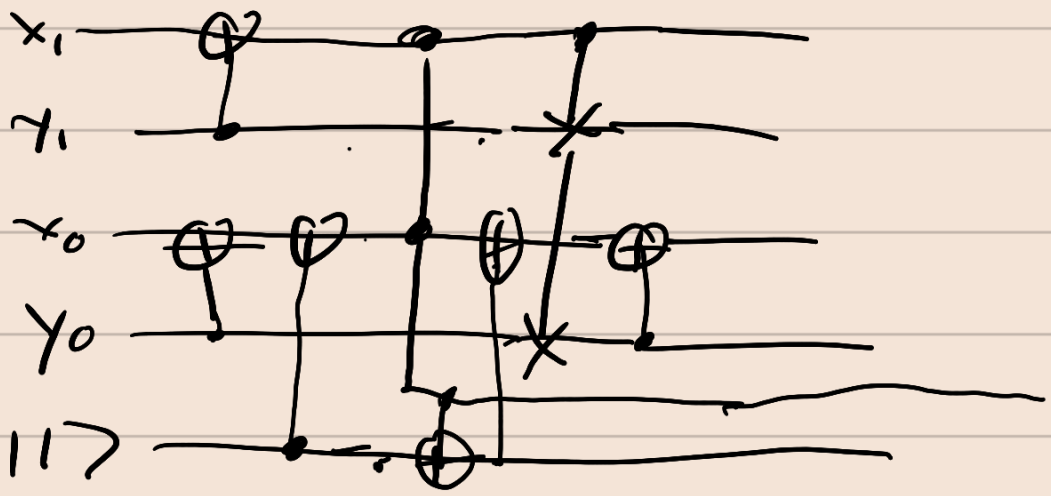
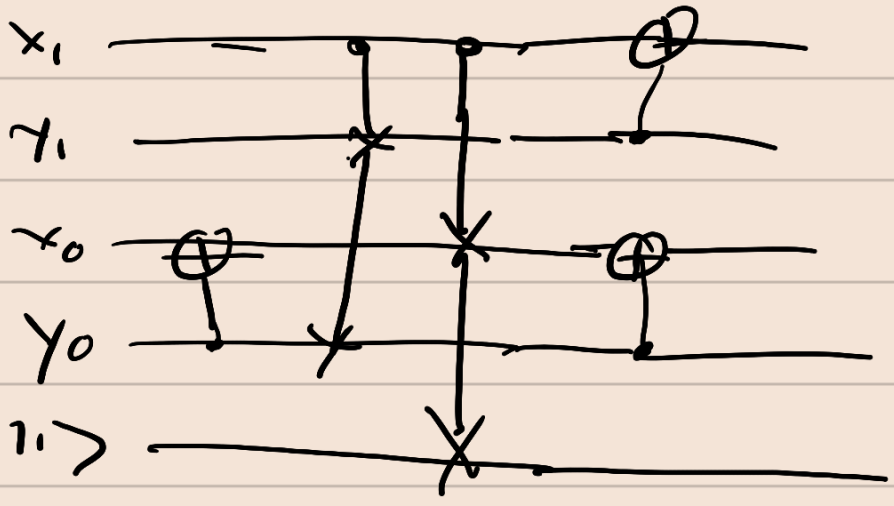
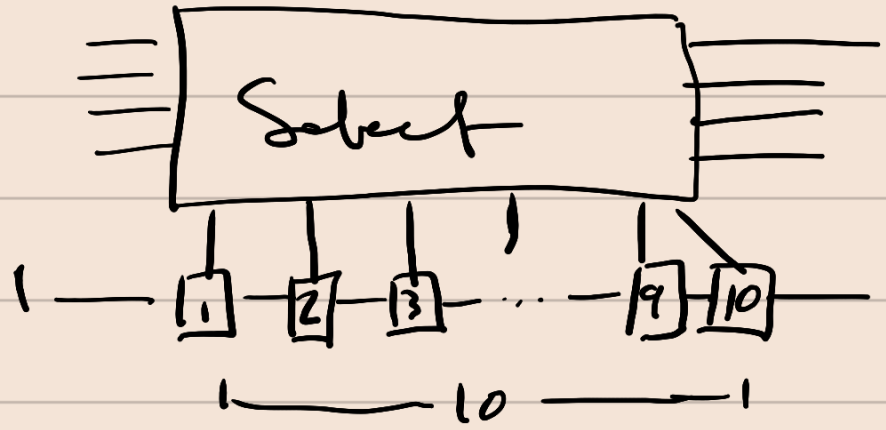
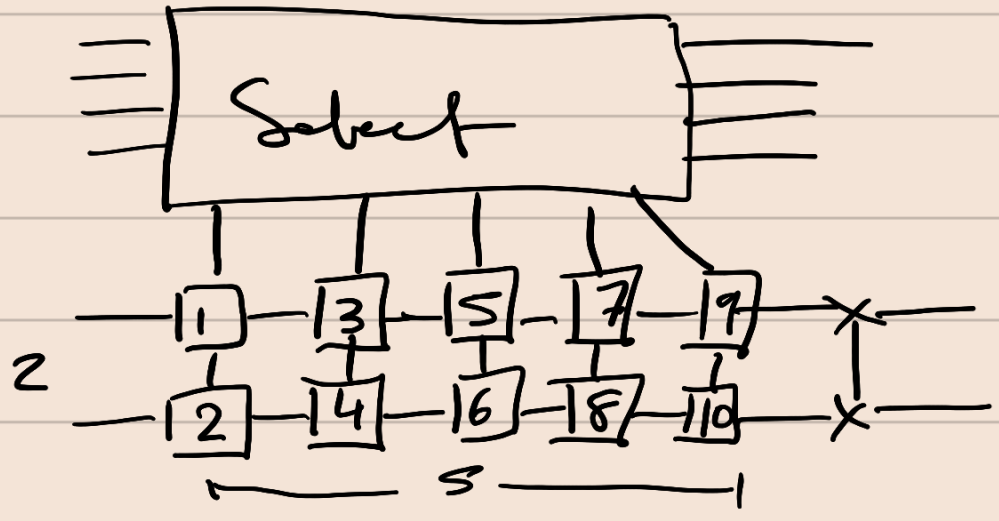

x_1
y_1
x_0
y_0
$|1\rangle$


x_1
y_1
x_0
y_0
$|1\rangle$

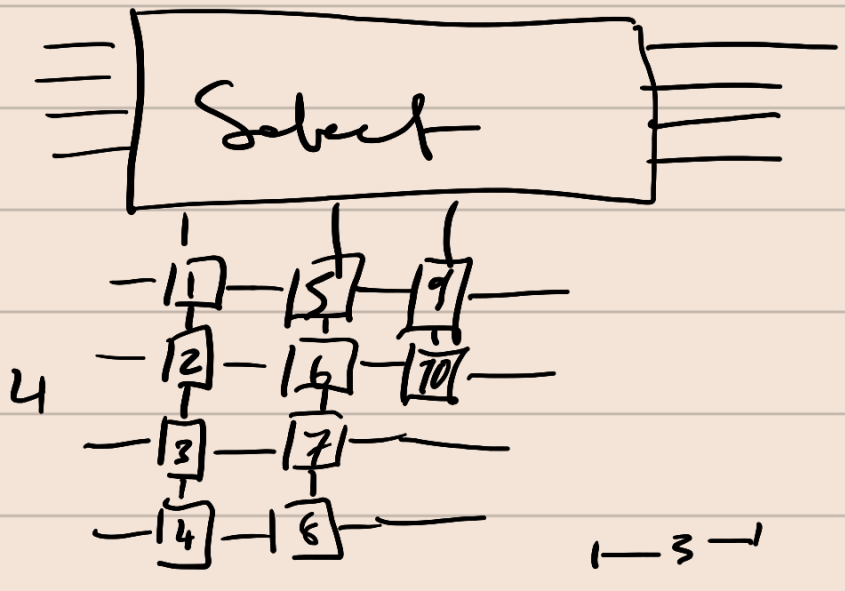
QROM(10, ⋯, selswap=1)



QROM(10, ⋯, selswap=2)



QROM(10, ⋯, selswap=4)



Counting Qubits

$$n_M = \lceil \log_2(M+1) \rceil$$

$n_M = 8$

N

58 orbitals

160 TR

M

$|4\rangle$ — 116 —

4-proj

$|\mu\rangle + |\nu\rangle = 16$

$$116 + 16 + 4 + (13)\cdot(57)$$

741

116

20

877

[select]

+7 [UI]

+13 [PG]

897

[prepare]

$\log_2\left(\frac{M(M+1)}{2} + N/2\right) = 10$

$+6$ [arrays] $+ 2 \cdot n_M + n_d$

$16 + 10 = 26$

$+ 2.13$ [prefixsum]

$+ 4$ [arr]

$10 + 26 + 26 = 62$

$+ \quad 897$

959

$$\frac{1}{\sqrt{2^b}} \sum_{k=0}^{2^b-1} \exp(-i 2\pi \cdot k/2^b) |k\rangle$$

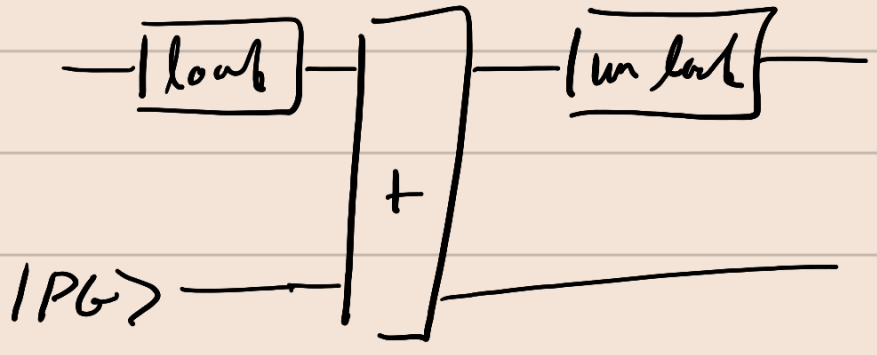
$b = 3$ $\quad 1 \cdot |0\rangle, \exp(i\pi/4)|1\rangle,$

$\exp(-i\pi/2)|2\rangle, \exp(-i 3\pi/4)|3\rangle,$

$\exp(-i\pi)|4\rangle, \exp(-i 5\pi/4)|5\rangle$

$\exp(-i 3\pi/2)|6\rangle, \exp(-i 7\pi/4)|7\rangle.$

$\exp\left(-i \pi/2^{b-1} \cdot \hat{Z}\right)$ $\qquad \frac{\pi}{2^{n-1}} \leq \varepsilon$



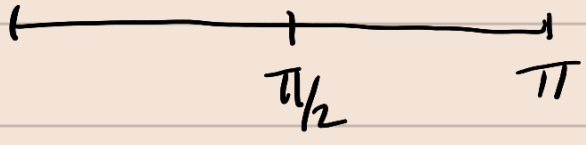
$e^{i\pi \cdot \hat{Z}}$

$\pi/\varepsilon \leq 2^{n-1}$

$\log_2(\pi/\varepsilon) \leq n-1$

$\Rightarrow \log_2(\pi/\varepsilon)+1 \leq n$

$\Rightarrow n \geq \lceil \log_2(\pi/\varepsilon)+1 \rceil$



$\{v_i\}$ each v_i is bit string of length $2N$

for example if $N = 2$

then a $v_i = [1\ 0\ 0\ 1]$

$\therefore$ set $\{v_i\}$ has at most 2^{2N}

consider: we have a subset of $\{v_i\}$ that contains D elements for $D << 2^{2N}$

Q: can we find a compression scheme that allows us to uniquely determine the subset w/o having to look at the whole v_i.

Example:

$N = 3$

$$S = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

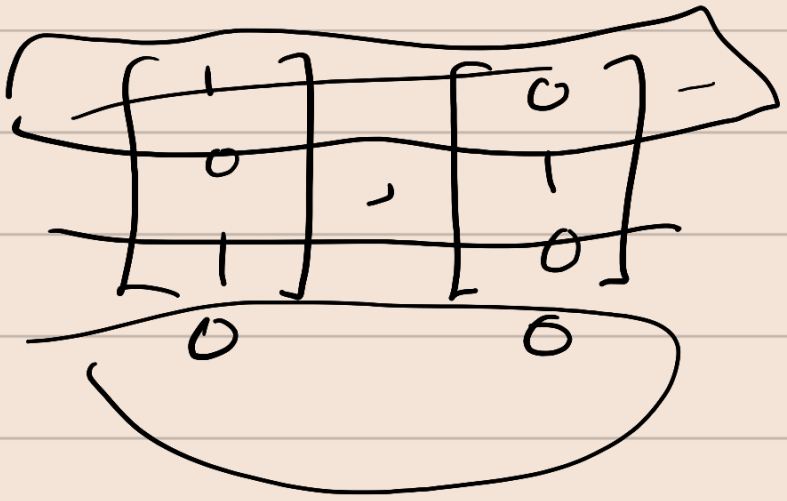
$$|S| << 2^6$$

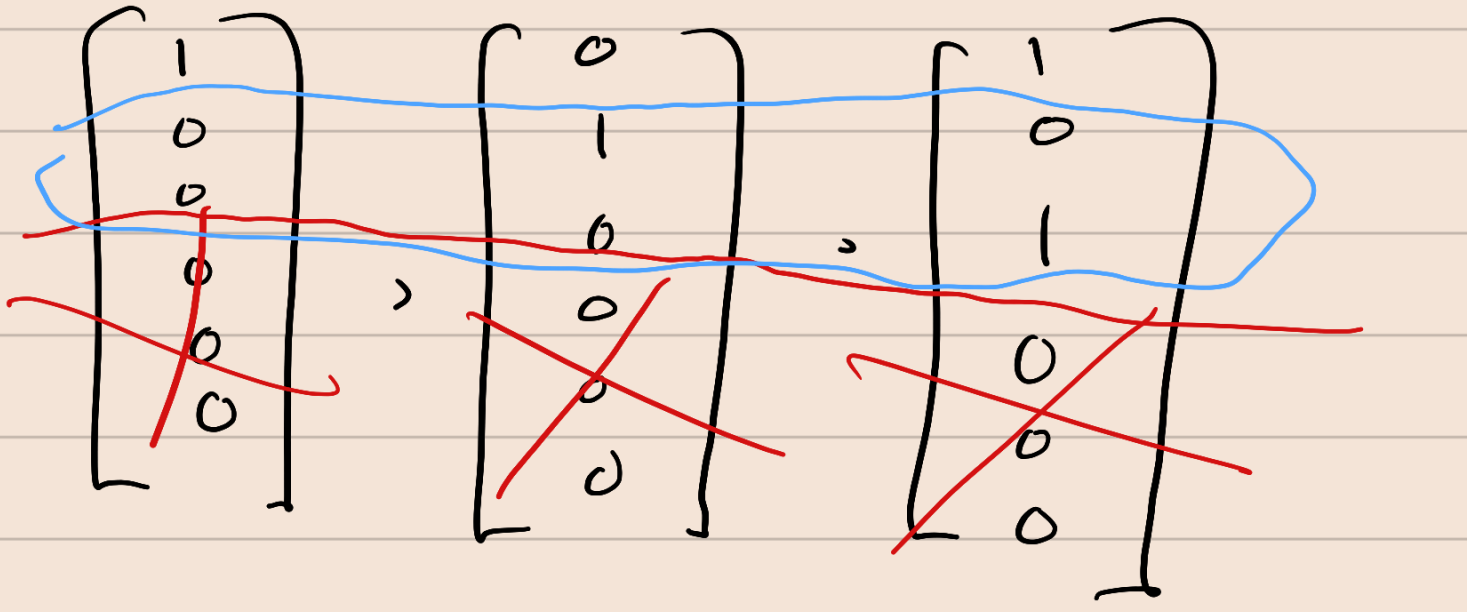
Suppose I get x as a bitstring

how do I check which basis element it is?

$$x = [\, x_0 \;\; x_1 \;\; x_2, \ldots, x_5 \,]$$

$$S_0 = [\, 1, 0, 0, \ldots, 0 \,]$$





① → [2, 3, 4 5]

↳ find matching indices V_1 V_2

② → [3, 4 5]

↳ find matching indices V_2, V_3

③ discard that indices

$$\theta \longrightarrow \tilde{\theta} \quad \text{s.t} \quad \|\theta - \tilde{\theta}\| \leq \varepsilon$$

Note: Suppose $\theta \in [0, 2\pi]$

$$\theta = \pi \cdot \left(\sum_{k=0}^{\infty} b_k \frac{1}{2^k} \right) \quad \forall \; b_k \in \{0, 1\}$$

$$\tilde{\theta} = \pi \cdot \left(\sum_{k=0}^{N-1} b_k \cdot \frac{1}{2^k} \right)$$

$[0, N-1] = N$ bits to represent

$$\Rightarrow \|\theta - \tilde{\theta}\| = \pi \cdot \left\| \sum_{k=N}^{\infty} b_k \cdot \frac{1}{2^k} \right\|$$

at most assume all $b_k = 1$ $\forall k \geq N$

$$\leq \pi \cdot \left\| \sum_{k=N}^{\infty} \frac{1}{2^k} \right\|$$

$$\leq \pi \cdot \frac{1}{2^{N-1}} \equiv \varepsilon$$

$\therefore$ If we use N bits to approximate θ, we get an accuracy of $\varepsilon = \pi / 2^{N-1}$

Vice Versa

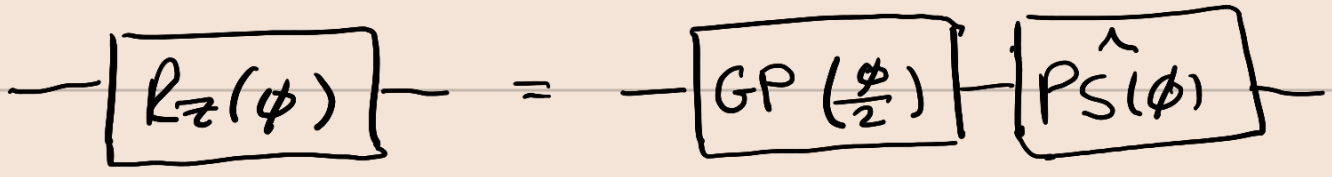
$\therefore$ If we have a target error ε, we need atleast $N = \lceil \log_2(\pi/\varepsilon) \rceil + 1$

Recall: $\hat{R}_z(\phi) := e^{-i\frac{\phi}{2}\hat{z}}$

$$= \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix},$$

$\hat{PS}(\phi) = e^{-i\phi/2} \cdot \hat{R}_z(\phi)$

$$= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \Rightarrow \hat{R}_z(\phi) = e^{i\phi/2}\, \hat{PS}(\phi)$$



$$\hat{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\hat{X}|\alpha\rangle = \lambda|\alpha\rangle$$

$$\Rightarrow \begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\Rightarrow$ $\Rightarrow$ $\alpha_2 - \lambda\alpha_1 = 0 \rightarrow \alpha_2 = \lambda\alpha_1$

$\alpha_1 - \lambda\alpha_2 = 0$

$\Rightarrow \alpha_1 - \lambda(\lambda\alpha_1) = 0$

$\Rightarrow \alpha_1 \cdot (1 - \lambda^2) = 0 \Rightarrow \lambda^2 = 1$

$\Rightarrow \boxed{\lambda = \pm 1}$

$\Rightarrow \begin{bmatrix} \alpha_2 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \Rightarrow \alpha_1 = \alpha_2$

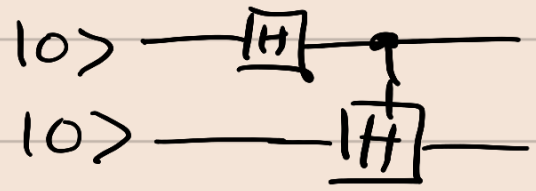
and $\alpha_1 = -\alpha_2$

$\Rightarrow$

$$R_x(\phi) := e^{-i\frac{\phi}{2}\hat{X}} \equiv e^{i2\pi\theta\hat{X}}$$

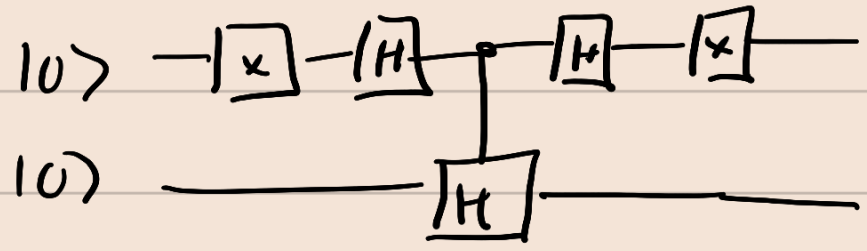
$$\Rightarrow -i\frac{\phi}{2} = i2\pi\theta$$

$$\phi = -4\pi\theta$$



$$|00\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle$$

$$\to \frac{1}{\sqrt{2}}|0\rangle|0\rangle$$



$$|00\rangle \to |10\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|0\rangle$$

$$\to \frac{1}{\sqrt{2}}\left(|00\rangle - \tfrac{1}{\sqrt{2}}|1\rangle(|0\rangle + |1\rangle)\right)$$

$$= \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{2}(|10\rangle + |11\rangle)$$

$$\downarrow H$$

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|0\rangle\right) - \frac{1}{2}\left(\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)\right)$$

$$= \frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle - \frac{1}{2\sqrt{2}}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$$

$$\downarrow X$$

$$\frac{1}{2}|10\rangle + \frac{1}{2}|00\rangle - \frac{1}{2\sqrt{2}}(|10\rangle + |11\rangle - |00\rangle - |01\rangle)$$

$$\left(\frac{\sqrt{2}+1}{2\sqrt{2}}\right)|00\rangle + \frac{1}{2\sqrt{2}}|01\rangle + \left(\frac{\sqrt{2}-1}{2\sqrt{2}}\right)|10\rangle - \frac{1}{2\sqrt{2}}|11\rangle$$

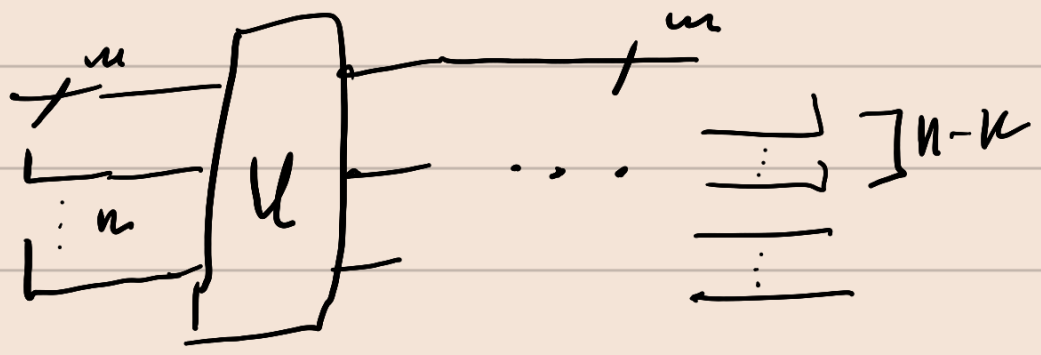
Qubit allocation and re-use cases:

- MCX:
  - clean ☐
  - borrowed ☐
- Toffoli:
  - default ☐
  - elbows ☐
- SemiAdder ☐
- controlled-semi adder ☐
- QROM select-swap ☐
- Multiplexer ☐
- Qubit unitary ☐
- QROM state preparation ☐
- MPS state preparation ☐
- prep sel prep† ☐

Dummy Circuit :

■ Algo qubits

■ Allocated qubits



Halloc(n), ..., Free(n-k)

Tracking algorithm :

Single Resource Operator

$Op(w_1, w_2, w_3)$

QM: $A_q : [w_1, w_2, w_3]$

$c_q : 0$, $D_q : 0$

decompose

QM: $(A_q = [w_1, w_2, w_3], c_q = 0, D_q : 0)$, $R = \{\}$

$[\ 3 \times SOp_1(w_1, w_2)$, $Alloc(2)$, $2 \times SOp_2(w_3, A_1)$,
$\hookrightarrow (A_1, A_2)$
$Free(1)\ ]$

decompose

QM: $(A_q : [w_1, w_2], c_q = 0, D_q = 0)$, $\times 3$, $R = \{\}$

$[\ Alloc(3)$, $SOp_{11}(w_1, w_2, A) \cdot free(2)\ ]$

$\hookrightarrow (A_1, A_2, A_3)$

QM: $(A_q : [w_1, w_2], c_q = 0, D_q = 3)$, $R = \{\}$

QM: $(A_q : [$

To do:

- -> fix _ops_to_compressed_reqs to not rely on queue category
- -> fix tracking to properly map PL ops -> RE ops and not rely on queue category

Problems / Questions:

① What happens when an algorithm doesn't free all of the qubits it allocated?

↳ How common is this edge case?

↳ What are reasonable assumptions here?

② At any stage/level of the decomposition how many wires are available for use?

aux ↳ How many clean?

↳ How many dirty?

↳ What if we guarantee they will be returned?

③ What features do we want to support?

↳ In resource estimation?

↳ In decomposition?

④ What happens to the Qubit resources as you go back up a layer in the decomposition?

Answers:

(1): In practice most algorithms that use auxiliary qubits WILL clean them within the scope of the same decomposition.

There is one edge case: SELSWAPQROM

This often appears in the following applications

→ QROM state preparation.

→ Phase Gradient decomposition of the multiplexer gate.

Another potential edge case could be: $\text{Prep Sel Prep}^\dagger$

this is another example where the compute-uncompute pattern would allow users to postpone the cleaning.

The pattern can be summarized as

reasonable assumptions

→ we could assume that the only time qubits are allocated and not freed in the same decomposition happens when we have a C-UC pattern. Thus we can always write decompositions assuming things are returned totally within the scope of the decomposition.

② : There are three options when requesting qubits in a circuit:

a) request qubits in a "clean" ($|0\rangle^{\otimes n}$) state

b) request qubits in an unknown "dirty" state

c) request qubits in a known fixed state

We focus on the first two for now:

We can source qubits of type (a) by

↳ allocating a fresh auxiliary qubit

↳ re-using a cleaned auxiliary qubit

↳ using an idle, clean, algorithmic qubit and returning it in the same scope/level of the decomposition.

We can source qubits of type (b) by:

→ allocating a fresh auxiliary qubit

→ re-using existing clean or dirty auxiliary qubits

→ using any idle (clean or dirty) algorithmic qubit. Provided it is returned to the previous state within the same scope/level of the decomposition.

Def: Idle qubits

Given an operator with wires $W_o = \{W_{o_1}, \dots\}$ Suppose this operator was called in a circuit with wires $W_c = \{W_{c_1}, \dots\}$. The set of Idle qubits (W_{Idle}) is a subset of W_c s.t

$$W_{Idle} \cap W_o = \emptyset.$$

Example 1

Gfunc

[Up1(W1, W2), Allocate_c(2), Up2(W1, W2, W3), Free_c(2)]

(1) (5) (6) (10) (11)

Up1(W1, W2) —decompose→ [Allocate_c(1), Allocate_d(1), ..., Free_c(1), free_d(1)]

(2) (3) (4)

Up2(W1, W2, W3) —decompose→ [Allocate_d(3), ..., Free_d(3)]

(7) (8) (9)

Iterate Gfunc and get list of wires → (W1, W2, W3)

(1)

Algo: 3

	clean	dirty
idle		
active	III	

Aux: 0

	C	D
I		
A		

(2) num_wires = 2

Algo: 3

	clean	dirty
idle	I	
active		II

Aux: 0

	C	D
I		
A		

(3)

Algo: 3

	clean	dirty
idle		
active		III

Aux: 1

	C	D
I		
A		I

④

Alge : 3

	C	D
I	I	
A		II

Aup : 1

	C	D
I	I	
A		

⑤

Alge : 3

	C	D
I		
A	I	II

Aup : 1

	C	D
I	I	
A		

⑥

Alge : 3

	C	D
I		
A	I	II

Aup : 2

	C	D
I		
A		II

⑦ num_wires = 3

Alge : 3

	C	D
I		
A		III

Aup : 2

	C	D
I		II
A		

⑧

Alge : 3

	C	D
I		
A		III

Aup : 3

	C	D
I		
A		III

(9)

Alge: 3

	C	D
I		
A		III

Aug: 3

	C	D
I	I	II
A		

(10)

Alge: 3

	C	D
I		
A		III

Aug: 3

	C	D
I	I	
A		II

(11)

Alge: 3

	C	D
I		
A		III

Aug: 3

	C	D
I	III	
A		

Example 2

gfunc : [Allocate_c(2) , $op_1(w_1, w_2, w_3)$, $Free_c(2)$]

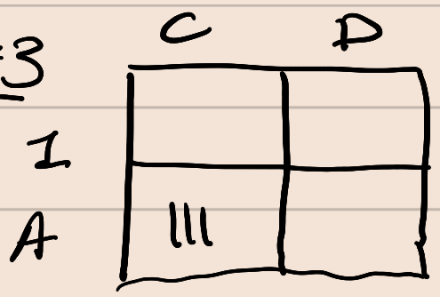
$op_1(w_1, w_2, w_3)$ → decompose → [$op_2(w_1)$, Allocate_c(2) , $op_3(w_1, w_2)$, free_c(2)]

$op_2(w_1)$ → decompose → [Allocate_d(1) , ... , Free_d(1)]

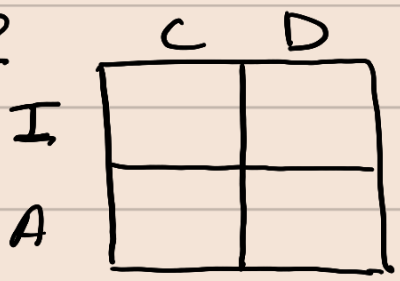
$op_3(w_1, w_2)$ → decompose → [Allocate_d(1) , Allocate_c(1) , ... , Free_d(1) , Free_c(1)]

(1)

Alge: 3

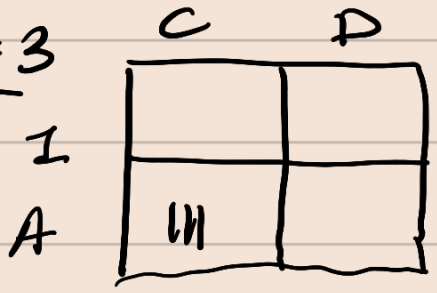


Amp: 0

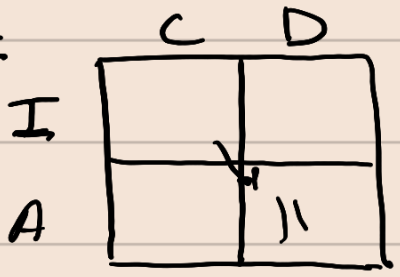


(2)

Alge: 3

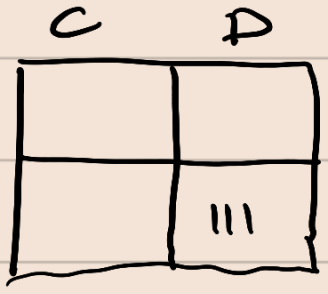


Amp: 2

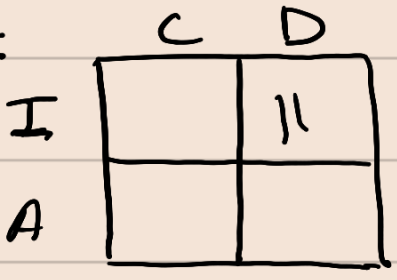


(3) num_wires = 3

Alge: 3

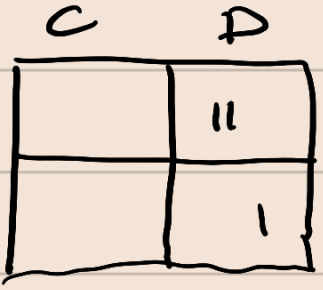


Amp: 2

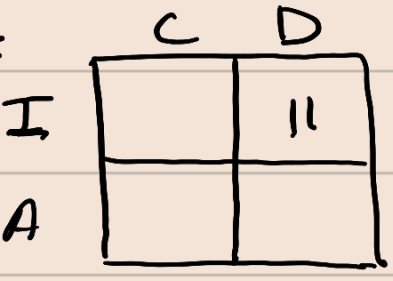


(4) num_wires = 1

Alge: 3

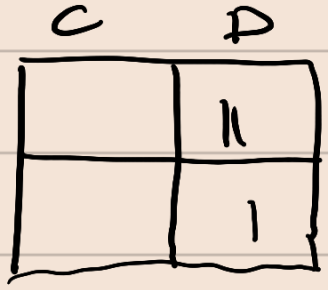


Amp: 2

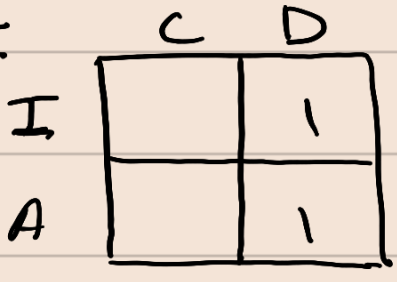


(5)

Alge: 3

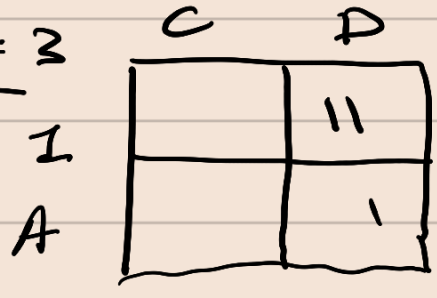


Amp: 2

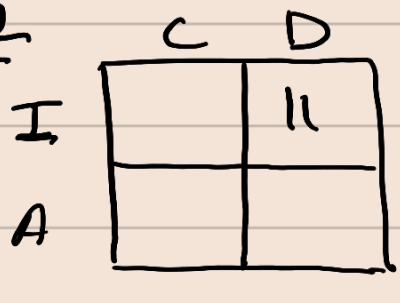


(6)

Alge: 3

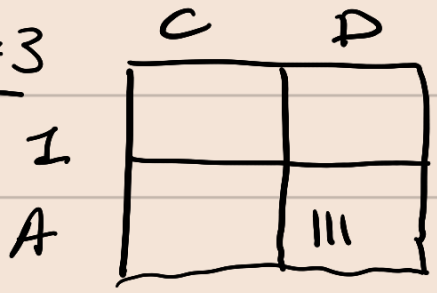


Amp: 2

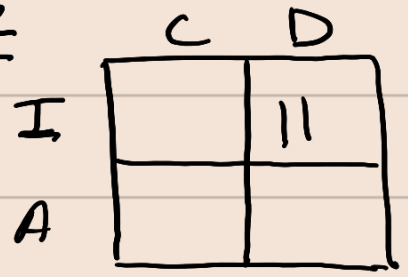


(8)

Algo: 3

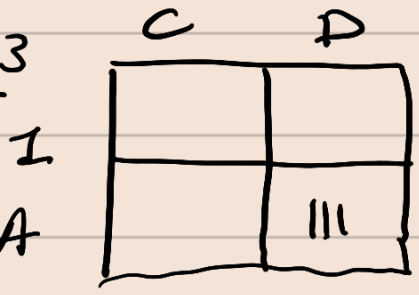


Aup: 2

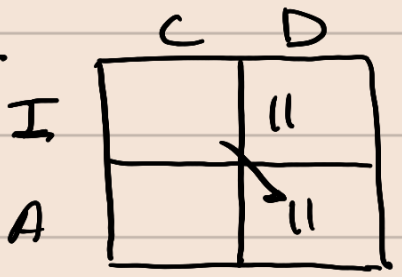


(9)

Algo: 3



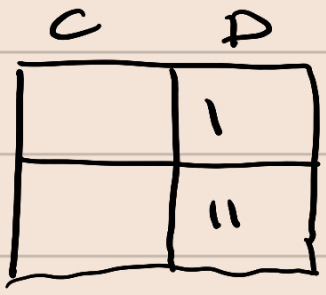
Aup: 4



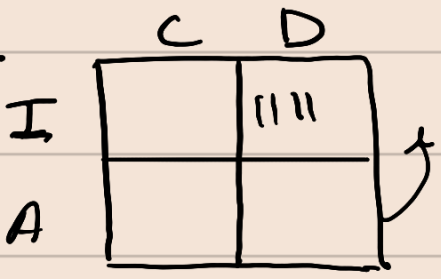
(10)

num_votes = 2

Algo: 3

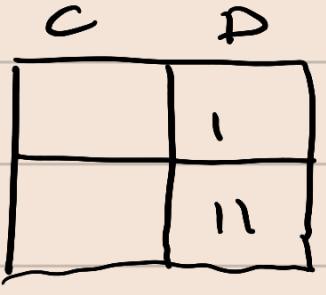


Aup: 4

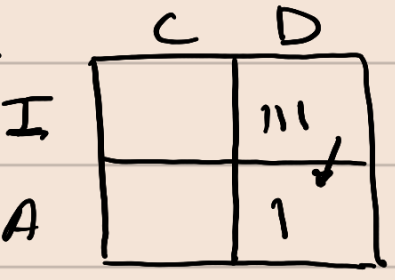


(11)

Algo: 3

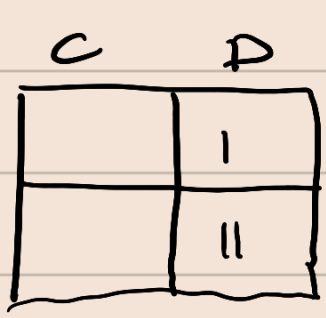


Aup: 3

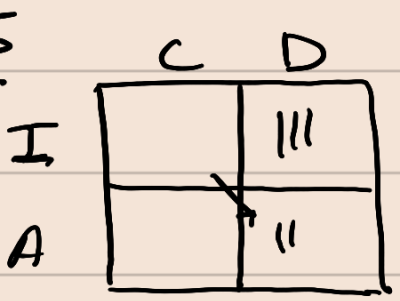


(12)

Algo: 3

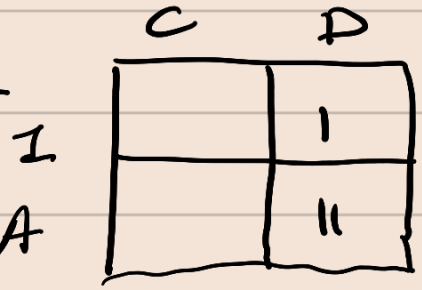


Aup: 5

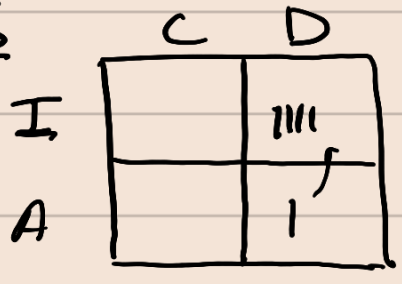


(13)

Algo:



Aup: 5



(13)

Alge : 3

	C	D
I		|
A		||

Aup : 5

	C	D
I	|	||||
A		

(14)

Alge : 3

	C	D
I		
A		|||

Aup : 5

	C	D
I	|	||
A		||

(15)

Alge : 3

	C	D
I		
A		|||

Aup : 5

	C	D
I	|||	||
A		

(16)

Alge : 3

	C	D
I		
A		|||

Aup : 5

	C	D
I	|||	
A		||

(17)

Alge : 3

	C	D
I		
A		|||

Aup : 5

	C	D
I	~~||||~~	
A		

Notes from example:

→ iterate over the tape and determine # algo wires (w/ names if any)

→ at the "Top level" (future or op)

↳ all algo wires are set to clean & active.

Step down instructions:

Example 3

[$Op_1(w_1, w_2)$, $Op_1(w_3, w_4)$, $Op_2(w_5)$, $Op_3(w_1, w_2, w_3, w_4)$]

① $Op_2(w)$ → decompose → [Allocate_c(3), ..., Free_c(3)]

② ③ ④ ⑤ ⑥ ⑦ ⑧

① Alge: 5

	C	D
I		
A	5	

Aux: 0

	C	D
I		
A	x	

② Alge: 5

	C	D
I		
A	3	2

Aux: 0

	C	D
I		
A	x	

③ Alge: 5

	C	D
I		
A	1	4

Aux: 0

	C	D
I		
A	x	

④ num_wires = 1

Alge:

	C	D
I		4
A		1

Aux: 0

	C	D
I		
A	x	

(5)

Alge: 5

	C	D
I		4
A		1

Aux: 3

	C	D
I		
A	x	3

(6)

Alge: 5

	C	D
I		4
A		1

Aux: 3

	C	D
I	3	
A	x	

(7)

Alge: 5

	C	D
I		
A		5

Aux:

	C	D
I	3	
A	x	

(8)

Alge: 5

	C	D
I		
A		5

Aux:

	C	D
I	3	
A	x	

→ Wire labels affect the borrowable qubits

Example 4:

[Alloc_c(2), $Op_1(A_1, W_1)$, $Op_2(A_2, W_2, W_3)$, Free_c(2)]

Op_1 → decomposes → [Alloc_c(1), ..., Free_c(1)]

Op_2 → decomposes → [Alloc_d(2), ..., Free_d(2)]

(1)

Alge: 3

	C	D
I		
A	3	

Aux: 0

	C	D
I		
A		

(2)

Alge: 3

	C	D
I		
A	3	

Aux: 2

	C	D
I	2	
A		

AW_1

I: A_1, A_2

A:

(3)

Alge: 3

	C	D
I		
A	3	

Aux: 2

	C	D
I	2	
A		

AW_2

I: $A_1 A_2$

A:

(4) num_wires = 2

Alge: 3

	C	D
I	2	
A		1

Aux: 2

	C	D
I	1	
A		1

AW_3

I: A_2

A: A_1

(5)

Alge : 3

	C	D
I	2	
A		1

Aup : 2

	C	D
I	0	
A		2

(6)

Alge : 3

	C	D
I	2	
A		1

Aup : 2

	C	D
I	1	
A		1

(7)

Alge : 3

	C	D
I		
A	2	1

Aup : 2

	C	D
I	1	
A		1

(8)

Alge : 3

	C	D
I		
A		3

Aup : 2

	C	D
I		
A		2

Aw
I:
A: 2

(9)

Alge : 3

	C	D
I		
A		8

Aup : 4

	C	D
I		
A		4

(0)

Alge: 3

	C	D
I		
A		3

Aup: 4

	C	D
I	2	
A		2

(1)

Alge:

	C	D
I		
A		3

Aup:

	C	D
I	2	
A		2

(12)

Alge:

	C	D
I		
A		3

Aup:

	C	D
I	2	2
A		

Alge:

	C	D
I		
A		

Aup:

	C	D
I		
A		