CMPT 409/981: Quantum Circuits and Compilation
Assignment 2 Solutions

December 1, 2022

Question 1 [3 points]: Concrete resource estimates

We first calculate the number of R_k gates in the n-qubit QFT (using 3 R_{k+1} gates per controlled R_k):

- $\#R_1 = \#R_2 = 0$
- $\#R_3 = 3(n-1)$
- $\#R_{k\geq 4} = \sum_{i=1}^{n-2} 3i = \frac{3(n-2)(n-2+1)}{2} = \frac{3n^2-9n+6}{2}$

For $n = 32$, we have 1395 R_k gates which need to be approximated. As we saw in class, errors add, hence we need to approximate each to precision $\epsilon/1395$. We expect that each approximation should require

$$3\log_2(1395/\epsilon) = 3\log_2(1395 * 10^7) \approx 101$$

T gates. giving a total T-count of somewhere in the ballpark of

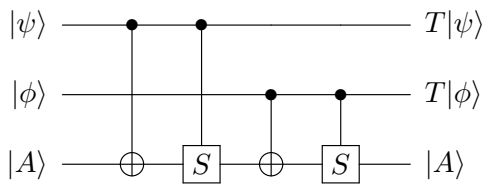
$$3(32-1) + 1395 * 101 = 140988$$

Question 2 [10 points]: Catalytic QFT

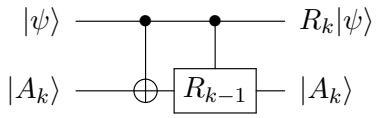
1. (2 points) First suppose $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and recall that $|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \omega|1\rangle)$. Then

$$\begin{aligned}
(CS \cdot CNOT)|\psi\rangle|A\rangle &= CS \cdot CNOT(\alpha|0\rangle \otimes |A\rangle + \beta|1\rangle \otimes |A\rangle) \\
&= CS(\alpha|0\rangle \otimes |A\rangle + \beta|1\rangle \otimes \frac{1}{\sqrt{2}}(\omega|0\rangle + |1\rangle)) \\
&= \alpha|0\rangle \otimes |A\rangle + \beta|1\rangle \otimes \frac{1}{\sqrt{2}}(\omega|0\rangle + i|1\rangle) \\
&= \alpha|0\rangle \otimes |A\rangle + \omega\beta|1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + \omega|1\rangle) \\
&= \alpha|0\rangle \otimes |A\rangle + \omega\beta|1\rangle \otimes |A\rangle \\
&= (T \otimes I)|\psi\rangle|A\rangle
\end{aligned}$$

2. (1 point)



3. (2 points) Observe that



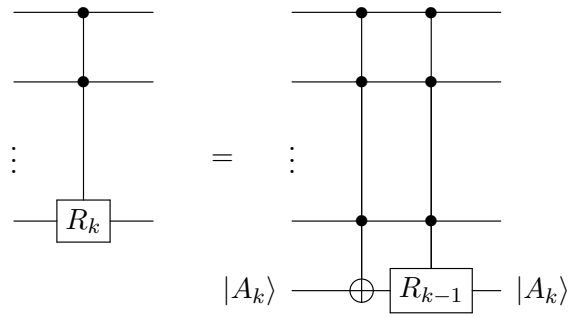
In particular,

$$\begin{aligned}(CR_{k-1}\cdot CNOT)|\psi\rangle|A_k\rangle &= \alpha|0\rangle\otimes|A_k\rangle + \beta|1\rangle\otimes\frac{1}{\sqrt{2}}(e^{2\pi i/2^k}|0\rangle + e^{2\pi i/2^{k-1}}|1\rangle)\\ &= \alpha|0\rangle\otimes|A_k\rangle + e^{2\pi i/2^k}\beta|1\rangle\otimes\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i/2^k}|1\rangle)\\ &= \alpha|0\rangle\otimes|A_k\rangle + e^{2\pi i/2^k}\beta|1\rangle\otimes|A_k\rangle\\ &= (R_k\otimes I)|\psi\rangle|A_k\rangle\end{aligned}$$

4. (1 point) Yes! We could instead measure the ancilla qubit after the $CNOT$ gate and apply a classically-controlled R_{k-1} correction.

5. (1 point) Since CT requires the R_4 gate to implement directly, we can't construct the R_4 gate this way... We'll have to come up with another solution.

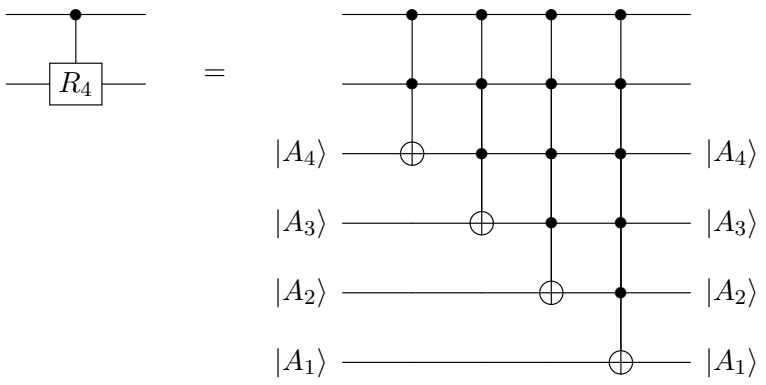
6. (2 points) Observe that



In particular, writing $|\psi\rangle = |\overline{0}\rangle + |\overline{1}\rangle$ where $|\overline{1}\rangle = P_{11\cdots1}|\psi\rangle = |11\cdots1\rangle\langle11\cdots1|\cdot|\psi\rangle$ and $|\overline{0}\rangle$ is the projection $P^{\perp}_{11\cdots1}|\psi\rangle$ of $|\psi\rangle$ onto the orthogonal complement of $P_{11\cdots1}$, we have

$$\begin{aligned}(C^nR_{k-1}\cdot C^nNOT)|\psi\rangle|A_k\rangle &= |\overline{0}\rangle\otimes|A_k\rangle + |\overline{1}\rangle\otimes\frac{1}{\sqrt{2}}(e^{2\pi i/2^k}|0\rangle + e^{2\pi i/2^{k-1}}|1\rangle)\\ &= |\overline{0}\rangle\otimes|A_k\rangle + e^{2\pi i/2^k}|\overline{1}\rangle\otimes\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i/2^k}|1\rangle)\\ &= |\overline{0}\rangle\otimes|A_k\rangle + e^{2\pi i/2^k}|\overline{1}\rangle\otimes|A_k\rangle\\ &= (R_k\otimes I)|\psi\rangle|A_k\rangle\end{aligned}$$

7. (1 point) We have



Note that this is a *doubly-controlled modular decrement* operator on the register $|A_1A_2A_3A_4\rangle$ viewed as a big endian integer. In particular, the circuit will flip every un-set bit started from the right until the first "1", so for instance $|1010\rangle$ would be mapped to $|1001\rangle$ and $|1100\rangle$ to $|1011\rangle$.

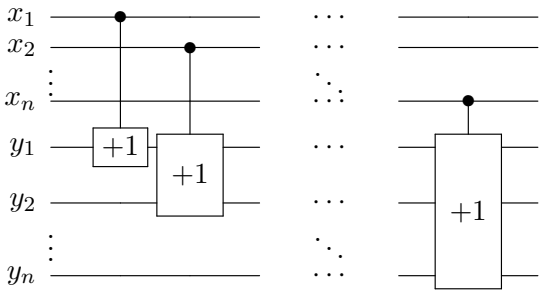
Question 3 [8 points]: Reversible arithmetic

We define the *modular increment function* as

$$(+1) : |\vec{x}\rangle \mapsto |\vec{x} + 1 \mod 2^n\rangle$$

where $\vec{x}$ is taken as a big-endian integer, i.e. x_1 is the high-order bit. That is, $(+1)$ adds 1 to a positive integer represented in big endian binary as an n-bit string $\vec{x} \in \{0,1\}^n$ ignoring overflow (i.e. $\vec{x} + 1 \mod 2^n$). Note that the inverse of a modular increment is a modular decrement.

1. (2 points) Consider the circuit in question,



First we note that adding 1 to a register $|y_1y_2\cdots y_i\rangle$ is equivalent to adding 2^{n-i} to the register $|y_1y_2\cdots y_n\rangle$. Explicitly, writing $[y_1y_2\cdots y_n]$ for the integer value represented by the binary string $y_1y_2\cdots y_n$ and recalling the definition $[y_1y_2\cdots y_n] = 2^{n-1}y_1 + 2^{n-2}y_2 + \cdots + y_n$ we see that

$$\begin{aligned}
[y_1y_2\cdots y_i\cdots y_n] &= 2^{n-1}y_1 + 2^{n-2}y_2 + \cdots + 2^{n-i}y_i + \cdots + y_n \\
&= 2^{n-i}(2^{i-1}y_1 + 2^{i-2}y_2 + \cdots + y_i) + \cdots + y_n \\
&= 2^{n-i}[y_1\cdots y_i] + [y_{i-1}\cdots y_n]
\end{aligned}$$

So, $2^{n-i}([y_1 \cdots y_i] + 1 \mod 2^i) + [y_{i-1} \cdots y_n] = [y_1 y_2 \cdots y_i \cdots y_n] + 2^{n-i} \mod 2^n$.

Now we can see that the circuit above maps the integer $[y_1 y_2 \cdots y_n]$ to

$$\sum_{i=1}^{n} 2^{n-1} x_1 + 2^{n-2} x_2 + \cdots + x_n + [y_1 y_2 \cdots y_n] = [x_1 x_2 \cdots x_n] + [y_1 y_2 \cdots y_n] \mod 2^n$$

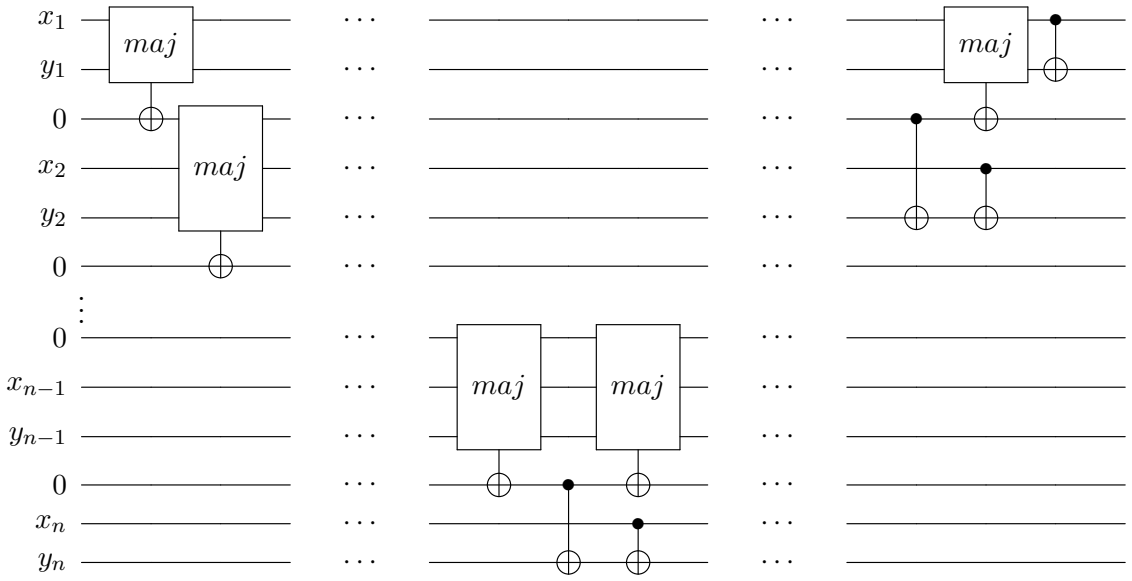
2. (4 points) Binary addition can be performed via long addition. Recall that in long addition, we sum three values in each column: the digits in question for the numbers being added (i.e. x_i and y_i) as well as the carry-in from the last digit, c_i. The resulting value for a binary adder is then $y'_i := x_i \oplus y_i \oplus c_i$, and we carry a 1 if at least 2 bits of x_i, y_i, and c_i are 1, so $c_{i+1} := maj(x_i, y_i, c_i)$ which we saw in assignment 1 can be implemented with just one Toffoli gate. We could construct an addition circuit in this way by using $n-1$ ancillas to store the carry bits as so:

   (a) $y_n = y_n \oplus x_n$
   (b) $c_{n-1} = \oplus c_{n-1} maj(x_n, y_n)$
   (c) For i from $n-1$ to 1:
      i. $y_i = y_i \oplus x_i \oplus c_i$
      ii. $c_{i-1} = c_{i-1} \oplus maj(x_i, y_i, c_i)$

   At this point we may notice that we can't simply clean up the carry bits, since we've changed the y register. For this reason, it makes more sense to first compute *all* the carry bits at once, then interleave computation of output bits followed by uncomputation of carry bits, like so:

   (a) $c_{n-1} = c_{n-1} \oplus maj(x_n, y_n)$
   (b) For i from $n-1$ to 1:
      i. $c_{i-1} = c_{i-1} \oplus maj(x_i, y_i, c_i)$
   (c) For i from 1 to $n-1$:
      i. $y_i = y_i \oplus x_i \oplus c_i$
      ii. $c_i = c_i \oplus maj(x_{i+1}, y_{i+1}, c_{i+1})$
   (d) $y_n = y_n \oplus x_n$

   The resulting circuit is shown below:



3. (2 points) With this particular addition circuit, it can be observed that if the sum computations (i.e. the individual $CNOT$ gates) are each controlled on a third bit, the sum is only computed if the control bit is in the $|1\rangle$ state. This turns the $2n-1$ $CNOT$ gates into $2n-1$ Toffoli gates. Moreover, since the carry computations and uncomputations don't depend on the final values we don't need to make them controlled — whether or not the control bit is set, the carries will always be computed and then uncomputed.

Question 4 [4 points]: Resource estimate redux

Using catalytic embeddings, we can write the QFT circuit as an alternating sequence of H gates and controlled *subtractions* into an ancillary register prepared in the state $|A_1 A_2 \cdots A_n\rangle = \otimes_{i=1}^{n} R_i H|0\rangle$. While it is not particularly important, it can be observed that we actually subtract at each stage a i-bit number from an $(i+1)$-bit substring of the ancillary register, which we can implement using an $(i+1)$ bit subtractor.

Hence we have the following accounting of gates:

- For the ancillary register, 1 $R_3 = T$ gate and $n-3$ R_k gates with $k \geq 4$.
- For the main circuit, a controlled i-bit subtractor from 2 to n.

For the 32 bit case in question, we hence have $n-3 = 29$ gates to approximate, giving a T count of

$$29 * 3 \log_2(29/\epsilon) = 29 * 3 \log_2(29 * 10^7) \approx 2436$$

T gates for gate approximations. The controlled i bit subtraction uses $2(i-3) + 2i - 1 = 4i - 7$ Toffoli gates, or $7(4i-7) = 28i - 49$ T gates. Suming up the T count from all the controlled substractions gives

$$\sum_{i=2}^{32} 28i - 49 = 14707$$

Adding these together gives us our final T-count estimate of 17143, which is an order of magnitude less than the "vanilla" QFT circuit.

While we didn't cover this technique, this can be further reduced by replacing the *maj* computations and uncomputations with *relative phase* implementations, as well as the controlled sums. This reduces the T-count per Toffoli gate to 4, leading to an approximate final T-count of 10840.

The final part of this question asked how this technique may be beneficial if we need to do two (or more) QFTs in an algorithm. Here it can be observed that we can use the same ancillary states to implement *both* QFTs, so the approximation cost is *fixed* no matter how many QFTs we perform. By contrast, the vanilla implementation would require an increasing number of gate approximations with each additional QFT, requiring even higher precision approximations.