ET4340 Electronics for Quantum Computing
Homework 4

Mick van Gelderen
4091566

November 2013

Problem 1: Quantum Fourier Transform

1. Write the 8×8 matrix (in the computational basis) corresponding to the quantum Fourier transform on a 3-qubit register.

The matrix $U_{QFT,8}$ is defined as:

$$U_{QFT,8} = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} e^{\frac{i2\pi lk}{N}} |l\rangle \langle k|$$

where $N = 8$.

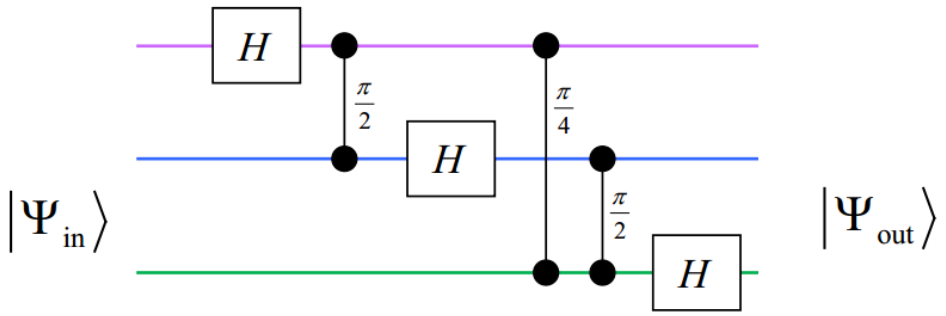
$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \sqrt{2}+\sqrt{2}i & i & -\sqrt{2}+\sqrt{2}i & -1 & -\sqrt{2}-\sqrt{2}i & -i & \sqrt{2}-\sqrt{2}i \\
1 & i & -1 & -i & 1 & i & -1 & -i \\
1 & -\sqrt{2}+\sqrt{2}i & -i & \sqrt{2}+\sqrt{2}i & -1 & \sqrt{2}-\sqrt{2}i & i & -\sqrt{2}-\sqrt{2}i \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & -\sqrt{2}-\sqrt{2}i & i & \sqrt{2}-\sqrt{2}i & -1 & \sqrt{2}+\sqrt{2}i & -i & -\sqrt{2}+\sqrt{2}i \\
1 & -i & -1 & i & 1 & -i & -1 & i \\
1 & \sqrt{2}-\sqrt{2}i & -i & -\sqrt{2}-\sqrt{2}i & -1 & -\sqrt{2}+\sqrt{2}i & i & \sqrt{2}+\sqrt{2}i
\end{bmatrix}$$

2. Show that this matrix is unitary

Use a computer to calculate $U_{QFT}U_{QFT} = I$. You can easily see that the matrix is symmetric so this is the only thing we have to show. Matlab indeed gives I with some near zero imaginary parts.

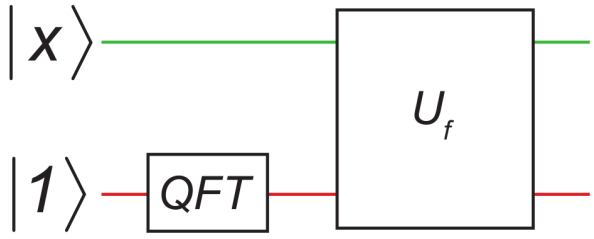
3. Draw the quantum circuit that implements this QFT

From the lecture slides where it was drawn for 4 qubits:



Problem 2: Generalized quantum kick-back

In the lectures, we have seen how the Deutsch and Bernstein-Vazirani quantum games exploit quantum kick-back to efficiently extract properties of n-to-1 bit boolean functions. In this problem, we generalize quantum kick-back to n-to-m bit boolean functions encoded in unitary functions as usual: $U_f |x\rangle |y\rangle = |x\rangle |(y + f(x)) \bmod M\rangle$ for computational states $|x\rangle$ and $|y\rangle$ in the top and bottom registers, respectively. Consider the circuit below.



1. What is the state of the bottom register after application of the m-bit QFT on the initial state $|1\rangle = |00000...1\rangle$?

$U_{QFT} |1\rangle$ equals the last column of U_{QFT}. So $U_{QFT} |1\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{i2\pi(N-1)k}{N}} |k\rangle$

2. Now apply U_f, with the top register starting in a computational state $|x\rangle$. What is the combined state of the top and bottom registers immediately after U_f? Show that this this state can be rewritten as:

$$e^{\frac{-i2\pi f(x)}{M}} |x\rangle \otimes U_{QFT} |1\rangle ,$$

where $M = 2^m$. Evidently, the top and bottom registeres are not entangled, and $f(x)$ is encoded in the quantum phase of the probability amplitude!

The final state is:

$$|x\rangle \otimes U_f |x\rangle |y\rangle = |x\rangle \otimes |x\rangle |(y + f(x)) \bmod M\rangle$$
$$|x\rangle |(y + f(x)) \bmod M\rangle = |x\rangle |y\rangle + |x\rangle |f(x)\rangle \bmod M$$

Argh. We probably need a lecture on how to do these function encoded multi bit bullshit things, I don't have a clue how to rewrite this simply because of the notation and its angering me because I'm not able to do it.

3. Finally, consider the case that the top register is initialized in the maximal superposition state $\frac{1}{\sqrt{N}}(|0\rangle + \cdots + |N-1\rangle)$. As usual, $N = 2^n$. What will be the final state after application of U_f?

Since $|x\rangle$ in maximal superposition is a vector of all ones, $e^{\frac{-i2\pi f(x)}{M}}$ is applied to all k where $0 \leq k < N$. The final state is $\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{-i2\pi f(k)}{M}} \otimes U_{QFT} |1\rangle$.

Problem 3: Breaking RSA

In this exercise, we will break RSA by period finding. N will be small enough that we will find periods by brute force.

1. List the integers $a < N$ that are co-prime with N. Let us pick one of these integers: let us agree to all 'randomly' pick $a = 8$.

I guess $N = 21$.

n	divisible by	co-primes
2	2	1
3	3	1 2
4	2 4	1 3
5	5	1 2 3 4
6	2 3 6	1 5
7	7	1 2 3 4 5 6
8	2 4 8	1 3 5 7
9	3 9	1 2 4 5 7 8
10	2 5 10	1 3 7 9
11	11	1 2 3 4 5 6 7 8 9 10
12	2 3 4 6 12	1 5 7 11
13	13	1 2 3 4 5 6 7 8 9 10 11 12
14	2 7 14	1 3 5 6 9 11 13
15	3 5 15	1 2 4 7 8 11 13 14
16	2 4 8 16	1 3 5 7 9 11 13 15
17	17	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18	2 3 6 9 18	1 5 7 11 13 17
19	19	1 3 5 7 8 9 10 11 12 13 14 15 16 17 18 19
20	2 4 5 10 20	1 3 7 9 11 13 17 19
21	3 7 21	1 2 4 5 8 10 11 13 16 17 19

2. Compute $8^0 \bmod 21$, $8^1 \bmod 21$, ... until you discover the period r of $f(x) = 8^x \bmod 21$.

n	$8^n \bmod 21$
0	1
1	8
2	1

The period seems to be 2.

3. Find the greatest common denominator (gcd) between, $8^{r/2} + 1$ and 21. Check whether the result is a prime factor of 21.

Since $r = 2$, $8^{r/2} + 1 = 9$. The gcd of 9 and 21 is 3 which is indeed a prime factor of 21.

4. Similarly, find the gcd between $8^{r/2} - 1$ and 21.

The gcd of 7 and 21 is 7 which is a prime factor of 21.

5. Repeat the process for $a = 10$.

Finding the period:

n	$10^n \bmod 21$
0	1
1	10
2	6
3	13
4	4
5	19
6	1

The period seems to be 6. The gcd's of $10^3 \pm 1$ and 21 are 3 and 7, magic!

Problem 4: Breaking RSA with Shor's algorithm

Now we will go through the steps of Shor's algorithm in order to find the period r of $8^x \bmod 21$. For simplicity, lest us use just four qubits for the top register which is enough for our choice of $a = 8$. *Note: Please use decimal bra-ket notation instead of binary.*

1. Initialize each register to $|0\rangle$.

Cool it down!

2. Apply the hadamard gate to each qubit in the top register.

The top register ends up in a state that is a combination of all possible 4 qubit computational states: $\frac{1}{\sqrt{16}}(|0\rangle + |1\rangle + \cdots + |15\rangle)$.

3. Add $8^x \bmod 21$ to the bottom register where x is the state of the top register.

We calculate $8^x \bmod 21$ for $x \in \{0, 1, \ldots 15\}$.

x	$8^x \bmod 21$	x	$8^x \bmod 21$
0	1	8	1
1	8	9	8
2	1	10	1
3	8	11	8
4	1	12	1
5	8	13	8
6	1	14	1
7	8	15	8

So, y must be $\frac{1}{\sqrt{16}}(|1\rangle + |8\rangle + |1\rangle + |8\rangle \ldots |1\rangle + |8\rangle)$.

4. Rewrite this state so you group all terms with identical $f(x)$ — observe the periodicity in the amplitudes which emerges, and observe also that you cannot efficiently reveal the period by any measurement.

This means that the y register is in the state $\frac{1}{\sqrt{2}}(|1\rangle + |8\rangle)$.

For this simple example you might be able to measure 100 times. If you measure $|1\rangle$ 31 times and $|8\rangle$ 69 times it is probable that the period is 2 because you only measured 2 different outcomes. In general, there is a chance that if you measure k times you will have not measured one or more possible outcomes which causes you to use an r that is too low. For bigger N, the amount of measurements k becomes very, very large to have confidence in your estimation of r.

Using the QFT is far more efficient.

5. Apply the Quantum Fourier Transform to the top register.

We apply the Quantum Fourier Transform to the top register after measurement of the bottom register.

U_{QFT} can be computed in Matlab with the following simple code:

```
function QFT = qft(n_qubits)
%qft Quantum Fourier Transform
%   Computes the unitary QFT matrix for a given number of
    qubits.

N = 2^n_qubits;

QFT = zeros(N,N);
for k = 0:(N - 1)
  for l = 0:(N - 1)
    QFT(k + 1, l + 1) = exp(2i*pi*k*l/N);
  end
```

```
end

QFT = QFT ./ sqrt(N);
```

The effects of measurement can be simulated by simply only keeping the indices of the input states for which the outcome equals that measurement. This is done in the following code which also plots the state of the top register (x) before and after the QFT transformation.

```
N = 16;
x1 = (0:N - 1)'; % x register in maximal superposition
y = mod(8.^x1, 21); % y register after modular exponentiation

QFT4 = qft(4);

outcomes = unique(y);
noutcomes = length(outcomes);
X = zeros(length(y), noutcomes);
for n = 1:noutcomes
  x2 = y==outcomes(n);
  x2 = x2 ./ sqrt(sum(x2)); % x register after y measurement
  X(:,n) = x2;

  figure
  subplot(1,2,1); stem(x1, x2);
  title(['outcome = ' int2str(outcomes(n))]);
  subplot(1,2,2); stem(x1, abs(QFT4*x2));
  title(['QFT4*x']);
end
```

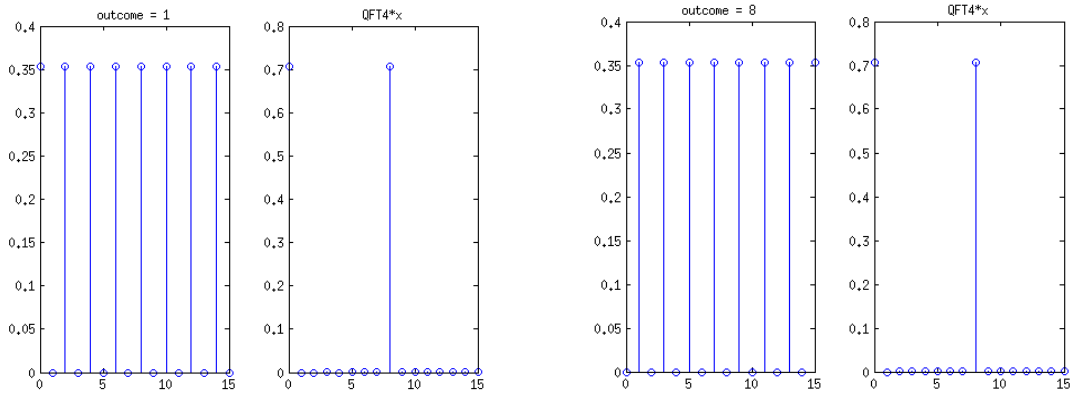
The above code produces the following two plots:

6. Measure the final state of the top register. What are the possible measurement outcomes, and how do they relate to r?

The final states of x seem to be equal when measuring both $|1\rangle$ and $|8\rangle$ for y. So independently of the y measurement (in this case) you will have a 50% chance of measuring $x = |0\rangle$ and a 50% chance of measuring $x = |8\rangle$.

If the final state of x is $|s\rangle$ we can try continued fractions of prime numbers of $\frac{s}{2^{|x|}}$ and see if the denominator is a valid r. Obviously, when $s = 0$ you get no information about r.

In our example we could measure $s = 8$ as well, this gives 8/16. This can be simplified to 1/2 which is a fraction of two primes. This means $r = 2$ which happens to be correct.



(a) Measured $|1\rangle$ (b) Measured $|8\rangle$

Figure 1: Shors algorithm for 8^x mod 21

After finding a possible period, we compute $\gcd(a^{r/2} \pm 1, N)$. In our case this amounts to finding $\gcd 7, 21 = 7$ and $\gcd 9, 21 = 3$. These values are primes p and q for which $pq = N$.

7. What fraction of the times that you run the algorithm above will you successfully find the factors of 21?

For $a = 8$, the only final state that gives us enough information to find p and q is $|8\rangle$. This state occurs 50% of the time.

8. What complication occurs for $a = 10$? Discuss how Shor deals with it. *Please answer this question in just a few sentences.*

For $a = 10$ the QFT introduces a lot of leakage. This lowers the chance of finding an s that is of use in finding r. Shor simply suggests to run the algorithm multiple times until you find a better s.

Problem 5: (EXTRA CREDIT) Tensor product matrices and product states

A general tensor product of two one qubit unitaries A and B is given by:

$$U = A \otimes B = \begin{bmatrix} A_{11}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} & A_{12}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \\ A_{21}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} & A_{22}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \end{bmatrix}$$

1. Show that a matrix of this form always takes product states to product states.

This property may be intuitive to you or it may not, the proof is actually not that hard. Lets first write U a little simpler:

$$U = A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B \\ A_{21}B & A_{22}B \end{bmatrix}$$

A product state can always be written as $|\psi\rangle \otimes |\phi\rangle$. Since vectors are essentially matrices where at least one of the dimensions is 1, the tensor product can also be written in expanded form:

$$|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \psi_1 |\phi\rangle \\ \psi_2 |\phi\rangle \end{bmatrix}$$

If we can show that $U |\psi\rangle \otimes |\phi\rangle = A |\psi\rangle \otimes B |\phi\rangle$ (assuming the $\otimes$ operation is applied before the matrix product operation) we will have proven that the result of the application of the matrix U can still be expressed as a product state.

$$\begin{aligned} A \otimes B |\psi\rangle \otimes |\phi\rangle &= \begin{bmatrix} A_{11}B & A_{12}B \\ A_{21}B & A_{22}B \end{bmatrix} \begin{bmatrix} \psi_1 |\phi\rangle \\ \psi_2 |\phi\rangle \end{bmatrix} \\ &= \begin{bmatrix} A_{11}\psi_1 B |\phi\rangle & A_{12}\psi_1 B |\phi\rangle \\ A_{21}\psi_2 B |\phi\rangle & A_{22}\psi_2 B |\phi\rangle \end{bmatrix} \\ &= \begin{bmatrix} A_{11}\psi_1 & A_{12}\psi_1 \\ A_{21}\psi_2 & A_{22}\psi_2 \end{bmatrix} \otimes B |\phi\rangle \\ &= A |\psi\rangle \otimes B |\phi\rangle \end{aligned}$$

Instead of a mathmatical proof, we can simply look at the way the quantum circuit is represented. U is essentially a two-qubit gate composed of two single qubit gates A and B. This means that it is impossible for the two qubits to interact which in turn means that they cannot become entangled. Since entanglement is required to prevent two qubits from being written as a product state, U cannot possibly prevent the outcome from being written as a product state.

2. Prove that the `CNOT` gate cannot be written in this form by comparing the matrix entries with the general tensor product matrix above.

The transformation matrix for the two qubit `CNOT` is:

$$\texttt{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Lets define A and B as follows:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$B = \begin{bmatrix} w & x \\ y & z \end{bmatrix}$$

From the entire matrix multiplication we can deduce the following contradicting equations:

$$\begin{aligned} aw &= 1 & ax &= 0 \\ dw &= 0 & dx &= 1 \end{aligned}$$

Because $ax = 0$ we must set either a or x to 0. In case we pick a, it becomes impossible to pick w so that $aw = 0w = 1$. In case we pick x it becomes impossible to find a d so that $dx = d0 = 1$.

This means that it is impossible to construct `CNOT` from $A \otimes B$.

This is only natural since the `CNOT` operation requires a qubit to flip depending on the state of the other. In other words, they must interact. This is impossible with two single qubit gates.

3. Show that the `SWAP` gate always takes a product state to another product state.

The transformation matrix for `SWAP` is:

$$\texttt{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Writing out a product state as a vector and applying `SWAP` yields:

$$|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \psi_1\phi_1 \\ \psi_1\phi_2 \\ \psi_2\phi_1 \\ \psi_2\phi_2 \end{bmatrix}$$

$$\texttt{SWAP}\,|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \psi_1\phi_1 \\ \psi_2\phi_1 \\ \psi_1\phi_2 \\ \psi_2\phi_2 \end{bmatrix} = |\phi\rangle \otimes |\psi\rangle$$

4. Can you write the `SWAP` gate in the form of $A \otimes B$? If so give your solution for A and B. If not give a proof.

The proof for this is similar to that of the `CNOT`. This time you have:

$$\begin{aligned} aw &= 1 & ax &= 0 \\ cw &= 0 & cx &= 1 \end{aligned}$$

These equations are also inconsistent.

This is a counter example to the statement 'if a two cubit gate cannot be expressed in single cubit gates, the result of applying the gate to a product state will not yield a product state'.