

A Realizable Distributed Ion-Trap Quantum Computer

Darshan D. Thaker[†] Tzvetan S. Metodi[†] Frederic T. Chong[§]

[†]University Of California at Davis, {*ddthaker, tsmetodiev*}@ucdavis.edu

[§]University Of California at Santa Barbara, *chong@cs.ucsb.edu*

Abstract. Recent advances in trapped ion technology have rapidly accelerated efforts to construct a near-term, scalable quantum computer. Micro-machined electrodes in silicon are expected to trap hundreds of ions, each representing quantum bits, on a single chip. We find, however, that scalable systems must be composed of multiple chips and we explore inter-chip communication technologies. Specifically, we explore the parallelization of modular exponentiation, the substantially dominant portion of Shor’s algorithm, on multi-chip ion-trap systems with photon-mediated communication between chips.

Shor’s algorithm, which factors the product of two primes in polynomial time on quantum computers, has strong implications for public-key cryptography and has been the driving application behind much of the research in quantum computing. Parallelization of the algorithm is necessary to obtain tractable execution times on large problems. Our results indicate that a 1024-bit RSA key can be factored in 13 days given 4300 (each of area 10 by 10 centimeters) ion-trap chips in a multi-chip system.

1 Introduction

Although quantum computers may have an exponential advantage over classical computers, the challenges in manipulating and preserving quantum data require substantial performance and design optimization to allow large problems to remain tractable. One of the most celebrated large-scale quantum applications is Shor’s algorithm for finding non-trivial factors of a large composite number in polynomial time [1], which leads to the ability to break the RSA public key cryptosystem. A classical computer may take millions of years for factoring numbers as large as 1024 bits, and even on a quantum computer such a calculation may take hundreds of years to complete. Consequently, this study focuses on effective parallelization of quantum computations on practical quantum architecture systems such as the Quantum Logic Array microarchitecture [2], which can improve large-scale execution times to useful levels. In particular, we find that area and reliability constraints require scalable systems to be constructed from multiple chips. Algorithms must then be designed to account for substantially different intra- and inter-chip communication bandwidth, much as in the classical multiprocessor domain.

The primary difficulty in the realization of quantum computation is that quantum data is inherently very unstable as it constantly interacts with the environment: a process called *decoherence*. To enable computation with such unreliable components, a rich theory of quantum fault-tolerance has been proposed [3–5]. Our previous work builds on these theories and the physical implementations of quantum computers, to

design a scalable and reconfigurable Quantum Logic Array (QLA) architecture [2]. The QLA is a data-path based, reconfigurable architecture design that offers an efficient way of structuring circuits with the much advanced trapped-ion technology [6, 7]. The QLA design supports the execution of quantum algorithms such as Shor’s factoring algorithm both flexibly and scalably through the complete overlap of computation and communication. We find, however, that the area requirements for building a single QLA chip to perform Shor’s algorithm for numbers as large 1024 bits would require unrealistic technological milestones to be met. To relax these requirements we explore *multi-chip* solutions with the QLA architecture that make such large-scale problems feasible for discussion.

The focus of this paper is the evaluation of quantum modular exponentiation, which is by far the most computationally intensive part of Shor’s factoring algorithm. Previous studies in this area have devised extensive quantum adder circuits that implement modular exponentiation [8–12]. However, this has been done with little regard to feasibility, or implementation on an existing quantum architecture models such as the QLA. We leverage the existing work on quantum adders to study and evaluate current and novel adder circuits on a multi-chip version of the QLA architecture. In particular, we show a circuit which through serialization requires the least possible area on a single QLA chip, and demonstrate that even maximal serialization yields area too great to physically overcome, in addition to unreasonable execution time. We introduce a *distributed quantum parallel prefix adder* based on the classical parallel prefix adder [13], and evaluate its communication and computation requirements on the multi-chip QLA architecture.

The paper is organized as follows: Section 2 provides a high-level description of the Quantum Logic Array architecture model, and a brief description of the logical and inter-chip interconnect technology. Sections 3 and 4 introduce Shor’s algorithm and give a simple, spatially-optimized design. Our multi-chip solutions are introduced and compared in Section 5. Finally results and conclusions are discussed in Section 6 and Section 7 respectively.

2 Quantum Computation and the QLA Architecture

Quantum mechanical systems such as the different electronic spins of an atom, or the polarization states of a photon can be used to hold and manipulate binary information just as the old vacuum tubes or the modern semiconductors do. Unlike classical binary bits, however, the fundamental units of quantum information, *qubits*, always exist in a superposition of the binary 0 and 1 states with some probability of obtaining one or the other upon observation. Measuring a quantum state not only yields the measured result with some probability, but forces the state to be in the measured value. A collection of N qubits exists in a superposition of 2^N different states at any given time denoted by the binary bit-strings representing the numbers 0 through $2^N - 1$. Logic gates are implemented as 2^N dimensional, complex valued, unitary matrices which act on the system state vector.

Our prior work, the QLA architecture [2] is designed to support large-scale quantum computation through the efficient manipulation of arbitrarily large number of qubits.

The major components of the QLA are: fault-tolerant logical qubits denoted with the letter Q , trapped atomic ions as the basic implementation technology at the bottom of the architecture and a teleportation based logical interconnect controlled.

The most important restriction on quantum circuits is the inability to form a sequence of gates that copies the state of a qubit, also known as the *no-cloning theorem* [14]. For a two-qubit operation the qubits must be brought together, and we cannot simply propagate the state of one to the other - we must physically transport each qubit. At inter-logical qubit distances ballistically moving the each ion-qubit holding valuable quantum data is too risky and will destroy the state.

2.1 Single-Chip Logical Interconnect

At small distances, the ballistic direct channels are enough for the communication requirements within logical qubits. However, the execution of any algorithm will depend of the efficient interaction among logical qubits. The concept of *quantum teleportation* [15] is utilized within each processor to provide high-speed, low-latency, fault-tolerant network interconnection between the logical qubits.

Quantum teleportation utilizes two qubits that have been previously prepared in an Einstein-Podolsky-Rosen (EPR) state [16]. The drawback is that these EPR qubits still have to be physically moved. However EPR pairs are replaceable and with enough resources we can establish entanglement between the source and the destination just in time for the communication to be completed.

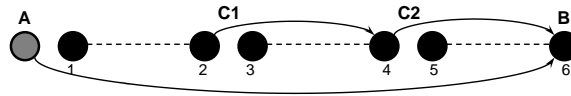


Fig. 1: A schematic of a single bandwidth channel between the logical qubits of our ion-trap processor. Rather than distributing an EPR pair of qubits directly from the source to the destination, the channel is divided into islands (here denoted as $C1$, and $C2$), that can be used to expand the entanglement over the entire channel.

In the intra-chip interconnect, our quantum processor solves the EPR transport problem by combining the concepts of *quantum repeaters* [17] and *entanglement purification* [18, 19, 17]. The quantum repeaters are islands that are strategically placed in the channels between the logical qubits to limit the distance traveled by each EPR pair. EPR pairs only travel to two neighboring islands, whose entanglement can then be efficiently purified using the purification protocols with some additional ancillary EPR pairs. Figure 1 the concept of quantum repeaters by showing a simple schematic of the channel between the logical qubits. The channel shown has a bandwidth of 1 physical bit at a time, and the ancillary EPR qubits are pipelined to the two opposing islands as they purify the data EPR pairs. EPR pairs (1, 2), (3, 4), and (5, 6) can be distributed and purified in parallel over the channel separating the source (A), and the destination (B).

The arrows in Figure 1 show how teleporting each individual pair, entanglement can be distributed between the source and the destination for one final teleportation stage.

2.2 Multi-Chip Interconnect

The above scheme however is not good enough when designing a multi-chip quantum network. The inter-chip distances, the difficulty in chip alignment, and the high failure rates at such conditions make it impossible to use the highly refined physical traps that make up the chips themselves. Instead of physically transporting ions encoded as an EPR pair between chips, two remote ions can establish entanglement through photon mediation [20, 21]. Photons emitted by the ions are collected with an optical fiber and sent to entangling stations to perform a collective measurement on the ions without knowing which ion sent which photon beam. The two ions are then projected into the two qubit entangled state. The application of an X gate on the second qubit easily converts this state to the familiar EPR pair. The communication process is then completed by teleporting the source ion to the other side as described above.

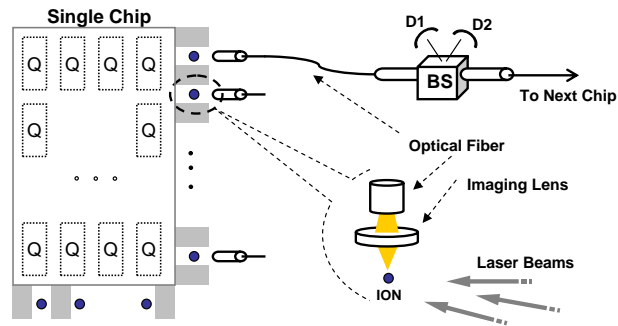


Fig. 2: Each chip has a connection pad on its sides with evenly distributed optical fiber locations. Remote trapped ions at each location on two opposing chip pads are entangled using photons emitted by the ions.

The ion-photon mapping method of creating entanglement between remote qubits is not possible inside the processors, because of the complex classical control required for quantum computation. However, outside of the processors it allows us to make an arbitrary quantum network by connecting adjacent processors as shown in Figure 2. Each chip has a connection pad on its sides with evenly distributed optical fiber locations. The bandwidth between two chips is defined by the number of optical fibers connecting them, which is limited by the chip dimensions. Due to the size of the optics, the fibers must be spaced at least $700\mu\text{m}$ apart [22], where a time of approximately $600\mu\text{s}$ for the entanglement operation in order to achieve near unit fidelity. The inherent entanglement purification of this channel [23] makes it much faster than the quantum repeater channel, thus we will not observe added latency when jumping from one chip to the next.

3 Shor’s Algorithm

Shors’ algorithm is widely studied due to its exponential advantage over conventional algorithms and due to its application to cryptography. Shor’s algorithm can be used to factor a product of two primes in polynomial time and comprises of the quantum fourier transform and quantum modular exponentiation. Although Shor’s insight was in evaluating the quantum fourier transformation in polynomial time, it is the modular exponentiation that consumes many times greater computational resources than the rest of the algorithm. In other words, if an ion trap processor that was 1 meter long on each side, could be built, it would take about 45 days to perform a heavily parallelized version of modular exponentiation. Whereas the same processor would be able to solve the quantum fourier transform in 0.5 days [2].

Performing Modular Exponentiation: A considerable amount of work has been done in developing implementations of adders for modular exponentiation [8, 10, 9, 12]. Nonetheless, the actual technology for quantum computing and the issue of fault-tolerance have not been considered in depth. Our previous work [2] develops a detailed architecture for a fault-tolerant, ion-trap based quantum processor. We employed a fast quantum carry-lookahead adder developed by [12] to efficiently perform modular exponentiation. The drawback of this design was the size of the quantum processor. It would need to be about $0.9m^2$ to accommodate all the required qubits. The next two sections address this very issue of size.

Table 1: Increase in area (in cm^2) with the number of logical qubits.

Number of Logical Qubits	Area of processor
100	2.65 cm^2
200	5.31 cm^2
300	7.963 cm^2
350	9.29 cm^2
400	10.617 cm^2

4 Single-Chip Solution

Our goal in this section is to design a circuit for modular exponentiation that is spatially optimized. The tradeoff is going to be extra time. From table 1, we determine that if we needed a processor that was less than $10cm^2$ it would restrict us to 350 logical qubits for the entire processor.

4.1 Serialized Adder

For performing addition in a small space, we choose the quantum ripple carry adder by Vedral et.al [8], which analogous to its classical counterpart, needs minimum extra qubits for an addition. Fig 3(a) shows a quantum circuit for determining sum and carry

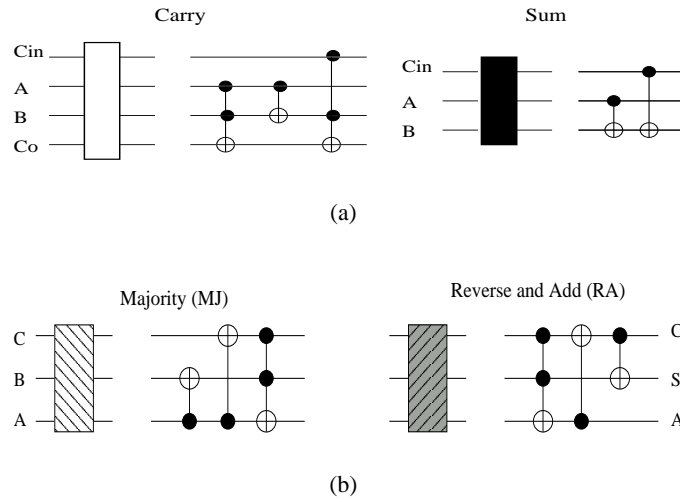


Fig. 3: (a) Standard method of computing carry and sum. (b) Using a majority (MJ) block and a reverse majority and add (RA) block to get the same result [11]

given inputs a and b and carry-in cin . The left-part of Fig 4 shows the original Vedral adder [8]. Note that the adder circuit has two carry blocks for the same inputs. The second block (on the right hand side of the circuit) performs the reverse computation, to erase the ancilla bits.

The Vedral adder can be improved if the reverse computations with the computations of the carry could be combined. This can be done using the *MJ* and *RA* blocks as shown in 3(b) as proposed by [11]. The *MJ* block computes the majority of three bits in place. The *RA* block adds two bits and reverses the majority. This allows us to save the n extra qubits that were needed for the Vedral ripple carry adder.

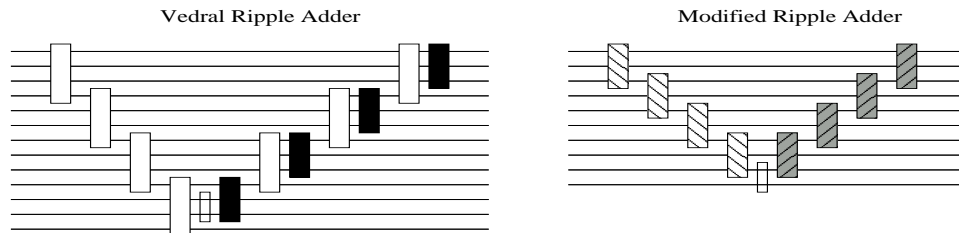


Fig. 4: 4-bit adder circuits using components from Figure 3(b) and Figure 3(a). The modified ripple carry adder requires $n - 1$ fewer qubits for adding two n qubit numbers

In Shor’s algorithm, a can be as large as n^2 . Thus we need $2n$ qubits to store a . Another n qubits are needed to store the result $(x^a \bmod n)$. The circuits of Vedral et.al use addition to perform modular addition; modular addition to perform controlled modular multiplication and finally modular exponentiation from repeated controlled modular multiplication. Modular addition, multiplication and exponentiation each require n temporary qubits. Our modified Vedral adder needs 1 temporary qubit. This gives a total of $6n + 1$ qubits. We can reduce this requirement by observing that during addition, n qubits hold classical values only and thus can be eliminated. Also during modular addition, a set of n temporary qubits holds either 0 or $2^i \bmod n$. These n qubits can be replaced by n classical bits and one qubit to keep track of entanglement. Our total required qubit count is now $4n + 2$. To this we add another 6 qubits which are required to perform a fault-tolerant toffoli gate; giving us a final count of $4n + 8$ qubits. The cost of keeping the required number of qubits low is that we have to perform all operations in serial order. They cannot be parallelized.

Table 2: Time and area for a serialized adder

N	Area in cm^2	ECC steps	Total Time
128	14.068	6.64 e9	3074 days
512	54.839	4.28 e11	542.87 years
1024	109.202	3.43 e12	4350.58 years

Table 2 shows the total area and time required for one complete modular exponentiation. In the table, ECC steps refers to the number error correction steps required. As can be seen, total area for the serialized version of the adder for the modular exponentiation of a 1024 bit number is quite large, $109.202cm^2$. At the other end, while the area for a 128 bit number is acceptable, the time it would take is approximately 8 years. The next section shows how we can reduce both the time and space requirements by using a parallel implementation of an adder.

5 Multi-Chip Solutions

To exploit parallelism, we explore the performance of larger quantum systems composed of multiple interconnected chips, which we shall refer to as *Qchips*. Specifically, we evaluate distributed quantum adder designs and the cost of inter-chip communication on a multi-chip system.

5.1 Carry-Select Adder

Let $n = g * k$, where n is size of the number to be factored in bits, which is divided into g groups of k bits each [10]. Thus there are g Qchips each of which is responsible for adding k bits. We then use a carry-select adder spread out over these Qchips. Let

Table 3: Number of Qchips of fixed area required for a given problem size

Carry Select Adder				Parallel Prefix Adder			
N	4 cm^2	8 cm^2	10 cm^2	N	4 cm^2	8 cm^2	10 cm^2
128	5	2	2	128	11	5	4
512	23	11	9	512	51	25	20
1024	47	23	19	1024	109	54	43

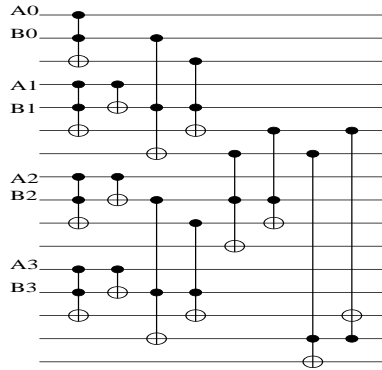


Fig. 5: Circuit of a parallel prefix adder for 4 bits. For reasons of clarity and space the reverse and sum computations are not shown

the Qchips be Qc_0, Qc_1, \dots, Qc_g . Each Qchip computes the sum and carry for its k bits in parallel. Since Qc_i does not have the carry out from Qc_{i-1} , it computes for both possibilities, the carry out being 0 and 1. When the carry from Qc_{i-1} is available, a multiplexer is used to select the correct value from the ones Qc_i calculated. In this manner, most of the computation can be done in parallel, except for the multiplexer. Two useful properties of such an adder are that the only communication between Qc_i and Qc_{i-1} is one bit, the carry from Qc_{i-1} , and that Qc_i communicates with Qc_{i-1} and Qc_{i+1} . Finally, we have to return the ancilla to their original state. This can be done by using an extra k qubits in each Qchip.

As outlined in [10], each Qchip will require about $6k$ qubits for the complete modular exponentiation. In order to perform parallel fault-tolerant toffoli's we conservatively increase it to $7k$ per Qchip. To calculate the time it would take for a complete, if inter-Qchip communication was free, we replace our parallel carry-select adder into the Vedral circuit. We also implement improvements suggested by van Meter and Itoh [10] namely using a lookup table to reduce the number of multiplications and their modulo adder block. Unlike their approach however, we only use one multiplier; this allows us to keep the area requirements within manageable levels.

For the carry-select adder, let $M1_i$ be a message from Qc_i to Qc_{i+1} . This is the carry out that Qc_i generated. When the ancilla have to be reset, Qc_i will receive the same

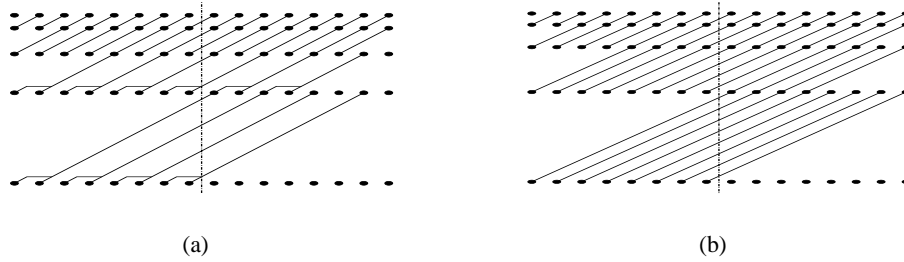


Fig. 6: Prefix graphs for Parallel-Prefix Adders [13]. Diagonal lines show communication between qubits at each stage. Each diagonal line carries two messages. Vertical line is Qchip boundary. The tradeoff lies between amount of communication and "fan-out". (a) The [2,2,1,1] scheme. (b) The [1,1,1,1] scheme

qubit as message $M2_i$ from $Q_{c_{i+1}}$. While all $M1$ messages cannot be parallelized, all $M2_j$ messages can be overlapped with $M1_{j+1}$ messages. Thus if we have g Qchips, the total communication cost we incur will only be g messages.

5.2 Parallel Prefix Adder

We now introduce our third adder which is a distributed parallel prefix adder. In a classical parallel prefix adder, partial information about the incoming carry is utilized to avoid ripple-carry computation. Let $C[i, j]$ denote the *carry status* on the interval $[i, j]$. It can have three possible values: k : kill, g :generate or p :propagate.

When adding a_i and b_i , we can determine something about the outgoing carry, c_{i+1} , without knowing c_i . If $a_i = b_i = 0$ then the $c_{i+1} = 0$ and $C[i, i+1] = k$. Similarly, if $a_i = b_i = 1$, then carry is generated and $c_{i+1} = 1$. Also $C[i, i+1] = g$. If $a_i \neq b_i$ then we cannot determine c_{i+1} and the carry is said to have "propagated". $C[i, i+1] = p$. The calculation of carry status is both associative and idempotent, which allows us to merge intervals. For any $i < k < j$,

$$p[i, j] = p[i, k] \cdot p[k, j] \quad (1)$$

$$g[i, j] = g[k, j] \oplus (g[i, k] \cdot p[k, j]) \quad (2)$$

The second expression is possible because $g[i, k]$ and $p[i, k]$ cannot both be 1 simultaneously.

Knowles [13] has shown that in the first stage of addition, each pair of qubit calculates its p and g values, the last stage calculates the sum while the intermediate stages all compute the *carry status*. There can be many approaches to communicating the p and g stages between qubits as shown by the prefix graphs in Figure 6. Each line represents two bits of data, p and g . Since we have to return the ancilla to their original state, these bits would also have to travel in the opposite direction at a later stage. Thus each lateral line in the graph, that crosses the Qchip boundary (represented by the dashed line),

Table 4: Time (in years) for a complete Modular Exponentiation when per message time is 800 E-6 sec

N	4 cm^2		8 cm^2	
	Carry Select	Par. Prefix	Carry Select	Par. Prefix
128	0.10	0.22	0.19	0.29
512	2.99	0.34	2.99	0.43
1024	19.9	1.35	15.64	1.69

represents four messages. For our distributed adder, a scheme like $[8, 4, 2, 1]$ would reduce the amount of inter-chip communication while increasing "fan-out". Since it is not possible to do "fan-out" in a quantum processor [24], we would be forced to do those computations in a serial fashion. On the other hand, a $[1, 1, 1, 1]$ scheme (Figure 6(b)) increases the amount of communication. For our adder we settle on a $[2, 2, 1, 1]$ scheme (Figure 6(a)). A quantum circuit of the parallel prefix adder for 4-qubits, based on $[2, 1]$ scheme, is shown in Figure 5; reverse and sum computations are not included. Communication requirements of our parallel prefix adder will increase with the number of stages in the addition (This is primarily due to our choice of the $[2, 2, 1, 1]$ scheme). Messages sent, between two Qchips, at stage i can be given by $M_i = M_{i-1} + 2^{i-1}$

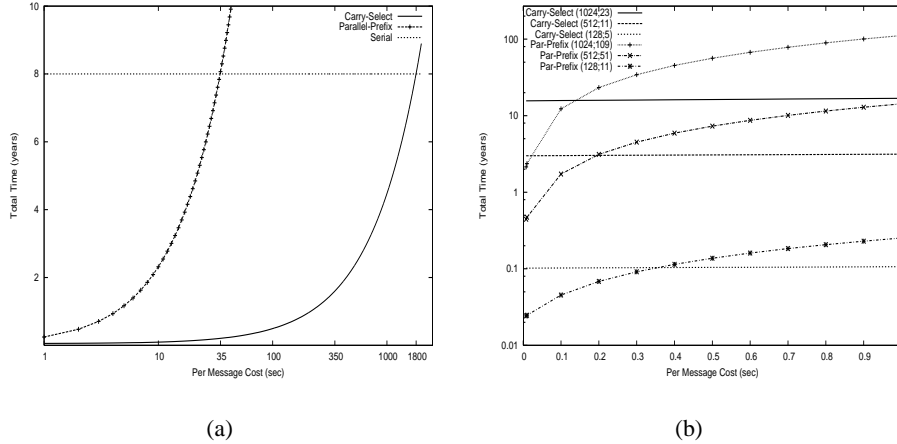


Fig. 7: (a) Shows when the multi-chips solutions outperform the serial version. (b) Performance of the Carry-Select and the Parallel-Prefix Adders

6 Results

Results for all our designs were obtained by hand calculations. Figure 7(a) shows the cost per message at which the multi-chip solutions outperform the single Qchip. Since

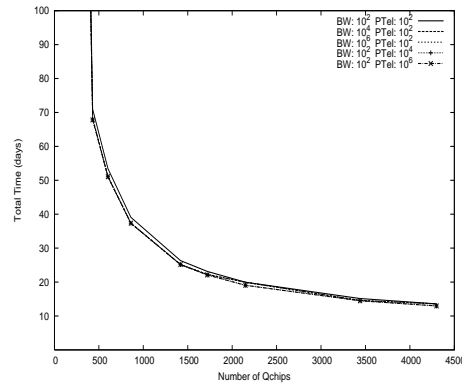


Fig. 8: Optimized Performance of the Parallel Prefix Adder with increasing number of Qchips

the number of messages between Qchips increases rapidly for the parallel prefix adder (while staying constant for the carry select), the cross-over point is at 35 sec/message; and almost 1800 sec/message for the carry select. In Figure 7(b), we compare the multi-chip solutions with message costs of $\leq 1\text{sec}$. We can see that the parallel prefix will clearly be the best solution as message costs decrease. To arrive at Table 4, we employ ideas from Section 2.2 and let cost/message equal $800\mu\text{sec}$.

To optimize the parallel prefix solution further, we consider the effects of having larger bandwidth and a greater number of multipliers for modular exponentiation. Increasing the bandwidth involves using the interconnect ideas from Section 2.2. Since one Toffoli gate takes at least 0.8 sec [2], while sending an EPR pair between two Qchips takes only $600\mu\text{sec}$ [22], given enough bandwidth we can easily pre-communicate all EPR pairs while the computation is in progress. Additional multipliers allow us to increase concurrency in the algorithm while requiring more Qchips. Figure 8 shows that it would take almost 13 days to factor a 1024-bit number if we could employ 4300 Qchips, and send 10,000 messages simultaneously between two Qchips. Each of these 4300 Qchips is 10cm^2 in area.

7 Conclusion

The technology for quantum computation has made extraordinary progress towards a practical quantum computer. The key to constructing a scalable system from these technologies, however, lies in classical principles of system design, parallelism, and communication. Our results indicate that a multi-chip implementation of ion-trap quantum computation is an attractive design point for the near future. Inter-chip communication through photon-mediated interaction allows low-latency and reasonable bandwidth. With careful partitioning of algorithms such as modular exponentiation, classically intractable problems can be solved by exploiting parallelism without overwhelming inter-chip bandwidth. Our hope is that this approach will facilitate practical implementation of future quantum algorithms as they arise.

References

1. P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *35'th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.
2. T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "A quantum logic array microarchitecture: Scalable quantum data movement and computation," *Proceedings of the 38th International Symposium on Microarchitecture MICRO-38*, 2005.
3. P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A* **54**, p. 2493, 1995.
4. A. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett* **77**, pp. 793–797, 1996.
5. D. Gottesman, "A class of quantum error-correcting codes saturating the quantum hamming bound," *Phys. Rev. A* **54**, p. 1862, 1996.
6. J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett* **74**, pp. 4091–4094, 1995.
7. M. Riebe, H. Haffner, C. Roos, and et. al., "Deterministic quantum teleportation with atoms," *Nature* **429**(6993), pp. 734–737, 2004.
8. V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A* **54**, p. 147, 1996.
9. D. Beckman, A. Chari, S. Devabhaktuni, and J. Preskill, "Efficient networks for quantum factoring," *Unpublished*, 1996.
10. R. V. Meter and K. M. Itoh, "Fast quantum modular exponentiation," *E-Print: quant-ph/0408006*, 2004.
11. S. Cuccaro, T. Draper, S. Kutin, and D. Moulton, "A new quantum ripple-carry addition circuit," *Unpublished*, 2004.
12. T. Draper, S. Kutin, E. Rains, and K. Svore, "A logarithmic-depth quantum carry-lookahead adder," *E-Print: quant-ph/0406142*, 2004.
13. S. Knowles, "A family of fast adders," *Unpublished*, 1985.
14. W. Wootters and W. Zurek, "A single quantum cannot be cloned," *Nature* **299**, pp. 802–803, 1982.
15. C. H. Bennett and et. al., "Teleporting an unknown quantum state via dual classical and EPR channels," *Phys. Rev. Lett.* **70**, pp. 1895–1899, 1993.
16. J. S. Bell, "On the Einstein-Podolsky-Rosen paradox," *Physics* **1**, pp. 195–200, 1964.
17. W. Dur, H. J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Phys. Rev. A* **59**, p. 169, 1999.
18. C. Bennett and et. al., "Purification of noisy entanglement and faithful teleportation via noisy channels," *Phys. Rev. Lett.* **76**, p. 722, 1996.
19. D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels.," *Phys. Rev. Lett.* **77**, pp. 2818–2821, 1996.
20. C. Monroe, "Quantum information processing with atoms and photons," *Nature* **416**, p. 238, 2002.
21. C. Cabrillo, J. Cirac, P. Garca-Fernandez, and P. Zoller, "Creation of entangled states of distant atoms by interference," *Phys. Rev. A* **59**, pp. 1025–1033, 1999.
22. B. Blinov, D. Moehring, L. Duan, and C. Monroe, "Observation of entanglement between a single trapped atom and a single photon," *Nature* **428**, pp. 153–157, 2004.
23. L. Duan, M. Lukin, J. Cirac, and P. Zoller, "Long-distance quantum communication with atomic ensembles and linear optics," *Nature* **414**, p. 413, 2001.
24. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.