# G53NSC and G54NSC Non-Standard Computation

Dr. Alexander S. Green

16th of March 2010

# Introduction

- Last week we looked a little at Simon's algorithm
- Simon's algorithm is an example of a period finding algorithm...
- finding the period $r$ for a function $f$ defined such that $f(x \oplus r) = f(x)$
- The period $r$ can be extracted from the superposition $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus r\rangle)$ using Hadamard rotations.
- We then started looking at Shor's factorisation algorithm...
- which it turns out is also a period finding problem

# Introduction

- If we can calculate the period $r$ of a function $f(x) = b^x(modN)$, we can factorise $N$...
  - where $b$ is a random integer coprime to $N$
  - and the period has two special properties
  - (which it will have with probability at least $\frac{1}{2}$ for a random choice of $b$)
- So, there are two things we need to be able to do...
- First, we need to construct a unitary that can calculate $f(x) = b^x(modN)$ over a quantum state
- Second, we need to be able to somehow extract the period $r$ from the superposition $\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0 + kr\rangle$
  - Where $m$ is the smallest integer such that $mr + x_0 \geqslant 2^n$

# Introduction

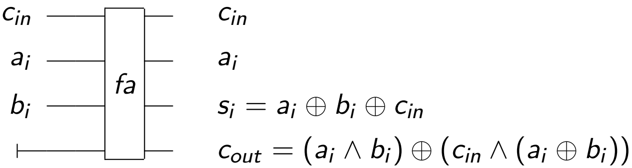
- So, can we do these two things?
- Yes, and we shall be looking at how to do them in the rest of today's lecture
- The first problem is just an exercise in reversible arithmetic circuits
- The second problem uses the quantum Fourier transform
- So we shall look at Fourier transforms, and the quantum Fourier transform in some detail
- We shall also go over a specific example of Shor's algorithm for $N = 15$

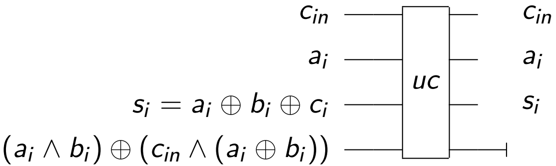
# Part I

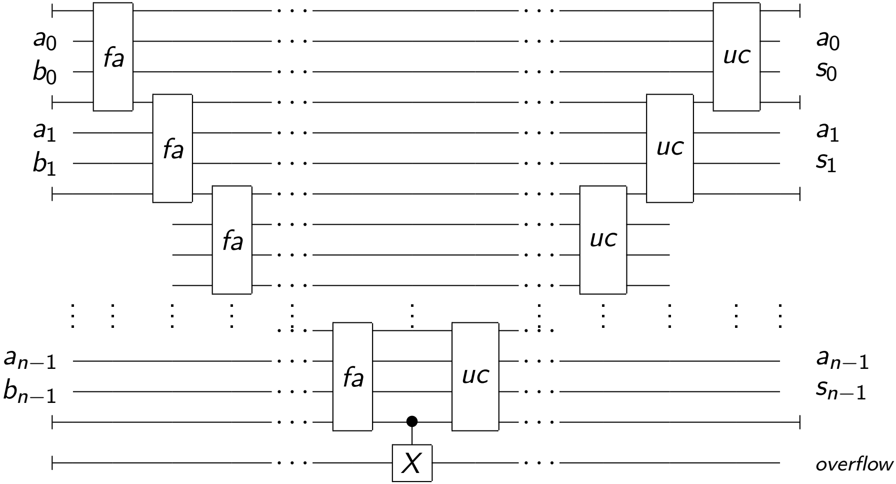
# Reversible Arithmetic

# Addition and Subtraction

- We have already seen a circuit that performs addition in a reversible manner...
- and how we can model it using *QIO* (exercise sheet 2)
- Although we only used the circuit with classical states, using *runC* and the classical subset of *QIO*
- It shouldn't be a surprise that we can use the same circuit (or unitary operator) over a quantum state.
- Lets have a look...

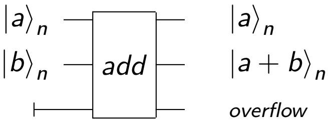

$c_{in}$
$a_i$
$b_i$
$fa$
$c_{in}$
$a_i$
$s_i = a_i \oplus b_i \oplus c_{in}$
$c_{out} = (a_i \wedge b_i) \oplus (c_{in} \wedge (a_i \oplus b_i))$


$c_{in}$
$a_i$
$s_i = a_i \oplus b_i \oplus c_i$
$(a_i \wedge b_i) \oplus (c_{in} \wedge (a_i \oplus b_i))$
$uc$
$c_{in}$
$a_i$
$s_i$


$a_0$
$b_0$
$a_1$
$b_1$
$a_{n-1}$
$b_{n-1}$
fa
fa
fa
fa
uc
uc
uc
uc
X
$a_0$
$s_0$
$a_1$
$s_1$
$a_{n-1}$
$s_{n-1}$
overflow

# Quantum Arithmetic

- We can simplify the notation for the reversible addition circuit




- We can think of this as the unitary $adder :: [Qbit] \rightarrow [Qbit] \rightarrow Qbit \rightarrow U$ in $QIO$
- and note that we get subtraction for free...
- What happens if $|a\rangle$ or $|b\rangle$ are quantum states?

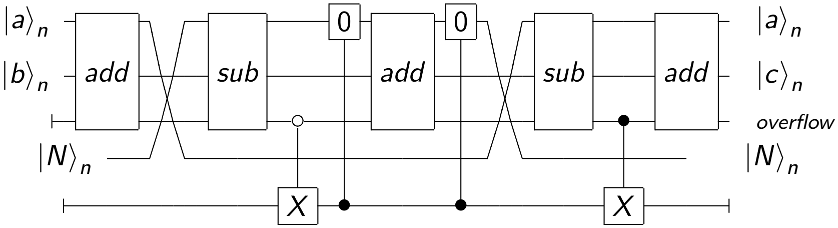
# Quantum Arithmetic

- For example, if the input $|a\rangle = \frac{1}{\sqrt{2}}(|2\rangle + |3\rangle)$ and the input $|b\rangle = \frac{1}{\sqrt{2}}(|4\rangle + |5\rangle)$
- What is the output?
- We end up with four additions taking place in parallel...
  - $|2+4\rangle = |6\rangle$
  - $|2+5\rangle = |7\rangle$
  - $|3+4\rangle = |7\rangle$
  - $|3+5\rangle = |8\rangle$
- We are left with the superposition $\frac{1}{2}(|6\rangle + 2\,|7\rangle + |8\rangle)$
- Any arithmetic circuits we define in the classical subset of *QIO* can be used over a quantum state.

# Quantum Arithmetic

- So, can we construct a unitary that performs modular exponentiation as required in Shor's algorithm?
- In fact, there's only three steps to get from reversible addition to reversible modular exponentiation...
  - We can use reversible addition to construct reversible modular addition
  - We can use reversible modular addition to construct reversible modular multiplication
  - We can use reversible modular multiplication to construct reversible modular exponentiation
- Lets look at some circuits...
- They are taken from the paper "quantum networks for elementary arithmetic operations" that is linked from the module web page

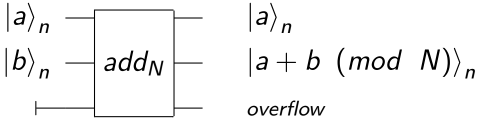
# Modular addition



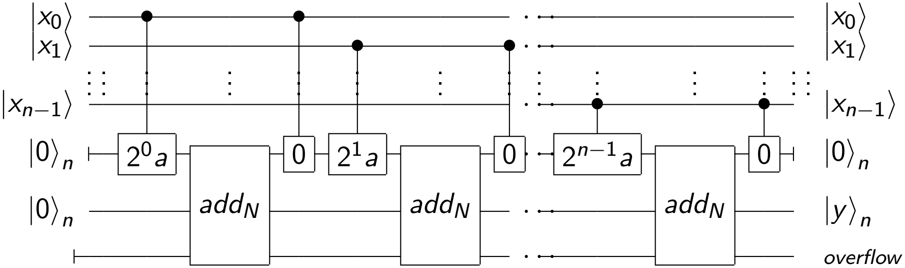

- Where $c = a + b \pmod{N}$

# Modular addition

- For Shor's algorithm, we only need $N$ to be a classical value...
- the number we are trying to factor
- We can define the circuit for reversible addition modulo $N$, with $N$ as an implicit argument



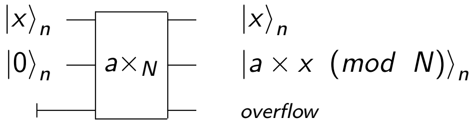
# Modular multiplication




- Where $y = a \times x \pmod{N}$
- Again, the argument $a$ can be classical, and built implicitly into the circuit.

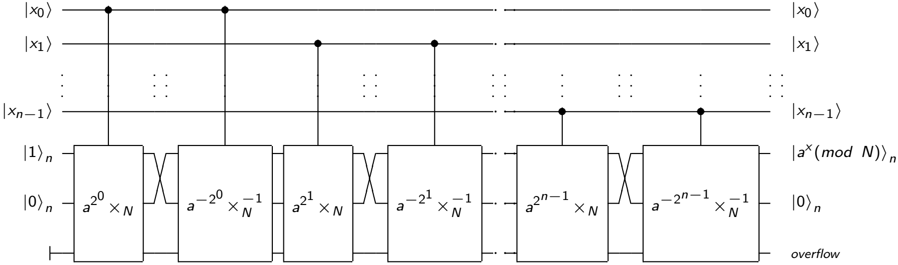
# Modular multiplication

- We can simplify the notation for multiplication by $a$ modulo $N$



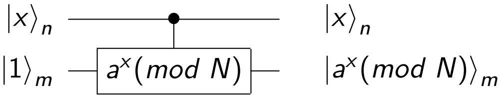

- Using a controlled version of modular multiplication, we can now construct the necessary modular exponentiation unitary.

# Modular exponentiation



# Modular exponentiation

- The powers of $a$ can be calculated (efficiently) classically
- We can choose $n$ so that overflow isn't a problem and can be ignored
- Giving us a reversible modular exponential function, that can be used over a quantum state as required by Shor's algorithm



# Part II

# The Quantum Fourier Transform

# Fourier transforms

Joseph Fourier

- A French mathematician and physicist
- Discovered what we now call Fourier series
- The Fourier transform is named in his honour

# Fourier transforms

- Fourier showed that periodic functions could be decomposed in to the sum of simple sine and cosine functions
- The Fourier transform is able to calculate such a decomposition
- It is often said to take functions from the *time domain* to the *frequency domain*
- Fourier transforms are already used extensively in classical computation
- But even the fastest implementation (known as the Fast Fourier Transform) is exponential in its arguments
- Its uses include...
  - Image compression (JPEG)
  - Audio compression
  - Noise reduction techniques

# The discrete Fourier transform

- The disrete Fourier transform takes a set of Complex arguments to a set of Complex results
- The Fourier transform from the vector $x_0, \ldots, x_{N-1}$ to the vector $y_0, \ldots, y_{N-1}$ is defined by

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi ijk}{N}}$$

- If we look at Euler's formula, we can see how this is a sum of sine and cosine functions

$$e^{i\theta} = cos\theta + i\ sin\theta$$

- Note that each element of $y$ is calculated using every element of $x$

# The quantum Fourier transform

- The quantum Fourier transform is the discrete Fourier transform applied to the amplitudes of a quantum state
- It is a unitary transform, which we will prove by providing a circuit made from unitary gates
- It is also efficient, with the number of gates required only polynomial in the number of qubits
- It doesn't provide an efficient means for calculating an arbitrary Fourier transform as it only acts on the amplitudes of the quantum state, and not the states themselves
- It has a similar definition to the discrete Fourier transform

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle$$

- We can simplify matters by restricting $N = 2^n$

# The quantum Fourier transform

- We can rewrite the quantum Fourier transform in terms of the binary expansion of $j$

$$j = j_0 2^{n-1} + j_1 2^{n-2} + \ldots + j_{n-1} 2^0$$

- and similar notation for a binary fraction of $j$

$$0.j_l j_{l+1} \ldots j_m = \frac{j_l}{2} + \frac{j_{l+1}}{4} + \ldots + \frac{j_m}{2^{m-l+1}}$$

- Such that the quantum Fourier transform can be given by

$$|j_0, j_1, \ldots, j_{n-1}\rangle \rightarrow \frac{(|0\rangle + e^{2\pi i 0.j_{n-1}}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_{n-2} j_{n-1}}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2\pi i 0.j_0 j_1 \cdots j_{n-1}}|1\rangle)}{2^{\frac{n}{2}}}$$

- The full derivation of this product representation can be found in the Nielsen and Chuang book (p.218)

# The quantum Fourier transform

- It is actually suggested that you treat this as the definition of the quantum Fourier transform
- The output state of each qubit is an equal superposition of $|0\rangle$ and $|1\rangle$, with a phase applied to the $|1\rangle$ part that depends on the input states.
- The phase of each qubit in the output state depends on one more member of the input state than the previous qubit...
- E.g. the state of the first output qubit $(\frac{1}{2^{\frac{n}{2}}}(|0\rangle + e^{2\pi i 0.j_n}|1\rangle))$ only depends on upon the last input qubit
- and the state of the second output qubit $(\frac{1}{2^{\frac{n}{2}}}(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle))$ depends on the last two input qubits...

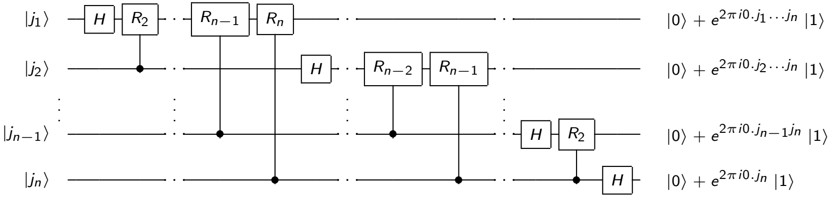
# The quantum Fourier transform

- How can we construct a circuit that performs this operation?
- We can use a Hadamard to create the equal superposition, and then perform controlled phase operations depending on the necessary input qubits
- E.g. the Hadamard gate takes an input qubit $|j_m\rangle$ to the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_m} |1\rangle)$
- and we can create a controlled version of the phase rotation

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

- The following circuit performs the quantum Fourier transform, although the output qubits are in reverse order

# The quantum Fourier transform




- The output can be reversed using swap operations
- We also get the inverse quantum Fourier transform for free
- Which is what we use in Shor's algorithm

# Using the QFT

- How can we use the quantum Fourier transform to extract the period of the function $b^x (mod\ N)$?
- We use a procedure known as phase estimation
- If we have a superposition of states

$$\frac{1}{2^{\frac{t}{2}}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi j} |j\rangle_n |u\rangle_m$$

- we can apply the inverse quantum Fourier transform and get the state

$$|\hat{\varphi}\rangle_n |u\rangle_m$$

- Where $\hat{\varphi}$ is an approximation of $\varphi$ to $n$ bits

# QFT and Shor's algorithm

- This is used in Shor's algorithm by noticing

$$\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle\,|x^j modN\rangle \approx \frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1}e^{2\pi ij\frac{s}{r}}\,|j\rangle\,|u_s\rangle$$

- Where

$$|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{\frac{-2\pi isk}{r}}\,|x^k modN\rangle$$

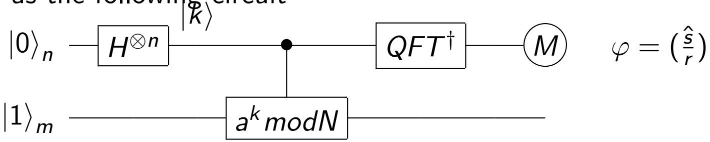
- Applying the inverse quantum Fourier transform to this state will give

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|(\frac{\hat{s}}{r})\rangle\,|u_s\rangle$$

- so we can measure an integer result $\varphi = (\frac{\hat{s}}{r})$

# The continued fractions algorithm

- We know $\varphi$ upto $n$ bits, and can use the continued fractions algorithm for $\frac{\varphi}{2^n}$ to calculate $r$
- The continued fractions algorithm is an efficient classical algorithm
- So, to recap, the quantum part of Shor's algorithm can be given as the following circuit




- the convergent of the continued fraction for $\frac{\varphi}{2^n}$ gives $\frac{s}{r}$
- As we only know $\varphi$ to $n$ bits, we need to choose $n$ as at least $2m + 1 + \lceil log(2 + \frac{1}{2\epsilon})\rceil$ so we have probability $1 - \epsilon$ of finding $r$, where $m$ is the number of bits needed to specify $N$

# Factorising 15

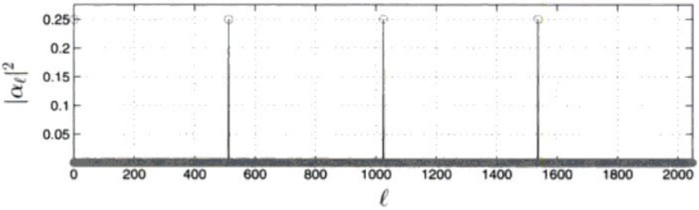
- Lets look at a concrete example...
- Factorising $N = 15$ using Shor's algorithm
- Step 1: pick a random number ($b$) that is coprime to 15
- This could be any of $2, 4, 7, 8, 11, 13,$ or 14
- Lets choose $b = 7$
- Step 2: construct the unitary for the function $f(x) = 7^x (mod\ 15)$
- To do this, we need to specify $m$ and $n$, the number of qubits in each register.
- $m$ is the number of bits needed to specify $N$, so $m = 4$
- $n = 2m + 1 + \lceil log(2 + \frac{1}{2\epsilon}) \rceil$
- For an error of at most $\epsilon = \frac{1}{4}$ we can calculate $n = 2 * 4 + 1 + \lceil log(2 + \frac{1}{\frac{2}{4}}) \rceil = 8 + 1 + 2 = 11$

# Factorising 15

- Step 3: Apply the unitary $f(x) = 7^x (mod\ 15)$ to the state $\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle_n |0\rangle_m$
- leaving the state $\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle_n |7^k (mod\ 15)\rangle_m$
- $= \frac{1}{\sqrt{2^n}}[|0\rangle|1\rangle+|1\rangle|7\rangle+|2\rangle|4\rangle+|3\rangle|13\rangle+|4\rangle|1\rangle+\ldots+|2047\rangle|13\rangle]$
- Step 4: Measure the second register
- This will leave (with equal probability) either $1, 7, 4$, or $13$
- In this instance, we measured a 4
- Leaving the state $\sqrt{\frac{4}{2^n}}[|2\rangle + |6\rangle + \ldots + |2046\rangle]$
- Step 5: Apply the inverse QFT and measure the result

# Factorising 15

- The inverse QFT leaves the following probability distribution




- We can see that each of the states $0, 512, 1024$, or $1536$ could be measured with almost probability $\frac{1}{4}$ each
- In this instance, we measured $\varphi = 1536$
- We can now calculate $r$, as $\frac{s}{r}$ is the convergent of the continued fraction for $\frac{1536}{2048}$

# Factorising 15

- The convergent is $\frac{3}{4}$ so we have $r = 4$
- Step 6: Check properties of $r$, and calculate the factors
  - 4 is even, so that is ok
  - $x = 7^{\frac{4}{2}} (mod 15) = 4$
  - $x + 1 \neq 0 (mod 15)$, so that is ok
- Factors are $gcd(x - 1, 15)$ and $gcd(x + 1, 15)$
  - $gcd(x - 1, 15) = gcd(3, 15) = 3$
  - $gcd(x + 1, 15) = gcd(5, 15) = 5$
- So, the factors of 15 are 3 and 5

# Thank you

- Remember, presentations start next week...
- Attendance will be taken!
- Submission deadline for your paper is 12:00 (midday) this Friday
- I will make copies of all the papers available after this
- Labs on Thursday as usual...
- Thank you.