El algoritmo de Shor — Parte 2

Computación Cuántica: Algorítmica y Software

Joaquín Keller

2026 UCV

Factorizar N en factores primos

El algoritmo de Shor permite conseguir un factor no trivial de N

Descomposición en factores primos:

0. Utilizar un test de primalidad, si N es primo el resultado es N, y sino
1. Utilizar Shor para conseguir un factor d de N con $1 < d < N$
2. Reiterar con d y N/d

Conseguir un factor de N (3)

1. Tomar $a < N$ al azar
   Si $mcd(a, N) \neq 1$ | Factor de N: $mcd(a, N)$
2. Calcular $r = ord_N(a)$
   Si r es impar $\longrightarrow$ 1.
   Si $a^{r/2} + 1 \equiv 0 \pmod{N} \longrightarrow$ 1.
3. $mcd(a^{r/2} - 1, N)$ y $mcd(a^{r/2} + 1, N)$

Descomposición en factores primos: $N = \prod_{i=1}^{k} p_i^{\alpha_i}$ con $p_i > 2$

Probabilidad de r par y $a^{r/2} + 1 \not\equiv 0 \pmod{N}$ es $1 - \frac{1}{2^{k-1}}$

Verificar que $N \neq d^k$

El número de dígitos binarios de N es $\lfloor \log_2 N \rfloor + 1$

Es de notar que: $N = d^k \Leftrightarrow k = \log_d N \quad \leq \log_2 N$

Algoritmo: para $k = 2, 3, ..., \lfloor \log_2 N \rfloor$

1. Calcular una aproximación numérica de $N^{1/k}$

2. Tomar $d = \lfloor N^{1/k} \rfloor$ y verificar que $\quad d^k \neq N \quad$ y $\quad (d+1)^k \neq N$

Conseguir un factor de N

0. Condiciones iniciales sobre N

   Si N es primo — Factorización terminada

   Si N es par — 2 y $N/2$

   Si $N = d^k$ — $d, d^1, d^2, ..., d^{k-1}$

1. Tomar $a < N$ al azar

   Si $mcd(a, N) \neq 1$ — $mcd(a, N)$

2. Calcular $r = ord_N(a)$

   Si r es impar o $a^{r/2} + 1 \equiv 0 \pmod{N}$ regresar a 1.

   eso se produce con una probabilidad $\leq \frac{1}{2}$

3. $mcd(a^{r/2} - 1, N)$ y $mcd(a^{r/2} + 1, N)$

Que hace Peter Shor

1. Muestra que problema de factorización es equivalente a

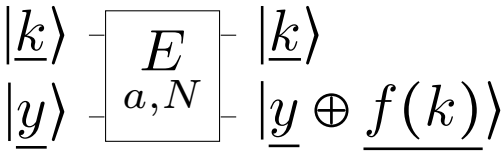
   Conseguir el periodo de una exponenciación modular

   Exponenciación modular: $f(k) = a^k \bmod N$ con $a < N$

2. Diseña un algoritmo cuántico de complejidad polinomial

   para conseguir ese periodo

Exponenciación modular cuántica (1)



$a < N \qquad f(k) = a^k \bmod N$

$E\left(|\underline{k}\rangle\,|\underline{y}\rangle\right) = |\underline{k}\rangle\,|\underline{y} \oplus \underline{f(k)}\rangle$

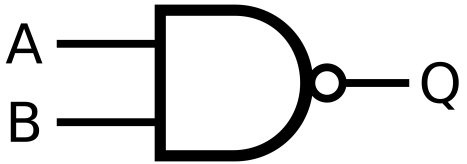
$\oplus$ es el XOR bit por bit

Circuito de $2 \times n$ qubits $\quad N \leq 2^n$

$$EE\left(|k\rangle\,|y\rangle\right) = E\left(|k\rangle\,|y \oplus f(k)\rangle\right) = |k\rangle\,|y \oplus f(k) \oplus f(k)\rangle = |k\rangle\,|y\rangle$$

$\Longrightarrow$ $E = E^{-1}$ es unitario

La compuerta NAND (clásica)



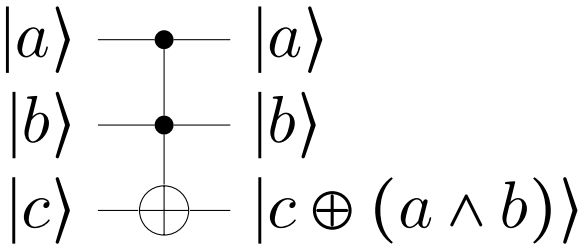
$$Q = \neg(A \wedge B)$$

NOT A = A NAND A

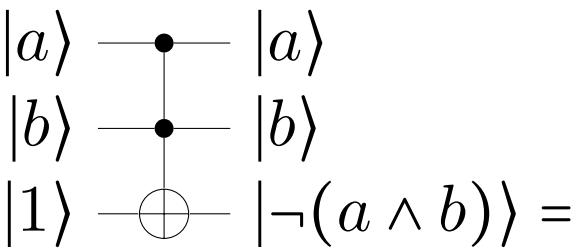
A AND B = NOT (A NAND B)

A OR B = (NOT A) NAND (NOT B)

Compuerta de Toffoli o CCNOT

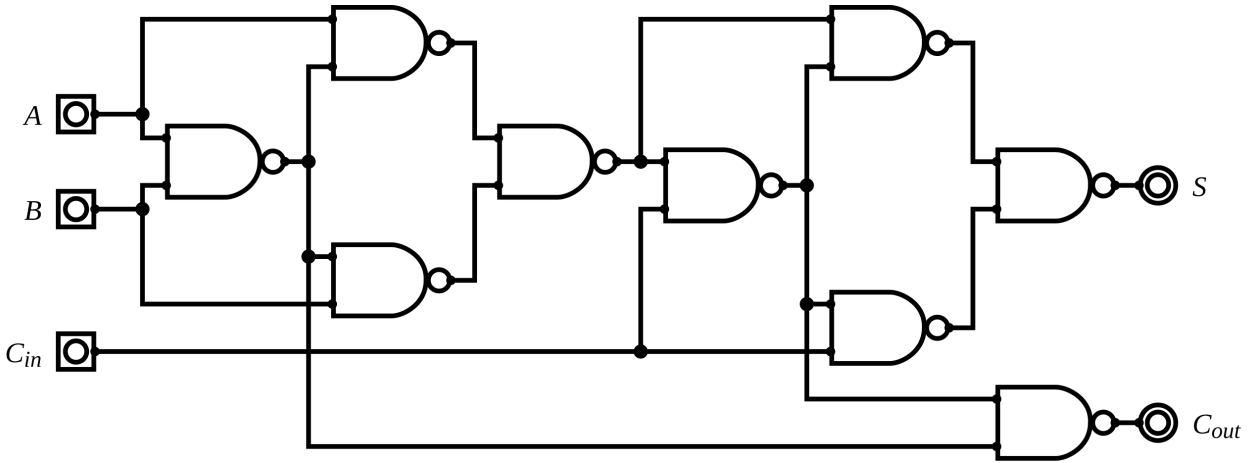


Si $|a\rangle = 1$ y $|b\rangle = 1$
el tercer qubit se inversa

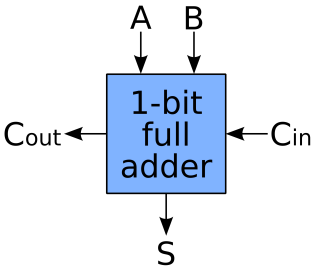


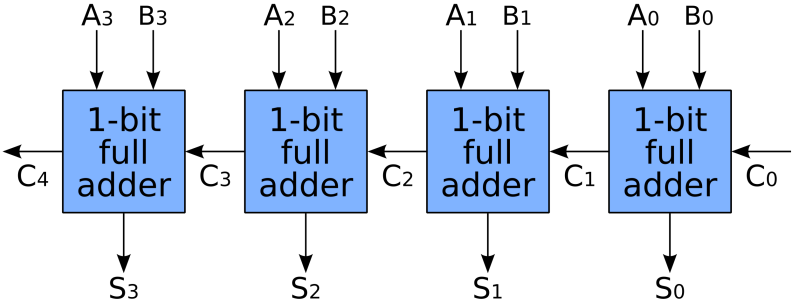
Compuerta NAND cuántica

Sumador completo clásico 1-bit

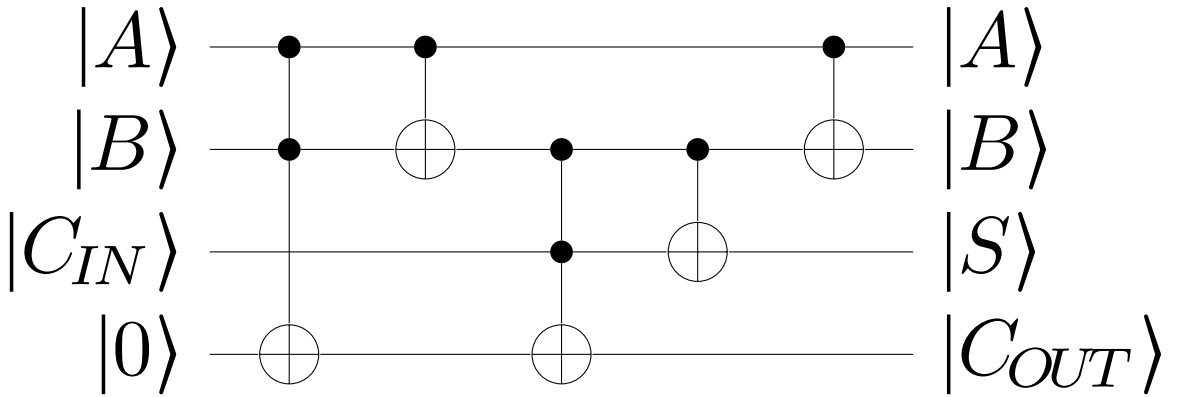


Sumador completo clásico n-bits





Sumador completo cuántico 1-bit



Computación teórica (informalmente)

Para toda función *Turing computable*: $f : \{0,1\}^n \to \{0,1\}^m$
existe un circuito C_n^m tal que: $C_n^m(x) = f(x) \quad \forall x \in \{0,1\}^n$

Si la complejidad algorítmica temporal de f es $\mathcal{O}(T_n^m)$
el tamaño del circuito C_n^m es $\mathcal{O}(T_n^m \textit{polylog} T_n^m)$

También existe un circuito cuántico Q_n^m que calcula f
con $|Q_n^m| = \mathcal{O}(|C_n^m|)$ y un número de qubits $\mathcal{O}(n + m + |C_n^m|)$

Algoritmo clásico de exponenciación módular

Para calcular $a^k \pmod N$ se escribe k en binario
y solo se calculan las potencias de dos correspondientes: a^{2^i}

Ejemplo $3^{21} \pmod{31}$

$21 = 16 + 4 + 1 \qquad 3^{21} \equiv 3^{16+4+1} \equiv 3^{16} \times 3^4 \times 3^1 \pmod{31}$

Podemos notar que: $a^{2^{i+1}} = a^{2 \times 2^i} = (a^{2^i})^2$

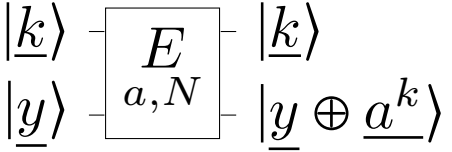
$3^1 \equiv 3 \pmod{31} \quad 3^2 \equiv 9 \pmod{31} \quad 3^4 \equiv 9^2 \equiv 81 \equiv 50 \equiv 19 \pmod{31}$

$3^8 \equiv 19^2 \equiv 361 \equiv 20 \pmod{31} \quad 3^{16} \equiv 20^2 \equiv 400 \equiv 28 \pmod{31}$

$3^{21} \equiv 3^{16} \times 3^4 \times 3^1 \equiv 28 \times 19 \times 3 \equiv 1596 \equiv 15 \pmod{31}$

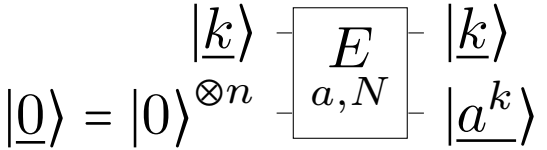
La complejidad algorítmica es polinomial en el número de digitos

Exponenciación modular cuántica (2)

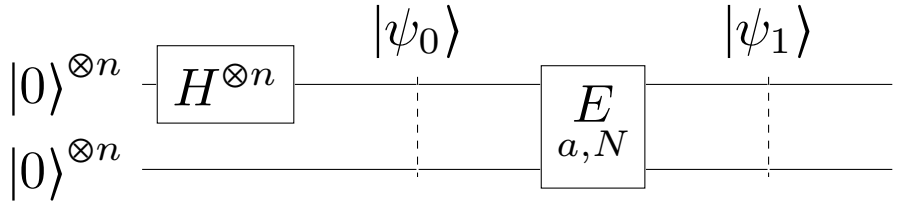


El número de compuertas de E es polinomial en n
n: número de dígitos de N

Si fijamos $y = 0$, obtenemos el circuito que calcula a^k mod N:



Circuito para el algoritmo de Shor



$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle \otimes |\underline{0}\rangle \qquad |\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (|k\rangle \otimes |a^k\rangle)$$

El circuito calcula todos los a^k mod N para $k \leq N$

Cómo conseguir el k que da $a^k \equiv 1 \pmod{N}$?