Pairwise Decomposition of Toffoli Gates in a Quantum Circuit

Nathan O. Scott
Faculty of Computer Science
University of New Brunswick
New Brunswick, Canada
Nathan.O.Scott@unb.ca

Gerhard W. Dueck
Faculty of Computer Science
University of New Brunswick
New Brunswick, Canada
gdueck@unb.ca

ABSTRACT

Quantum circuit synthesis is the procedure of automatically generating quantum circuits to represent specified functions. A common gate in quantum circuits is the reversible Toffoli gate, a type of generalized controlled NOT operation. There are physical barriers to implementing large quantum gates. Large Toffoli gates can be decomposed into equivalent sets of smaller, quantum elementary gates. The cost of a quantum circuit can be measured by counting the number of elementary gates in the circuit after all gates have been decomposed. Traditionally this decomposition is done independently for each gate in the circuit. This thesis identifies pairs of gates that, if decomposed together, result in fewer total elementary gates than they would otherwise. These improvements are incorporated into a simple decomposition algorithm which manipulates the circuit in order to search for such pairs. The decomposition algorithm is compared to a naive implementation, and the resulting gate costs are presented and compared.

Categories and Subject Descriptors

B6.3 [**Design Aids**]: Automatic synthesis

General Terms

Design, Theory

Keywords

Reversible Logic, Quantum Circuits, Elementary Quantum Gates, Synthesis, Minimization

1. INTRODUCTION

Quantum computing is an area of research that has attracted significant attention in recent years. There are several physical obstacles to the physical implementation of quantum computers of any practical scale[3]. This motivates researchers in quantum circuit synthesis to develop techniques that produce circuits that are efficient according to various metrics. One metric is counting the total number of elementary gates required to realize the circuit after applying the decomposition procedures detailed by Barenco et al.[2] to the circuit.

We have focused on decomposing circuits made up of Toffoli gates[10]. Barenco et al.[2] provide a simple decomposition procedure that, for a Toffoli gate of a given size, produces a linear number of elementary gates to realize it. As can be seen by looking at the implementation for the Peres gate [7, 8], certain pairs of Toffoli gates can be implemented using fewer elementary gates than would result from applying the Barenco decompositions individually to each gate. We have identified several other cases where this sort of improvement can occur. Using a simple decomposition algorithm that attempts to pair Toffoli gates together in a circuit to take advantage of these improvements, we have found that a significant reduction in overall elementary gate count can occur.

GLSVLSI'08, May 4–6, 2008, Orlando, Florida, USA.

2. BACKGROUND

A *qubit* is the quantum analogue of a classical bit in computation. Rather than being limited to two states, 0 and 1, a qubit can be in a state of superposition of both 0 and 1.

Let $|0\rangle$ and $|1\rangle$ be orthogonal, two dimensional complex vectors. A qubit $|q\rangle$ is a vector such that $|q\rangle = a\,|0\rangle + b\,|1\rangle$, where a and b are complex numbers such that $|a|^2 + |b|^2 = 1$. $|q\rangle$ may be written in a traditional vector notation as $|q\rangle = \begin{bmatrix} a & b \end{bmatrix}$.[6]

For the purposes of our work, we restricted the possible states for a qubit to be in to 4 possibilities:

$$\begin{array}{lll} |0\rangle & = 1\,|0\rangle + 0\,|1\rangle & = \begin{bmatrix} 1 & 0 \end{bmatrix} \\ |+\rangle & = \left(\frac{1-i}{2}\right)|0\rangle + \left(\frac{1+i}{2}\right)|1\rangle & = \begin{bmatrix} \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix} \\ |1\rangle & = 0\,|0\rangle + 1\,|1\rangle & = \begin{bmatrix} 0 & 1 \end{bmatrix} \\ |-\rangle & = \left(\frac{1+i}{2}\right)|0\rangle + \left(\frac{1-i}{2}\right)|1\rangle & = \begin{bmatrix} \frac{1+i}{2} & \frac{1-i}{2} \end{bmatrix} \end{array}$$

Rather than consider all possible quantum operations, we also have only considered a subset of them: NOT, V and $V^\dagger$. They affect the 4 considered states as given in Table 1.

2.1 Elementary Gates

The following set of gates are the *elementary gates* for our purposes:

NOT **Gate** Performs the NOT operation unconditionally.

V **Gate** Performs the V operation unconditionally.

$\|q\rangle$	$NOT(\|q\rangle)$
$\|0\rangle$	$\|1\rangle$
$\|+\rangle$	$\|-\rangle$
$\|1\rangle$	$\|0\rangle$
$\|-\rangle$	$\|+\rangle$

$\|q\rangle$	$V(\|q\rangle)$
$\|0\rangle$	$\|+\rangle$
$\|+\rangle$	$\|1\rangle$
$\|1\rangle$	$\|-\rangle$
$\|-\rangle$	$\|0\rangle$

$\|q\rangle$	$V^\dagger(\|q\rangle)$
$\|0\rangle$	$\|-\rangle$
$\|+\rangle$	$\|0\rangle$
$\|1\rangle$	$\|+\rangle$
$\|-\rangle$	$\|1\rangle$

Table 1: Output tables for single qubit gates.

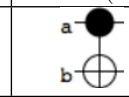
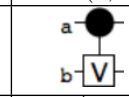
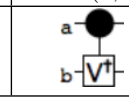
$NOT(a)$	$CNOT(a,b)$
	
$V(a)$	$CV(a,b)$
	
$V^\dagger(a)$	$CV^\dagger(a,b)$
	

Figure 1: Visualisations of the elementary gates.

$V^\dagger$ **Gate** Performs the $V^\dagger$ operation unconditionally.

$CNOT$ **Gate** Performs a controlled version of the NOT gate. (Only applied when the control is $|1\rangle$)

CV **Gate** Performs a controlled version of the V gate.

$CV^\dagger$ **Gate** Performs a controlled version of the $V^\dagger$ gate.

While they are not universal for quantum computation, they are sufficient to implement the size 3 Toffoli gate. The size 3 Toffoli gate is universal for reversible boolean computations.[10]

The *Toffoli gate* has an arbitrary number of control lines, and a single target line. If all of the control lines are $|1\rangle$, then the NOT operation is applied to the target.

The minimal implementation of the 3-bit Toffoli gate using the elementary gates requires 5 such gates, and is due to Barenco et. al.[2] There are multiple ways to do this, but they share a common structure. The basic version of this follows:

$$T(ab,c) = V(b,c)T(a,b)V^\dagger(b,c)T(a,b)V(a,c) \text{ [2]}$$

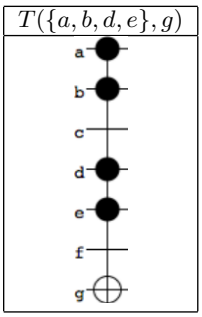


Figure 2: Visualisation of a Toffoli gate.

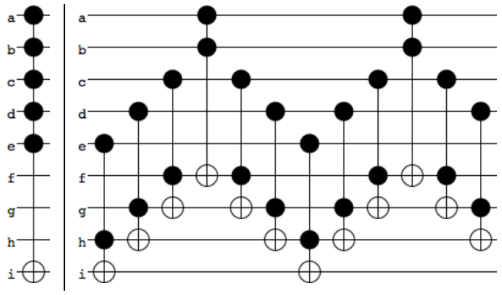


Figure 3: An example stairway decomposition.

This particular decomposition will be referred to as the *basic Toffoli gate decomposition.* Alternate decompositions can be obtained by reversing the order of the gates, inverting the circuit, or moving gates around within the circuit (two gates can be safely exchanged if their controls and targets do not intersect.) The gates that apply to the target are called the *target gates* and the gates that apply on other lines are called the *work gates.*

There are two main decompositions provided by Barenco et al. for larger Toffoli gates. There are restrictions on when the decompositions can be applied based on how many lines are available in the circuit, compared to how many lines are used by the gate being decomposed. Lines that exist in the circuit but are not used by the gate will be referred to as "work lines."

The first decomposition is taken from Lemma 7.2 in the paper by Barenco et al. [2]. For a circuit with $n \geq 5$ lines, and a Toffoli gate of size $m, 4 \leq m \leq \lceil \frac{n}{2} \rceil + 1$, it is possible to implement the gate using $4(m-3)$ size 3 Toffoli gates. Using the basic decomposition on each of these gates results in a total of $20(m-3)$ elementary gates. This will be referred to as the *stairway decomposition*, due to its distinctive appearance in its simplest form. See Figure 3 for an example.

The stairway decomposition can only be applied when the gate is at most half as wide as the circuit itself. If there is at least one work line available, but not enough for the above decomposition, the following decomposition is available, taken from Lemma 7.3 from [2]. For a circuit with $n \geq 5$ lines, and a Toffoli gate of size $n-1$, it is possible to implement the gate using 2 Toffoli gates of size $k+1$ and 2 of size $n-k$, for any $k, 3 \leq k \leq n-2$. See Figure 4 for an example with $n = 9$ and $k = 4$.

In addition to the restricted state and gate sets, some further simplifying assumptions were made. We only consider circuits that realize reversible boolean functions. Our decompositions assume that only boolean values ("pure" states of $|0\rangle$ or $|1\rangle$) are ever input to control lines, even during intermediate calculations, and our decomposition algorithm is engineered to ensure that this remains the case throughout the decomposition procedure.

3. SMALL TOFFOLI GATE PAIRS

Small Toffoli gate pairs are pairs where both gates are of size 3, or one is of size 3 and the other is of size 2. In the following examples, G_1 and $G2$ are two such gates, and we are attempting to decompose the pair G_1G_2. We assume without loss of generality that the size of G_1 is greater than or

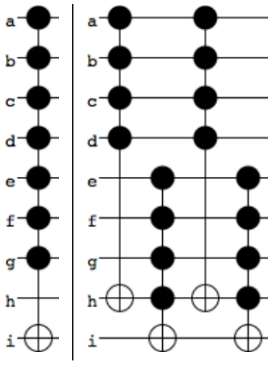


Figure 4: An example splitting decomposition.

equal to the size of G_2. (If not, reverse the order of the gates to satisfy this condition, and then invert the decomposition that results.)

All possible configurations of this type were considered, and the following improvements were identified:

- Let $G_1 = T(ab, c)$ and $G_2 = T(a, c)$. By applying the basic decomposition to G_1, the following decomposition is obtained:

$$G_1G_2 = V(b,c)T(a,b)V^\dagger(b,c)T(a,b)V(a,c)T(a,c)$$

  $V(a,c)T(a,c)$ is equivalent to a single $V^\dagger$ gate, $V^\dagger(a,c)$, therefore this circuit can be realized with 5 total gates:

$$G_1G_2 = V(b,c)T(a,b)V^\dagger(b,c)T(a,b)V^\dagger(a,c)$$

- Let $G_1 = T(ab, c)$ and $G_2 = T(a, b)$. An efficient implementation of G_1G_2 is known as the "Peres gate" and can be implemented using only four elementary gates:

$$G_1G_2 = V(b,c)V(a,c)T(a,b)V^\dagger(b,c)\ [7,\ 8]$$

- Let $G_1 = T(ab, c)$ and $G_2 = T(ad, c)$. In this case, the target is inverted if and only if, a is 1 and $b \oplus d$ is 1. $b \oplus d$ can be computed using a single Toffoli gate, and this computation can be undone afterward. See figure 5 for an example, before decomposing G_2 further. After G_2 has been decomposed, this circuit uses 7 total elementary gates, an improvment over the 10 required by treating each gate independently.

$$G_1G_2 \quad = T(b,d)T(ad,c)T(b,d)$$

- Let $G_1 = T(ab, c)$ and $G_2 = T(ab, d)$. Here the work gates of the basic decomposition of either gate can be made identical, and so the work done by these gates can be "re-used" for the two unique target gates, saving 2 elementary gates in total.

$$\begin{aligned} G_1G_2 \quad &= V(b,c)V(b,d)T(a,b)V^\dagger(b,c) \\ &\quad V^\dagger(b,d)T(a,b)V(a,c)V(a,d) \end{aligned}$$

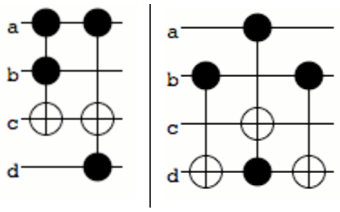


Figure 5: Example improvement, before final decomposition.

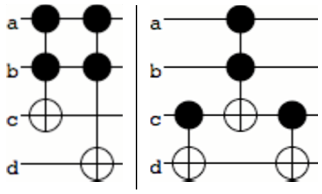


Figure 6: Example improvement, before final decomposition.

- Let $G_1 = T(ab, c)$, $G_2 = T(ab, d)$, and that $c = 0$ or $c = 1$.

$$G_1G_2 = T(ab,c)T(ab,d) = T(c,d)T(ab,c)T(c,d)$$

  Applying the basic decomposition to G_1 following this step results in 7 total elementary gates. This decomposition was discovered with the aid of tools from [4]. Figure 6 shows an example of this decomposition, before G_1 is decomposed further.

- Let $G_1 = T(ab, c)$, $G_2 = T(ad, b)$. When decomposing G_1, we can choose to use b as the target line of our work gates, which can be combined with one of the target gates of G_2's decomposition if you decompose G_2 using a reversed decomposition, resulting in a circuit with 9 elementary gates.

- Let $G_1 = T(ab, c)$, $G_2 = T(ac, b)$. We can choose our work gates of G_1 to coincide with the target gates of G_2, and moving some gates around, we can obtain a 2 gate improvement over the independent decomposition by merging V and $V^\dagger$ gates with Toffolis. This decomposition was discovered with the aid of the tools developed in [4].

The results of each of the above cases were verified as being minimal with the aid of the SAT-based synthesis tool developed by Große et al. at the University of Bremen[4].

4. LARGE TOFFOLI GATES

For larger pairs of Toffoli gates, it was not as straightforward to enumerate all possible situations. Instead, a few particular instances of decompositions that could potentially be improvements were identified.

Apart from the constraints on the number of available work lines, the splitting decomposition is very flexible. When splitting the controls into the two sets, the choice of which lines to put in each set is arbitrary. Thus we may take a

# of circuits	40310
# of improved circuits	36864 (91.45%)
Average improvement (gate count)	3.517 gates
Average improvement (%)	19.565%

Table 2: Results for 3-bit reversible boolean functions.

similar approach to decomposing large Toffoli gates as seen in Section 3. Generally, we look for two gates with intersecting controls, split them in such a way as to produce identical work or target gates, and thus eliminate a duplicate pair.

This sometimes produces decompositions that are actually worse than treating the gates individually. This occurs if the gates have intersecting controls, but the size of the intersecting set is small in comparison to the size of the individual gates. When working with small gates our testing procedure has each possible case that results in an improvement explicitly enumerated in advance. Rather than attempt something similar for larger gates, the program instead generates the decompositions on demand according to our heuristics, and the optimizing algorithm compares the resulting costs to decide whether to accept or reject the pairwise decomposition as an improvement.

5. THE DECOMPOSITION ALGORITHM

To determine what amount of improvment can be expected by searching for improved decompositions within a circuit, a simple optimizing decomposition algorithm was compared against a straightforward application of the Barenco decompositions.

The pairwsie decomposition algorithm first enumerates all pairs of gates that can be safely moved together, based on the techniques developed in [9]. For each pair, the algorithm checks to see if any of the pairwise decompositions that we have identified apply. It then greedily chooses the largest improvement, breaking ties arbitrarily. If no pair offers an improvement, a gate chosen arbitrarily from the largest gates in the circuit is decomposed using a Barenco decomposition. This continues until all gates in the circuit are elementary and no optimizations are found.

The pairwise algorithm was not optimized and introduces a significant cost in computing time. The intent was to identify how much of a reduction in gate cost could be expected in various circuits, and performance measurements were not gathered.

6. RESULTS

Three testing grounds were used to measure the improvements.

6.1 All 3-bit reversible boolean functions

Using the algorithms developed by Dueck et al.[5], initial circuits were synthesized for all size 3 reversible boolean functions. The identity function and functions realizable with a single elementary gate were ignored. Table 2 summarizes the key results from this test. The average improvement was 3.517 gates, and each circuit had an average 20% reduction in gate count. Almost all circuits (over 90%) had some amount of improvement.

Table 3 shows the distribution of improvements. The largest improvement, in terms of gate count, was 9 gates, and the most frequently occurring improvement was 4 gates.

Difference in gate count	Frequency
0	3446
1	1121
2	7429
3	6445
4	9408
5	7703
6	3336
7	1117
8	274
9	31

Table 3: Distribution of improvements for 3-bit reversible boolean functions.

Size	Difference	Percent improved
4	2	10.0%
5	5	12.5%
6	8	13.3%
7	11	13.8%
8	14	14.0%
9	17	14.2%
10	58	24.2%
11	74	26.4%
12	98	30.6%
13	94	26.1%
14	102	25.5%
15	116	26.3%

Table 4: The final costs for decomposable Toffoli gates in a 16 line circuit.

6.2 Single large Toffoli gates

Barenco's decompositions for Toffoli gates are impressive, and scale linearly with the size of the gate. However, the improved decompositions of Section 3 also led to significant improvements on the final elementary gate count required to implement a single Toffoli gate of any size.

Fixing a specific number of lines (n) in a circuit, the naive and improved algorithms were compared for decomposing single Toffoli gates up to size $n-1$. Table 4 summarises the results for $n = 16$.

6.3 Benchmarks

Finally, the decomposition process was tested on a series of benchmark functions. The objective is to determine the performance on circuits with numerous generalised Toffoli gates, as well as circuits that produce functions already deemed interesting by the research community. Sample circuits for each benchmark function were taken from the online repository at [1] which contains notable reversible circuits implementing various benchmark functions. Table 5 summarises the results.

7. CONCLUSIONS AND FUTURE WORK

We have identified several pairs of small Toffoli gates that can be implemented using fewer elementary gates than would normally be obtained by their decompositions. We have also

Benchmark	Barenco	Improved	%
2-4	45	36	20%
2of5	256	182	28.9%
4_49	74	62	16.2%
4mod5	24	16	33.3%
5mod5	250	92	63.2%
6sym	72	63	12.5%
9sym	108	94	13%
cycle10_2	1892	1268	33%
ham7	111	85	23.4%
hwb4	79	54	31.6%
hwb5	434	298	31.3%
hwb6	2114	1436	32.1%
mod5adder	181	143	21%
rd53	337	264	21.7%
rd73	76	65	14.5%
rd84	112	99	11.6%

Table 5: Summary of decompositions applied to benchmark circuits.

identified a general approach for decomposing larger pairs of gates to result in fewer elementary gates. We implemented a simple optimization algorithm that incorporated these improvements and found that in virtually all cases some degree of improvement can be obtained, and in some cases a dramatic reduction in gate counts results.

These results are promising given the restrictions and assumptions made at the outset, and given that the optimizing algorithm takes a very naive, greedy, hill-climbing approach to searching for these pairs. We expect that even better improvements will result from implementing a more robust optimization routine, and incorporating these techniques into a fuller model of quantum computation.

8. REFERENCES

[1] Revlib: The online resource for reversible functions and circuits. http://www.revlib.org/.

[2] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. Di Vincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52(5):3457–3467, November 1995.

[3] David P. DiVincenzo. The physical implementation of quantum computation. *Progress of Physics*, 48(9):771–783, September 2000.

[4] Daniel Große, Xiaobo Chen, Gerhard W. Dueck, and Rolf Drechsler. Exact sat-based toffoli network synthesis. In *Proceedings of the 17th great lakes symposium on VLSI*, pages 96–101, March 2007.

[5] D. Maslov, G. W. Dueck, and D. M. Miller. Toffoli network synthesis with templates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):807–817, June 2005.

[6] M. Nielson and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[7] Asher Peres. Reversible logic and quantum computers. *Physical Review A*, 32:1556, 1985.

[8] M. Perkowski, M. Lukac, M. Pivtoraiko, P. Kerntopf, and M. Folgheraiter. A hierarchical approach to computer aided design of quantum circuits. In *6th International Symposium on Representations and Methodology of Future Computing Technology (RM)*, pages 201–209, Trier, Germany, March 2003.

[9] N. Scott, G. W. Dueck, and D. Maslov. Improving template matching for minimizing reversible toffoli cascades. In *7th International Symposium on Representations and Methodology of Future Computing Technologies (RM)*, Tokyo, Japan, September 2005.

[10] Tommaso Toffoli. Reversible computing. In J. W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming*, pages 632–644. Springer-Verlag, 1980.