csDF: a double-float arithmetic library for the Cerebras CS-2

Reo Nagashima†, Akeru Nakamura†, Kai Murakami†, Ryunosuke Matsuzaki†, Daichi Mukunoki††, and Takaaki Miyajima†
†Meiji University (Japan), ††Nagoya University (Japan), e-mail:takaaki.miyajima@cs.meiji.ac.jp

明治大学 MEIJI UNIVERSITY

Overview

Recently, there have been attempts to utilize AI accelerators for scientific computing; however, these devices generally lack hardware support for double-precision floating-point arithmetic, which is essential for many scientific applications. The Cerebras CS-2 system (CS-2) delivers extremely high single-precision performance of 1.06 PFlops/s but does not support native double-precision arithmetic. To overcome this limitation and enable scientific computations requiring double precision, a software-based approach is essential. We propose csDF, a double-float (DF) arithmetic library for the CS-2 that provides DF numeric types and arithmetic operations based on QD library [1]. To demonstrate the capability of csDF, we implemented a naive pseudo-double-precision matrix multiplication using DF addition and multiplication, and measured its strong scaling performance. Our result shows 8.09 Tera DF-Flops/s, which shows the feasibility of software-based double-precision arithmetic and enables previously infeasible scientific computations.

Cerebras Wafer Scale Engine 2 (WSE-2)

The entire 300 mm wafer is used as one chip

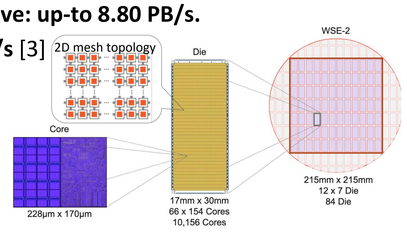
- 84 dies are interconnected with a parallel interface called Cross Scribe Line Connections.
- Process rule: TSMC 7 nm, chip area: 46,225 mm^2.
- Running at 850MHz and 16.5kW for scientific computation.

Cerebras CS-2 System

Specifications

- Total number of PEs: 853,104 (792x1078), **effective: 745,500 (750x994).**
- Total number amount of on-chip SRAM: 40 GB, **effective: 35.78 GB.**
- Memory bandwidth: 20 PB/s, **effective: up-to 8.80 PB/s.**
- SGEMM performance: **349.0 TFlops/s** [3]


WSE-2 Chip

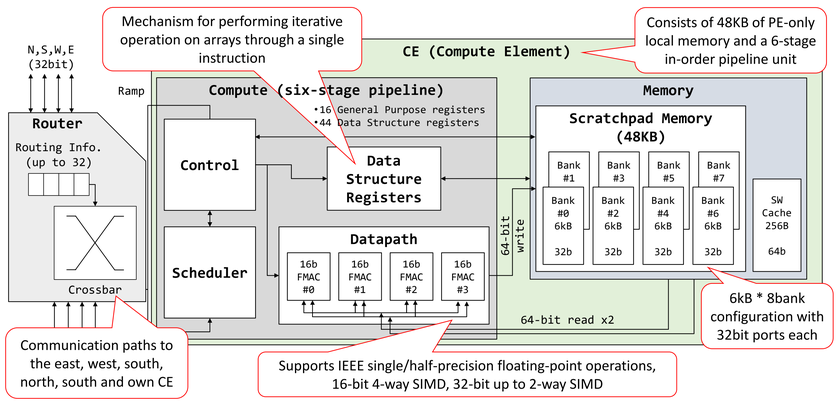


Physical structure of WSE-2

Processing Element (PE): Memory, CE, and Router

PE comes with scratchpad memory, compute element (CE) and router

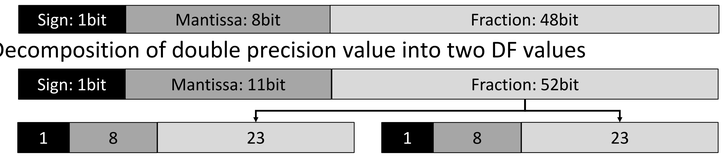
- CE and memory are connected to two 64-bit read ports and one write port.
- 853,104 Processing elements (PEs) with 2-D mesh topology.



csDF: a double-float (DF) arithmetic library

Feature of the DF type

- Represents numbers by combining two single-precision floating-point values.
- The first value holds the high part, the second holds the low part.
- The DF type achieves about 48 bits of precision from its fraction, while the 8-bit exponent defines the scale.



- Decomposition of double precision value into two DF values

Providing DF type and arithmetic operations

- csDF implements fundamental arithmetic operations such as addition, subtraction, multiplication, division, square root, commonly used transcendental functions and hyperbolic functions.
- The DF algorithm is conceptually similar to manual longhand arithmetic.
- E.g. multiplication: $(a_{hi}, a_{lo}) \times (b_{hi}, b_{lo}) \approx a_{hi} \times b_{hi} + a_{hi} \times b_{lo} + a_{lo} \times b_{hi} + a_{lo} \times b_{lo}$. [2]
- **quick_two_sum algorithm**

```
inline fn quick_two_sum(a: f32, b:f32, err: *f32) f32 {
  var s: f32 = a + b;  // s ← a ⊕ b  floating point arithmetic
  err.* = b - (s - a); // e ← b ⊖ (s ⊖ a) rounding error
  return s;}
```

- **two_prod algorithm**

```
inline fn two_prod(a: f32, b: f32, err: *f32) f32 {
  var p: f32 = a * b;
  a_df[0] = a; err_df[0] = err.*; p_df[0] = -p;
  var   a_df_dsd = @get_dsd(mem1d_dsd, .{.tensor_access = |i|{1} -> a_df[i]});
  var err_df_dsd = @get_dsd(mem1d_dsd, .{.tensor_access = |i|{1} -> err_df[i]});
  var   p_df_dsd = @get_dsd(mem1d_dsd, .{.tensor_access = |i|{1} -> p_df[i]});
  @fmacs(err_df_dsd, p_df_dsd, a_df_dsd, b); // utilize fused multiply-add
  err.* = err_df[0];
  return p;}
```

- **Addition and multiplication of two DF values**

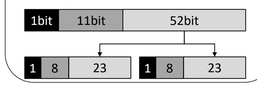
```
fn add(A: df_real, B: df_real) df_real {
  var s: f32; var e: f32;
  s = df_inline.two_sum(A.hi, B.hi, &e);
  e += (A.lo + B.lo);
  s = df_inline.quick_two_sum(s, e, &e);
  return get_df_real(s, e); }
```

```
fn mul(A: df_real, B: df_real) df_real {
  var p1: f32; var p2: f32 = 0.0;
  p1 = df_inline.two_prod(A.hi, B.hi, &p2);
  p2 += (A.hi * B.lo + A.lo * B.hi);
  p1 = df_inline.quick_two_sum(p1, p2, &p2);
  return get_df_real(p1, p2); }
```

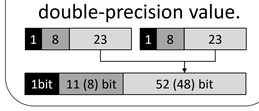
Implementation: DF Matrix Multiplication using csDF

- DF multiplication (df_real.mul) and addition (df_real.add) are used to perform the dot product (multiply and accumulation). (Not Ozaki-Scheme.)

Step 1. Split double-precision value into two single-precision values.



Step 3. Merge two single-precision values into double-precision value.



Step 2. Perform DF MatMul on CS-2
Pseudo-code of DF MatMul

```
// Import DF libraryconst
df_real = @import_module("df_real/df_real.csl");
// Loop over each row of DF matrix A
for (@range(i16, Kt)) |k|{
   // Loop over each column of DF matrix B
  for (@range(i16, Nt)) |j|{
     // Make a DF type (df_real) from two 32-bit values
    B[0] = df_real.get_df_real(B[j*Kt+k,0], B[j*Kt+k,1]);
    // Compute the dot product of A and B
    for (@range(i16, Mt)) |i|{
       // Accumulate the mult. of corresponding elements
      product[i] = df_real.mul(A[i+z], B[0]);
       C[i+x] = df_real.add(C[i+x], product[i]);
    1x = x + Mt;}
  z = z + Mt;}}
```

Host CPU Cluster — Cerebras CS-2 System

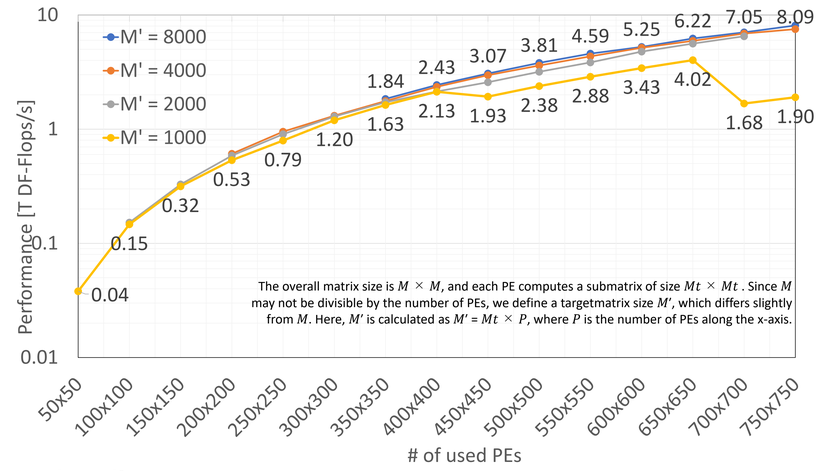
An overview of DF Matrix multiplication on CS-2

Performance evaluation of DF-GEMM on CS-2

- Performance evaluation is conducted on CS-2 System @Argonne National Laboratory, ALCF AI Testbed w/ SDK v1.2.0.
- The number of DF floating point operations (DF-Flop), 1DF =~ 20 single Flops

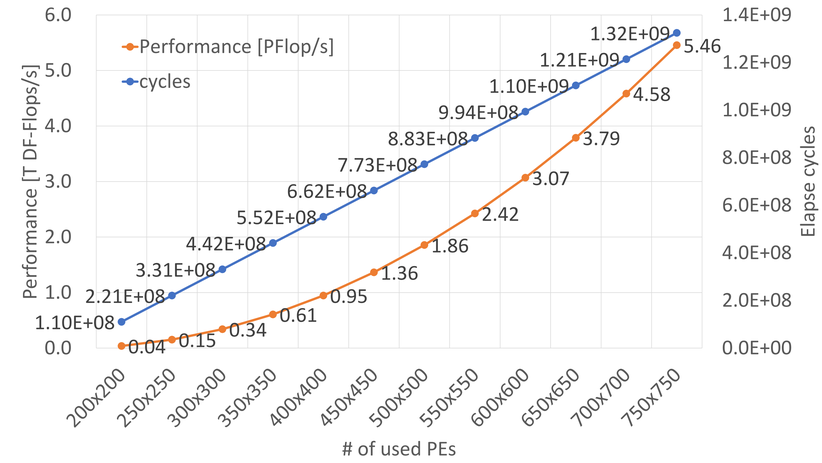
Strong scaling:

- The overall matrix size is kept in $M' \times M'$, varies M' = 1000 to 8000.
- Even when Mt becomes smaller, the performance does not saturate.



Weak scaling:

- The size of submatrices to be computed on each PE is kept in *Mt* = 22.
- Achieved ideal parallel efficiency (1.0).



Future work

- We will vectorize csDF and apply it to practical applications.

Acknowledgement and references

[1] DavidH. Bailey, XiaoyeS. Li, and Yozo Hida. 2003. QD: A Double-Double/Quad- Double Package. https://doi.org/10.11578/dc.20210416.14
[2] Y. Hida, X. S. Li and D. H. Bailey, "Algorithms for quad-double precision floating point arithmetic," Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001, Vail, CO, USA, 2001, pp. 155-162, doi: 10.1109/ARITH.2001.930115.
[3] Ryunosuke Matsuzaki, Daichi Mukunoki, and Takaaki Miyajima. 2025. Performance evaluation and modelling of single-precision matrix multiplication on Cerebras CS-2. In Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis (Atlanta, GA, USA) (SC-W '24). IEEE Press, 727–731. https://doi.org/10.1109/SCW63240. 2024.00101

This work was supported by Japan Science and Technology Agency (JST) as part of Adopting Sustainable Partnerships for Innovative Research Ecosystem (ASPIRE), Grant Number JPMJAP2341. This work was supported by JSPS KAKENHI Grant Number 24K14972. This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.