Fast and accurate algorithm for matrix multiplication using fused multiply-add

Katsuhisa Ozaki（Shibaura Institute of Technology）
Joint work with
Toru Koizumi（Nagoya Institute of Technology）

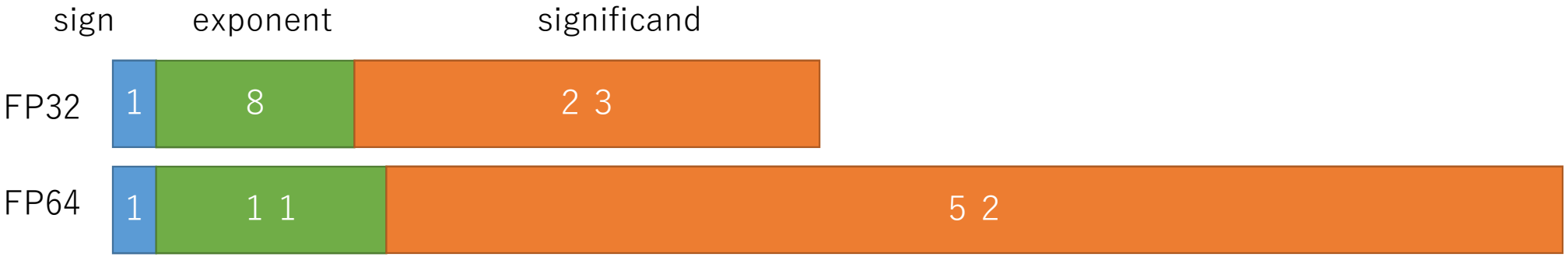
The Second French-Japanese Workshop on Numerical Computations
Pierre and Marie Curie campus, Sorbonne University, Paris,
March 12, 2025

Agenda

- Introduction
- Notation, Basic and Known-Algorithms
- Proposed Method
  - accurate numerical algorithm for matrix multiplication
- Numerical Examples
  - double → double word
  - double word → double word on CPU
- Conclusion

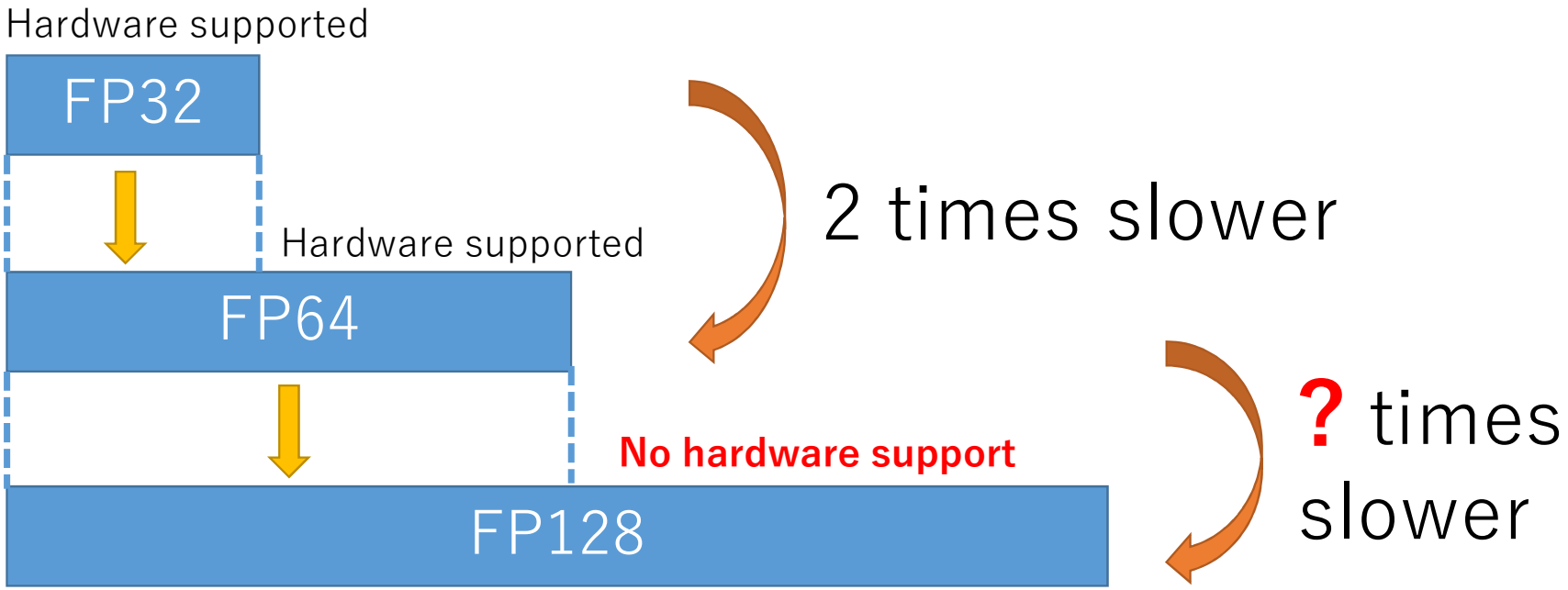
Introduction

- Numerical computation is widely used.
- IEEE 754 binary formats are used for numerical computations.



- if an accuracy of the computed result is unsatisfactory, FP128 or multiple precision arithmetic is used.

Introduction （on CPU）



Example for MPFR

- MPFR: C library for multiple-precision floating-point computations with correct rounding

GFLOPS

N	FP32	FP64	FP128
1000	152	70	0.14
2000	233	94	0.17

500 times slower

Maximum relative error

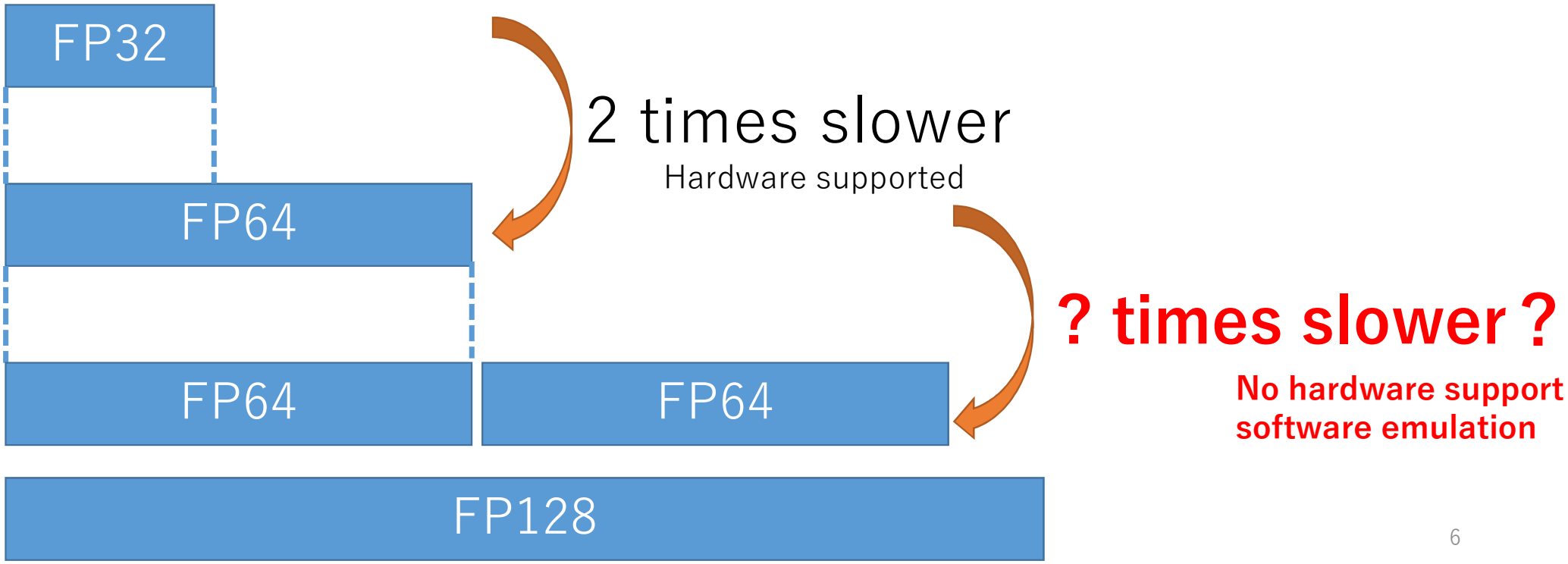
N	FP32	FP64	FP128
1000	5.51e-02	3.10e-10	3.87e-28
2000	3.00e-01	2.10e-09	2.36e-28

Core i7-8665U Windows 10, MATLAB R2021a
FP128: Multiprecision Computing Toolbox for MATLAB
(well tuned commercial toolbox)

Double-double (double-word) arithmetic is introduced later

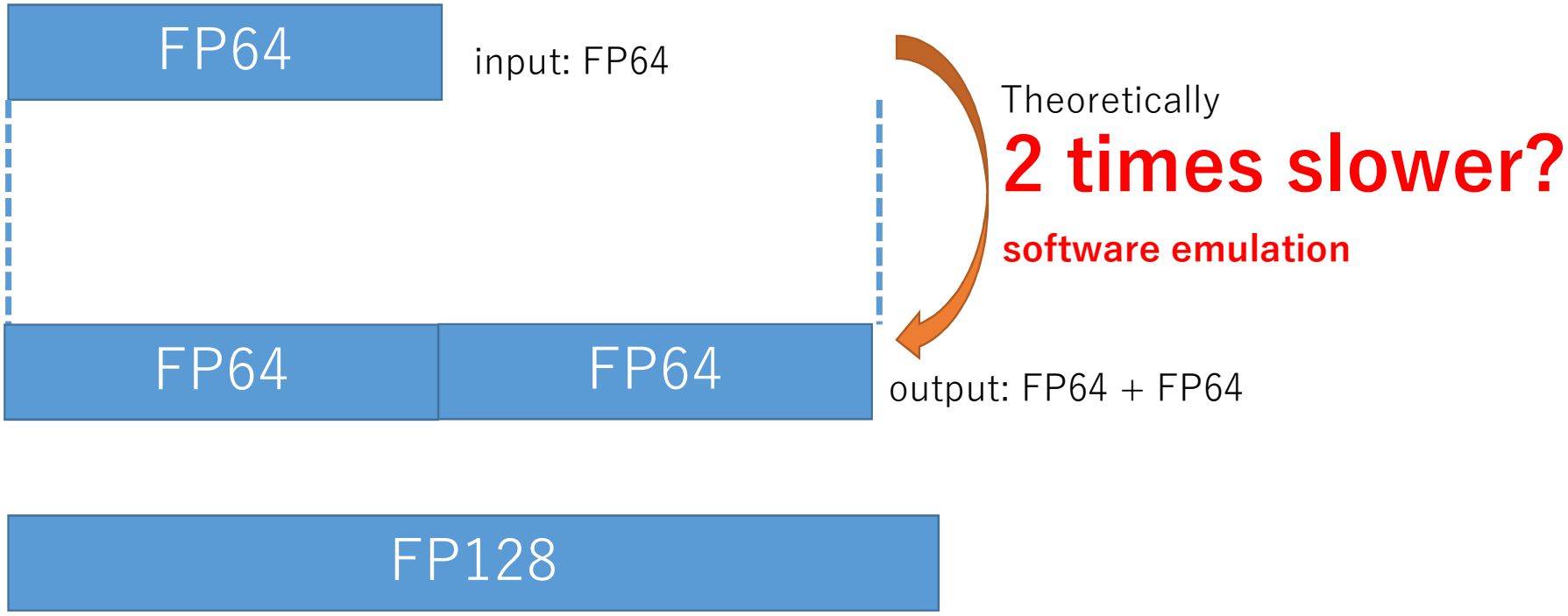
Overview of the proposed method (CPU)

- increase precision



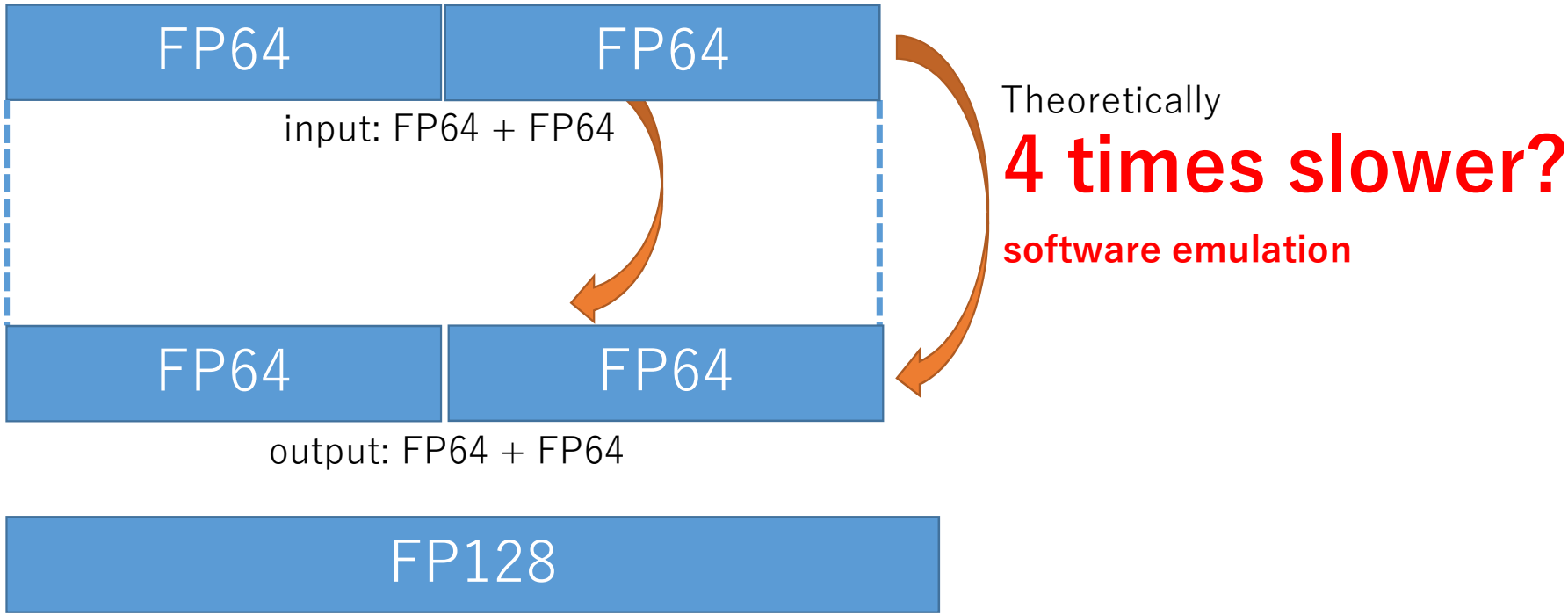
Overview of the proposed method (CPU)

- increase precision



Overview of the proposed method (CPU)

- increase precision



Notation

- We use binary fl-pt numbers defined by IEEE 754
- fl(): a result of numerical computations
- fma(a,b,c): ab+c by fused multiply-add
- u: the roundoff unit
  - FP32→ 2^{-24}, FP64→2^{-53}
- We use MATLAB like notation for algorithm
  - `function [x, y] = TwoSum(a, b)`

output　　input

function name

Error-free Algorithms

Knuth (1969)

```
function [x,y] = TwoSum(a,b)
  x = a + b;
  z = x - a;
  y = ( a - (x-z) ) + (b-z);
end
```

$a + b = x + y,\ x = fl(a+b)$

Dekker (1971)

```
function [x,y] = FastTwoSum(a,b)
  x = a + b;
  y = (a-x) + b;
end
```

$a + b = x + y,\ x = fl(a+b)$
a is multiple of unit in the last place of b.

```
function [x,y] = TwoProdFMA(a,b)
  x=a*b;
  y=fma(a,b,-x);
end
```

$a * b = x + y,\ x = fl(a*b)$

First Problem

- Accurate numerical algorithm for A*B

- A*B → Ch + Cl
- Ch: high part
- Cl : low part

- We focus on $a_{ik}*b_{kj}$+ch+cl → ch+cl in the dot product.

Double-word: Multiplication (DW: Sloppy Ver.)

- function [zh,zl] = DW_mul(ah,bh)
- [p1, p2] = TwoProductFMA(ah, bh);
- End

2

ah*bh+ch+cl ≈ dh+dl

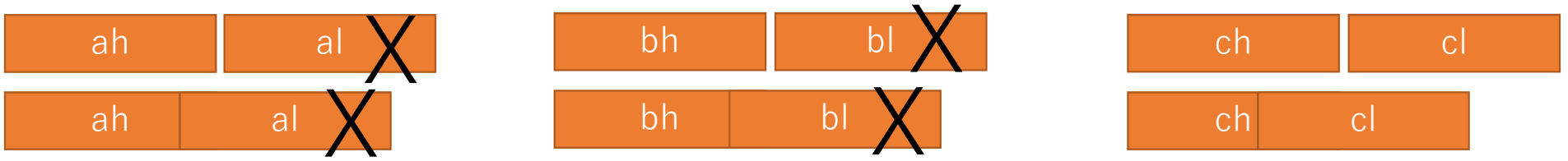
- function [dh,dl] = DW_sum(ch, cl, zh, zl)
- [s, e] = TwoSum(ch, zh);
- e = e + cl + zl;
- [dh, dl] = FastTwoSum(s, e);
- end

11

13 operations
11ADD+1FMA+1MUL

D.H. Bailey:
QD library
https://www.davidhbailey.com/dhbsoftware/

Pair-Arithmetic (PA)



- function [zh,zl] = PA_mul(ah, bh)
- [zh,zl] = TwoProductFMA(ah,bh);
- end

2

10 operations
8ADD+1FMA+1MUL

- function [dh,dl] = PA_sum(ch, cl, zh, zl)
- [dh,t] = TwoSum(ah,bh);
- dl = cl + zl + t;
- end

8

M. Lange and S.M. Rump.
Faithfully rounded floating-point computations.
ACM transactions on mathematical software, 46(3), 2020.

Proposed Method efficiently using FMA

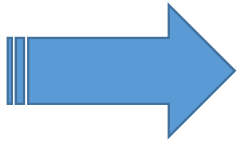
Koizumi et al. (FastTwoFMA)

General methods in

S. Boldo and J.M. Muller,
Exact and Approximated Error of the FMA,
IEEE TRANSACTIONS ON COMPUTERS,
VOL. 60, NO. 2, 2011, 157--164.

- fl-pt numbers a, b, c, we obtain x and y such that
- $ab+c \doteqdot x + y, \quad x = \mathrm{fma}(a,b,c); \quad y = \mathrm{fma}(a,b,\mathrm{fl}(c\text{-}x));$

T. Koizumi, T. Tsumura, H. Irie and S. Sakai, A fast implementation of perfect-accuracy double-precision exponential function, IPSJ SIG Technical Report, 2023-HPC-192 (2023), 21:1–21:12.



Our analysis

If $2|ab| \leqq |c|$, $\mathrm{fl}(c\text{-}x) = c\text{-}x$ and
$|x + y - (ab + c)| \leqq u^2|ab + c|.$

u is roundoff unit

Proposed Method

- For floating-point numbers a, b, ch, cl
- $ab+ch+cl \doteqdot dh + dl,$ $\quad dh = \mathrm{fma}(a,b,ch);$ $\quad dl = \mathrm{fl}(\mathrm{fma}(a,b,\mathrm{fl}(ch\text{-}dh))+cl);$

2FMA+2ADD→**4** ops

u is roundoff unit

- Dot product:
- $(((((\boldsymbol{\alpha} + x_1y_1)+x_2y_2)+x_3y_3)+x_4y_4)+\ldots+x_ny_n)\text{-}\boldsymbol{\alpha}$

  double-word

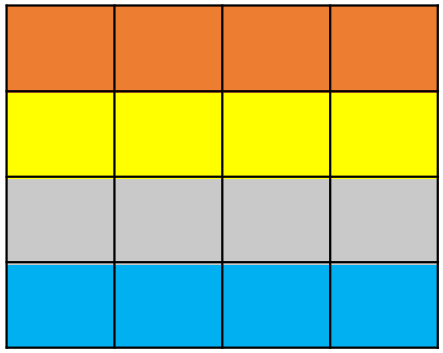
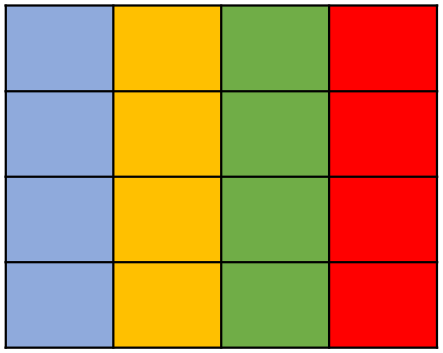
- ***α is greater than*** $3(1+\mathrm{nu})|x^T||y|,$
- ***the important inequalities are always satisfied.***

$$|AB - T_1 - T_2| \leq n^2u^2W + n\omega\mathrm{E}, W \geq 3(1+nu)|Ah||Bh|$$

Proposed Method

- Candidate 1 ：Cauchy–Schwarz inequality

  cost of α **is negligible**

A  B 

$$|x^T y| \leq \|x\|\|y\|$$

- Candidate 2：compute $|x^T||y|$ with rounding upward (low-prec)
- **Cost *of* α is not negligible**

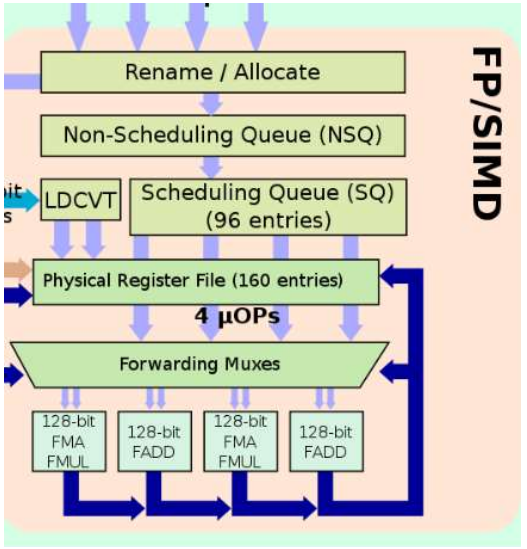
- Small α is important

Cost by operations and bottle neck

Methods	Add	Mul	FMA	Total	Bottleneck
Proposed	2	0	2	4	2
PA≒Dot2	8	1	1	10	8
DW	11	1	1	13	11

FAdd unit

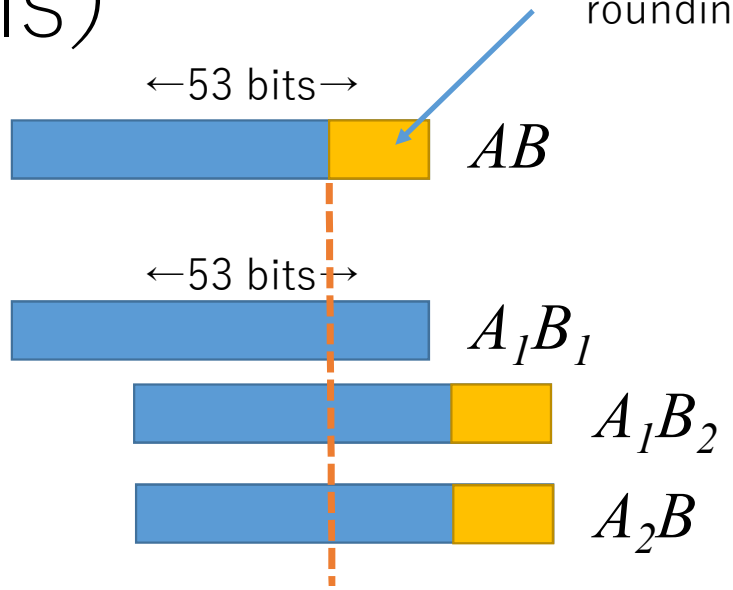
- Intel：12 gen ～ 15 gen (now)
  (up to 11 gen, no support)
- AMD→ from Zen+



Zen (family 17h)

Figure from
Zen - Microarchitectures - AMD – WikiChip

Ozaki Scheme (3～4 muls)



- $A = A_1 + A_2,\ B = B_1 + B_2$
- $AB = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2$
  $= A_1B_1 + A_1B_2 + A_2B$

- $A = A_1 + A_2 + A_3,\ B = B_1 + B_2$
- $AB = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2 + A_3B_1 + A_3B_2$
- $= A_1B_1 + A_2B_1 + (A - A_3)B_2 + A_3B$

No rounding error in red color

Ozaki Scheme （5～6 muls）

- $A = A_1 + A_2 + A_3 + A_4,\ B = B_1 + B_2$
- $AB = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2 + A_3B_1 + A_3B_2 + A_4B_1 + A_4B_2$
  $= A_1B_1 + A_2B_1 + A_3B_1 + (A - A_4)\,B_2 + A_4B$

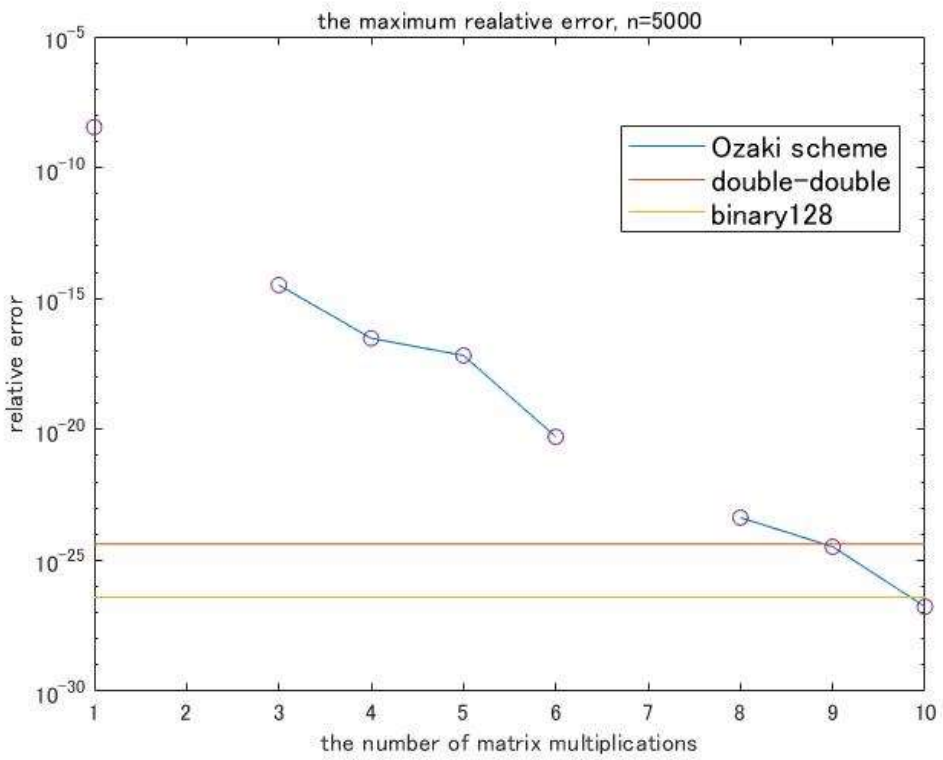
- $A = A_1 + A_2 + A_3,\ B = B_1 + B_2 + B_3$
- $AB = A_1B_1 + A_1B_2 + A_2B_1 + A_1B_3 + A_2B_2 + A_3B_1 + A_2B_3 + A_3B_2 + A_3B_3$
  $= A_1B_1 + A_2B_1 + A_1B_2 + A_2B_2 + (A_1 + A_2)B_3 + A_3B$

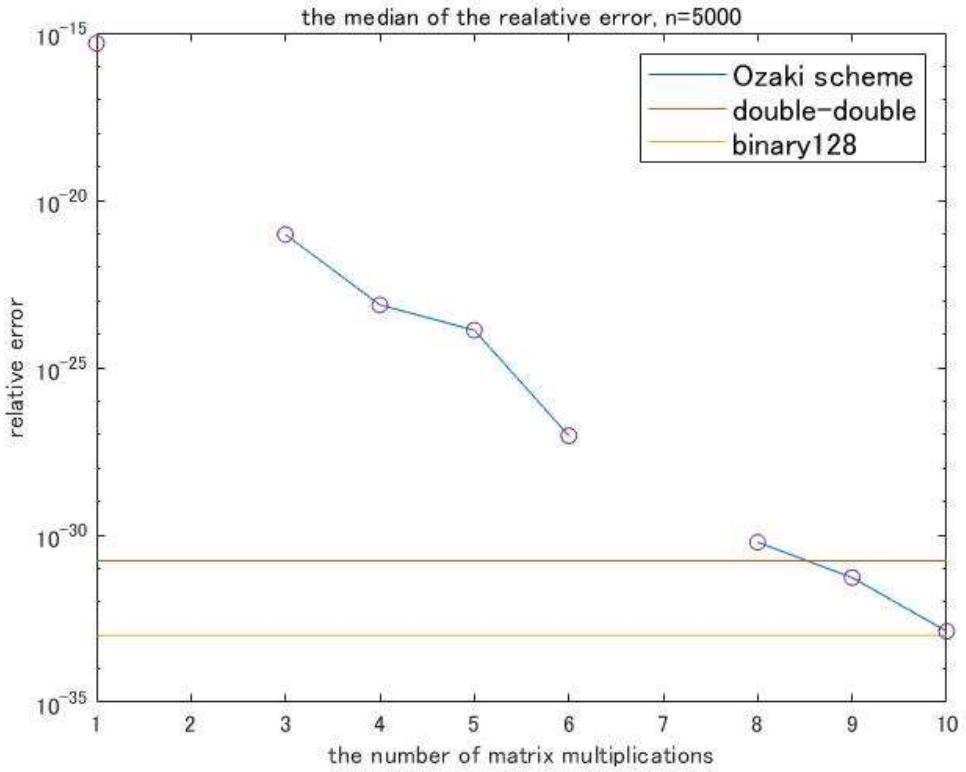
There are methods with 8, 9, 10 matrix multiplications.

For A, B,
$\frac{1}{2}k(k+1)$ *muls is required for* k *slices*

No rounding error in red color

Accuracy of Ozaki scheme





K. Ozaki, D. Mukunoki, T. Ogita:
Extension of accurate numerical algorithms for matrix multiplication based on error-free transformation,
Japan Journal of Industrial and Applied Mathematics, accepted for publication.

Result is FP128

Comparison of Accuracy

Maximum relative error for ill-conditioned problems

Slices of A	Slices of B	Num. of Muls	N＝１００	N＝１０００	N＝５０００
FP64			3.1046e+03	1.5884e+05	1.3665e+04
2	2	3	1.6933e-03	1.7492e-01	2.2076e-02
3	2	4	2.0770e-06	4.3283e-04	3.0816e-04
4	2	5	7.4756e-07	2.7860e-04	2.7903e-05
3	3	6	7.9869e-11	2.4893e-08	1.5389e-07
4	3	8	9.4641e-14	3.1482e-11	5.1181e-11
5	3	9	4.6171e-16	1.9053e-13	2.2640e-12
4	4	１０	1.2893e-16	1.1725e-14	2.4737e-14
Proposed Method			1.1040e-11	7.3810e-09	1.1443e-08

Comparison of performance

Ratio of computation time (time of dgemm is one)

Slices of A	Slices of B	Num. of muls	N=1000	N=4000
2	2	3	3 ↑	3 ↑
3	2	4	4 ↑	4 ↑
4	2	5	5 ↑	5 ↑
3	3	6	6 ↑	6 ↑
4	3	8	8 ↑	8 ↑
5	3	9	9 ↑	9 ↑
4	4	1 0	10 ↑	10 ↑
Proposed method			2.52	2.76

Efficient
・Register blocking
・Cashe blocking
・OpenMP
by Dr. Koizumi

CPU: Ryzen9 7950X
g++ e-std=c++20-O3-march=native-fopenmp
dgemm in OpenBLAS (version 0.3.27).

Double-word: Multiplication (Sloppy Ver.)

- function [zh,zl] = DW_mul(ah, al, bh, bl)
- [p1, p2] = TwoProductFMA(ah, bh);
- p2 = fma(ah,bl,p2);
- p2 = fma(al,bh,p2);
- [zh, zl] = FastTwoSum(p1, p2);
- End

7

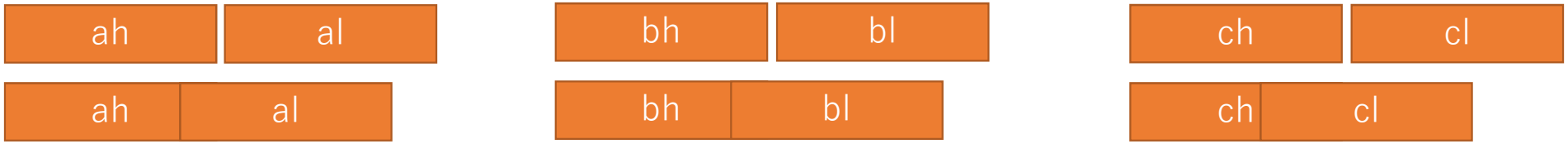
18 perations

- function [dh,dl] = DW_sum(ch, cl, zh, zl)
- [s, e] = TwoSum(ch, zh);
- e = e + cl + zl;
- [dh, dl] = FastTwoSum(s, e);
- end

11

D.H. Bailey:
QD library
https://www.davidhbailey.com/dhbsoftware/

Pair-Arithmetic:



- function [zh,zl] = PA_mul(ah, al, bh, bl)
- [zh,t] = TwoProductFMA(ah,bh);
- zl = fma(ah,bl,fma(bh,al,t));
- end

4

12 operations

- function [dh,dl] = PA_sum(ch, cl, zh, zl)
- [dh,t] = TwoSum(ch,zh);
- dl = cl + zl + t;
- end

8

M. Lange and S.M. Rump.
Faithfully rounded floating-point computations.
ACM transactions on mathematical software, 46(3), 2020.

Proposed Method

- For floating-point numbers ah, al, bh, bl, ch, cl
- $(ah+al)(bh+bl)+ch+cl \doteqdot dh + dl,$
- $dh = \mathrm{fma}(ah,bh,ch);$
- $dl = \mathrm{fl}(\mathrm{fma}(ah,bh,\mathrm{fl}(ch\text{-}dh))+\mathrm{fma}(ah,bl,\mathrm{fma}(al,bh,cl)));$

4FMA+2ADD→**6** ops

Numerical examples (Accuracy and time)

Maximum relative error for pseudo random matrices

Method	N=500	N=1000	N=4000
FP64	7.1717e-11	2.1286e-10	1.5386e-08
Proposed method	2.9222e-23	8.8718e-23	7.1396e-21
PA (dot2)	7.6952e-26	2.9847e-25	8.1142e-23
DW	3.2856e-27	3.3025e-26	1.5594e-24
Binary128	3.6391e-28	1.7560e-27	1.0017e-25

Ration of Computation time (sec)

Method	N=500	N=1000	N=4000
FP64	1	1	1
Proposed method	4.25	4.24	4.54
PA (dot2)	5.93	7.26	8.05
DW	10.5	12.7	13.9

Conclusion

Katsuhisa Ozaki , Toru Koizumi:
Fast and accurate algorithm for matrix multiplication using fused multiply-add
JSIAM Letters, 17 (2025), 13-16.

- We proposed the accurate numerical method for matrix multiplication using Fused Multiply-Add.
- Fast and little bit less accurate to pair arithmetic.
- When the size of matrix is not so large, the proposed method is faster than Ozaki scheme.
- We have already extended this approach to Triple word arithmetic and Quadruple word arithmetic.
- We have numerical results for gaming GPUs.

Thank you very much for your attention!