Communications Topology and Distribution of the Quantum Fourier Transform

Rodney Van Meter
rdv@tera.ics.keio.ac.jp

Graduate School of Science and Technology
Keio University
3-14-1 Hiyoushi, Kohoku-ku, Yokohama-shi, Kanagawa 223-8522, Japan
TEL: +81-90-8012-3643

Abstract— This paper quantifies the necessary movement of qubits in calculating the quantum Fourier transform (QFT) used in Shor's algorithm for factoring large composite numbers, and proposes a distributed form of the QFT. We can reduce the number of gates in the quantum Fourier transform by roughly 33% by moving from a linear array of qubits to a three-rail layout. Next, we consider using multiple quantum computers (or, a *quantum multicomputer*) to execute the QFT; the communications costs incurred are evaluated for several topologies. We find that inter-device swaps are 1-5% of the intra-device swaps for kilobit-capacity systems.

Keywords: quantum Fourier transform, quantum multicomputer

1 Introduction

Shor's algorithm shows how to factor very large numbers in polynomial time, with implications for public-key cryptography [Sho94]. Naturally, with public key syndromes in the range of kilobits, it is desirable to execute Shor's algorithm on multi-kilobit numbers. However, no proposed quantum computing technology has yet demonstrated the scalability necessary to reach such large numbers. Therefore, it makes sense to ask if we can combine multiple quantum computers into a single, larger system: a distributed-memory quantum multiprocessor, or *quantum multicomputer*. In this paper, we examine how a known algorithm can be run on such a system.

The heart of Shor's algorithm is the use of the *quantum Fourier transform* (QFT) in order-finding problems, including factoring large numbers. We concentrate on the full QFT, expecting it to be the most demanding communication, or qubit movement, task a quantum multicomputer might face. In the full QFT, each qubit must interact with each other qubit. In practice, an approximate QFT may be used when the size of the problem is large, reducing the number of long-distance interactions necessary [Cop94, BEST96].

First, we examine the QFT assuming a monolithic quantum computer consisting of a large number of qubits. Initially, we assume a linear array of qubits, then analyze possible topological changes to see what improvements might be made. While the motion of qubits does not change the complexity class of the algorithm, our results show that we can reduce the number of swap gates in the quantum Fourier transform by roughly 66%, and the total number of gates by 33%, by moving from a linear array to a two-dimensional mesh.

Oskin's proposed lattice layout [OCCK03], which is a realistically constrained version of Kane's model [Kan98], brings no benefits to the QFT compared to the linear ring model. We propose that a two-rail or three-rail structure, which appears to be feasible within Oskin's engineering rules, be used, saving 25% or 33% of the total gate count, respectively.

These results suggest that pursuing quantum architectures that support at least a partial 2-D qubit layout will bring significant benefits.

Next, we consider using a quantum multicomputer to execute the QFT; the interconnect topology required and the changes in the communications costs relative to monolithic systems are detailed. We find that inter-device swaps are 1-4% of the intra-device swaps for large systems.

2 Counting Ops in the Quantum Fourier Transform

In this section, we examine in detail the number and types of quantum gates necessary to execute the QFT. We classify the gates as "action" or "communication" gates. Only the former are part of the actual value calculation; the communication gates move qubits around so the calculations can be performed. The communication gates, therefore, are pure overhead, to be optimized or eliminated where possible.

To calculate the QFT for l qubits, apply H_j in reverse order from H_{l-1} to H_0, and in between do all $R_{j,k}$ for $k > j$, where H_j is the one-qubit Hadamard gate applied to the jth bit, and

$$R_{j,k} = e^{i\pi/2^{k-j}} \tag{1}$$

is the two-qubit gate on the $|11\rangle$ state of the designated pair of qubits j, k.

For example, in the quantum Fourier transform for 4 bits, the necessary operations are (left to right):

$$H_3 R_{2,3} H_2 R_{1,3} R_{1,2} H_1 R_{0,3} R_{0,2} R_{0,1} H_0 \tag{2}$$

The abstract form of the QFT manipulates pairs of qubits that are far apart, logically. In the Lloyd model operations can only take place on neighboring qubits, so swap gates are used to bring them together as necessary [Llo93].

The total cost of the QFT is therefore l H gates, $\sum_{j=1}^{l-1} j = l(l-1)/2$ R gates, and a number of swap gates dependent on the topology of the quantum computer and the starting and ending layout of data qubits.

The group of $R_{j,k}$ gates between each pair of H gates commute. We can take advantage of this by not maintaining the obvious, regular layout of qubits in the system during the QFT calculation. We will call this the "no return" policy.

Table 1 shows the gate counts necessary in the QFT for some powers of two, both with return and without. We will use the no return case as our baseline for comparing the efficiency of various topologies.

3 Topology and Communication

In the previous section, swap gates were counted assuming a linear structure. In this section, we investigate varying topologies.

It can be seen from table 1 that, in the linear case, the communication gates (all swap gates) outnumber the action gates roughly two to one for the return case, while in the no return case, the communication and computation costs are roughly equal. Perfect optimization, eliminating all of the communication, would give us a factor of two (no return) to three (return) improvement in total gate count.

3.1 Two-D Mesh

Let us next look at an extended two-dimensional mesh, in which Manhattan neighbors can interact, either swapping or performing two-qubit action gates, as shown in the left side of figure 2.

For an N-dimensional mesh, nodes away from the faces and edges have $2N$ links, of which one must be the inbound link. The theoretical maximum, then, is $2N - 1$ new neighbors per swap, or 3 for a 2-D mesh.

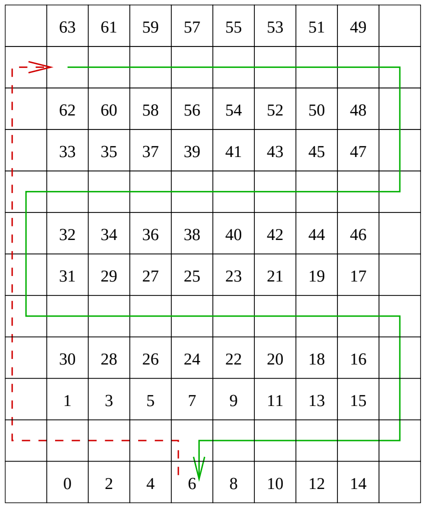

Figure 1: Serpentine qubit layout and motion for 64 qubits (return, with communications channels)

Figure 2 shows several steps in the evolution of the layout of a 32-bit system. Table 2 shows the gate counts for an $l = 3 \times m$ layout of qubits.

For physical systems with more square layouts (e.g., $m \times p, m \geq p > 3$), this three-rail layout may be folded, using the edges as communications channels.

For the return case, a more square layout brings significant additional reduction in total gate costs. Figure 1 shows qubits laid out with spare sites used as communication channels, expanding storage requirements in exchange for reduced gate count.

3.2 Oskin Lattice

Kane proposed a solid-state nuclear spin device, which uses standard VLSI techniques to control individual ^{31}P atoms embedded in the substrate, and suggested that one- and two-dimensional layouts of qubits may be possible [Kan98]. Oskin et al have proposed a structure which is linear, with occasional four-way intersections, based on engineering refinements of the placement and wiring constraints in the Kane model [OCCK03]. The authors show how to pack the classical control structures around a quantum wire (a swap channel of qubits), and how to create four-way intersections. Although their focus is on long-distance wiring and buses, their design appears to suggest that packing the control structures will preclude a full two-dimensional mesh. Figure 3 shows an extension of this approach to a multi-cell layout in a larger lattice, which we will call the Oskin lattice.

This architecture clearly provides only a single new neighbor per swap, except at the infrequent junctions. Thus, the cost is the same the simple linear array for the no return case, and we have been unable to find a bit layout that improves the return case, as well.

l	H	R	return swap	return total	no return swap	no return total
16	16	120	210	346	105	241
64	64	2016	3906	5986	1953	4033
256	256	32640	64770	97666	32385	65281
1024	1024	523776	1045506	1570306	522753	1047553
4096	4096	8386560	16764930	25155586	8382465	16773121

Table 1: Gate counts for the quantum Fourier transform for length l (linear Lloyd model)

l	H gates	R gates	swap gates	total gates	% red. in swap	% red. in total	swap %age of rem. tot.
16	16	120	44	180	63.3	25.3	24.4
64	64	2016	692	2772	64.6	32.3	25.0
256	256	32640	10964	43860	66.2	32.8	25.0
1024	1024	523776	174932	699732	66.5	33.2	25.0
4096	4096	8386560	2796884	11187540	66.6	33.3	25.0

Table 2: Gate counts for the quantum Fourier transform ($3 \times m$ mesh, no return)

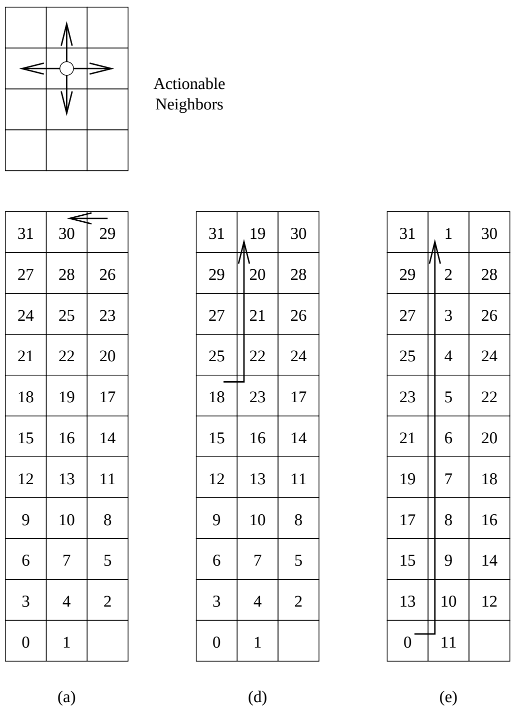

Figure 2: Changes in Non-Return 32-bit Layout as QFT Executes

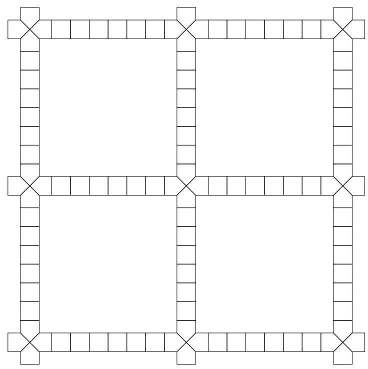

Figure 3: Lattice configuration (Oskin model).

Kane and Oskin have suggested placement of the phosphorus atoms on a pitch of 15-100nm, dependent on engineering tradeoffs; Oskin's work assumes an intermediate value of 60nm. With this wide variance in feature sizes and constraints, it seems likely that the Oskin classical control constraints, specified for a line (wire), can be met in a layout of two or three parallel lines. This will require three to five times the control line density of a single swap channel.

Using a simple data mapping to this layout results in a 50% reduction in swaps for the two-rail case, and, as we have seen, a 66% reduction for the three-rail case.

As a simple extension to the two-rail layout, if the qubits can be arranged to form a loop, a further reduction in swap count is easily achieved for the return case. This gives us a final reduction of 5/8 in the swap gate count, a 42% reduction in total gate cost for the QFT. This may be the sweet spot in engineering tradeoffs of difficulty of implementation versus functionality.

4 Distributing the Function

We now shift our attention to mapping the QFT to a quantum multicomputer. We usually refer to each of the quantum computing units as a *segment*, reserving the term *node* for the graph theory discussion of individual qubits. We assume each segment is a Lloyd-model device, as above.

4.1 Segment Input/Output

To build a quantum multicomputer, we need to transfer qubits among the QC segments. In this section we are not referring to classical data input, such as the number to be factored; that is usually "input" into the computer via single-qubit gates on the initial $|0\rangle$ state. Likewise, the final output, or measurement, can be achieved in parallel for some technologies, without moving qubits to specific I/O locations in the computer.

Depending on the physical implementation of a quantum computer (or computing segment), it is possible to imagine a variety of qubit input and output mechanisms and layouts. It may be possible to transfer qubits out only one or both ends, or out of any qubit. Similarly, if the segments are two-dimensional grids, qubits may be input or output from one or more edges, or from the face. Lloyd's proposal is that input and output take place only at the corners.

We are now faced with a two-level interconnect topology problem. The internal topology of each segment, plus its input/output capabilities, impact the interconnection of the segments. For this paper, we will concentrate on the one- and two-dimensional Lloyd models, with I/O at the ends and corners, respectively.

4.2 Connecting 1D Segments

Under the constraints above, linear segments can only be connected in a line or a ring. For s segments holding $n = l/s$ data qubits each, arranged linearly, we map the qubits directly onto the segments as if the whole system were a single linear array. Some of the swaps become inter-segment swaps, which are more expensive and more error-prone. With no buffer space around the data qubits, the number of swaps would be exactly the same as in the single linear array. However, we put a spare buffer qubit onto one end of each segment; otherwise it would be difficult to bring neighboring ends of segments together to perform the action gates. This results in the number of intra-segment swaps being the same as our original total, with the inter-segment swaps being extra.

The number of inter-segment swaps is

$$n\sum_{i=1}^{s-1} i = n(s)(s-1)/2 = l(s-1)/2 \qquad (3)$$

l	segs	qubits/seg	intra	inter
8	2	4	42	4
8	4	2	42	12
16	4	4	105	24
1024	2	512	522753	512
1024	8	128	522753	3584
4096	4	1024	8382465	6144
4096	16	256	8382465	30720
4096	64	64	8382465	129024

Table 3: Swap gate counts for the linear distributed-memory quantum Fourier transform (no return)

l	segs	qubits/seg	intra	inter
8	2	4	11	4
8	4	2	11	12
16	4	4	44	24
1024	2	512	174932	512
1024	8	128	174932	3584
4096	4	1024	2796884	6144
4096	16	256	2796884	30720
4096	64	64	2796884	129024

Table 4: Swap gate counts for the $3 \times m$ distributed-memory quantum Fourier transform (no return)

The number of inter-segment swaps grows linearly with s, for a fixed l. The exact numbers for some configurations are shown in table 3.

For the larger configurations, the inter-segment swaps are less than 1.6% of the intra-segment swap count. This suggests that there is room for inter-segment swaps to be significantly slower and more error-prone than intra-segment swaps without dramatically affecting either the run time or the reliability of the computation.

As in the monolithic device, connecting in a ring can eliminate roughly 1/4 of the swaps, both intra- and inter-segment, for the return case.

4.3 Connecting 2D Rails and Meshes

Two- and three-rail devices, with I/O at the ends, can be connected in several configurations, the simplest being a linear one, as in figure 4. Table 4 shows the inter-segment swap gate count for this configuration. In the last line, the inter-segment swaps are 4.6% of the intra-segment swaps; the inter-segment number is the same as in table 3, but the intra-segment numbers have declined.

Figure 5 shows one of the many possibilities for interconnecting sections of larger 2D mesh. For this simple chain of meshes, each of s meshes holds $n = l/s$ logical qubits. Assuming $n = hw$, with h and w both integers, the basic analysis is the same as for a two-dimensional mesh h by sw. The figure shows a detail of a 2D mesh chain, with a vertical serpentine bit layout, for a return policy. The storage requirements are $(3/2)w * (h + 2)$ for each mesh. To reach the close-to-square minimum overall configuration, we would like to have $(h+2) \approx (3/2)sw$.

5 Related Work

The Lloyd model [Llo93] offers 1, 2, and 3-D meshes, though the paper talks primarily about a linear array with bits going in one end. DiVincenzo et al propose a two-dimensional layout of solid-state qubits [DBK+00].

Early suggestions of distributed quantum computation include Grover [Gro97], Cirac [CEHM98], and Steane [SL00].

Cleve and Watrous have shown that the QFT can be calculated using integer multiplication, reducing the asymptotic running time to $O(n \log^2 n \log\log n)$ [CW00]. However, the high constant factors mean that the actual circuit size is larger for the problem sizes we present in this paper.

Yimsiriwattana and Lomonaco have presented a distributed version of Shor's algorithm [YL04]. Their approach uses entanglement to execute non-local operations, whereas our approach moves the qubits together before performing the operation.

6 Future Work

We have analyzed other two- and three-dimensional topologies, including the Copsey tree structure [COM+03], for which there isn't room in this paper. More detailed analysis of engineering constraints specific to certain quantum computing technologies is under way.

We are currently extending this analysis to the rest of Shor's algorithm, especially modular exponentiation, which is the most computationally demanding portion of the algorithm. We are also working to make the cost functions for gates implemented in various technologies more realistic, including both primitive gate times and error correction.

We are currently studying the feasibility of two-dimensional layout and distributed computing in an all-silicon NMR device [LGY+02].

Perhaps the most important problem to be solved is reliable inter-segment swap. The no-cloning theorem states that a perfect, independent copy of an arbitrary, unknown quantum state cannot be made. Thus, the usual networking approach of copy, buffer, transmit, retransmit on error cannot be used.

7 Conclusions

This work is part of an effort to determine the necessity and viability of a quantum multicomputer, with the ultimate goal of solving the practical system-level engineering issues in scaling up a quantum computer. We have investigated mapping the quantum Fourier transform (QFT) to both a monolithic quantum computer and a quantum multicomputer.

We have shown that the internal connection topology of a quantum computer has a significant impact on the number of quantum gates that must be executed to run the quantum Fourier transform.

In the linear Lloyd model, roughly half of the gates are swap gates used solely for communication. A $3 \times m$ two-dimensional grid can eliminate almost two-thirds of the swap gates, for an aggregate savings of one-third of the total gates. We believe this layout is potentially achievable under the Oskin engineering constraints.

We have also done an analysis of a distributed-memory quantum computer, concluding that the more expensive and error-prone inter-segment qubit swaps are from less than 1% up to about 5% of the total for large systems.

Acknowledgments

This work was supported in part by Ken Adelman and Dave Kashtan under the Network Alchemy basic research funding plan.

I have benefited greatly from discussions with Drs. Kawano and Takahashi of NTT Communications Research Laboratories, Prof. Lloyd of MIT, Dr. Touch of USC/ISI, and especially Prof. Itoh and Eisuke Abe of Keio University, as well as several anonymous reviewers.

References

[BEST96] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, and Päivi Törma. Approximate quantum fourier transform and decoherence. *Physical Review A*, 54:139–146, 1996.

[CEHM98] J.I. Cirac, A. Ekert, S.F. Huelga, and C. Macchiavello. Distributed quantum computation over noisy channels. http://arXiv.org/quant-ph/9803017, March 1998.

[COM+03] Dean Copsey, Mark Oskin, Tzvetan Metodiev, Frederic T. Chong, Isaac Chuang, and John Kubiatowicz. The effect of communication costs in solid-state quantum computing architectures. In *Proceedings of the fifteenth annual ACM Symposium on Parallel Algorithms and Architectures*, pages 65–74, 2003.

[Cop94] D. Coppersmith. An approximate fourier transform using quantum factoring. Technical Report 19642, IBM Research, 1994. http://arXiv.org/quant-ph/0201067.

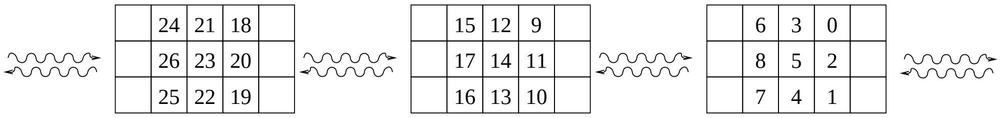

Figure 4: Detail for distributed 3-rail chain

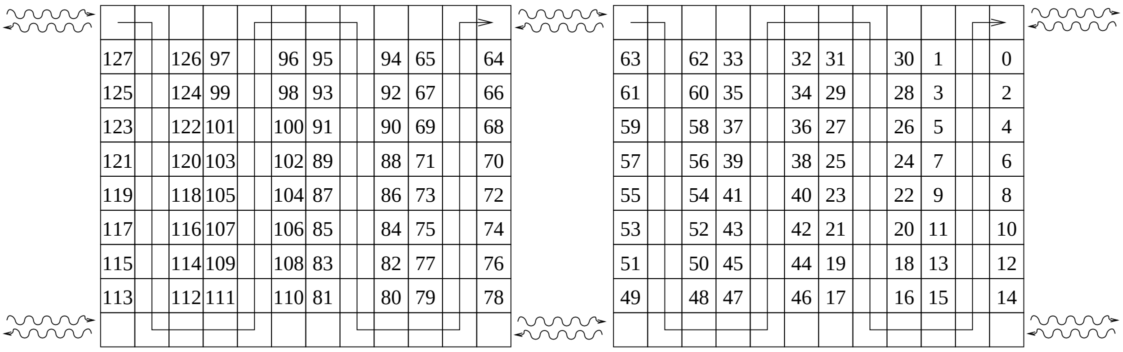

Figure 5: Detail for 2D mesh chain

[CW00] R. Cleve and J. Watrous. Fast parallel circuits for the quantum fourier transform. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, pages 526–536. ACM, 2000.

[DBK+00] D. P. DiVincenzo, D. Bacon, J. Kempe, G. Burkard, and K. B. Whaley. Universal quantum computation with the exchange interaction. *Nature*, 408:339–342, 2000.

[Gro97] Lov. K. Grover. Quantum telecomputation. http://arXiv.org/quant-ph/9704012, April 1997.

[Kan98] B. E. Kane. A silicon-based nuclear spin quantum computer. *Nature*, 393:133–137, May 1998.

[LGY+02] T. D. Ladd, J. R. Goldman, F. Yamaguchi, Y. Yamamoto, E. Abe, and K. M. Itoh. All-silicon quantum computer. *Physical Review Letters*, 89(1), July 2002.

[Llo93] Seth Lloyd. A potentially realizable quantum computer. *Science*, 261:1569–1571, 1993.

[OCCK03] Mark Oskin, Frederic T. Chong, Isaac L. Chuang, and John Kubiatowicz. Building quantum wires: The long and short of it. In *Computer Architecture News, Proc. 30th Annual International Symposium on Computer Architecture*. ACM, June 2003.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society Press, 1994.

[SL00] A. M. Steane and D. M. Lucas. Quantum
computing with trapped ions, atoms, and light. *Fortschritte der Physik*, April 2000. http://arXiv.org/quant-ph/0004053.

[YL04] Anocha Yimsiriwattana and Samuel J. Lomonaco Jr. Distributed quantum computing: A distributed Shor algorithm. http://arxiv.org/quant-ph/0403146, March 2004.