PIMUTATION

Exploring the Potential of Real PIM Architecture for Quantum Circuit Simulation

Dongin Lee* (National University of Singapore), **Enhyeok Jang***, Seungwoo Choi, Junwoong An, Cheolhwan Kim, and Won Woo Ro (Yonsei University)

*These two authors contributed equally to this work

E-mail: dongin.lee@u.nus.edu, {enhyeok.jang, wro}@yonsei.ac.kr

Session 3D-3 (T3-2) Frameworks and Modeling for Computing-In-Memory, Room Mars/Mercury

Tuesday, January 21st, 2025

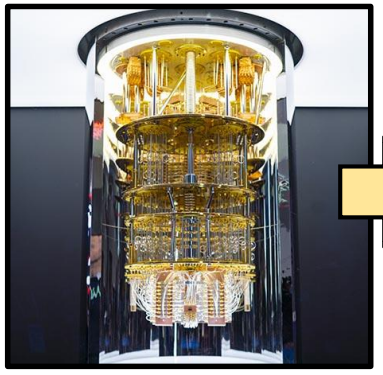
Contents

Contents

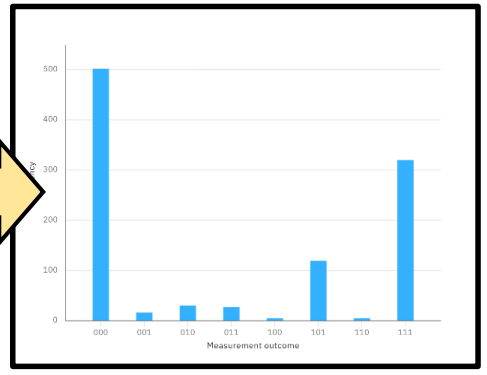
Why Quantum Circuit Simulation is important ?

- Current quantum computers have low accuracy, long job wait times, and high execution fee.
  - ✓ QCS is advantageous to study computational processes of quantum processors alernatively.
- QCSs are also essential for the verification of intermediate states in quantum algorithms,
  - ✓ Real QCs lose their superposed/entangled states after qubit measurements.

Real Quantum System



Noisy Results

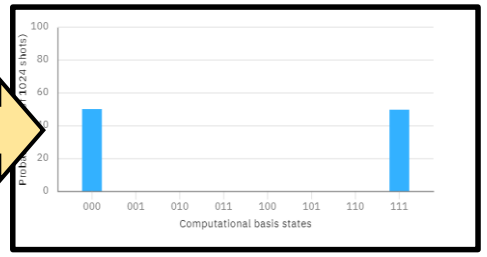


Real quantum computers suffer from
- High error rates
- Low fidelity
- Low accessibility

☹

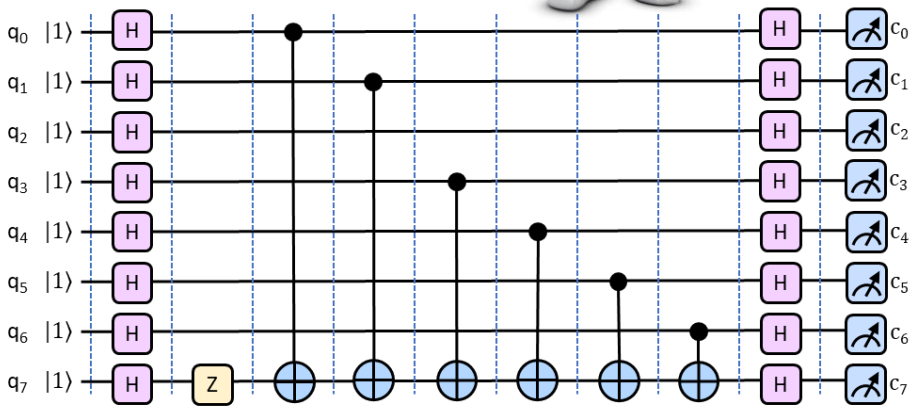
Classical Simulation

Theoretical Probabilities



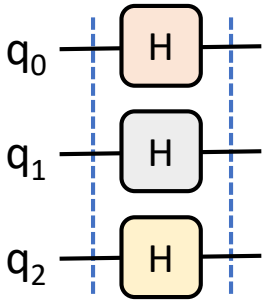
Quantum circuit simulation can provide
- Accurate probability distribution
- Intermediate states verification
- Error / noise modeling

☺



Basic of Quantum Circuit Simulation

- State vector-based QCS is the most common method.
- State vectors for n-qubit system requires memory capacity of $\mathbf{2^n}$ elements of complex amplitudes.
  - ✓ EX) 1Q gate for 10-qubit system correspond to 512 times of 2x2 Matrix - 2x1 Vector Multiplications.

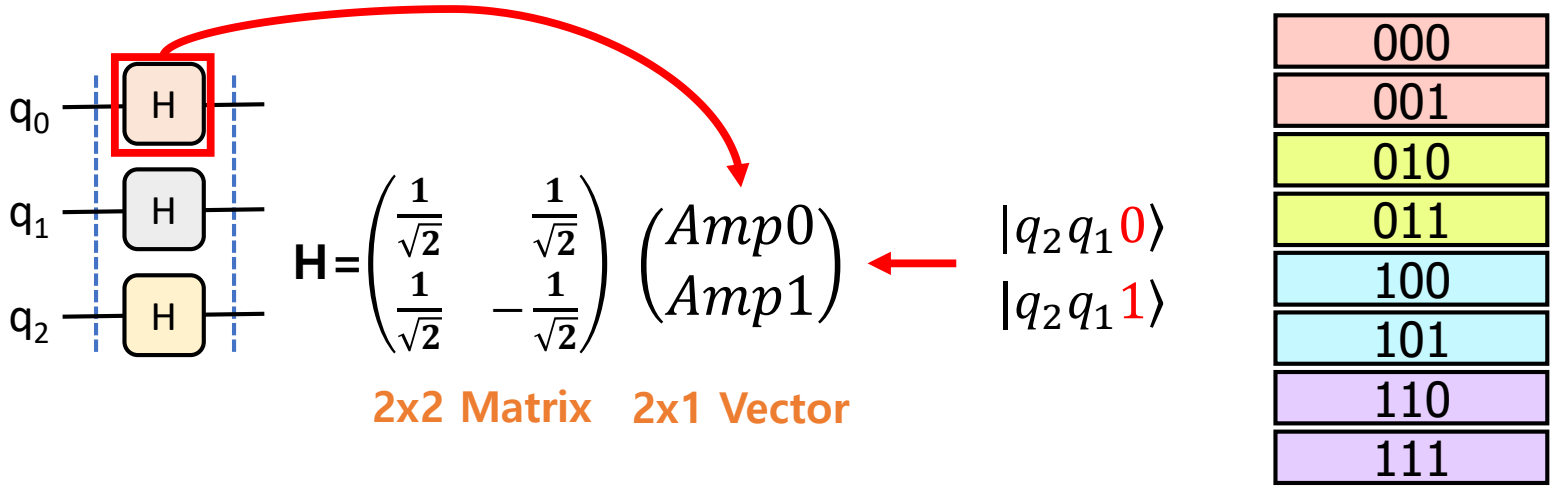


$$\mathbf{H}=\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix}\begin{pmatrix}Amp0\\ Amp1\end{pmatrix}\qquad \begin{matrix}|q_2q_1q_0\rangle\\ |q_2q_1q_0\rangle\end{matrix}$$

2x2 Matrix 2x1 Vector

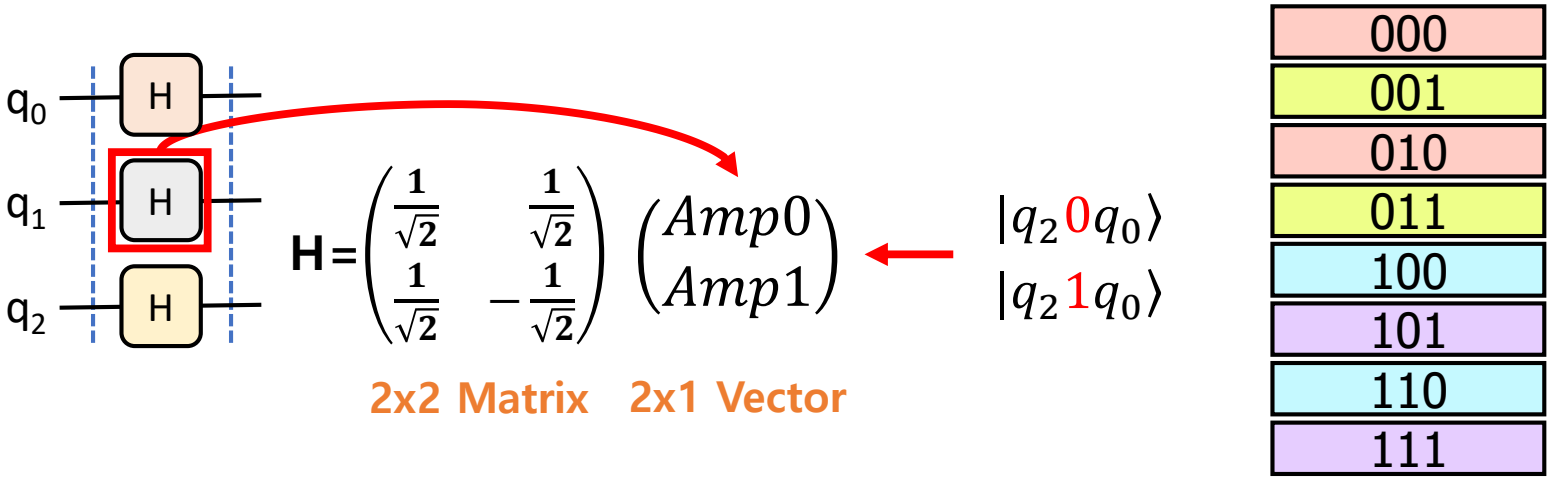
Basic of Quantum Circuit Simulation

- State vector-based QCS is the most common method.
- State vectors for n-qubit system requires memory capacity of $\mathbf{2^n}$ elements of complex amplitudes.
  - ✓ EX) 1Q gate for 10-qubit system correspond to 512 times of 2x2 Matrix - 2x1 Vector Multiplications.



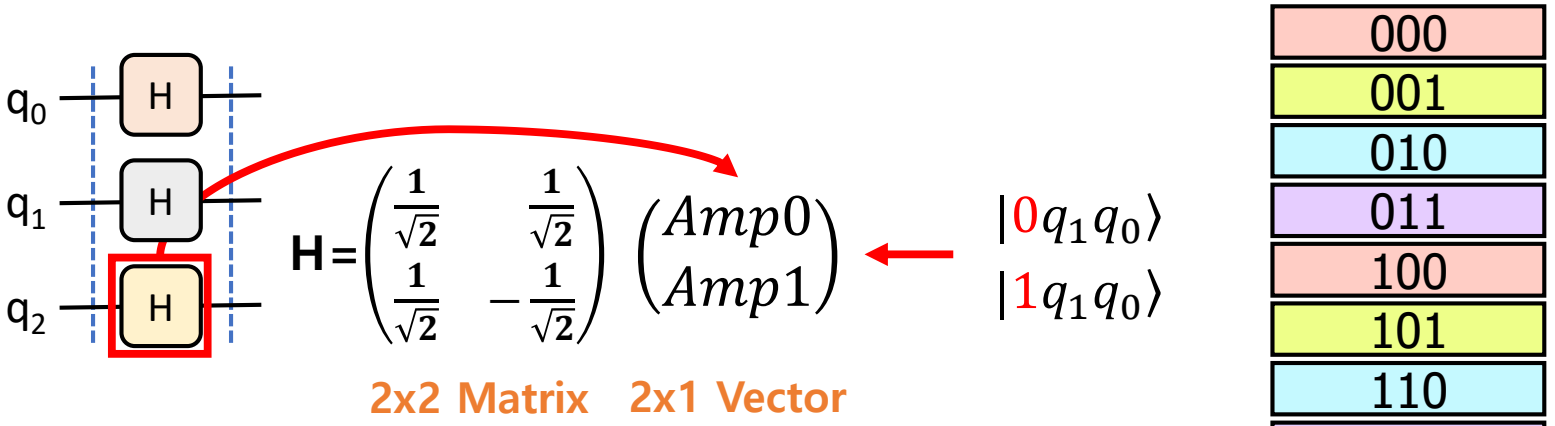
Basic of Quantum Circuit Simulation

- State vector-based QCS is the most common method.
- State vectors for n-qubit system requires memory capacity of $\mathbf{2^n}$ elements of complex amplitudes.
  - ✓ EX) 1Q gate for 10-qubit system correspond to 512 times of 2x2 Matrix - 2x1 Vector Multiplications.



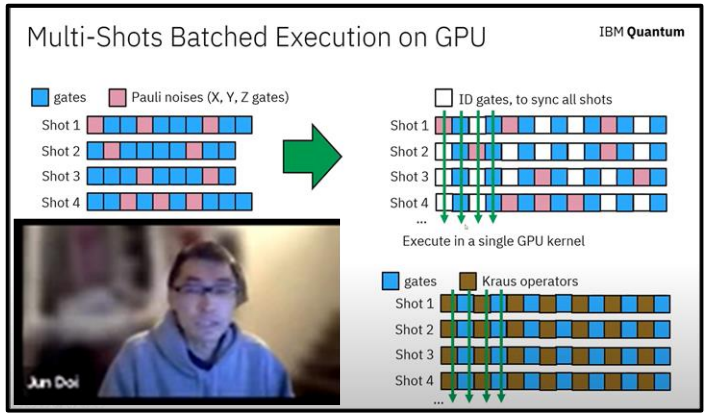
Basic of Quantum Circuit Simulation

- State vector-based QCS is the most common method.
- State vectors for n-qubit system requires memory capacity of $\mathbf{2^n}$ elements of complex amplitudes.
  - ✓ EX) 1Q gate for 10-qubit system correspond to 512 times of 2x2 Matrix - 2x1 Vector Multiplications.



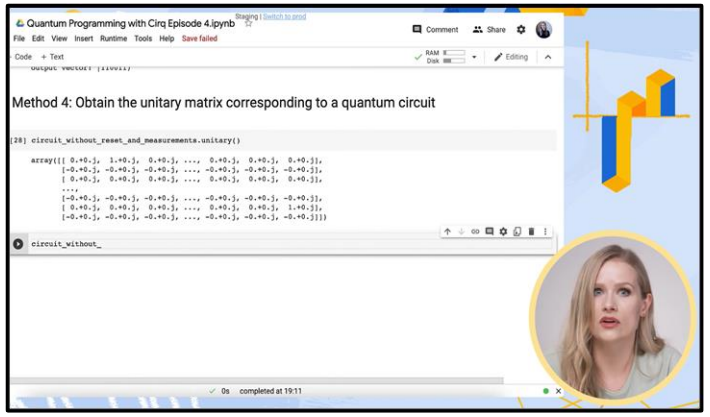
Trends in Quantum Circuit Simulation in Industry Fields

IBM – Qiskit Aer on GPU



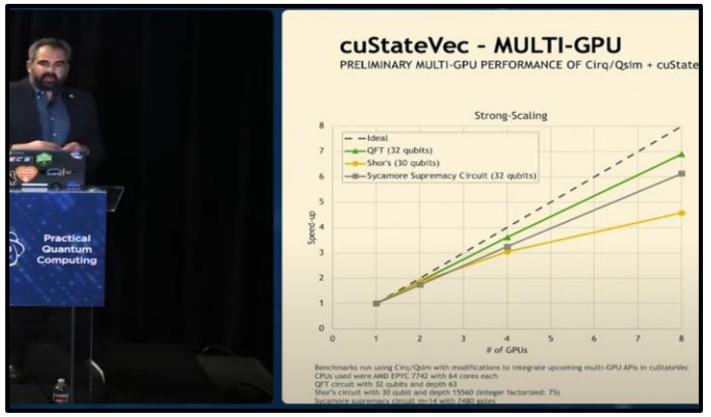
Demonstration: 6 x Nvidia 4-GPU V100
Achievement: 32-Qubit Simulation

Google – Qsim on GPU



Demonstration: 8 x Nvidia 8-GPU A100
Achievement: 36-Qubit Simulation

NVIDIA – cuQuantum SDK



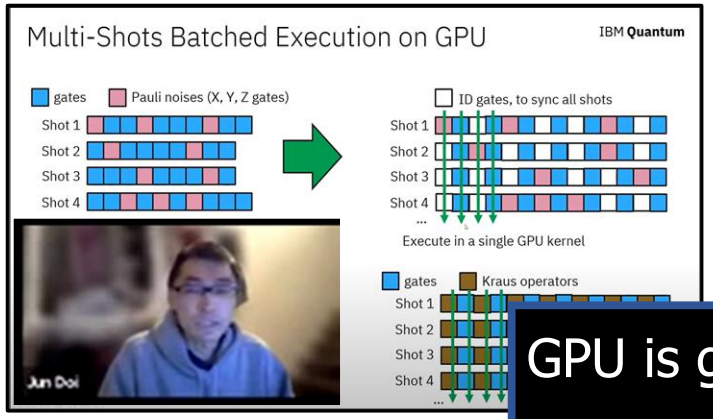
Demonstration: 32 x Nvidia 8-GPU A100
Achievement: 40-Qubit Simulation

- IBM, Google, and NVIDIA have competitively reported large-scale quantum circuit simulations leveraging multiple GPU systems (V100 or A100).

https://www.youtube.com/watch?v=T-a__rCfKTE
https://quantumai.google/qsim/choose_hw
https://www.youtube.com/watch?v=Qdb3xrzT1gc

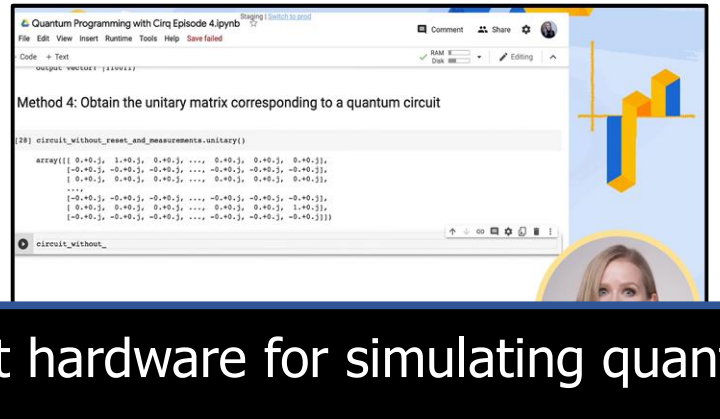
Trends in Quantum Circuit Simulation in Industry Fields

IBM – Qiskit Aer on GPU

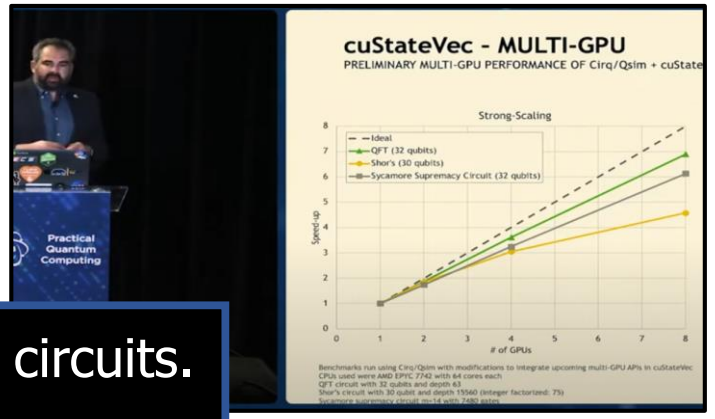


Demonstration: 6 x Nvidia 4-G
Achievement: 32-Qubit Simula

Google – Qsim on GPU



NVIDIA – cuQuantum SDK



on: 32 x Nvidia 8-GPU A100
: 40-Qubit Simulation

GPU is great hardware for simulating quantum circuits.

However, the workload property of quantum circuit simulations are not so compute-intensive.

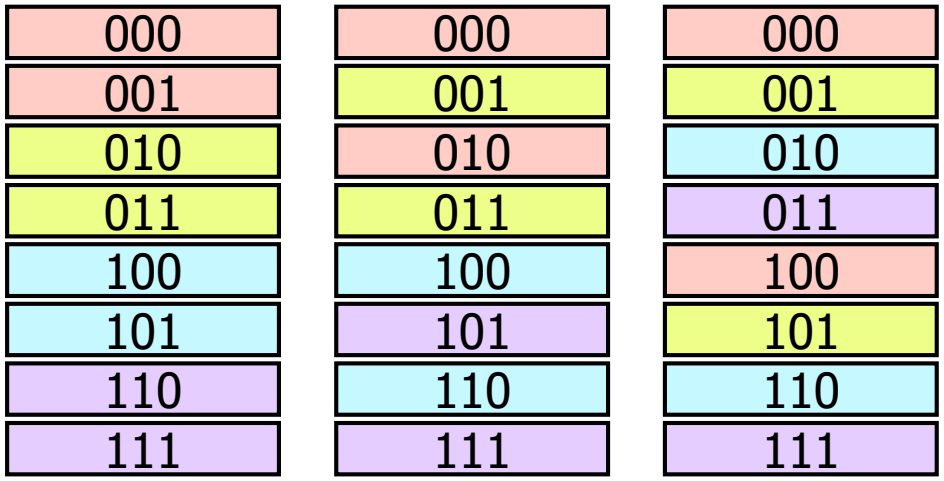
=> Simulating quantum programs can also be efficient in low-power systems such as process-in-memory.

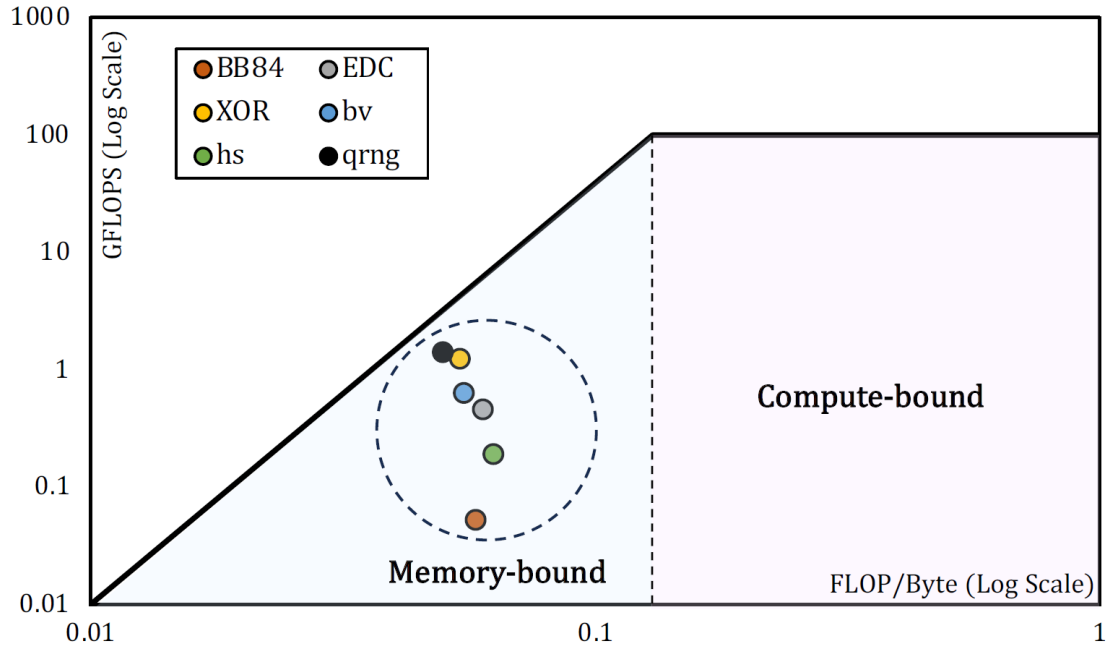
- IBM, Google, and N t simulations leveraging multiple

https://www.youtube.com/watch?v=T-a__rCfKTE
https://quantumai.google/qsim/choose_hw
https://www.youtube.com/watch?v=Qdb3xrzT1gc

Workload Characteristics of QCSs

- QCS consists of iterations of small-sized matrix-vector multiplications.
- Each gate requires *distinct access patterns* to all 2^n amplitudes at least once.
  - ✓ Low locality data access, which Incurs a lot of cache misses.
  - ✓ Frequent data movements between CPU and main memory.
- Various quantum applications exhibit *memory (bandwidth)-bound* properties.





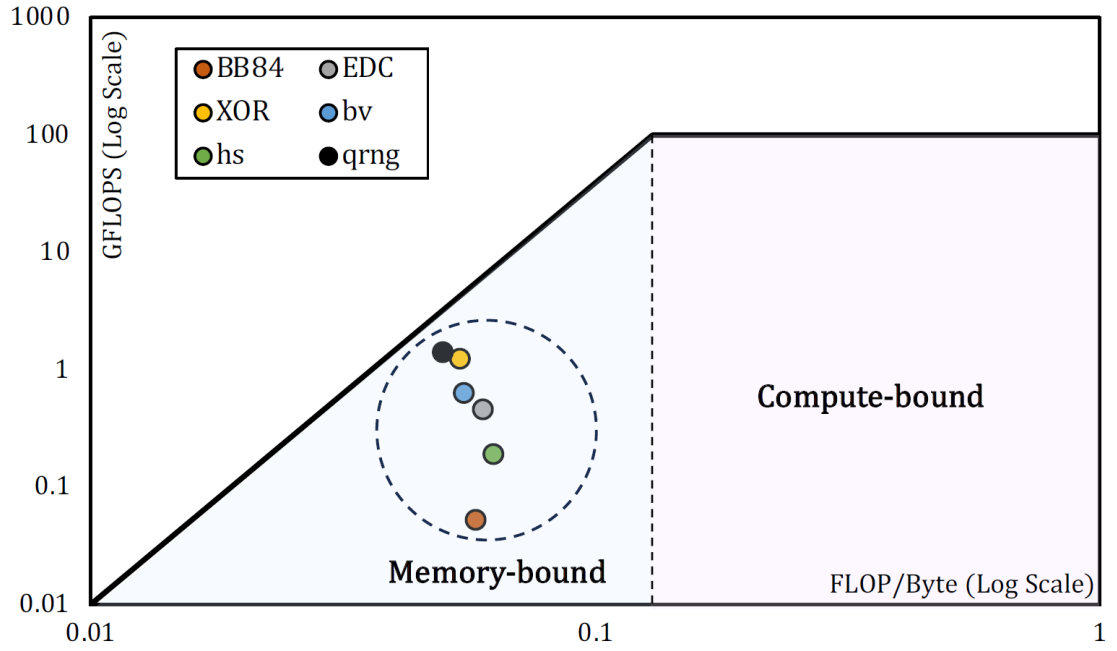
< Roofline Analysis on Intel Xeon Silver 4215 Processor >

Workload Characteristics of QCSs

- QCS consists of iterations of small-sized matrix-vector multiplications.
- Each gate requires *distinct access patterns* to all 2^n amplitudes at least once.
  - ✓ Low locality data access, which Incurs a lot of cache misses.
  - ✓ Frequent data movements between CPU and main memory.
- Various quantum applications exhibit *memory (bandwidth)-bound* properties.

Thus, processing-in-memory (PIM)
is expected to be an architecture
suitable for quantum circuit simulations.

011	011	011
100	100	100
101	101	101
110	110	110
111	111	111



< Roofline Analysis on Intel Xeon Silver 4215 Processor >

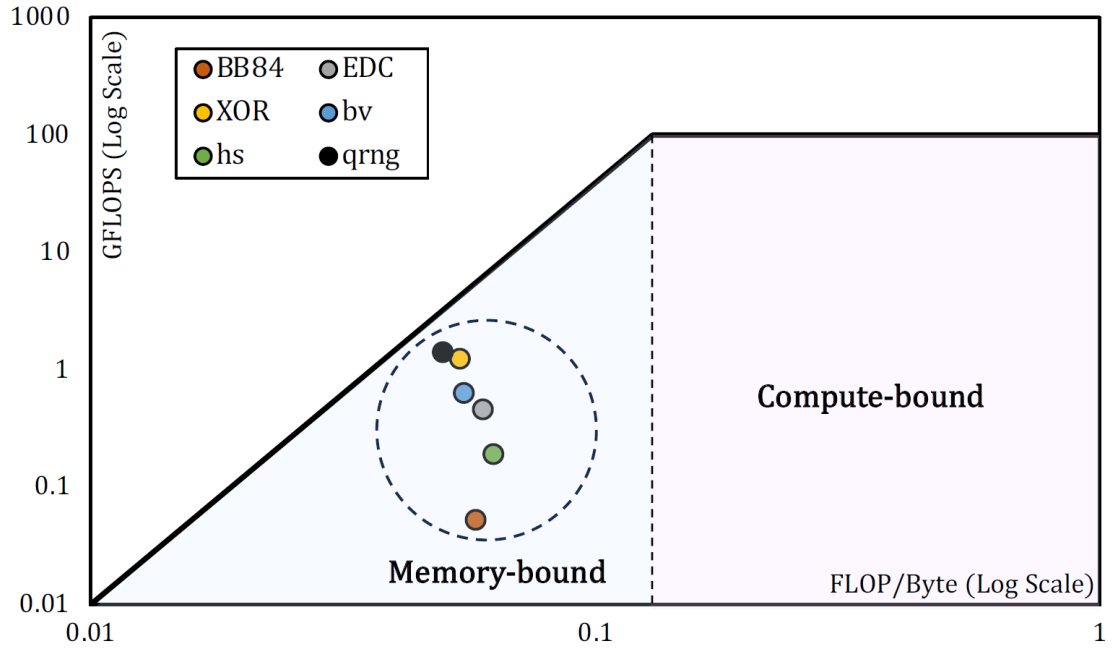
Workload Characteristics of QCSs

- QCS consists of iterations of small-sized matrix-vector multiplications.
- Each gate requires *distinct access patterns* to all 2^n amplitudes at least once.
  - ✓ Low locality data access, which Incurs a lot of cache misses.
  - ✓ Frequent data movements between CPU and main memory.
- Various quantum applications exhibit *memory (bandwidth)-bound* properties.

Thus, processing-in-memory (PIM) is expected to be an architecture suitable for quantum circuit simulations.

011 011 011

To demonstrate our hypothesis that PIM can be efficient for QCS, we adopt UPMEM, the first publicly-available real-world PIM architecture.

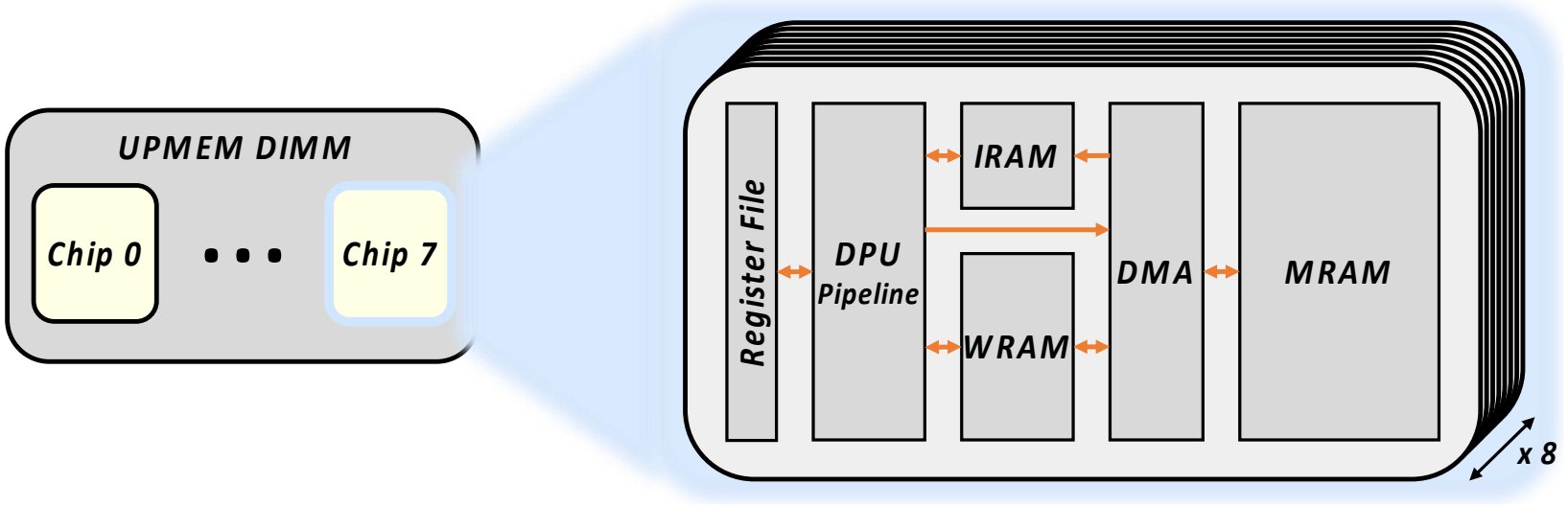


< Roofline Analysis on Intel Xeon Silver 4215 Processor >

Design Challenges to Run QCS with UPMEM

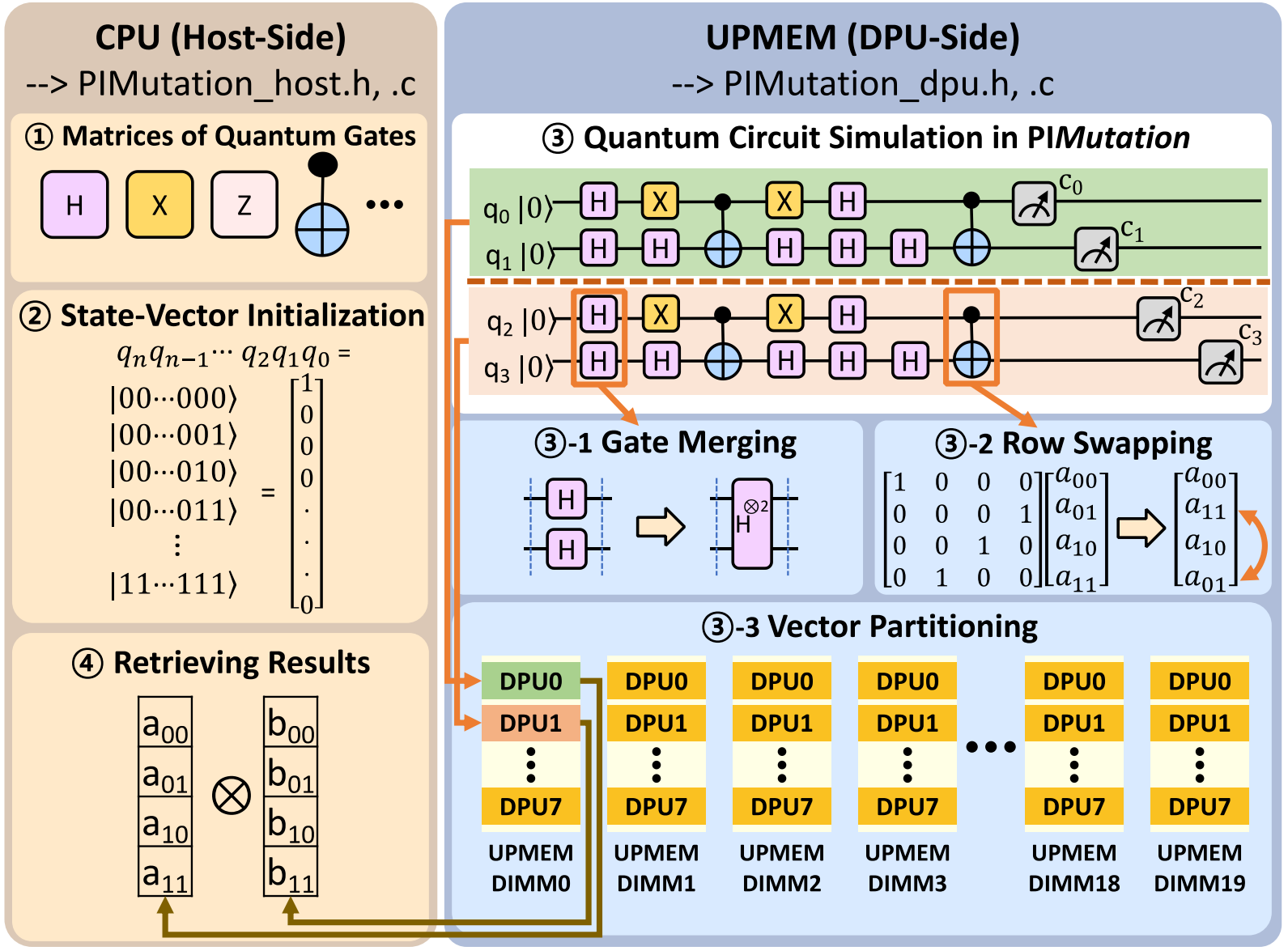
- QCSs require FP Ops for calculating complex-valued amplitudes.
  - ✓ However, DPU (UPMEM's processing unit) only supports integer Ops.
  - ✓ FP Ops are implemented using software libraries, but, which degrades the performance significantly. ☹

- Depending on the gate's positions, the memory access patterns can vary greatly.
  - ✓ However, there is no direct communication channel between DPUs. ☹



Contents

PIMUTATION: PIM for qUanTum circuit SimulATION



First, the host CPU sends quantum ***gate*** information and initialized ***state vectors*** to the DPU.

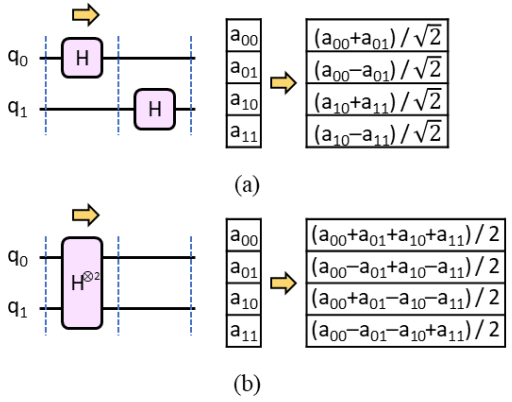
To minimize ***FP Ops and multiplication*** operations, PIMutation utilizes Gate Merging and Row Swapping.

To reduce ***communication between DPUs***, it is divided into sub-circuits and transmitted to each DPU if the original QC is separable.

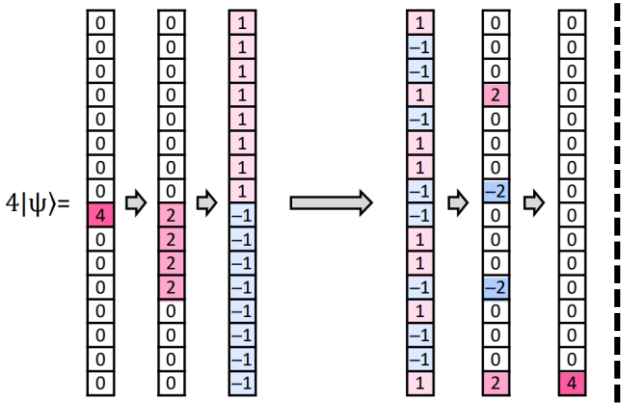
Finally, sub-circuit results are reconfigured in the host CPU and then restored to the full state vector, suppressing communication between DPUs.

PIMUTATION Optimization Techniques

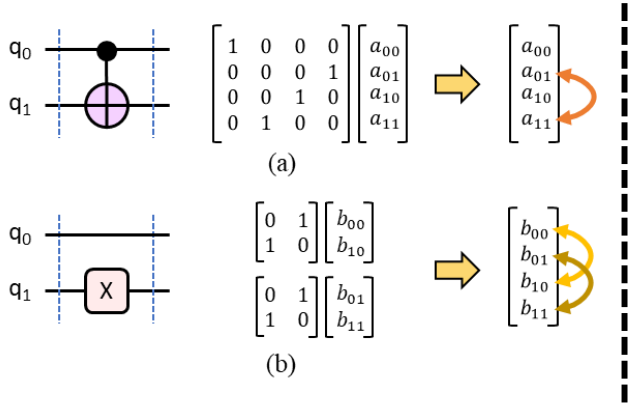
- Gate merging: Exploiting UPMEM native Integer Ops
  - ✓ This replaces irrational number divisions in gates with bit-wise shifts that UPMEM can process.
- State Resizing: Maintaining data type of amplitudes as Integer
  - ✓ By normalizing the min. amplitude to 1, the amplitudes can be integers during the entire process.
- Row swapping: Avoiding matrix multiplication for CX, SWAP gates
  - ✓ For matrices with only 0 or 1 elements, amplitudes are processed by move Ops, instead of MUL.
- Vector Partitioning: Partitioning separable quantum programs, if possible
  - ✓ Sub-circuits are isolated each other and processed at each DPU w/o communication between DPUs.



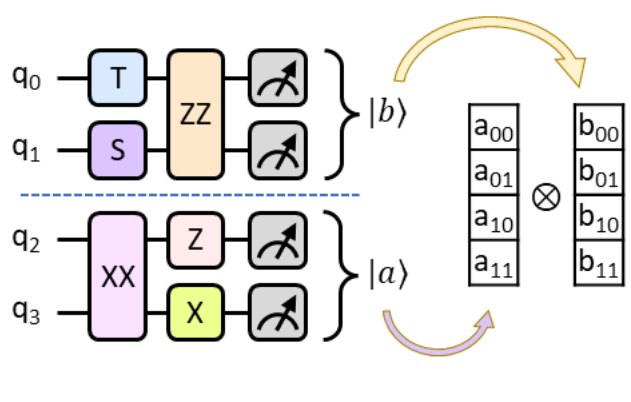
Gate Merging



State Resizing



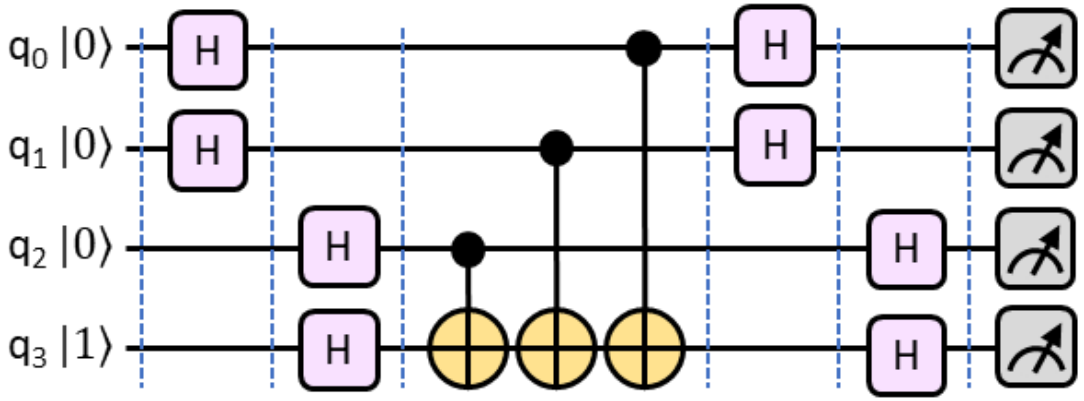
Row Swapping



Vector Partitioning

Gate Merging

- Quantum gates often require irrational division.
  - ✓ For example, RX($\frac{\pi}{2}$) = $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}$, RY($\frac{\pi}{2}$) = $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, and H= $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- GM provides an avoidance of FP Ops.
- Merged gates can be processed just using integer Ops or bit-wise Ops.

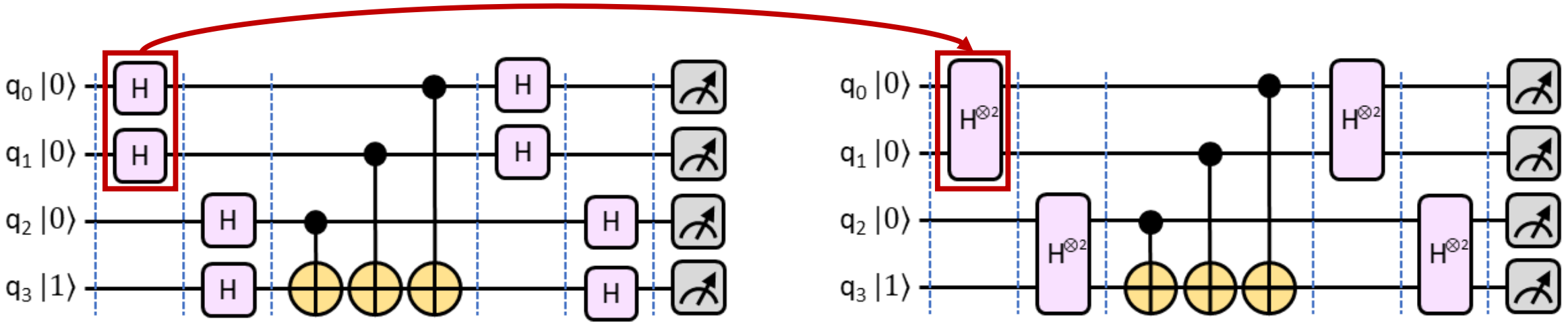


< Bernstein-Vazirani Circuit for the secret string of 111 >

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Gate Merging

- Quantum gates often require irrational division.
  - ✓ For example, $RX(\frac{\pi}{2}) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}$, $RY(\frac{\pi}{2}) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, and $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- GM provides an avoidance of FP Ops.
- Merged gates can be processed just using integer Ops or bit-wise Ops.



< Bernstein-Vazirani Circuit for the secret string of 111 >

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

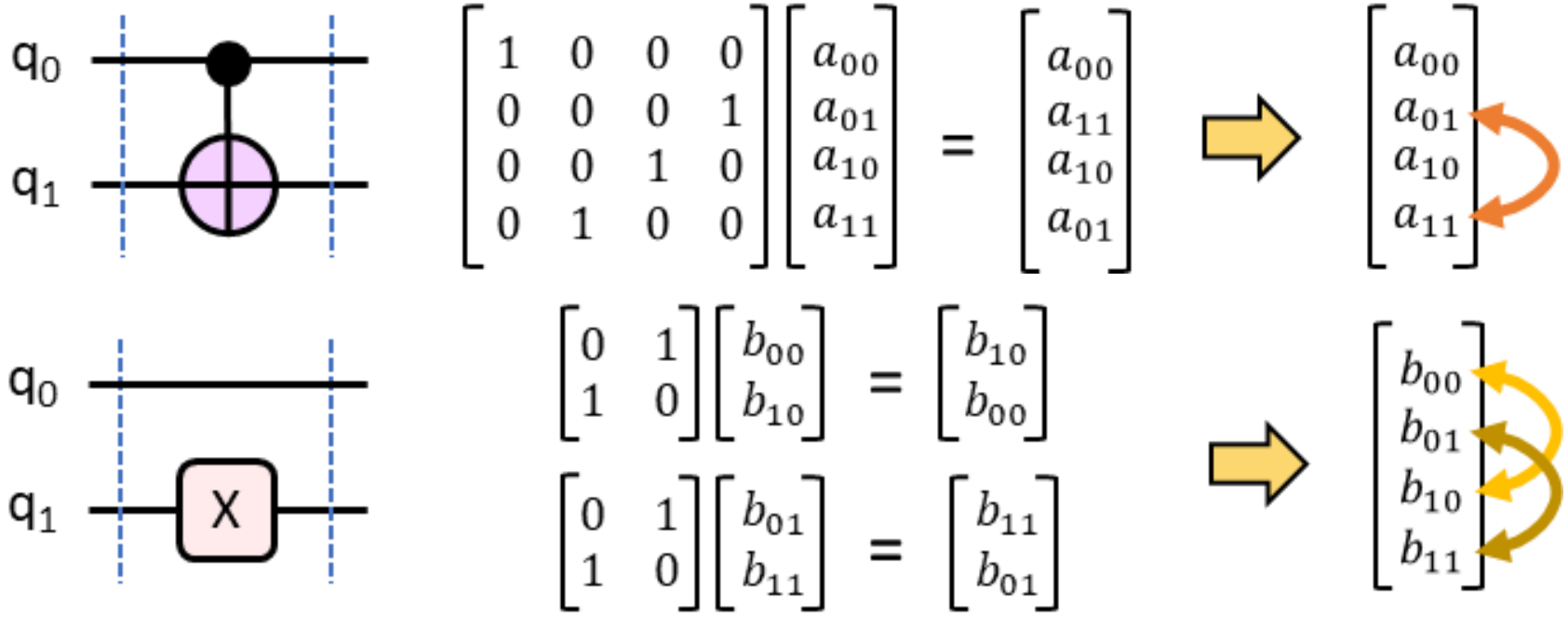
$$H \otimes H = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Adjusting the unitary coefficient by GM
Irrational Division => Bit-Wise Shift Ops

State Resizing is implemented.

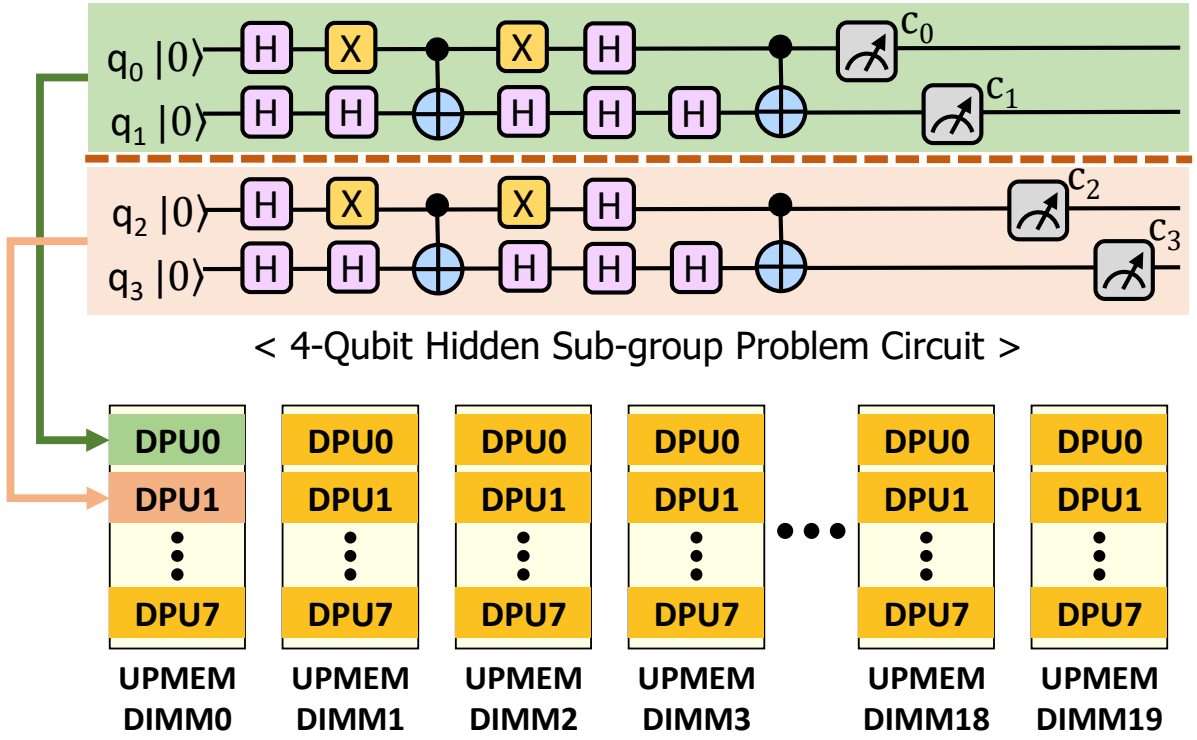
Row Swapping

- Some gates (CX or SWAP) are just switching amplitudes.
- These are implemented by directly swapping the corresponding row pairs.
  - ✓ FP Mult Ops (hard to UPMEM ☹) => *Move arrays directly*



Vector Partitioning

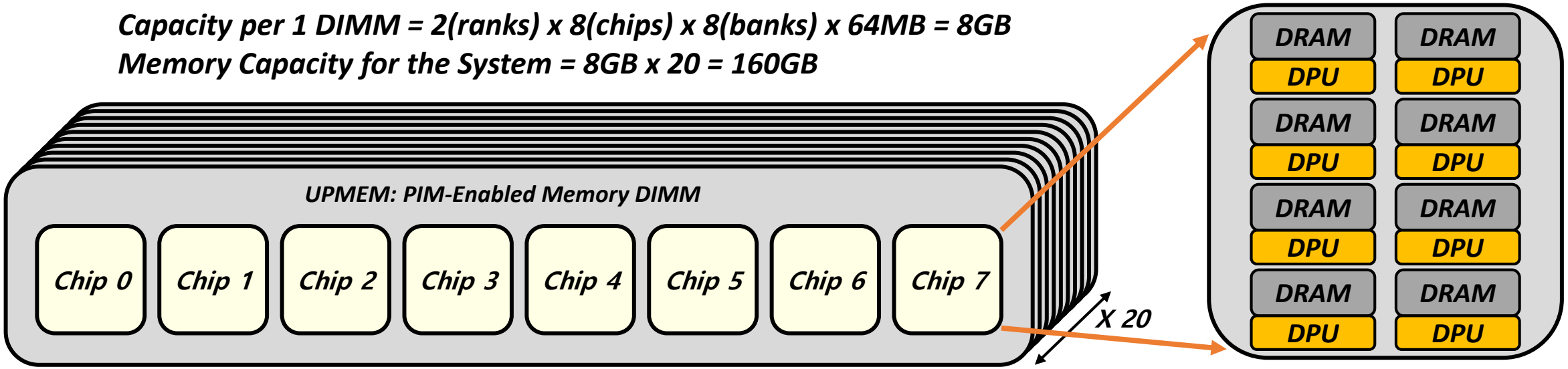
- Separable quantum circuit can be independently processed into sub-circuits.
- They are handled at each DPU and reconstructed to original vector on CPU.



Contents

Backend: Real UPMEM PIM Server (provided by UPMEM Co.)

- A single server provides 20 UPMEM DIMMs
- Each UPMEM DIMM includes 8 memory chips per rank
- Each memory chip is coupled with 8 processing element
- Processing element is called ***D***RAM ***P***rocessing ***U***nit (***DPU***)



Experimental Set-Up

Table 1: The Platform for Experiments

CPU Model	Intel(R) Xeon(R) Silver 4215
Sockets	2
Memory	4 x 64 GB DDR4-2666 RDIM Dual Rank DRAM (256 GB)
PIM Memory	20 × DDR4-2400 PIM Modules (160 GB)
UPMEM SDK version	2023.2.0
QuEST version	v3.7.0 [15]
Compiler version	gcc 8.3.0
CMake version	3.13.4
Energy Measurement	Intel RAPL [7]

Table 2: Evaluated Benchmark Quantum circuits

Algorithm [Reference]	Qubits	# of 1Q Gates	# of 2Q Gates	Abbr.
BB84 Protocol [5]	n	2n	0	BB_n
Bernstein–Vazirani [6]	n	2n	n-1	BV_n
Error Detection Code [9]	n	2n	2n-2	EDC_n
Hidden Subgroup Problems [20]	2n	6n	2n	HS_2n
Quantum Random Number Generator [20]	n	n	0	QRNG_n
Exclusive-OR [25]	n	0	n-1	XOR_n

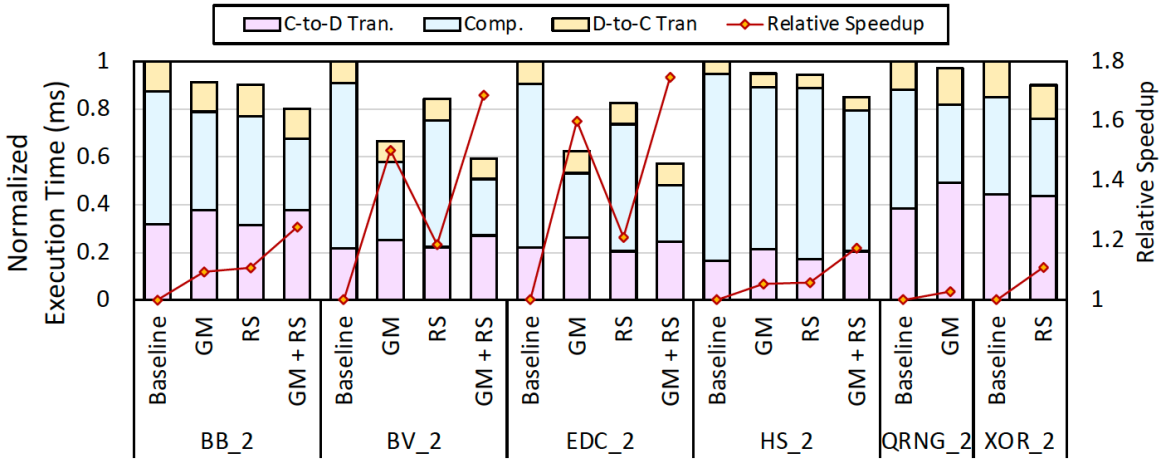
Based on Float64, UPMEM PIM memory can store up to 33-qubit systems' full state-vector, which requires 128 GB.

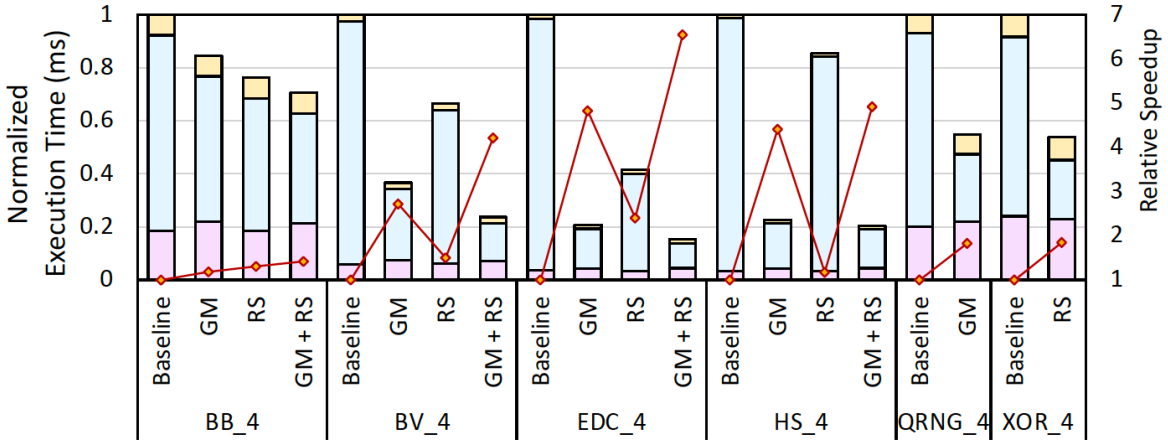
Evaluated Scenarios

- Single-DPU experiments
  - Evaluating the performance for:
    - ✓ Gate Merging (GM) + State Resizing Included
    - ✓ Row Swapping (RS)
- Multi-DPU experiments
  - Evaluating the performance for:
    - ✓ Gate Merging (GM) + State Resizing Included
    - ✓ Row Swapping (RS)
    - ✓ Vector Partitioning (VP)
  - Evaluating Speed-up and Energy consumption by PIMutation over the ***C language-based cutting-edge QuEST simulator*** on CPU
  - Evaluations on large-scale (16- and 32-qubit) benchmarks

Contents

Single-DPU Performance

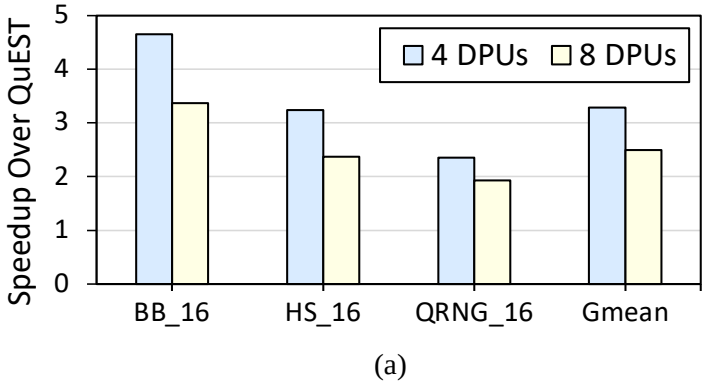


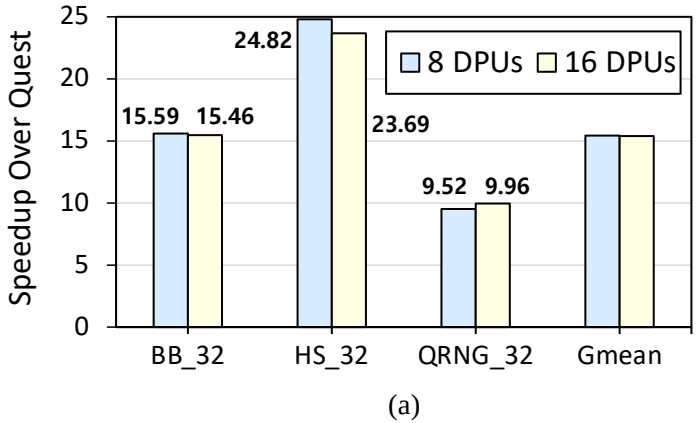


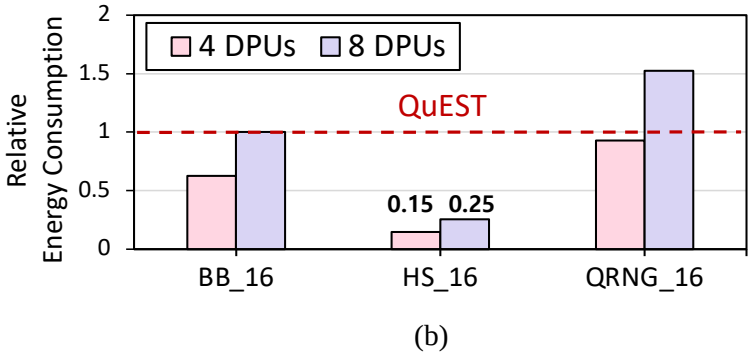
- In 2-qubit benchmarks (XX_2), Gate Merging (GM) and Row Swapping (RS) achieves a speedup of 0.26x and 0.13x over Baseline (No Optimization).

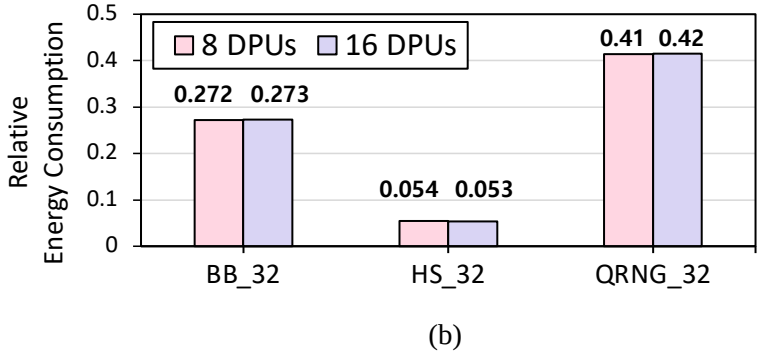
- In 4-qubit benchmarks (XX_4), Gate Merging (GM) and Row Swapping (RS) achieves a speedup of 2.99x and 1.65x over Baseline (No Optimization).

Multi-DPU Performance



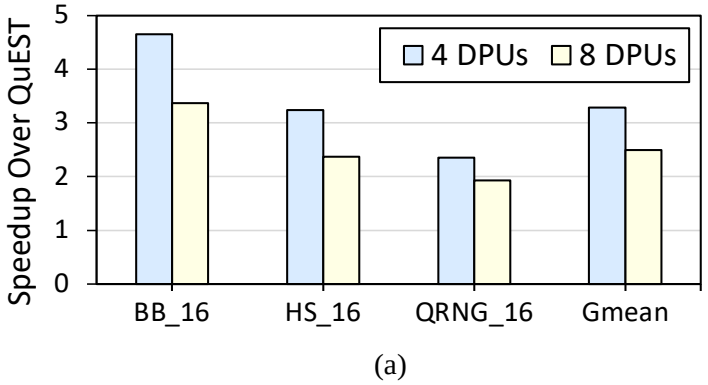


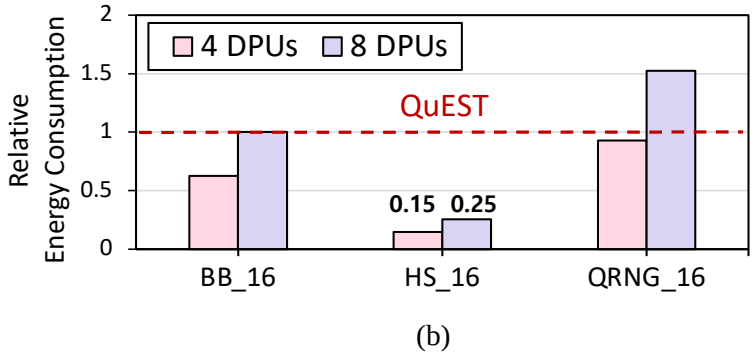


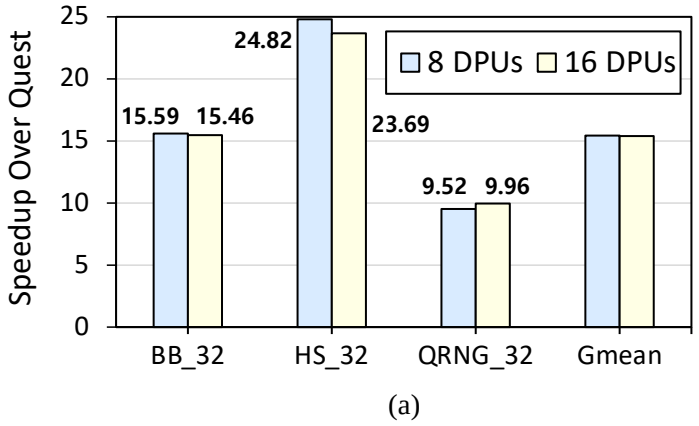


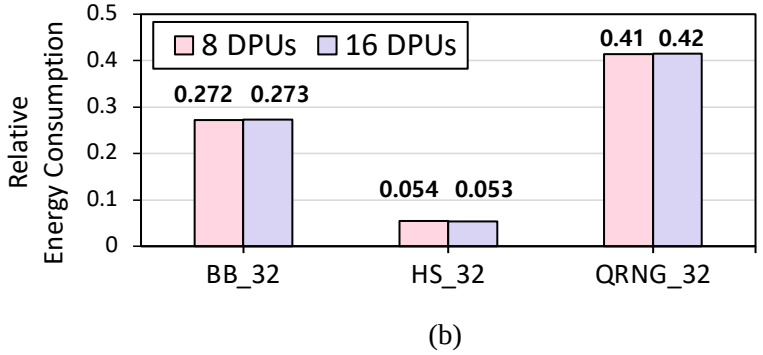
- 16-qubit Workloads
  - ✓3.42x and 2.56x speedup over the QuEST in 4 and 8 DPUs, respectively.
  - ✓25% energy saving over the QuEST in average of 4 and 8 DPUs.
- 32-qubit Workloads
  - ✓16.64x and 16.37x speedup over the QuEST in 8 and 16 DPUs, respectively.
  - ✓75% energy saving over the QuEST in average of 8 and 16 DPUs.

Multi-DPU Performance









- More DPUs do not always provide better performance.
- More DPUs can leverage parallelism, but its increased reconfiguration overhead can degrade overall performance.
  - ✓ For example, 4 DPUs perform better than 8 DPUs.

=> Estimating the optimal number of DPUs for QCS would be an interesting future work.

- 16-qubit Workloads
  - ✓3.42x and 2.56x speedup over the QuEST in 4 and 8 DPUs, respectively.
  - ✓25% energy saving over the QuEST in average of 4 and 8 DPUs.
- 32-qubit Workloads
  - ✓16.64x and 16.37x speedup over the QuEST in 8 and 16 DPUs, respectively.
  - ✓75% energy saving over the QuEST in average of 8 and 16 DPUs.

Contents

Conclusion

- We note that QCSs are bandwidth-bound workloads.
  - ✓ If we adopt PIM, therefore, *fast and low-energy* quantum circuit simulations would be achieved.

- We leverage UPMEM, the first commercialized PIM, to verify our observation.
  - ✓ UPMEM can only support Integer Ops natively and communication between DPUs is not available.
  - ✓ These design challenges should be considered when implementing quantum circuit simulations.

- PIMUTATION can minimize FP Ops and communications between DPUs.
  We propose 4 optimization techniques.
  - **Gate Merging**
  - **State Resizing**
  - **Row Swapping**
  - **Vector Partitioning**

- PIMUTATION achieves faster and lower energy consumption,
  compared to the state-of-the-art high-speed QuEST simulator.

PIMUTATION Contributors

Dongin Lee

Enhyeok Jang

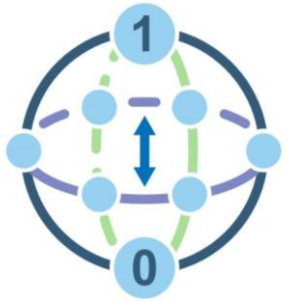
Seungwoo Choi

Junwoong An

Cheolhwan Kim

Dr. Won Woo Ro

NUS National University of Singapore



ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

Thank You Your Time!

Q & A