Shor's Factoring

Sources

Richard Spillman

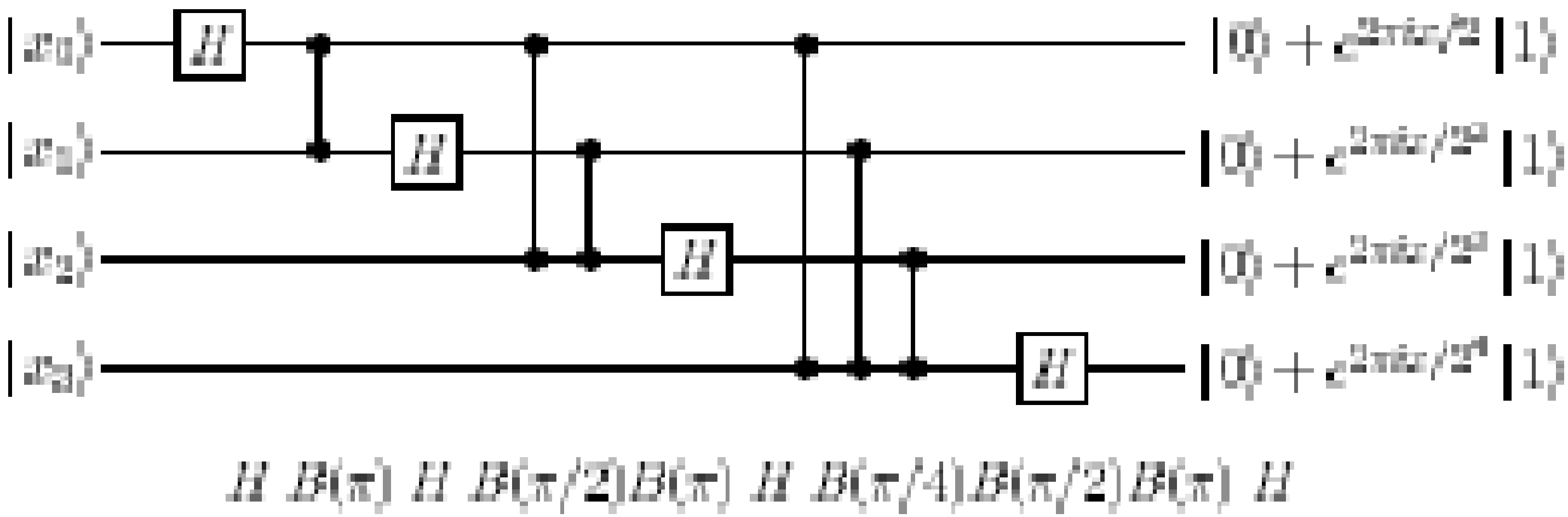
Mike Frank

Isaac Chuang

Quantum Circuits

Quantum Fourier Transform Circuit

$$|y\rangle = \frac{1}{2^{n/2}} \sum_x e^{i\frac{2\pi}{2n} y \cdot x} |x\rangle$$



Shor's Factoring Algorithm

- Solves the >2000-year-old problem:
  - Given a large number *N*, quickly find the prime factorization of *N*. (At least as old as Euclid.)
- No polynomial-time (as a function of $n = \lg N$) classical algorithm for this problem is known.
  - The best known (as of 1993) was a *number field sieve* algorithm taking time $\mathbf{O}(\exp(n^{1/3} \log(n^{2/3})))$
  - However, there is also no proof that a fast classical algorithm does *not* exist.
- Shor's quantum algorithm takes time $\mathbf{O}(n^2)$
  - No worse than multiplication of *n*-bit numbers!

More Details of Shor's Algorithm

- Uses a standard reduction of factoring to another number-theory problem called the *discrete logarithm* problem.
- The discrete logarithm problem corresponds to finding the *period* of a certain periodic function defined over the integers.
- A general way to find the period of a function is to perform a *Fourier transform* on the function.
  - Shor showed how to generalize an earlier algorithm by **Simon**, to provide a **Quantum Fourier Transform** that is exponentially faster than classical ones.

Main Idea: Factoring

- Given two large prime numbers ***p*** and ***q*** it is easy to calculate their product
  - ***p*** = 15485863 and ***q*** = 15485867 then ***p*** x ***q*** = 239813014798221
- On the other hand, given a large number ***n*** it is very difficult to find two integers ***p*** and ***q*** such that ***n*** = ***p*** x ***q***

Powers of numbers mod N

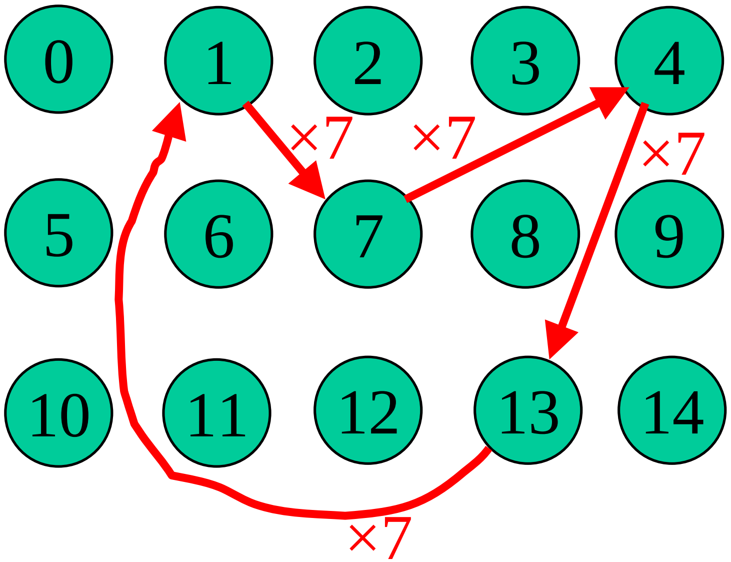
- Given natural numbers (non-negative integers) $N \geq 1$, $x < N$, and x, consider the sequence:

  $x^0 \bmod N$, $x^1 \bmod N$, $x^2 \bmod N$, …

  $= 1, x, x^2 \bmod N$, …
- If x and N are relatively prime, this sequence is guaranteed not to repeat until it gets back to 1.
- *Discrete logarithm* of y, base x, mod N:
  - The smallest natural number exponent k (if any) such that $x^k = y \pmod N$.
  - *I.e.*, the integer logarithm of y, base x, in modulo-N arithmetic. Example: $\mathrm{dlog}_7\ 13 \pmod N) = ?$

Discrete Log Example

- $N=15$, $x=7$, $y=13$.
- 1
- x
- $x^2 = 49 = 4 \pmod{15}$
- $x^3 = 4 \cdot 7 = 28 = 13 \pmod{15}$
- $x^4 = 13 \cdot 7 = 91 = 1 \pmod{15}$

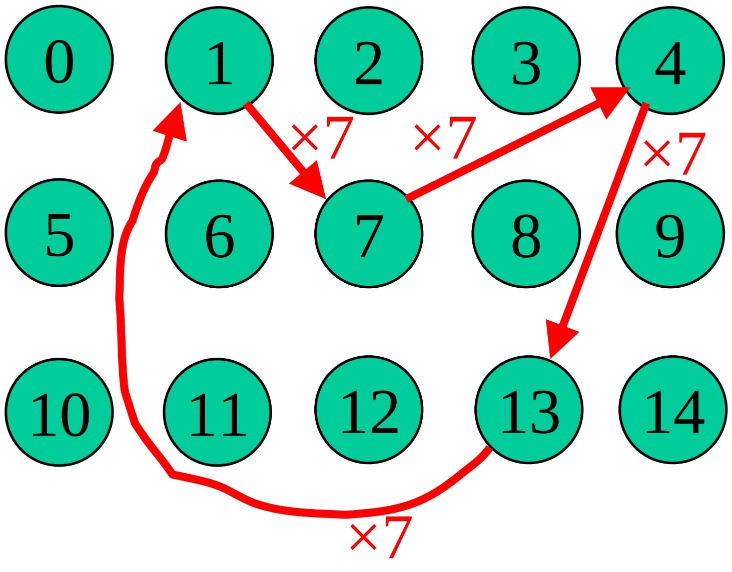


- So, $\text{dlog}_7\ 13 = 3 \pmod{N}$,
  - Because $7^3 = 13 \pmod{N}$.

Period 4

The *order* of x mod N

- **Problem:** Given $N>0$, and an $x<N$ that is relatively prime to N, what is the smallest value of $k>0$ such that $x^k = 1 \pmod{N}$?
  - This is called the ***order of x*** (mod N).
- From our previous example, the order of 7 mod N is…?

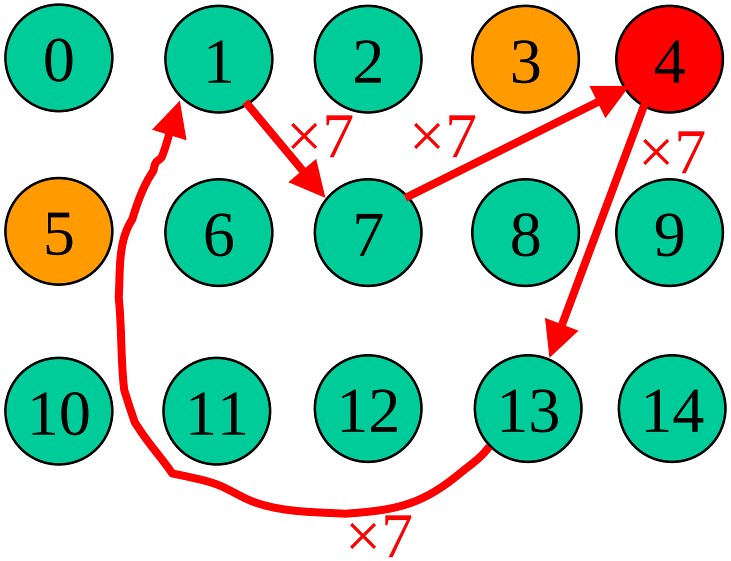


Order-finding permits Factoring

- A standard reduction of factoring ***N*** to finding orders mod *N*:
  - 1. Pick a random number $x < N$.
  - 2. If $\gcd(x,N) \neq 1$, return it (it's a factor).
  - 3. Compute the order of x (mod N).
    - Let ***r*** := min $k>0$: $x^k \bmod N = 1$
  - 4. If $\gcd(x^{r/2} \pm 1, N) \neq 1$, return it (it's a factor).
  - 5. Repeat as needed.
- The expected number of repetitions of the loop needed to find a factor with probability > 0.5 is known to be only polynomial in the length of *N*.

Factoring Example

- For ***N***=15, x=7…
- Order of x is ***r*=4.**
- $r/2 = 2$.
- $x^2 = 5$.
- In this case (we are lucky), *both* x^2+1 and x^2-1 are factors (3 and 5).



- Now, how do we compute orders efficiently?

Main Idea: Number Theory Trick

- Given an integer **N** to factor, create a function:
  - **$f_N(a) = x^a \bmod N$**
  - **x** is a random integer such that **$\gcd(x,N) = 1$**

- It turns out that **$f_N(a)$** is periodic
  - For successive inputs **a = 0, 1, 2, . . .** The function values **$f_N(0), f_N(1), . . .$** will repeat (different **x** values will produce different patterns)
  - For a given **x**, the period of the pattern is ***r***

There is a very good chance that the $\gcd(N, x^{r/2} - 1)$ is a factor of N

Example

- Given an integer **N** to factor, create a function:
  - $f_n(a) = x^a \bmod N$
  - **x** is a random integer such that **gcd(x,N) = 1**
- It turns out that $f_N(a)$ is periodic
  - For successive inputs **a = 0, 1, 2, . . .** The function values $f_N(0), f_N(1), \ldots$ will repeat (different **x** values will produce different patterns)
  - For a given **x**, the period of the pattern is ***r***

There is a very good chance that the gcd($N, x^{r/2} - 1$) is a factor of N

Select x = 8 then $f_{15}(a) = 8^a \bmod 15$

a	$f_{15}(a)$
0	1
1	8
2	4
3	2
4	1
5	8
6	4

r = 4

gcd($N, x^{r/2} - 1$)

x = 8

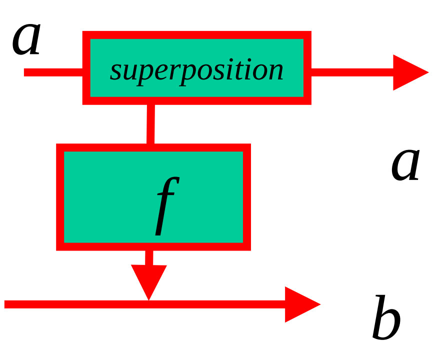
$8^2 - 1 = 63$

Find the gcd(63, 15) = 3

3 is a factor of 15

Main Idea: Quantum Approach

- **Goal**: Find the period of $f_N(a)$
- **PROCESS**: construct a single quantum register then partition it into two parts
  - R1 and R2
- Store a superposition of all values of ***a*** in **a**
  - Evaluate $f_N(a)$ and place the result in **b**



Select x = 8 then $f_{15}(a) = 8^a \bmod 15$

a	$f_{15}(a)$
0	1
1	8
2	4
3	2
4	1
5	8
6	4

r = 4

$gcd(N, x^{r/2} - 1)$

x = 8

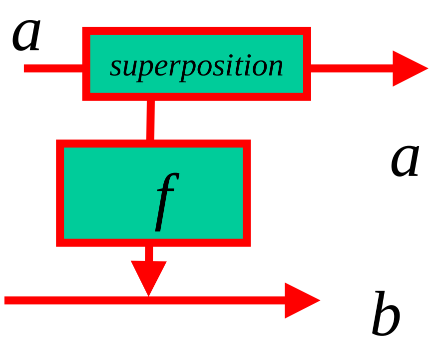
$8^2 - 1 = 63$

Find the gcd(63, 15) = 3

3 is a factor of 15

Main Idea: Using b

- Now b is a superposition of all possible function values (it only took 1 evaluation)
  - Measure b – this causes it to collapse to a single value, say **k**
  - This means that for some **a**, **x^a mod N = k**
  - Because **a** and **b** are entangled, **a** now contains a superposition of only those values of **a** such that **x^a mod N = k**



Select x = 8 then $f_{15}(a) = 8^a$ mod 15

$gcd(N, x^{r/2} - 1)$

a	$f_{15}(a)$
0	1
1	8
2	4
3	2
4	1
5	8
6	4

r = 4

x = 8

$8^2 - 1 = 63$

Find the gcd(63, 15) = 3

3 is a factor of 15

Main Idea: Fourier Transform

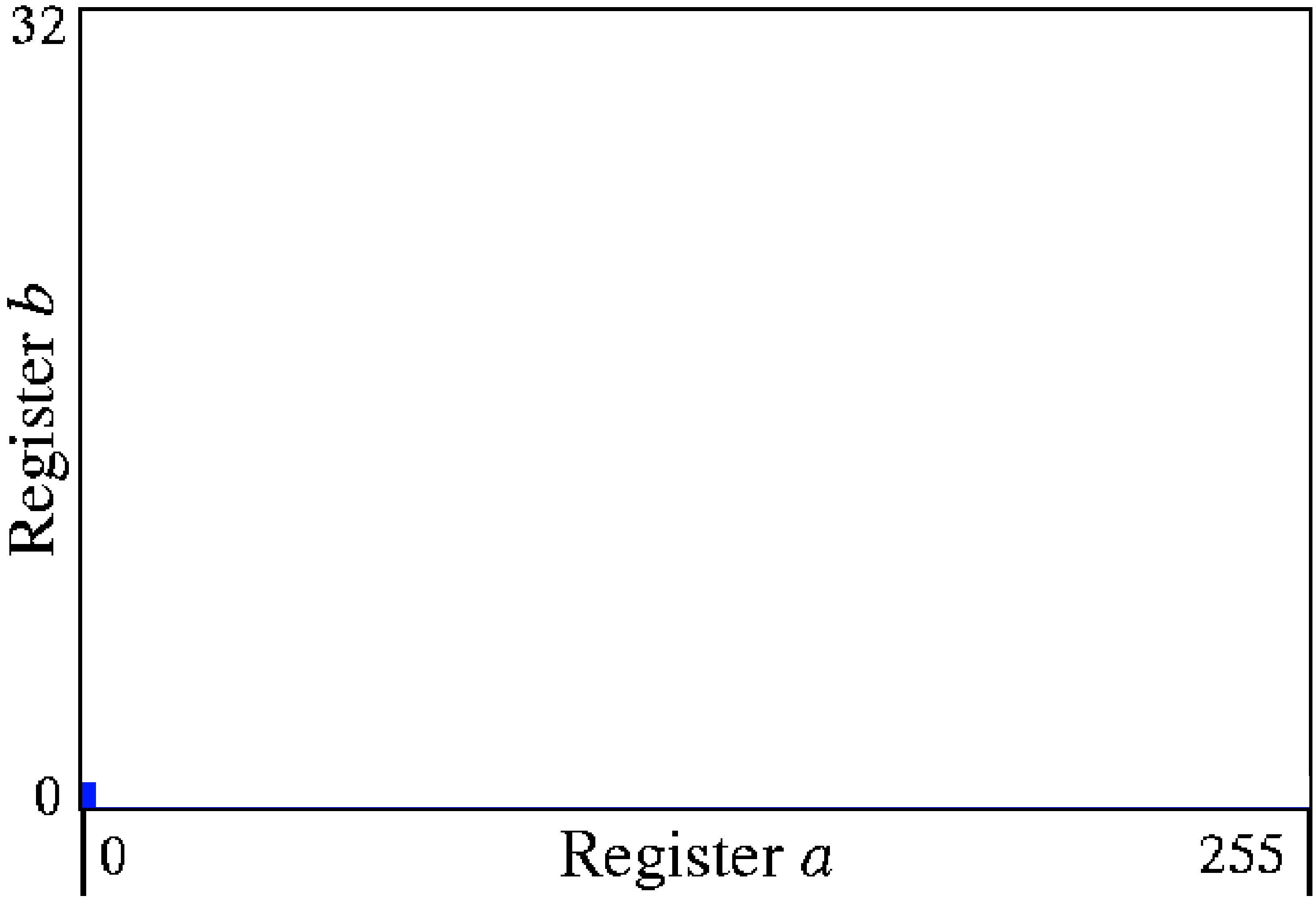
- Perform a Fourier Transform on **a** to find the period **r**

- Calculate the gcd to find a possible factor

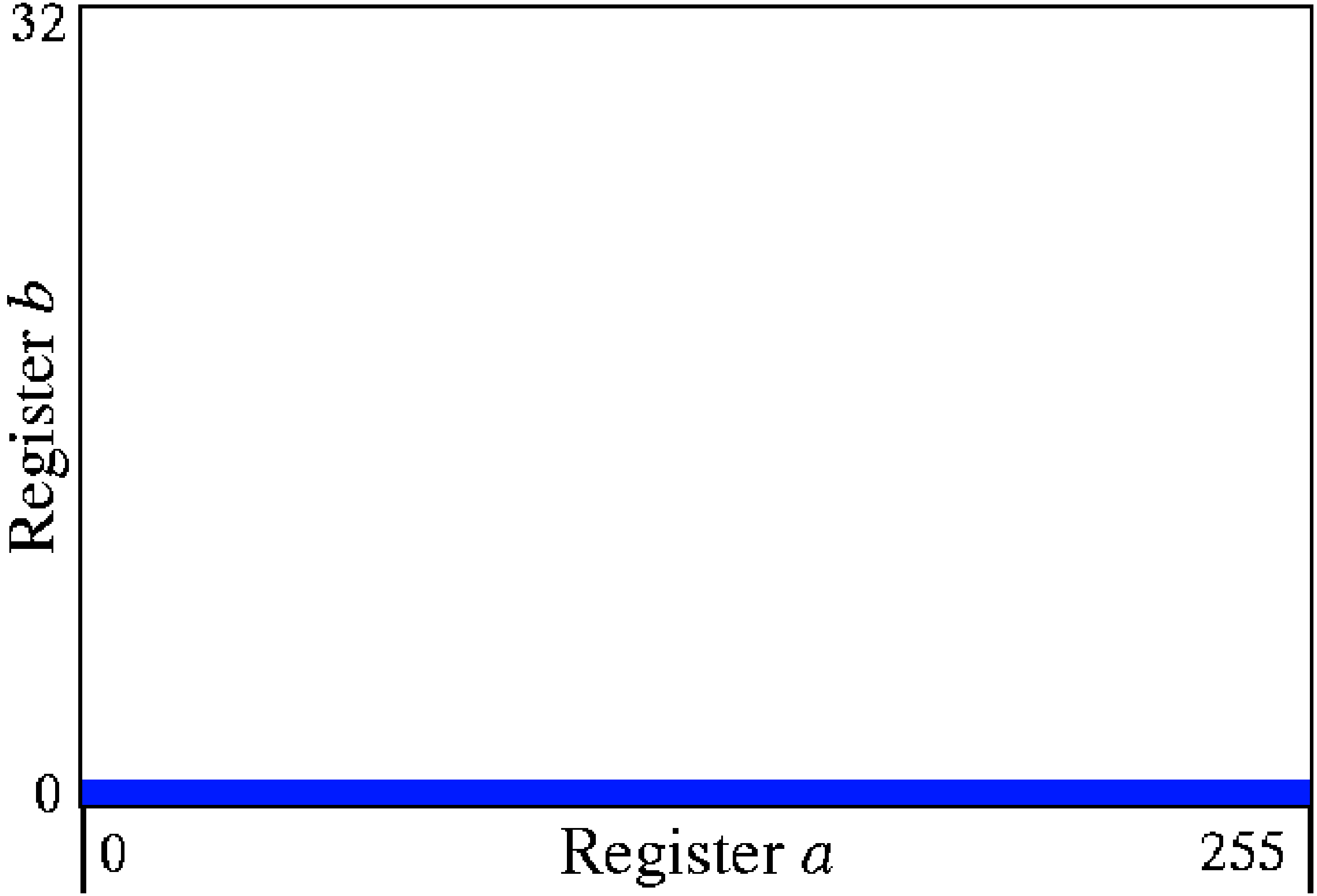
Quantum Order-Finding

- Uses 2 quantum registers (*a*,*b*)
  - $0 \le a < q$, is the *k* (exponent) used in order-finding.
  - $0 \le b < n$, is the *y* (x^k mod *n*) value
  - *q* is the smallest power of 2 greater than N^2.
- Algorithm:
  - 1. Initial quantum state is |0,0>, *i.e.*, (a=0, b=0).
  - 2. Go to superposition of all possible values of *a*:

Initial State

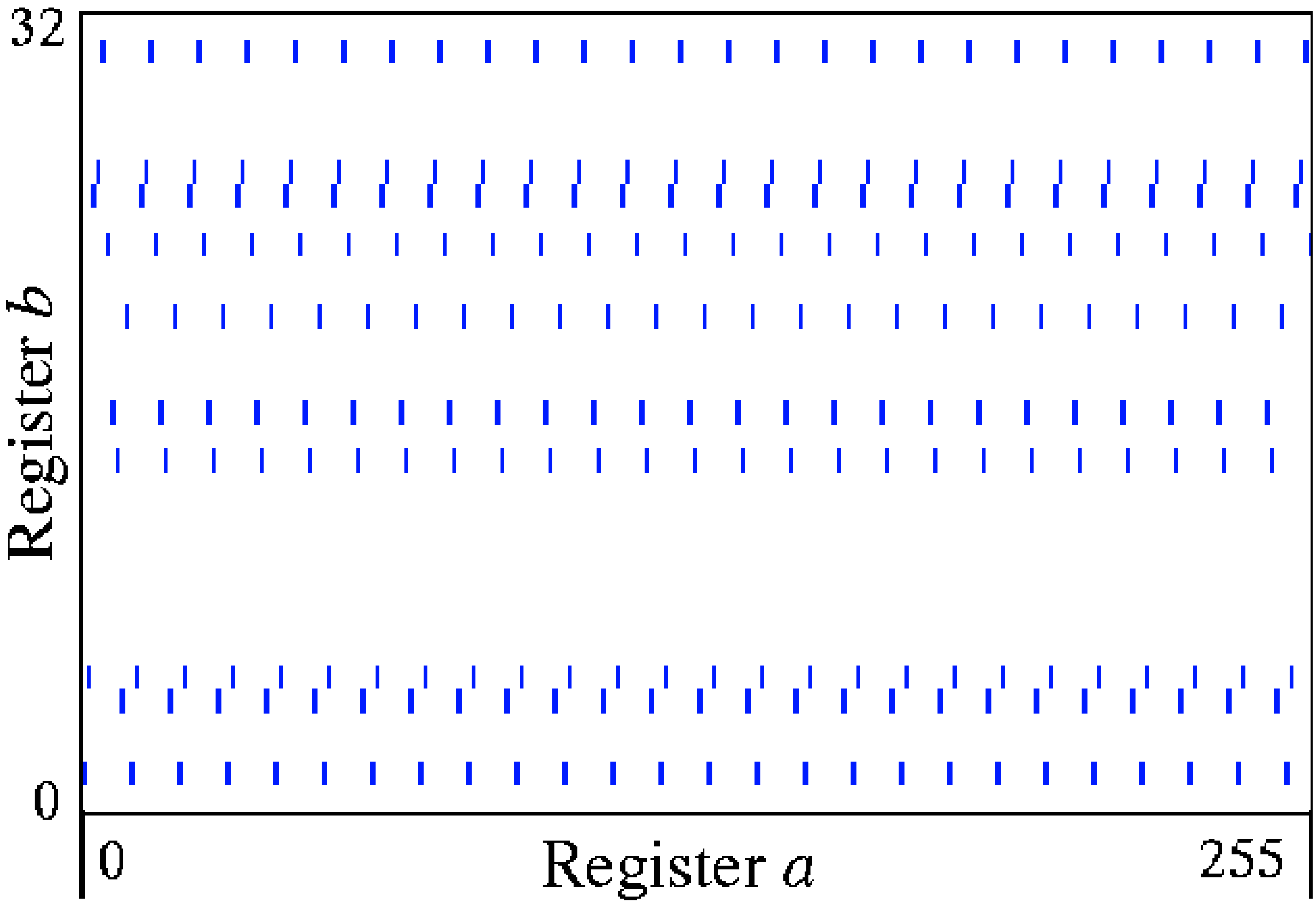


After Doing Hadamard Transform on all bits of a



After modular exponentiation

$b = x^a \pmod{N}$



State After Fourier Transform