This Week

Lecture 9
Fourier Transformations, Regular Fourier Transform, Fourier Transform as a matrix, Quantum Fourier Transform, QUI examples, Inverse QFT

Lecture 10
Shor's Quantum Factoring algorithm, Shor's algorithm for factoring and discrete logarithm, HSP Problem

Lab 5
QFT and Shor's algorithm

Quantum Factoring Algorithm

Physics 90045
Lecture 10

Quantum factoring algorithm

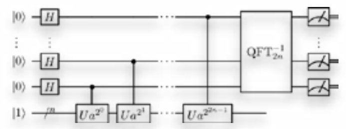
- Shor's Factoring algorithm
  - Shor's algorithm for factoring and discrete logarithm
  - HSP Problem
  - RSA cryptography

Reiffel, Chapter 8
Kaye, Chapter 7
Nielsen and Chuang, Chapter 5

Shor's algorithm

- Efficient quantum algorithms for **factoring** semiprime numbers
- Best known classical algorithm is number field sieve (exponential in bit-length).
- Underpins the RSA cryptosystem
- Hidden Subgroup Problems (eg. Discrete logarithm) similar.

Peter Shor



Shor, Proc 35th Ann Symp of Comp Sci, 26, (1995)

Factoring and Period Finding

We want to factor N=15. Take a number a=2 (say) relatively-prime to N (ie. no prime factors in common) and find *the order* r of a. That is the least r, such that $a^r = 1$ mod 15:

$$\begin{aligned} 2^0 &= 1 \quad \mod 15 \\ 2^1 &= 2 \quad \mod 15 \\ 2^2 &= 4 \quad \mod 15 \\ 2^3 &= 8 \quad \mod 15 \\ 2^4 &= 1 \quad \mod 15 \end{aligned}$$

After which the pattern repeats.

Formally, we say: the **order** of 2 mod 15 is 4. Or, if we defined a function:

$$f(k) = a^k \mod N$$

We would say that the **period** of f is r, since f(x+r) = f(x).

Example of finding factors from a period

In our case, we have a=2, N=15 and r=4. Happily r=4 is even. We can rearrange:

$$\begin{aligned} a^r &= 1 \mod N \\ a^r - 1 &= 0 \mod N \\ (a^{r/2}+1)(a^{r/2}-1) &= 0 \mod N \end{aligned}$$

In our case,

$$\begin{aligned} a^{r/2} - 1 &= 2^{4/2} - 1 = 3 \\ a^{r/2} + 1 &= 2^{4/2} + 1 = 5 \end{aligned}$$

and $\boxed{3 \times 5 = 15}$

Divisors of N

In our case 3 and 5 divide N=15 exactly, but we're not guaranteed that always, only that:

$$(a^{r/2}+1)(a^{r/2}-1) = 0 \mod N$$

ie. that

$$(a^{r/2}+1)(a^{r/2}-1) = kN$$

As long as neither factor is a multiple of N, then both will have non-trivial factors with N. To find these factors, we find the greatest common divisors (for which the Euclidean algorithm is efficient):

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

These give a **non-trivial factor of N**.

If r is even or if the factors found are trivial, we repeat the algorithm with a different choice of a.

TLDR: Factoring and Period finding

If we can find the period of

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

efficiently, we can factor efficiently.

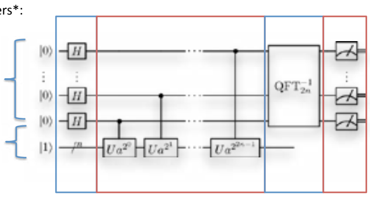
Shor's algorithm finds this period efficiently, and we can then use classical techniques to factor semi-prime numbers into their prime factors.

Shor's algorithm

Two registers*:

2L qubits

L qubits



(1) Equal superposition

(2) Calculate function:
f(x) = a^x mod N

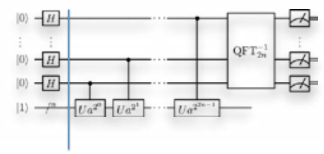
(3) QFT

(4) Measure result

* L = number of bits in N

Shor, Proc 35th Ann Symp of Comp Sci, 26, (1995)

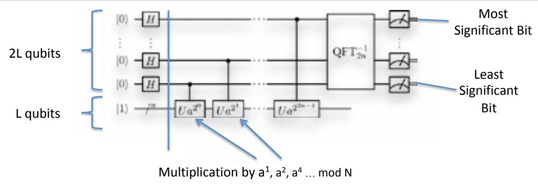
Shor's algorithm explained



After the Hadamard gates, the top register is in the equal superposition:

$$|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|1\rangle$$

Modular Exponentiation

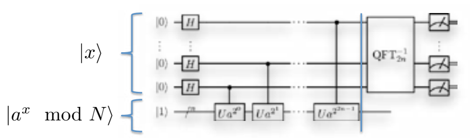


For example if the top register contained x = 101, and a=2, and N=15 then we would:

- Start with 1
- Multiply by $a^1=2^1=2$ giving 2 mod 15
- Not multiply by $a^2=2^2=4$
- Multiply by $a^4=2^4=16$ giving 32 or 2 mod 15

Top register in superposition, so bottom register is correlated (entangled) with the top register

Example of Modular Exponentiaion



After modular exponentiation:

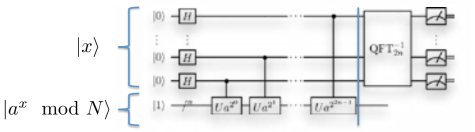
$$|\psi\rangle = \sum_x |x\rangle|a^x \mod N\rangle$$

Note: States are unnormalized! (for simplicity)

e.g. For a=2, N=15:

$$\begin{aligned}|\psi\rangle &= (|0\rangle + |4\rangle + |8\rangle + |12\rangle)\otimes|1\rangle\\ &+ (|1\rangle + |5\rangle + |9\rangle + |13\rangle)\otimes|2\rangle\\ &+ (|2\rangle + |6\rangle + |10\rangle + |14\rangle)\otimes|4\rangle\\ &+ (|3\rangle + |7\rangle + |11\rangle + |15\rangle)\otimes|8\rangle\end{aligned}$$

Shor's algorithm explained



If, at this point the bottom register is measured to be 2 (at random), we may collapse to the state:

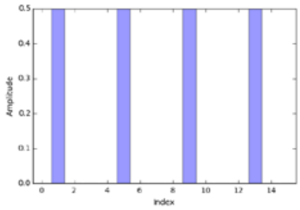
$$|\psi\rangle = (|1\rangle + |5\rangle + |9\rangle + |13\rangle) \otimes |2\rangle$$

PHYC90045 Introduction to Quantum Computing

The Fourier Transform

Imagine we measure the bottom register, and plot the amplitudes in the top register:

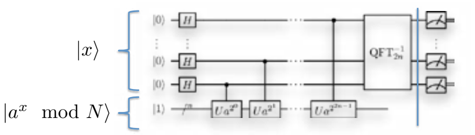
$$|\psi\rangle = (|1\rangle + |5\rangle + |9\rangle + |13\rangle) \otimes |2\rangle$$



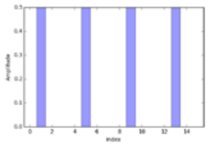
This function is periodic, with a period of r.

PHYC90045 Introduction to Quantum Computing

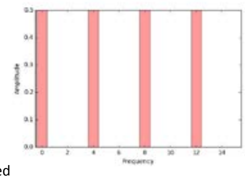
Taking the QFT



Inverse QFT for N=15, a=2



To find the period, we will take the Quantum Fourier Transform to reveal **r** (and so also, if r is even, factors of N).



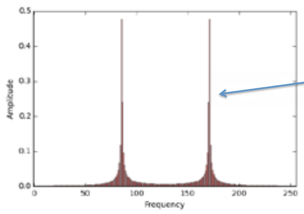
The result of taking a Fourier transform is a “spectrum” peaked around (for integer, k):

$$k\frac{2^n}{r}$$

n (2L) is number of qubits in the top register r is the period being determined

When r doesn’t divide evenly

What happens when r doesn’t divide evenly into the top register? Then we still get a very peaked distribution around the same values:

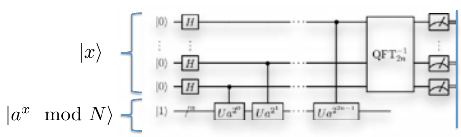


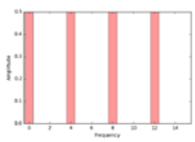
Peaked around:

$$k\frac{2^n}{r}$$

Here is an example for r=3 and 2^n=256.

Measurement





Measurement will randomly give one of these values, close to:

$$m = k\frac{2^n}{r}$$

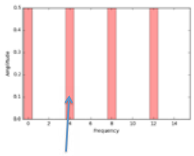
or

$$\frac{k}{r} = \frac{m}{2^n}$$

We need **a rational approximation** of $m/2^n$ to find r.

Example for a=2, N=15

In our example:



We might randomly measure m=4

$$\frac{k}{r} = \frac{m}{2^n}$$ and in this case: $$\frac{m}{2^n} = \frac{4}{16} = \frac{1}{4}$$

Since this is equal to k/r,
We have correctly found
r=4

Note: This step might only reveal a factor of r, and so might have to be repeated…

Continued Fractions

The result of taking a Fourier transform is a spectrum peaked around (for integer, k):

$$k\frac{2^n}{r}$$

Unless r divides 2^n exactly, we will only get an approximation to $k2^n/r$ when measured.

Most of the time 2^n and r will be relatively prime. The problem then is find good approximations to the measured value $m/2^n = k/r$. The "correct" approximation yields the period, r, as the denominator.

A good method for making *rational approximations* is to use the **continued fractions** method.

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\cdots \frac{1}{a_n}}}}$$

Continued Fraction of Pi

As an example, let's try to make a rational approximation to pi. Our first approximation is

$$\pi \approx 3 \qquad (a_0 = 3)$$

The remaining decimal part is 0.14159265… = 1/7.0625… This gives a second approximation:

$$\pi \approx 3 + \frac{1}{7} \qquad (a_1 = 7)$$

The remaining decimal part 0.0625 = 1/15.9966… This gives a third approximation:

$$\pi \approx 3 + \frac{1}{7 + \frac{1}{15}} \qquad (a_2 = 15)$$

And so on. This method can be used to find good rational approximations to $v/2^n$ and find r.

Example: Factoring the number

$$a^{r/2} - 1 = 2^{4/2} - 1 = 3$$

$$a^{r/2} + 1 = 2^{4/2} + 1 = 5$$

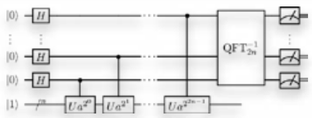
Not really necessary here, but in general you'd have to evaluate:

$$gcd(3, 15) = 3$$
$$gcd(5, 15) = 5$$

And so we've found two non-trivial factors of 15:

$$3 \times 5 = 15$$

Shor's algorithm Summary

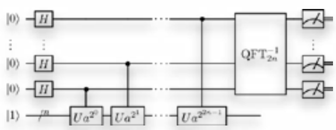


1. Randomly pick integer 0 < a < N (and check a is not a factor of N)
2. Apply the circuit above, using modular exponentiation to calculate a^x, QFT x.
3. Measure to obtain and approximation to v = k $2^n/r$
4. Use continued fractions of v/2^n to obtain even r
5. Use Euclidean algorithm to find common factors of N with ($a^{r/2}$+1) and ($a^{r/2}$-1)
6. Repeat if necessary

Shor's algorithm

- Efficient quantum algorithms for **factoring** semiprime numbers
- Best known classical algorithm is number field sieve (exponential in bit-length).
- Underpins the RSA cryptosystem
- Hidden Subgroup Problems (eg. Discrete logarithm) similar.

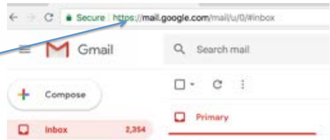
Peter Shor



Shor, Proc 35th Ann Symp of Comp Sci, 26, (1995)

Private Key Cryptography

Much of internet security relies on 'public key cryptography'.



RSA cryptography relies on the difficulty of factoring large semi-primes.

The best known **classical algorithm** is the number field sieve:

$$O(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}})$$

Shor's factoring **quantum algorithm** solves the same problem in poly-log time:

$$O((\log N)^2(\log\log N)(\log\log\log N))$$

RSA Factoring Challenge

RSA-180	180	596		May 8, 2010	S. A. Danilov and I. A. Popovyan, Moscow State University[7]
RSA-190	190	629		November 8, 2010	A. Timofeev and I. A. Popovyan
RSA-640	193	640	$20,000 USD	November 2, 2005	Jens Franke et al., University of Bonn
RSA-200	200	663		May 9, 2005	Jens Franke et al., University of Bonn
RSA-210	210	696		September 26, 2013[8]	Ryan Propper
RSA-704	212	704	$30,000 USD	July 2, 2012	Shi Bai, Emmanuel Thomé and Paul Zimmermann
RSA-220	220	729		May 13, 2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé and P. Zimmermann
RSA-230	230	762		August 15, 2018	Samuel S. Gross, Noblis, Inc.
RSA-232	232	768			
RSA-768	232	768	$50,000 USD	December 12, 2009	Thorsten Kleinjung et al.
RSA-240	240	795			

RSA Factoring Challenge, Wikipedia

Factoring a 768 bit number took ~1,500 years CPU time (largest RSA factoring challenge solved).

Factoring a 2048 bit number is estimated will take ~1 day on a large scale quantum computer (running at typical speeds, and low error).

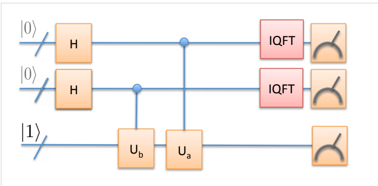
Discrete Logarithm

A closely related class of problems which are important for cryptography are solving discrete logarithm problems:

> Given, **a**, **b** and **N**, st.
>
> $$\mathbf{a = b^t \bmod N}$$
>
> find **t.**

RSA is based on factoring. Diffie-Hellman key exchange, El Gamal and elliptic curve cryptography rely on discrete logarithm being a hard problem.

Circuit for Discrete Logarithm



Measurement of the second register reveals: **k/r**
Measurement of the first register reveals: **kt mod r/r**

Note: same k!

At least in principle we can know **r** by Shor's factoring algorithm, so only t is unknown, and can easily found:

$k^{-1}kt$ = t mod r

Hidden Subgroup Problems

The generalisation of Shor's algorithm to arbitrary groups is known as the Hidden Subgroup Problem:

> Let G be a group. Suppose a subgroup H<G is implicity defined by f on G st f is a constant (and distinct) on every coset of H. Find the generators of H.

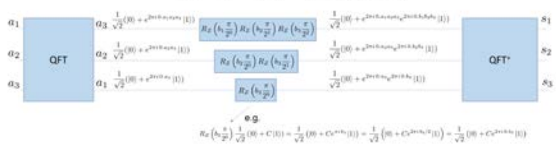
Simon's algorithm and Shor's algorithm are examples of Hidden Subgroup Problems (HSPs).

Addition with QFT

Now we need to build the basic arithmetical operations to implement Shor's algorithm.

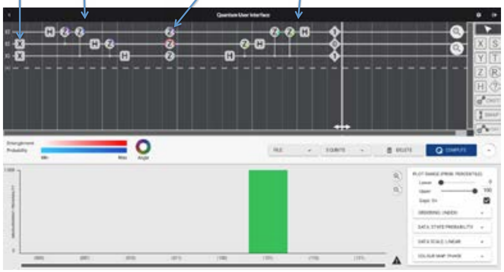
Addition using the QFT (more in Lab-5):

$$a = a_1 2^{n-1} + a_2 2^{n-2} + \ldots + a_{n-2}2 + a_n$$
$$b = b_1 2^{n-1} + b_2 2^{n-2} + \ldots + b_{n-2}2 + b_n$$
$$s = a + b = s_1 2^{n-1} + s_2 2^{n-2} + \ldots + s_{n-2}2 + s_n$$



Addition (3+2) using QFT

$3=011_2$ QFT Phase rotations representing 2 Inverse QFT



Addition using QFT: First qubit

$$|j_1,\ldots,j_n\rangle \to \frac{|0\rangle + e^{2\pi i\; 0.j_n}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i\; 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}} \otimes \ldots \otimes \boxed{\frac{|0\rangle + e^{2\pi i\; 0.j_1 j_2 \cdots j_{n-1}j_n}|1\rangle}{\sqrt{2}}}$$

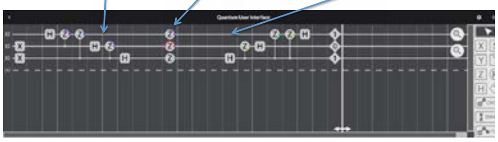
Local phase as a fraction of 2π

0.011 = 3/8

Rotation by
0.010 = 2/8

Total phase rotation:
5/8 = 0.101

Same as for 5=101!



Addition using QFT: Second qubit

$$|j_1,\ldots,j_n\rangle \to \frac{|0\rangle + e^{2\pi i\; 0.j_n}|1\rangle}{\sqrt{2}} \otimes \boxed{\frac{|0\rangle + e^{2\pi i\; 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}}} \otimes \ldots \otimes \frac{|0\rangle + e^{2\pi i\; 0.j_1 j_2 \cdots j_{n-1}j_n}|1\rangle}{\sqrt{2}}$$

Leave off the first qubit!

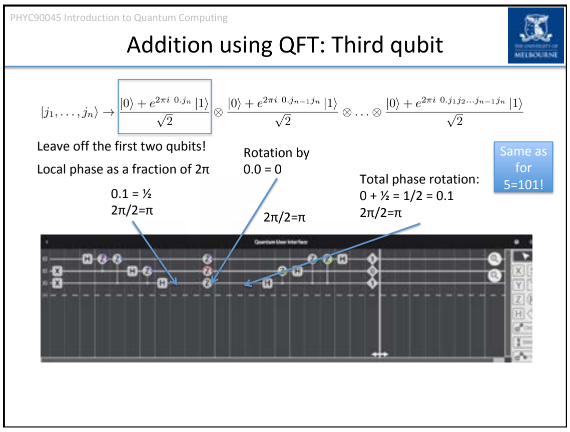
Local phase as a fraction of 2π

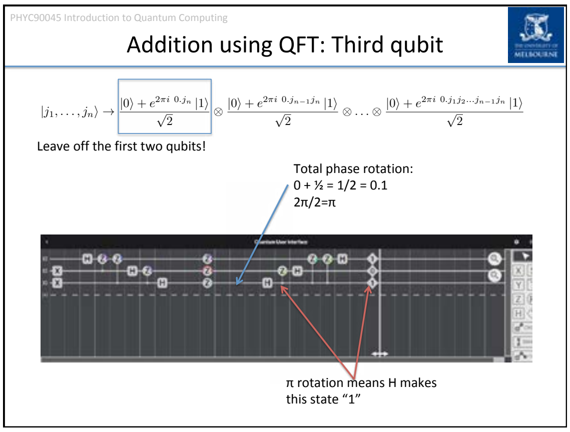
0.11 = 3/4

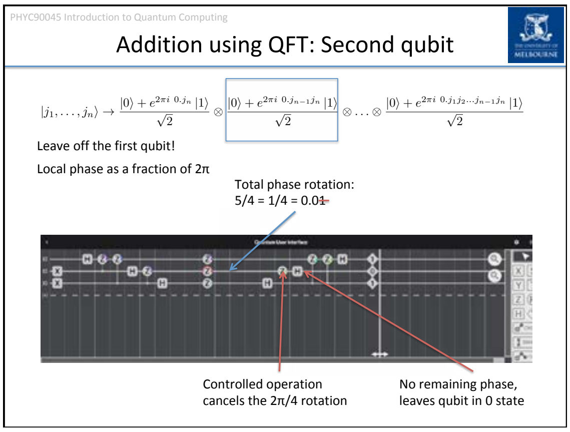
Rotation by
0.10 = 2/4

Total phase rotation:
5/4 = 1/4 = 0.01

Same as for 5=101!


PHYC90045 Introduction to Quantum Computing
Addition using QFT: Third qubit
$|j_1,\ldots,j_n\rangle \rightarrow \boxed{\frac{|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle}{\sqrt{2}}} \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}} \otimes \ldots \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_1j_2\ldots j_{n-1}j_n}|1\rangle}{\sqrt{2}}$
Leave off the first two qubits!
Local phase as a fraction of 2π
0.1 = ½
2π/2=π
Rotation by
0.0 = 0
2π/2=π
Total phase rotation:
0 + ½ = 1/2 = 0.1
2π/2=π
Same as for 5=101!


PHYC90045 Introduction to Quantum Computing
Addition using QFT: Third qubit
$|j_1,\ldots,j_n\rangle \rightarrow \boxed{\frac{|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle}{\sqrt{2}}} \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}} \otimes \ldots \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_1j_2\ldots j_{n-1}j_n}|1\rangle}{\sqrt{2}}$
Leave off the first two qubits!
Total phase rotation:
0 + ½ = 1/2 = 0.1
2π/2=π
π rotation means H makes this state "1"


PHYC90045 Introduction to Quantum Computing
Addition using QFT: Second qubit
$|j_1,\ldots,j_n\rangle \rightarrow \frac{|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle}{\sqrt{2}} \otimes \boxed{\frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}}} \otimes \ldots \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_1j_2\ldots j_{n-1}j_n}|1\rangle}{\sqrt{2}}$
Leave off the first qubit!
Local phase as a fraction of 2π
Total phase rotation:
5/4 = 1/4 = 0.0~~1~~
Controlled operation cancels the 2π/4 rotation
No remaining phase, leaves qubit in 0 state

PHYC90045 Introduction to Quantum Computing

Addition using QFT: First qubit

$$|j_1,\ldots,j_n\rangle \to \frac{|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}} \otimes \ldots \otimes \boxed{\frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \ldots j_{n-1} j_n}|1\rangle}{\sqrt{2}}}$$

Local phase as a fraction of 2π

Total phase rotation:
5/8 = 0.101

Gives the answer, 3+2=5

Controlled operation cancels the 2π/8 rotation

Remaining π phase, leaves qubit in 1 state

PHYC90045 Introduction to Quantum Computing

Multiplier

$$\begin{aligned} a \cdot b &= (a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_2 4 + a_1 \cdot 2 + a_0 \cdot 1)b \\ &= a_n 2^n b + a_{n-1} 2^{n-1} b + \ldots + a_2 4b + a_1 2b + a_0 b \end{aligned}$$

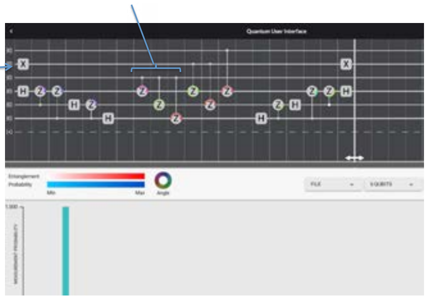
Add 2^nb iff a_n=1. Key idea: Use a_n as a control qubit for addition

PHYC90045 Introduction to Quantum Computing

Multiplication (2x3) using QFT

Add 3 iff the ones bit is a 1

2 = 010_1

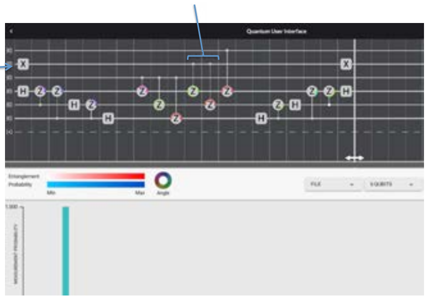


PHYC90045 Introduction to Quantum Computing

Multiplication (2x3) using QFT

Add 6 iff the twos bit is a 1

2 = 010$_2$

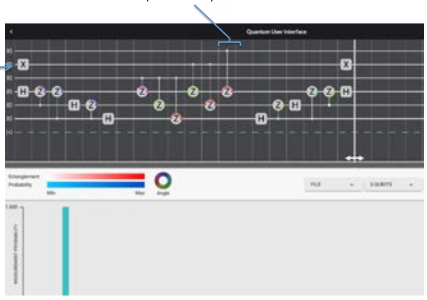


PHYC90045 Introduction to Quantum Computing

Multiplication (2x3) using QFT

Add 4 (=12 mod 8) iff the fours bit is a 1

2 = 010$_2$

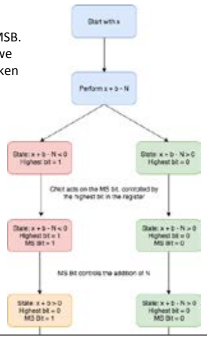


PHYC90045 Introduction to Quantum Computing

Modular adder

For **modular** arithmetic:

Add an extra qubit, the MSB. This qubit keeps track if we are >N. Care has to be taken to reset it unitarily.

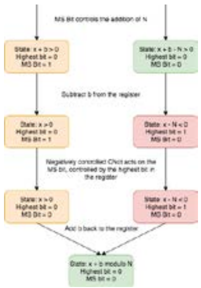


x+b<N

x+b>N

PHYC90045 Introduction to Quantum Computing

Modular adder



Correct sum, but MSB is not reset!

PHYC90045 Introduction to Quantum Computing

This Week

Lecture 9
Fourier Transformations, Regular Fourier Transform, Fourier Transform as a matrix, Quantum Fourier Transform, QUI examples, Inverse QFT

Lecture 10
Shor's Quantum Factoring algorithm, Shor's algorithm for factoring and discrete logarithm, HSP Problem, QFT Arithmetic

Lab 5
QFT and Shor's algorithm