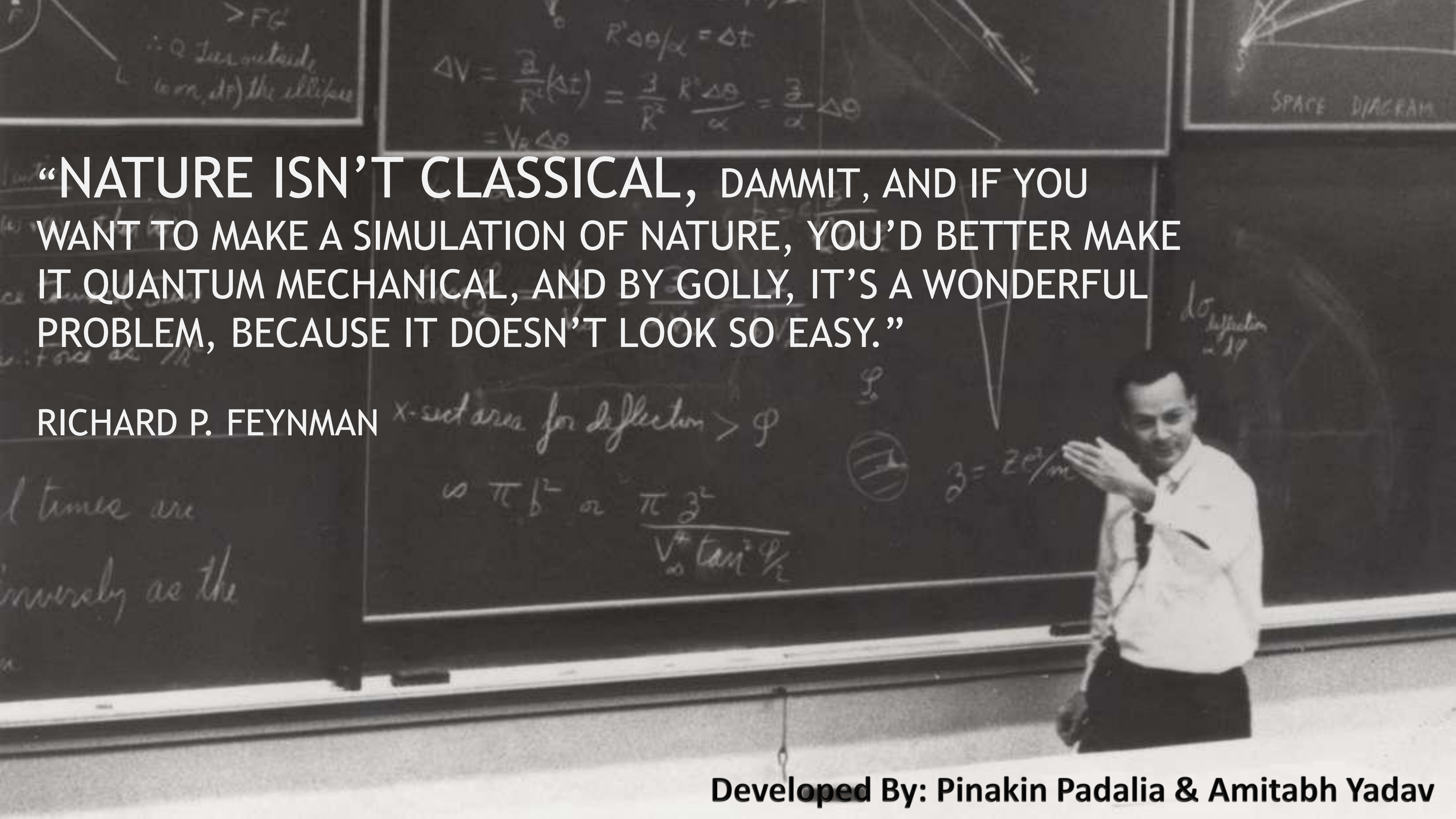
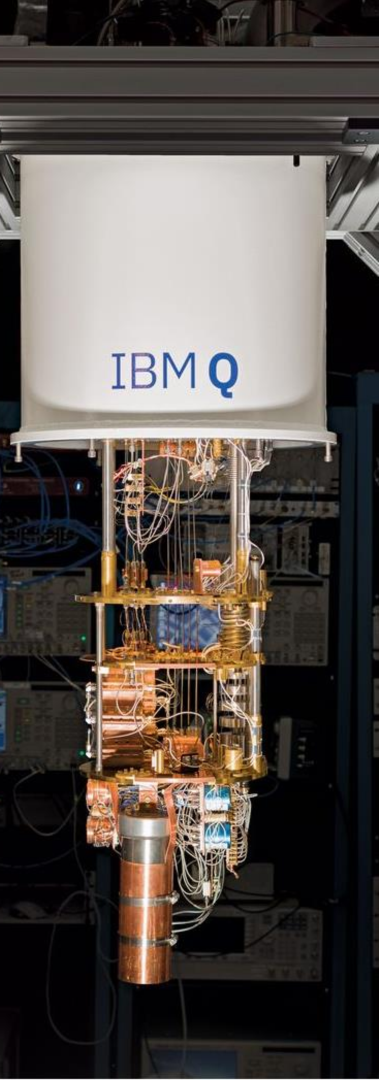


"NATURE ISN'T CLASSICAL, DAMMIT, AND IF YOU WANT TO MAKE A SIMULATION OF NATURE, YOU'D BETTER MAKE IT QUANTUM MECHANICAL, AND BY GOLLY, IT'S A WONDERFUL PROBLEM, BECAUSE IT DOESN'T LOOK SO EASY."

RICHARD P. FEYNMAN

Developed By: Pinakin Padalia & Amitabh Yadav



Running Shor's Algorithm on IBM Quantum Experience

Presentation By:

Pinakin Padalia & *Amitabh Yadav*

Student No.4743644 *Student No. 4715020*

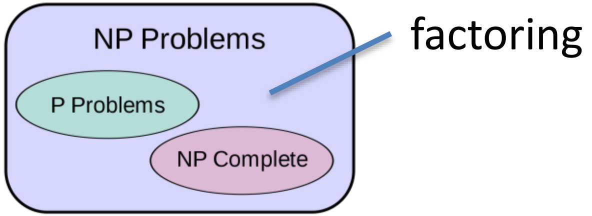
Agenda

- The Factoring Problem
- Shor's Algorithm for Factoring Integers
- Quantum Circuit for Period Finding
- IBM Quantum Experience
- Running Shor's Algorithm on IBM-Q: Methodology and Challenges
- Design Space Exploration
- Results & Conclusion

The Factoring Problem

- RSA uses a public key N which is the product of two large prime numbers.
- One way to crack RSA encryption is by factoring N, but with classical algorithms, factoring becomes increasingly time-consuming as N grows large.

- **RSA-1024** has 1,024 bits (309 decimal digits), and has not been factored so far.



- No classical algorithm is known that can factor in polynomial time.
- Shor's (Quantum) Algorithm can crack RSA in polynomial time.

Shor’s Algorithm

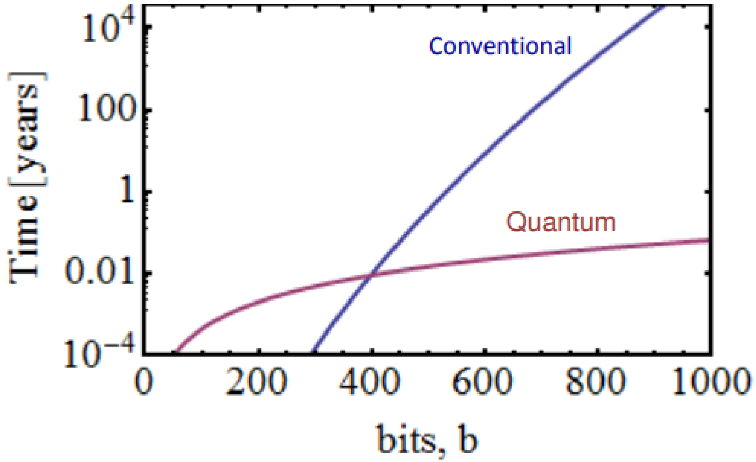
Peter Shor

Shor's algorithm is a quantum algorithm for factoring a number N in $O(n^3)$ time, named after Peter Shor.

Factor a number into primes:

$M = p * q$

How long will it take ? (t)



Source: http://www.ibm.com/

<u>Classical</u>

$t \sim O(2^{n^{(1/2)}})$

<u>Quantum</u>

$t \sim O(n^3)$

Shor's Algorithm: Implementation

Number to be factored (let), $N = 15$.

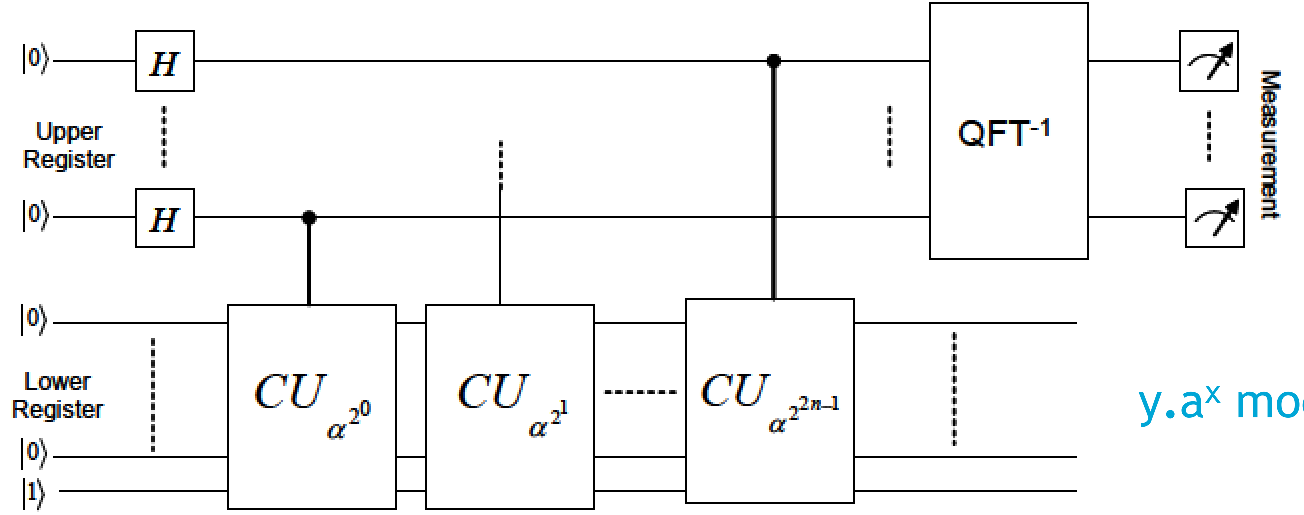
1. If N is even/integer power of prime number $\rightarrow$ can be factored classically.
2. Choose a random integer $a \in [2,\ldots,N-1]$, (let) $a = 11$ and compute t=GCD(a,N).
3. If t>1, t is a factor; else if t=1 (here t=1)
4. Find r = period(a^x mod N) [Using Shor's Algorithm] (here r=2)
5. If r is odd $\rightarrow$ $a^{r/2} + 1 = 0 \bmod N$ $\rightarrow$ Cannot Infer Factors. Go To Step 2.
6. Else If r is even,
   factor_1 = gcd($a^{r/2}$ + 1,N) = 3
   factor_2 = gcd($a^{r/2}$ - 1,N) = 5

Challenge: Period Finding!

TUDelft

Developed By: Pinakin Padalia & Amitabh Yadav

Quantum Circuit for Period Finding (Shor's Algorithm)



$y.a^x$ mod N

$$a^x \bmod N = (a^{2^0} \bmod N)^{x_0} \cdot (a^{2^1} \bmod N)^{x_1} \cdots (a^{2^{2n-1}} \bmod N)^{x_{2n-1}} \bmod N$$

Ref: Fast Quantum Modular Exponentiation Architecture for Shor's Factorization Algorithm [4]

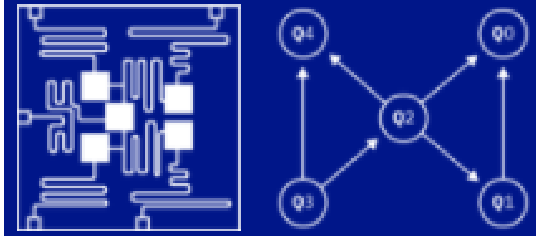
IBM Quantum Experience

http://research.ibm.com/ibm-q/

- First commercially available Quantum Computer and Developer Ecosystem
- Simulation Tools
- Quantum Experiments through cloud
- QASM Programming
- QISKit SDK
  (simulation and quantum execution using Python API)
- Active User Community
  (qiskit.slack.com)

QISKit

- Python API and SDK
- Contains:
  QISKit SDK : Python Interface for programming
  QISKit API : Python Wrapper to connect to IBM's Quantum Chip.
  QISKit OpenQASM : Execute OPENQASM code from Python
- Enables working with quantum circuits
- Quantum Processor: Raven (ibmqx4)



	Q0	Q1	Q2	Q3	Q4
Frequency (GHz)	5.24	5.30	5.35	5.43	5.18
T1 (µs)	46.10	49.30	46.30	44.10	61.00
T2 (µs)	38.30	68.40	46.70	12.70	29.60

Backend: ibmqx4 (5 Qubits)

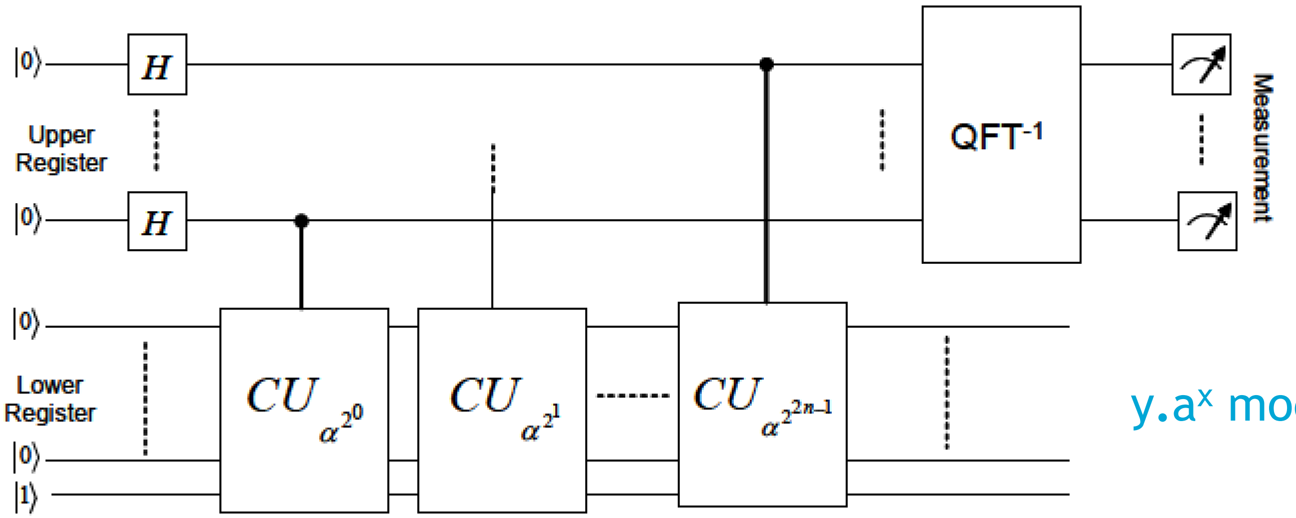
Shor’s Algorithm on IBM Q: Design Space Exploration

- 12 qubit simulation
- 5 qubit simulation
- 5 qubit hardware – general and particular (a = 11)

TUDelft

Developed By: Pinakin Padalia & Amitabh Yadav

Quantum Circuit for Period Finding (Shor's Algorithm)



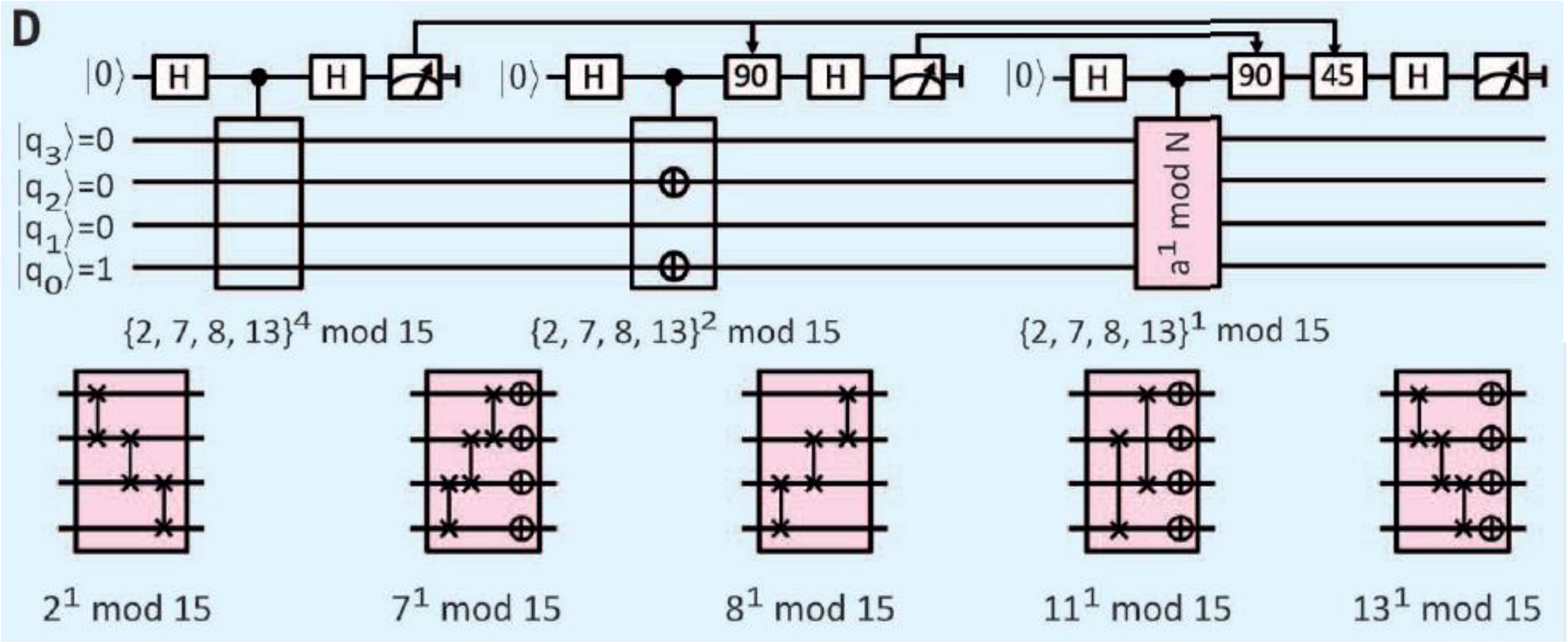
y.a^x mod N

$$a^x \bmod N = (a^{2^0} \bmod N)^{x_0} \cdot (a^{2^1} \bmod N)^{x_1} \cdots (a^{2^{2n-1}} \bmod N)^{x_{2n-1}} \bmod N$$

TUDelft

Developed By: Pinakin Padalia & Amitabh Yadav

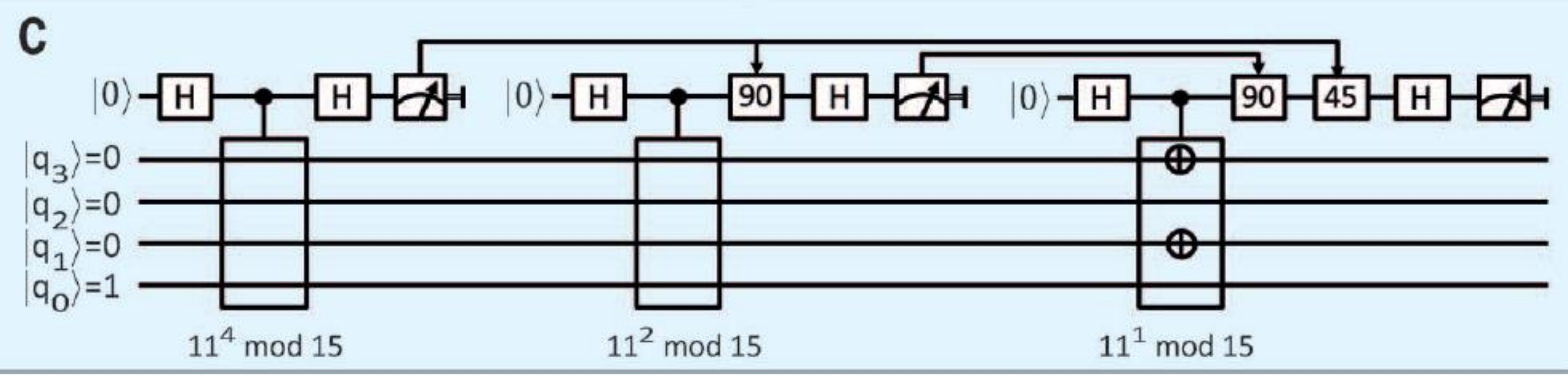
Running Shor's Algorithm on IBM Quantum Experience



Quantum Circuit for Period Finding (Qubit Recycling) [2]

Realization of a scalable Shor algorithm

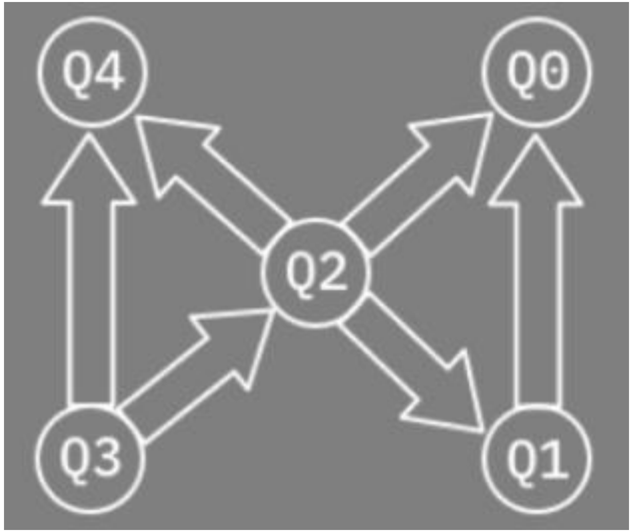
Shor’s Algorithm on IBM Q (ibmqx4)

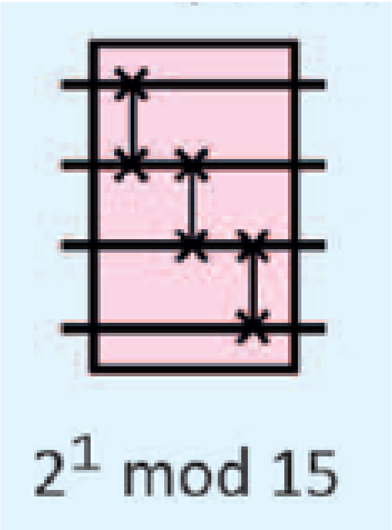


Quantum Circuit for Period Finding a=11 [2]

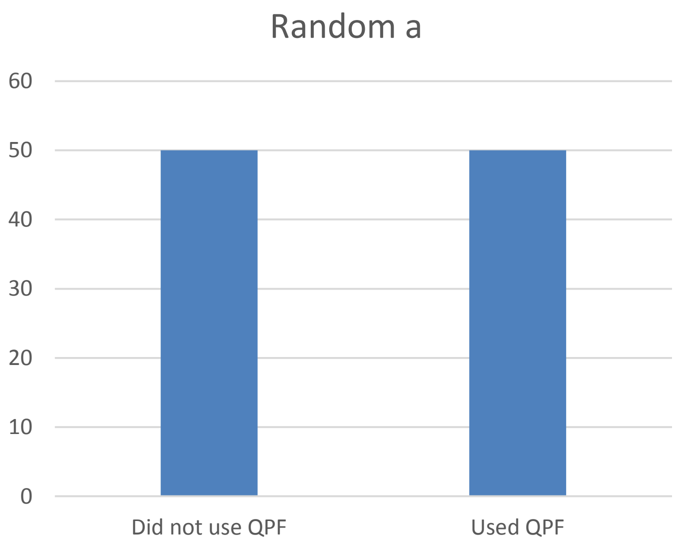
Shor’s Algorithm on IBM Q: Challenges

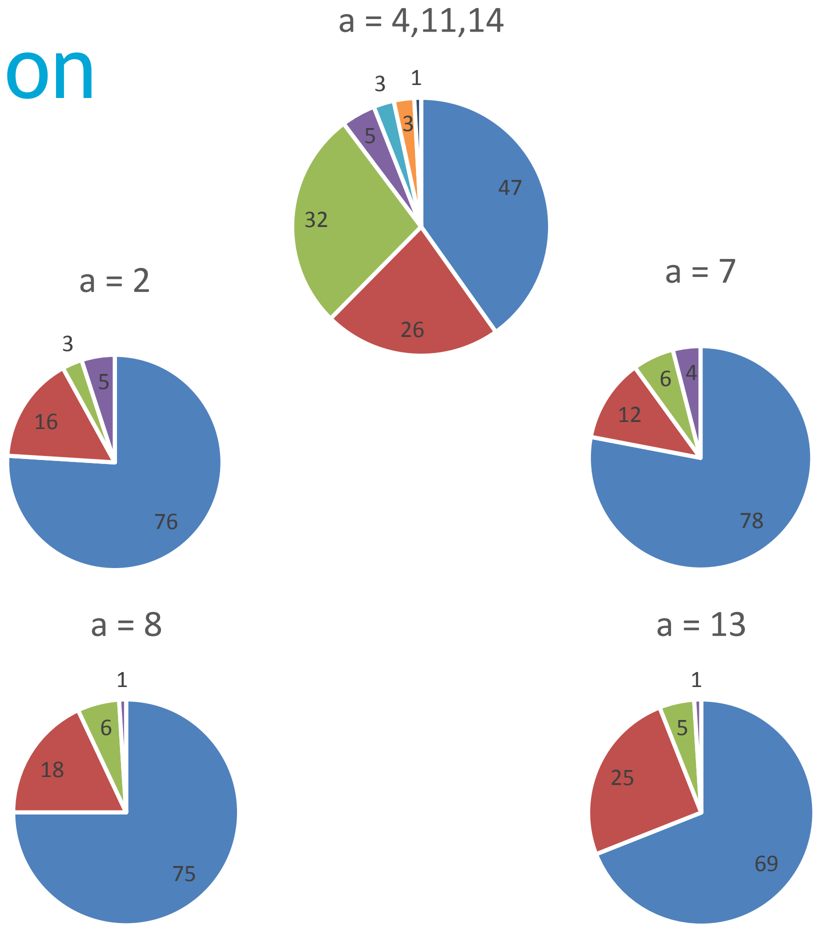
- Installation!
- Coupling Map!





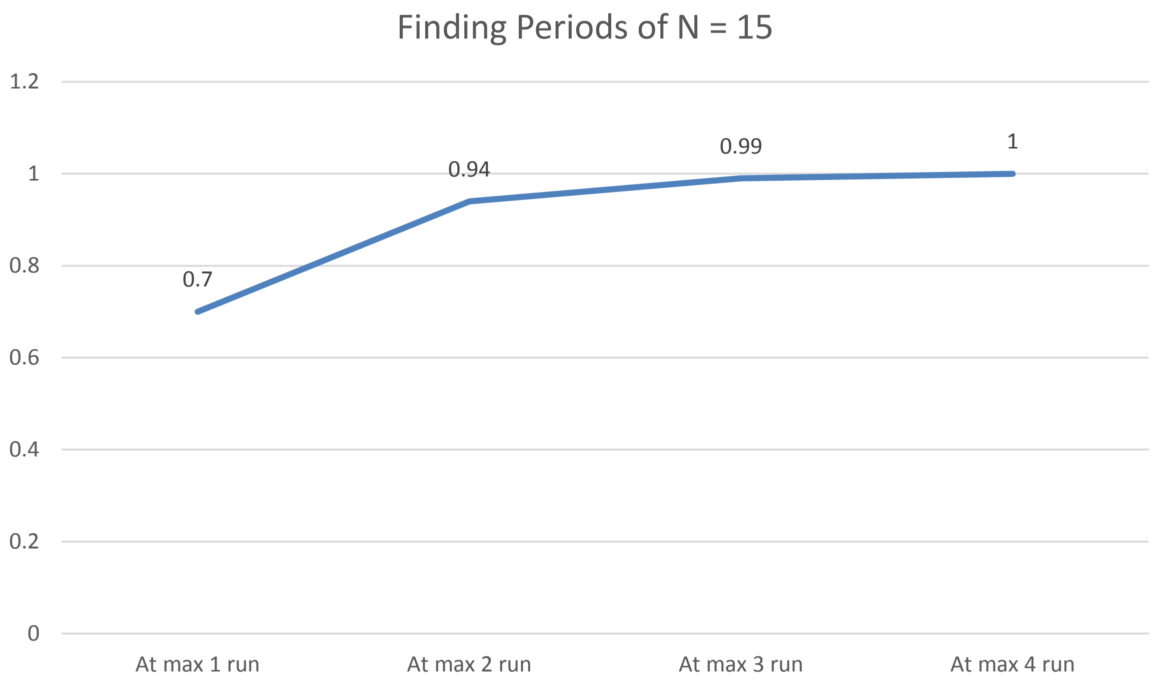
Results – 5 qubit simulation





TU Delft

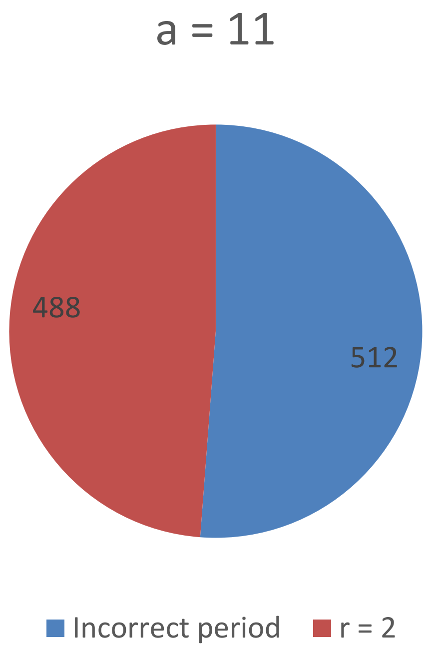
Results – 5 qubit simulation



TU Delft

Developed By: Pinakin Padalia & Amitabh Yadav

Results – 5 qubit Hardware (a = 11)

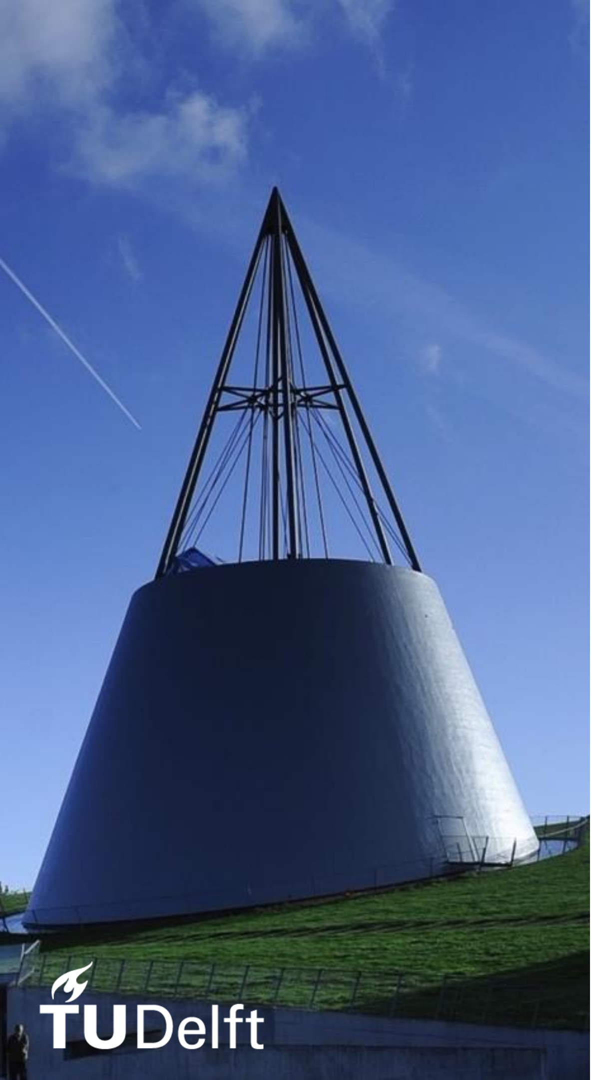


TUDelft

Developed By: Pinakin Padalia & Amitabh Yadav

Conclusion

- Algorithm runs – 5 qubit hardware and simulation.
- Challenges –
  - scalability
  - fidelity
  - generality



References

1) Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. arXiv:quant-ph/9508027
2) Monz, T., Nigg, D., Martinez, E.A., Brandl, M.F., Schindler, P., Rines, R., Wang, S.X., Chuang, I.L. and Blatt, R., 2016. Realization of a scalable Shor algorithm. Science, 351(6277), pp.1068-1070.
3) A. Y. Kitaev, http://arxiv.org/abs/quant-ph/9511026 (1995).
4) Fast Quantum Modular Exponentiation Architecture for Shor's Factorization Algorithm. arXiv:1207.0511
5) Circuit for Shor's algorithm using 2n+3 qubits. arXiv:quant-ph/0205095
6) IBM Quantum Experience Documentation/Full User Guide. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=introduction
7) https://github.com/QISKit/qiskit-tutorial
8) https://github.com/QISKit/qiskit-sdk-py