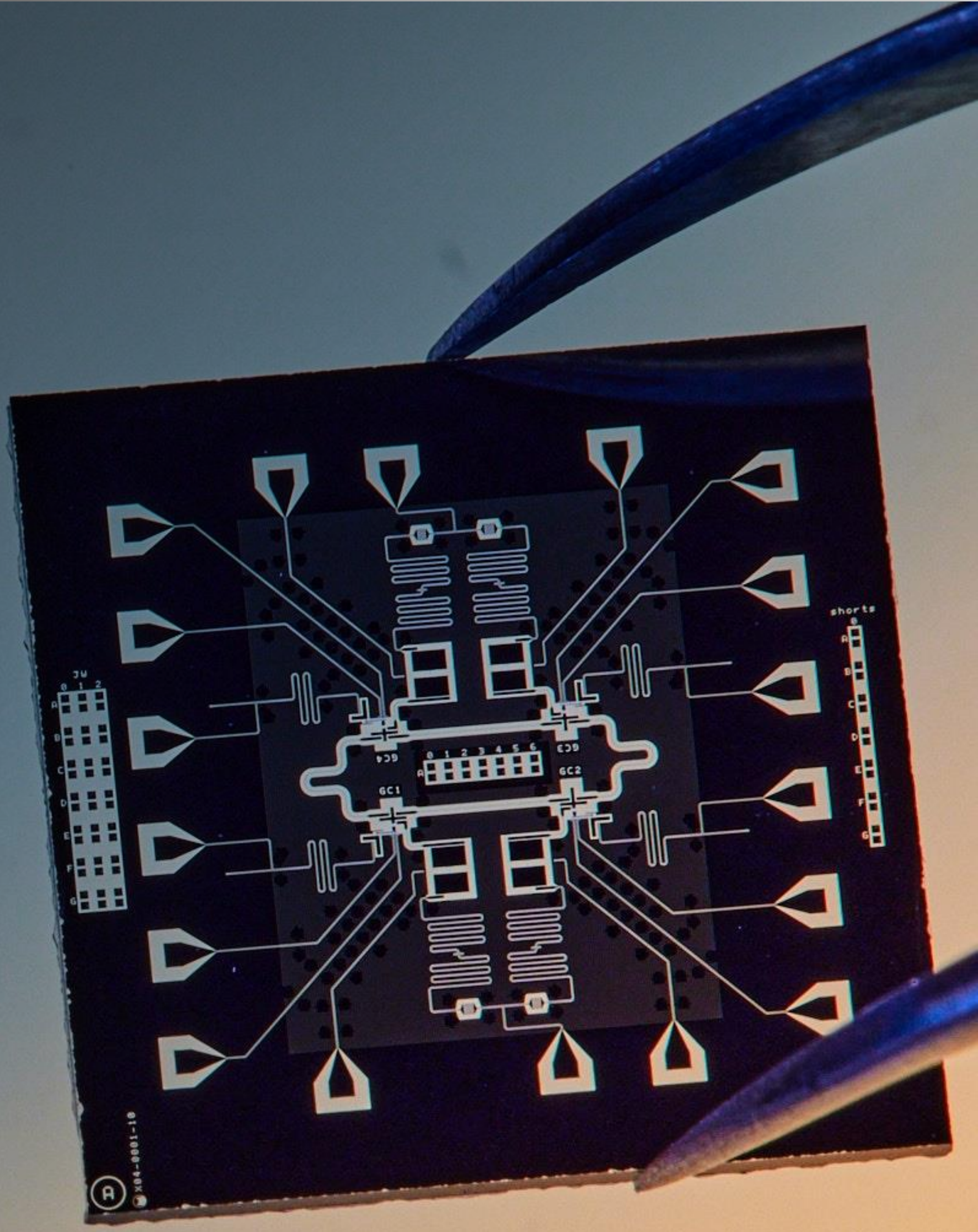
ALICE & BOB

Compiling Shor's algorithm: tricks for programming fault-tolerant quantum computers

12 November 2025

Quarc, UCLA, Los Angeles, USA

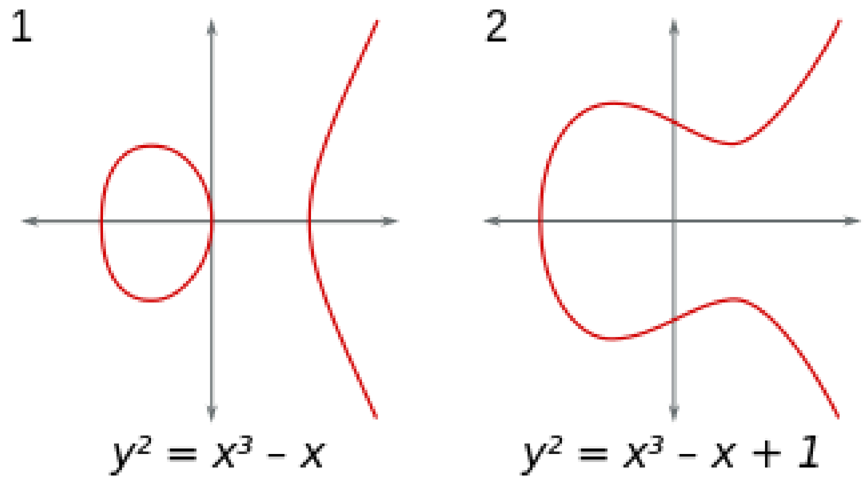


Elliptic Curve Cryptography

Elliptic curve group

$$y^2 = x^3 + ax + b$$

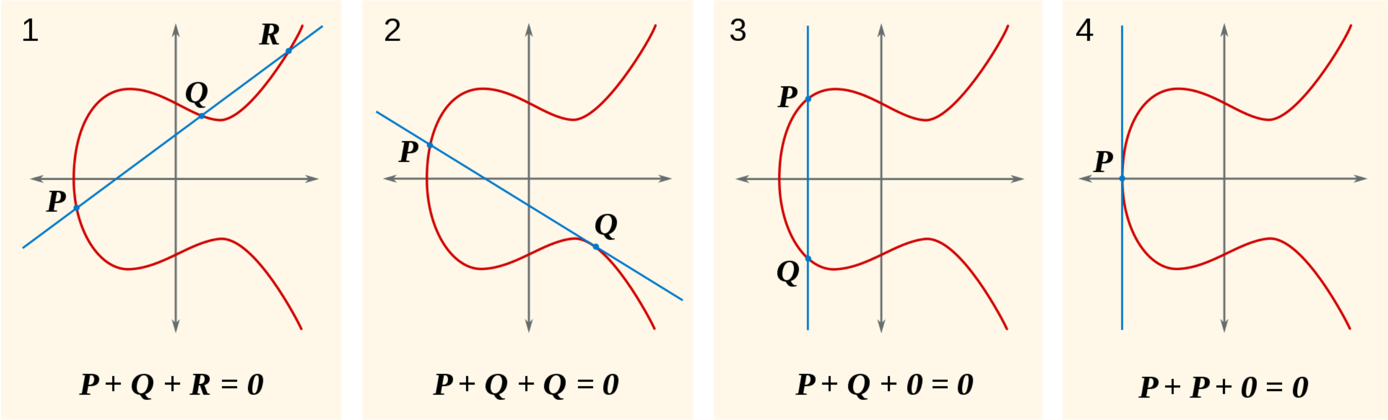
$a, b \in \mathbb{Z}_p$ with p prime in crypto
(x, y) also taken module p



Elliptic Curve Cryptography

Elliptic curve group

Addition



$$\begin{cases} x_R = \lambda^2 - x_P - x_Q \\ y_R = y_P + \lambda(x_R - x_P) \end{cases}$$

$$\lambda = \frac{x_P - x_Q}{y_P - y_Q}$$

Multiplication

$$kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$$

Elliptic Curve Cryptography

Diffie-Hellman key exchange

Shared knowledge (public)

$$y^2 = x^3 + ax + b$$

$$G = (x_0, y_0)$$

k_a

$$\xrightarrow{k_a G}$$

$$\xleftarrow{k_b G}$$

Public channel

k_b

$$k_a(k_b G)$$

$$k_b(k_a G)$$

Elliptic Curve Cryptography

Elliptic Curve Digital Signature Algorithm (ECDSA): Bitcoin parameters

Elliptic curve **secp256k1** (public) $y^2 = x^3 + 7$ and $p = 2^{256} - 2^{32} - 977$

Public point of the curve. $G = (x_0, y_0)$

$x_0 =$ 55066263022277343669578718895168534326
250603453777594175500187360389116729240

$y_0 =$ 32670510020758816978083085130507043184
471273380659243275938904335757337482424

Order of the group (prime number)
r = 115792089237316195423570985008687907852837564279074904382605163141518161494337

World-record (discrete-logarithm): 114-bit private key

Computing a discrete logarithm

Knowing $(G, P = lG)$, how to compute l ?

$$f(x_1, x_2) = x_1 G - x_2 P$$

Prepare 2 registers

$$\frac{1}{r}\sum_{x_1=0}^{r-1}\sum_{x_2=0}^{r-1}|x_1\rangle|x_2\rangle$$

Apply f

$$\frac{1}{r}\sum_{x_1=0}^{r-1}\sum_{x_2=0}^{r-1}|x_1\rangle|x_2\rangle|f(x_1,x_2)\rangle$$

Quantum Fourier transform

$$\sum_{y_1,y_2=0}^{r-1}\sum_{k=0}^{r-1}\left[\frac{1}{r^2}\sum_{\substack{x_1,x_2=0\\ f(x_1,x_2)=kG}}^{r-1} e^{2\pi i(x_1y_1+x_2y_2)/r}\right]|y_1\rangle|y_2\rangle|kG\rangle$$

Measure

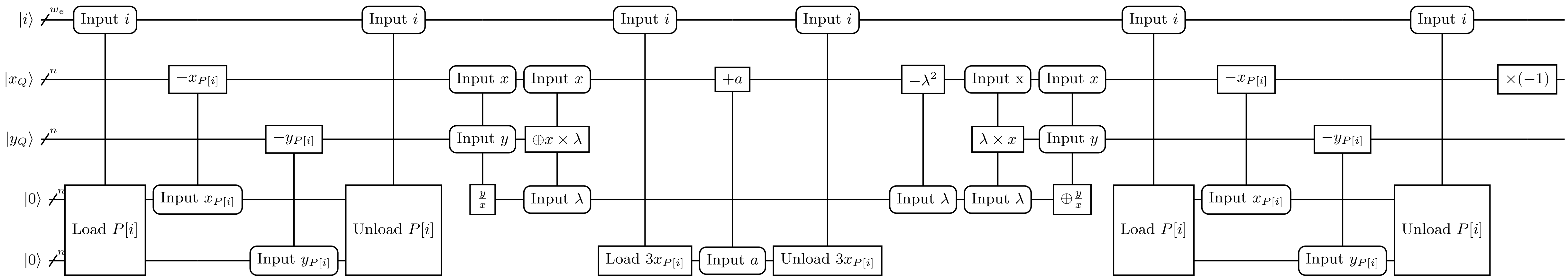
$$l = -y_2 y_1^{-1}$$

Computing a discrete logarithm

Elliptic curve multiplication: $|e\rangle|P\rangle \mapsto |e\rangle|eP\rangle$

$$eP = \sum_{i=0}^{n_e-1} 2^i e_i P$$

Elliptic curve Lookup-addition: $|i\rangle|Q\rangle \mapsto |i\rangle|P[i] + Q\rangle$



What is an addition?

Decimal addition

$$
\begin{array}{cccccc}
 & 1 & & 1 & 1 & \\
 & 2 & 3 & 1 & 6 & 7 \\
+ & & 9 & 2 & 6 & 7 \\
\hline
 & 3 & 2 & 4 & 3 & 4
\end{array}
$$

Binary addition

$$
\begin{array}{cccccccccccccccc}
 & & & & & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
+ & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0
\end{array}
$$

How to compute the carries?

Truth table

a_k	b_k	c_k	c_{k+1}	r_k
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Expression

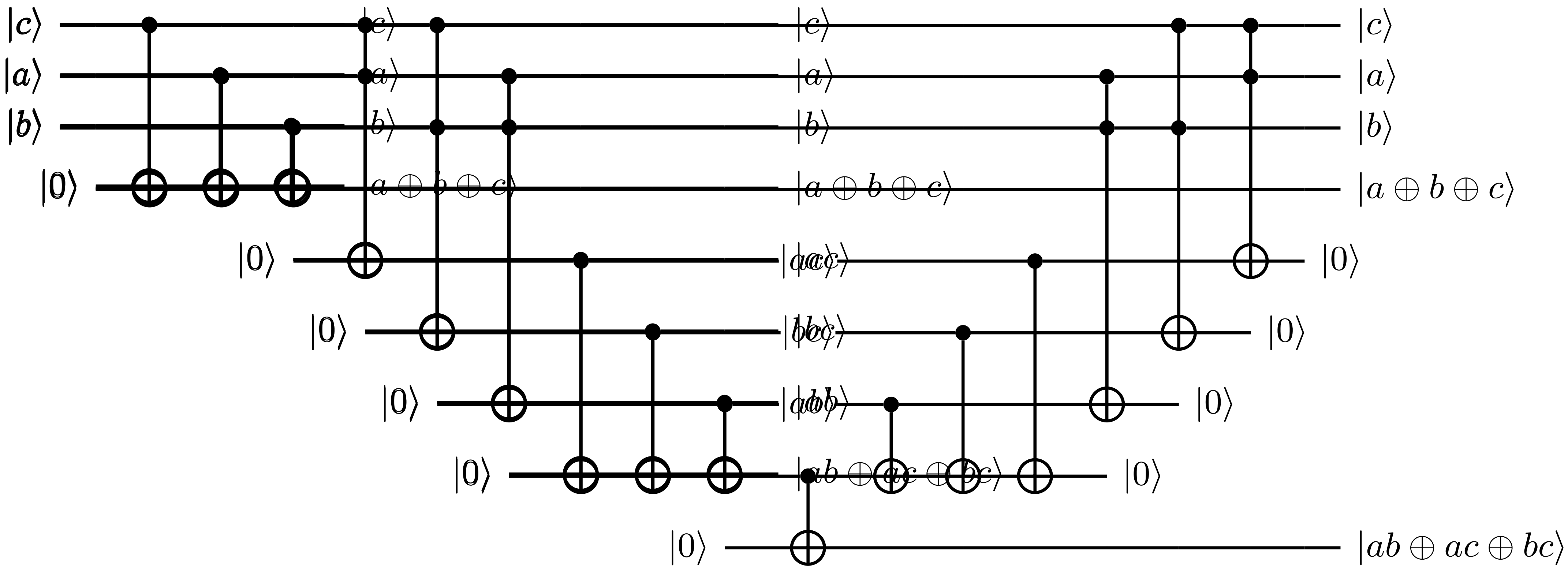
$$c_{k+1} = a_k b_k \oplus b_k c_k \oplus a_k c_k$$

$$r_k = a_k \oplus b_k \oplus c_k$$

Bennet's trick

$$c_{k+1} = a_k b_k \oplus b_k c_k \oplus a_k c_k$$

$$r_k = a_k \oplus b_k \oplus c_k$$



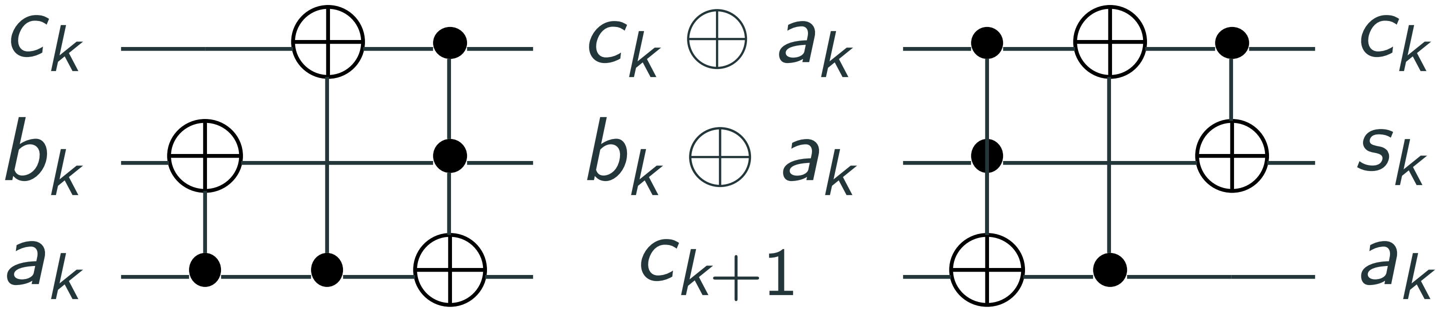
Can we do better?

Simplification

$$c_{k+1} = a_k b_k \oplus b_k c_k \oplus a_k c_k = (a_k \oplus b_k)(a_k \oplus c_k) \oplus a_k$$

$$s_k = a_k \oplus b_k \oplus c_k$$

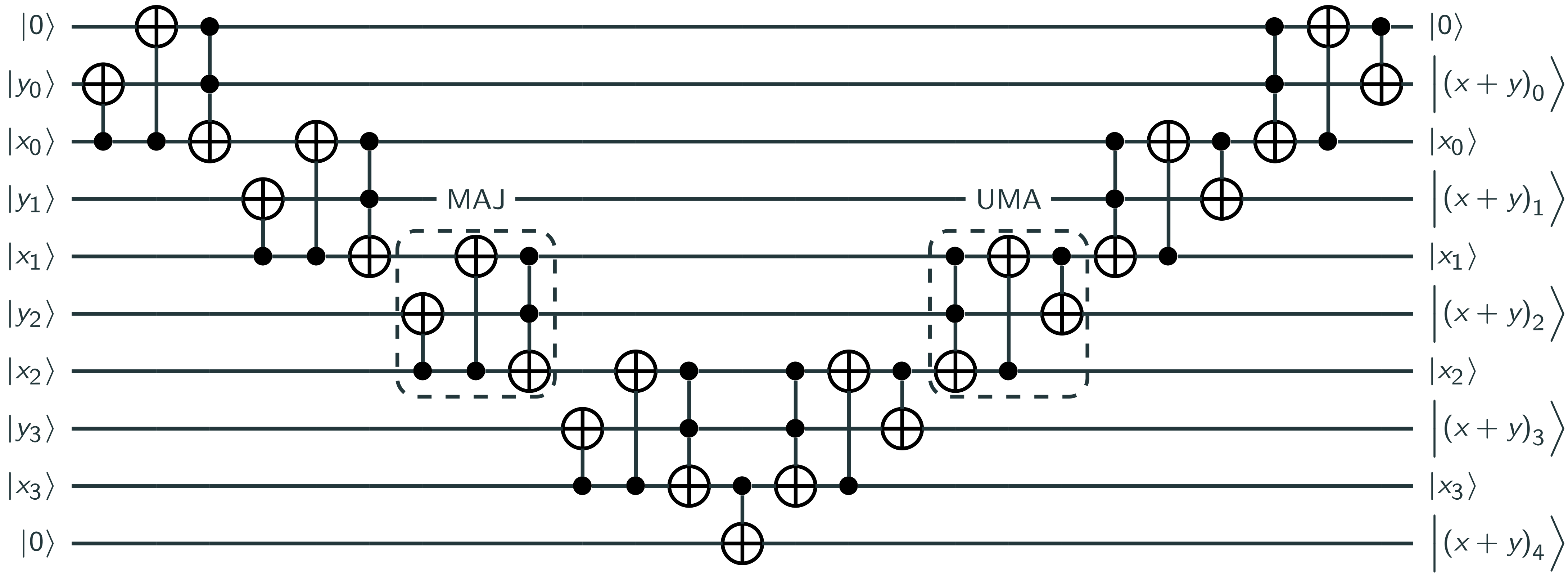
3-bits adder



Convention: left part = MAJ (MAJority); right part = UMA (UnMajority and Add)

Cuccaro et al. 2004. arXiv: quant-ph/0410184

Full Cuccaro’s adder



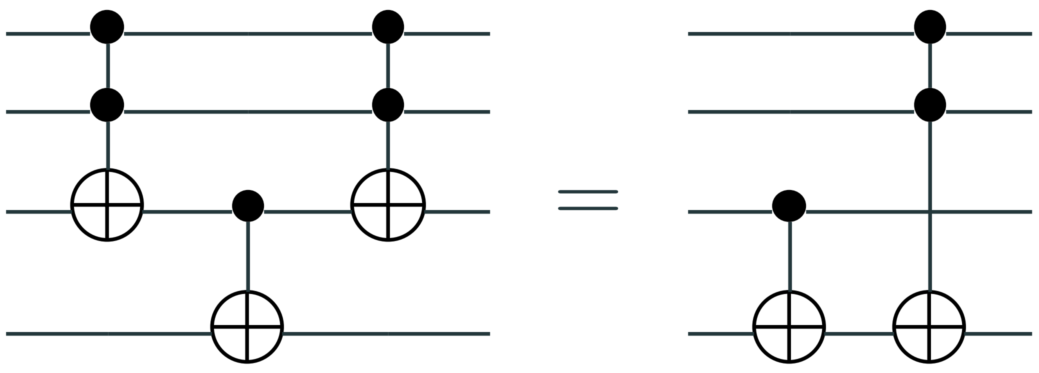
Improvements?

First qubit

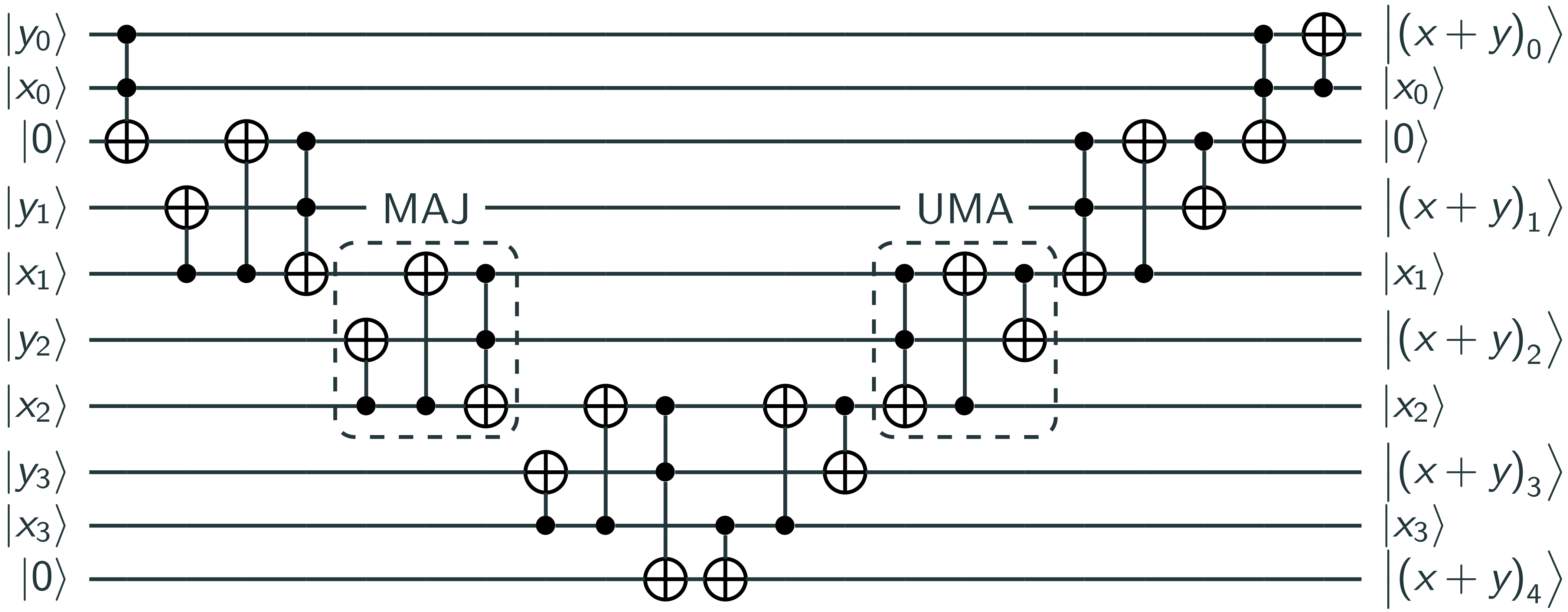
a_0	b_0	c_1	r_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

No need for full MAJ

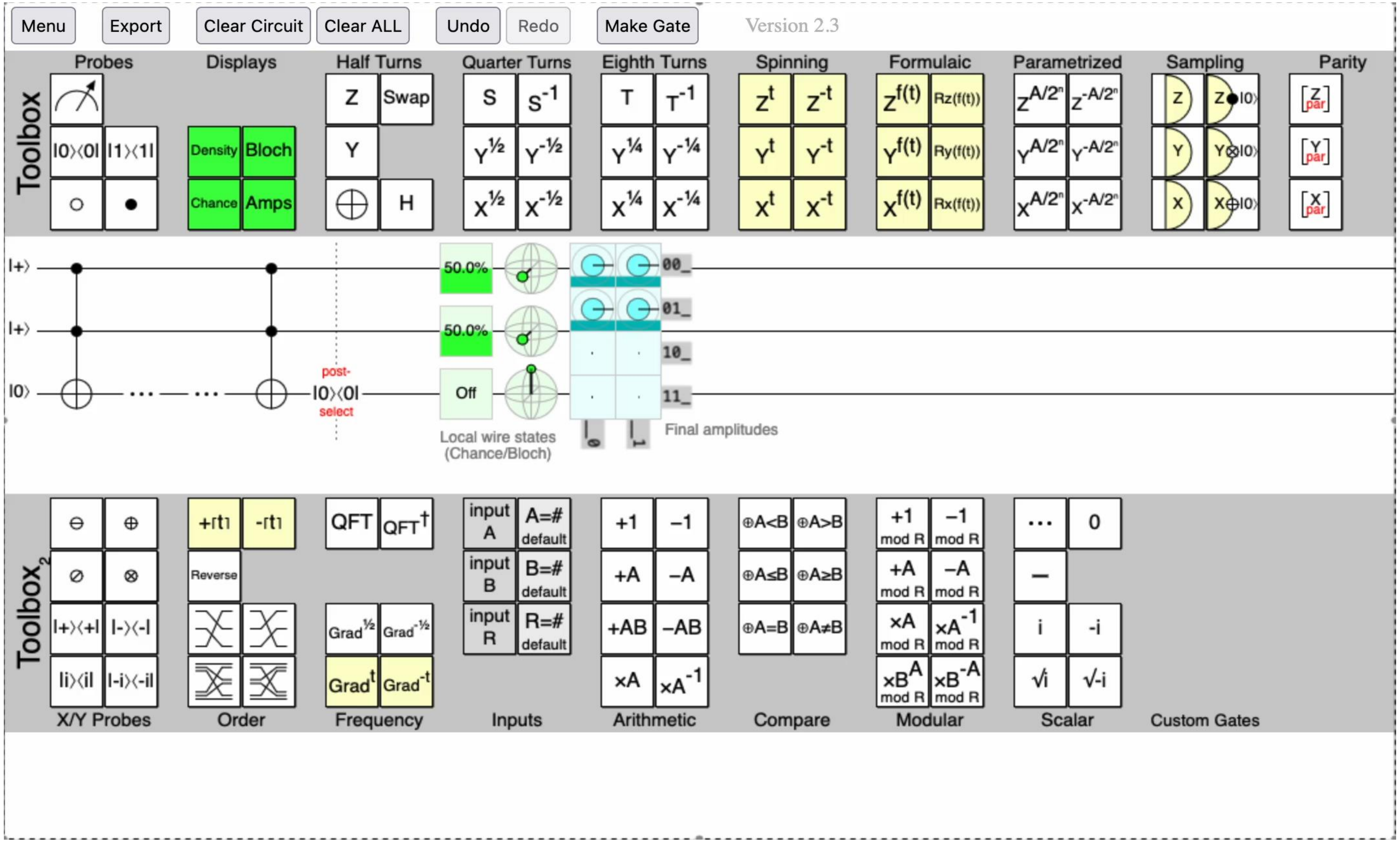
Last qubit



Improved Cuccaro adder

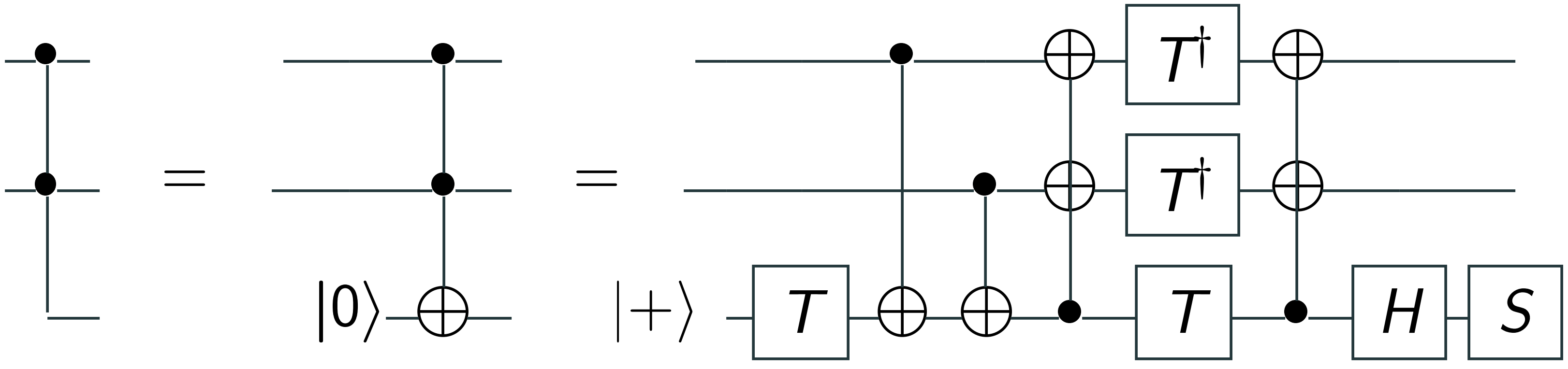


Measurement-based uncomputation

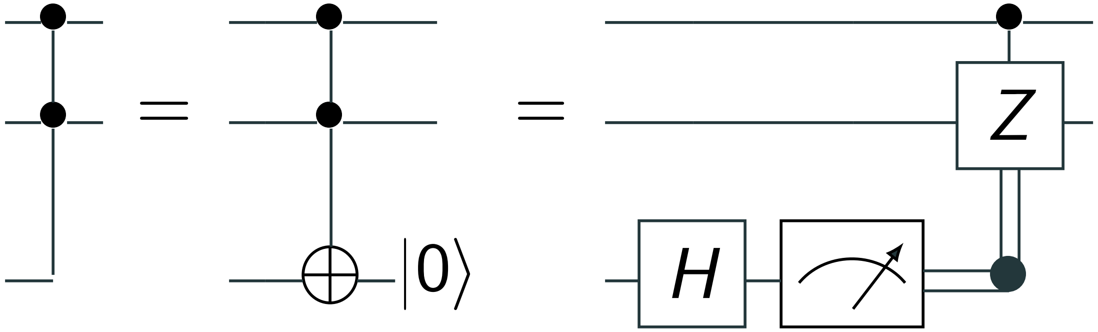


And operation

And: not Clifford



And†: Clifford



Craig Gidney, arXiv:1709.06648

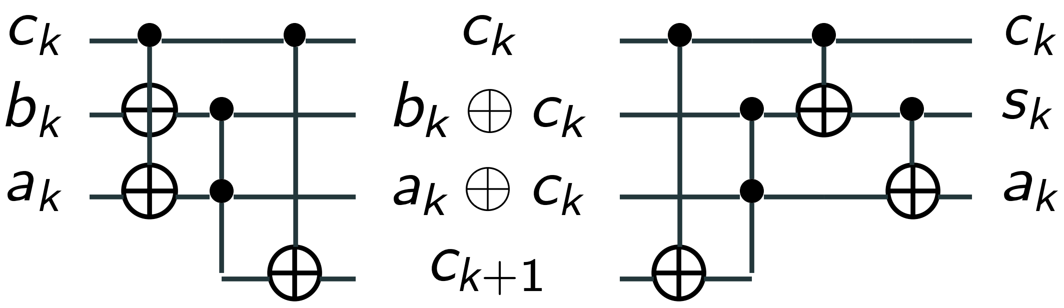
Gidney's adder elementary bloc

Alternative factorization

$$c_{k+1} = a_k b_k \oplus b_k c_k \oplus a_k c_k = (a_k \oplus c_k)(b_k \oplus c_k) \oplus c_k$$

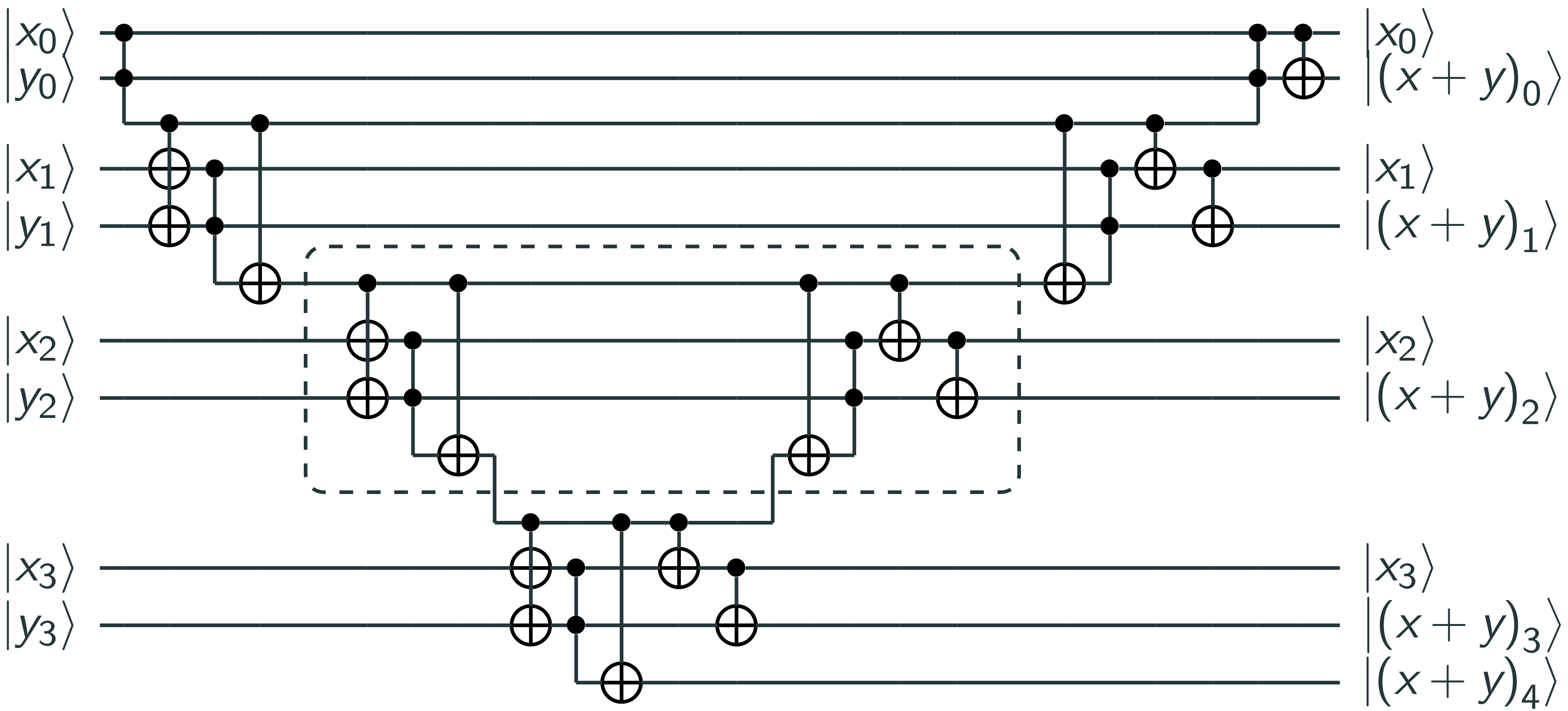
$$s_k = a_k \oplus b_k \oplus c_k$$

3-bit adder

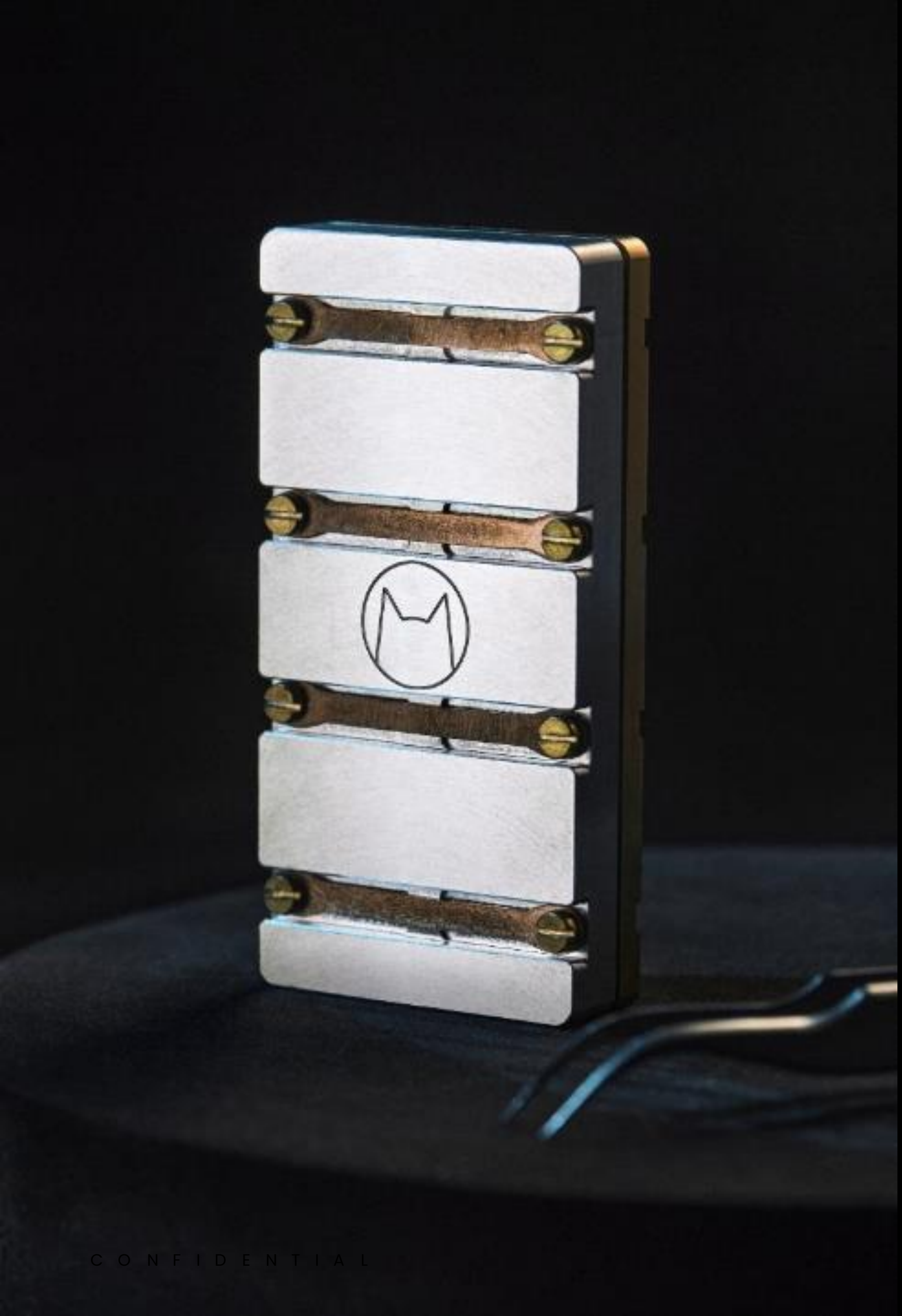


Craig Gidney, arXiv:1709.06648

Gidney's adder



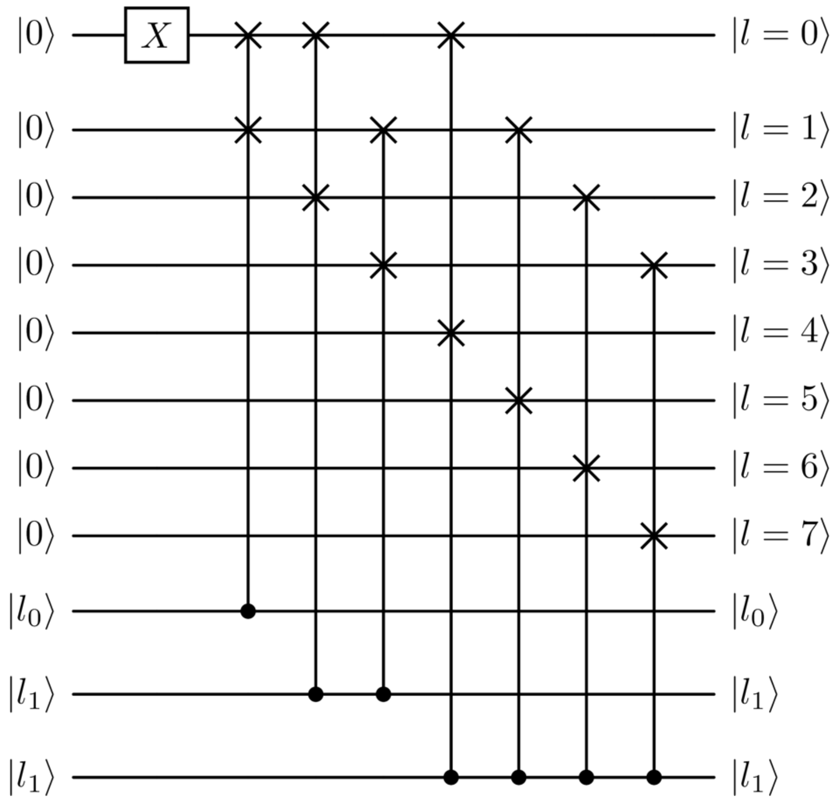
Craig Gidney, arXiv:1709.06648



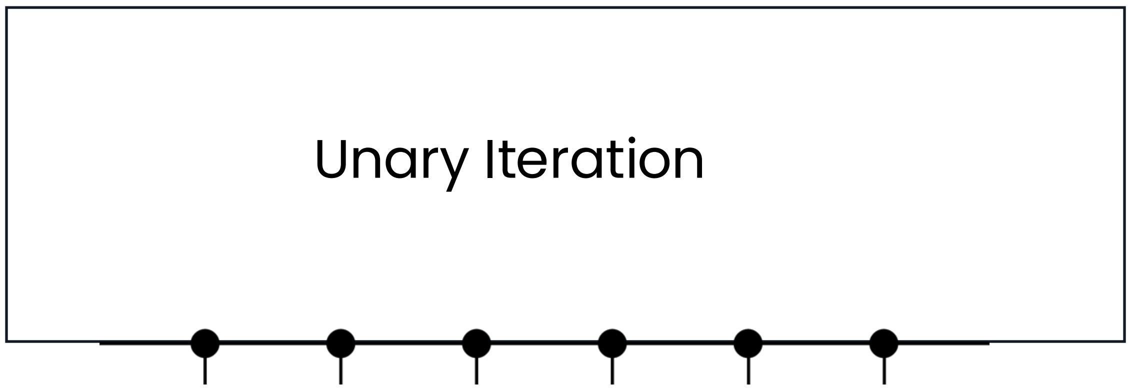
Time for hands-on workshop

Unary representation vs unary iteration

Unary representation

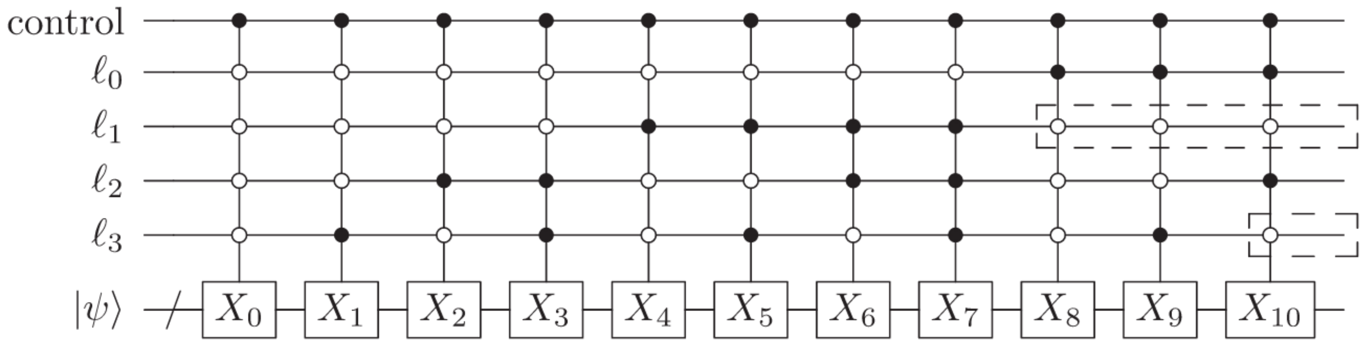


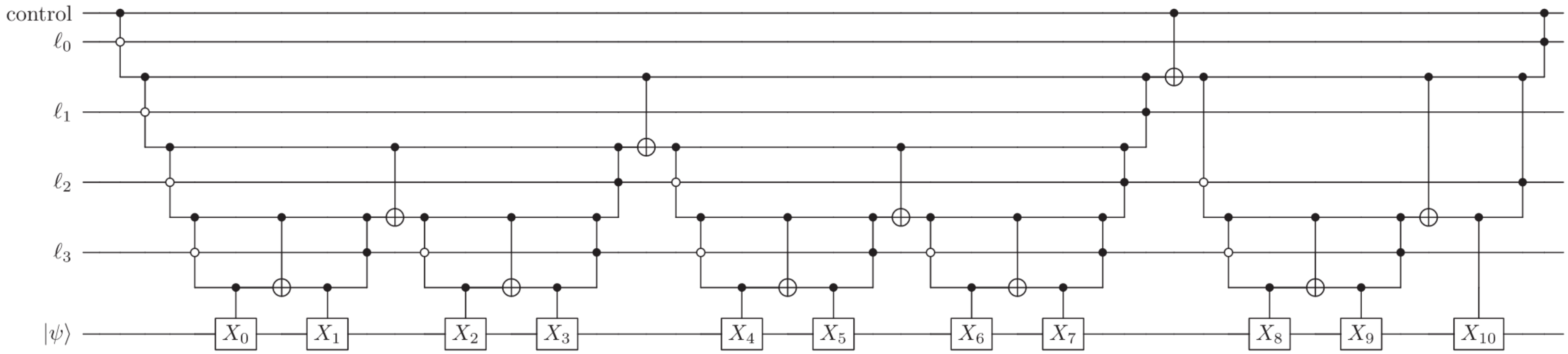
Space encoding



Time encoding

Unary iteration

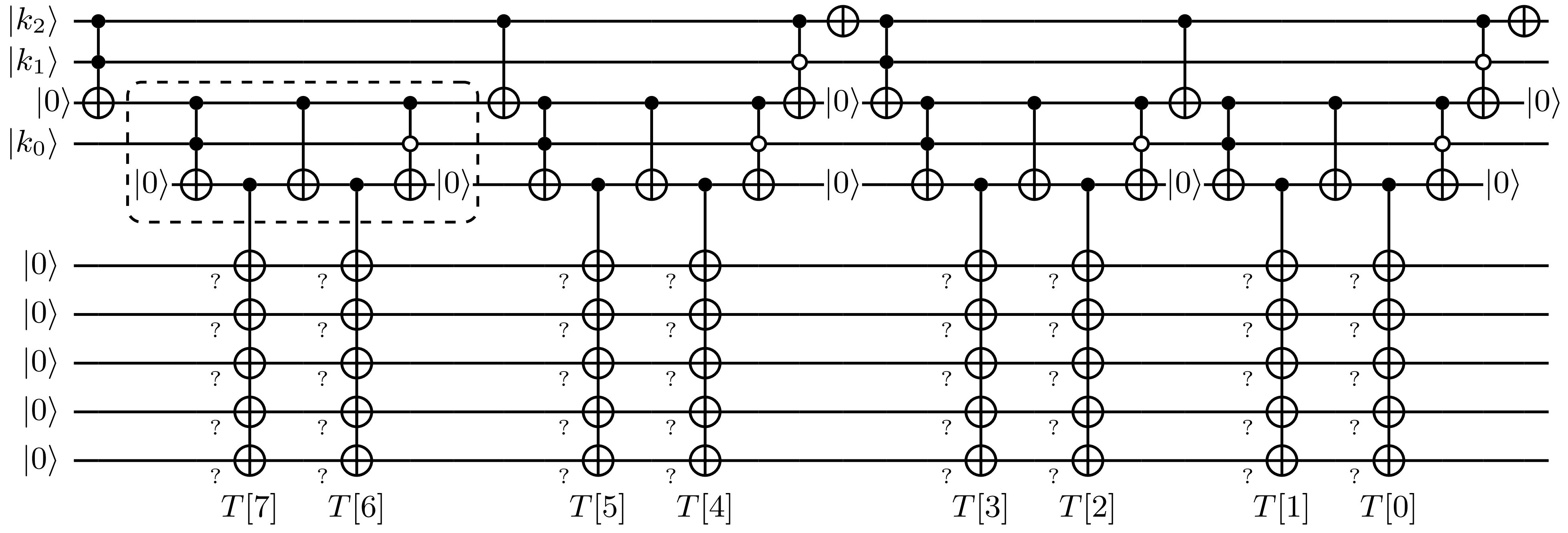




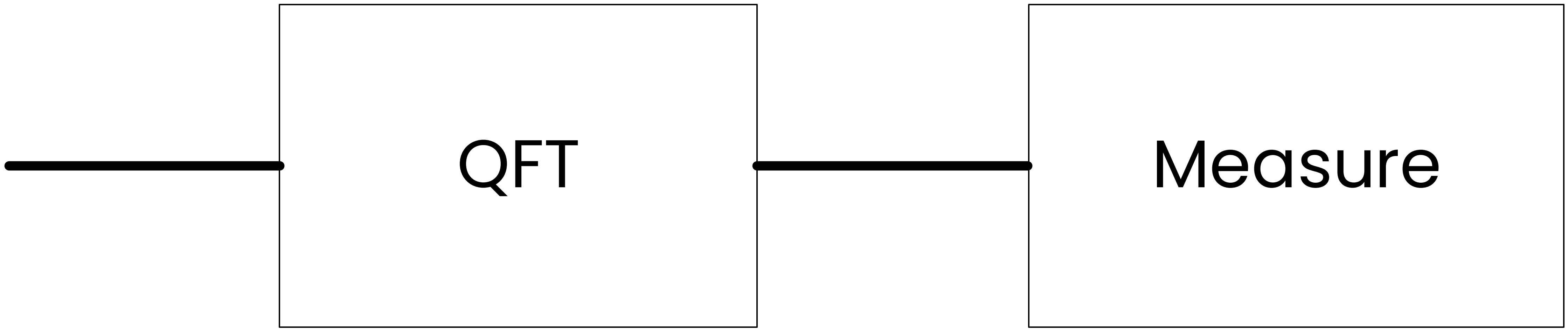
Babbush *et al.*, arXiv:1805.03662

Table lookup

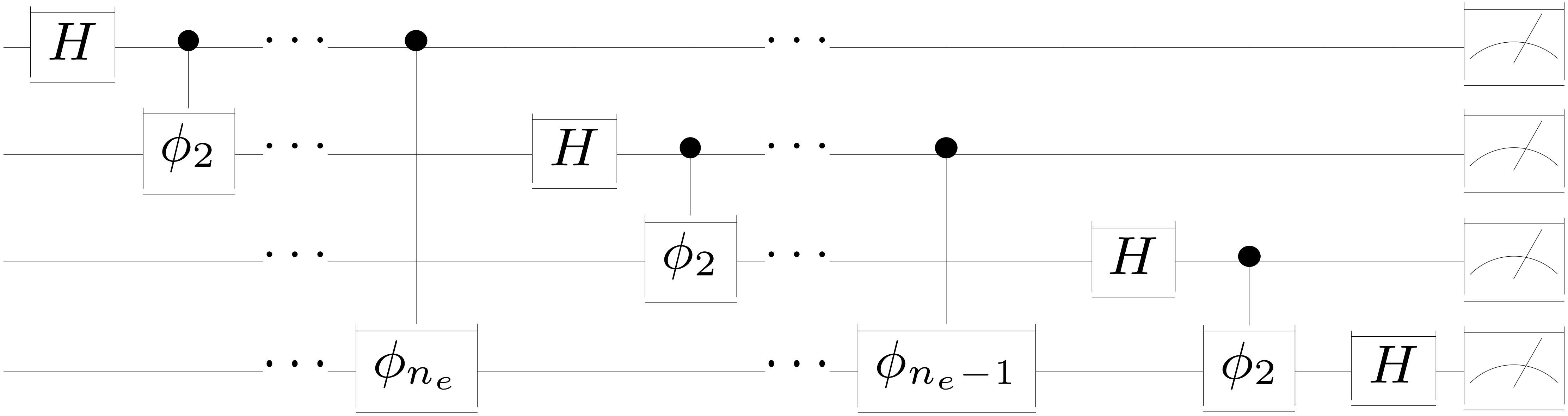
Table lookup: $|i\rangle \mapsto |i\rangle|T[i]\rangle$



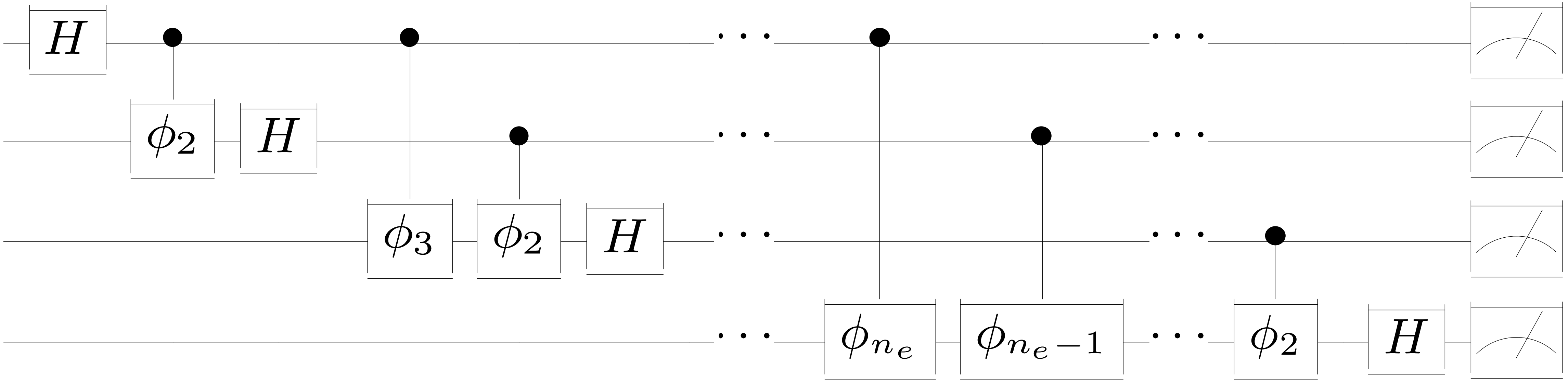
Semi-classical Fourier transform



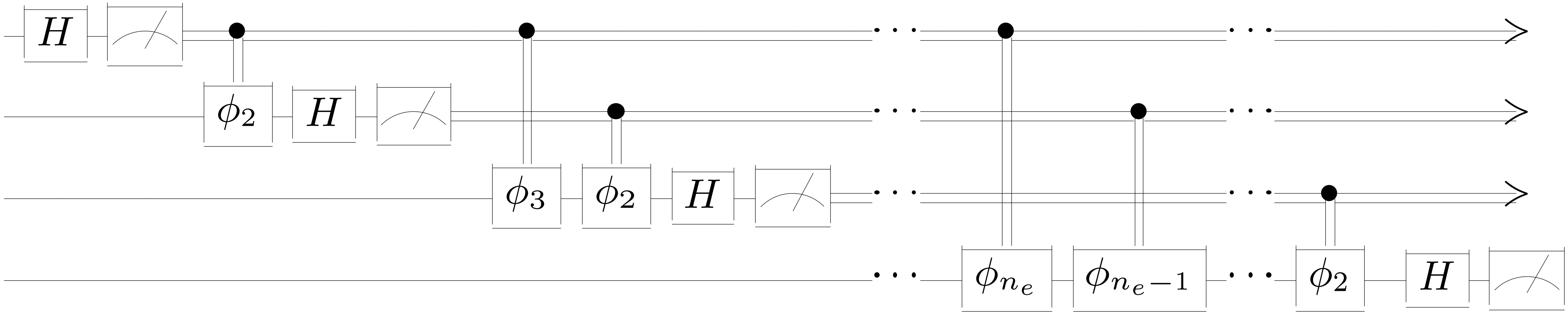
Semi-classical Fourier transform



Semi-classical Fourier transform



Semi-classical Fourier transform



Conclusion

Key takeaways

- Bennett's trick: any classical computing can be done, but massive cost
- You always handle worse case:
  - ~~Check before~~ → do and undo
  - Loops can't return early
- Ways around: approximate and optimistic quantum algorithm
- Offload everything you can to the classical computer

- Time-space tradeoff at algorithm level: to uncompute or not: pebbling game
- Classical data can be loaded by choice of gates

- Quantum computers are not unitary: measurement-based uncomputing, superpostition masking, etc.
- Non-Clifford gates are still costly, and the ultimate limit
- Can we do better than classical + QFT?
- Don't overoptimize circuits: not adapted for fault-tolerant

ALICE & BOB