QMDDを用いた量子回路シミュレータの並列化手法

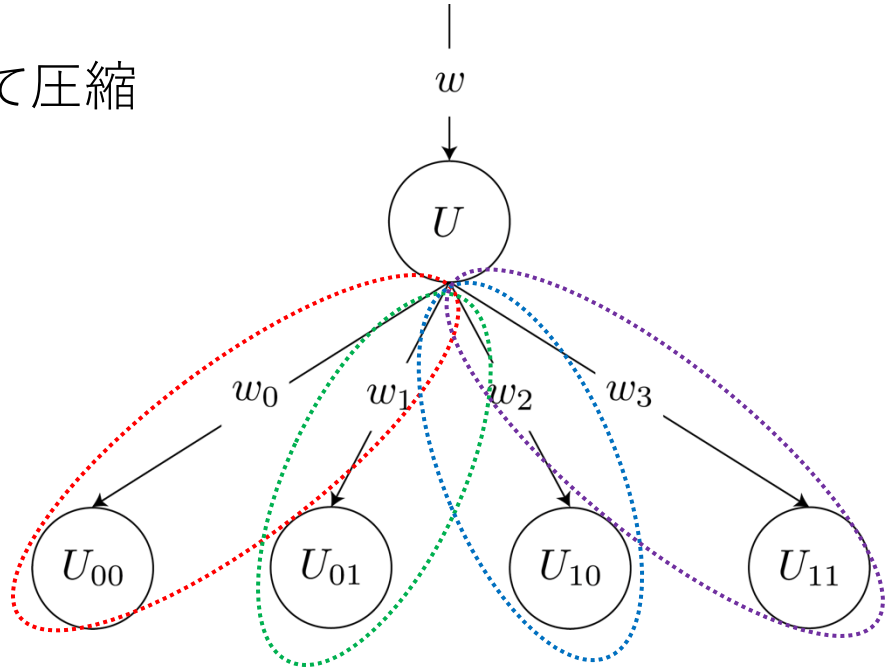
立命館大学 情報理工学部
次世代コンピューティング研究室
三石 海人

QMDDを用いた表現

- QMDD : Quantum Multiple-valued Decision Diagrams
  - 量子回路を表現・操作するためのデータ構造
  - 行列表現よりも必要メモリを削減
    - 零行列や共通する部分行列を利用して圧縮

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$$

行列表現

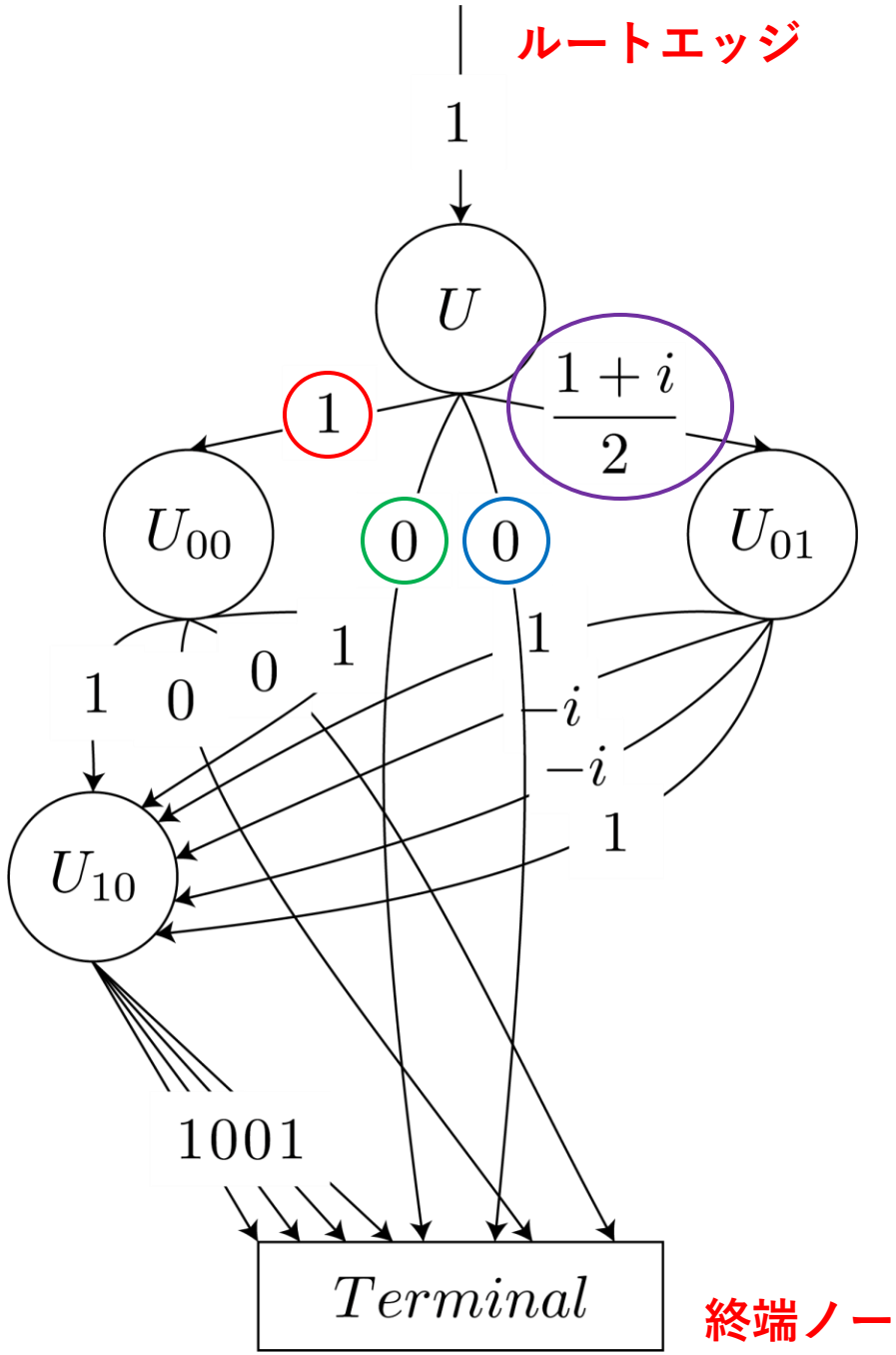


QMDD

QMDDの構造①

- 例) CVゲート

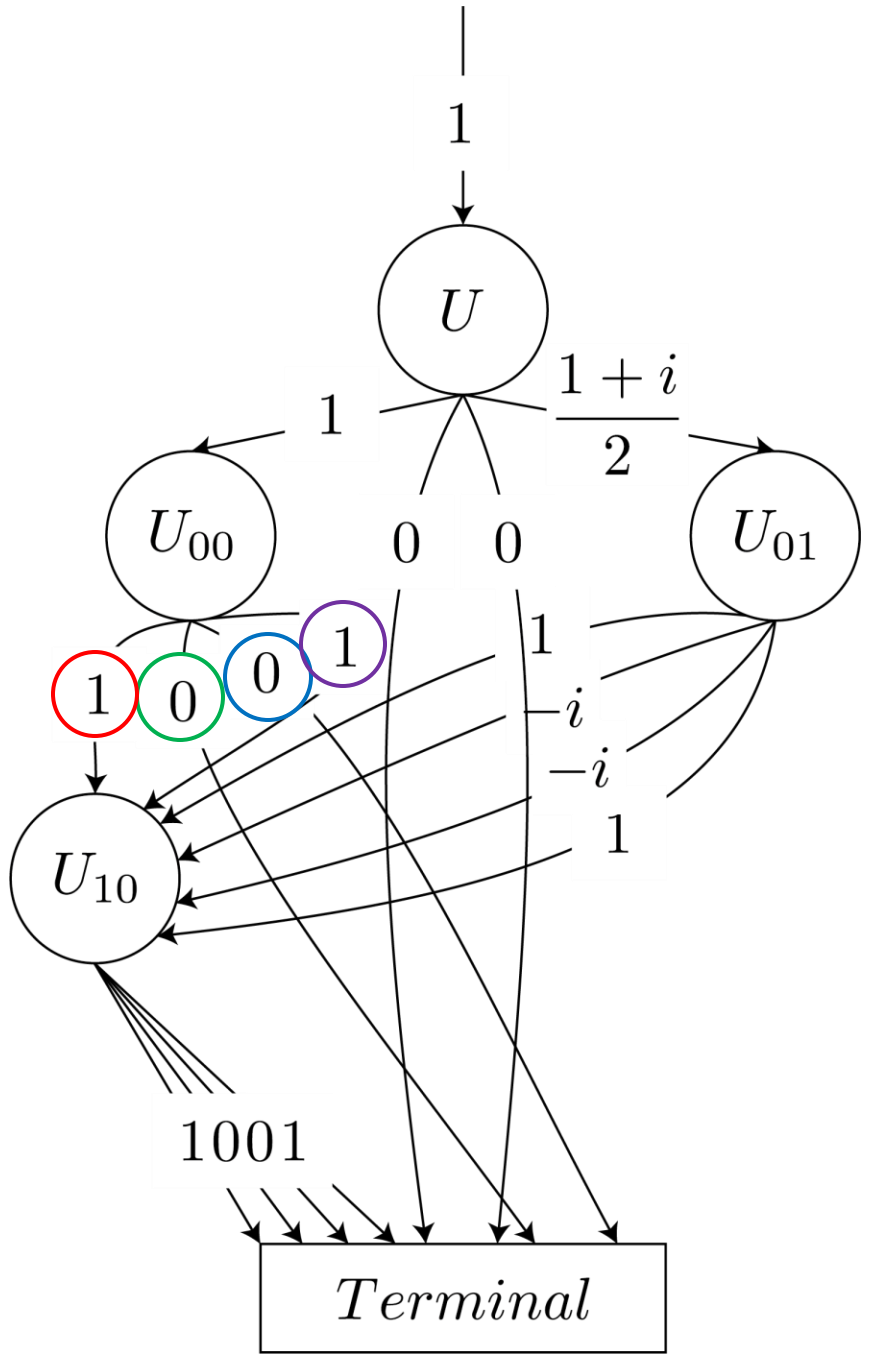
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} \\ 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} \end{bmatrix}$$



QMDDの構造②

- 例) CVゲート

$$
\left[\begin{array}{cccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} \\
0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2}
\end{array}\right]
$$



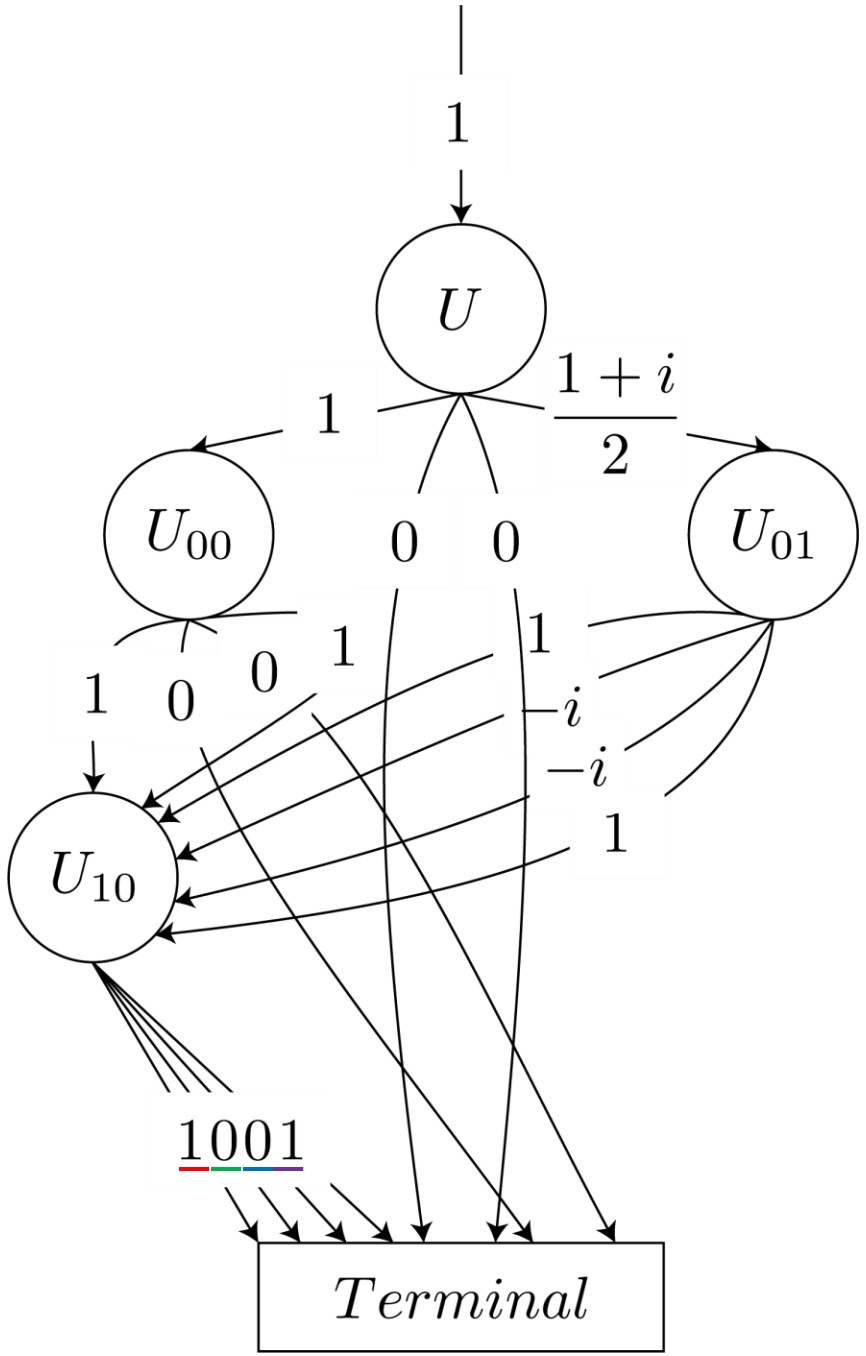
QMDDの構造③

- 例) CVゲート

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} \\ 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} \end{bmatrix}$$

$$CV_{0,0} = 1 * 1 * 1 * 1 = 1$$

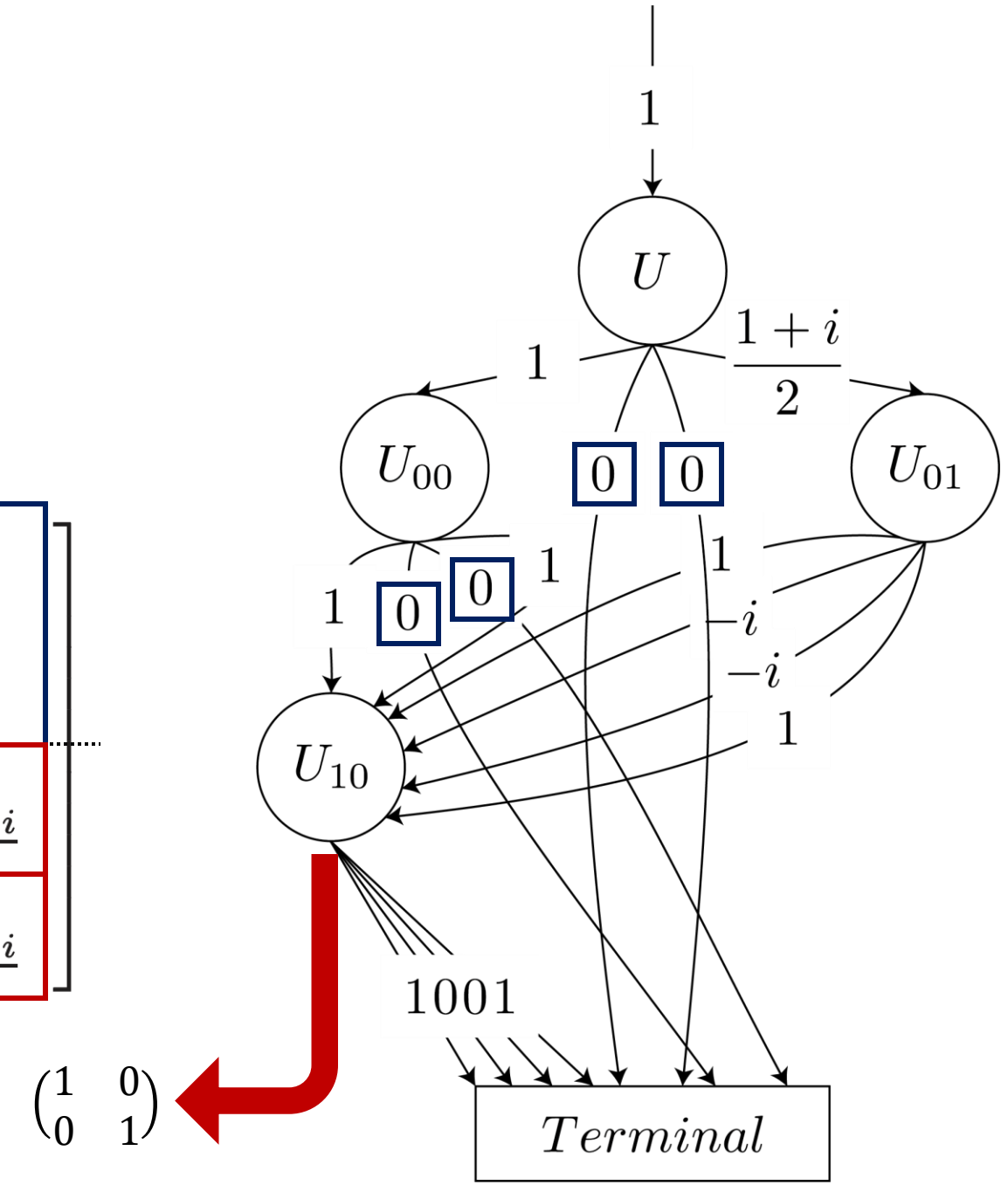
$$CV_{7,7} = 1 * \frac{1+i}{2} * 1 * 1 = \frac{1+i}{2}$$



QMDDの構造④

- 例) CVゲート

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & 0 & \frac{1-i}{2} \\
0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & 0 & \frac{1+i}{2}
\end{bmatrix}$$



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

処理効率を向上させるための仕組み

ユニークテーブル

- QMDDのノードを保存する
- 保存したノードの再利用

→メモリ効率向上

演算キャッシュ

- 加算・乗算・テンソル積の演算結果をキャッシュ
- 過去の演算結果を再利用

→演算の処理速度向上

提案手法

1. QMDDの演算アルゴリズムの非同期並列化

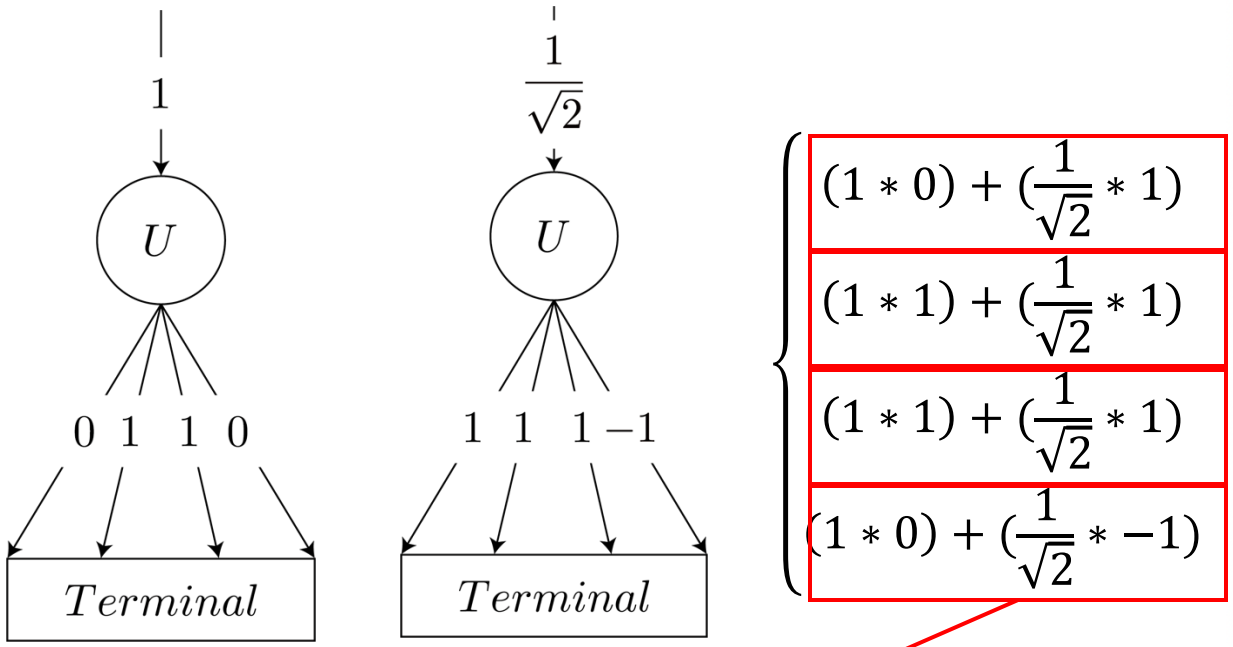
2. 排他制御における待機時間の回避

3. 対角行列に相当するQMDDの演算効率化
   - 付録に掲載

QMDD演算の並列化

例: X + H

$$= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



$$(1 * 0) + (\frac{1}{\sqrt{2}} * 1)$$
$$(1 * 1) + (\frac{1}{\sqrt{2}} * 1)$$
$$(1 * 1) + (\frac{1}{\sqrt{2}} * 1)$$
$$(1 * 0) + (\frac{1}{\sqrt{2}} * -1)$$

各エッジの処理をスレッドに分割

Algorithm 1 QMDD の加算処理

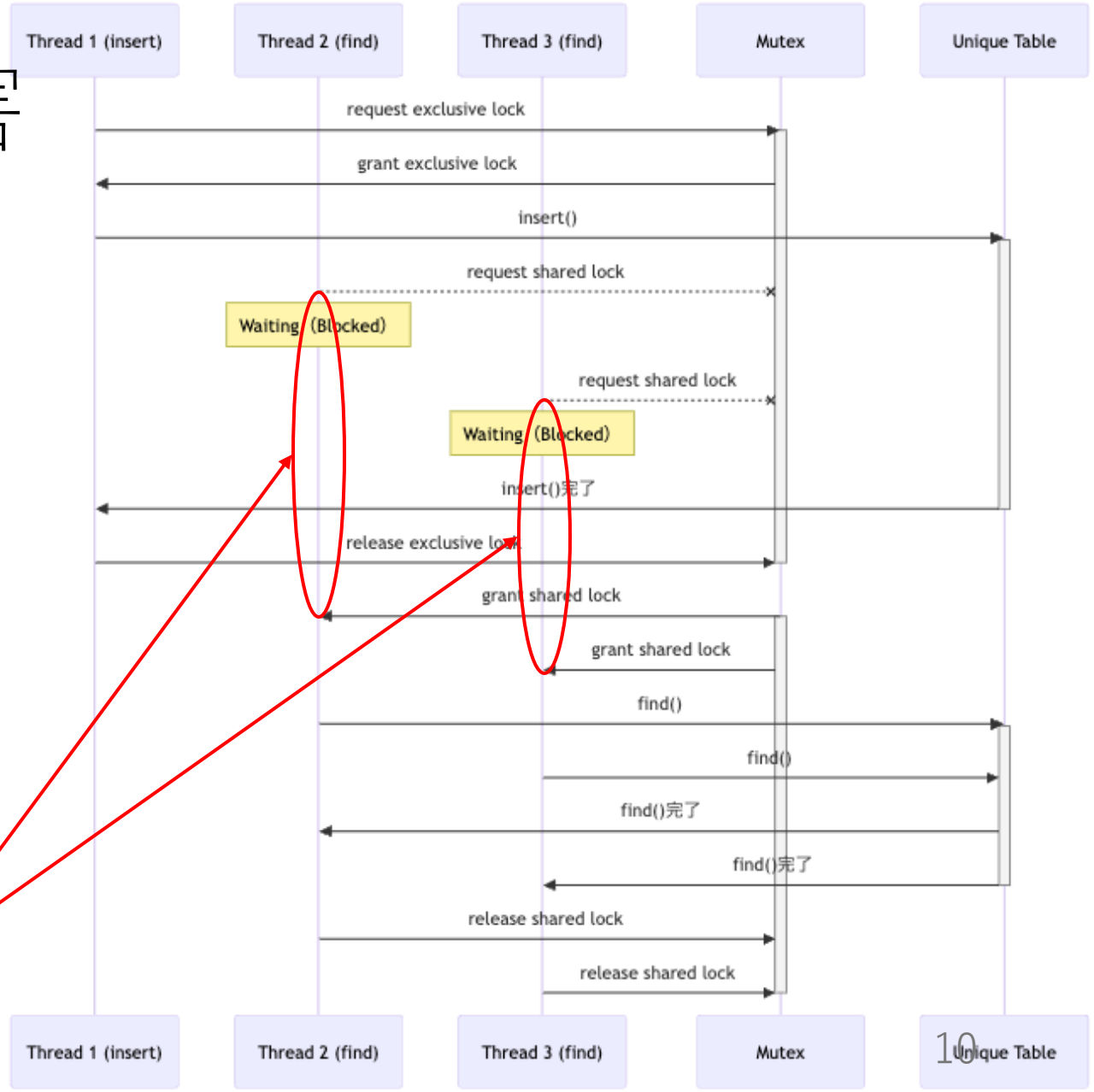
Input: $e0, e1$：QMDD のルートエッジ
Output: QMDD の和 $result$

```
1: if cache hit then
2:     result ← cache. find(key(e0, ADD, e1))
3: end if
4:
5: if T(e1) then
6:     swap(e0, e1)
7: end if
8:
9: if T(e0) then
10:    if w(e0) == 0 then
11:        result ← e1
12:    else
13:        result ← edge(w(e0) + w(e1), v(e1))
14:    end if
15: end if
16:
17: for i = 0 to 2^2 − 1 do
18:    p = edge(w(e0) * w(E_i(e0)), v(E_i(e0)))
19:    q = edge(w(e1) * w(E_i(e1)), v(E_i(e1)))
20:    z[i] = add(p, q)
21: end for
22: result ← edge(node(z))
23: cache. insert(key(e0, ADD, e1), result)
24:
25: result
```

子ノードのエッジを再帰的に処理

並列処理による弊害

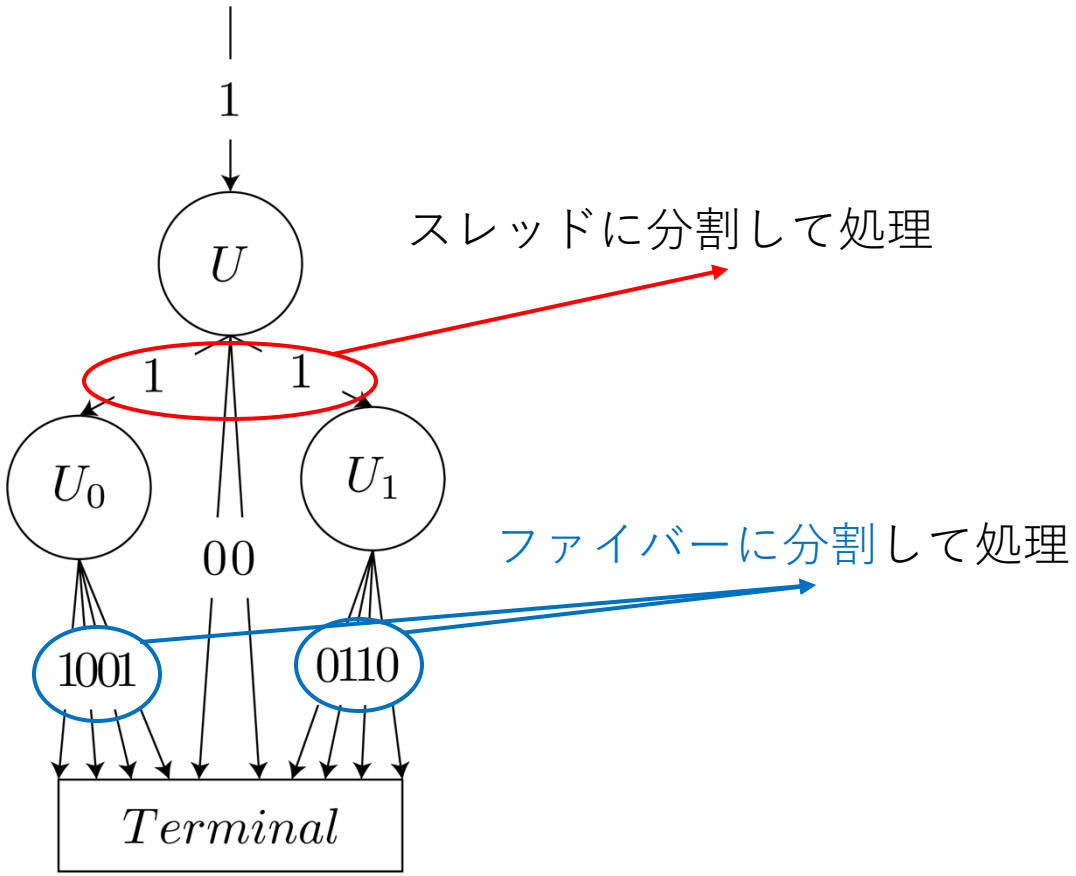
- 排他的なアクセス
  - $read$→共有ロック
    - 他スレッドの$write$不可
  - $write$→占有ロック
    - 他スレッドの$read$, $write$不可



排他制御による待機時間

待機時間の回避

- スレッドのタスクを複数のファイバーに分割



Algorithm 1 QMDD の加算処理

Input: $e0, e1$：QMDD のルートエッジ
Output: QMDD の和 $result$

```
1: if cache hit then
2:     result ← cache. find(key(e0, ADD, e1))
3: end if
4:
5: if T(e1) then
6:     swap(e0, e1)
7: end if
8:
9: if T(e0) then
10:    if w(e0) == 0 then
11:        result ← e1
12:    else
13:        result ← edge(w(e0) + w(e1), v(e1))
14:    end if
15: end if
16:
17: for i = 0 to 2^2 − 1 do
18:    p = edge(w(e0) ∗ w(E_i(e0)), v(E_i(e0)))
19:    q = edge(w(e1) ∗ w(E_i(e1)), v(E_i(e1)))
20:    z[i] = add(p, q)
21: end for
22: result ← edge(node(z))
23: cache. insert(key(e0, ADD, e1), result)
24:
25: return result
```

子ノードのエッジを再帰的に処理

共有資源へのアクセス

ファイバーでの改善

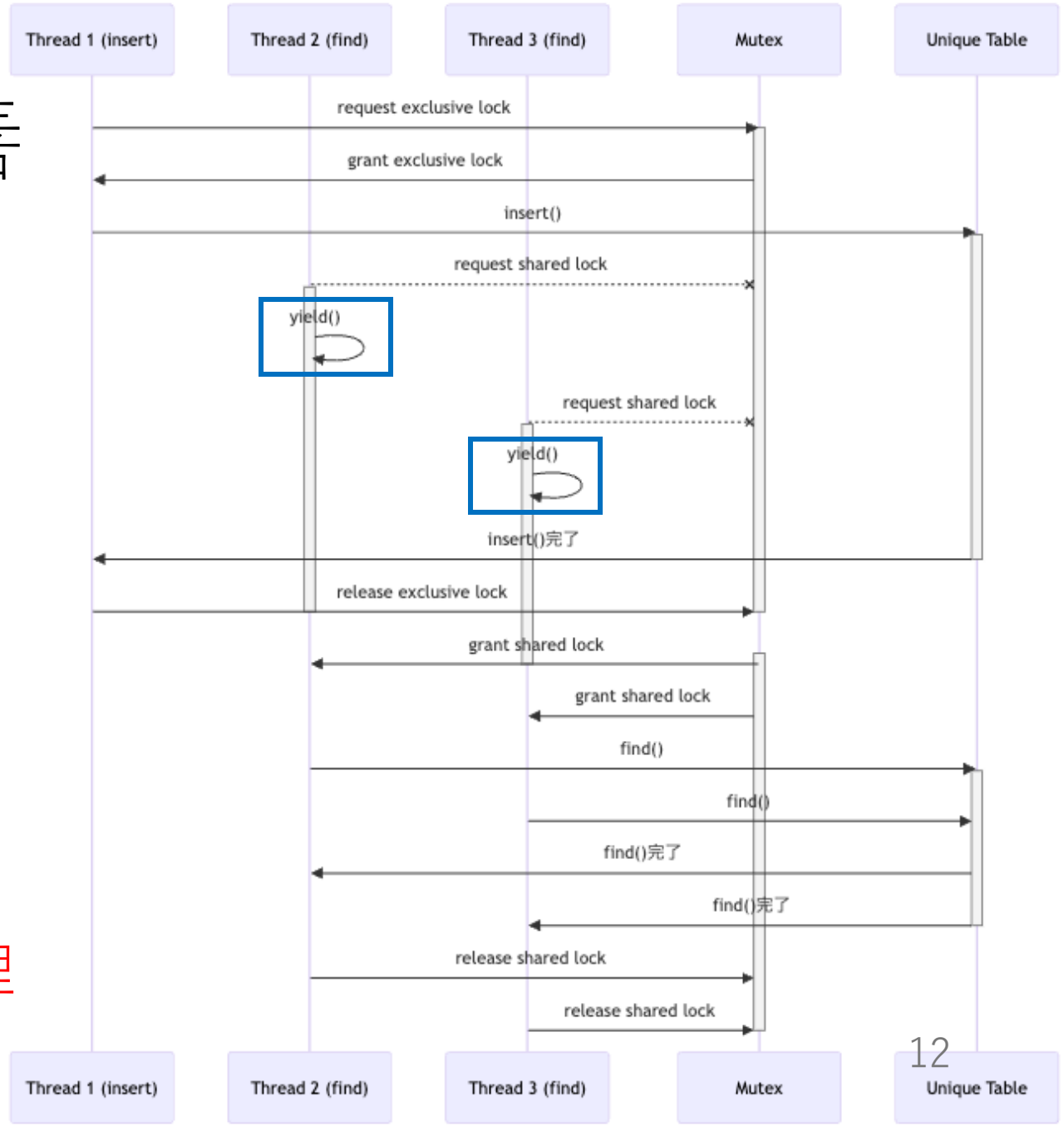
ロック待ち発生

↓

ファイバーの切り替え

↓

非クリティカルセクションを処理



比較実験

- ランダムな回転角とターゲットビットを持つ量子回転ゲート（Rx, Ry, Rz）200 個をシミュレーション
- 量子ビット数1~20
- 既存手法、マルチスレッドによる並列処理、提案手法で比較

環境項目	パラメータ値
OS	macOS15.2
チップ	Apple M3
メモリ	8GB
コア	8
コンパイラ	Apple Clang 16.0.0
C++バージョン	C++20
Boost バージョン	1.87.0

実験結果

それぞれ10回ずつ処理を行った際の平均処理時間(ms)と変化率(%)

量子ビット	既存手法	スレッドのみを用いた並列処理		提案手法	
	平均処理時間	平均処理時間	変化率	平均処理時間	変化率
2	**6.23**	13.35	114.36	13.02	109.16
4	**19.80**	32.27	63.01	32.38	63.56
6	**60.25**	87.41	45.08	72.59	20.48
8	220.39	263.37	19.50	**199.03**	-9.69
10	844.45	896.73	6.19	**636.19**	-24.66
12	3400.67	3231.48	-4.98	**2546.35**	-25.12
14	14001.26	12412.10	-11.35	**10453.05**	-25.34
16	62582.81	51552.37	-17.63	**43996.75**	-29.70
18	292819.60	217616.90	-25.68	**167629.00**	-42.75
20	1264967.70	873290.70	-30.96	**636008.86**	-49.72

考察

- 規模の小さい量子回路の処理は逐次処理が高速
  - スイッチングなどのコスト > 並列化による恩恵
- ファイバーはクリティカルセクションに対して有効
- 量子回路が大きくなるにつれて並列化による恩恵が大きくなる

まとめと今後の課題

まとめ

- QMDDの演算アルゴリズムの並列化手法の提案
  - マルチスレッド処理による非同期並列化
  - マルチファイバー処理による待機時間の回避
- 提案手法は一定以上の大きさの量子回路において有効
  - ベンチマークテストでは最大49%の処理時間を短縮

課題

- 回路全体の並列化
  - 対角行列や演算キャッシュを考慮した優先度設定
- 更なるメモリ効率の向上
  - ユニークテーブルに保存するQMDDのノードの寿命の考慮

付録

付録: 量子状態の行列表現

- 量子状態 → 一次元ベクトル

例）$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

- 複数量子ビットの量子状態はテンソル積で導出

例）$|00\rangle = |0\rangle|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

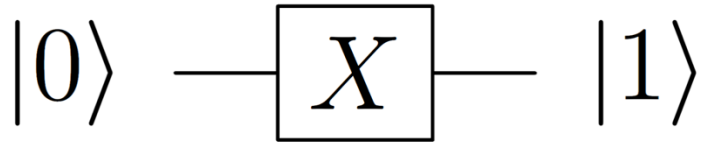
付録: 量子ゲートの行列表現

- 量子ゲート → 二次元のユニタリ行列

例）$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

- 量子ゲートの作用は行列積で導出

例）$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$



付録: 主なQMDD表現① 量子状態

行列表現

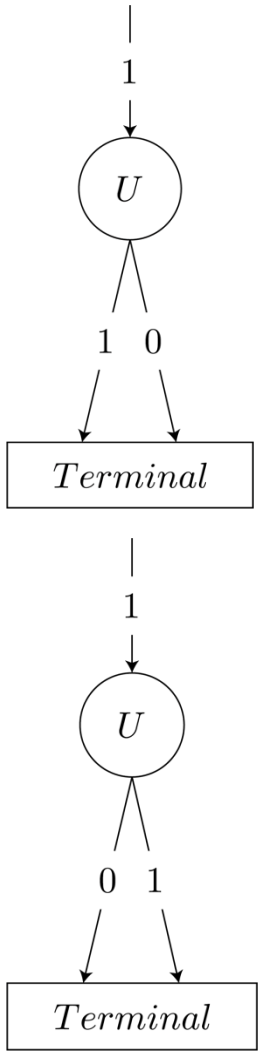
- 例） $|0\rangle$

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

- 例） $|1\rangle$

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

QMDD



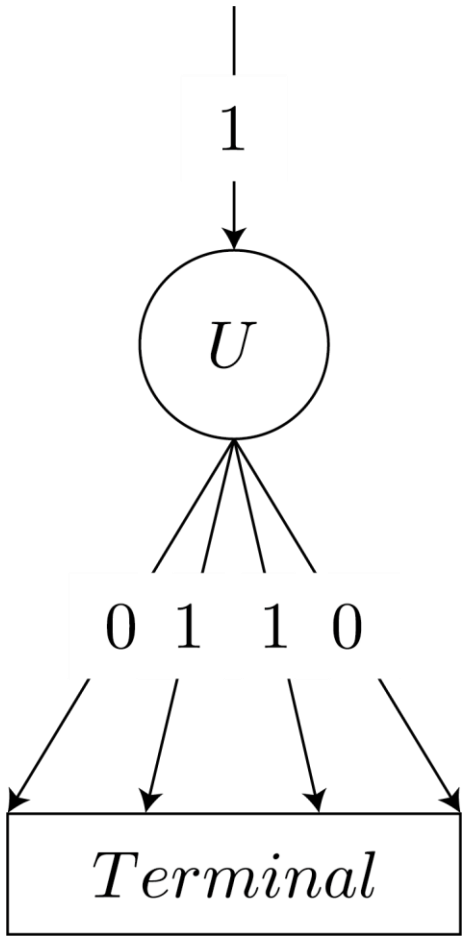
付録: 主なQMDD表現② 単一量子ゲート

行列表現

- 例）Xゲート

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

QMDD



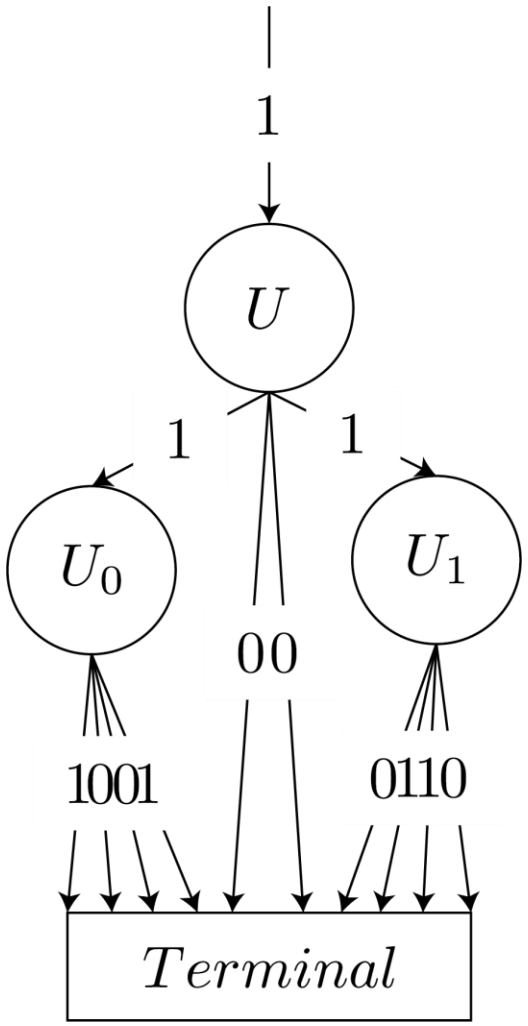
付録: 主なQMDD表現③ 複数量子ゲート

行列表現

- 例）$CNOT$ゲート

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

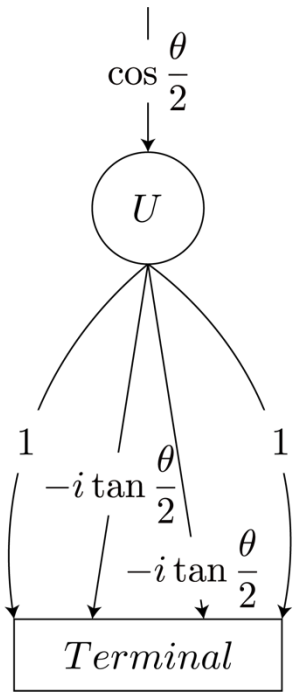
QMDD



付録: 主なQMDD表現④
実験で使用したゲート

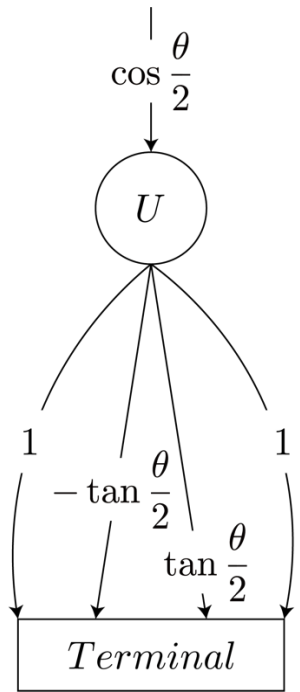
Rxゲート

$$\begin{pmatrix} cos\frac{\theta}{2} & -i\,sin\frac{\theta}{2} \\ -i\,sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{pmatrix}$$



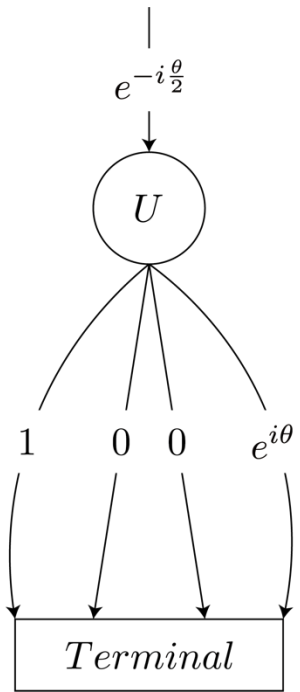
Ryゲート

$$\begin{pmatrix} cos\frac{\theta}{2} & -sin\frac{\theta}{2} \\ sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{pmatrix}$$



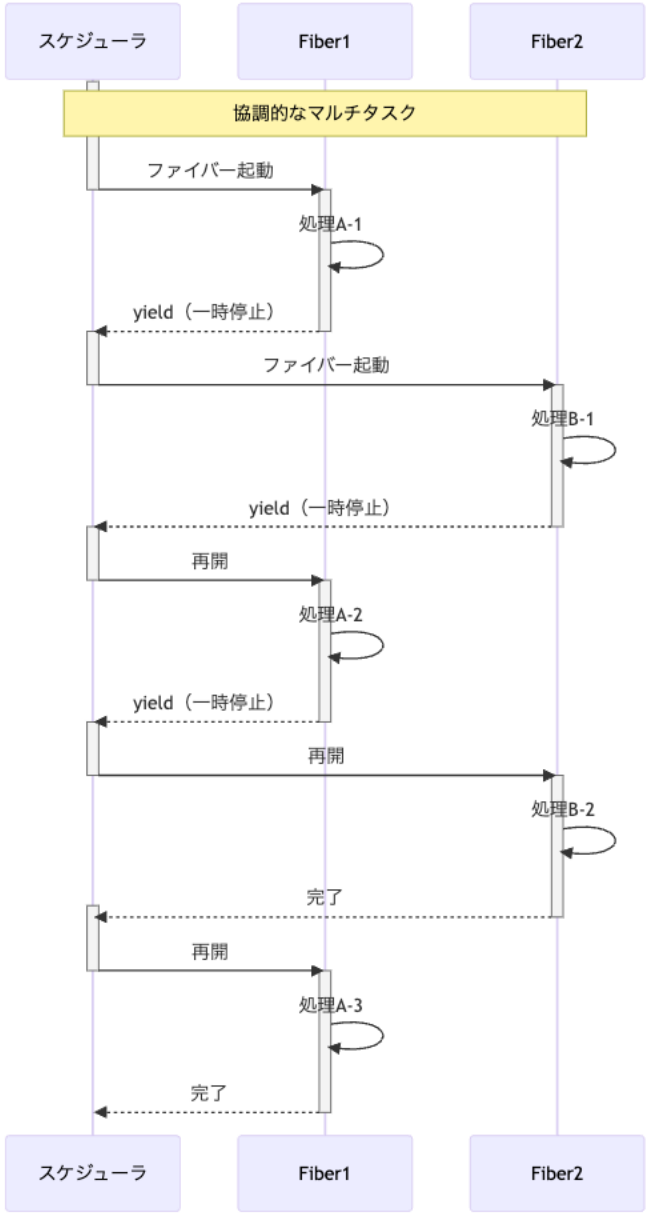
Rzゲート

$$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$



付録: ファイバーの概要

- 軽量な並行処理の単位
- 単一スレッド内で動作
- プログラム内で明示的に切り替え
  - ノンプリエンプティブ
    - $yield()$関数によってファイバーを切り替え
    - OSによるスレッドのスイッチングに比べて時間的なコストが小さい
- Boostライブラリで実装



付録:
対角行列に相当するQMDDの演算効率化

- 多くのゲートが対角行列
  - 位相ゲートやパウリZゲート

$$A = \begin{pmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & & \ddots & \\ 0 & & & a_{nn} \end{pmatrix} = diag(a_{11}, a_{22}, \ldots, a_{nn})$$

- 単位行列
  - 量子ゲートの拡張などに多用

- 演算中の再帰するエッジを限定的に
  - 重みが0になるエッジへの再帰を省略