



Towards Scalable Quantum Simulation on Wafer-Scale Engines

Ben Phillips, Dylan Kneidel, Alvir Nobel, and Esam El-Araby

Department of Electrical Engineering and Computer Science (EECS), University of Kansas (KU)



Introduction and Motivation

- Quantum computers provide **exponential computational benefits** over classical computers.
- Existing quantum computers' efficiency is limited by **high noise** and **low qubit count**.
- Specialized hardware** platforms like **GPUs** and **FPGAs** are preferred for **classical simulation** to validate quantum algorithms [1-4].
- We propose an optimized method for the complex **General Matrix-Vector product (GEMV)** operation that:
 - Uses **Cerebras Wafer Scale Engine (WSE)** architecture
 - Facilitates **scalable**, general purpose **quantum simulation**
- We experimentally demonstrate:
 - The **scalability** of the proposed method using results from **larger-scale quantum circuits**
 - The **suitability** of Cerebras Wafer Scale Engines (**WSEs**) for scalable **quantum simulations**

Related Work

Cerebras Architecture Deep Dive [5]

- Introduces distributed built-in **floating point multiply accumulate (FMAC)** instructions on WSE architecture
- Emphasizes **high-bandwidth**, **low latency** nature of WSE architecture
- Utilizes **vector weight streaming** to enable all AI model sizes on 1 chip

Hardware Acceleration for Quantum Simulation

- Qibo [11]**
 - Utilizing multi-threading CPUs, single GPUs, and multi-GPU accelerators
- Multi-Shot [2]**
 - Using batch execution and shot-branching to optimize multi-shot quantum computing simulations on GPUs
- QuEST [12]**
 - Hybrid GPU-accelerated simulator designed for universal quantum circuits that can handle pure and mixed states
- Reconfigurable Emulation [4]**
 - Reconfigurable emulation of quantum algorithms focusing on achieving high precision and high throughput

WSE Applications

- Stencil-Based Computation Codes [6]**
 - WSE outperforms 4 Nvidia V100 GPUs by 2.5 and 2 Intel Xeon Platinum CPUs by around 114 times for solving Laplace's equation
- Multi-Dimensional Seismic Processing with Algebraic Compression [7]**
 - Accelerates tile low-rank matrix-vector multiplications (TLR-MVMs), assuming sparsity
 - Scaling only achieved by utilizing additional hardware, requiring a minimum of 6 CS-2 WSEs and tested at a maximum of 48
- Fast Stencil-Code Computation on a Wafer-Scale Processor [8]**
 - Numerically solves PDEs without designing to scale, using 65% of CS-1
 - Uses half-precision (16-bit) floats for hardware sparsity optimizations
- Cerebras-GPT [9]**
 - Compute-optimal language models ranging from 111M to 13B parameters, trained on the Eleuther Pile dataset
- Fast Fourier Transforms [10]**
 - Up to 3-dimensional arrays on the Cerebras CS-2 system, which uses a wafer-scale engine (WSE) with around 850,000 processing elements

Experimental Setup

Analysis Platform

- Cerebras' WSE simulator used to run and profile our method
 - Counts clock cycles required for entire GEMV operation
 - Simulates variable number of PEs
 - Can't simulate large (WSE-3 scale) PE grids
- We demonstrate scalability by **varying PE grid size**
 - Compare results for 1 PE up to 128 x 82 PEs (1.3% WSE-3)
 - WSE-3 aspect ratio of 1.555:1 maintained
- Results compared against Qiskit Aer simulator
- Input Data**
 - Input matrix and vector created with random values
 - QHT circuits created for realistic application Qiskit [11]
 - Gray-scale images of size 8x8 to 64x64 pixels were used

Relevant Variables

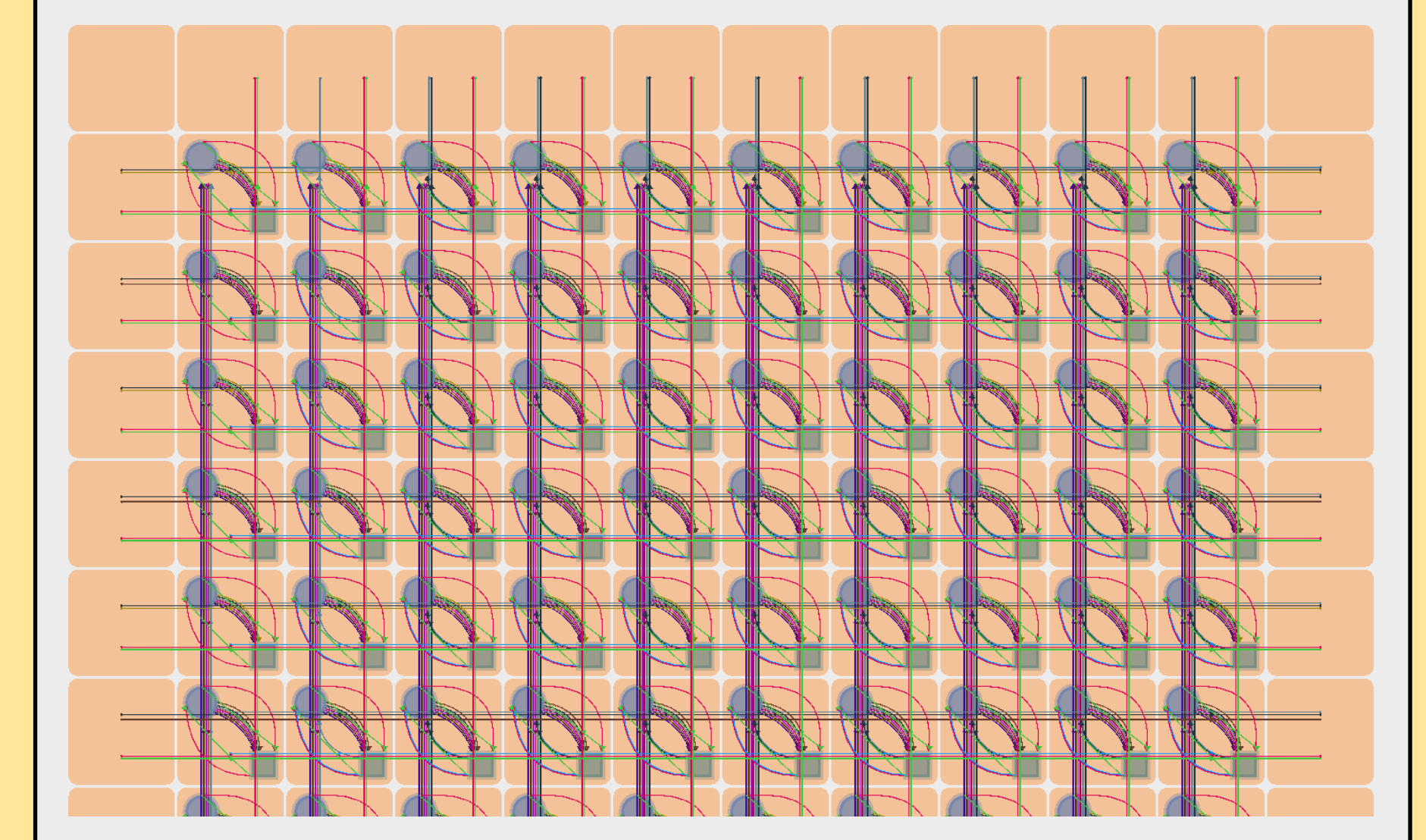
- PE grid size - demonstrates scaling and distribution
- Memory availability - restricted to match smaller WSE size
- M_p & N_p - controls job size and aspect ratio
- Input channels - always 16 used, except 1 for 1x1 PE
 - PE grid height is always a multiple of 16 to minimize IO irregularities influencing performance
 - More than 100 actually available on WSE-3
- Algorithm - matrix buffering / vector buffering differences
- Qubit count (n) - number of qubits simulated

Performance Metrics

- Throughput (GB/s) - calculated from clock cycle count
- Speedup - parallel advantage over a serial processor

Cerebras SDK Simulator Debug View

Dataflow view of algorithm on scaled-down WSE



Background

Fundamentals of Quantum Computing

- Quantum computers leverage **superposition** and **entanglement** of quantum states for advantage over classical computers in certain workloads.
- Near-term noisy-intermediate-scale-quantum (NISQ)** hardware possesses strict decoherence constraints where quantum states break down after a certain amount of time.
- Representation of an **n-qubit quantum state vector**:

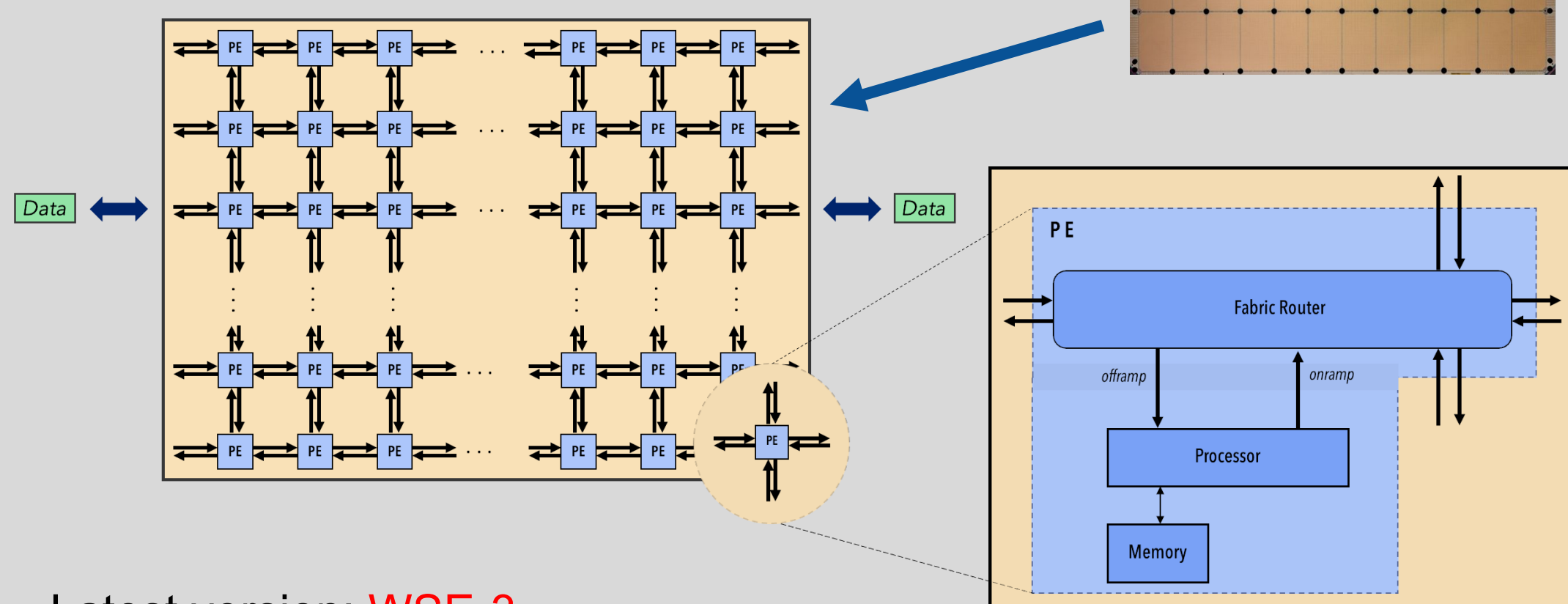
$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle = \begin{bmatrix} c_0 \\ \vdots \\ c_{2^n-1} \end{bmatrix} \quad \langle\psi|\psi\rangle = \sum_{i=0}^{2^n-1} |c_i|^2 = 1 \quad c_i \in \mathbb{C}$$
- Quantum operations** act on quantum states and can be represented as **unitary matrices** or quantum "gates".
 - All quantum gates can be decomposed into fundamental **single-qubit rotation** and **two-qubit CNOT gates**.
- Quantum circuits should be optimized in terms of **circuit depth** and **gate count** to avoid decoherence and gate errors.
- All quantum operations can be **simulated using matrix multiplication (GEMV)** between the circuit matrix and state vector.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_M \end{bmatrix} \leftarrow \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & a_{M,3} & \cdots & a_{M,N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} \quad y_r = \sum_{c=1}^N a_{r,c} x_c, r \in [1, M]$$

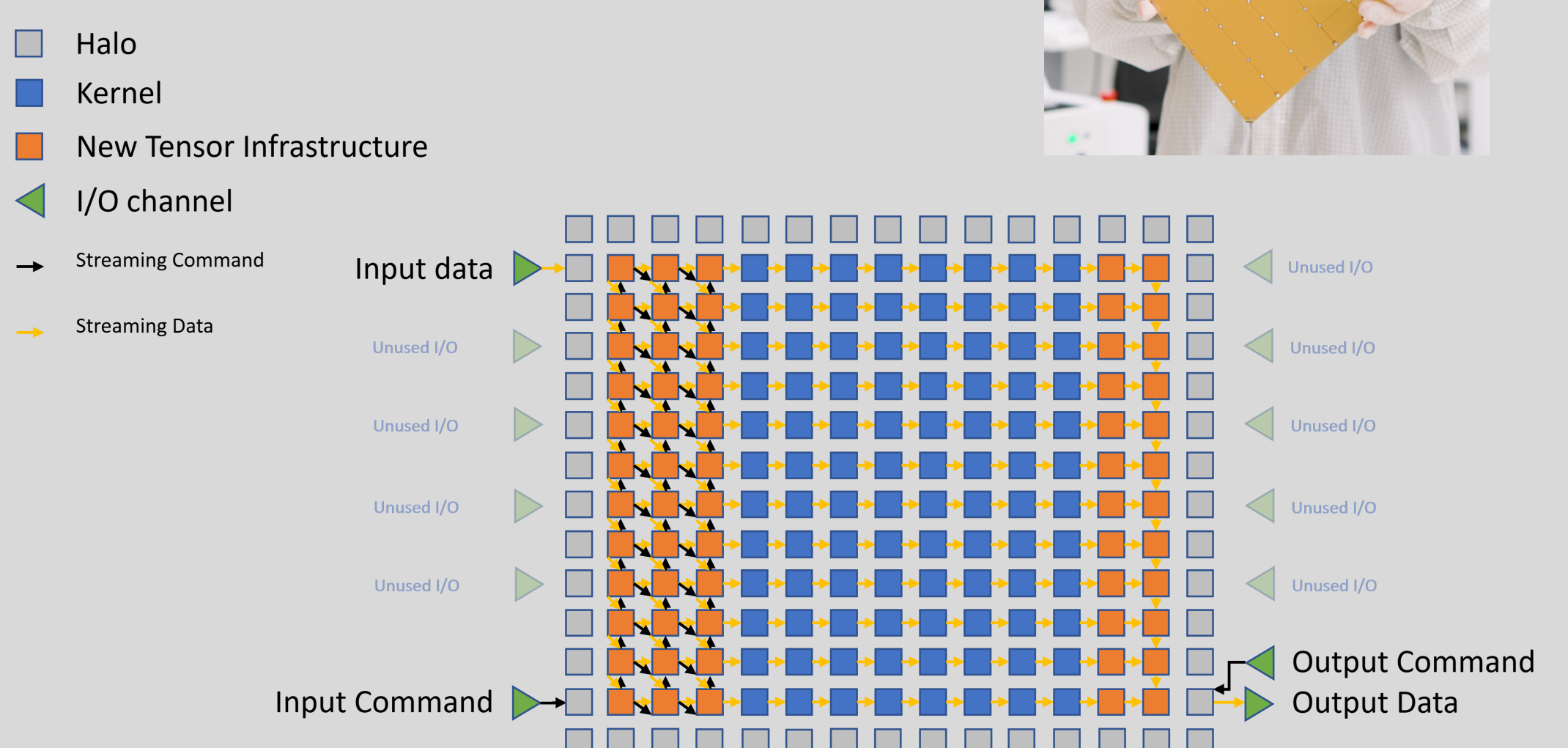
$$\vec{y} \leftarrow \mathbf{A} \cdot \vec{x}$$

Cerebras Wafer Scale Engine (WSE)

- AI processor** containing a grid (fabric) of **processing elements (PEs)**:
 - Individual 48kB memory per PE
 - No central memory** or control hardware
 - Communicate only with 4 neighboring PEs
 - One-cycle memory access and communication
- Dataflow architecture** with high internal bandwidth
- Designed for tensor operations in AI workloads



- Latest version: **WSE-3**
 - 762 x 1176 = **896,112 PEs**
 - 1.1 GHz** global clock frequency
 - Limited to **single precision** (32 bits) for **floating-point** values



- IO occurs at the edges of the wafer
 - 124 channels on WSE-3
 - Current SDK IO framework limits usage to 16 channels

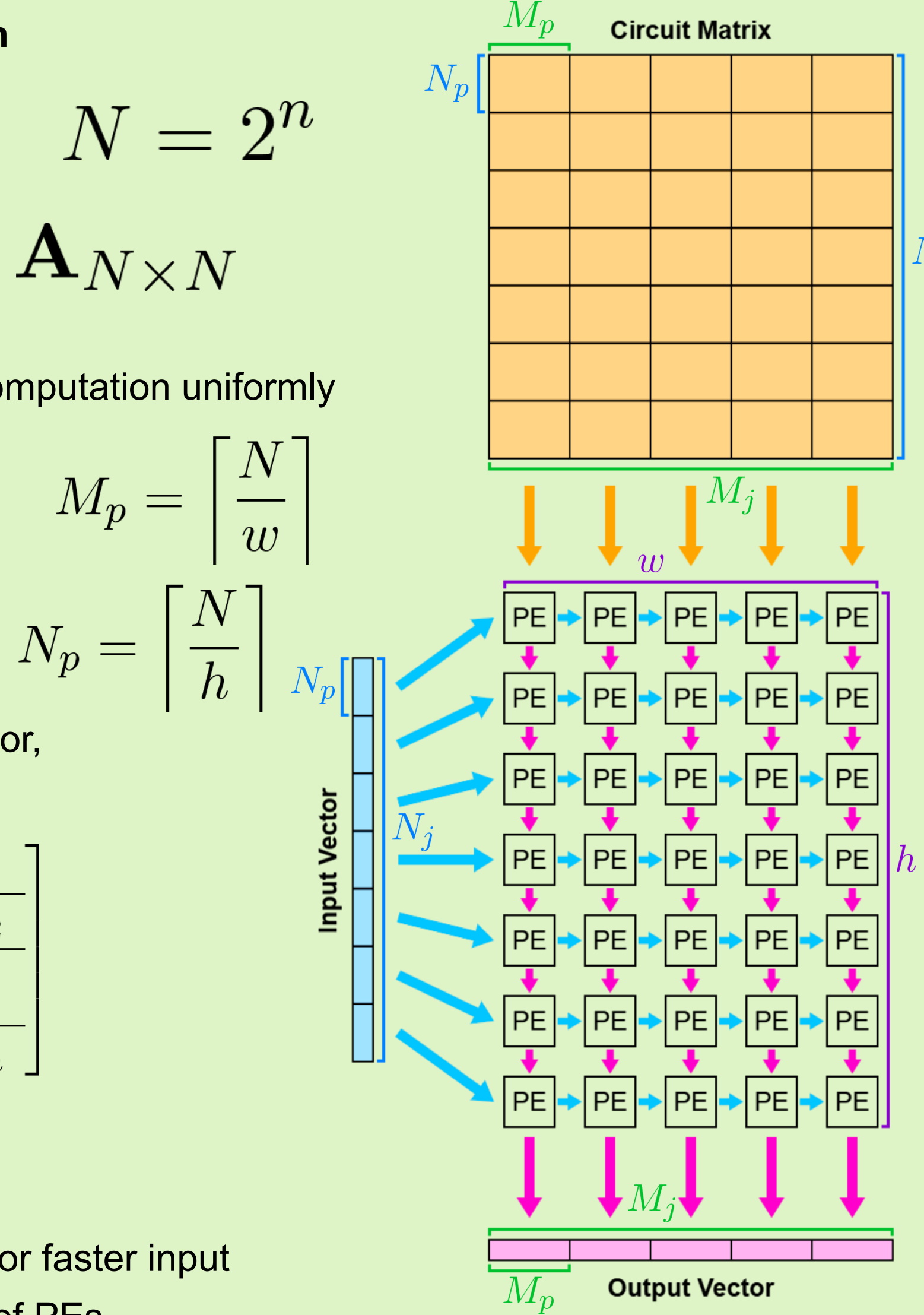
Proposed Methodology

Compute Distribution for Maximum Parallelization

- With n qubits:
 - Input and output vectors have $N = 2^n$ values
 - Circuit matrix has N^2 values
 - Each vector value accessed N times
 - Each matrix value accessed once
- Distribute matrix elements uniformly \rightarrow distribute computation uniformly
 - WSE grid has $w \times h$ PEs
 - Partition matrix into $w \times h$ blocks of size $M_p \times N_p$

$$M_p = \begin{bmatrix} N \\ w \end{bmatrix}$$
 - Extra space is padded with zeros
 - Input vector has h blocks of N_p values
 - Output vector has w blocks of M_p values
 - Each PE gets one block of matrix and input vector, producing one partial block of output vector

$$\begin{bmatrix} \vec{y}_1 \\ \vec{y}_2 \\ \vdots \\ \vec{y}_w \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,h} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,h} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{w,1} & \mathbf{A}_{w,2} & \cdots & \mathbf{A}_{w,h} \end{bmatrix} \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_h \end{bmatrix}$$



- Assign blocks to PEs spatially
 - Input vector blocks **shared across row of PEs**
 - Row/column convention flipped from matrix for faster input
 - Output vector block accumulated down column of PEs
 - Bottom row outputs final collected result

$$\vec{y}_{x_f} = \sum_{y_f=1}^h \mathbf{A}_{x_f,y_f} \vec{x}_{y_f}, x_f \in [1, w]$$

Algorithm on PEs

- Matrix and input vector are two inputs
- Must buffer one, then compute when the other arrives
- Two variants of algorithm explored
 - Buffer matrix, compute by vector data
 - Great for multiple input vectors
 - Buffer vector, compute by matrix data
 - Better structure for specialization

Three stages to algorithm:

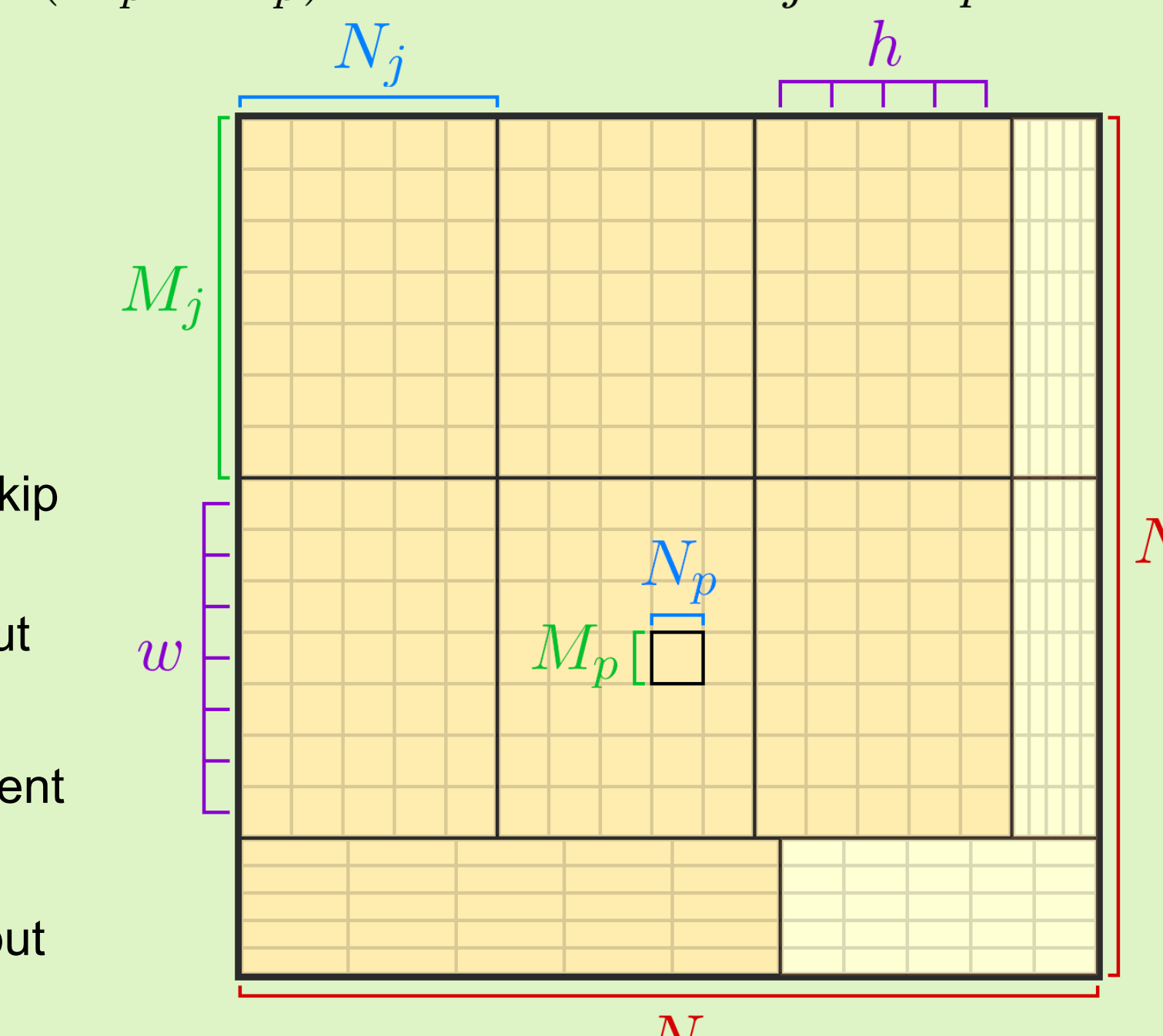
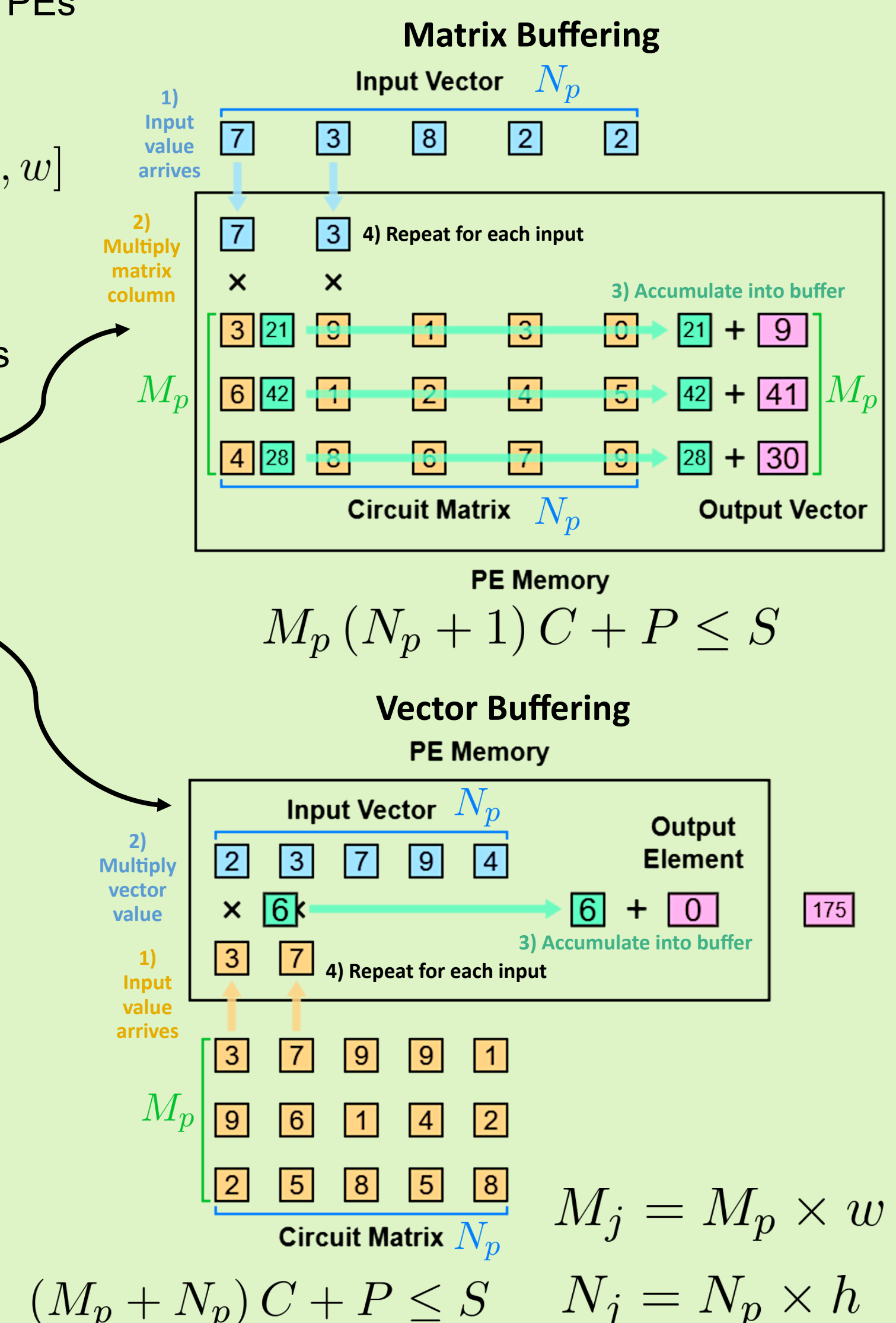
- Input matrix/vector and buffer
- Input vector/matrix and compute
- Collect output vector and output

Scaling Beyond Memory Constraints

- Memory capacity $S = 48\text{kB}$
- Complex value size $C = 8\text{B}$ (two 32-bit floats)
- Program size $P \approx 5\text{kB}$
- Space for about 5500 buffered values
 - Matrix buffer too small for 17 qubits on WSE-3
 - Vector buffer too small for 23 qubits

Partition matrix again into multiple "jobs"

- Jobs **run in sequence**, in **any order**
- Job dimensions $M_j \times N_j$ **configurable**
- Optimization opportunities while scheduling jobs
 - Choose $M_j \times N_j$ to minimize zero padding
 - Align across rows to skip collection & output
 - With vector buffering, align down columns to skip vector input
 - Step 3 output can overlap with next step 1 input
- Jobs **allow adjustable** $M_j : N_j$ **aspect ratio**
 - Larger M_j : Fewer times input vector needs to be sent
 - Larger N_j : Less partial result data to collect & output
 - Generally better for vector buffering



PE Grid Size

- Data for **1x1 PE** was **extrapolated** for $n > 8$
 - Simulator SDK struggled to handle qubit count larger than 8
 - Throughput stayed constant and close to 98MB per second
- At 12 qubits, 128x82 PE grid has a **671x speed advantage**
- Continuous performance improvement with more qubits
 - Better **parallel advantage** with **larger circuits**
- Improvements with more PEs
 - Significant given constant input channels
 - Larger variance with more qubits

Memory Usage

- Larger allocation improves performance
 - Larger buffers **reduce job count** and **associated costs**
- Less noticeable at the high end
 - 48kB is suitable for methodology on WSE-3

M_p/N_p Ratio (Matrix Buffering)

- 1:1 (square) and 1:4 perform best
- Much higher N_p performs poorly
- Larger M_p potentially scales better than 1:1
 - Slower at low qubit counts, but just as fast at $n = 12$
 - Deserves a more detailed future investigation

M_p/N_p Ratio (Vector Buffering)

- Performance is strictly better with larger N_p
 - Benefits most from buffer re-use
 - Likely related to slow output on WSE
- More extreme ratios should be explored

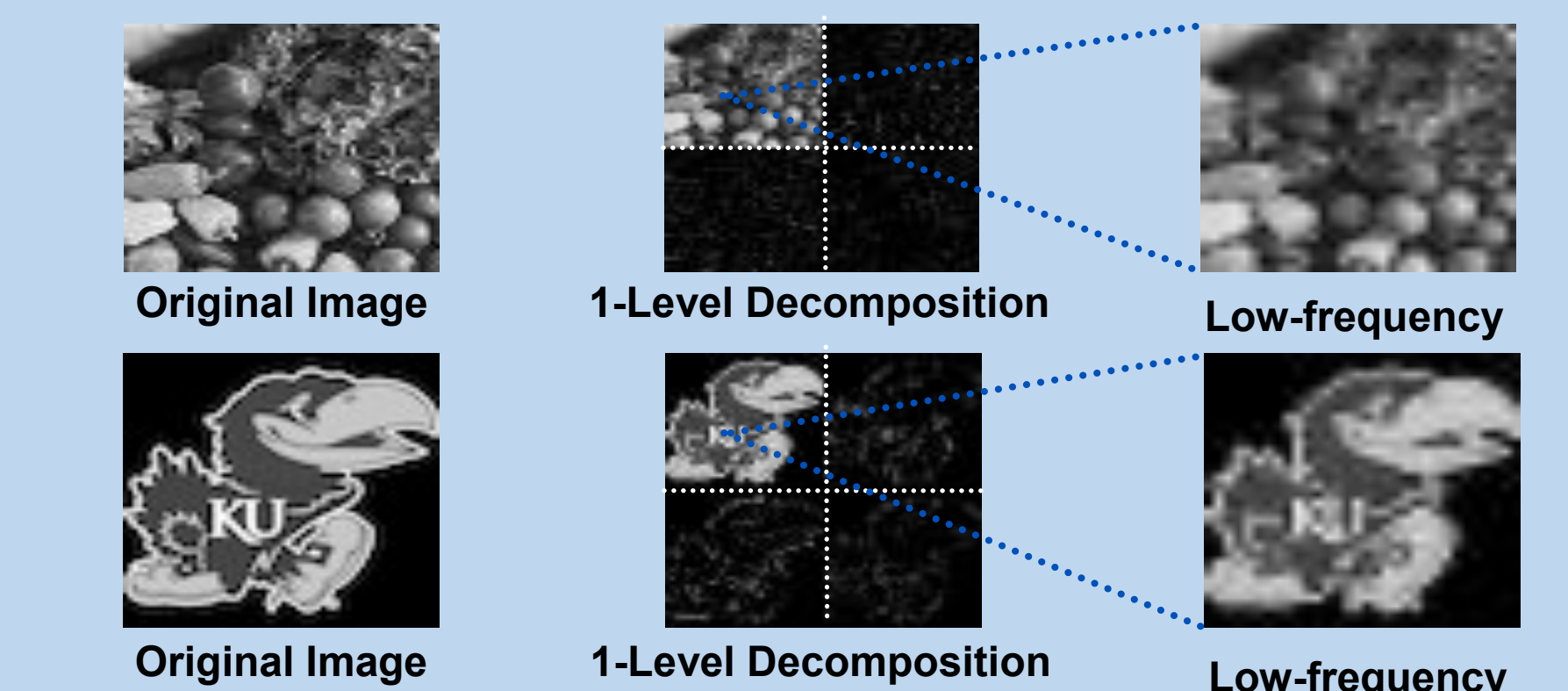
Algorithm Type

- Matrix buffering shows clear advantage at $n = 10$
 - Matrix buffering **benefits from input/output overlap**
 - Output is less optimized, hurting vector buffering more
- Advantage diminishes by $n = 12$
 - Vector buffering may scale better with $n > 12$
- Methods are too similar overall to conclude that one is better than the other

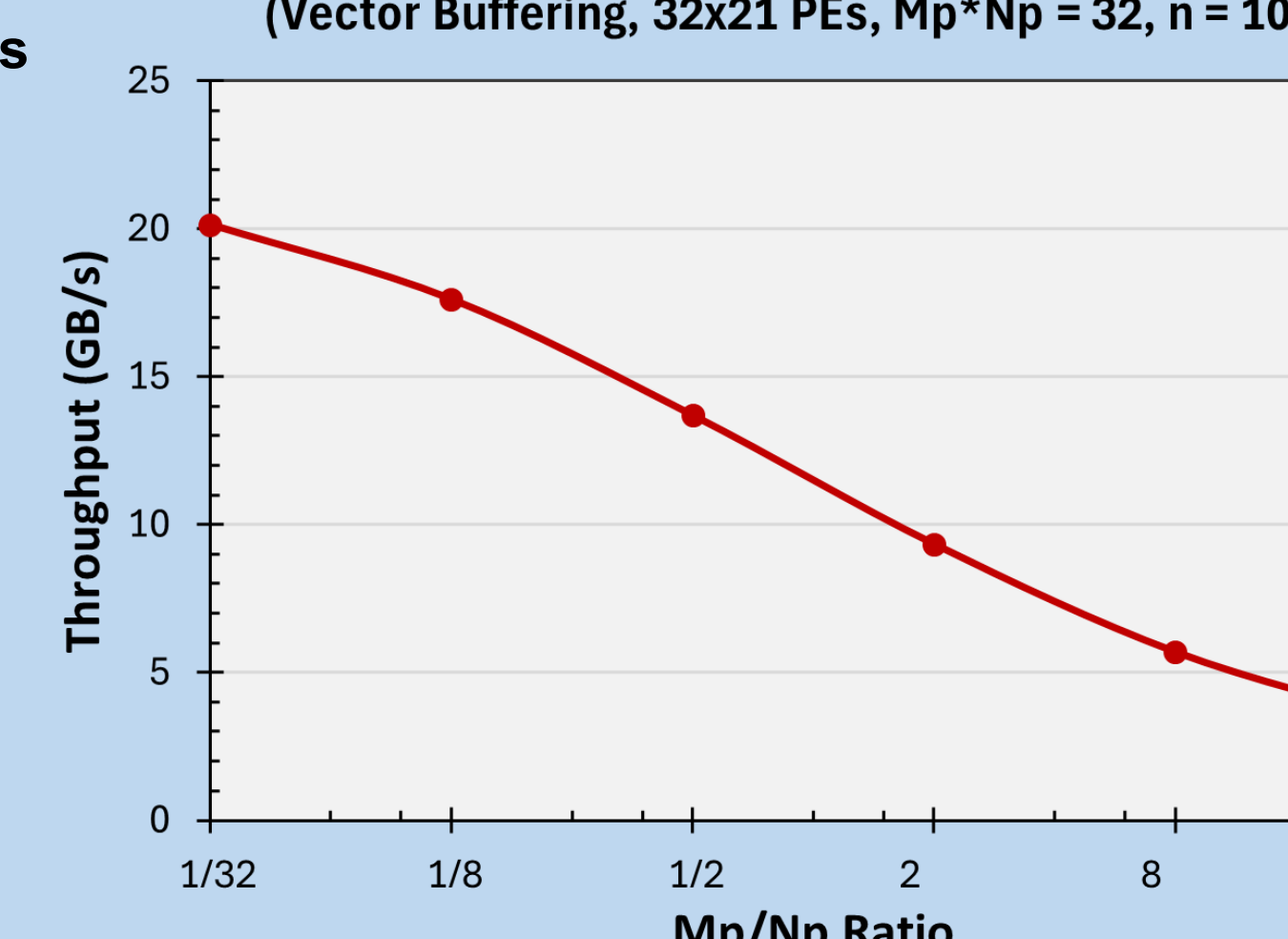
QHT Image Fidelity

- Compared against Qiskit Aer and classically-performed GEMV for correctness
- Tested at 100%, **verifying no loss** from the simulated methodology

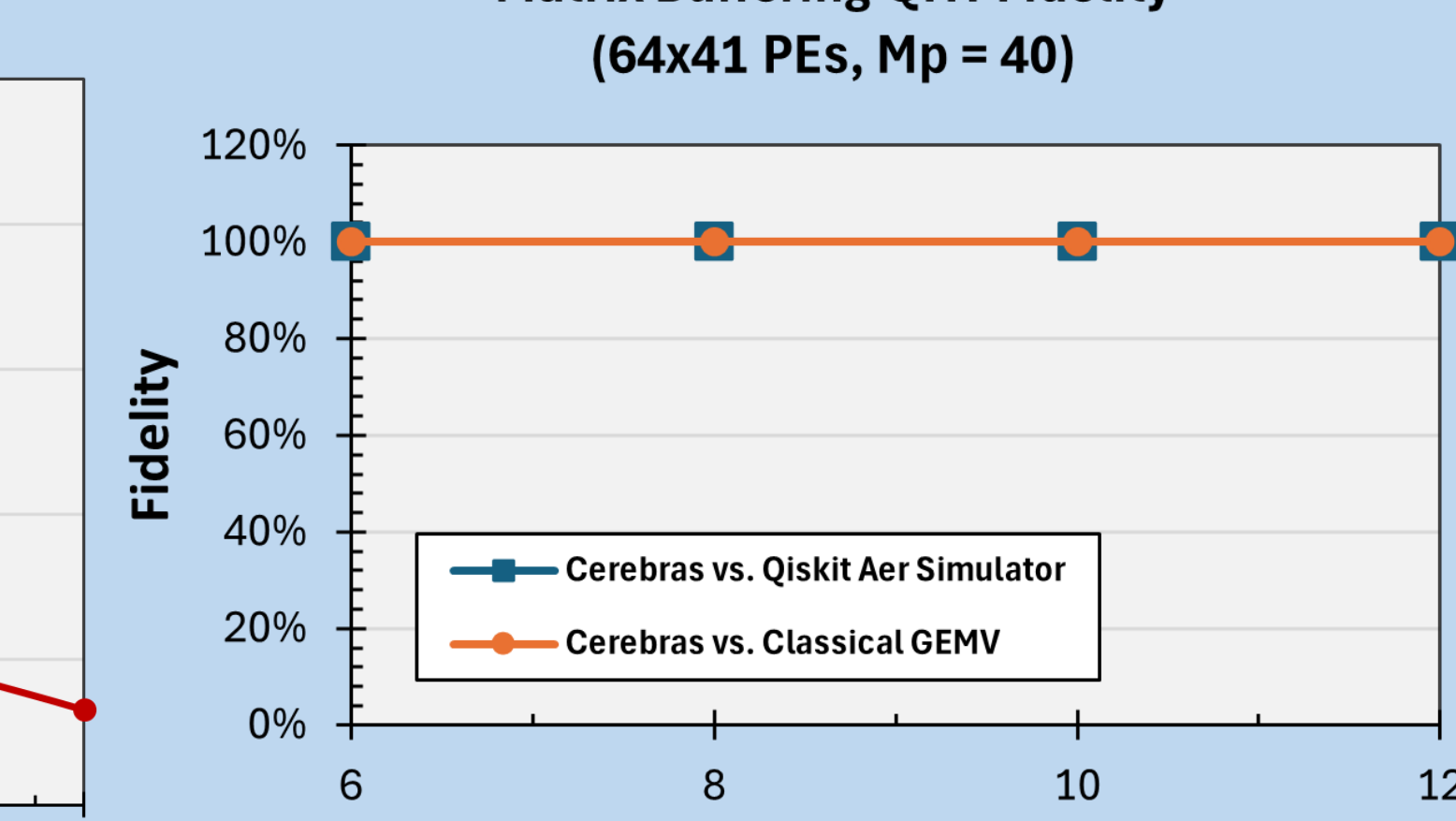
Simulated QHT-based Dimension Reduction on (64x64) Images



Mp/Np Ratio Effect on Throughput (Vector Buffering, 32x21 PEs, Mp*Np = 32, n = 10)



Matrix Buffering QHT Fidelity (64x41 PEs, Mp = 40)



Conclusion and Future Work

In This Work

- We proposed a method for **quantum simulation** using the **General Matrix-Vector product (GEMV)** operation on Cerebras' Wafer-Scale Engines (**WSEs**)
- We demonstrated **scalability** and **parallel advantage** using WSE simulations
 - Results indicate **increasing parallel performance benefits** with larger PE grids and qubit counts
 - Method scales well to the size of a physical WSE
- We investigated **optimal configurations** for multiple variables of our method
 - Block aspect ratio & buffering scheme

Future Work

- Investigate optimizations to proposed techniques and implementations
- Specialize to specific quantum circuit types, accelerating the method to compete with other hardware platforms
 - Sparse matrix optimization
 - Reduced matrix value range (integers, 0s & 1s, etc.)
 - Tensor contraction & circuit pipelining
- Include quantum error correction (QEC) techniques
- Port to Cerebras WSE hardware

Acknowledgements: This research used resources of the **Oak Ridge Leadership Computing Facility**, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

References

- [1] Eftymiou, S., Ramos-Calderer, S., Bravo-Prieto, C., P.erez-Salinas, A., Garc'ia-Mart'ın, D., Garcia-Saez, A., Latorre, J.I. and Carranza, S., 2021. Qibo: a framework for quantum simulation with hardware acceleration. Quantum Science and Technology, 7 (1), p.015018.
- [2] Hori, H. and Wood, C., 2023. Efficient techniques to gpu accelerations of multi-shot quantum computing simulations. arXiv preprint arXiv:2308.03399.
- [3] Jones, T., Brown, A., Bush, L. and Benjamin, S.C., 2019. QuEST and high performance simulation of quantum computers. Scientific reports, 9(1), p.10736.
- [4] Malmrud, N., El-Araby, E. and Caliga, D., 2019. Scaling reconfigurable emulation of quantum algorithms at high precision and high-throughput. Quantum Engineering, 1(2), p.e19.
- [5] S. Lie, "Cerebras Architecture Deep Dive: First Look Inside the Hardware/Software Co-Design for Deep Learning," in IEEE Micro, vol. 43, no. 3, pp. 18-30, May-June 2023, doi: 10.1109/MM.2023.3256384.
- [6] Brown, N., Echols, B., Zarins, J. and Grosser, T., 2022, August. Exploring the Suitability of the Cerebras Wafer Scale Engine for Stencil-Based Computation Codes. In European Conference on Parallel Processing (pp. 51-65). Cham: Springer Nature Switzerland.
- [7] Liut, H., Hong, Y., Wilson, L., Jacquelin, M., Ravasi, M. and Keyes, D.E., 2023, November. Scaling the "memory wall" for multi-dimensional seismic processing with algebraic compression on cerebras cs-2 systems. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp.1-12).
- [8] Rooki, K., Essendelft, D.V., Sharapov, I., Schreiber, R., Morrison, M., Kibardin, V., Portnoy, A., Dietiker, J.F., Syamali, M., James, M., 2020, October. Fast Stencil-Code Computation on a Wafer-Scale Processor. In Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis. arXiv preprint arXiv:2010.03660.
- [9] Dey, N., Gosal, G., Khachane, H., Marshall, W., Pathria, R., Tom, M. and Hestress, J., 2023. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. arXiv preprint arXiv:2304.03208.
- [10] Orenes-Vera, M., Sharapov, I., Schreiber, R., Jacquelin, M., Van-demersch, P. and Chetlur, S., 2023, June. Wafer-scale fast fourier transforms. In Proceedings of the 37th International Conference on Supercomputing (pp. 180-191).
- [11] IBM Quantum. Qiskit: An open-source framework for quantum computing, 2021.