Algorytmy kwantowe z biblioteką Qiskit

Hubert Malinowski i Jerzy Szyjut

Qubit – kwantowy bit

- W komputerach klasycznych najmniejszą jednostkę przechowywania informacji nazywamy bitem – może on przyjmować wartość 0 lub 1
- W komputerach kwantowych odpowiednikiem bitu jest qubit
- Mówimy, że qubit jest superpozycją stanów $|0\rangle$ oraz $|1\rangle$
- Współczynnikami przy stanach podstawowych są liczby zespolone

Stan kwantowy

Qubit jest superpozycją dwóch stanów podstawowych, które są do siebie ortogonalne (prostopadłe):

$$|0\rangle = \binom{1}{0} \qquad |1\rangle = \binom{1}{0}$$

Możemy myśleć o qubicie jako o pewnym wektorze w przestrzeni:

$$\begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix} = \frac{\sqrt{3}}{2} |0\rangle + \frac{1}{2} |1\rangle$$

Współczynniki przy stanach kwantowych są powiązane z prawdopodobieństwem wystąpienia danego stanu podstawowego podczas pomiaru, a dokładniej:

$$P(s = |0\rangle) = \left(\frac{\sqrt{3}}{2}\right)^2 = \frac{3}{4} \qquad P(s = |1\rangle) = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$$

Wektor stanów

Mając n qubitów, możemy utworzyć n-qubitowy wektor stanów, który po pomiarze może przyjąć jedną z 2^n wartości.

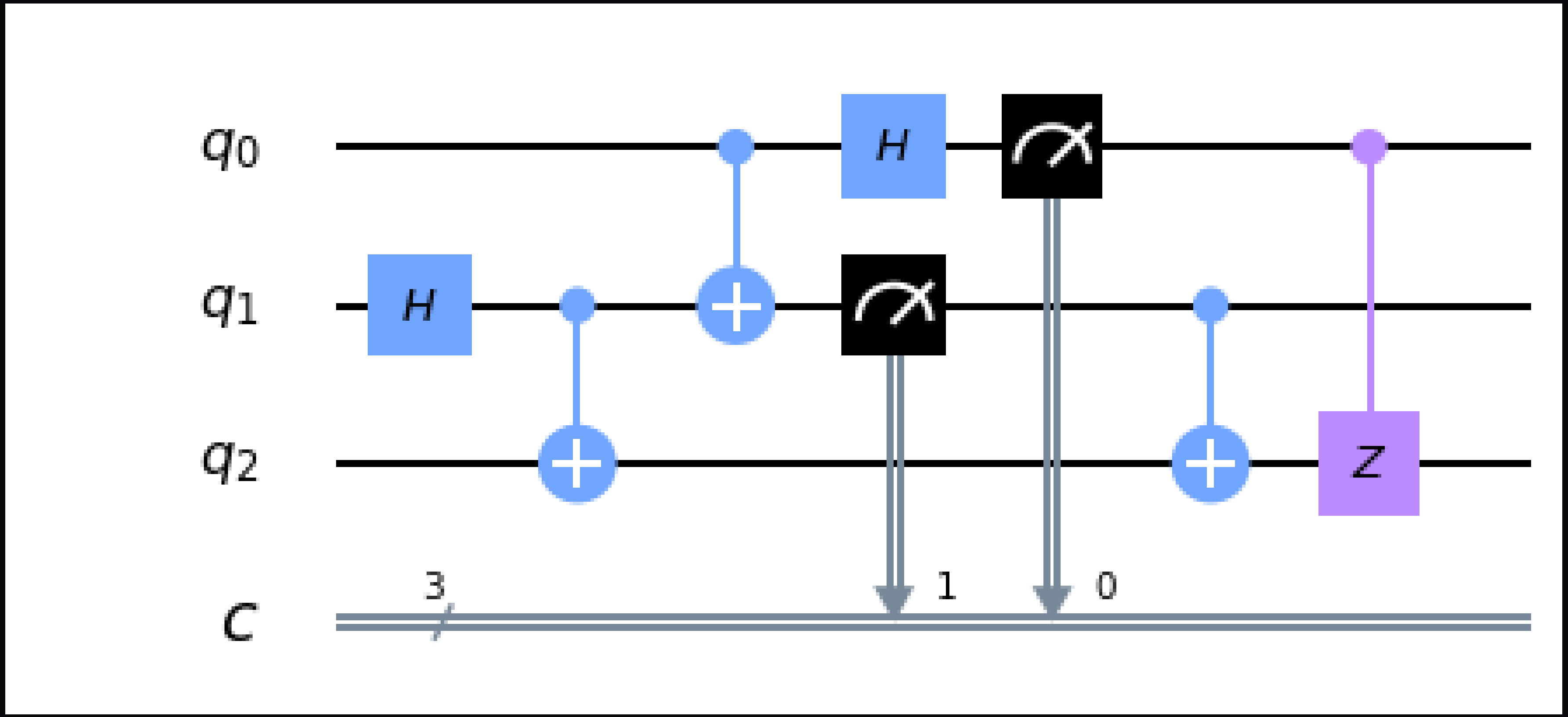
2-qubitowy wektor stanu możemy zapisać:

$$|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

Po odczycie z jednakowym prawdopodobieństwem przyjmuje jeden z 4 stanów podstawowych.

Układy kwantowe

Obliczenia kwantowe, podobnie jak w przypadku obliczeń klasycznych, możemy przedstawić w postaci bramek (kwantowych)



Bramki kwantowe

Wektory stanów kwantowych przedstawiamy za pomocą macierzy – operacje na nich przeprowadzane (reprezentowane jako bramki), są po prostu mnożeniem wektorów przez pewną macierz przekształcenia.
Co ważne, macierze te muszą być macierzami ortonormalnymi – czyli ich kolumny są wektorami o długości 1.

Podstawowe bramki kwantowe

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

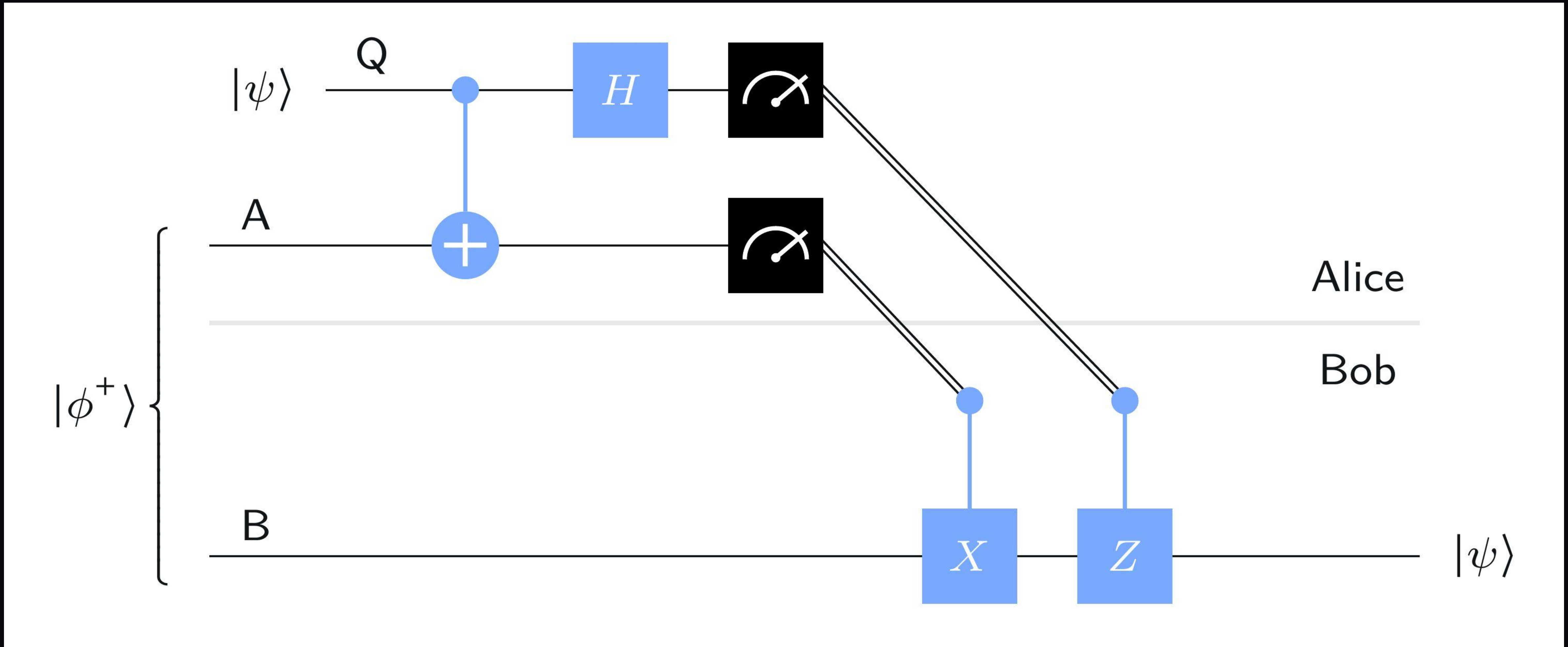
$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Stany splątane (entangled states)

Stany splątane to stany, w którym zmierzenie jednego qubitu wpływa na stan drugiego qubitu. Przykładem jest jeden ze stanów Bella:

$$|\Phi^+\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|11\rangle$$

Teleportacja kwantowa



Biblioteka Qiskit

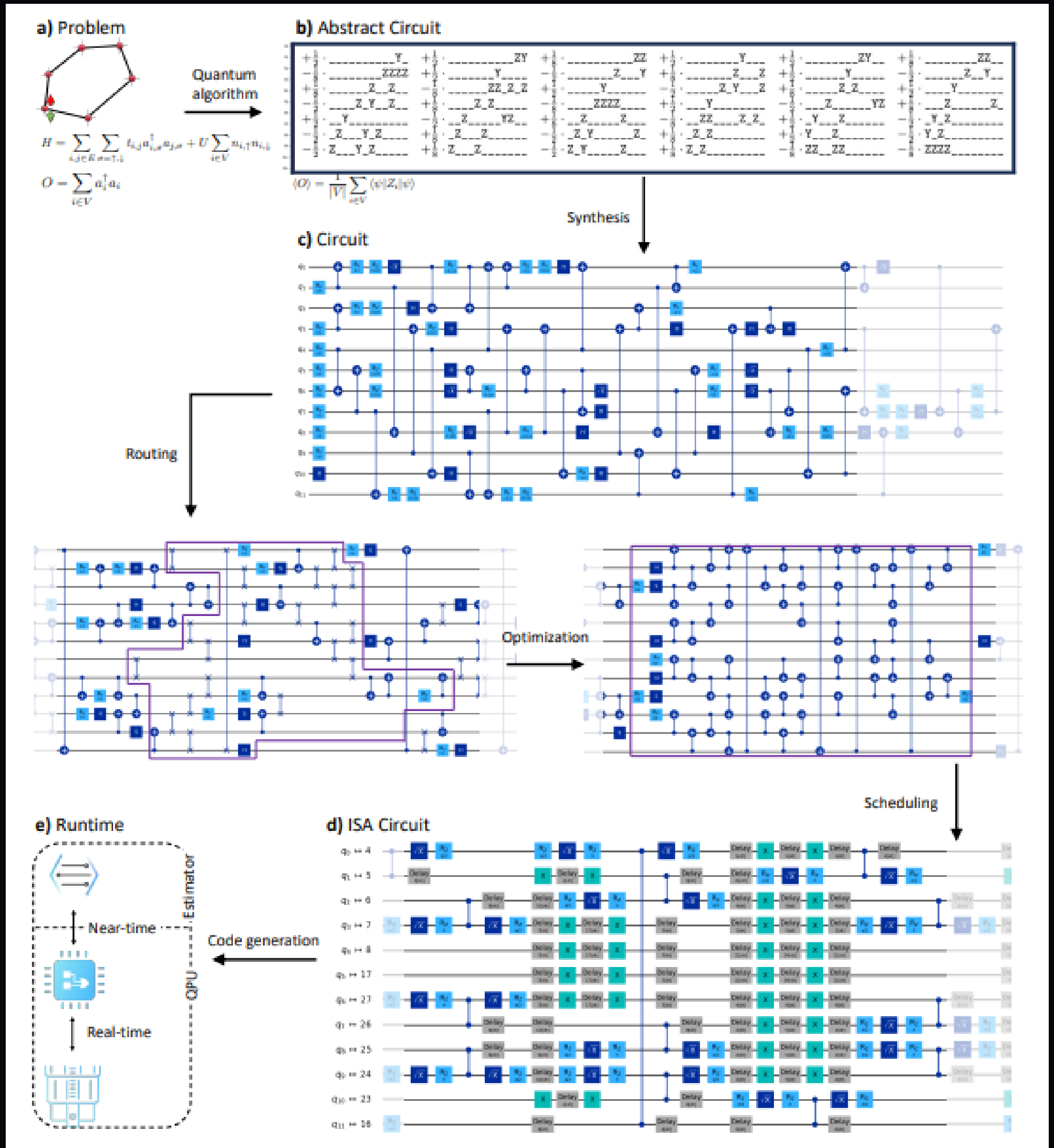
Co to jest Qiskit?

Open-source SDK do programowania komputerów kwantowych od IBM (2017)

Statystyki

- 6 mln instalacji
- 500+ kontrybutorów
- 2000+ publikacji na arXiv

Główna koncepcja w Qiskit



Poziomy abstrakcji

Od problemu fizycznego do instrukcji sprzętowych

Etapy przetwarzania

1. Tworzenie abstrakcyjnych obwodów
2. Synteza: bramki uniwersalne
3. Transpilacja: optymalizacja i dostosowanie
4. Wykonanie na sprzęcie lub symulatorze

Faktoryzacja liczb

- Cel – dla danej liczby, znaleźć czynniki będące jej liczbami pierwszymi
- Najlepsze znane algorytmy mają złożoność subwykładniczą
- Kluczowe dla bezpieczeństwa systemów szyfrowania

Algorytm Shora

1. Wybierz losowe $a < N$, takie że $\gcd(a, N) = 1$

2. Zdefiniuj funkcję $f(x) = a^x \, mod \, N$

3. Znajdź okres r: najmniejsze r, dla którego $a^r \equiv 1 \, mod \, N$
   - wykonywane na komputerze kwantowym

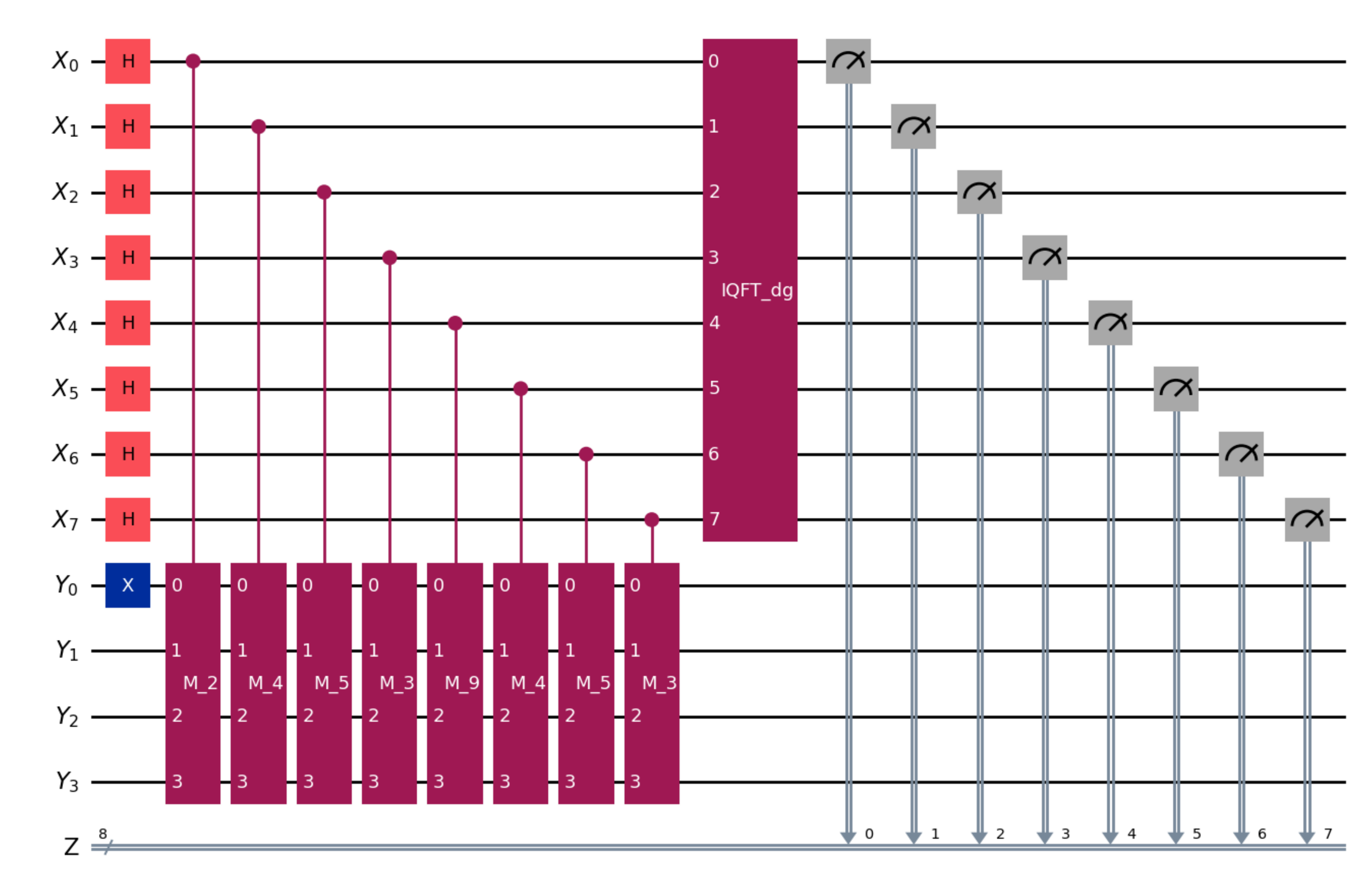
4. Jeśli r jest parzyste i $a^{r/2} \not\equiv 1 \, mod \, N$ to czynnikiem N jest $\gcd(a^{\frac{r}{2}} \pm 1, N)$

Algorytm Order Finding

- Znajdowanie okresu r dla którego $a^r \equiv 1 \; mod \; N$ jest znane jako problem order finding
- Wszystkie algorytmy klasyczne rozwiązują ten problem w czasie wykładniczym względem liczby bitów liczby
- Komputer kwantowy jest w stanie rozwiązać problem w czasie wielomianowym

Order Finding - implementacja

```python
def create_order_finding_circuit(a, N):
    if gcd(a, N)>1:
        print(f"Error: gcd({a},{N}) > 1")
    else:
        n = floor(log(N - 1, 2)) + 1
        m = 2*n

        control = QuantumRegister(m, 'X')
        target = QuantumRegister(n, 'Y')
        output = ClassicalRegister(m, 'Z')
        circuit = QuantumCircuit(control, target, output)

        circuit.x(m)

        for k, qubit in enumerate(control):
            circuit.h(k)
            b = pow(a,2**k,N)
            circuit.compose(
                mod_mult_gate(b,N).control(),
                qubits = [qubit] + list(target),
                inplace=True)

        circuit.compose(
            QFT(m, inverse=True),
            qubits=control,
            inplace=True)

        circuit.measure(control, output)

        return circuit
```



Dziękujemy za uwagę!