Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 15, Number 1, pages 38–43, January 2026

Error Correction Methods of Low-precision Analog Matrix Processors for High-precision Matrix Multiplications

Shoichi Hirasawa, Michihiro Koibuchi
Information Systems Architecture Science Research Division
National Institute of Informatics
{hirasawa, koibuchi}@nii.ac.jp

***Abstract*—Low-precision analog matrix processors using optical and other analog technologies are attracting attention in applications such as AI inference. However, little research has been conducted on analog computing, which enables high-precision matrix calculations using single-precision floating-point numbers (FP32) or double-precision floating-point numbers (FP64), commonly used in scientific and engineering calculations. In this report, we present a method for achieving high-precision matrix multiplications by using analog processors, applying the Ozaki scheme, which enables high-precision floating-point calculations using low-precision analog processors. This method also introduces redundancy at the algorithm level, enabling error detection and correction in the results. Evaluation results show that, assuming an optical accelerator, the proposed method has the potential to achieve effective performance of over 2.5 TFLOPS and a sufficiently low error rate for high-precision FP64 matrix calculations on the cutting-edge optical-computer model.**

***Keywords*-Auto tuning, interconnection network, approximate computing, bit flips.**

I. Introduction

Optical 8- to 16-bit low-precision matrix calculation technology is attracting attention in applications such as AI inference. LightMatter's Mars optical core can simultaneously calculate 64×64 matrices and 64-element vectors at 1 GHz. Operands can be handled as 8-bit values. The company's recently released optical accelerator [1] is controlled by an embedded OS and a RISC-V core. The future optical tensor core within it can handle 256×256 matrix calculations at a maximum speed of 2 GHz with comparable precision. As such, optical computing technology has undergone dramatic advances in recent years.

However, these optical computing units have not been used for general matrix calculations using single- or double-precision floating-point numbers, limiting their use. In this report, we propose a technology that uses analog optical computing to perform matrix calculations using single- or double-precision floating-point numbers while maintaining precision.

The proposed technology is based on two existing techniques: (1) the Ozaki scheme, which enables high-precision floating-point calculations using low-precision arithmetic units [2], [3], and (2) Algorithm-Based Fault Tolerance (ABFT), which detects and corrects soft errors using software algorithms rather than hardware [4].

The Ozaki scheme is a high-precision matrix multiplication method that divides matrices composed of high-precision values and accumulates the matrix calculation results using low-precision arithmetic units. In recent years, Basic Linear Algebra Subprograms (BLAS), which utilize the low-precision arithmetic units of GPUs, memory reduction techniques, and thread parallelization have been actively researched in the HPC field. However, the application of the Ozaki scheme to analog optical computing has not been explored. In analog computation, in addition to the issue of accuracy due to low-precision numerical representation, soft errors (bit corruption in the digital domain) caused by noise in optoelectronic and analog-to-digital conversion must also be considered. In other words, while existing digital processors have achieved a bit error rate of 10^{-12}, it would be desirable for analog optical computing units to be usable without achieving that level.

Therefore, we address soft errors in analog optical computing units by using ABFT to investigate a method for achieving high-precision matrix computations using optical circuits and evaluate its performance. In this report, we assume that the processor chip package includes general digital processor elements such as a processor core like RISC-V and a cache in addition to the optical matrix computing unit.

The contributions of this report are as follows:

- By combining the Ozaki scheme and ABFT, we demonstrate that matrix computations of arbitrary precision can be performed using an optical analog matrix computing unit.
- We achieved an effective performance of over 2.5 TFLOPS for double-precision floating-point matrix calculations of 4096×4096 using a future optical analog matrix processor.

The following section describes related research in the II section, and presents a matrix calculation method using an optical analog processor in the III section. The IV section presents evaluation results of the proposed method, and the V section provides a summary.

II. Background and Related Work

A. Optical Accelerators

Optical AI inference accelerators, such as LightMatter's Mars, are attracting attention [5]. Recently, accelerators with optical tensor cores, RISC-V cores, and embedded OSs have appeared, and they can perform multiplications of 64×64 matrices and 64×1 matrices, and 128×128 matrices and 128×1 matrices at once for numerical values with about 8-bit accuracy [1]. In this way, research on optical accelerators is moving to a level where contributions can be made not only from optical science and physics, but also from the fields of information science.

It is self-evident that the low calculation accuracy and low power consumption of optical accelerators make them suitable for AI inference processing at the edge. However, AI inference algorithms and the calculation accuracy required therein are changing daily. Therefore, in the future, there is a possibility that a mismatch will occur between the inference processing system and the calculation accuracy of the optical accelerator, which is its infrastructure. Furthermore, to broaden the applications of optical accelerators, it is desirable to use them in matrix calculations with general precision, such as single-precision floating-point numbers, which are used in scientific and technological calculations.

B. Ozaki scheme

The Ozaki scheme [2] is a matrix multiplication algorithm that uses real-precision arithmetic that is lower than the target precision of the matrix multiplication. For example, the Ozaki scheme can be used to calculate double-precision matrix multiplication using single-precision or integer arithmetic. Using the Ozaki scheme, matrix calculations of arbitrarily high precision numbers can be performed on low-precision floating-point units [6] or on low-precision integer units [3].

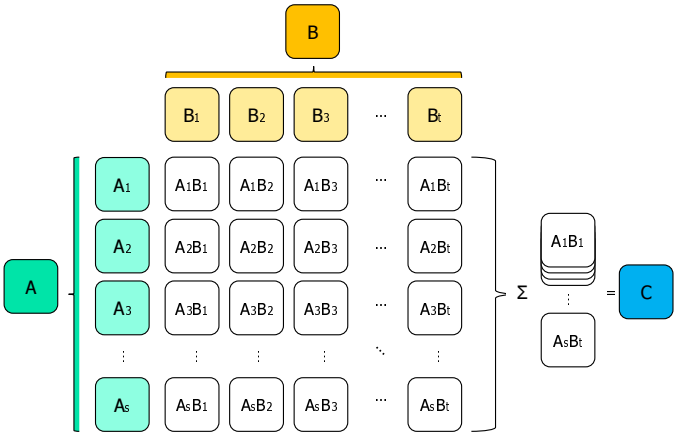

Fig. 1. Matrix multiplication using the Ozaki scheme

Figure 1 shows how to calculate the matrix multiplication $C = A \times B$ using the Ozaki scheme. Matrices A and B are each split into multiple submatrices, and matrix operations are performed on each submatrix to obtain multiple result matrices (A_iB_j). The resulting matrices (A_iB_j) are summed to obtain the desired matrix C.

C. Algorithm Based Fault Tolerance (ABFT)

In digital system communications, as a method of dealing with soft errors (temporary bit corruption), the sender adds an ECC (Error Checking and Correcting) code to the original data before transferring it, allowing the receiver to detect and correct soft errors within a certain number of errors.

Similarly, in Algorithm-Based Fault Tolerance (ABFT), an array for error detection and correction is added to the original array in advance in matrix calculations, making it possible to detect and correct soft errors within a certain number of errors from the calculation results [4], [7]. ABFT is characterized by its ability to provide reliability for the entire hardware process, including arithmetic units and memory access, because it operates at the application level.

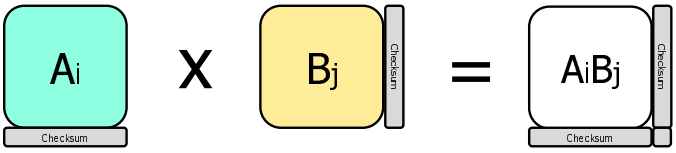

Fig. 2. Matrix multiplication with checksum

For example, as shown in Figure 2, by adding a checksum vector row to matrix A_i and adding a checksum vector column to matrix B_j and then performing a matrix multiplication, we obtain a matrix A_iB_j with the checksum vector row and checksum vector column added. The resulting matrix A_iB_j has fault tolerance, allowing for double-bit error detection and single-bit error correction.

III. Matrix Multiplication Method Using Analog Processor

A. Assuming optical analog matrix processor

In this study, we target an optical analog matrix processor(Figure 3) that performs matrix calculations $A \times B$, where $A = N \times N$, $B = N \times 1$, at once. Calculations other than matrix calculations that arise in ABFT, Ozaki scheme, etc., are performed using existing digital circuits.

Hereafter, we will refer to this optical analog low-precision matrix processor as an optical tensor core (TC), following the notation used for GPU and TPU processors.

The accuracy of the optical-electrical conversion and analog-to-digital conversion of numbers is assumed to support P-bit representation. However, we assume that the bit error rate of the optical analog matrix processor is less than the error-free environment of classical computers (bit error rate 10^{-12}).

B. ABFT based error correction method

The objective is to perform matrix calculations $C \times D$, where $C = M \times M$, $D = M \times M, M \geq N$, using an optical analog

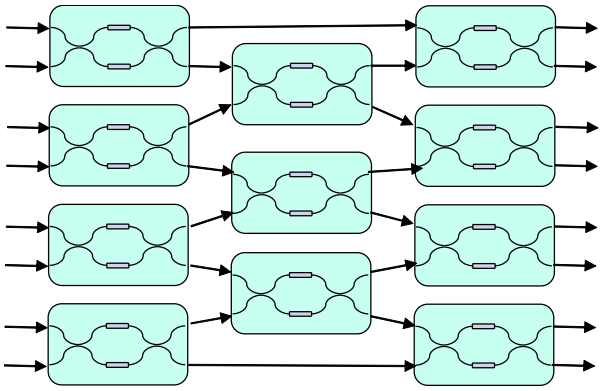

Fig. 3. Matrix-vector multiplication circuit using light (Mach-Zehnder interferometer)

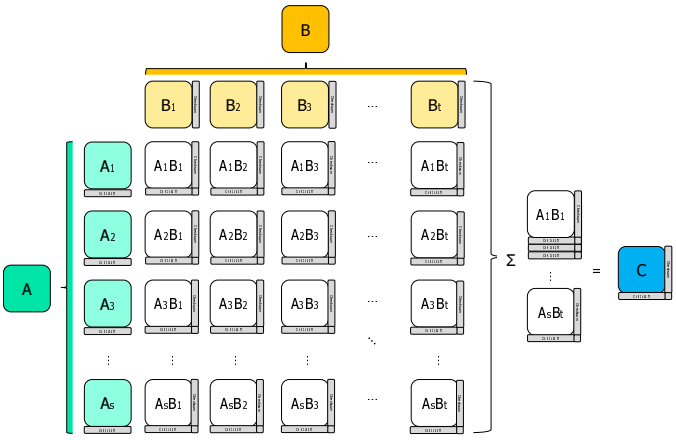

Fig. 4. Ozaki scheme with checksum

Algorithm 1 Split_A_with_Checksum

function SPLIT_A_WITH_CHECKSUM(A, s)
  $A_1, A_2, \dots A_s \leftarrow SplitFP(A, s)$
  for i = 1…s **do**
    $A'_i = \begin{pmatrix} A_i \\ e^\top A_i \end{pmatrix}$
    return $A'_1, A'_2, \dots A'_s$
  end for
end function

Algorithm 2 Split_B_with_Checksum

function SPLIT_B_WITH_CHECKSUM(B, t)
  $B_1, B_2, \dots B_t \leftarrow SplitFP(B, t)$
  for j = 1 …t **do**
    $B'_j = \begin{pmatrix} B_j & B_j e \end{pmatrix}$
    return $B'_1, B'_2, \dots B'_t$
  end for
end function

matrix processor. Each element of the matrix is a floating-point number such as FP32.

The proposed calculation method is executed as follows.

1) (Digital circuit) Using the Ozaki scheme in fast mode, divide each element of matrices C and D into P-bit integers.
2) (Digital circuit) For ABFT calculation, add a row consisting of only element 1 to the divided matrix C, and a column consisting of only element value 1 to the matrix D, to obtain $C'andD'$, respectively.
3) (Analog optical circuit) Execute the divided matrix calculation $C' \times D'$ using the Ozaki scheme.
4) (Digital circuit) Perform error correction using ABFT on the matrix calculation result obtained in the previous step.
5) (Digital circuit) Store the matrix calculation result in the original floating-point number using the Ozaki scheme.

In Steps 2 and 3, ABFT applies a simple algorithm [4]. Up to two errors can be detected or one error can be corrected for each partitioned matrix. In Steps 1˜3, time-series pipeline processing is used to efficiently execute matrix calculations $A \times B$. In Step 4, parallel execution is also possible using wavelength multiplexing on the same optical tensor core.

We will provide a simple example of how ABFT works in the proposed method. Figure 5 shows an example of error correction using the added checksum. In this example, assume that the $(2, 2)$ element of the result matrix, which should correctly be calculated as 50, is garbled to 48, as shown in red. In this case, we can see that the (2,2) element is garbled from $22 + 48 \neq 72$ in the second column and $43 + 48 \neq 93$ in the second row. Using this, we can calculate $72 - 22$ in the second column or $93 - 43$ in the second row of the result matrix to obtain the correct result, 50.

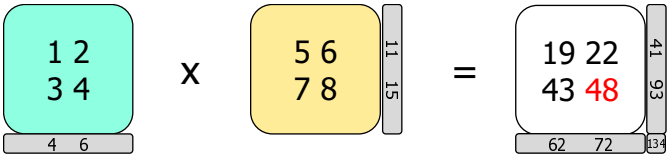

Fig. 5. Example of error correction using checksum

Algorithm 3 Compute_C_with_Checksum

function COMPUTE_C_WITH_CHECKSUM(A, s, B, t)
  $A'_1, A'_2, \dots A'_s \leftarrow Split\_A\_with\_Checksum(A, s)$
  $B'_1, B'_2, \dots B'_t \leftarrow Split\_B\_with\_Checksum(B, t)$
  $C = 0$
  for i = 1…s **do**
    for j = 1 …t **do**
      $C'_{ij} = A'_i B'_j$
      $C+ = Correct\_Error(C'_{ij})$
    end for
  end for
  return C
end function

C. Re-execution based different checksum calculation method

The proposed method using ABFT involves including the checksum in the matrix for calculation, which may result in significant overhead relative to the number of TC calculations. Therefore, we propose an different checksum calculation method that collects the checksums of each submatrix to form a checksum matrix and calculates this matrix only once in the TC (Figure 6). This checksum calculation method eliminates the space overhead of including the checksum in a matrix when calculating the submatrix, potentially reducing the overhead of the number of TC calculations.

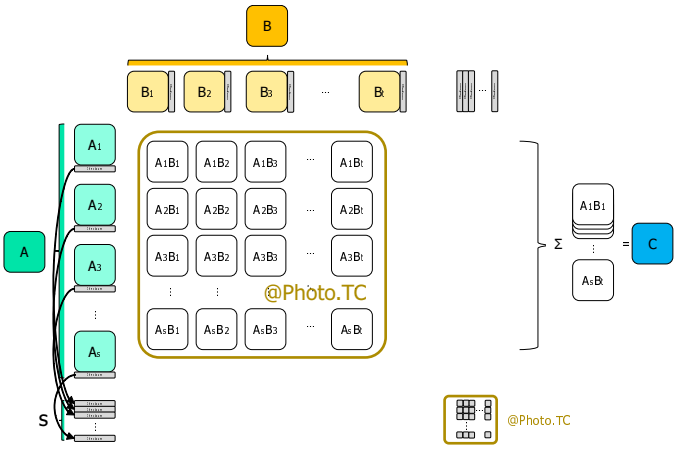

Fig. 6. Optimizing Checksum Calculation for TC

In this method, the calculation of the checksum row and column of the submatrix is omitted. Instead, only the dot product of the checksum row vector and the checksum column vector is calculated. The obtained dot product is equal to the norm of the submatrix product. Therefore, by comparing the norm of the submatrix product with the dot product value, it is possible to detect errors in the submatrix product.

An example is shown in Figure 7. Our method omits the calculation of the checksum vectors $\begin{pmatrix}62 & 72\end{pmatrix}$ and $\begin{pmatrix}41\\93\end{pmatrix}$ and instead calculates only the inner product value 134. This value is equal to the norm of the partial matrix product of $\begin{pmatrix}19 & 22\\43 & 50\end{pmatrix}$, which is $134 = 19 + 22 + 43 + 50$, making error detection of the partial matrix product possible.

However, since errors cannot be corrected using only the product of the checksum matrix, this method is limited to error detection. Error correction can be achieved by re-executing the process when an error is detected.

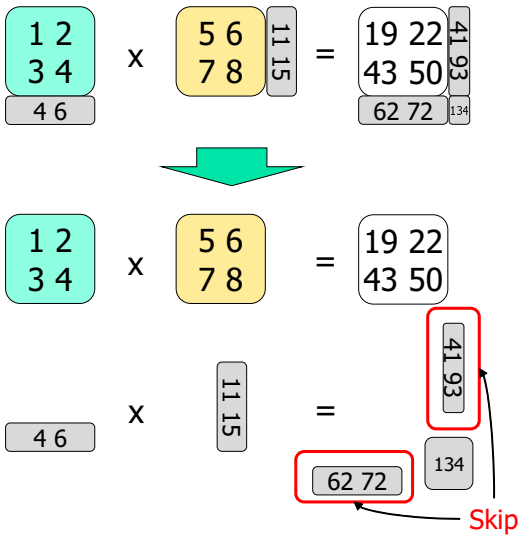

Fig. 7. Example of different checksum calculation method

TABLE I
PARAMETER VALUES USED IN EVALUATION.

N	64, 128, 256(default)
P	8
M	8, 16, ..., 4096
Bit error rate of optical analog operation	$10^{-15}, 10^{-14} \ldots 10^{-3}$
Operating frequency of digital circuit	1GHz

IV. Evaluation

A. Assumptions

The parameter values used in this evaluation are shown in Table I. An ideal environment is assumed for input and output from the digital circuit to the optical tensor core, with no stalls. In other words, a matrix can be input to the optical tensor core every clock cycle. Each value in the target matrix calculation is a double-precision floating-point number (FP64). Furthermore, the optical tensor core supports P-bit representation for input values, and $2P$-bit representation for output. This is to prevent overflow in the multiplication results. For each divided matrix, the optical tensor core performs INT8 ($P = 8$) matrix calculations.

We evaluate the following metrics.

- Number of calculations performed using the optical tensor core (= number of inputs)
- Percentage of errors that could not be corrected using ABFT(Subsection III-B) or Re-executions(Subsection III-C)
- Effective performance when using an optical tensor core

B. Number of optical tensor core executions

When performing FP64 matrix operations (multiply-and-accumulate calculations) using the proposed method, We counted the number of calls to the optical tensor core required, taking into account this ABFT, in addition to the number of calls to the integer matrix operations in [3]. The number of calls to the optical tensor core is shown in Figure 8.

Note that in digital circuits, the operations required for the Ozaki scheme can be realized with shift and addition operations if the original matrix is FP32 or FP64 and the optical tensor core is a 4-, 8-, or 16-bit floating-point number.

In Figure 8, the horizontal axis represents the FP64 input matrix size (M), and the vertical axis represents the number of executions of the optical tensor core. The number of

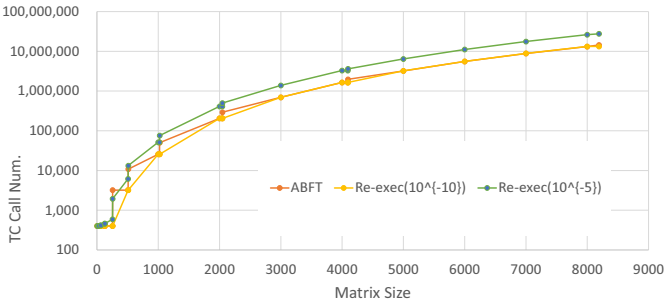

Fig. 8. Number of executions of the optical tensor core required to perform the matrix calculation of ABFT(Subsection III-B) and Re-executions(Subsection III-C) for BER of 10^{-10} and 10^{-5}.

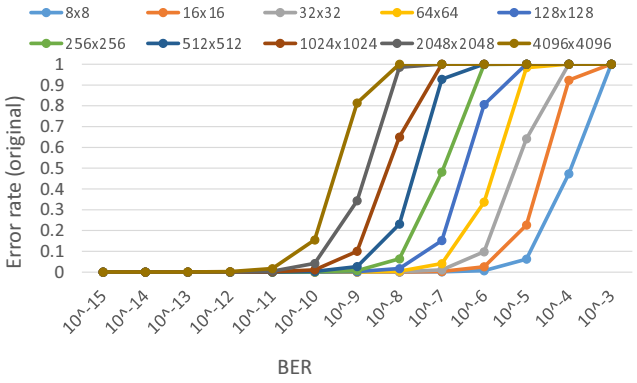

Fig. 9. Failure rate of computations using the optical tensor core with the Ozaki scheme without error correction

retries depends on the error rate. Therefore, the figure shows the cases where the error rate is 10^{-10} and 10^{-5}. ABFT performs redundant calculations regardless of whether errors exist or not, without retransmission. Therefore, the number of executions does not depend on the bit error rate. Figure 8 shows that the overhead of error correction by ABFT may increase the number of executions of the optical tensor core, but in many cases this is extremely small. Furthermore, as expected, we can see that the number of executions of the optical tensor core is determined by the matrix size that can be executed at one time (N).

C. Computation failure rate

Here, we analyze the matrix calculation failure rate in the optical tensor core relative to the bit error rate. The computation failure rate refers to the probability that at least one uncorrected value remains when all matrix calculations are completed. The analysis method was to calculate the following when bit errors occur at a uniformly specified bit error rate in all divided matrices input to the optical tensor core: (1) The rate at which an error occurs in the final calculation result (without error correction) (2) The rate at which an error occurs in the final calculation result due to three or more bit errors occurring in the divided matrices, resulting in an error that cannot be corrected (with error correction)

Figure 9 shows the computation failure rate when using an optical tensor core with the Ozaki scheme without error correction. Figure 9 shows that error correction is sufficient when the bit error rate is equal to or less than the 10^{-12} imposed on each device by classical computers. On the other hand, for matrix sizes of 10^{-10} or larger, the failure rate dramatically increases when the matrix size is 4096×4096, indicating the need for ABFT.

Next, Figure 10 shows the failure rate of computations using the optical tensor core with the Ozaki scheme with error correction for each submatrix $(A_i B_j)$. Figure 10 shows that using error correction reduces the probability of errors in the computation results regardless of matrix size. Specifically, it can be said that a bit error rate of 10^{-5} is sufficient.

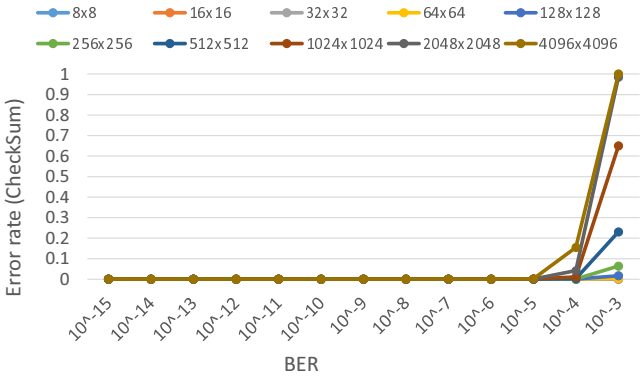

Fig. 10. Failure rate of computation using the optical tensor core with the Ozaki scheme with ABFT based error correction for each submatrix

D. Effective performance of optical tensor cores

Figure 11 shows the maximum effective performance of the proposed method assuming a future 256 optical tensor core [1]. The evaluation was calculated using the following.

$$Performance = num.\ FP64\ OPS / \frac{num.\ TC\ calls}{num.\ TC/sec}$$

Figure 11 shows that without error correction, for sufficiently large matrix sizes (specifically, N or larger), effective performance of over 2.5 TFLOPS can be achieved on 256 TC core. This indicates that if optical tensor cores can be used appropriately in matrix calculations, computational performance surpassing that of GPUs can be achieved. Furthermore, even with error correction, performance comparable to that without error correction can be achieved. On the other hand, the effective performance for small matrix sizes remains low due to the overhead of calls.

Furthermore, while the ABFT method has a lower effective efficiency than re-execution when the error rate is low (e.g. 10^{-10}), it can achieve higher effective performance compared to re-execution when the error rate is high (e.g. 10^{-5}).

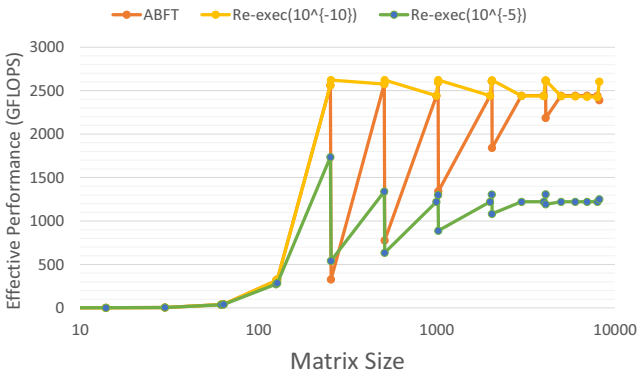

Fig. 11. Maximum effective performance (GFLOPS) using optical tensor cores of ABFT(Subsection III-B) and Re-executions(Subsection III-C) for BER of 10^{-10} and 10^{-5}.

V. Summary

In this report, we applied the Ozaki scheme, which enables high-precision floating-point calculations using low-precision arithmetic units, to demonstrate a method for achieving high-precision matrix calculations using light. This method also enables error detection and correction in optical calculation results by introducing redundancy at the algorithm level.

Evaluation results show that, assuming LightMatter's optical tensor core, the proposed method can achieve an effective performance of up to 2.5 TFLOPS or more in double-precision floating-point calculations. This far exceeds the 124 GFLOPS of the NVIDIA RTX A2000 GPU in 2022.

We also found that ABFT, which adds a checksum, can reduce the error rate in calculation results, with minimal overhead. These results demonstrate the feasibility of using the Ozaki scheme and ABFT to achieve high-precision matrix calculations using optical circuits with high performance.

Future challenges include the following:

- Design of a digital circuit to control the input/output data flow of the optical tensor core to maximize its performance.
- A method to maximize performance by co-designing three elements: the introduction of block floating-point numbers, the Ozaki scheme partitioning method, and ABFT.

VI. Acknowledgments

This research was supported by Grant-in-Aid for Scientific Research JP22H05193 and JST CREST JPMJCR24R3.

References

[1] S. Ahmed, R. Baghdadi, and M. e. a. Bernadskiy, "Universal photonic artificial intelligence acceleration." *Nature 640*, pp. 368–374, 2025.

[2] K. Ozaki, T. Ogita, S. Oishi, and S. M. Rump, "Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications," *Numerical Algorithms*, vol. 59, pp. 95–118, 2012.

[3] H. Ootomo, K. Ozaki, and R. Yokota, "Dgemm on integer matrix multiplication unit," *The International Journal of High Performance Computing Applications*, vol. 38, no. 4, pp. 297–313, 2024.

[4] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput.*, vol. 33, no. 6, p. 518?528, Jun. 1984. [Online]. Available: https://doi.org/10.1109/TC.1984.1676475

[5] C. Ramey, "Silicon photonics for artificial intelligence acceleration : Hotchips 32," in *2020 IEEE Hot Chips 32 Symposium (HCS)*, 2020, pp. 1–26.

[6] D. Mukunoki, K. Ozaki, T. Ogita, and T. Imamura, "Dgemm using tensor cores, and its accurate and reproducible versions," in *International Conference on High Performance Computing*. Springer, 2020, pp. 230–248.

[7] A. Bouteiller, T. Herault, G. Bosilca, P. Du, and J. Dongarra, "Algorithm-based fault tolerance for dense matrix factorizations, multiple failures and accuracy," *ACM Trans. Parallel Comput.*, vol. 1, no. 2, Feb. 2015.