Memory Hierarchies for Quantum Data

Dean Copsey[‡], Mark Oskin[†], Frederic T. Chong[‡], Isaac Chuang[◇] and Khaled Abdel-Ghaffar[‡]
[‡]University of California at Davis
[†]University of Washington
[◇]Massachusetts Institute of Technology

Abstract

Computing with quantum states may be the most efficient approach to solving some problems which take exponential time on conventional computers. Quantum states, however, are short-lived without constant error correction involving tremendous overheads. We propose to reduce these overheads by storing quantum data in a hierarchical structure. Large groups of quantum bits can be encoded together to reduce overhead in mass storage, but smaller groups are needed to support the working set of a computation.

1 Introduction

Quantum computation seeks to exploit the physics of quantum phenomena to achieve computation that scales exponentially with data size. The basic building block is a quantum bit, referred to as a *qubit*, is represented by nanoscale physical properties such as nuclear spin. Scalable quantum computation, however, cannot be achieved without extremely powerful error correction. In nature, quantum phenomena occur at extremely small scales. Environmental noise causes large scale systems to behave with the classical physics that we are all accustomed to seeing in everyday life. To build a quantum computer with thousands or even millions of quantum bits requires substantial effort to overcome environmental noise.

In fact, only recently have usable codes been developed and sustainable quantum computation been shown to be possible [Ste96a]. The key is to allow the basic operations of quantum computing to be applied directly to coded data without decoding and re-encoding the data. Such codes have made possible the Threshold Theorem: as long as the probability of error of each operation on a quantum computer is less than some constant (estimated to be as high as 10^{-4}), scalable quantum computers can be built using faulty components[KLZ98, Pre98, ABO97].

Existing error correction codes were developed primarily as theoretical constructs to enable the ground-breaking work that led to the Threshold Theorem. These codes are actually applied recursively to achieve the desired level of reliability. Unfortunately, the temporal and spatial overheads of such schemes is tremendous [OCC02].

The goal of this paper is to reduce the overhead of error correction for one major component of a quantum computer – the memory system. We observe that many quantum operations necessary for computation are not needed when storing bits in the memory system. This allows us to use lower overhead codes in memory and revert back to higher overhead codes for a "working set" of quantum data. This basic premise allows us to design a set of codes for a quantum memory hierarchy. The remainder of this paper describes work-in-progress towards this goal.

In the next section we provide the reader with some background material on quantum algorithms, primitive operations, error correction methods and teleportation. Using this prior work in quantum research, we examine a quantum memory hierarchy in Section 3 and then discuss future work and conclusions.

2 Background

While a bit in a classical computer represents either zero or one, a quantum bit can be thought to simultaneously represent both states. More precisely, the state of a qubit is described by *probability amplitudes*, which only turn into probabilities upon external observation. Unlike classical probabilistic computation, probabilities for different computational pathways can cancel each other out through interference, because the amplitudes are complex values (i.e, real/imaginary pairs). The actual probabilities are determined by the square of the amplitudes.

The key is that quantum computers directly manipulate probability amplitudes to perform a computation, and multiple qubits form vectors that can represent 2^n amplitudes with n qubits. In other words, a two-qubit vector simultaneously represents each of the states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ with some probability when measured. Each additional qubit in a qubit vector doubles the number of amplitudes represented. The work of a quantum computer is to manipulate these qubit vectors and their associated am-

plitudes in a useful manner. Manipulation of these qubit vectors leads to what is often called *quantum parallelism*, a useful way to begin thinking about what gives quantum computers such high potential speedups over classical computers. The difficulty is that we generally cannot look at the answer until the end of a computation, and then we only get a single random value from the vector! More precisely, measuring a qubit vector collapses it into a probabilistic classical bit vector, yielding a single state randomly selected from the exponential set of possible states. For this reason, quantum computers are best at "promise" problems – applications which use some hidden structure in a problem to find an answer which can be easily verified.

Designers of quantum algorithms must be very clever about how to get useful answers out of their computations. One method is to iteratively skew probability amplitudes in a qubit vector until the desired value is near 1 and the other values are close to 0. This is used in Grover's algorithm for searching an unordered list of n elements [Gro96]. The algorithm goes through $\sqrt{n}$ iterations, at which point a qubit vector representing the keys can be measured. The desired key is found with high probability.

Another option in a quantum algorithm is to arrange the computation such that it does not matter which of many random results is measured from a qubit vector. This method is used in Shor's algorithm for prime factorization of large numbers [Sho94], which is built upon the quantum Fourier transform, an exponentially fast version of the classical discrete Fourier transform. Essentially, the factorization is encoded within the period of a set of highly probable values, from which the desired result can be obtained no matter what value is measured. Since prime factorization of large numbers is the basis of many modern cryptographic security systems, Shor's algorithm has received much attention.

2.1 Basic Quantum Operations

Figure 1 gives a few basic quantum operations that we will use in our quantum architecture. These include one-bit operations such as the Hadamard, bit-flip, and phase-flip; as well as the two-bit controlled-not. These are given in both their circuit representation and their matrix representation. The matrix representation involves multiplying the operator times the amplitude vector of the quantum states. C_i^2 gives the probability of state i (denoted with the notation $|i\rangle$) and the sum of all these probabilities must equal one. Preserving this sum is equivalent to conserving energy and requires that all operations be reversible.

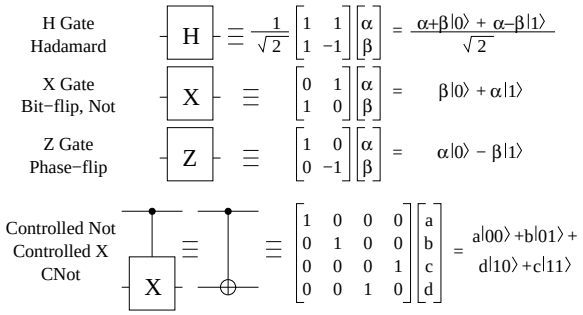

Figure 1: Basic quantum operations

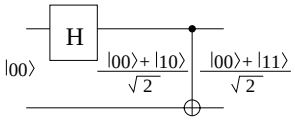

Figure 2: Creating a "cat" state

The bit-flip exchanges the probabilities of the two states, while the phase flip changes the sign between them. The Hadamard takes the two states and "mixes" them to a "halfway" state. The controlled-not does a bit-flip if the control qubit is 1. These basic gates, along with the measurement of qubits, form the basic operations used for data transport.

Figure 2 shows the *entanglement* of two qubits in a Schrödinger "cat" state, also known as a Bell state or an EPR state after Bell, Einstein, Podolsky and Rosen, who were among the first to investigate such states. The values of the two qubits are tied together: whatever value is measured for the first qubit will also be measured for the second qubit. The amplitudes for $|01\rangle$ and $|10\rangle$ are zero.

2.2 Quantum Error Correction

Quantum phenomena are constantly evolving with time. Atoms decay. Electrons change orbitals by absorbing or emitting photons. Magnetic spin states of nuclei flip due to external magnetic fields. A system cannot be isolated to the point where it is completely stable. Hence, if two qubits are in an entangled state, they will eventually *decohere* due to entanglements with the environment.

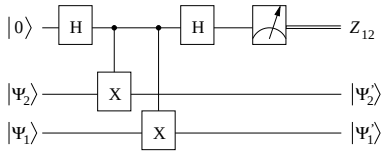

Figure 3: Measuring Z_{12}, the phase difference between Ψ_2 and Ψ_1

Quantum error correction has much in common with its classical counterpart. A logical qubit is encoded redundantly in a number of physical qubits. The *syndrome* of the physical qubits can be measured to determine whether an error may need to be corrected. One difficulty in quantum codes, however, is that the physical qubits can not be measured without destroying their state.

We can get around this by using an ancillary zero qubit, shown in Figure 3, which is interacted (using an H-gate, or "mixing" operation) with the original physical qubit, $|\Psi\rangle$. The ancillary qubit extracts information about the error within $|\Psi\rangle$, but no information about the data value of $|\Psi\rangle$ itself. A simple analogy is imagine the XOR operation of two bits; the result is whether or not the two bits are different (e.g. error), but not any information about the bits absolute value. We can then measure the ancilla to determine the syndrome of $|\Psi\rangle$. What is interesting is that $|\Psi\rangle$ is not unaffected by this interaction and measurement. The measurement collapses the waveform in the error-syndrome basis, and the resulting $|\Psi'\rangle$ is either the correct value, the value with the opposite phase, the value inverted, or the value inverted with the opposite phase. So syndrome measurement actually accomplishes two tasks: the syndrome is measured and the *continuous* errors are transformed into *discrete* errors which are easy to fix using a Z-gate. Table 1 gives a simple example of how these fixes are done for a 3-qubit Shor code [Sho95], where Z_{ij} is -1 if the i-th and j-th qubits differ in phase, +1 otherwise. Since errors can occur in both phase and amplitude, we perform a similar operation using X gates to measure and correct amplitude. To have both amplitude and phase correction at the same time, however, the Shor code requires that 1 logical qubit to be encoded into 3 bits for phase correction, then those 3 bits need to each be encoded into 3 bits for amplitude correction, resulting in a 9-bit code.

Shor's code, based on the classical error correction

Z_{01}	Z_{12}	Error Type	Action
+1	+1	no error	no action
+1	-1	bit 3 flipped	flip bit 3
-1	+1	bit 1 flipped	flip bit 1
-1	-1	bit 2 flipped	flip bit 2

Table 1: Phase correction for a 3-qubit code

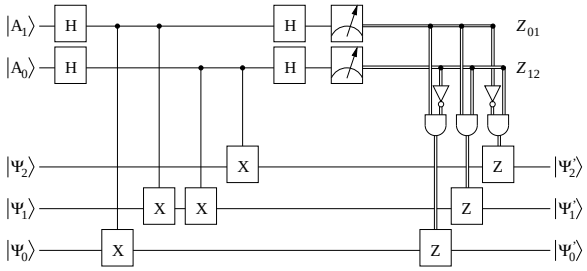

Figure 4: Syndrome Measurement for 3-bit Code

method of repetition, is termed a $[[9,1,3]]$ code: nine physical qubits, encoding one logical qubit, with a Hamming distance of three. (A code with a distance of d is able to correct $(d-1)/2$ errors.) Shortly after Shor demonstrated the $[[9,1,3]]$ code, he and Calderbank[CS96], and independently Steane[Ste96b], showed how to create quantum error correction codes based on classical codes. One such code is the $[[7,1,3]]$ code attributed to Steane. Further refinements and generalizations led to *stabilizer* codes, such as the $[[5,1,3]]$ code [LMPZ96], which is the smallest (densest) known encoding of a single qubit, and the $[[8,3,3]]$ code [Ste96c, Got96, ACS97], the densest three-qubit code.

As mentioned above, errors in quantum circuits are not limited to full phase or bit flips, but can be any complex-valued linear combination of the two. However, when the error syndrome is measured, the state of the qubits is collapsed in the error syndrome basis: measuring the error effectively quantizes it so that only X, Z, and XZ operators need be applied to correct it.

Qubits are subject to decoherence when they interact with the environment. Applying an operator to a qubit is just such an interaction. It would be nice to be able to apply operators in such a manner that any errors introduced could be corrected. In fact, if operators could be applied to encoded qubits, any errors introduced by the operator *could* be detected and corrected. The Calderbank-Shor-

Steane (CSS) codes have a nice property: operators can be applied on the encoded (logical) bits by applying relatively simple operators on the encoding (physical) bits. For example, to apply an X operator on a qubit encoded by the $[[7,1,3]]$ code, simply apply the X operator on each physical qubit. The same is true for the Z, H, Y ($= iXZ$), and $CNot$ operators. The S (rotate by $\pi/4$) operator requires applying ZS. The last operator required to create a *universal* set [1], the T (rotate by $\pi/8$) operator, requires a slightly more complicated procedure. But as long as the probability of an error, p, is below a certain threshold, $1/c$ (about 10^{-4} for the $[[7,1,3]]$ Steane code), any number of operations may be performed with an overall probability of error of cp^2.

If a logical qubit is encoded in n physical qubits, it is possible to encode each of those n qubits with an m-qubit code to produce an mn encoding. Such *concatenation* of codes can reduce the overall probability of error even further. For example, concatenating the $[[7,1,3]]$ with itself gives an overall probability of error of $c(cp^2)^2$. Concatenating it k times gives $(cp)^{2^k}/c$, while the size of the simulating circuit increases by d^k and the time complexity increases by t^k, where d is increase in circuit complexity for a single encoding, and t is the increase in operation time for a single encoding. For a circuit of size $p(n)$, to achieve an accuracy of ϵ, then k must be chosen such that [NC00]:

$$\frac{(cp)^{2^k}}{c} \leq \frac{\epsilon}{p(n)}$$

The number of gates (operators) to achieve this result is $O(\mathrm{poly}(\log p(n)/\epsilon)p(n))$, provided p is below some threshold.

2.3 Teleportation

Although many quantum error correction codes could be used in a quantum computer, converting between them can be problematic. In fact, conversion between codes can randomly propagate errors across qubits and compromise reliability. Fortunately, there is a special way to convert between codes that avoids this problem. This method involves the quantum primitive of teleportation [GC99]. As it turns out, teleporation is not only a good way to convert between codes, but it is also a good way to transport quantum data between the different parts of our memory system.

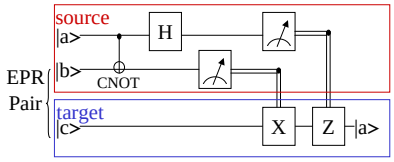

Figure 5: Quantum Teleportation

Contrary to its science fiction counterparts, quantum teleportation is not the instantaneous transmission of information. Rather, it is the re-creation of a quantum state at a destination using some classical bits that must be communicated along conventional wires or other mediums. In order for this to work, we need to pre-communicate an EPR pair. We use the relationship between the pair to achieve teleportation.

Figure 5 gives a more precise view of the process. We start with an EPR pair at the source end of the wire. We separate the pair, keeping one qubit (b) at the source and transporting the other (c) to the destination (quantum data transport is denoted by the solid lines). When we want to send a quantum bit of data (a), we first interact a with b using a $CNot$ and a Hadamard gate. We then measure the phase and the amplitude of a (measurement is denoted by the meters), send the two one-bit results to the destination classically (classical communication is denoted by the double lines), and use those results to re-create the correct phase and amplitude in c such that it takes on the state of a. The re-creation of phase and amplitude is done with controlled-X and Z gates, which perform the same function as the gates described in Figure 1 but contingent on a classical control bit (the measurements of a).

Note that the original state of a is destroyed once we take our two measurements. This is consistent with the "no-cloning" theorem, which states that a quantum state can not be copied. Intuitively, since c has a special relationship with b, interacting a with b makes c resemble a, modulo a phase and/or amplitude error. The two measurements allow us to correct these errors and recreate a at the destination.

In order to use teleporation to convert between different error coding schemes, the two halves of our EPR pair are encoded into the source and destination codes. The source and destination qubits are then interacted bitwise with the

[1] With a universal set of operators, any quantum operator may be approximated to any desired accuracy.

Recursion level (k)	Storage overhead	Operation overhead	Min. time overhead
0	1	1	1
1	7	153	5
2	49	23,409	25
3	343	3,581,577	125
4	2,401	547,981,281	625

Table 2: Overhead of recursive error correction for a single qubit operation

respective EPR halves just as in the basic teleportation algorithm. The result is a "quantum wire" which gives us a means to both transport quantum data and convert between different error correction codes.

3 Quantum Memory Hiearchy

The previous section introduced a number of quantum concepts, some theoretical (algorithms, error correction, teleportation), and some practical (decoherence rates, primitive operations). Our goal, as architects, is to reason about and abstract the structural components and organization of a future quantum processor. To do this, we will examine the practical aspects, and what challenges will arise from them. Then by applying the theoretical tools we can deduce the structure of a quantum processing system from the optimization of the underlying tradeoffs.

We focus our attention on Shor's algorithm. Due to its exponential speedup over any known classical algorithm for factoring, it is one of the primary motivatations to study quantum computing. Our goal in this section is put together an analysis of the spatial and temporal locality of Shor's algorithm together with reasonable estimates of quantum device technology to arrive at a system structure for a quantum computer. One assumption that we will make is that a quantum processor will compute in a $[[7,1,3]]$ code, or a recursive (concatenated) variant ($[[49,1,7]]$, $[[343,1,15]]$, etc.). The reason for this is the relative ease with which most quantum primitives are performed on this code. Table 2 depicts the storage and operational overhead of using these codes. The numbers are for a single logical qubit, with rows depicting increased levels of concatenation, and thus additional error correction capability.

To factor a 1024-bit product of two primes, thereby breaking 1024-bit RSA encryption, requires roughly

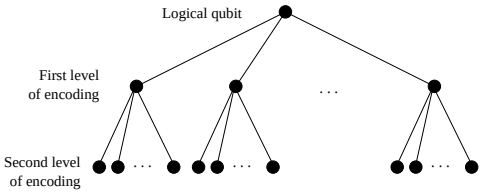

Figure 6: Tree structure of concatenated codes

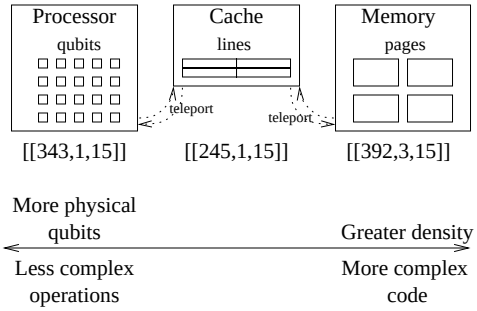

Figure 7: Trading computational ease for density

$p(n) =$520 billion (5.2×10^{11}) operations on a quantum computer. For our calculations we will assume an aggressive, but not overly optimistic quantum device technology with a decoherence rate per operation of $p = 10^{-6}$. This implies that in order to reliably execute Shor's algorithm the operations will be performed on logical qubits in a $[[343,1,15]]$ code word (i.e. k, the recursion level is equal to three).

The underlying architectural question is: should we use the $[[343,1,15]]$ code throughout the quantum processor, or should we use teleportion as a code-conversion tool and exploit denser codes for storage and the $[[343,1,15]]$ code only for the computational components of the system? If we move to a denser code for storage there will be costs, such as increased access time, and increased structure to the quantum data.

One method of visualizing a $[[343,1,15]]$ code is as a tree of codes, with each node in the tree being a $[[7,1,3]]$ code (see Figure 6. It is straightforward to replace the topmost node of this tree with either the $[[5,1,3]]$ code or the $[[8,3,3]$ code and thereby generate a $[[245,1,15]]$ code or a $[[392,3,15]]$ code. While the number of physical qubits for the $[[392,3,15]]$ code is larger (392 compared to 343), this code encapsulates 3 logical qubits instead of just 1. Hence, the overhead per-logical qubit is $\approx$ 131 (Table 3).

Code	Storage overhead	Operation overhead	Min. time overhead
none	1	1	1
$[[8,3,3]]$	2.67	$276/3 = 92$	7
$[[8\times 7,3,7]]$ $= [[56,3,7]]$	18.67	$92 \times 153 =$ $14,076$	35
$[[8\times 7\times 7,3,15]]$ $= [[392,3,15]]$	130.67	$14,076 \times 153 =$ $2,153,628$	175

Table 3: Overhead of $[[8,3,3]]$ concatenated with $[[7,1,3]]$ on a per-qubit basis

We can exploit this denser packing by creating a quantum memory hierarchy that is analogous to classical hierarchies. Instead of a classical cache based upon high-speed SRAM, we use a "quantum cache" based upon less-dense error correction codes. This cache (which has a line size of 3 qubits in this paper) provides the random access required by the quantum ALU.

Once we have such a cache we will want some method to efficiently utilize it. Fortunately, like their classical counterparts, quantum algorithms can be structured to exhibit spatial and temporal locality. The Quantum Fourier Transform (QFT), integral to Shor's algorithm can be blocked, to further increase locality.

Figure 8 shows a nine-bit QFT. In Figure 9, the operators have been reordered so that groups of operators involve two of three sets of three qubits each. (The X—X operators "swap" two qubits, and are implemented by three $CNot$'s. The R_k gates are "rotate by $\pi/2^k$" gates.) In this case, the third set of qubits can be cached while operators are applied to the first and second sets; the first set is then cached while operators are applied to the second and third sets, and finally, the second set is cached while operators are applied to the first and third sets. The three logical qubits being cached require 392 physical qubits, compared to 1,029 if they had not been cached, a space savings of more than 60%.

Operationally, there is considerable savings. The savings is actually greater than that suggested by Tables 2 and 3. The number of operations required by the $[[343,1,15]]$ code is stated in per-logical operation terms. Since operators typically are not applied to the cache and memory (with the exception of teleportation), the overhead is relative to the refresh rate of the cache. A general assumption is that qubits decohere much more slowly when operations aren't being applied. So the operational savings is potentially much more, depending on the properties of the underlying quantum physical system. If we assume the decoherence rate for an operation is ten times what it is for an idle qubit in the same time frame, then the operational overhead for the $[[343,1,15]]$ code is about fifteen times that of the $[[392,3,15]]$ code.

4 Future Work

In this paper, we focused on concatenated and CSS codes due to their ease of use. Other non-concatenated (flat) codes exist that are denser and may perform better.

In general, the more qubits encoded together, the higher the operational overhead. Also, achievable parallelism depends on the ability to get adjacent qubits to interact, since operators require physical interations. Lastly, the refresh rate depends on the underlying quantum physical properties. More overhead can be tolerated with greater parallelism or longer times between refreshes (i.e., less environmental decoherence per unit time). Thus, the choice of an error correction code depends on the physical properties and structure of the underlying quantum phenomena.

As noted above, the quantum Fourier transform is currently the most important quantum algorithm. More work needs to be done on the relationship between the physical structure and properties of quantum computers, and the impact of cacheing on QFT.

Larger codes in general have more operational overhead, but work with the $[[8,3,3]]$ code indicates that the per-qubit cost may be less. More investigation needs to be done on the properties of dense-code operators.

Finally, the current work does not look at enough qubits to warrant a three-level hierarchy. However, as quantum algorithms evolve, one can envision a need for thousands or millions of qubits. At that point, dense coding schemes become very important. Other codes should be investigated that can provide better density and/or stability. For example, preliminary work on toric and high-genus surface codes suggests they can be self-stabilizing [DKLP01, Den00]. Also, quantum codes based on iteratively decoded classical codes [MN95] have the promise of being extremely dense.

5 Conclusion

We envision a hierarchical quantum memory that exploits temporal and spatial locality to make use of multiple

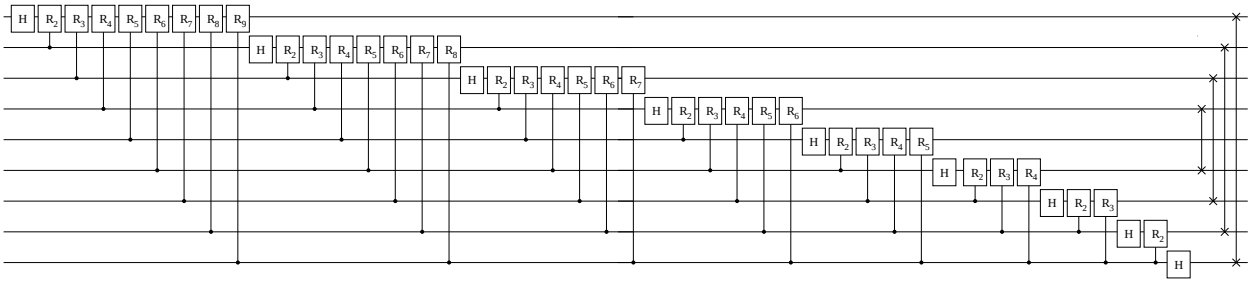

Figure 8: Quantum Fourier transform on nine qubits

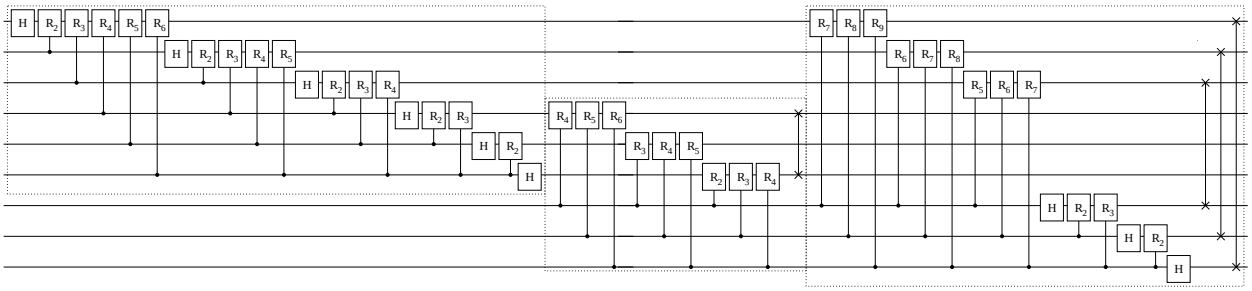

Figure 9: Locality in the quantum Fourier transform

levels of error coding. A concatenated code based on $[[7, 1, 3]]$ could serve for registers on which calculations are performed. Multi-qubit codes could serve for quantum data at the cache line granularity. Other denser and more stable codes could serve for quantum data at the page level granularity.

Preliminary results on the quantum Fourier transform algorithm are promising, and warrant further investigation.

References

[ABO97] D. Aharonov and M. Ben-Or. Fault tolerant computation with constant error. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 176–188, 1997.

[ACS97] P. Shor A. Calderbank, E. Rains and N. Sloane. Quantum error correction and orthogonal geometry. *Phys. Rev. Lett.*, 78:405–409, 1997.

[CS96] A. Calderbank and P. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098, 1996.

[Den00] Eric Dennis. Quantum codes on high-genus surfaces. 2000.

[DKLP01] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. 2001.

[GC99] D. Gottesman and I. L. Chuang. Quantum teleportation is a universal computational primitive. *Nature*, 402:390–392, 1999.

[Got96] D. Gottesman. A class of quantum error-correcting codes saturating the quantum hamming bound. *Phys. Rev. A*, 54:1862–1868, 1996.

[Gro96] L. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28^{th} Annual ACM Symposium on the Theory of Computation*, pages 212–219, New York, 1996. ACM Press.

[KLZ98] E. Knill, R. Laflamme, and W. H. Zurek. Resilient quantum computation. *Science*, 279(5349):342–345, 1998. arXive e-print quant-ph/9702058.

[LMPZ96] R. Laflamme, C. Miquel, J.-P. Paz, and W. H. Zurek. Perfect quantum error correction code. *Phys. Rev. Lett.*, 77:198, 1996. arXive e-print quant-ph/9602019.

[MN95] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In Colin Boyd, editor, *Cryptography and Coding. 5th IMA Conference*, number 1025 in Lecture Notes in Computer Science, pages 100–111. Springer, Berlin, 1995.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2000.

[OCC02] Mark Oskin, Frederic T. Chong, and Isaac L. Chuang. A practical architecture for reliable quantum computers. *IEEE Computer*, 35(1):79–87, January 2002.

[Pre98] J. Preskill. Fault-tolerant quantum computation. In H.-K. Lo, T. Spiller, and S. Popescu, editors, *Quantum information and computation*. World Scientific, Singapore, 1998.

[Sho94] P. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35^{th} Annual Symposium on Foundations of Computer Science*, page 124, Los Alamitos, CA, 1994. IEEE Press.

[Sho95] P. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:2493–2496, 1995.

[Ste96a] A. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77:793–797, 1996.

[Ste96b] A. Steane. Multiple particle interference and quantum error correction. *Proc. R. Soc. London A*, 452:2551–76, 1996.

[Ste96c] A. Steane. Simple quantum error correcting codes. *Phys. Rev. Lett.*, 77:793–797, 1996.