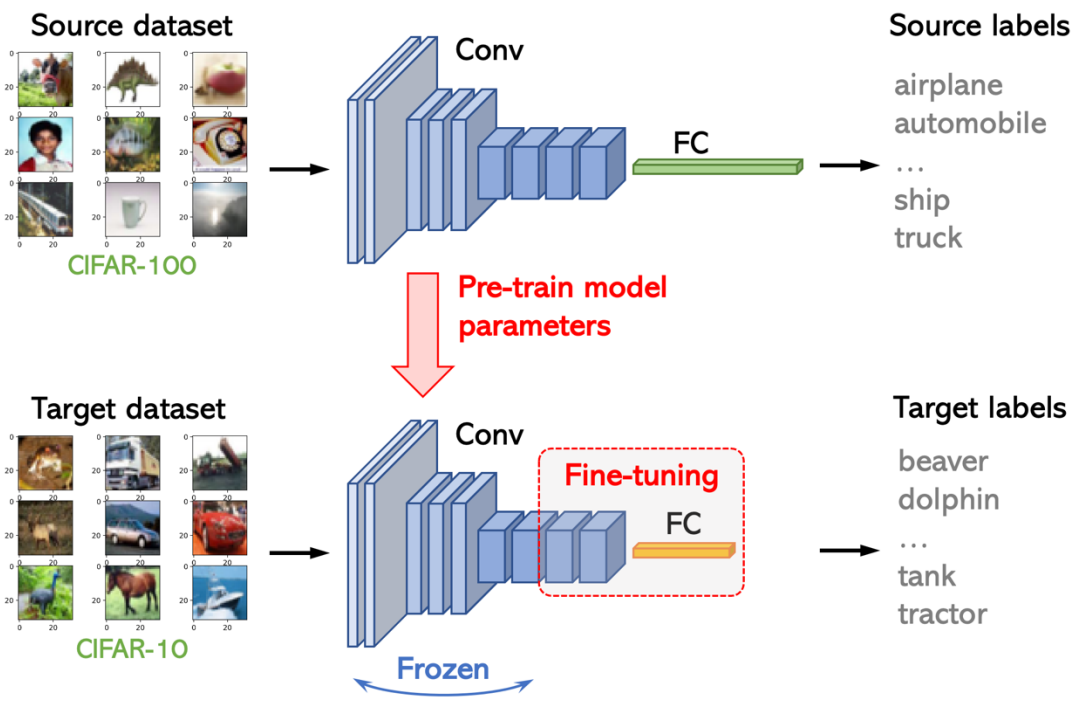
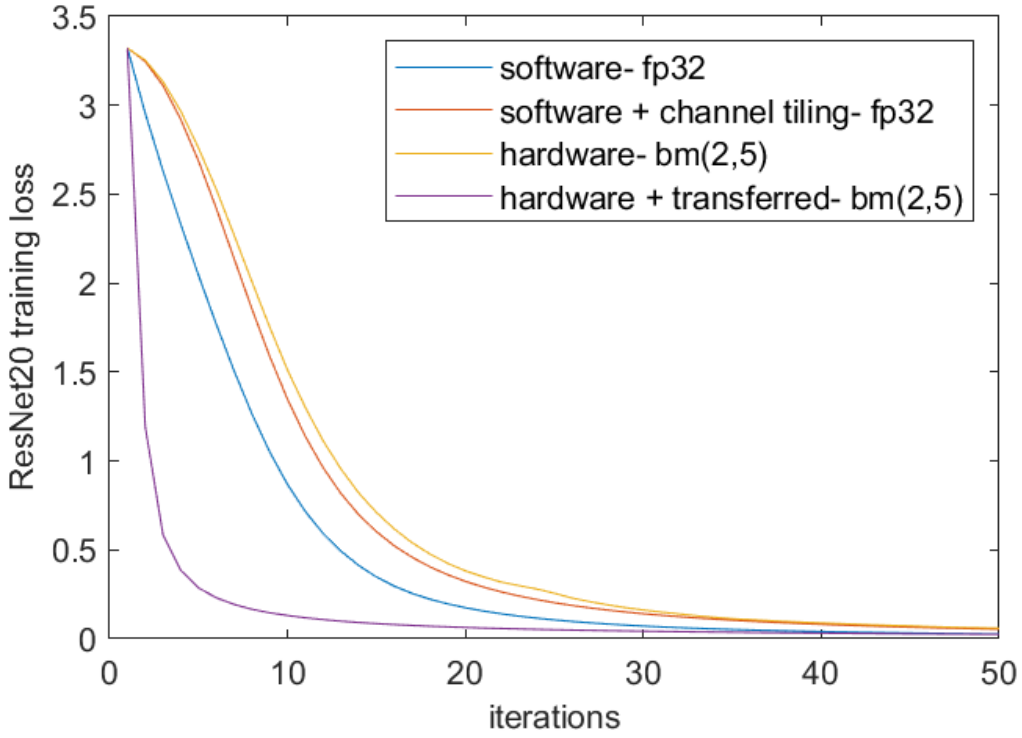


Fig. 3 Overall architecture of the generic training accelerator for layer-by-layer processing. BN and ReLU are fused.

CNN BM-based TL (ICCAD23)





Transfer learning example from CIFAR-100 to CIFAR-10

Resource and Power

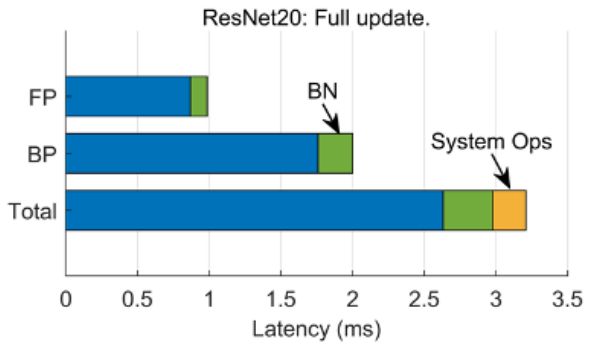
TABLE III
RESOURCE UTILISATION OF AND POWER THE RESNET20 ACCELERATOR (WITH THE STATIC POWER OF 30W).

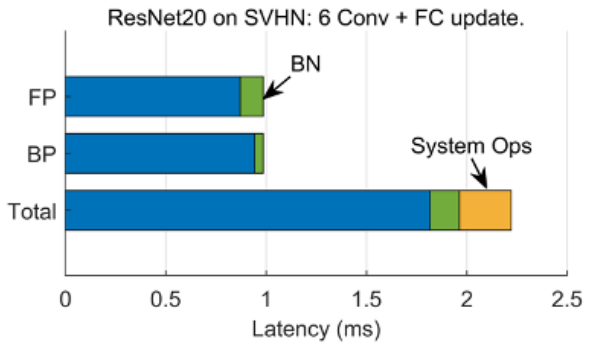
	CLB	LUT	DSP	BRAM	Vivado(W)	PPS(W)
Full update	28824	166502	686	1171	8.714	35
6 Conv+FC	25589	161129	685	671	7.725	34
2 Conv+FC	21340	129453	621	571	6.779	34

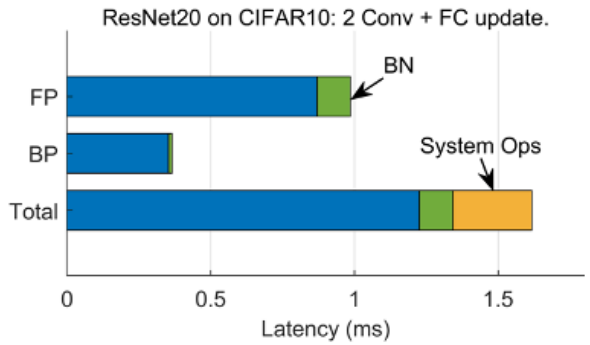
TABLE IV
RESOURCE UTILISATION AND POWER OF THE VGG-LIKE ACCELERATOR (WITH THE STATIC POWER OF 30W).

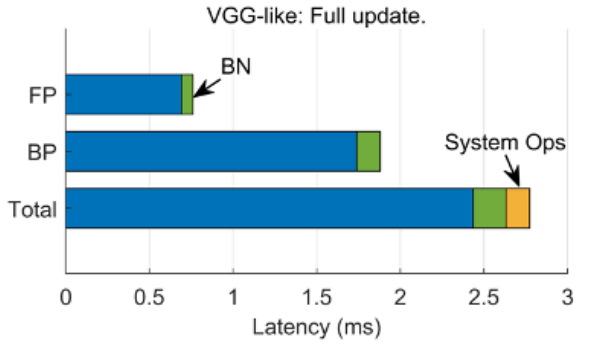
	CLB	LUT	DSP	BRAM	Vivado(W)	PPS(W)
Full update	20688	119086	614	505	6.824	34
3 Conv+FC	20489	119740	613	325	6.499	34

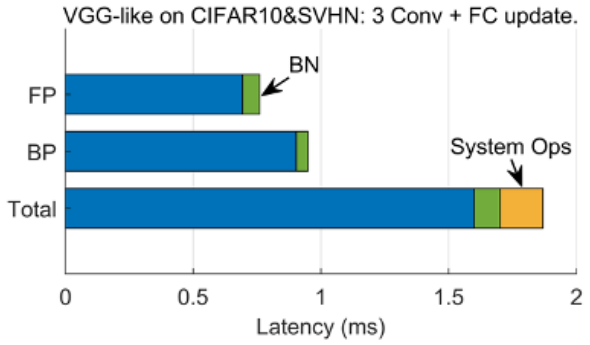
Latency Breakdown











Overview

- CNN training
- NBeats training

This Work

- BM arithmetic
  - Block size: A novel technique for cross-block BM matrix multiplication
  - Mixed precision: BM GEMM kernel with runtime configurable precision for FW/BP with pipeline optimization
  - DSP packing: 4-bit BM MAC unit with 6 multiplication packed per DSP

- Training accelerator
  - First implementation of 4-bit BM training for time series network (NBEATS)

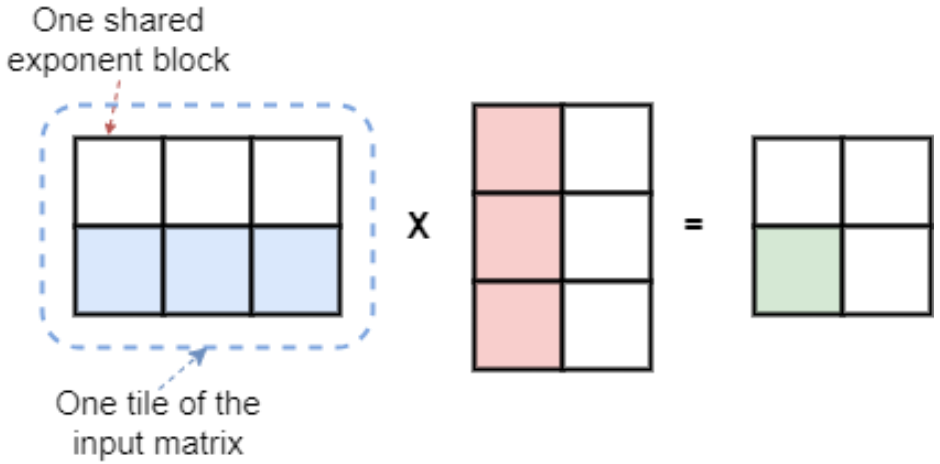
Block size & mixed precision in BM Matrix multiply

Block size ↓

A_k, e_k, W_k

(mixed precision)

Accuracy ↑

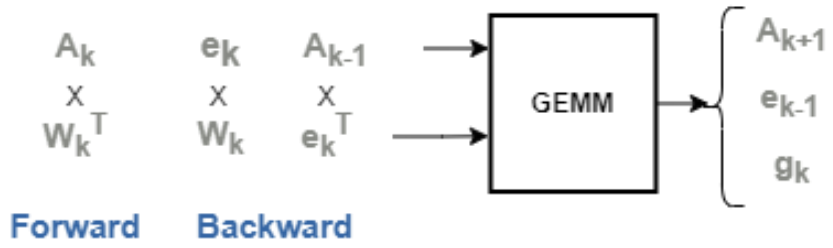


Cross block Matrix Multiplication

- Problem:
  - Block size ≠ tile size
  - Computation error in cross-block MM
  - Runtime configurable precision GEMM

PE array

Shared exponent

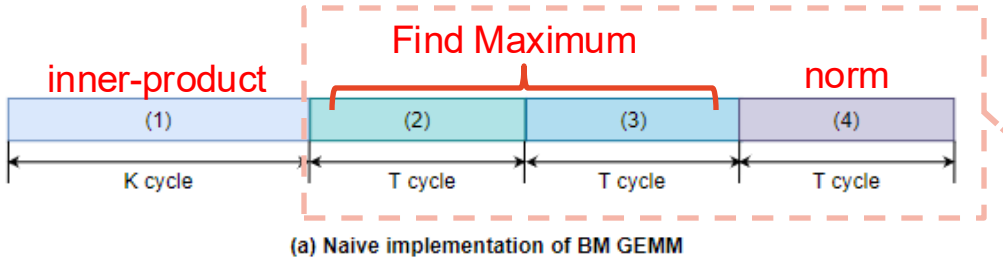


Mixed-precision GEMM for different training phase

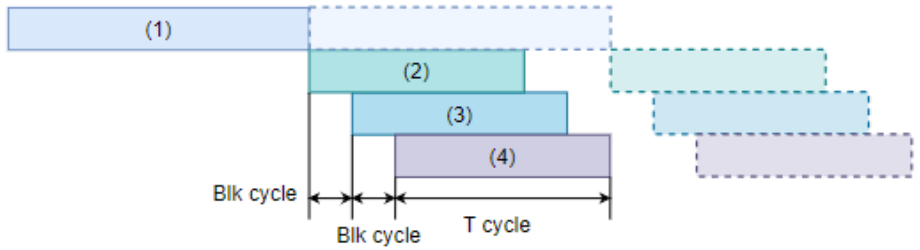
BM GEMM kernel

› Pipelined implementation

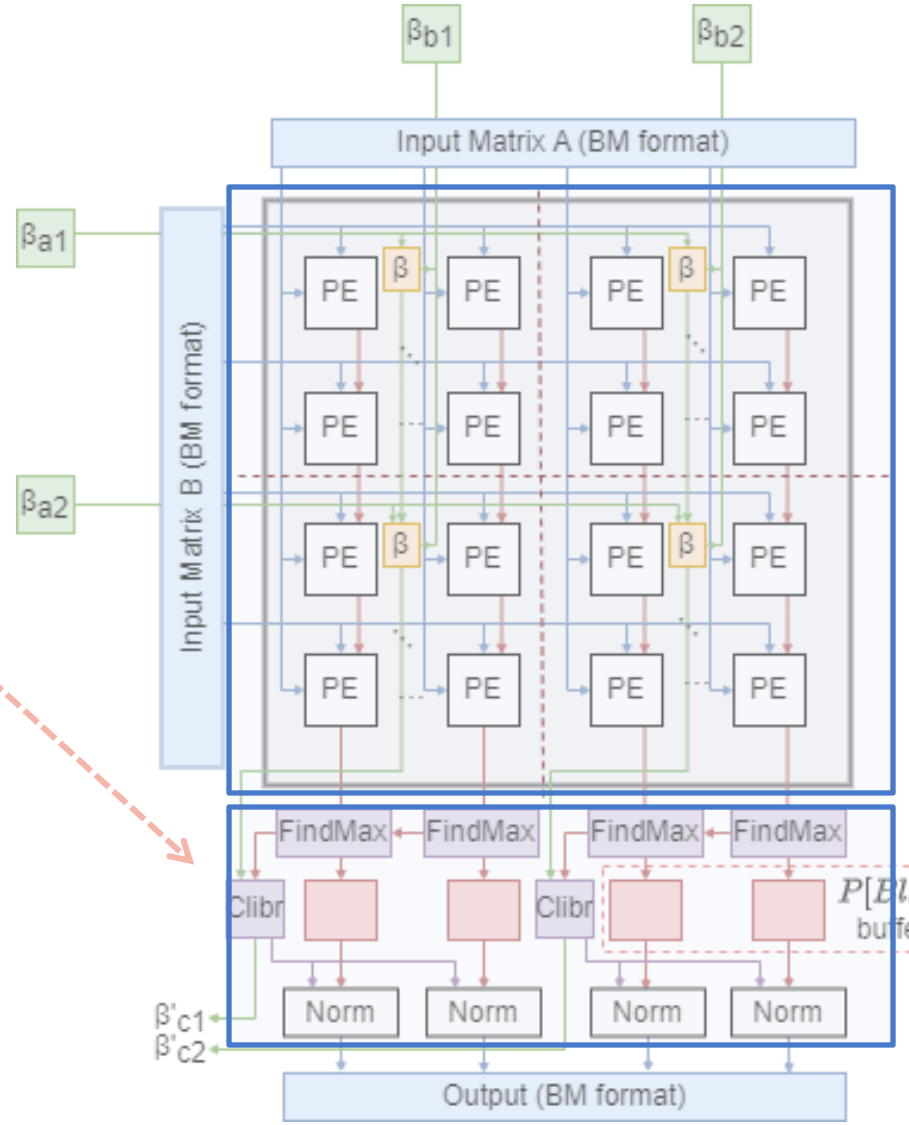
- Inter block & Intra-Normalization Pipeline



(a) Naive implementation of BM GEMM



(b) Pipelined implementation of BM GEMM



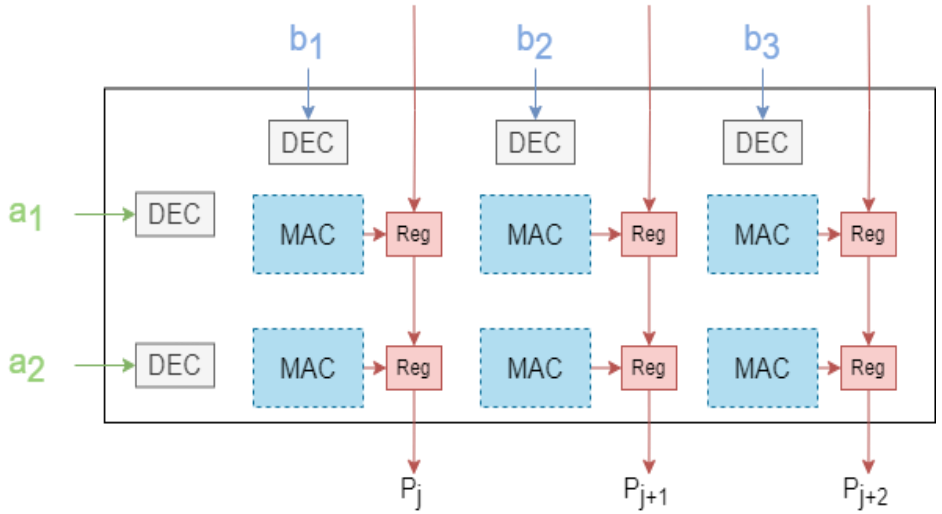
PE array (inner-product)

Normalization

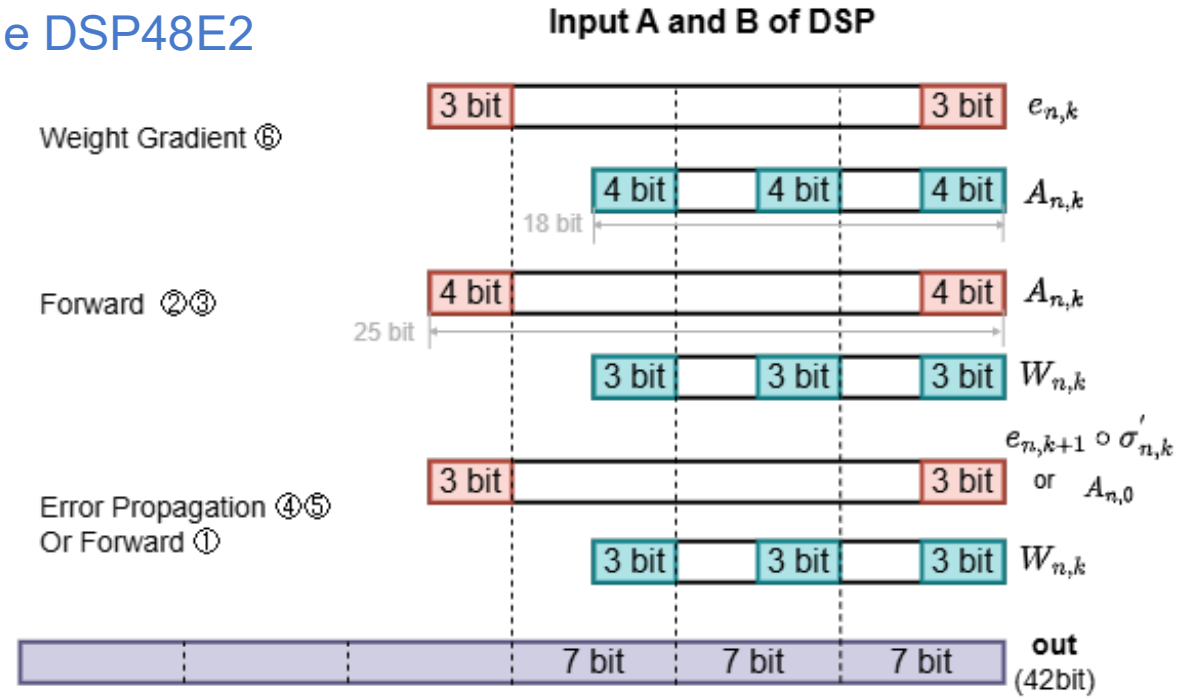
DSP packing

› DSP packed PE optimized for 4-bit BM training

- Each PE has 2*3 input, 6 MAC unit
- 6 Significand multiplication in MAC mapped to one DSP48E2
- Support different precisions during training



DSP packed PE design for 4-bit BM MAC

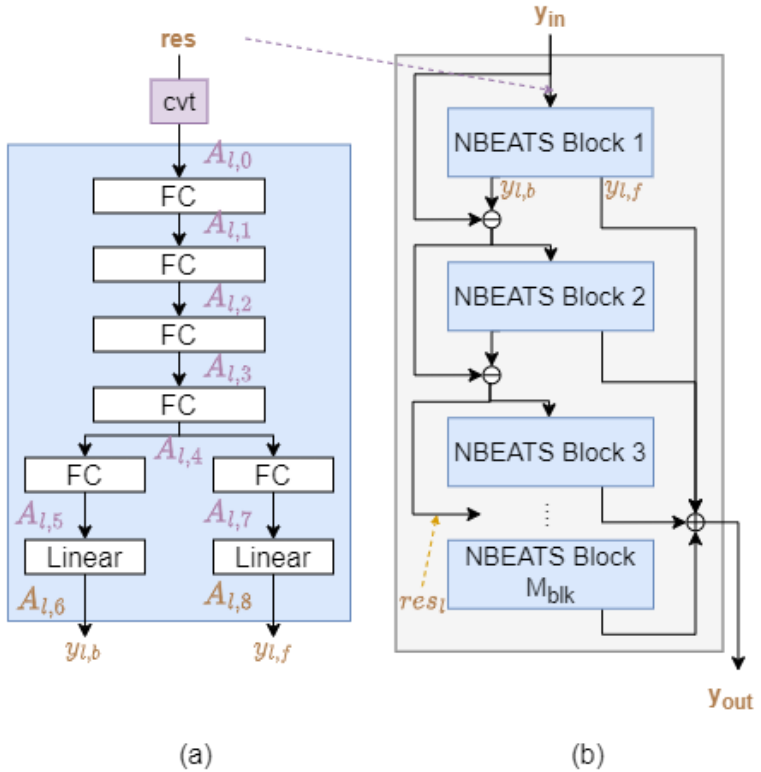


Bit Arrangement in Different Training Phases

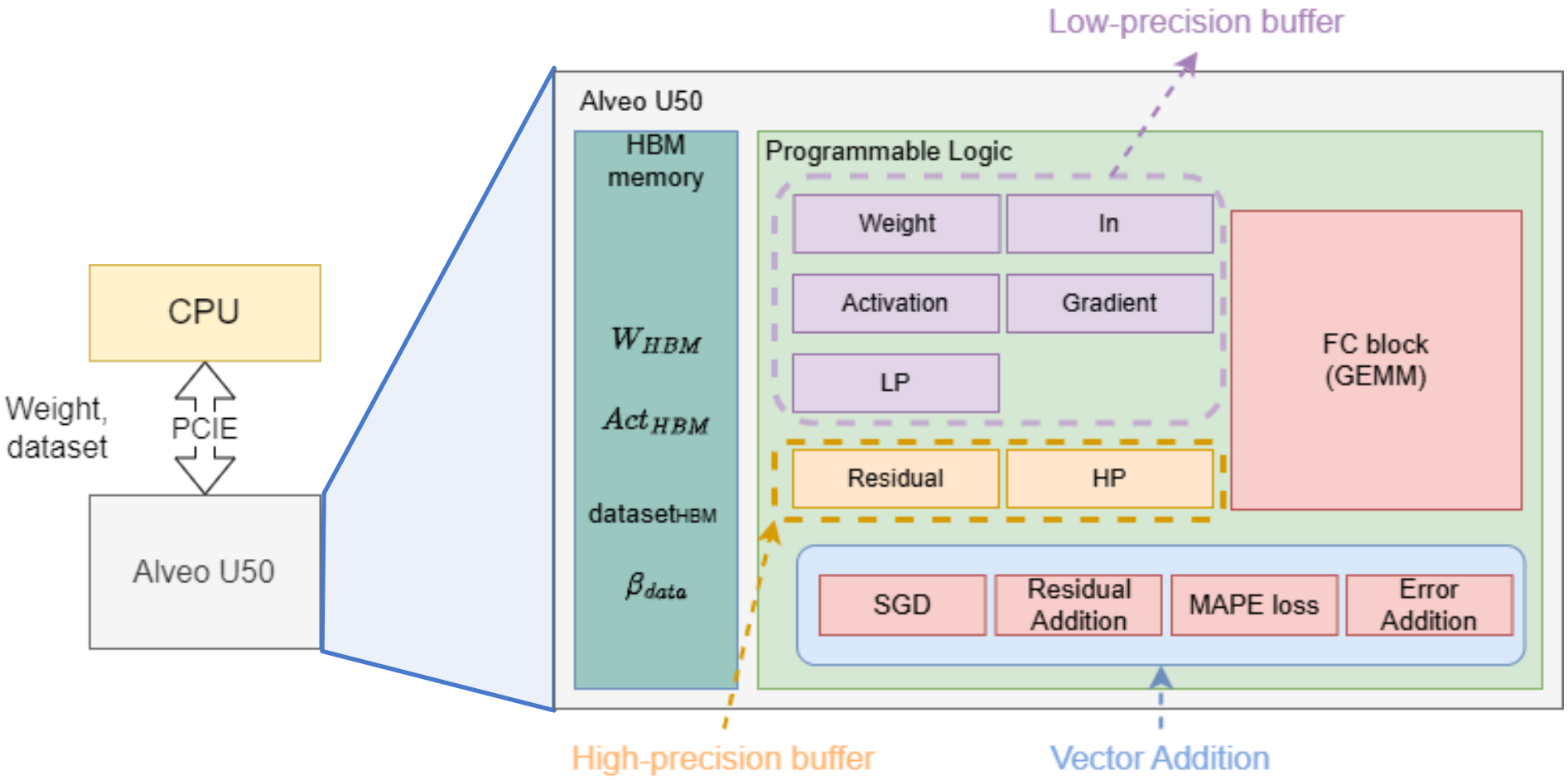
Time series prediction network - NBEATS

› NBEATS: A time series prediction network with regular architecture (FC layer & residual)

› SOTA accuracy result on M3, M4 and statistics benchmark

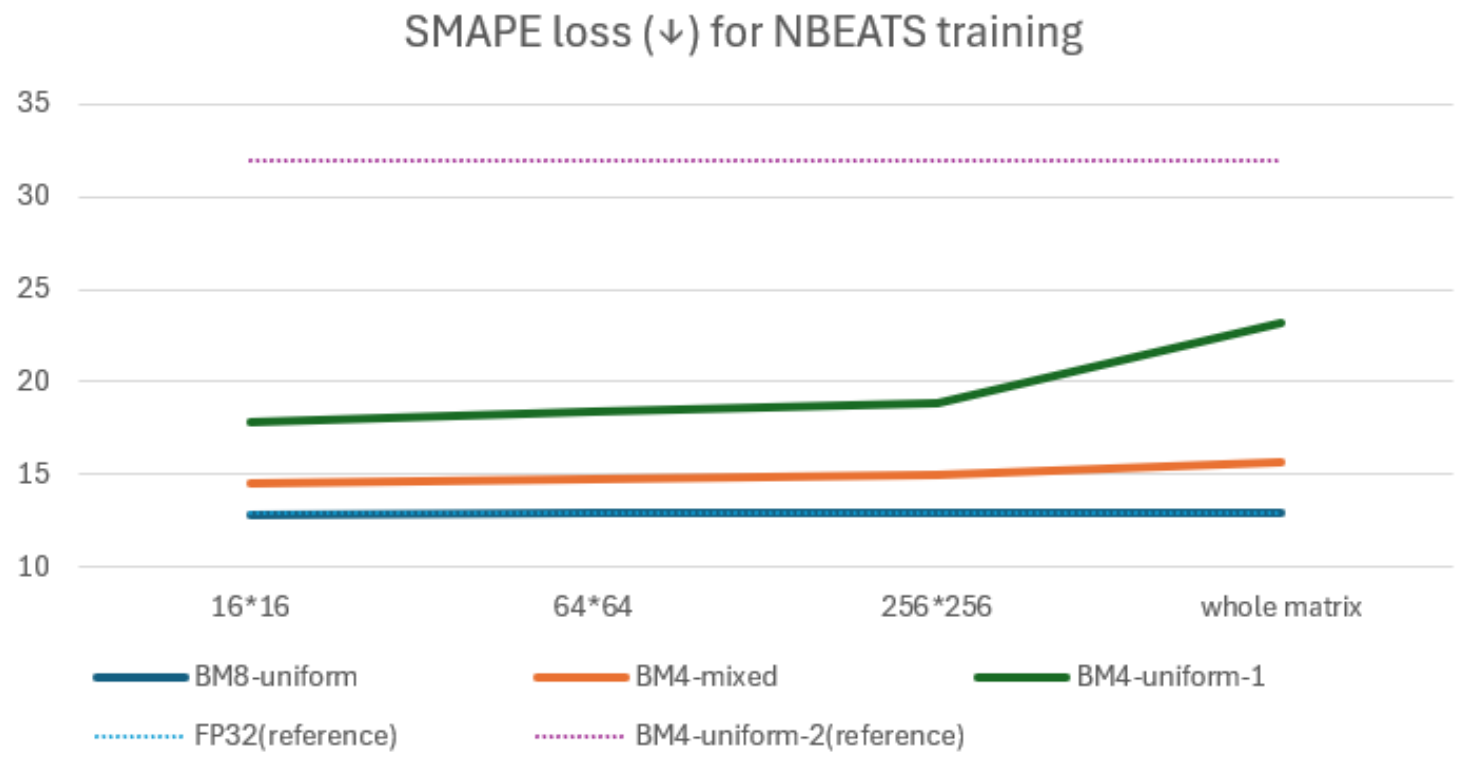


NBEATS training accelerator



NBEATS training accuracy

› Smaller block size, high precision residual, Mixed precision improves accuracy



NBEATS training accelerator

› BM4-mixed packed

- Accelerator achieves 779 Gops throughput, DSP utilization 0.7 Gops/DSP (SOTA), power efficiency 42.4 Gops/W (3.1x)

	Ours (no packing)	Ours (packing)	GPU
Device	Alveo U50		GTX 1080
Process	TSMC 16nm		
Num system	BM4-mixed	BM4-mixed	FP32
Freq.(MHz)	183	159	**1733**
Tput. (Gops)	299.03	779.12	**2991**
Power(W)	20.86	**18.38**	220
Gops/W	14.34	**42.4**	13.6
Gops/DSP	0.27	**0.71**	-

Summary

- Demonstrated FPGA-based 4-bit training of time series prediction networks with minimal impact on accuracy
  - Systolic array that combines mixed precision and DSP packing
  - 3x Gops/W in same technology as GPU

Suitable for edge-based systems with modest energy resources

References (available at https://phwl.org/assets/papers/papers)

[1] Chuliang Guo, Binglei Lou, Xueyuan Liu, David Boland, Philip H.W. Leong, and Cheng Zhuo. BOOST: block minifloat-based on-device CNN training accelerator with transfer learning. In *Proc. ICCAD*, 1–9. 2023.

[2] Wenjie Zhou, Haoyan Qi, David Boland, and Philip H. W. Leong. FPGA-based block minifloat training accelerator for a time series prediction network. *ACM Trans. Reconfigurable Technol. Syst.*, 18(2):1–23, March 2025.