28th Asia and South Pacific Design Automation Conference

Graph Partitioning Approach for Fast Quantum Circuit Simulation

Jaekyung Im, Seokhyeong Kang

CONTACT

Pohang University of Science and Technology
Department of Electrical Engineering
CAD and SoC Design Lab.
Tel. +82-54-279-2883
Web. http://csdl.postech.ac.kr

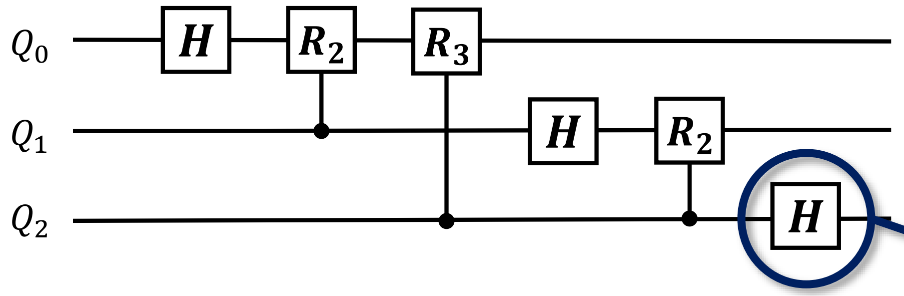
CONTENTS

1. INTRODUCTION
2. PROPOSED METHOD
3. EXPERIMENTAL RESULTS
4. CONCLUSION

INTRODUCTION

What is Quantum Circuit?

Example) **3-Qubit Quantum Fourier Transform**



"***Quantum Circuit*** is an efficient and powerful method of ***describing quantum computation*** procedures."

M. Nielsen and I. Chuang,
"***Quantum Computation and Quantum Information***."

- **Hadamard Gate**

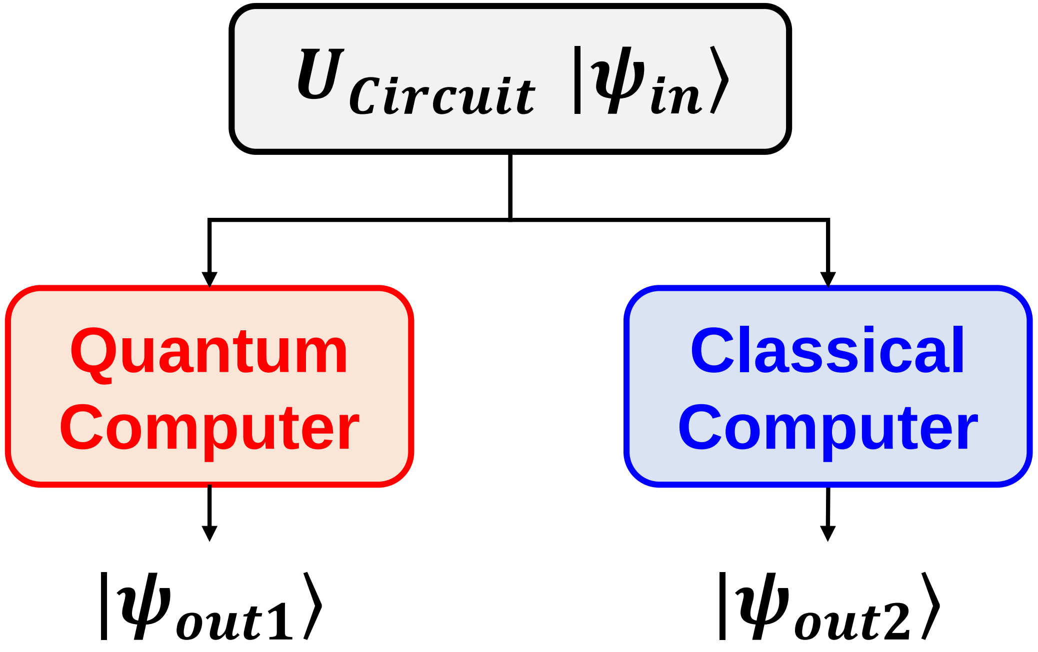
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- **Unitary Matrix**

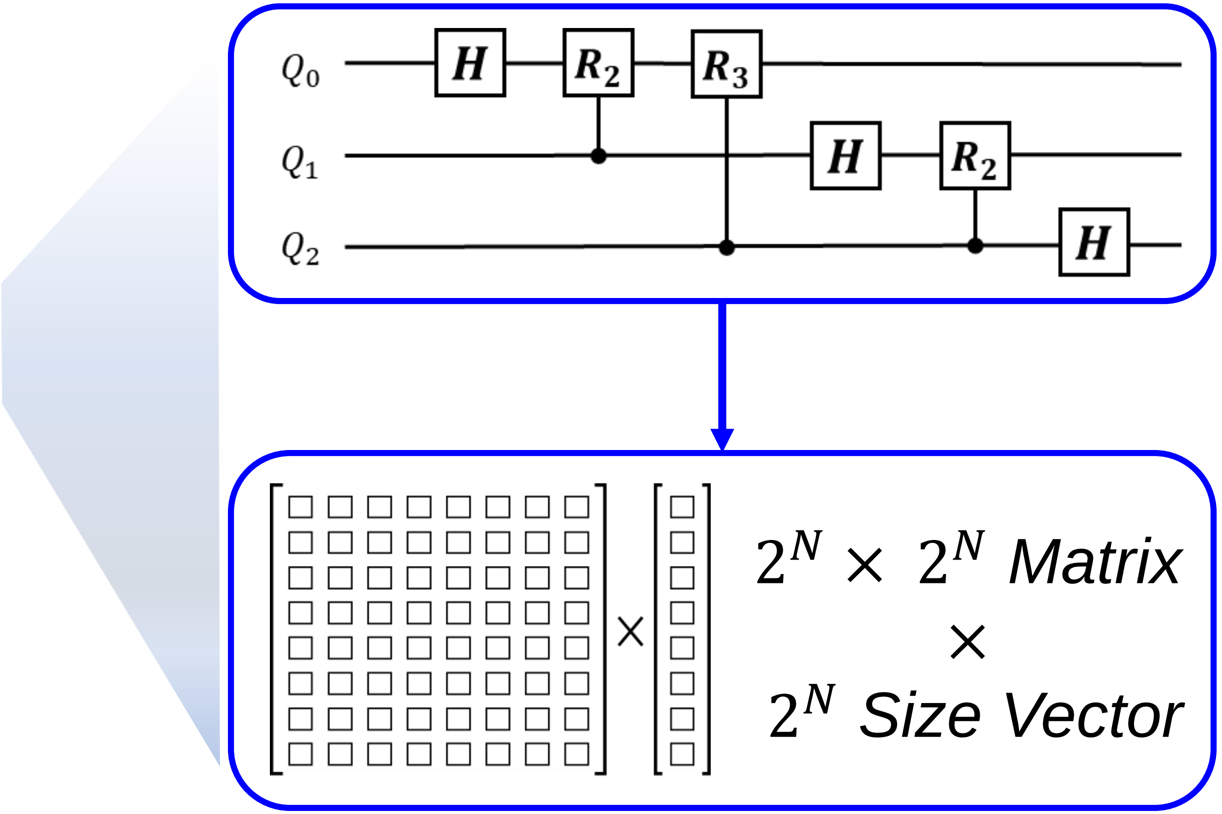
$$HH^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

- **Quantum circuit describes quantum computation procedures**.
- It consists of **quantum gates**, which can be represented by **unitary matrix**.

What is Quantum Circuit Simulation and Why it is difficult?

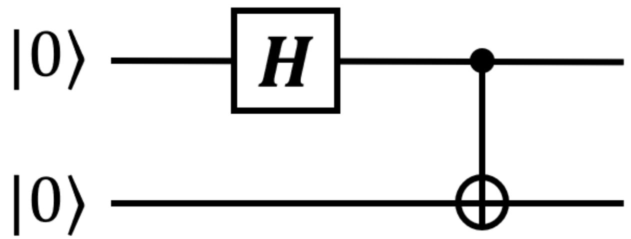


$$|\psi_{out1}\rangle == |\psi_{out2}\rangle\,?$$



- Simulation plays a key role in **verification** of quantum computing.
- But it is heavily **time-consuming**.

Two different Simulation Methods



Schrödinger-Style Simulation

$$|\psi_{out}\rangle = CNOT \times H \otimes I \times |00\rangle$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Feynman-Style Simulation

$$Path1: CNOT \cdot \frac{1}{\sqrt{2}}|00\rangle = \frac{1}{\sqrt{2}}|00\rangle$$

$$Path2: CNOT \cdot \frac{1}{\sqrt{2}}|10\rangle = \frac{1}{\sqrt{2}}|11\rangle$$

$$|\psi_{out}\rangle = Path1 + Path2 = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

- Schrödinger-Style: **Fast** but exponentially increasing **memory usage**.
- Feynman-Style: **Memory-efficient** but its **runtime depends on # of Paths.**

Hybrid Schrödinger-Feynman (HSF) Simulation

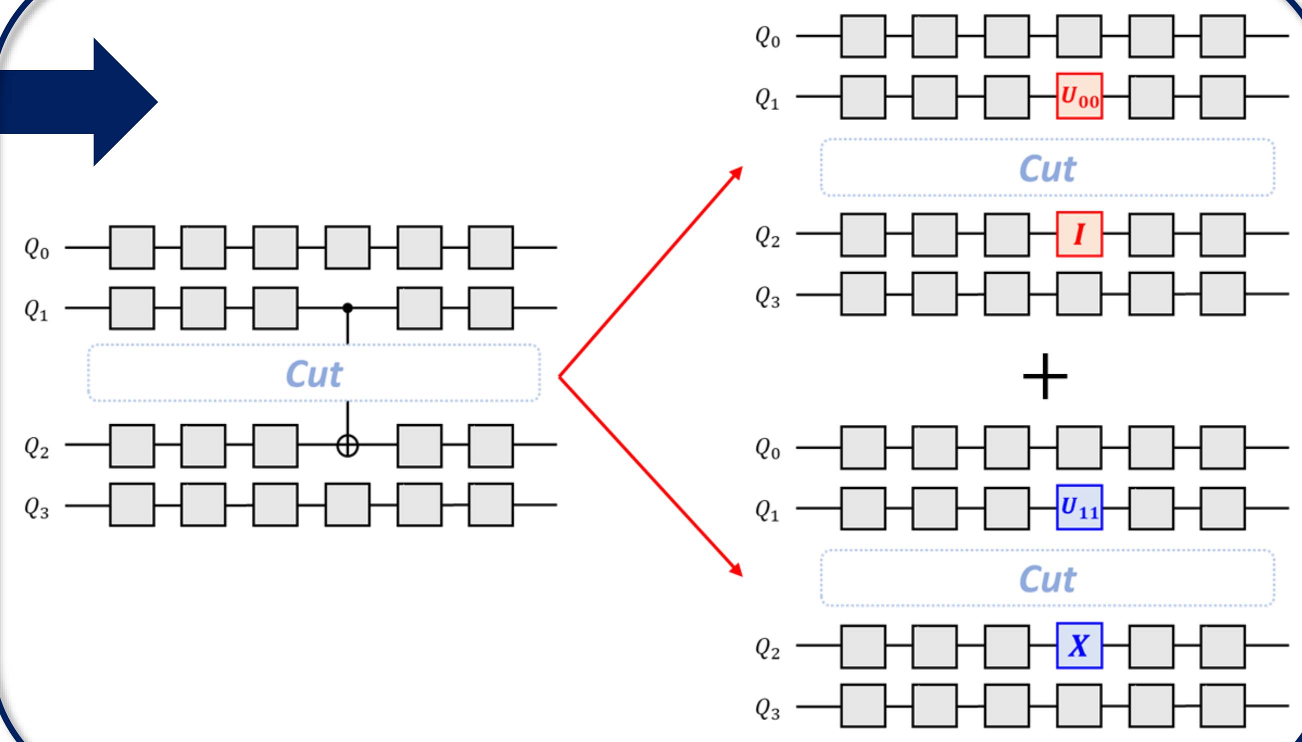
Gate Decomposition

$$\mathbf{G} = |0\rangle\langle 0| \otimes \mathbf{G}_{00} + |0\rangle\langle 1| \otimes \mathbf{G}_{01} + |1\rangle\langle 0| \otimes \mathbf{G}_{10} + |1\rangle\langle 1| \otimes \mathbf{G}_{11}$$

Example)

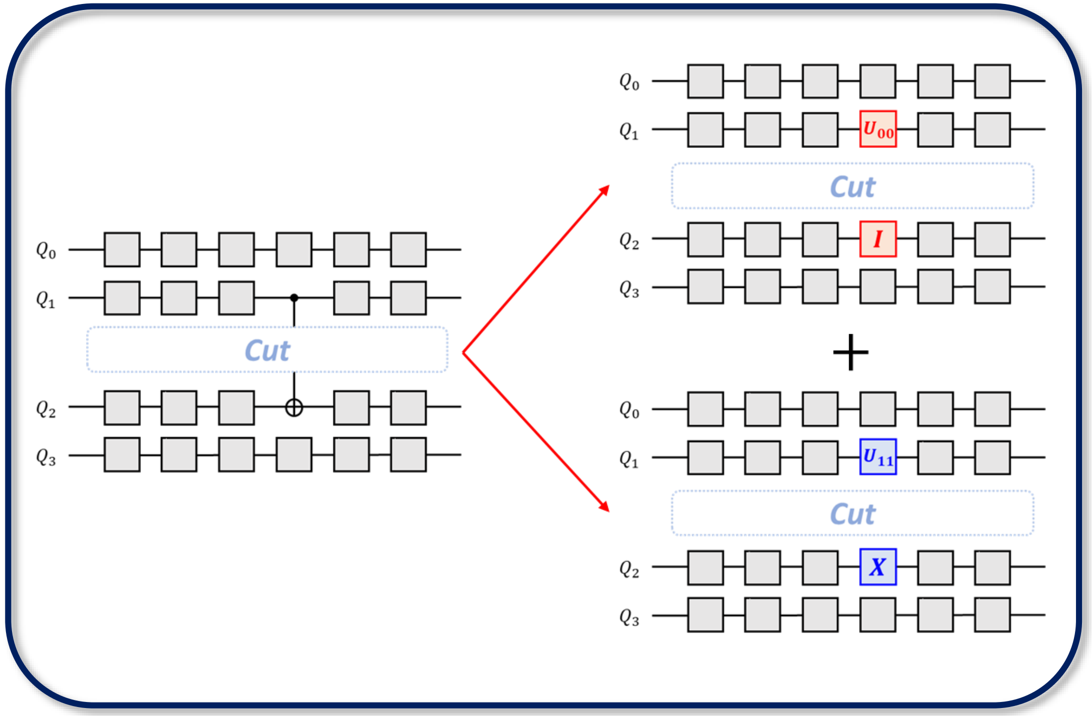
$$CNOT = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes I + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes X$$

$$\rightarrow U_{00} \qquad \rightarrow U_{11}$$

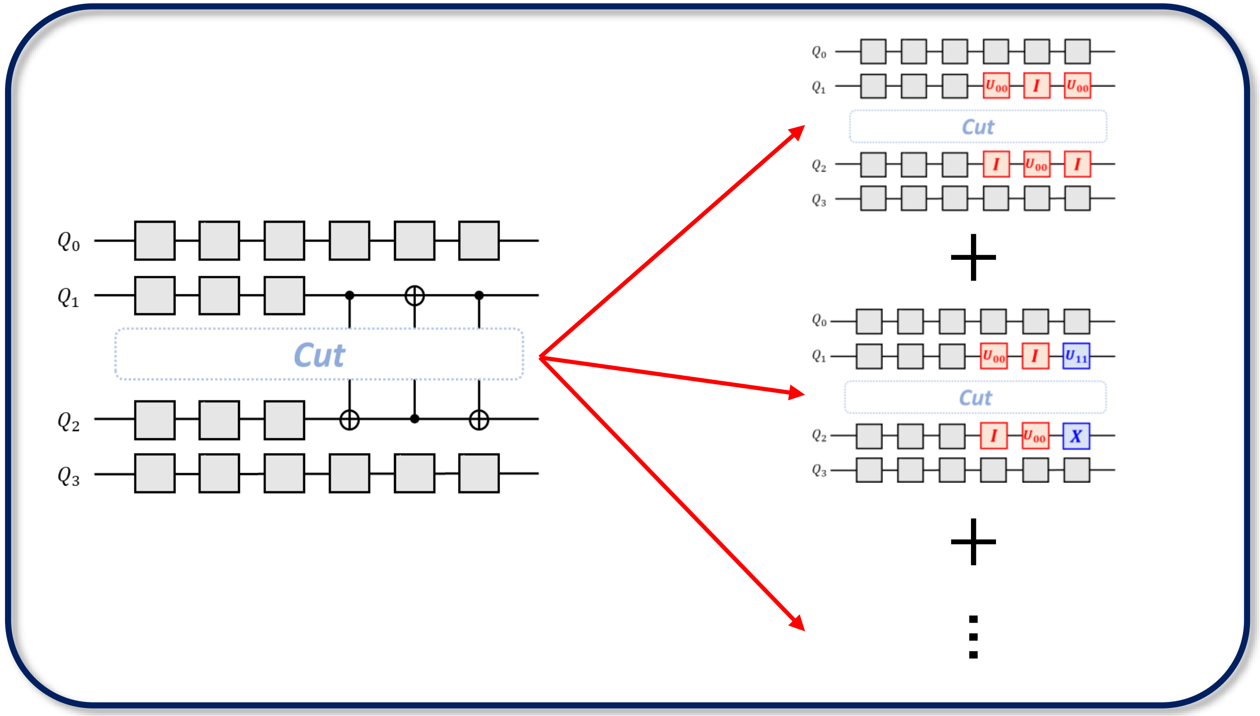


- **HSF** is a **mixture** of Schrodinger and Feynman Simulation.
- It can deal with circuits of smaller size.

Motivation



$\mathbf{1}$ Crossing Gates $\longrightarrow$ $\mathbf{2^1}$ Branches



$\boldsymbol{N}$ Crossing Gates $\longrightarrow$ $\mathbf{2^N}$ Branches

- HSF does not do well when there are too many crossing gates.
- **How can we reduce the the number of Crossing Gates?**

PROPOSED METHOD

Overall Flow

1. Graph Partitioning

- Build Weighted-Graph
- Fidducia-Mattheyses Algorithm

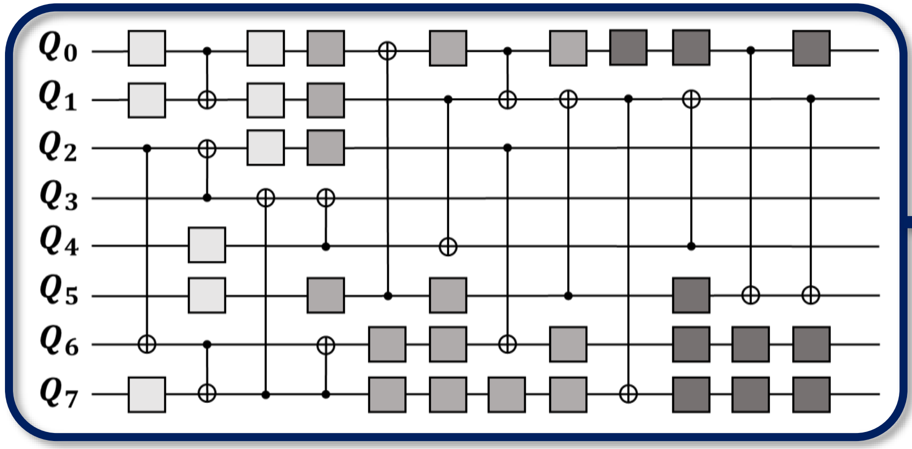
2. Layered Simulation

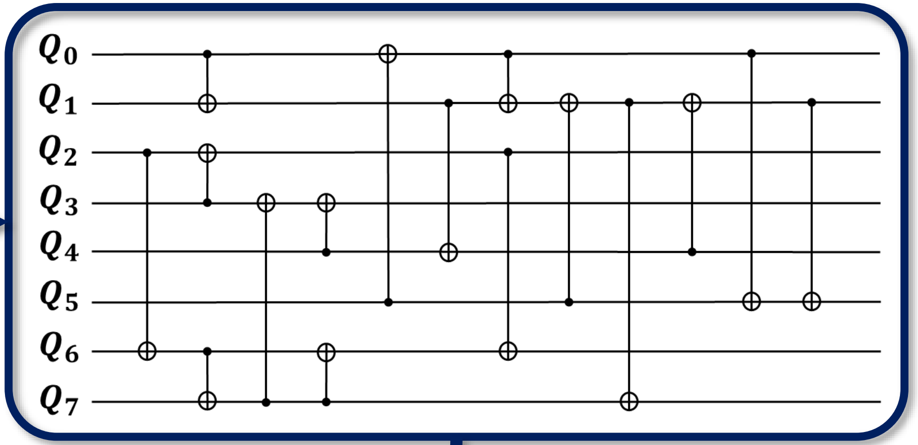
- Layer Assignment
- DFS Order Simulation

3. Qubit Re-Ordering

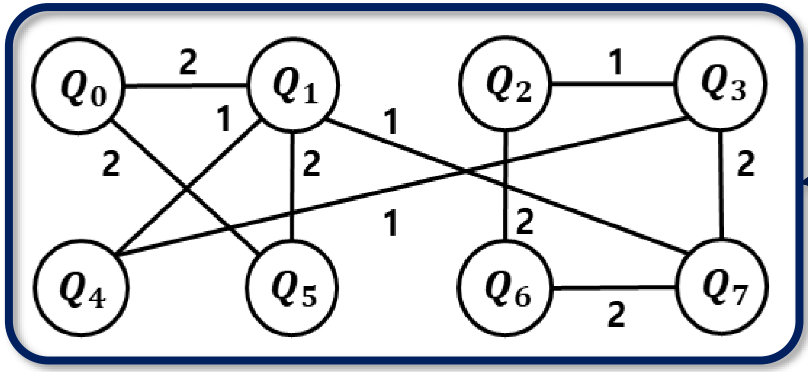
- Fast Qubit SWAP Procedure

1. Graph Partitioning - (1) Build Weighted-Graph



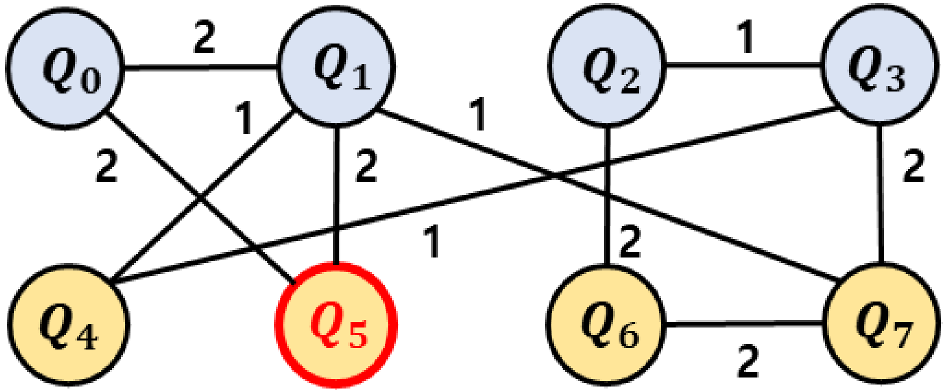


Weighted Graph



- **Ignore single-qubit gates**.
- **1)** Qubit ⟶ Vertex, **2)** # of two-qubit gates ⟶ Weight of edge.

1. Graph Partitoining – (2) FM Partitioning



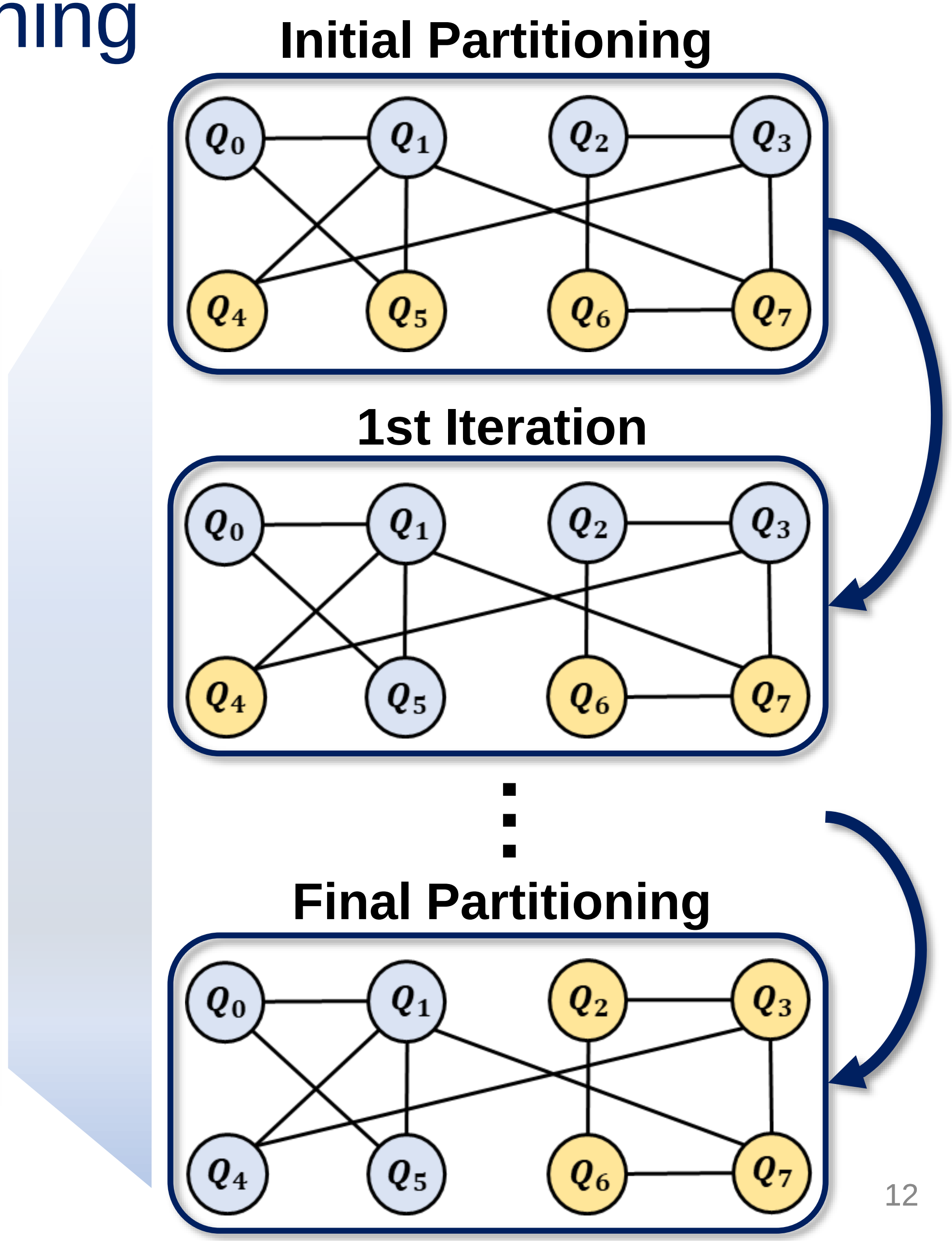
Gain of moving Q_5 to the opposite partition

$$= \sum(\textit{uncut edges}) - \sum(\textit{cut edges})$$

$$= 4 - 0 = 4$$

Qubit	*Gain*	*Status*
Q_0	0	*free*
Q_1	2	*free*
Q_2	0	*free*
Q_3	2	*free*
Q_4	2	*free*
Q_5	4	*free* → *fixed*
Q_6	-1	*free*
Q_7	1	*free*

Iteration	*Qubit*	*ΣGain*
1	Q_5	4
2	Q_3	6
3	Q_2	8
4	Q_4	**8**
5	Q_6	6
6	Q_0	2
7	Q_7	3
8	Q_1	0



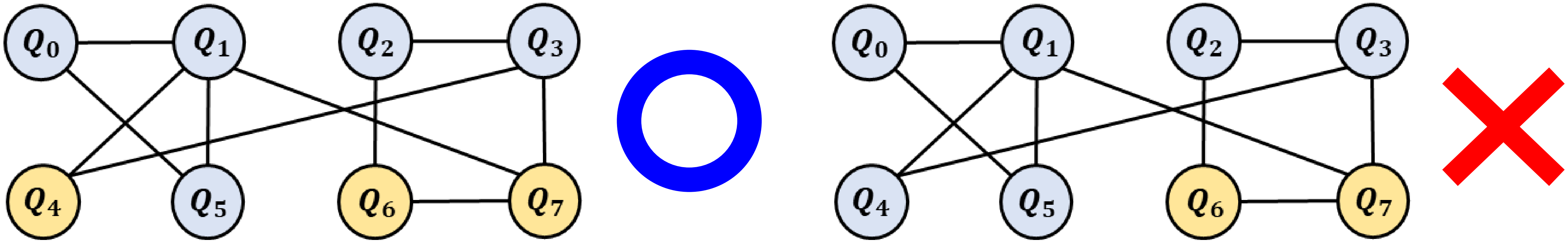
Why FM Algorithm?

- **Low Complexity**
  - Fast (Linear time-complexity)
  - Easy to implement

- **Balance Ratio**
  - Can control the partition balance

Example

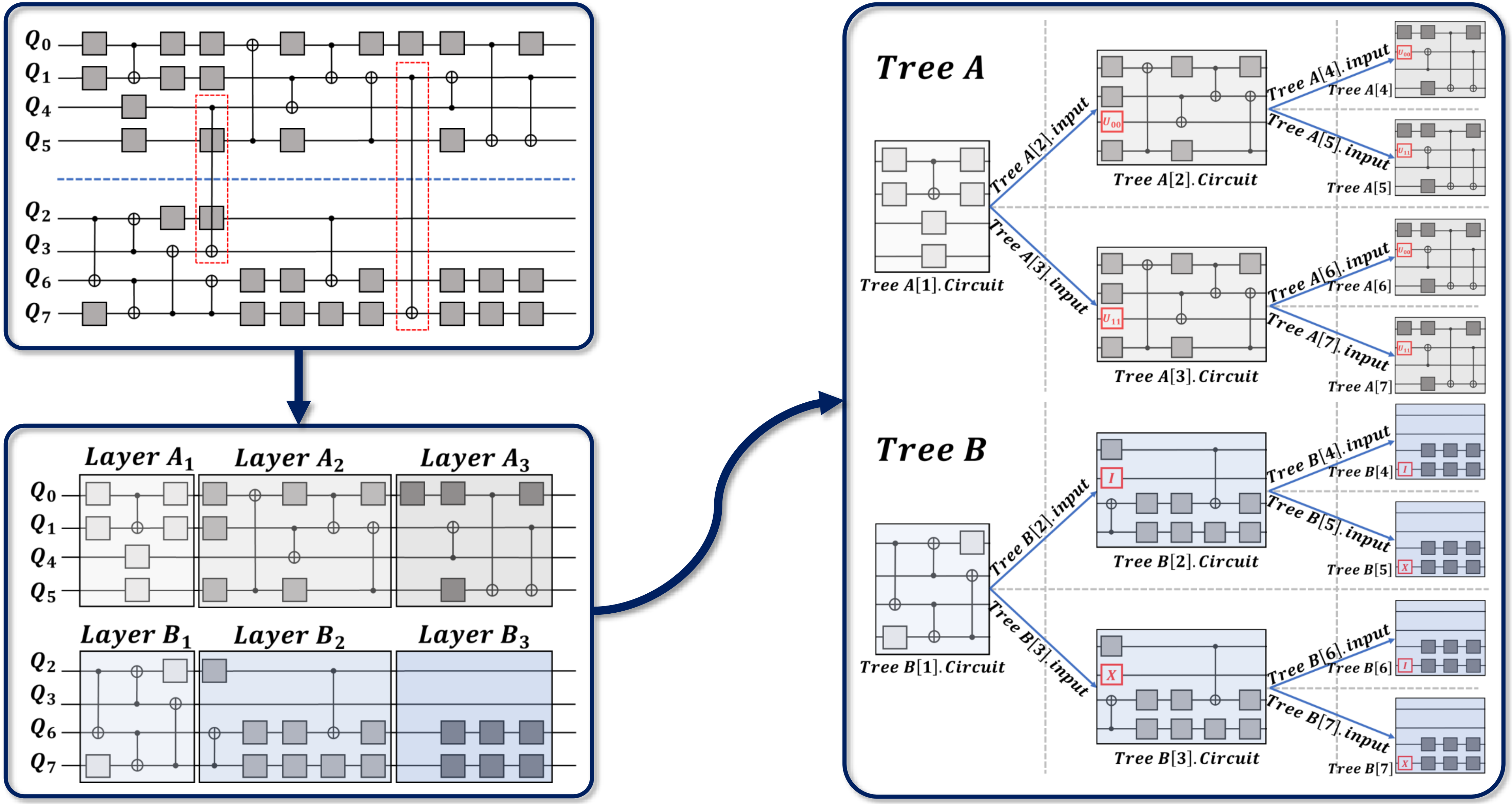
If balance ratio $\boldsymbol{r = 0.3}$



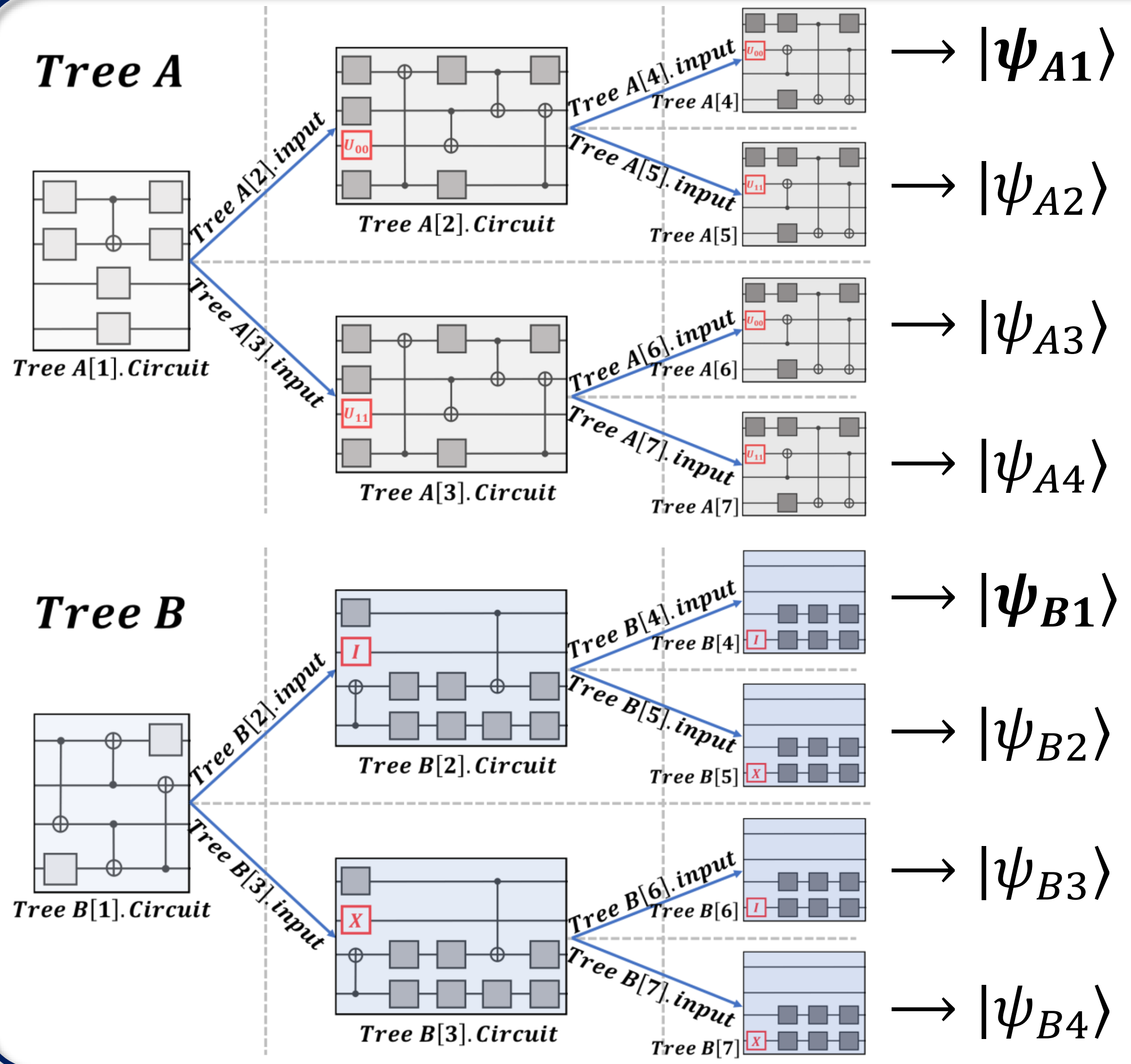
$$\boldsymbol{min\left(\frac{5}{8}, \frac{3}{8}\right) = 0.375 > 0.3}$$

$$\boldsymbol{min\left(\frac{6}{8}, \frac{2}{8}\right) = 0.25 < 0.3}$$

2. Layered Simulation - (1) Layer Assignment



2. Layered Simulation - (2) DFS Order Simulation



$$|\psi_1\rangle = |\psi_{A1}\rangle \otimes |\psi_{B1}\rangle$$

$$\begin{aligned}|\boldsymbol{\psi}_{\boldsymbol{Final}}\rangle &= |\psi_{A1}\rangle \otimes |\psi_{B1}\rangle \\ &+ |\psi_{A2}\rangle \otimes |\psi_{B2}\rangle \\ &+ |\psi_{A3}\rangle \otimes |\psi_{B3}\rangle \\ &+ |\psi_{A4}\rangle \otimes |\psi_{B4}\rangle \\ &= \sum |\psi_{Ai}\rangle \otimes |\psi_{Bi}\rangle \\ &= \sum |\boldsymbol{\psi}_{\boldsymbol{i}}\rangle\end{aligned}$$

3. Qubit Re-Ordering – Why do we need SWAP?

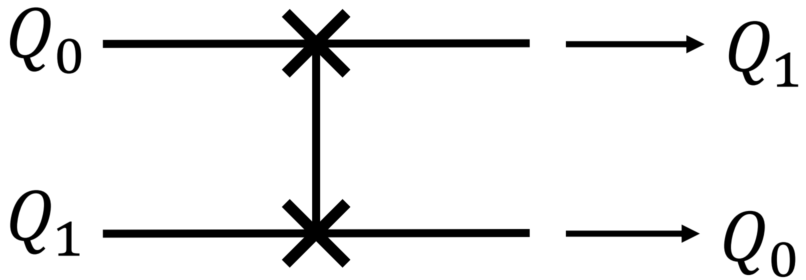
$$|\psi_{Final}\rangle = |Q_7 Q_6 Q_3 Q_2 Q_5 Q_4 Q_1 Q_0\rangle$$

$$\boldsymbol{SWAP(Q_3, Q_5)}$$
$$\boldsymbol{SWAP(Q_2, Q_4)}$$

$$|\psi_{Final}\rangle = |Q_7 Q_6 Q_5 Q_4 Q_3 Q_2 Q_1 Q_0\rangle$$

Approach #1. Applying SWAP Gate

$$\boldsymbol{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- We have to restore the initial qubit order.
- Our unsuccessful first approach was applying SWAP gates.

3. Qubit Re-Ordering – Fast Qubit SWAP

Approach #2. Fast SWAP with bit-masking

Algorithm 3: SWAP(i, j, Amp)

Input :Qubit index $i, j\,(i > j)$, Amplitude Vector $Amp[\,]$

```
Set TargetMask1 ← 000...001 << i
Set TargetMask2 ← 000...001 << j
Set MaskR ← (111...111 >> N − j)
Set MaskL ← (111...111 << i + 1)
Set MaskM ← 111...111 ⊕ (MaskR ∨ MaskL ∨ TargetMask1 ∨ TargetMask2)
Set Idx ← 0
while Idx < 2^(N−2) do
    Set RealIdx ←
      (Idx ∧ MaskR) ∨ ((Idx << 1) ∧ MaskM) ∨ ((Idx << 2) ∧ MaskL)
    Set TargetIdx1 ← RealIdx ∨ TargetMask1
    Set TargetIdx2 ← RealIdx ∨ TargetMask2
    Exchange(Amp[TargetIdx1], Amp[TargetIdx2])
    Idx ← Idx + 1
end
```

Example) **SWAP**(3, 5, $|\psi\rangle$)

$$|Q_7Q_6\mathbf{Q_3}Q_2\mathbf{Q_5}Q_4Q_1Q_0\rangle = \boldsymbol{\alpha_0}|00000000\rangle$$

$$\ldots$$

$$+\ \boldsymbol{\alpha_8}|00001000\rangle$$
$$+\ \boldsymbol{\alpha_9}|00001001\rangle$$
$$\ldots$$
$$+\ \boldsymbol{\alpha_{32}}|00100000\rangle$$
$$+\ \boldsymbol{\alpha_{33}}|00100001\rangle$$
$$\ldots$$
$$+\ \boldsymbol{\alpha_{223}}|11011111\rangle$$
$$\ldots$$
$$+\ \boldsymbol{\alpha_{247}}|11110111\rangle$$

$\boldsymbol{Idx} = 0$: Exchange Amplitudes ($\alpha_8 \leftrightarrow \alpha_{32}$)

$\boldsymbol{Idx} = 1$: Exchange Amplitudes ($\alpha_9 \leftrightarrow \alpha_{33}$)

$\boldsymbol{Idx} = 2^6 - 1$: Exchange Amplitudes ($\alpha_{223} \leftrightarrow \alpha_{247}$)

Experimental Results

Experimental Setup

- **Benchmark**
  - Google Supremacy Circuits **[1]**
- **Implementation**
  - Base: Decision-Diagram Simulator (DDSim) **[2]**
  - C++, OpenMP
- **Comparison**
  - Hybrid Schrodinger-Feynman Decision-Diagram Simulator (HSF-DDSim) **[3]**

[1] Google Random Circuit Samplings, https://github.com/sboixo/GRCS

[2] A. Zulehner and R. Wille, "*Advanced Simulation of Quantum Computations*", IEEE Transactions on CAD, 2019 https://github.com/cda-tum/ddsim

[3] L. Burgholzer, H. Bauer and R. Wille, "Hybrid Schrodigner-Feynman Simulation of Quantum Circuits with Decision Diagrams", International Conference on Quantum Computing and Engineering, 2021

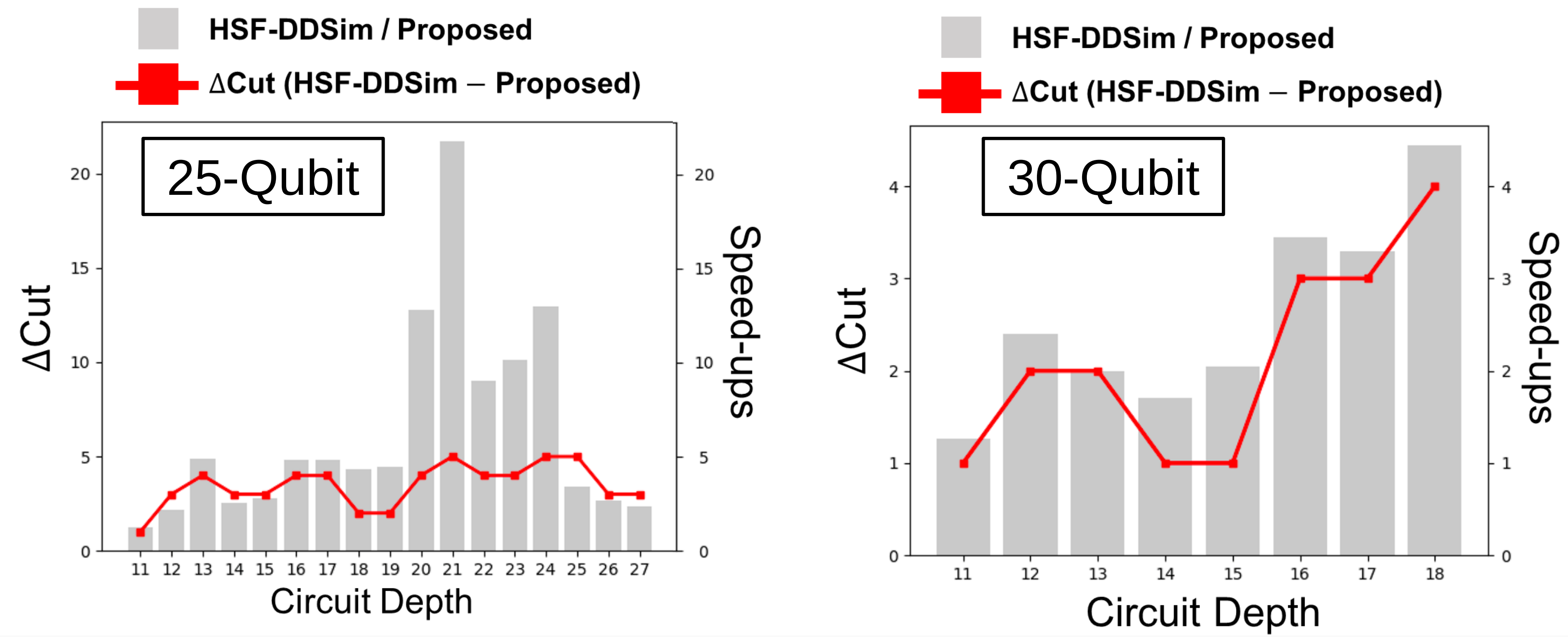
Experimental Results (1) Effectiveness of fast SWAP

Benchmark	Baseline - DDSim	Graph Partitioning + Applying SWAP gate	Graph Partitioning + Optimized SWAP
Name	Runtime [s]	Runtime [s]	Runtime [s]
inst_4x5_10_0	244.9	2.3	1.4
inst_4x5_12_0	803.4	15.7	0.7
inst_4x5_16_0	1374.5	164.6	7.4
inst_4x5_20_0	2748.2	12680.0	57.2
inst_5x5_14_0	>24 h	816.7	87.5

* $inst\_A\text{x}B\_C$ means a circuit of $A * B$ qubits and C depths

- HSF with graph partitioning can bring a huge speed-up.
- The overhead of applying swap gates can make it disappear.
- Our fast SWAP procedure can be a solution for the problem.

Experimental Results (2) Comparison with HSF-DDSim



- FM partitioning makes fewer cuts than the half slicing.
- Our approach achieved runtime improvement up to 20X times.

CONCLUSION

Conclusion

Summary

- Traditional HSF simulation uses simple half slicing.
- The number of crossing gates increases the simulation time exponentially.
- Partitioning approach can make fewer crossing gates than half slicing.
- This leads to significant speed-up of simulation.

Future Work

- How can we consider noise in the partitioning approach?
- How about other benchmarks?

THANK YOU


CSDL
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교