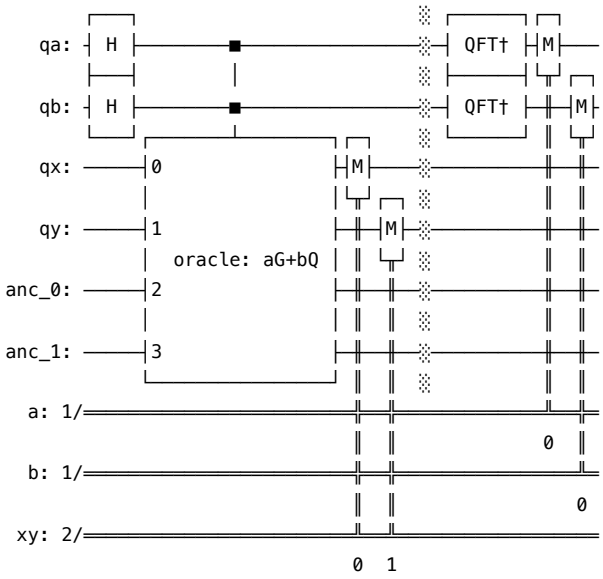
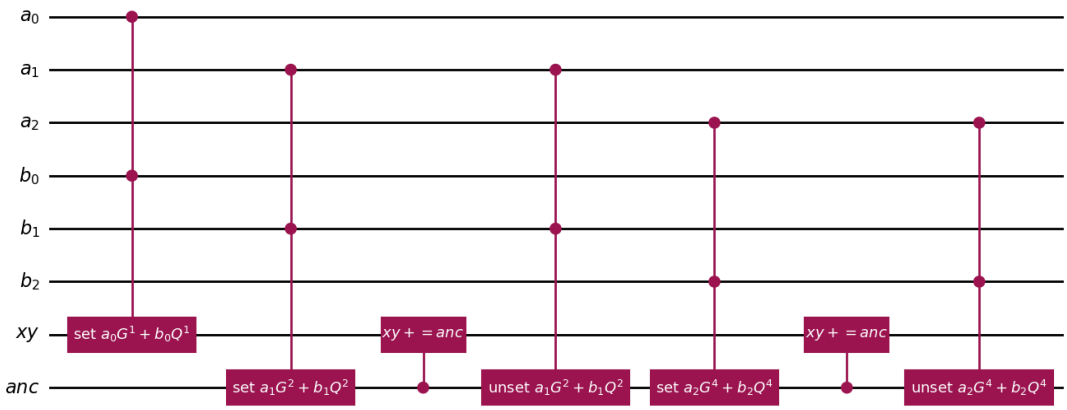
Details of the Quantum Circuit

The overall structure of the quantum circuit utilizing Shor's algorithm is shown below.

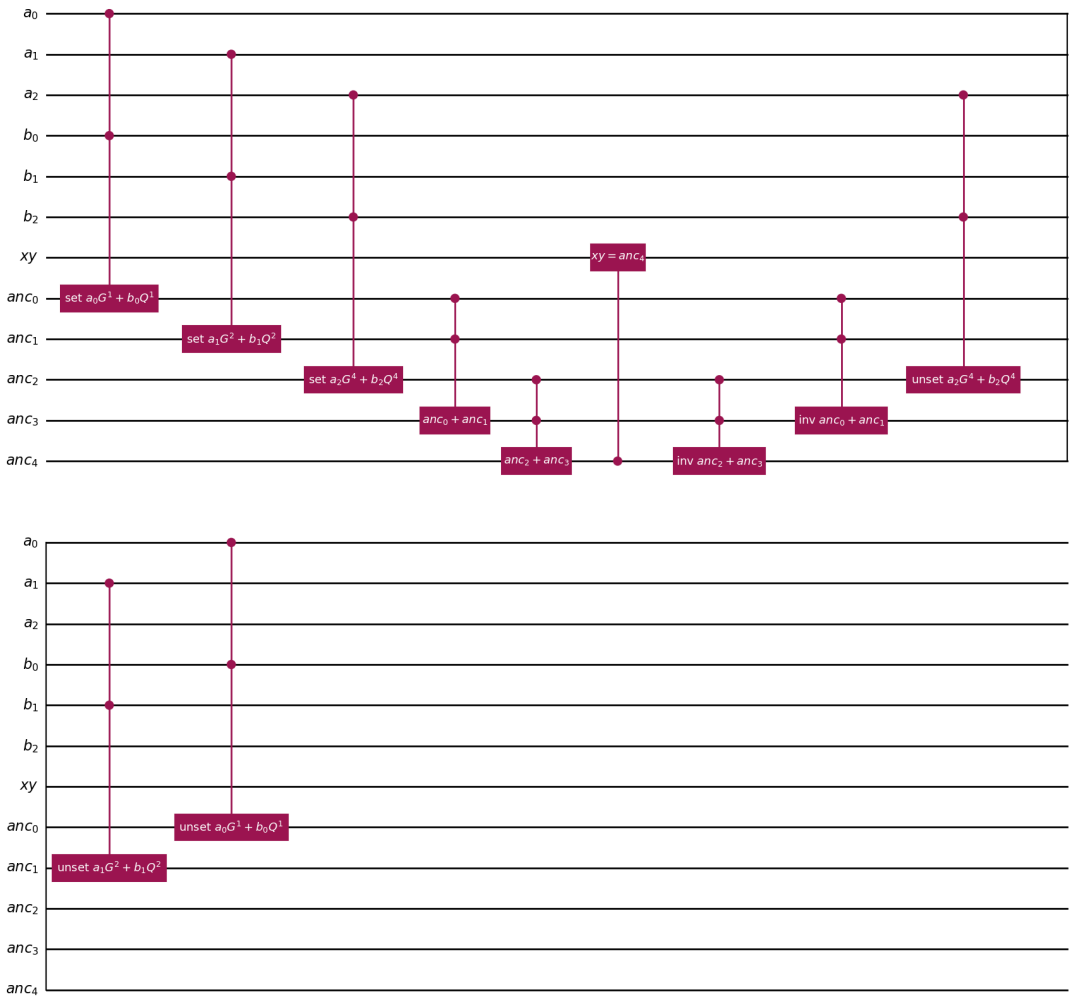


The core of this quantum circuit is the $aG + bQ$ operation. There are two implementations: a "compact" version, which reduces the number of qubits at the cost of deeper circuits, and a "wide" version, which uses more qubits but shallower circuits.

- **Compact version**
  - By uncomputing after each coordinate addition, the number of ancilla qubits is minimized for each ECC addition.
  - Effective for simulators where the number of qubits affects memory usage.



- **Wide version**
  - All ECC additions are performed first, then the result is copied and uncomputed at the end.
  - Requires as many qubits as the number of ECC additions, but is effective for real quantum hardware.



The set operation $a_iG^{2^i} + b_iQ^{2^i}$ and the ECC addition ($y^2 = x^3 + ax + b \mod p$) used in the above circuit are described below.

Quantum Circuit: set $a_iG^{2^i} + b_iQ^{2^i}$

$aG + bQ$ can be decomposed into bitwise additions as $\sum_{i=0}^{n}(a_iG^{2^i} + b_iQ^{2^i})$.
For each bit, the addition result is one of four values depending on the combination of $|b_ia_i\rangle$: $|00\rangle\, O + |01\rangle\, G^{2^i} + |10\rangle\, Q^{2^i} + |11\rangle\, (G^{2^i} + Q^{2^i})$.
$G^{2^i}, Q^{2^i}, G^{2^i} + Q^{2^i}$ are precomputed classically, and the quantum circuit sets the result to the target register using b_i, a_i as control bits.
This reduces the number of ECC additions required in the quantum circuit.

$$|a_i\rangle\,|b_i\rangle\,|0\rangle \mapsto \begin{cases} |0\rangle\,|0\rangle\,|0\rangle\,, & \text{if}(a_i = 0, b_i = 0) \\ |1\rangle\,|0\rangle\,|G^{2^i}\rangle\,, & \text{if}(a_i = 1, b_i = 0) \\ |0\rangle\,|1\rangle\,|Q^{2^i}\rangle\,, & \text{if}(a_i = 0, b_i = 1) \\ |1\rangle\,|1\rangle\,|G^{2^i} + Q^{2^i}\rangle\,, & \text{if}(a_i = 1, b_i = 1) \end{cases}$$

Quantum Circuit: ECC Addition

ECC addition is a very complex quantum circuit.
The addition $|xx\rangle + |yy\rangle \mapsto |zz\rangle$ is realized with the following quantum register structure:

$$|xx\rangle\,|yy\rangle\,|zz\rangle\,|ancilla\rangle$$

Each quantum register is further decomposed as follows:

$$\begin{aligned}|xx\rangle &= |y_1, x_1\rangle \\ |yy\rangle &= |y_2, x_2\rangle \\ |zz\rangle &= |y_3, x_3\rangle \\ |ancilla\rangle &= |f_1, f_2, f_3, f_4, carry, \lambda, a_1, a_2\rangle\end{aligned}$$

Details of ancilla:

- f_1: Flag for $(x_1, y_1) = O$
- f_2: Flag for $(x_2, y_2) = O$
- f_3: Flag for $(x_1, y_1) + (x_2, y_2) = O$
- f_4: Flag for $(x_1, y_1) = (x_2, y_2)$ (addition of identical coordinates)
- $carry$: Carry bit used in various calculations
- λ: Quantum register for $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ or $\frac{3x_1^2 + a}{2y_1}$ during ECC addition
- a_1: Used for Y coordinate calculation in the judgment $(x_1, y_1) + (x_2, y_2) = O$
- a_2: Quantum register for modular inverse calculation in λ denominator

In Version 3, by efficiently using these registers, the number of quantum gates was greatly reduced.
The detailed calculation process for ECC addition is as follows:

Target Register	Control Bits	Quantum Operation	Value After	Remarks
f_1	y_1, x_1	$\|0\rangle \mapsto \|1\rangle, \text{if}(x_1 = 0, y_1 = 0)$	$\begin{cases}\|1\rangle, & \text{if}(x_1 = 0, y_1 = 0) \\ \|0\rangle, & \text{otherwise}\end{cases}$	Set flag for $(x_1, y_1) = O$
f_2	y_2, x_2	$\|0\rangle \mapsto \|1\rangle, \text{if}(x_2 = 0, y_2 = 0)$	$\begin{cases}\|1\rangle, & \text{if}(x_2 = 0, y_2 = 0) \\ \|0\rangle, & \text{otherwise}\end{cases}$	Set flag for $(x_2, y_2) = O$
x_3	f_1, f_2, x_1, x_2	$\|0\rangle \mapsto \begin{cases}\|x_2\rangle, & \text{if}(f_1 = 1) \\ \|x_1\rangle, & \text{if}(f_2 = 1)\end{cases}$	$\begin{cases}\|x_2\rangle, & \text{if}(f_1 = 1) \\ \|x_1\rangle, & \text{if}(f_2 = 1) \\ \|0\rangle, & \text{otherwise}\end{cases}$	Set x_3 if either (x_1, y_1) or (x_2, y_2) is O
y_3	f_1, f_2, y_1, y_2	$\|0\rangle \mapsto \begin{cases}\|y_2\rangle, & \text{if}(f_1 = 1) \\ \|y_1\rangle, & \text{if}(f_2 = 1)\end{cases}$	$\begin{cases}\|y_2\rangle, & \text{if}(f_1 = 1) \\ \|y_1\rangle, & \text{if}(f_2 = 1) \\ \|0\rangle, & \text{otherwise}\end{cases}$	Set y_3 if either (x_1, y_1) or (x_2, y_2) is O
x_3	f_1, x_2	$\|0\rangle \mapsto \|-x_2 \mod p\rangle, \text{if}(f_1 = 0)$	$\begin{cases}\|x_2\rangle, & \text{if}(f_1 = 1) \\ \|x_1\rangle, & \text{if}(f_2 = 1) \\ \|-x_2 \mod p\rangle, & \text{otherwise}\end{cases}$	Compute $-x_2$ term in $x_3 = \lambda^2 - x_1 - x_2$
x_2	x_1	$\|x_2\rangle \mapsto \|x_2 - x_1 \mod p\rangle$	$\|x_2 - x_1 \mod p\rangle$	Compute denominator of λ

Target Register	Control Bits	Quantum Operation	Value After	Remarks
a_1	y_1, y_2	$\|0\rangle \mapsto \|y_1 + y_2 - p\rangle$	$\|y_1 + y_2 - p\rangle$	For checking $(x_1, y_1) + (x_2, y_2) = O$
y_2	y_1	$\|y_2\rangle \mapsto \|y_2 - y_1 \mod p\rangle$	$\|y_2 - y_1 \mod p\rangle$	Compute numerator of λ
f_3	x_2, a_1	$\|0\rangle \mapsto \|1\rangle, \text{if}(x_2 = 0, a_1 = 0)$	$\begin{cases} \|1\rangle, & \text{if}(x_1 = x_2, y_1 + y_2 = p) \\ \|0\rangle, & \text{otherwise} \end{cases}$	Set flag for $(x_1, y_1) + (x_2, y_2) = O$
f_4	x_2, y_2	$\|0\rangle \mapsto \|1\rangle, \text{if}(x_2 = 0, y_2 = 0)$	$\begin{cases} \|1\rangle, & \text{if}(x_1 = x_2, y_1 = y_2) \\ \|0\rangle, & \text{otherwise} \end{cases}$	Set flag for identical coordinates
x_3	f_3, x_1	$\|x_3\rangle \mapsto \|x_3 + x_1\rangle, \text{if}(f_3 = 1)$	$\begin{cases} \|x_2\rangle, & \text{if}(f_1 = 1) \\ \|x_1\rangle, & \text{if}(f_2 = 1) \\ \|0\rangle, & \text{if}(f_3 = 1) \\ \|-x_2 \mod p\rangle, & \text{otherwise} \end{cases}$	Set $x_3 = 0$ if $(x_1, y_1) + (x_2, y_2) = O$
y_2	f_4, x_1	$\|0\rangle \mapsto \|3x_1^2 + a \mod p\rangle, \text{if}(f_4 = 1)$	$\begin{cases} \|3x_1^2 + a \mod p\rangle, & \text{if}(f_4 = 1) \\ \|y_2 - y_1 \mod p\rangle, & \text{otherwise} \end{cases}$	Numerator of λ for identical coordinates
x_2	f_4, y_1	$\|0\rangle \mapsto \|2y_1 \mod p\rangle, \text{if}(f_4 = 1)$	$\begin{cases} \|2y_1 \mod p\rangle, & \text{if}(f_4 = 1) \\ \|x_2 - x_1 \mod p\rangle, & \text{otherwise} \end{cases}$	Denominator of λ for identical coordinates
a_2	x_2	$\|0\rangle \mapsto \|x_2^{p-2} \mod p\rangle \|junk\rangle$	$\begin{cases} \|(2y_1)^{-1} \mod p\rangle \|junk\rangle, & \text{if}(f_4 = 1) \\ \|(x_2 - x_1)^{-1} \mod p\rangle \|junk\rangle, & \text{otherwise} \end{cases}$	Compute modular inverse for denominator of λ
λ	y_2, a_2	$\|0\rangle \mapsto \|y_2 \cdot a_2 \mod p\rangle$	$\begin{cases} \|(3x_1^2 + a) \cdot (2y_1)^{-1} \mod p\rangle, & \text{if}(f_4 = 1) \\ \|(y_2 - y_1) \cdot (x_2 - x_1)^{-1} \mod p\rangle, & \text{otherwise} \end{cases}$	Compute λ
x_3	f_1, f_2, f_3, x_1	$\|x_3\rangle \mapsto \|x_3 + \lambda^2 - x_1 \mod p\rangle, \text{if}(f_1 = 0, f_2 = 0, f_3 = 0)$	$\begin{cases} \|x_2\rangle, & \text{if}(f_1 = 1) \\ \|x_1\rangle, & \text{if}(f_2 = 1) \\ \|0\rangle, & \text{if}(f_3 = 1) \\ \|\lambda^2 - x_2 - x_1 \mod p\rangle, & \text{otherwise} \end{cases}$	Final calculation of x_3
x_1	x_3	$\|x_1\rangle \mapsto \|x_1 - x_3 \mod p\rangle$	$\|x_1 - x_3 \mod p\rangle$	Prepare for y_3 calculation
y_3	$f_1, f_2, f_3, x_1, y_1, \lambda$	$\|0\rangle \mapsto \|\lambda \cdot x_1 - y_1 \mod p\rangle, \text{if}(f_1 = 0, f_2 = 0, f_3 = 0)$	$\begin{cases} \|y_2\rangle, & \text{if}(f_1 = 1) \\ \|y_1\rangle, & \text{if}(f_2 = 1) \\ \|0\rangle, & \text{if}(f_3 = 1) \\ \|\lambda(x_1 - x_3) - y_1 \mod p\rangle, & \text{otherwise} \end{cases}$	Final calculation of y_3

After obtaining the result, the ECC addition circuit must be uncomputed to return all ancilla to $|0\rangle$.
In the wide version, uncomputation is performed after all $aG + bQ$ calculations, requiring extra qubits for each addition.
In the compact version, uncomputation is performed after each addition, allowing efficient reuse of qubits.

Furthermore, in the compact version, the addition is performed as $|xx\rangle\,|yy\rangle \mapsto |xx + yy\rangle\,|yy\rangle$, and ancilla bits are returned to $|0\rangle$ as follows:

1. $|xx\rangle\,|yy\rangle\,|0\rangle\,|0\rangle$
2. $|xx\rangle\,|yy\rangle\,|0\rangle\,|xx + yy, junk\rangle$: ECC addition
3. $|xx\rangle\,|yy\rangle\,|xx + yy\rangle\,|xx + yy, junk\rangle$: Copy result to third register
4. $|xx\rangle\,|yy\rangle\,|xx + yy\rangle\,|0\rangle$: Uncompute ECC addition to return fourth register to $|0\rangle$
5. $|xx + yy\rangle\,|yy\rangle\,|xx\rangle\,|0\rangle$: SWAP $|xx\rangle$ and $|xx + yy\rangle$
6. $|xx + yy\rangle\,|-yy\rangle\,|xx\rangle\,|0\rangle$: Prepare for subtraction by mapping $|yy\rangle \mapsto |-yy\rangle$
7. $|xx + yy\rangle\,|-yy\rangle\,|xx\rangle\,|xx, junk\rangle$: ECC addition to compute $|xx\rangle$
8. $|xx + yy\rangle\,|-yy\rangle\,|0\rangle\,|xx, junk\rangle$: Return third register to $|0\rangle$
9. $|xx + yy\rangle\,|-yy\rangle\,|0\rangle\,|0\rangle$: Uncompute ECC addition to return fourth register to $|0\rangle$
10. $|xx + yy\rangle\,|yy\rangle\,|0\rangle\,|0\rangle$: Restore $|-yy\rangle$ to original

Sub-Quantum Circuits

Various sub-quantum circuits are prepared for ECC addition.

- $|x\rangle_n \mapsto |x + 1\rangle$
  - Increment circuit for arbitrary bit width. This is the basis for all calculations.
- $|x\rangle_n \mapsto |x + a\rangle$
  - Addition of a constant. Realized by combining increment circuits.
- $|x\rangle_n\,|y\rangle_m \mapsto |x\rangle\,|y + x\rangle$
  - Addition between quantum registers. Realized by combining controlled increment circuits.
- $|x\rangle_{n+1} \mapsto |x + a \mod p\rangle$
  - Modular addition of a constant. Implemented so that the carry bit returns to $|0\rangle$.
- $|x\rangle_n\,|y\rangle_{n+1} \mapsto |x\rangle\,|y + x \mod p\rangle$
  - Modular addition between quantum registers. Implemented so that the carry bit returns to $|0\rangle$.
- $|x\rangle_{n+1} \mapsto |-x \mod p\rangle$
  - Modular additive inverse of the quantum register itself.
- $|x\rangle_n\,|0\rangle_n \mapsto |x\rangle\,|-x \mod p\rangle$
  - Set modular additive inverse to another quantum register.
- $|x\rangle_n\,|0\rangle_{n+1} \mapsto |x\rangle\,|x^2 \mod p\rangle$
  - Modular squaring. This is realized by, for all i, j, using controlled operations on $|x_i\rangle$, $|x_j\rangle$ and adding the classically precomputed constant $2^{i+j} \mod p$ to the target register. (The same approach is used for subsequent multiplications.)
- $|x\rangle_n\,|y\rangle_{n+1} \mapsto |x\rangle\,|y + x^2 \mod p\rangle$
  - Modular squaring and addition at the same time.
- $|x\rangle_n\,|0\rangle_{n+1} \mapsto |x\rangle\,|a \cdot x^2 \mod p\rangle$
  - Modular squaring and multiplication by a constant at the same time.
- $|x\rangle_n\,|y\rangle_{n+1} \mapsto |x\rangle\,|y + a \cdot x^2 \mod p\rangle$
  - Modular squaring, multiplication by a constant, and addition at the same time.
- $|x\rangle_n\,|0\rangle_{n+1} \mapsto |x\rangle\,|a \cdot x \mod p\rangle$
  - Modular multiplication by a constant.
- $|x\rangle_n\,|y\rangle_{n+1} \mapsto |x\rangle\,|y + a \cdot x \mod p\rangle$
  - Modular multiplication by a constant and addition at the same time.
- $|x\rangle_n\,|y\rangle_n\,|0\rangle_{n+1} \mapsto |x\rangle\,|y\rangle\,|x \cdot y \mod n\rangle$
  - Modular multiplication between quantum registers.
- $|x\rangle_n\,|y\rangle_n\,|z\rangle_{n+1} \mapsto |x\rangle\,|y\rangle\,|z + x \cdot y \mod n\rangle$
  - Modular multiplication and addition at the same time.
- $|x\rangle_n\,|0\rangle_n\,|0\rangle_{m \cdot n+1} \mapsto |x\rangle\,|x^a\rangle\,|junk\rangle$
  - Modular exponentiation by a constant. Realized by combining modular squaring and modular multiplication. Used for modular inverse calculation.

Even for similar calculations, the quantum circuit differs between constant operations and register-to-register operations.
Also, the following two circuits yield the same result when $|y\rangle = |0\rangle$, so only the latter is strictly necessary, but the former can be implemented with fewer quantum gates and is used as appropriate.

- $|x\rangle_n |0\rangle_{n+1} \mapsto |x\rangle |a \cdot x \mod p\rangle$
- $|x\rangle_n |y\rangle_{n+1} \mapsto |x\rangle |y + a \cdot x \mod p\rangle$

Scale of the Quantum Circuit

The following table shows the number of qubits, quantum gates, and circuit depth for each ECC bit size.
The number of quantum gates is counted as one for each MCX or SWAP gate, so the actual number after transpilation is much larger, but this is sufficient to represent the scale of the circuit (especially for simulator execution time).

ecc bits	qbits (compact)	gates (compact)	depth (compact)	qbits (wide)	gates (wide)	depth (wide)
3	50	5,100	4,563	82	3,388	3,026
4	75	19,514	18,720	128	12,992	12,467
6	123	165,008	162,005	390	94,214	69,363
7	145	479,286	473,012	660	261,272	128,935
8	181	1,102,960	1,090,685	979	593,664	251,620
9	239	2,510,080	2,497,470	1534	1,338,402	665,831
10	245	3,322,938	3,305,080	1725	1,758,806	777,520
11	291	6,937,706	6,901,693	2316	3,650,848	1,452,787
12	341	9,232,912	9,203,552	3031	4,835,708	1,752,882

As shown above, the compact version increases the number of qubits gradually, while the wide version increases rapidly.
However, the wide version allows more parallel execution, so as the ECC bit size increases, the circuit depth increases more slowly compared to the number of gates.

For actual quantum computers, the wide version is more effective, but with the current error rates, it is difficult to execute even 3-bit ECC without errors.