Figure 1: CCNTC Circuit for unfanout for odd number of qubits

Figure 2: CCNTC Circuit for unfanout for even number of qubits

1 The Overall Plan

Here are the main steps remaining for the constant-depth factoring implementation:

1. Finish testing out and writing up constant-depth fanout, which I've given steps for below.
2. Calculate the resource overhead for circuit depth, size, width, including module equivalents, for overall multiple product and modular reduction, in 2D CCNTCM model.
3. Calculate the resources for a single majority gate in 2D CCNTCM, using constant-depth unfanout resources.
4. Calculate the same resources for implementing a majority gate directly with rotation by Hamming value and a threshold of $n/2$. See whether this is less than Step 3.
5. Combine steps 2 and 3 (via multiplication) to get total resources for iterated log-depth factoring.

Here are the substeps to do for constant-depth unfanout.

1. Figure out a compact, or at least systematic, way of correcting the resulting source qubit. Otherwise, we can hardly be said to have a constant-depth fanout. Do this for the odd case.
2. Figure this out for the even case (do the above for even n).
3. Unify the two of them into the same general formulae.

2 Constant-Depth Unfanout

We now present a circuit for constant-depth unfanout in CCNTC assuming that all the fanned out qubits are adjacent in some contiguous line. The actual unfanout only requires 1D, however, normally the fanned out qubits have been teleported else and must be teleported back into a contiguous line. This operation normally requires 2D in order to be constant-depth. However, we neglect that here, only assuming that the "user" or "client" module of constant-depth unfanout is able to get their qubits in the right form.

The general technique for both fanning out and unfanning out is to increase the subspace inhabited by the state. For fanning out, we first consider the

submodule of cat state creation. An unbounded product state of k constant-sized cat states inhabits a subspace of 2^k computational basis states. CNOTs provided entanglement between the constant-sized cat states, thereby making irreducible back to a product of k cat states, but without reducing the dimension of the subspace from 2^k. Then the measurements (they do not need to be Bell basis measurements) project the entangled state onto a smaller subspace by half. Therefore, to get back from a 2^k-dimensional subspace to a 2-dimensional subspace, we need $k-1$ measurements, which indeed is what happens.

For the unfanout, we cannot simply perform the same fanout procedure twice. (Why not?) However, we can take the same approach. We start with a 2-dimensional subspace, the fanned out state which is analogous to the n-qubit cat state with coefficients α and β instead of $1/\sqrt{2}$. To do this, we must enlarge the subspace of our entangled state back to 2^n, which we can do by applying Hadamards to every single qubit. At this point, the state is still entangled, and we need to disentangle them the same way that we entangled them in the first place. To do so, it turns out we enact two layers of interleaved CNOTs and must project the entangled state back down to a 2-dimensional subspace, namely by projective measurements on all qubits except the "target" qubit.

Because the fanned out state is symmetric, any of these qubits can be left in the original "source" state of $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. At the end of which the state is no longer entangled, i.e., it can be expressed as the tensor product of n single-qubit states. Therefore, we must perform some *asymmetric* operation to the fanned-out state to choose which qubit is the target, which does not necessarily have to be the same as the source qubit. However, it will be in a similar location to the original source qubit, if further uncomputing is necessary, of which this unfanout is merely a subroutine. This asymmetric action is to projectively measure in the Z-basis on all other qubits *except* the target qubit. On the target qubit, we apply the Hadamard, since it will be left in the source qubit state but in an alternate basis.

Finally, the resulting target qubit state will need some Pauli correction (it turns out, only a Pauli Z correction is necessary) based on the classical measurement outcomes of the other qubits. This is a classical boolean function with 1-bit outputs and $n-1$-bit inputs, (define what n is above), so this can be done on a classical computer in polynomial time simply by using a lookup table. (If this requires 2^n exponential space, however, can we still claim this? I think it is necessary to find a compact representation, via Karnaugh maps or the like).

Using this new constant-depth unfanout, it is now possible to reduce the factoring depth of the previous implementation to $O(\log^2 n)$ rather than $O(\log^3 n)$.

The question, remains, is this optimal? That is also a question for another time.

3 The Threshold Model

We must emphasize, as Aram says, that our implementation of factoring down to the CCNTC model is $O(\log \log n)$ depth overhead due to quantum compil-

ing, which any other implementation will have to face, even if they use the results from classical threshold circuit complexity. However, the threshold gate itself can be performed in CCNTC with only constant-depth overhead, and a polynomial (say $O(n^2)$) increase in size and width (for ancillae)

4 New Idea: Recursive Approximation of KSV

Instead of approximating the rotation $R_Z(1/2^n)$ in one go, what if we approximated two rotations whose difference would equal the desired rotation, but would be asymptotically faster to approximate? Or at least, in expectation, be faster. A bold new idea, for another time.