Quantum Approaches to Sequence Alignment

Vivatsathorn Thitasirivit

2023/1 2110581 Bioinformatics I
Faculty of Engineering, Chulalongkorn University
23 Nov 2023

Outlines

- The Sequence Alignment Problem
- Classical Sequence Alignment and Parallelization

- Basics of Quantum Computing
- Quantum Algorithms for Sequence Alignment

- Experimentations and Results
- Challenges and Prospects
- Summary and Further Research

Disclaimers

- This project focuses on experimentations on a real quantum device compared to a simulator.
- Methods proposed in this project are plausible and practical to run on a real quantum device.
- The author has suggested many possibilities to address and solve the stated problems in further research.

The Sequence Alignment Problem (Pairwise)

- Aims to align 2 sequence with most similarity.
- Global and Local Alignment

- How similar is these two sequences?

AACGG

TGCGT

The Sequence Alignment Problem (Pairwise)

- Aims to align 2 sequence with most similarity.
- Global and Local Alignment

- How similar is these two sequences?

AACGG

TGCGT

```
AA--CGG-
    **
--TGCG-T
```

or

```
AACGG
  **
TGCGT
```

or … ?

Classical Sequence Alignment

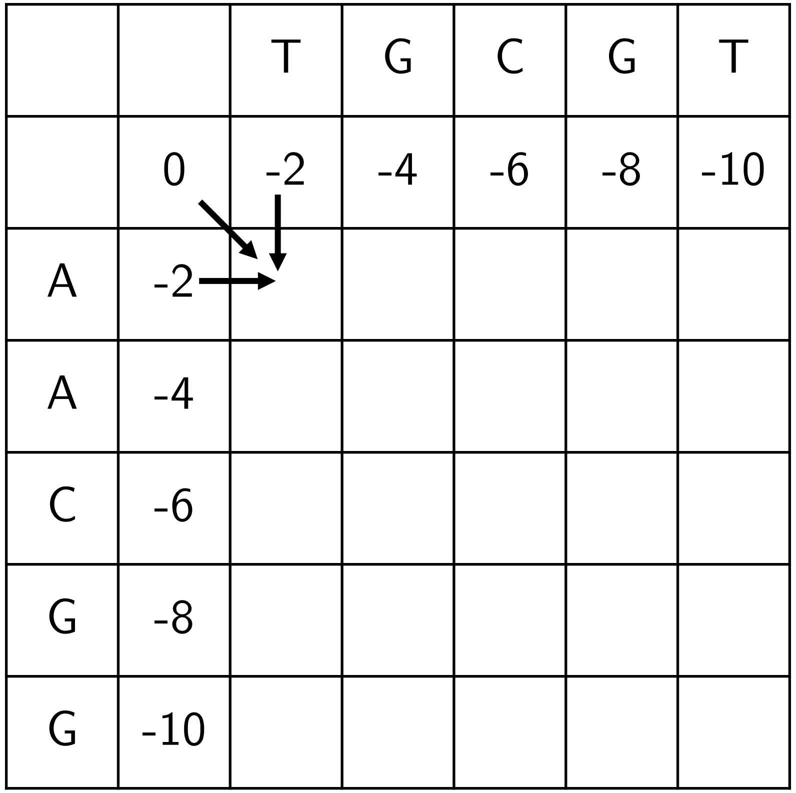
- Dynamic Programming
- $O(n^2)$ in both time and space (score matrix + backtracking)

$$F(0,j) = dj$$
$$F(i,0) = di$$
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(A_i, B_j) \\ F(i,j-1) + d \\ F(i-1,j) + d \end{cases}$$

$$S(a,b) = \begin{cases} c_1 & \text{if } a = b, \\ -c_2 & \text{if } a \neq b. \end{cases}$$

Classical Sequence Alignment

- Dynamic Programming
- $O(n^2)$ in both time and space (score matrix)

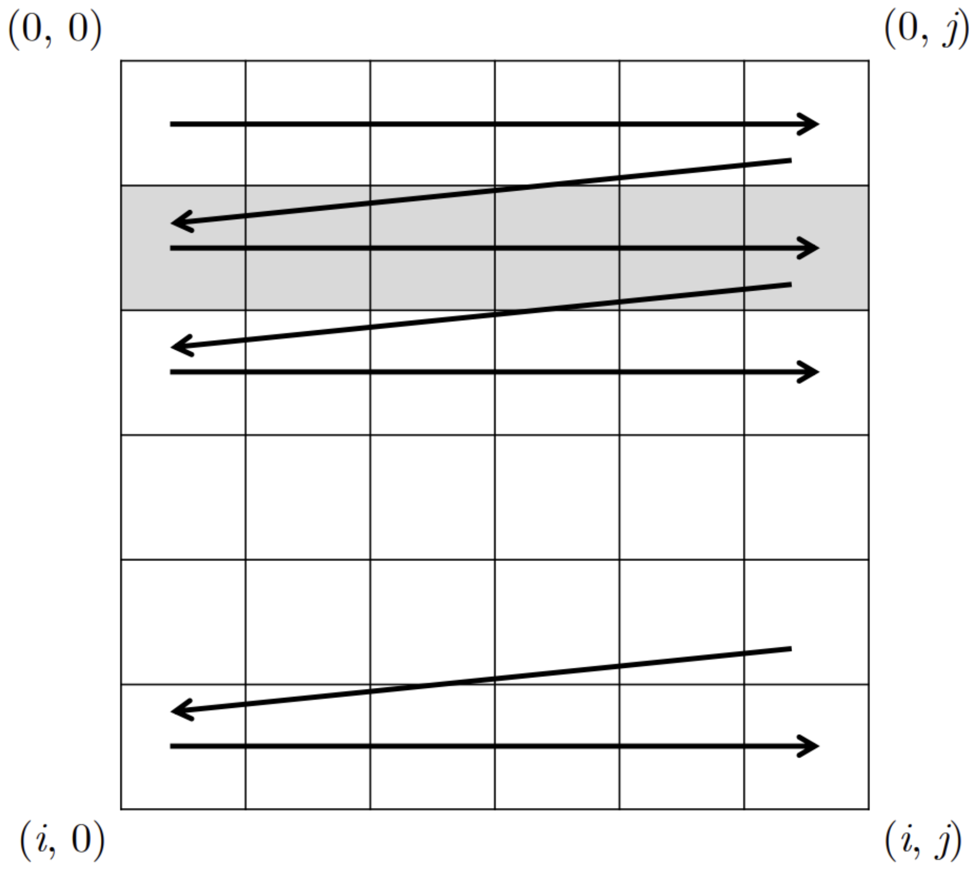


		T	G	C	G	T
	0	-2	-4	-6	-8	-10
A	-2					
A	-4					
C	-6					
G	-8					
G	-10					

Classical Sequence Alignment

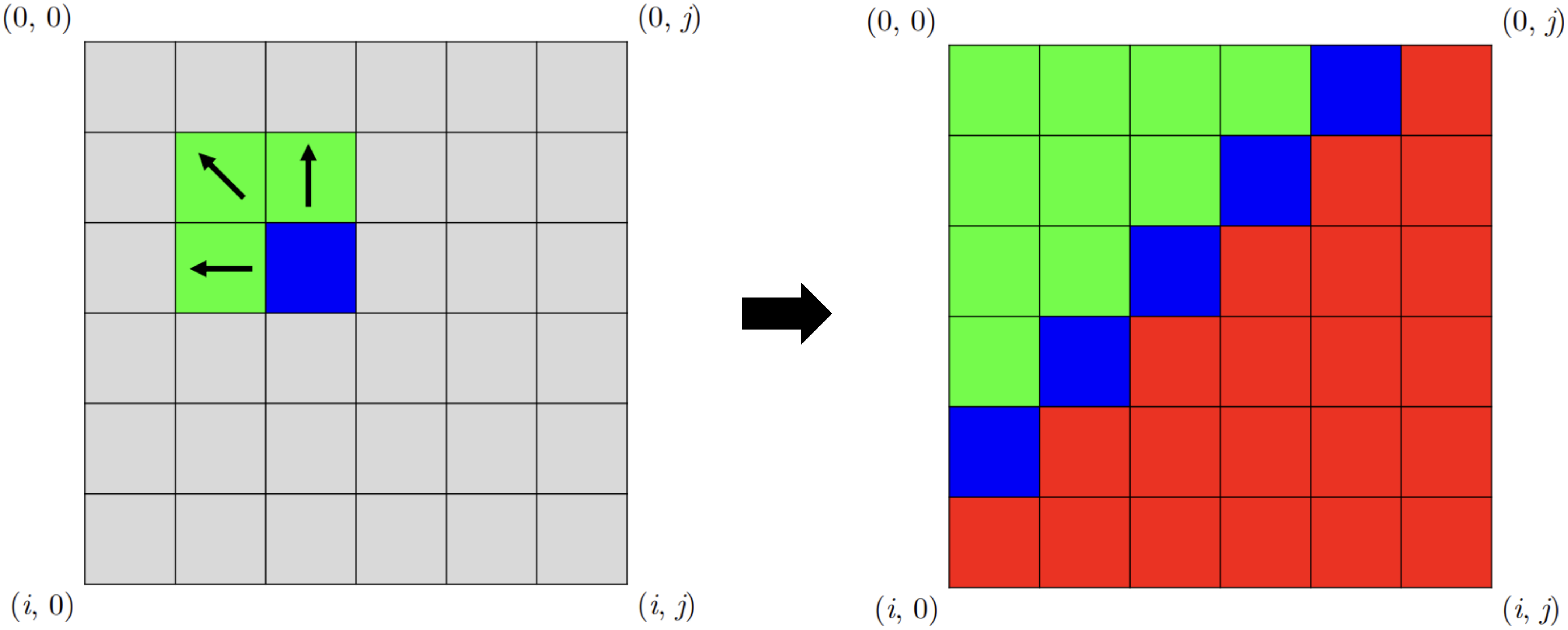
- Sequential Sweeping
- 2 nested for loops

- Can it be parallelized?
- What if the array is too large?



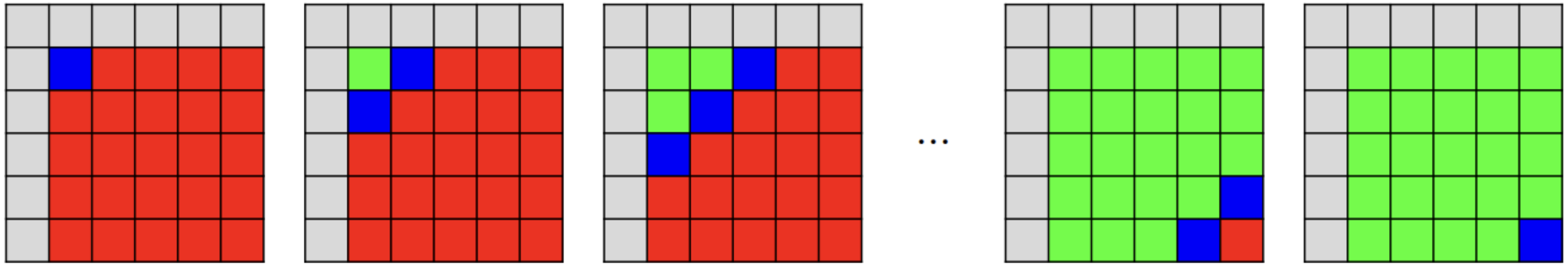
Classical Algorithms Parallelization

- Analyzing Cell's Dependencies



Classical Algorithms Parallelization

- Naïve Parallelization Method
- Doing each anti-diagonal wavefront in parallel
- Generally, slow access pattern for CPU
- Very slow in very large array that can't fit in CPU's cache (no spatial localization)
- Large threading overhead ratio.

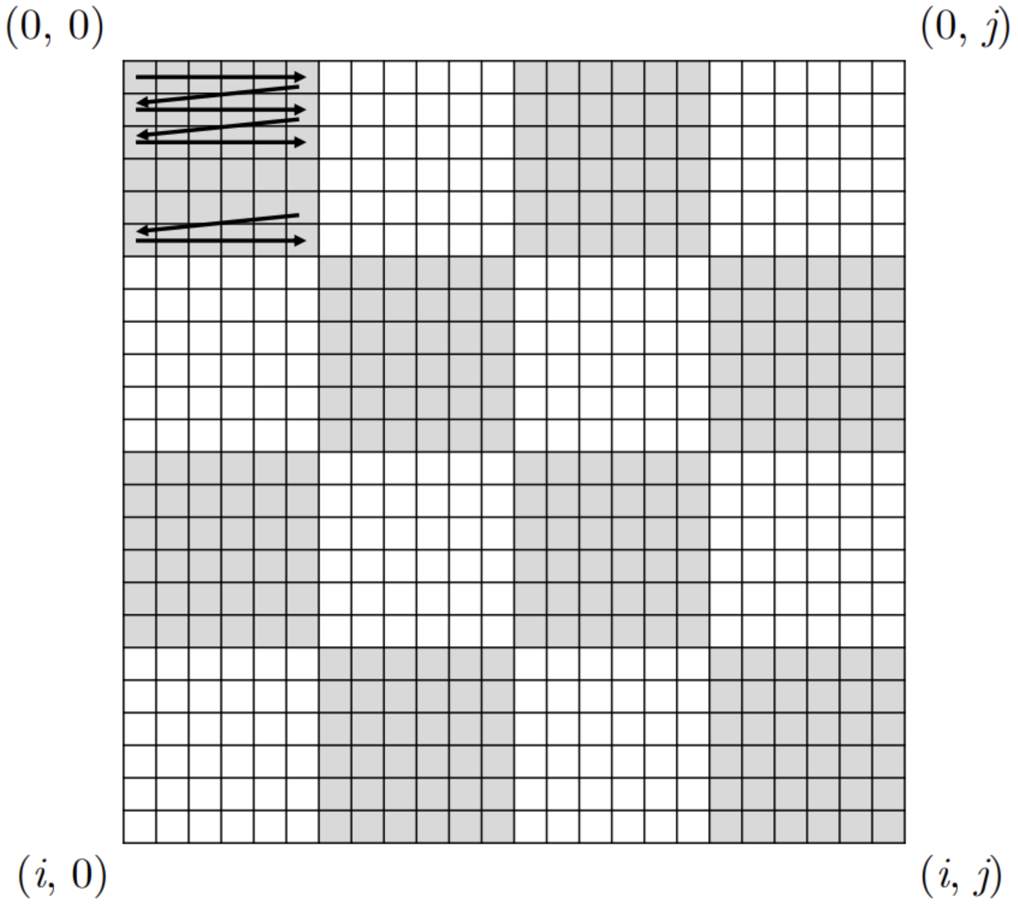


Iterations

Classical Algorithms Parallelization

- More thoughtful way:
  Split array into blocks conceptually

- Normal sweeping within each block

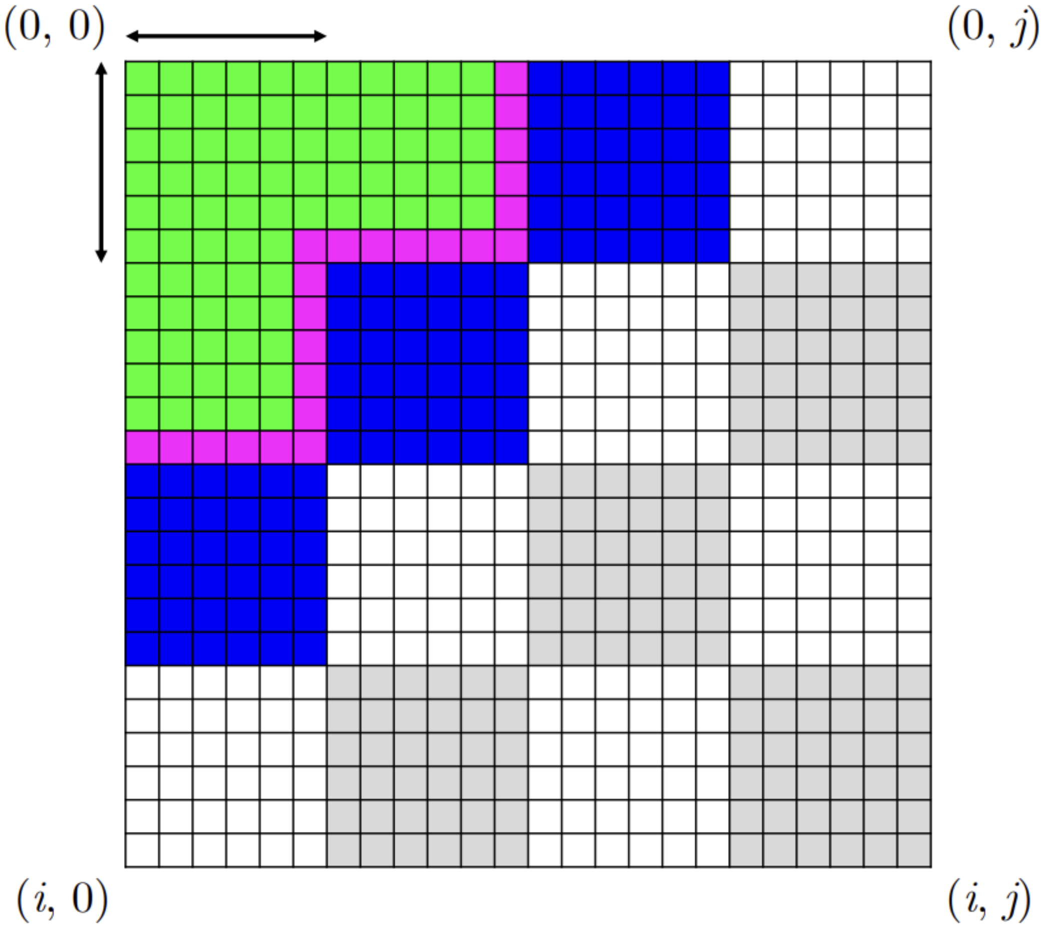


Classical Algorithms Parallelization

- Now, you can parallelize a wavefront in "blocks."

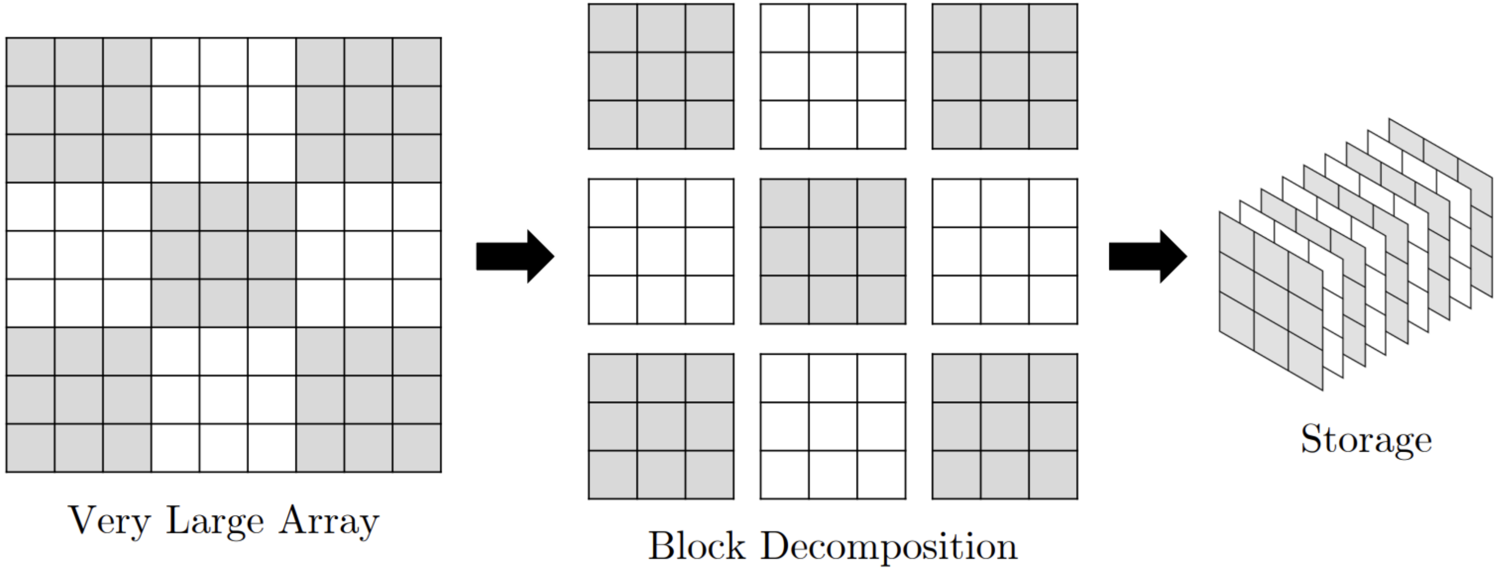
- Reduce threading overhead ratio
- Can fix localization problem
- Still taking very large memory

- How can we improve further?



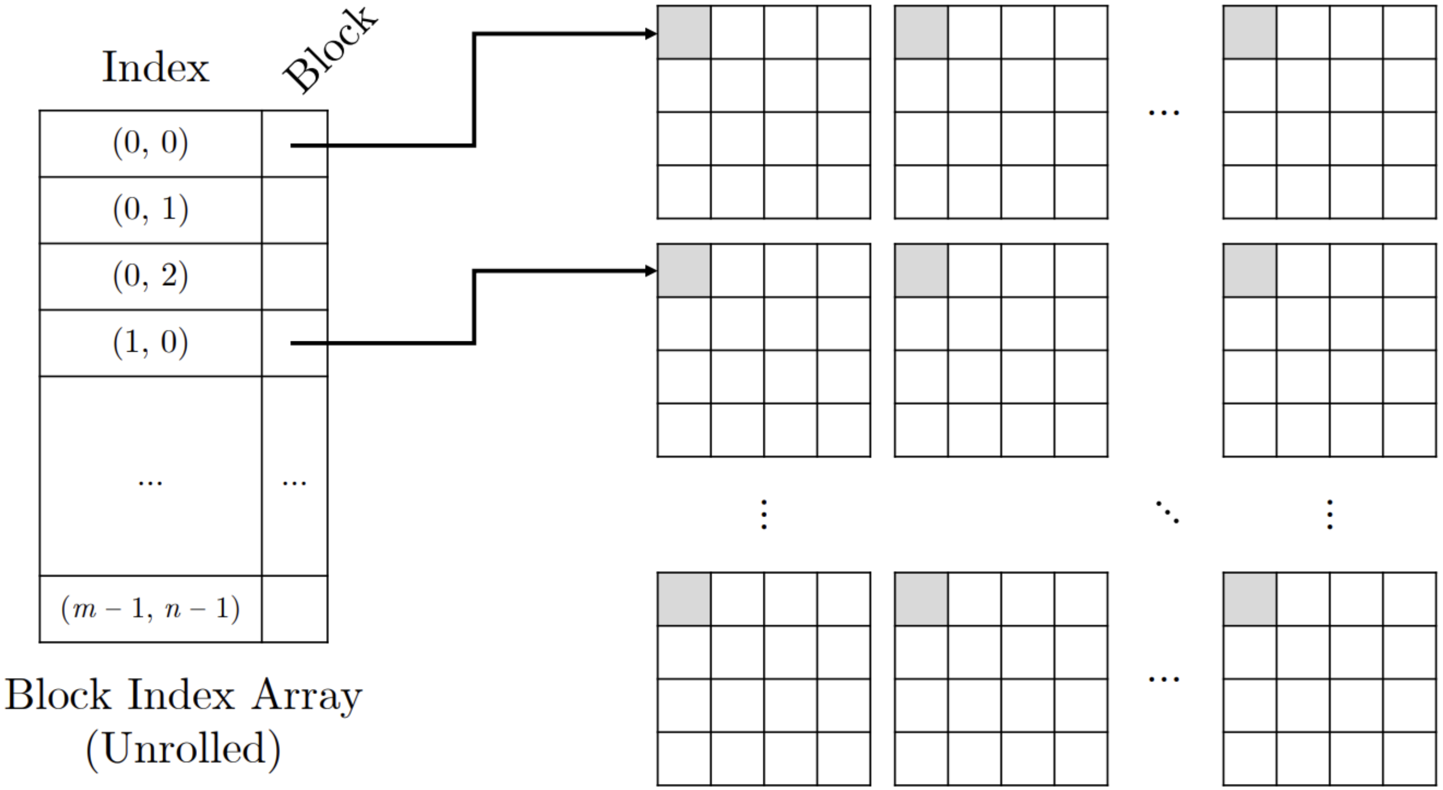
Classical Algorithms Parallelization

- Offloading and Lazy loading
- Load blocks currently in use at a time
- *Drawback: disk access is typically slower.*

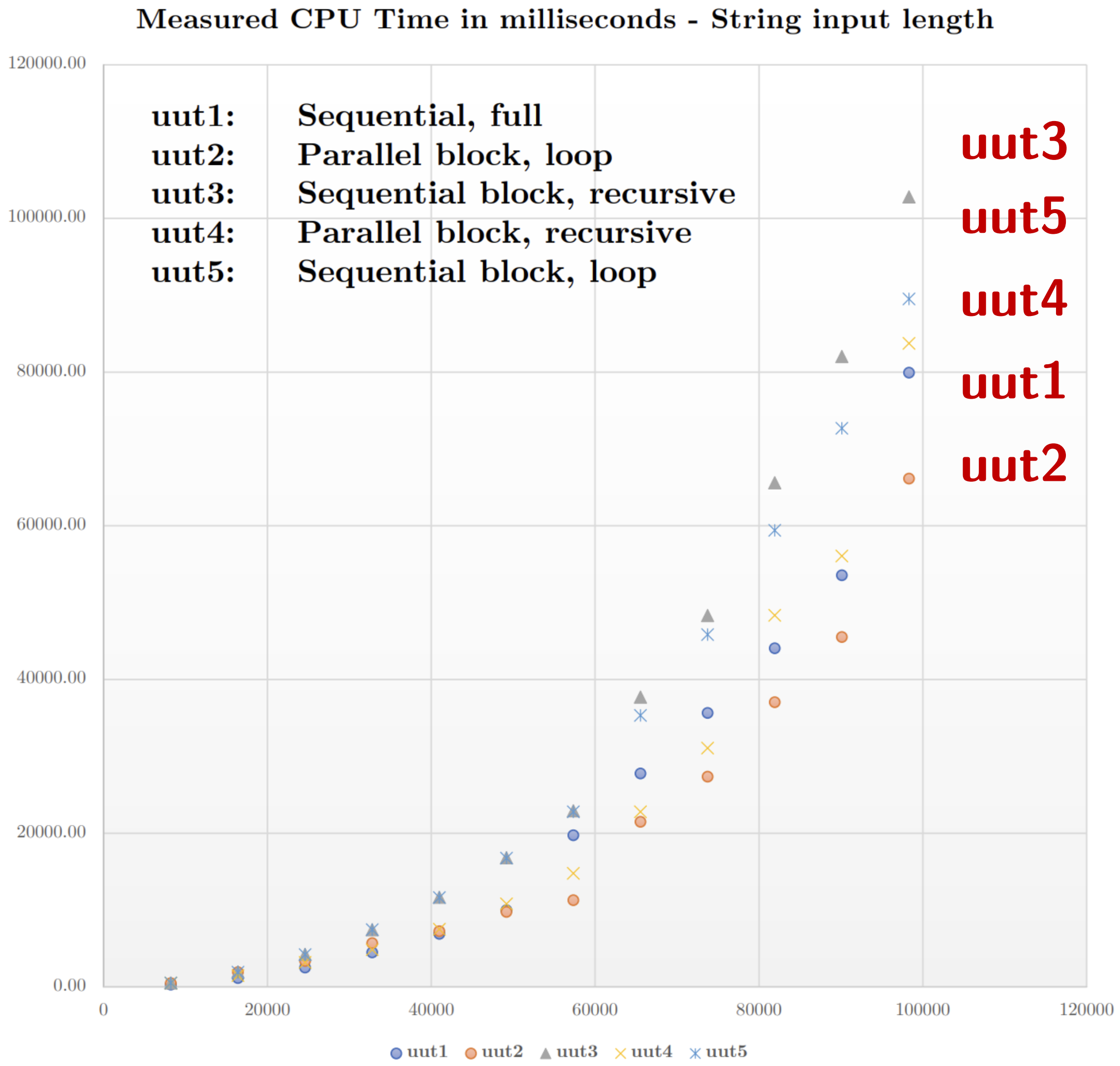


Classical Algorithms Parallelization

- Block access referencing done in index array and lookup function.



Classical Algorithms Parallelization



uut1: Sequential, full
uut2: Parallel block, loop
uut3: Sequential block, recursive
uut4: Parallel block, recursive
uut5: Sequential block, loop

uut3
uut5
uut4
uut1
uut2

Basics of Quantum Computing

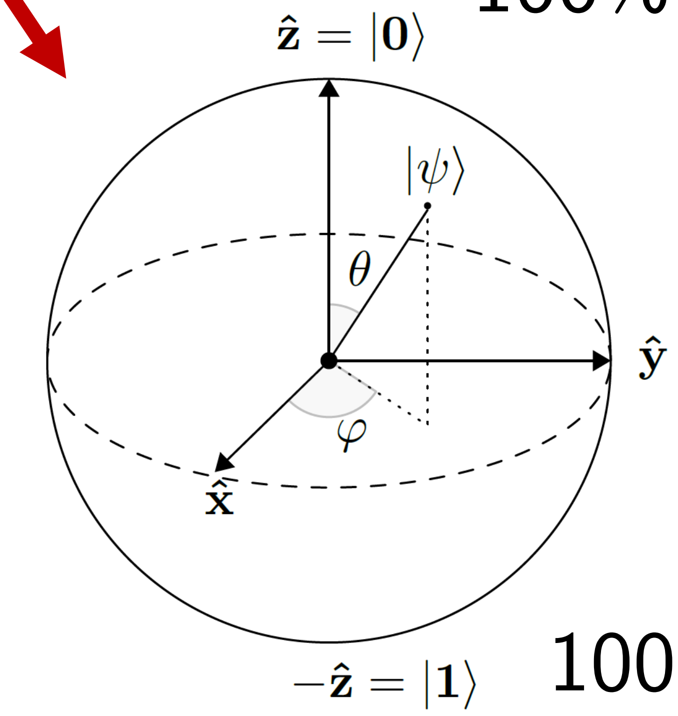
Qubits

Not real!

Either **0**

or **1**

Classical bit



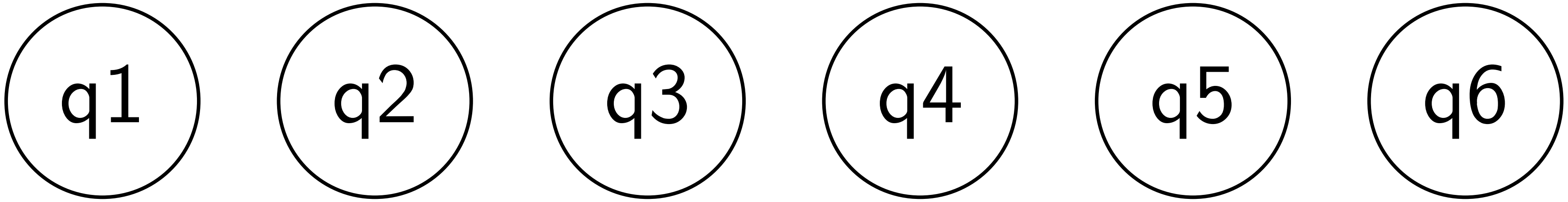
100% be 0

somewhere in between

100% be 1

Qubit representation
(Bloch sphere)

Superposition → Parallelism & Searching

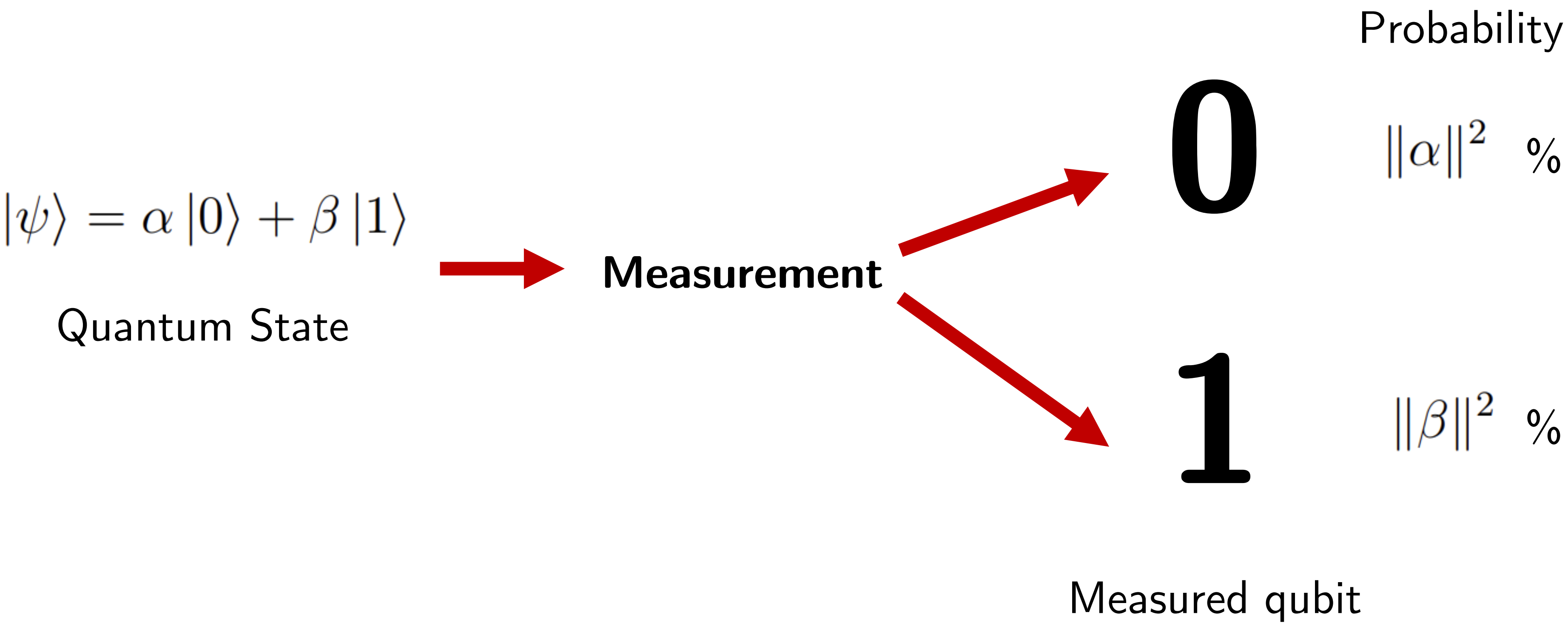


n qubits → Superposition of 2^n states.

$$\text{e.g., } \underbrace{000000,\ 000001,\ \ldots\ ,\ 111110,\ 111111}_{\textit{6 qubits} \rightarrow \textit{Superposition of } 64\ (2^6)\ \textit{states.}}$$

Basics of Quantum Computing

Decoherence



$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$

Quantum State

Measurement

Probability

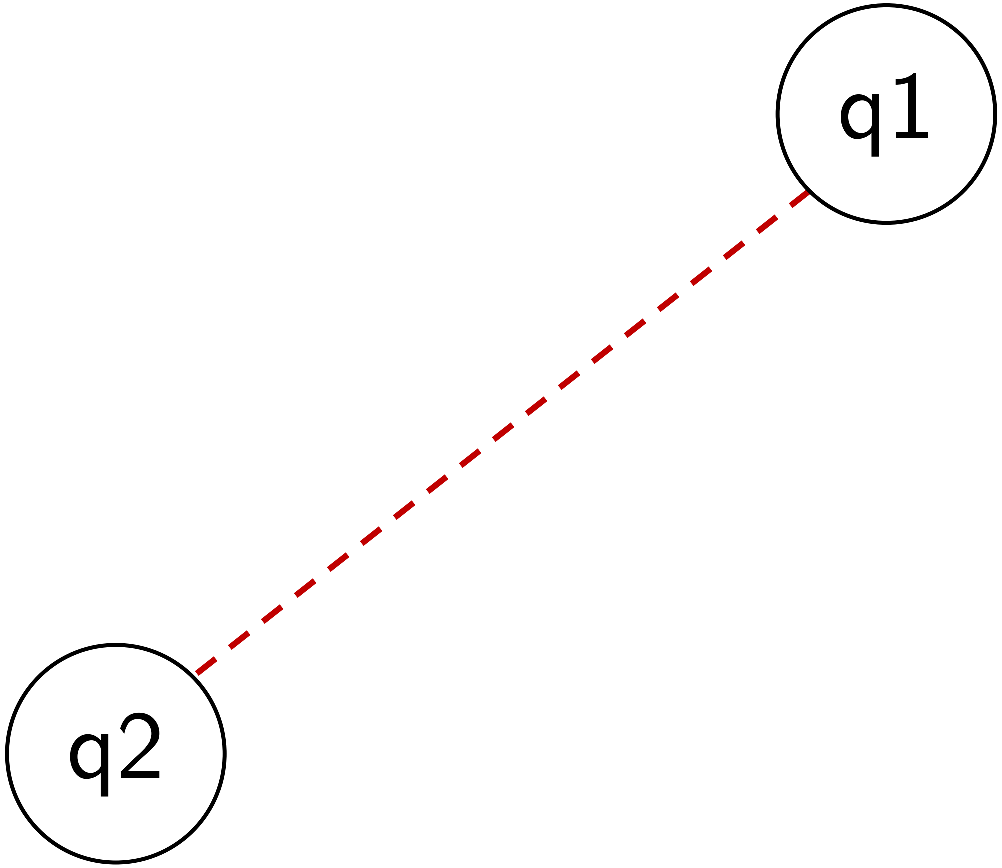
0 $\|\alpha\|^2$ %

1 $\|\beta\|^2$ %

Measured qubit

Basics of Quantum Computing

Quantum Entanglement



"Spooky action at a distance"
- Albert Einstein

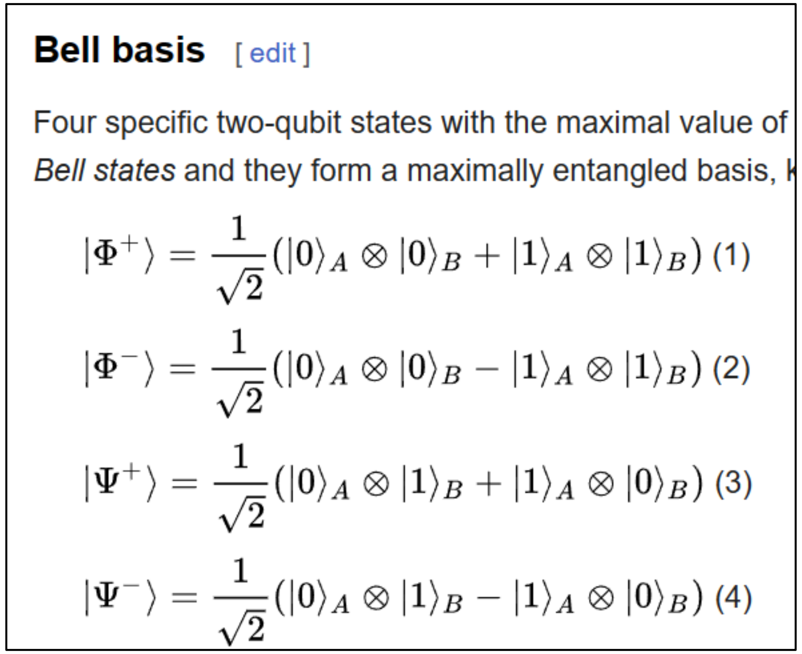
Simplification

Any changes on q1 affects q2 instantly regardless of distance between them.

Basics of Quantum Computing

Quantum Entanglement: Example

Bell State



Bell basis [edit]

Four specific two-qubit states with the maximal value of

Bell states and they form a maximally entangled basis,

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B) \quad (1)$$

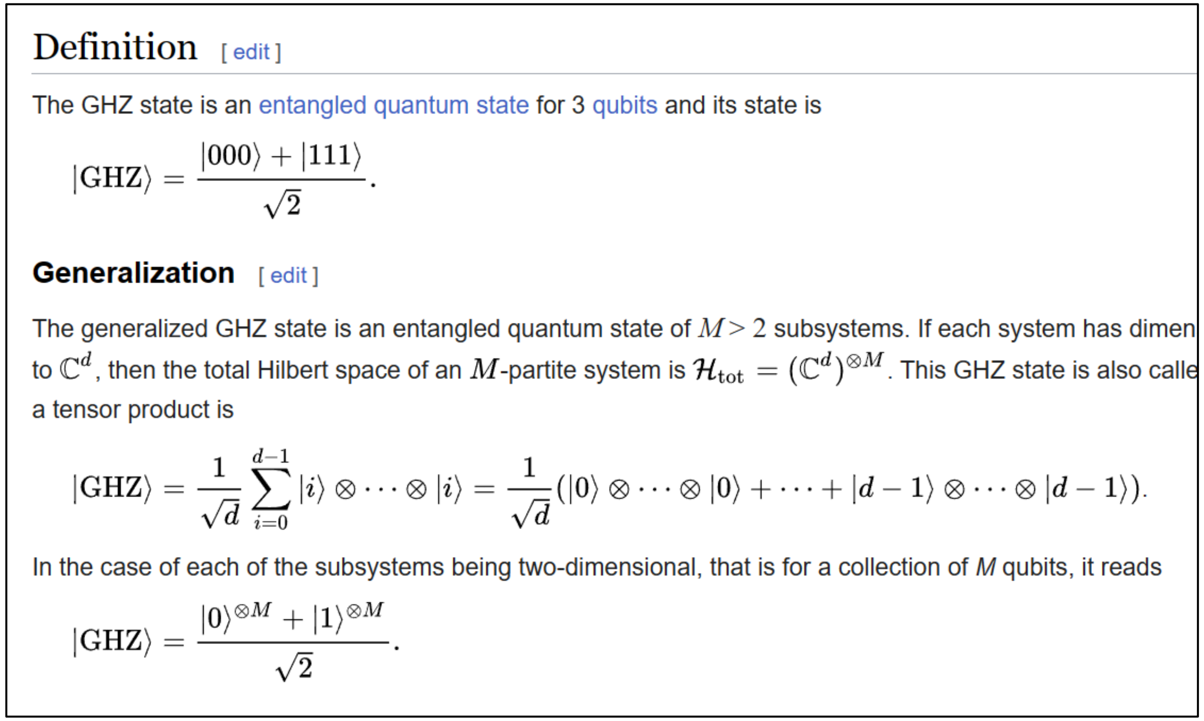
$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B) \quad (2)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B) \quad (3)$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B) \quad (4)$$

Wikipedia

GHZ State



Definition [edit]

The GHZ state is an entangled quantum state for 3 qubits and its state is

$$|\mathrm{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}.$$

Generalization [edit]

The generalized GHZ state is an entangled quantum state of $M > 2$ subsystems. If each system has dimen

to $\mathbb{C}^d$, then the total Hilbert space of an M-partite system is $\mathcal{H}_{\mathrm{tot}} = (\mathbb{C}^d)^{\otimes M}$. This GHZ state is also calle

a tensor product is

$$|\mathrm{GHZ}\rangle = \frac{1}{\sqrt{d}}\sum_{i=0}^{d-1}|i\rangle \otimes \cdots \otimes |i\rangle = \frac{1}{\sqrt{d}}(|0\rangle \otimes \cdots \otimes |0\rangle + \cdots + |d-1\rangle \otimes \cdots \otimes |d-1\rangle).$$

In the case of each of the subsystems being two-dimensional, that is for a collection of *M* qubits, it reads

$$|\mathrm{GHZ}\rangle = \frac{|0\rangle^{\otimes M} + |1\rangle^{\otimes M}}{\sqrt{2}}.$$

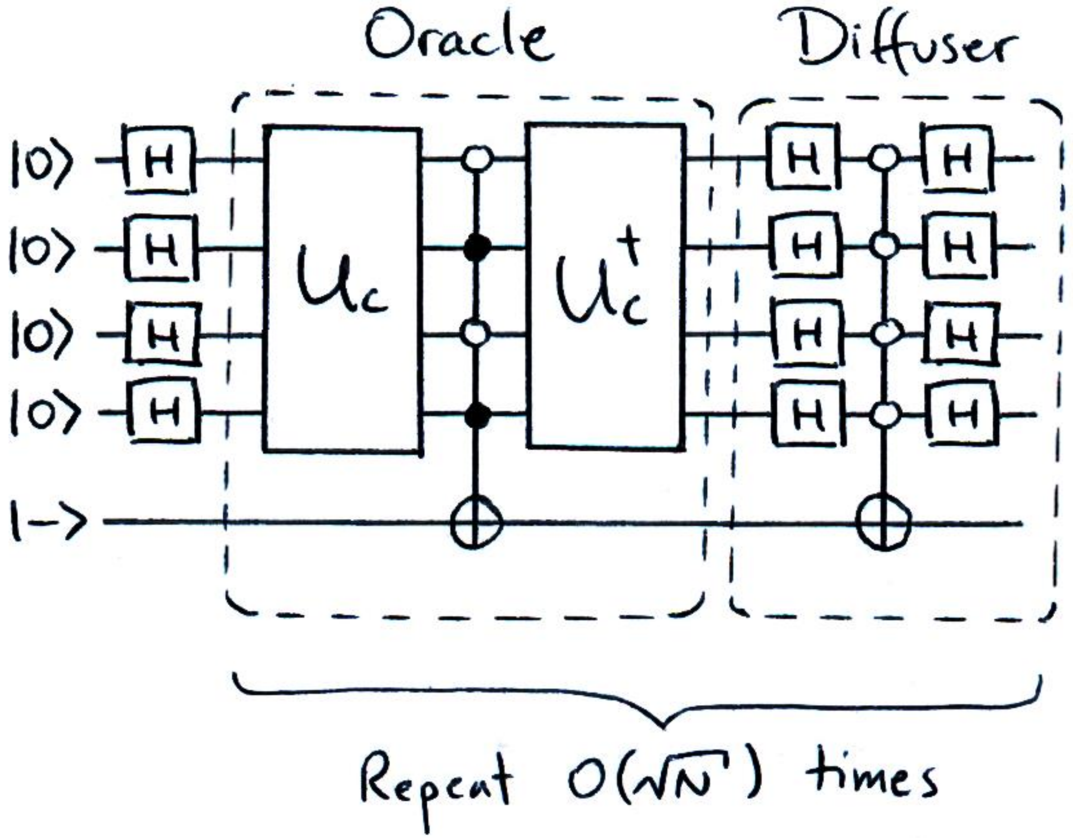
Wikipedia

Basics of Quantum Computing

Quantum Algorithm Example

- Grover's Algorithm for searching database.
  - Focuses on unordered search
  - Query only $O(\sqrt{N})$ times.

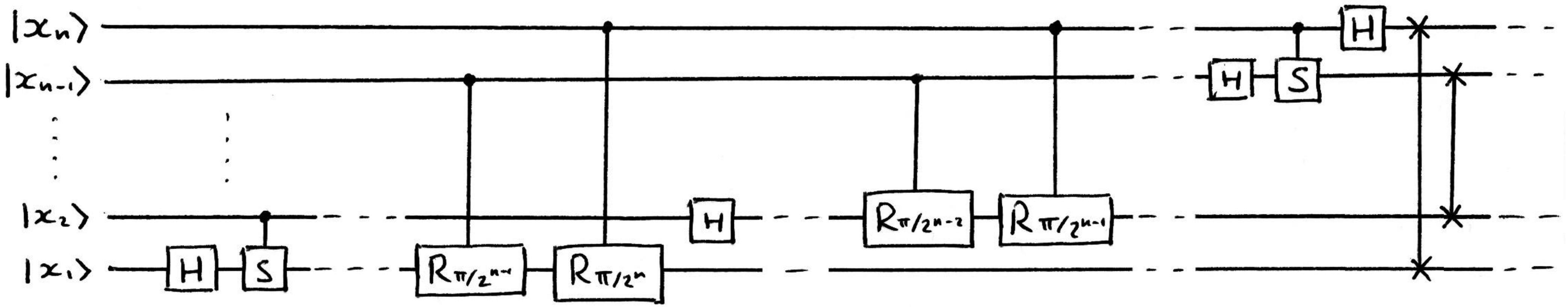
- **Oracle**: mark correct answer by applying negative phase.
- **Diffuser**: amplify correct answer back to original phase.



https://cnot.io

Quantum Algorithm Example

- Quantum Fourier Transform ⇔ Quantum version of DFT



https://cnot.io

Basics of Quantum Computing

Some Basic Quantum Gates

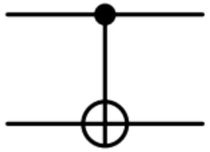
Single qubit gates

Hadamard	$-\boxed{H}-$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$
Pauli-X (Bit flip)	$-\boxed{X}-$	$\sigma_X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$
Pauli-Z (Phase flip)	$-\boxed{Z}-$	$\sigma_Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$

Some Basic Quantum Gates

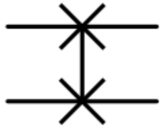
Two qubit gates

Controlled-NOT (CNOT, CX)



$$\begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix}$$

SWAP



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Quantum Algorithms for Sequence Alignment

- Hamming Edit Distance (Possible, Implementable)
- Levenshtein Edit Distance (Theoretical with qRAM)
- Needleman-Wunsch (Theoretical with qRAM)
- Smith-Waterman (Theoretical with qRAM)
- Cosine Similarity (Possible)
- Graph Edit Distance (Theoretical)
- Pattern Matching Approximation with QFT (Quantum DFT/FFT)
- BLAST Database Search Matching (Never proven as of now)
- Knuth-Morris-Pratt string search algorithm (Never proven as of now)

Quantum Algorithms for Sequence Alignment

- ***Hamming Edit Distance (Possible, Implementable)***
- Levenshtein Edit Distance (Theoretical with qRAM)
- Needleman-Wunsch (Theoretical with qRAM)
- Smith-Waterman (Theoretical with qRAM)
- Cosine Similarity (Possible)
- Graph Edit Distance (Theoretical)
- ***Pattern Matching Approximation with QFT (Quantum DFT/FFT)***
- BLAST Database Search Matching (Never proven as of now)
- Knuth-Morris-Pratt string search algorithm (Never proven as of now)

Quantum Algorithms for Sequence Alignment

- DNA Sequence string is represented by $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^n$
- Sequence Encoding
  - Minimal Encoding (2 bits): $\{00, 01, 10, 11\}$
  - One-hot Encoding (4 bits): $\{0001, 0010, 0100, 1000\}$
- #Qubits needed if:
  - 2 input quantum registers
  - 1 output quantum register
  - 1 output classical register (for measurement)
- $= 3 \times N_{\text{encode length}} \times N_{\text{string length}}$

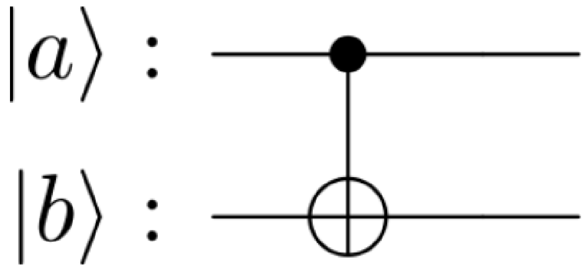
- Example: 127-qubit system supports 42-bit string (21 characters) maximum.

Quantum Algorithms for Sequence Alignment

- Concept: “Pairwise Comparison”

- In classical term, we would use XOR operation.
- In quantum system, we can use Controlled-NOT (CNOT, CX).

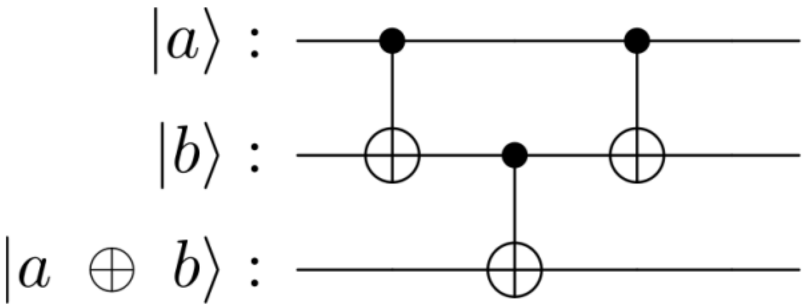
- Naïve approach turns $|b\rangle \rightarrow |a \oplus b\rangle$



- Input qubits are not conserved!

Quantum Algorithms for Sequence Alignment

- We can add an output quantum register.
- CNOT is its own inverse, so we can “sandwich” operations.

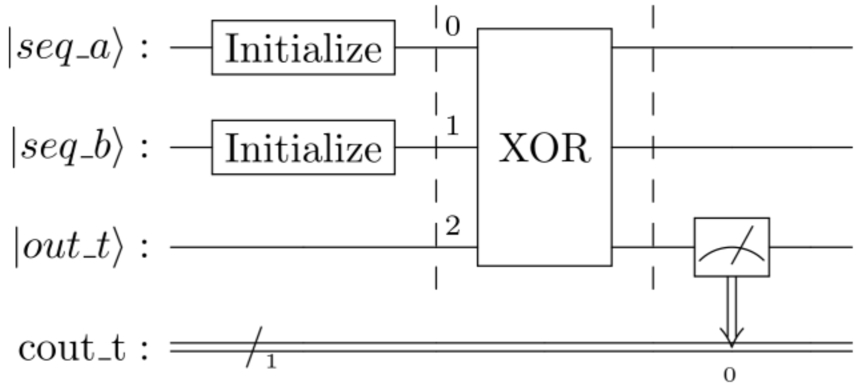


- Now, input qubits can conserve (if they are not entangled).

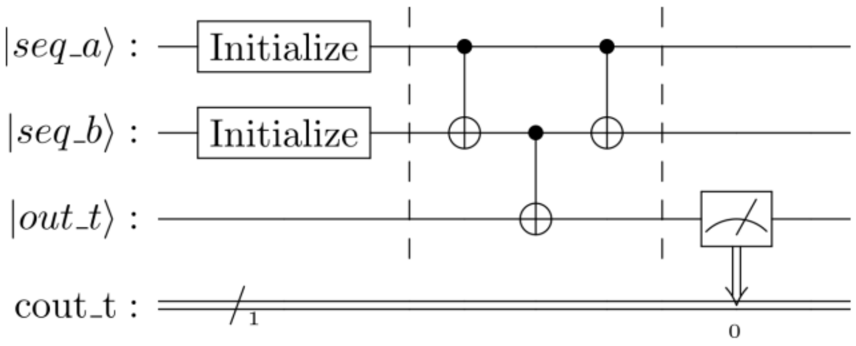
- *If input qubits are entangled, the output qubit will also be entangled with input qubits. Measurement will collapse the superposition.*

Quantum Algorithms for Sequence Alignment

- Method 1: Direct pairwise comparison using XOR



(a) Top-level block



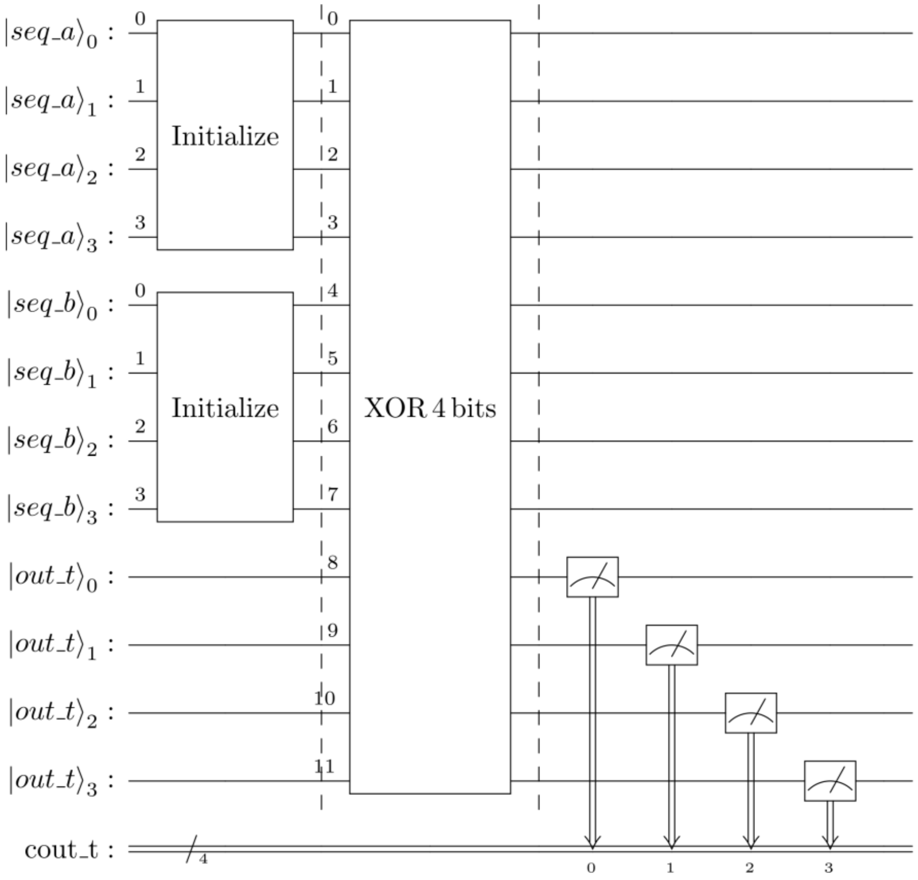
(b) Decomposed XOR

Quantum Algorithms for Sequence Alignment

- Method 1: Direct pairwise comparison using XOR

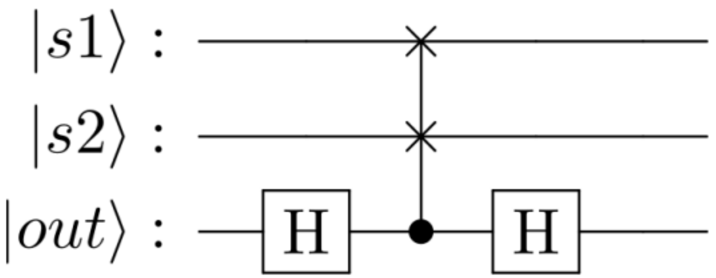
- Example:
  2-character (4-bit) input

- Potential improvements:
  - Implement quantum adder circuit
    - QFT Adder
    - Ripple Carry Adder
    - Adder using QPE



Quantum Algorithms for Sequence Alignment

- Method 2: Direct pairwise comparison using Swap Test



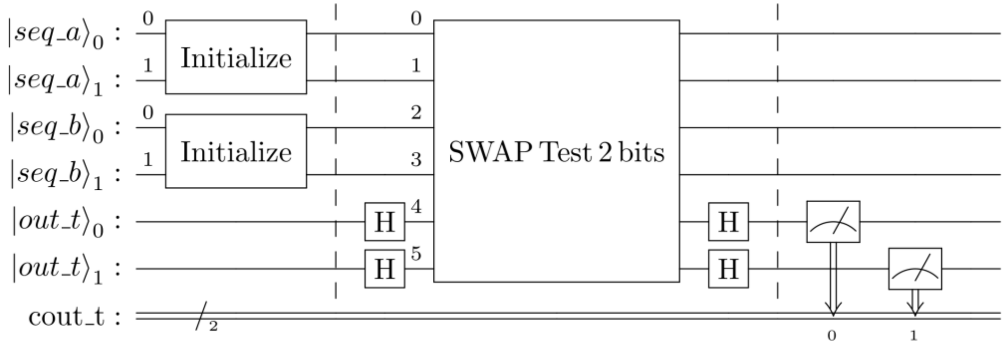
- This utilizes quantum properties of putting a control qubit in a superposition.
- This causes qubits entanglement.
- How do we determine the output?
    - If s1 equals s2, the output measurement is always 0.
    - Else, the output measurement is 0 for 50% and 1 for 50%.

Quantum Algorithms for Sequence Alignment

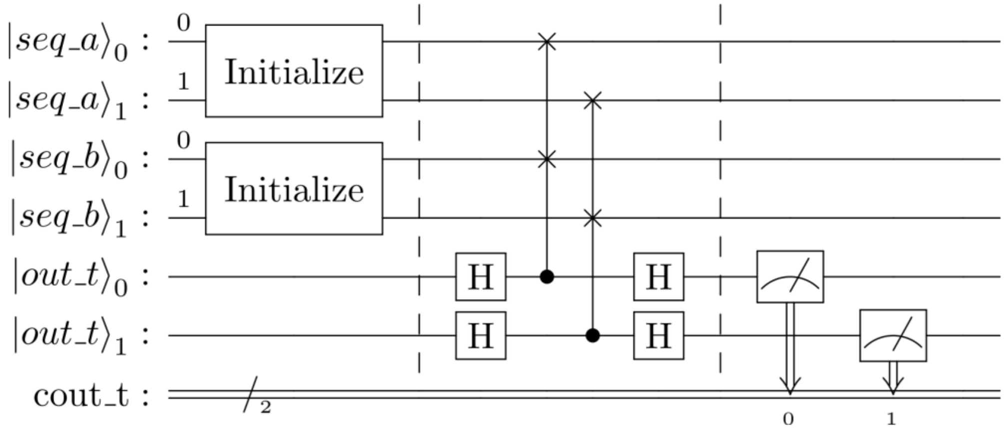
- Method 2: Direct pairwise comparison using Swap Test

- Example:
  1-character (2-bit) input

- What if we treated it as a signal?
  - Time-domain Comparison
  - Frequency-domain Comparison



(a) Top-level block

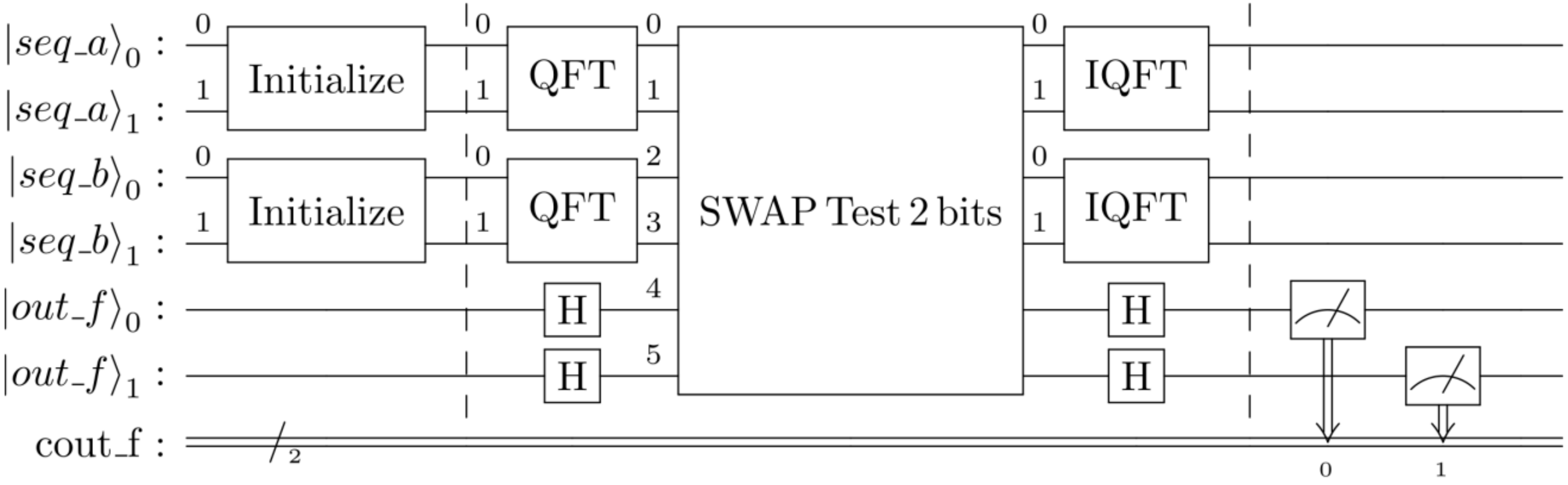


(b) Decomposed Swap Test

Quantum Algorithms for Sequence Alignment

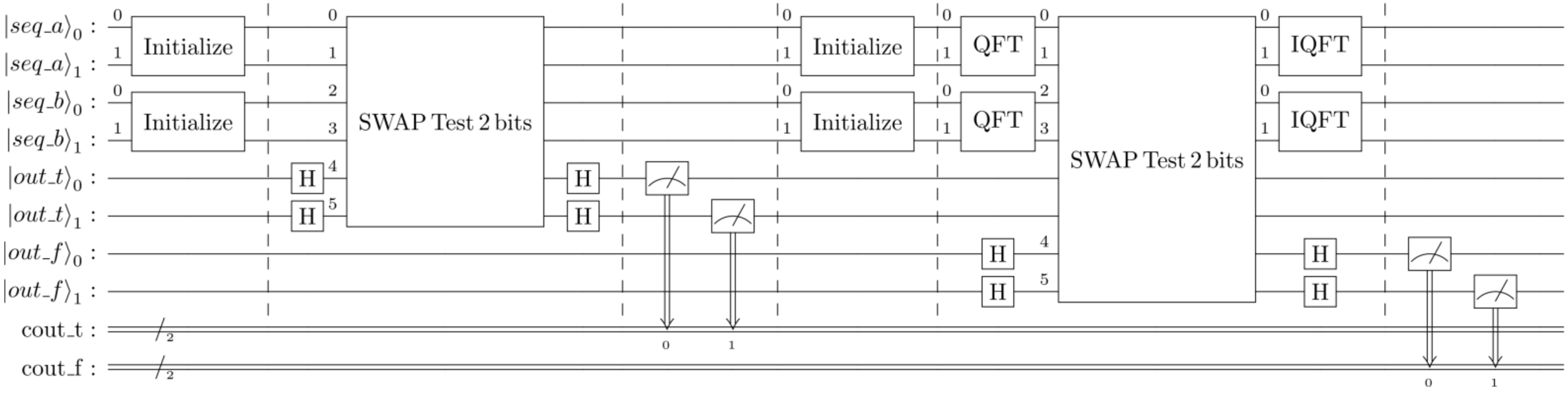
- Addition to Method 2: Compare in a Frequency Domain
- We can apply QFT and QFT^{-1} on both input quantum registers.
  - The state after initialization can be described as:

$$(\mathrm{H}\,|0\rangle)^{\otimes n} \otimes \mathrm{QFT}\,|s_1\rangle \otimes \mathrm{QFT}\,|s_2\rangle$$



Quantum Algorithms for Sequence Alignment

- Can we combine both domain in 1 circuit?
  - Yes, but it would be impractical: large circuit, potential noise, rigorous post-processing.
  - After the Swap Test and measurement, qubits are not reusable.
  - Must initialize qubits again.*



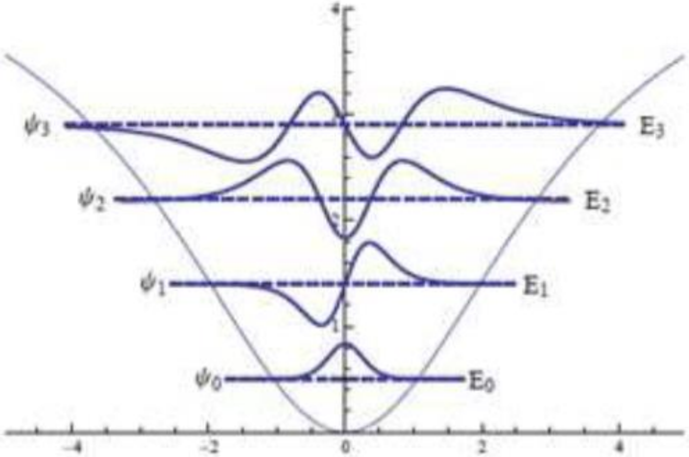
Quantum Algorithms for Sequence Alignment

- Further improvements:
  - I. Swap Test on every permutation of qubits.
    - From n to n^2

$$s_1\{0,1,...,n-1\} \times s_2\{0,1,...,n-1\}$$

  - II. Using QPE for frequency domain
  - III. Maybe go for a qudit (d-level quantum system)?

$$|\psi\rangle = \sum_{i=0}^{d-1} c_i |i\rangle \quad ; \quad \sum_{i=0}^{d-1} \|c_i\|^2 = 1$$



Non-linear oscillator

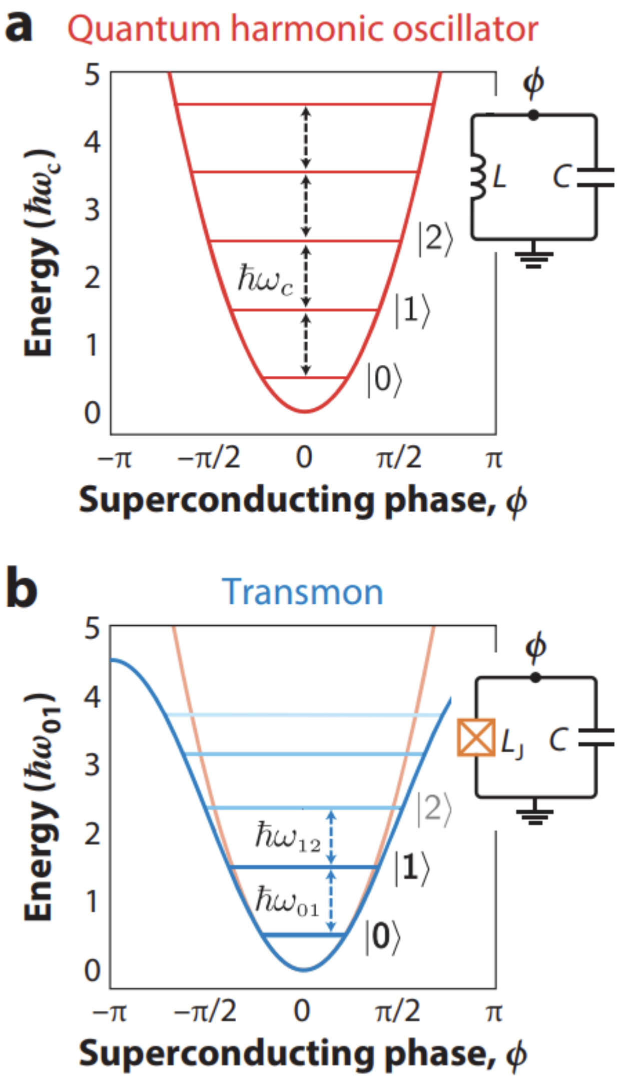
10.1063/1.4954283

A quantum computer

- There are many types of quantum computers!
- Analog Quantum Computer
  - Initializing a quantum state
  - Control the Hamiltonian to evolve the state directly
  - E.g., quantum annealing [D-Wave!] , adiabatic computation, quantum simulation
- Digital Quantum Computer
  - It is "gate-based" with universal set of gates
  - Typically, is a two-level quantum system.
  - Digital outcomes by measurement
  - Similar to classical computing.

A quantum computer

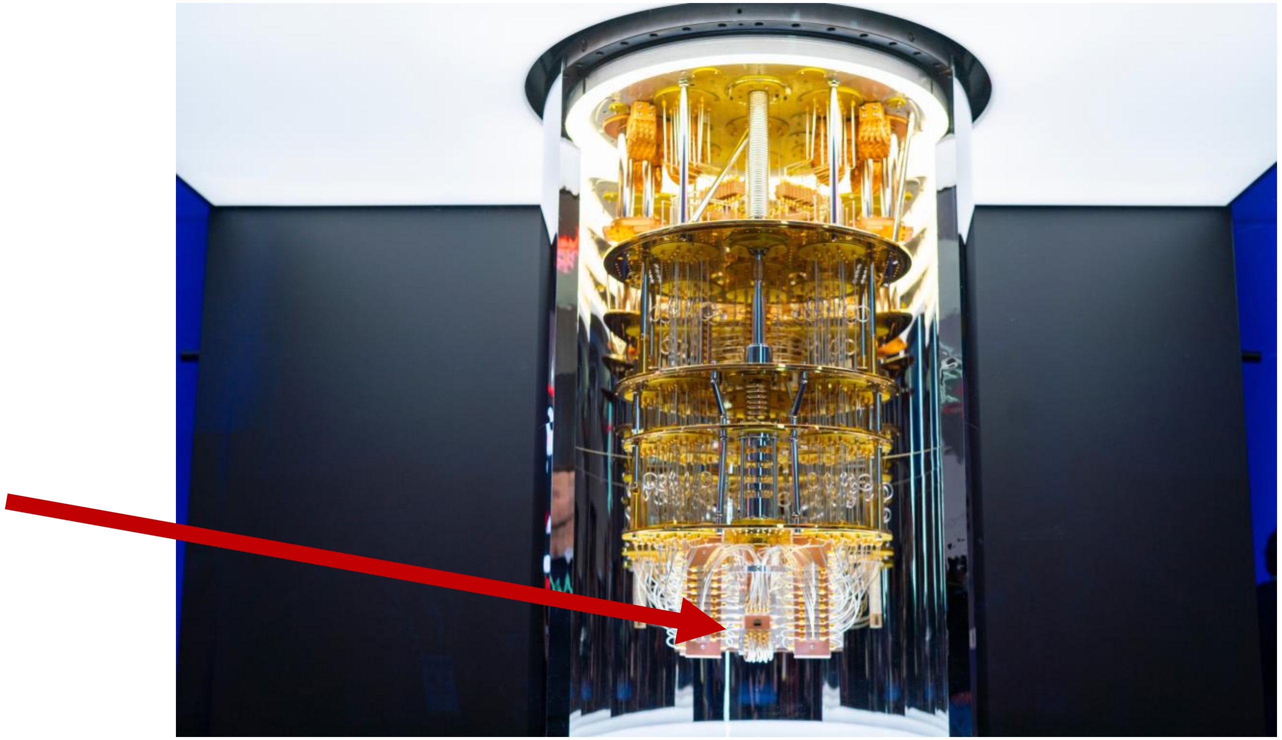
- There are also many qubit technologies used!
- Current technologies in NISQ era
  - Superconducting qubits
    - Transmon charge, nonlinear
  - Trapped ion
- Candidate technologies after NISQ
  - Photonic
  - Silicon-based
  - Topological



Superconducting Qubits: Current State of Play (annualreviews.org)

A quantum computer

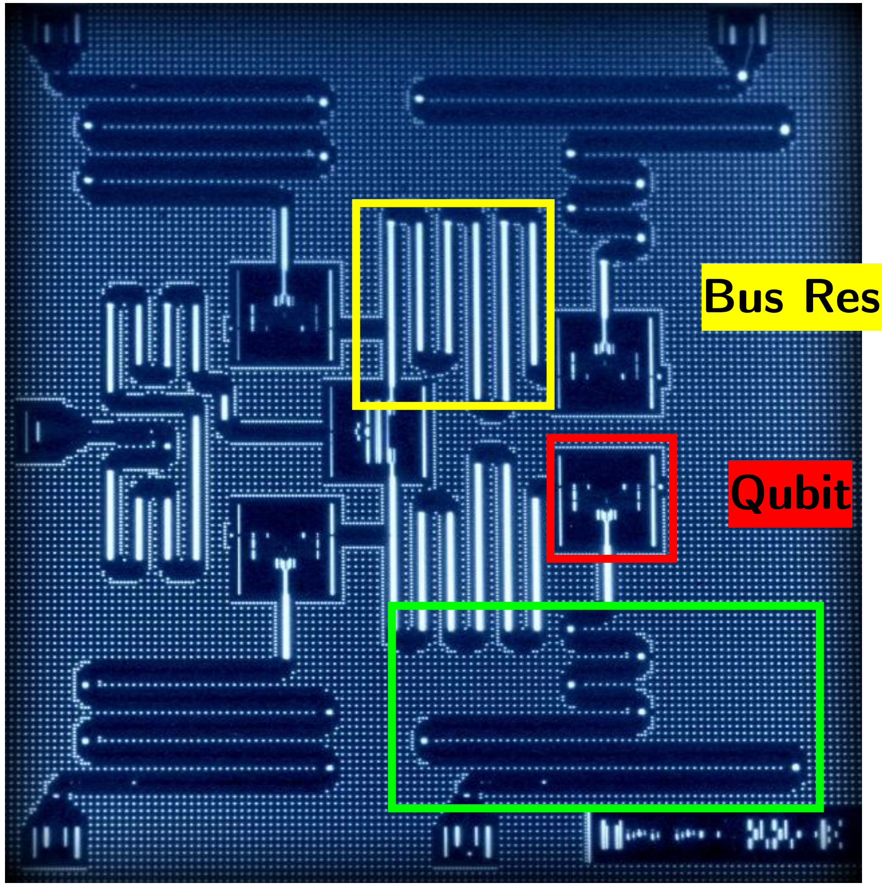
- This is IBM’s 127 qubit quantum computer on which the circuit was run.



- Courtesy: IBM Quantum

A quantum computer

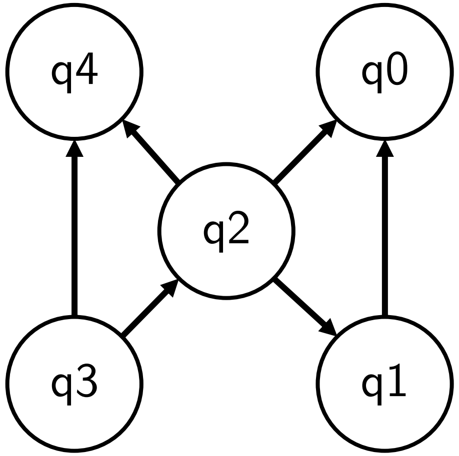
- Superconducting 5-qubit quantum processor (IBM).



Bus Resonator

Qubit

Control and Readout Resonator

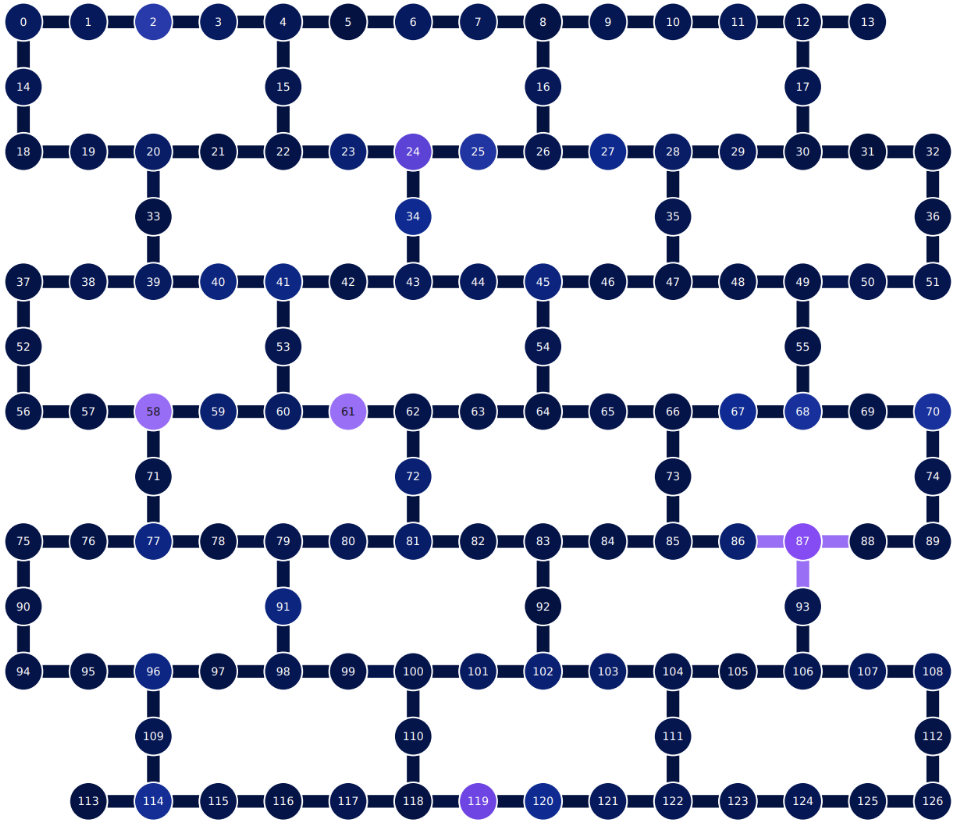


Qubit Diagram

Courtesy: IBM Quantum

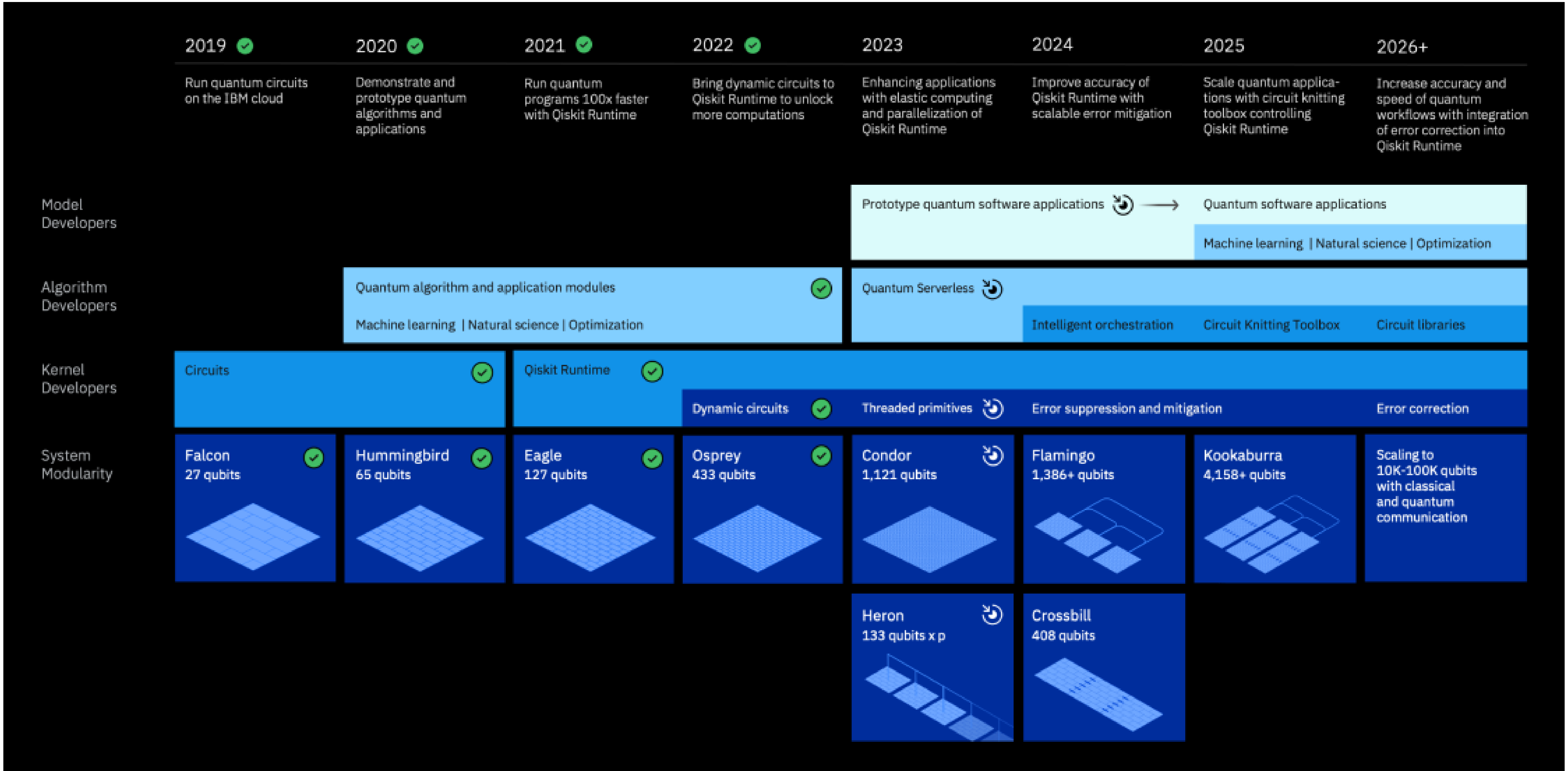
A quantum computer

- Superconducting 127-qubit quantum processor lattice (IBM).



Courtesy: IBM Quantum

A quantum computer: IBM Roadmap



2019 ✓	2020 ✓	2021 ✓	2022 ✓	2023	2024	2025	2026+
Run quantum circuits on the IBM cloud	Demonstrate and prototype quantum algorithms and applications	Run quantum programs 100x faster with Qiskit Runtime	Bring dynamic circuits to Qiskit Runtime to unlock more computations	Enhancing applications with elastic computing and parallelization of Qiskit Runtime	Improve accuracy of Qiskit Runtime with scalable error mitigation	Scale quantum applications with circuit knitting toolbox controlling Qiskit Runtime	Increase accuracy and speed of quantum workflows with integration of error correction into Qiskit Runtime

Model Developers

Prototype quantum software applications → Quantum software applications

Machine learning | Natural science | Optimization

Algorithm Developers

Quantum algorithm and application modules ✓

Machine learning | Natural science | Optimization

Quantum Serverless

Intelligent orchestration

Circuit Knitting Toolbox

Circuit libraries

Kernel Developers

Circuits ✓

Qiskit Runtime ✓

Dynamic circuits ✓

Threaded primitives

Error suppression and mitigation

Error correction

System Modularity

Falcon ✓
27 qubits

Hummingbird ✓
65 qubits

Eagle ✓
127 qubits

Osprey ✓
433 qubits

Condor
1,121 qubits

Flamingo
1,386+ qubits

Kookaburra
4,158+ qubits

Scaling to
10K-100K qubits
with classical
and quantum
communication

Heron
133 qubits x p

Crossbill
408 qubits

Experimentation Setup

Translate + Compile

- Used IBM Qiskit framework (Python) to generate and transpile circuits.

- Experiment Setups:
  - Control: Running circuits on IBM's matrix product state (MPS) simulator
  - Independent: Running circuits on 127-qubit IBM Eagle r3 (ibm_brisbane)
  - Trials on 8-bit, 16-bit, 32-bit, 40-bit controlled encoded sequences.

- Circuits Tested:
  - Time-domain comparison with XOR (t dom XOR)
  - Time-domain comparison with Swap Test (t dom SWAP)
  - Frequency-domain comparison with Swap Test (f dom SWAP)

Initial Results: 8 bits

Sequences `TTGC` and `TGCT` are the test sample.

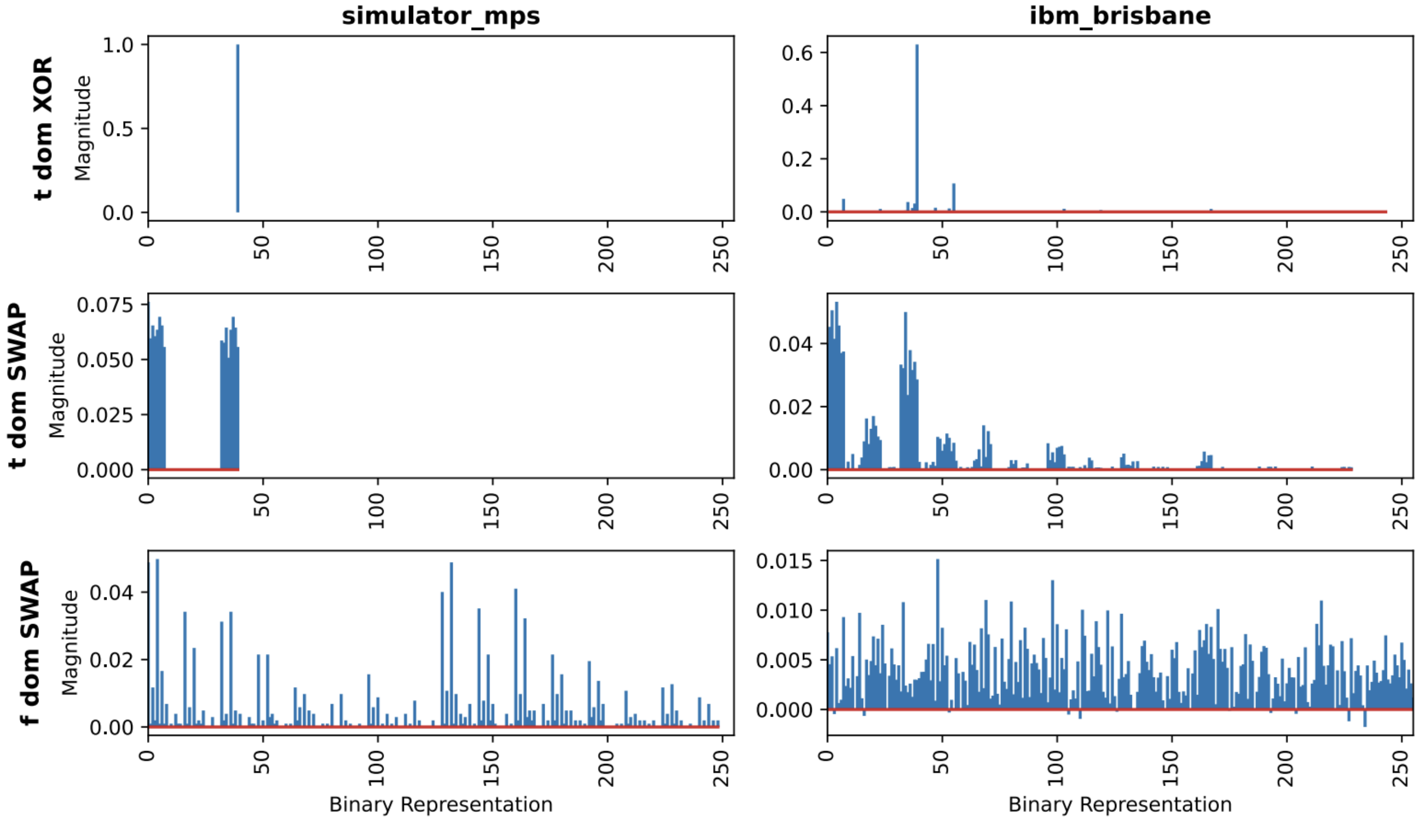


Figure 31: 8-bit String Running Results Comparison

Initial Results: 16 bits

Sequences `ATGCTTGC` and `TGCCTGCA` are the test sample.

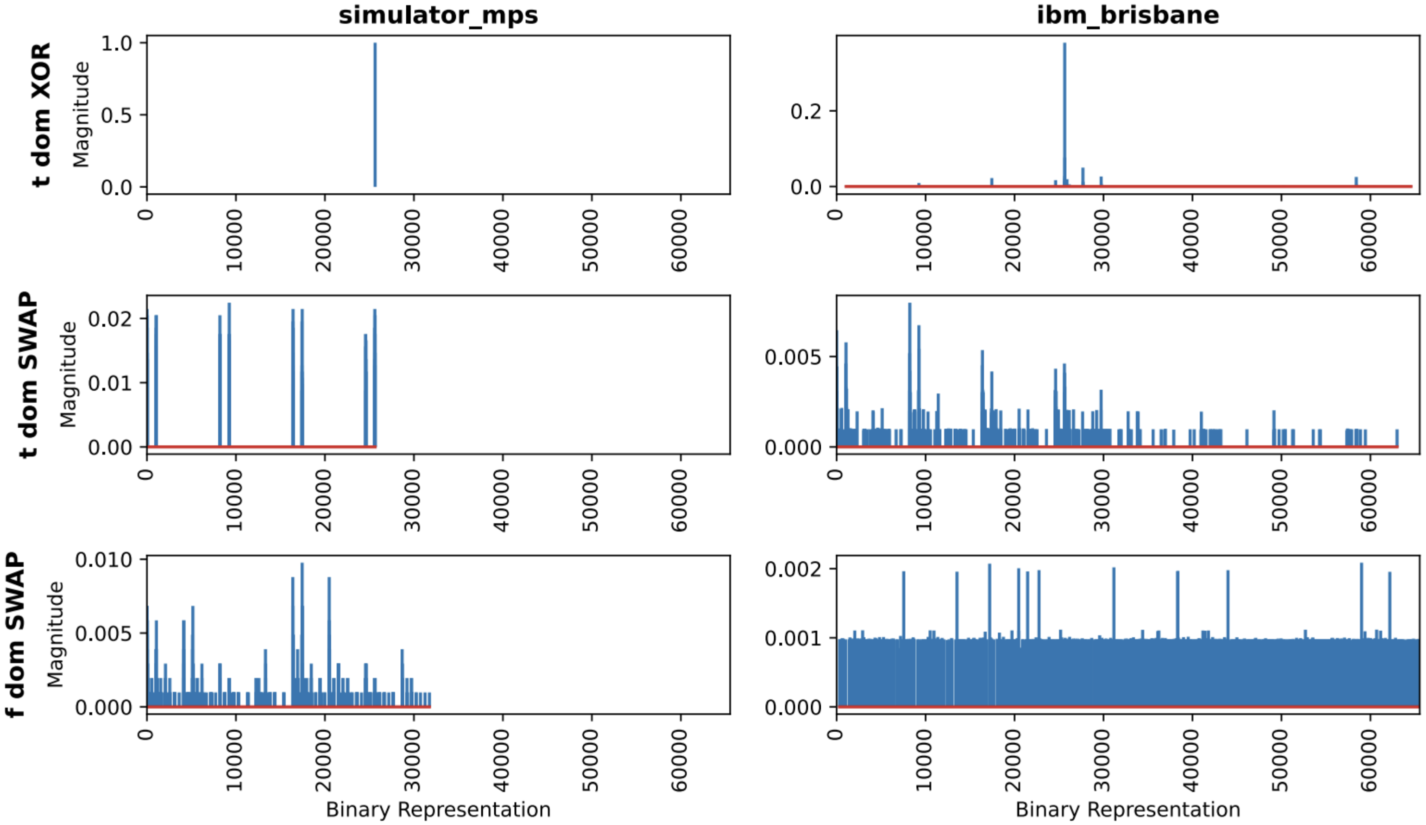


Figure 32: 16-bit String Running Results Comparison

Initial Results: 32 bits

Sequences `ATGCTTGCGGGGGGGG` and `TGCCTGCACGCGCGCA` are the test sample.

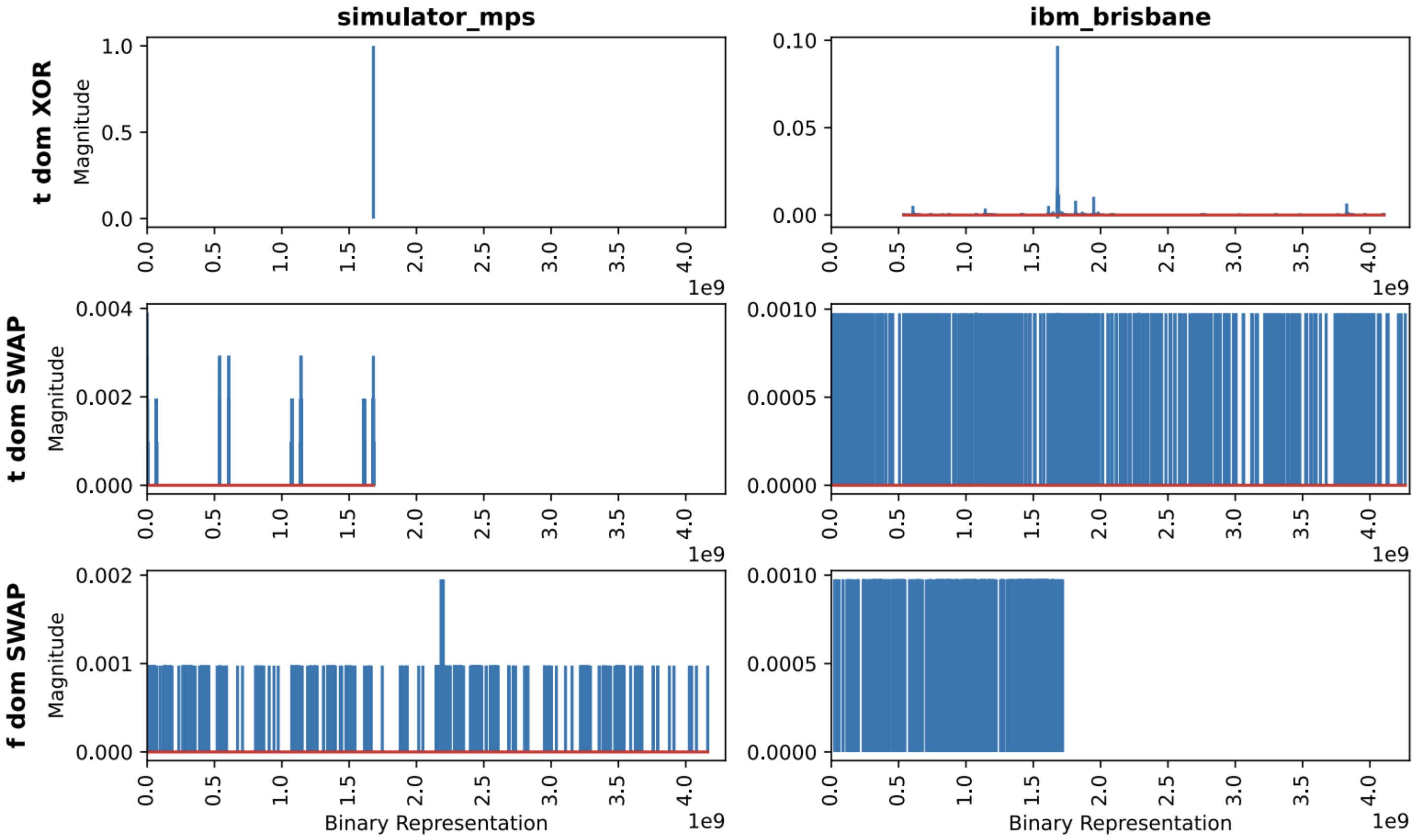


Figure 33: 32-bit String Running Results Comparison

Initial Results: 40 bits

Sequences `ATGCTTGCGGGGGGGACAG` and `TGCCTGCACGCGCGCATCAG` are the test sample.

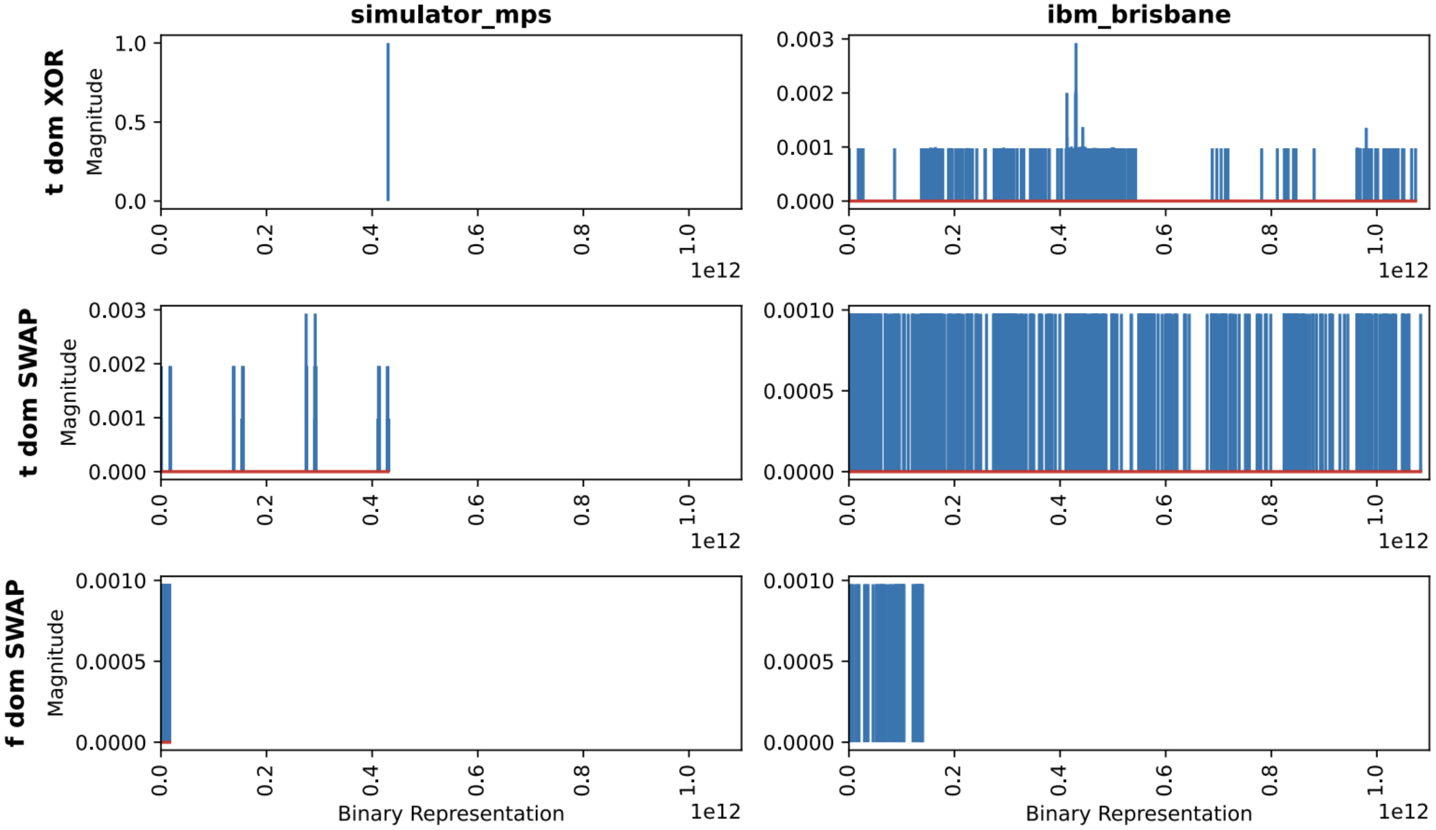


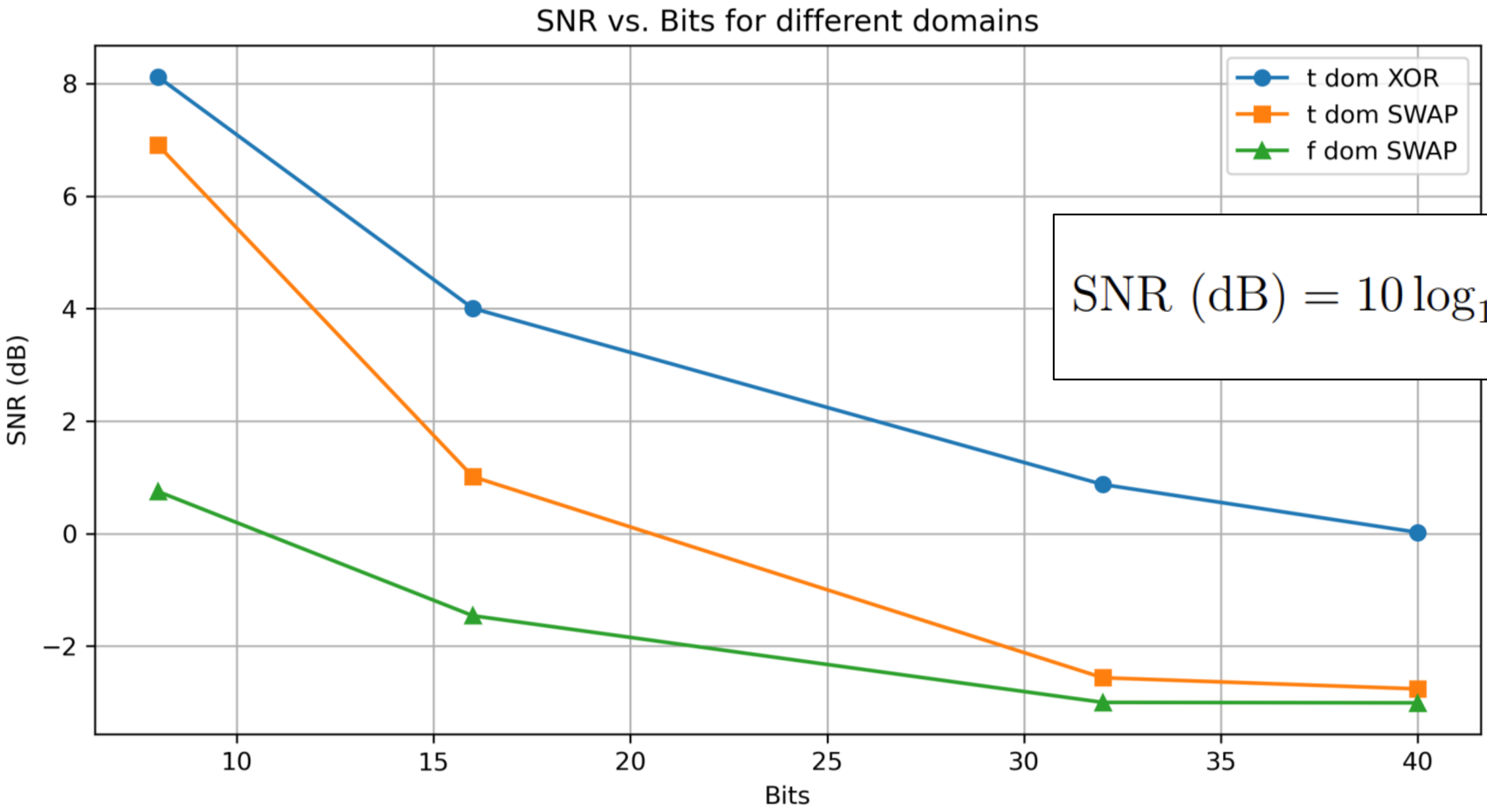
Figure 34: 40-bit String Running Results Comparison

Circuit Optimization

- Circuit & Transpilation Optimization:
  - Remove barrier separating each section entirely (except before measurement).
  - Increase optimization flag from 1 (default) to 3 (maximum).

- Sampler Primitive Improvement:
  - Increase sampling shot counts from 1,024 shots to 10,000 shots.

Noise Analysis



$$\mathrm{SNR\ (dB)} = 10\log_{10}\frac{\sum A_{\mathrm{signal}}^2}{\sum A_{\mathrm{noise}}^2}$$

Optimized Results: 8 bits

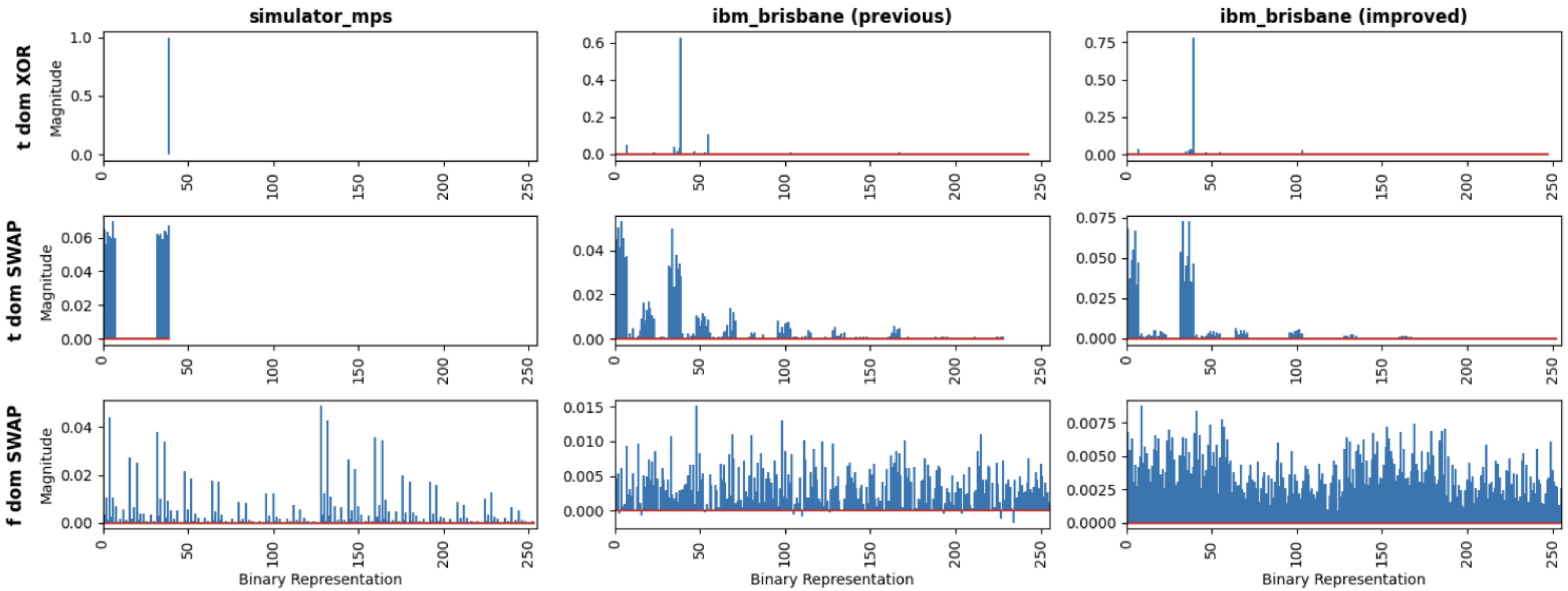


Figure 37: Quasiprobability Distribution Comparison (8 bits)

Optimized Results: 16 bits

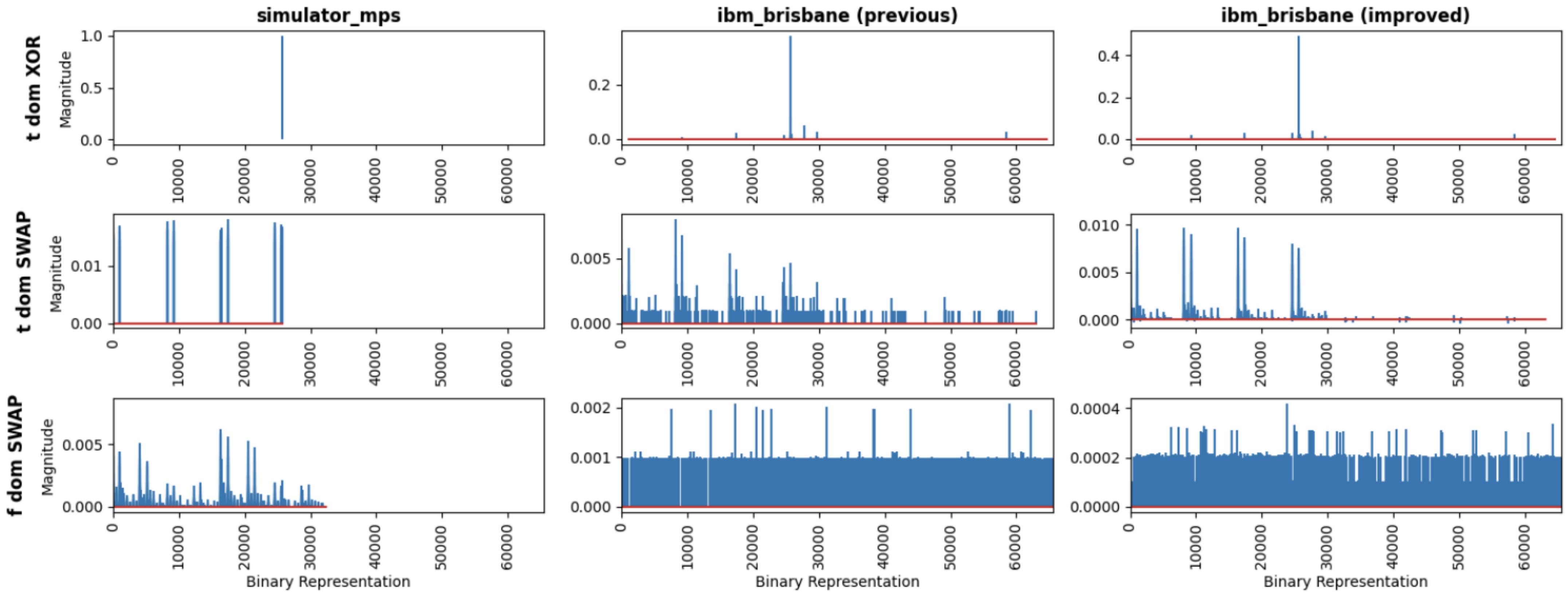


Figure 38: Quasiprobability Distribution Comparison (16 bits)

Optimized Results: 32 bits

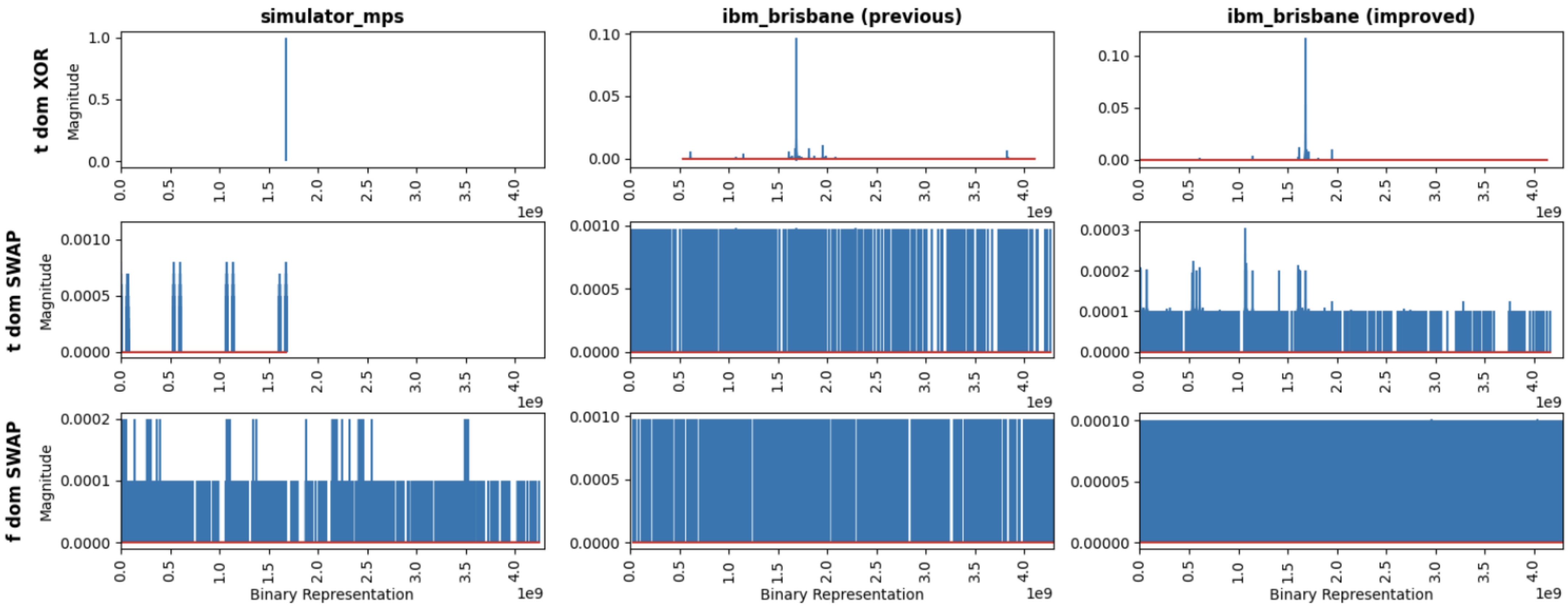


Figure 39: Quasiprobability Distribution Comparison (32 bits)

Optimized Results: 40 bits

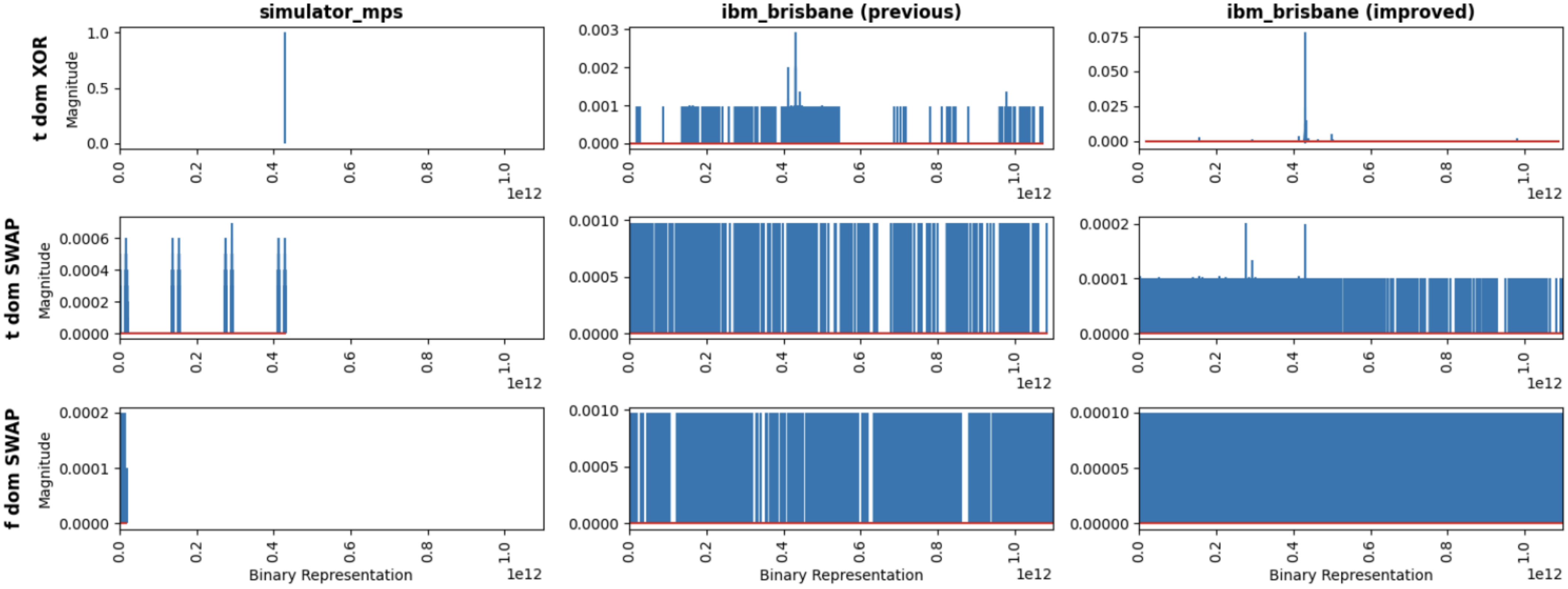


Figure 40: Quasiprobability Distribution Comparison (40 bits)

Summary

- Limited technology at this time.
  - Device noise, measurement error, etc.
  - Hope for future advancements

- Many potential alternatives in numerous research, this is only one of the approaches.

Further Research

- Mitigating Noise
- Interpreting QFT Result
- Alternative Problem Statements
  - Energy-Based Models
  - Graph-Theoretic Approaches
  - Quantum Machine Learning
- Quantum RNA Folding
- Motif Finding^{+}
  - Algorithms proposed in this project may be more suitable for Motif finding problem.

References

- See the full report for more details.
- Visit **vt.in.th/quantum** for further readings, files and this slide.

vt.in.th/quantum

- More on quantum computing: Qiskit and other reading.