Microcoded Architectures for Ion-Trap Quantum Computers

Lucas Kreger-Stickles and Mark Oskin
University of Washington
Department of Computer Science & Engineering
{lucasks,oskin}@cs.washington.edu

Abstract

In this paper we present the first ever systematic design space exploration of microcoded software fault tolerant ion-trap quantum computers. This exploration reveals the critical importance of a well-tuned microcode for providing high performance and ensuring system reliability. In addition, we find that, despite recent advances in the reliability of quantum memory, the impact of errors due to stored quantum data is now, and will continue to be, a major source of systemic error. Finally, our exploration reveals a single design which out performs all others we considered in run time, fidelity and area. For completeness our design space exploration includes designs from prior work [13] and we find a novel design that is $\frac{1}{2}$ the size, 3 times as fast, and an order of magnitude more reliable.

1. Introduction & Prior Work

Ion traps present the most promising technology from which to build a large-scale quantum computer. They are fabricated using existing silicon-manufacturing facilities at scales that are relatively easy to experiment with. Nonetheless, only extremely small-scale experimental systems have been built, and many of the fundamental technology parameters, such as fidelity and operation timing, remain in flux. Despite this degree of technological uncertainty, computer architects [15] have been successful in helping to shape the directions of device physics research [7]. Additionally, prior work [2] has demonstrated the importance of architecture development, showing it to be key in making progress towards large scale quantum computing a reality. Ultimately then, it is the partnership between device researchers, algorithm specialists, and computer architects that will lead to the advancement of quantum computing technology.

As prior work has articulated [2], microarchitecture research requires at least two things: a representative benchmark suite and a solid grasp of the underlying technology. Prior work [2] in the architecture of quantum computers has established that, due to the extremely high overhead of software-based error correction techniques, the best representative benchmark is the error correction process. In fact, the overhead of quantum error correction is so high ($> 99\%$) that the actual computation that a quantum computer performs is only of secondary concern to the architect. Truly critical is insulating user programs from the error correction process by designing an architecture and a *microcode* that leverages the architecture to provide an ISA abstraction of fault-tolerant quantum operation.

Unfortunately, quantum computing device technology is evolving in ways that are difficult to predict. Researchers have struggled with how to proceed and taken several approaches: Quantum theoreticians [5, 18, 1] and algorithms designers [16] take a very high level approach, either ignoring error and communication costs or using generic abstractions. Our earlier work [2] presumed that while the specific timing and error rates across operations and communication methods might be in flux, the *relationship* between them would remain fixed. This allowed us to view all sets of technology parameters on a single axis from worst to best. That assumption has begun to break down, as certain components experienced dramatic performance and fidelity improvements [10] while others have progressed at a slower pace [6]. More recent work such as [13, 20] operates under the presumption that the long-term projections [21] of device researchers in the space will eventually be realized and use those as the basis for design. We are concerned by this strictly optimistic approach since relying on such projections, even within conventional computing, is tenuous at best. Moreover, such an approach isn't conducive to leveraging architecture research to hasten the arrival of large-scale quantum computation, tells us nothing about how to best utilize the components we do have, and doesn't help us generate general lessons about the design space.

In this paper, we take a different approach. We develop an evaluation methodology based on using behavioral simulation coupled with potential fault point counting. This method allows us to be largely technology parameter agnostic and cleanly separate research results which are predicated on a set of parameters from those which are universal and applicable independent of where the underlying technology progresses.

We utilize this methodology as the basis for the first-ever thorough design space exploration of ion-trap quantum microarchitectures performed to date. Our design space exploration includes designs found in prior art [13, 20], several logical extensions which are nearby in the design space, and some entirely new configurations. In addition, we take on the challenge of designing architecturally fault tolerant microcode[1]. This is a requirement for fault tolerant quantum computation, and one that has been neglected by prior work for its perceived complexity.

While work continues on the development of the component pieces of a quantum computer and on algorithms for fault tolerant quantum computation, work on the study of quantum architectures has been limited. Our previous work [2] developed a straw-man architecture for use with our evaluation tools but since then the only two attempts at designing ion-trap quantum architectures which have appeared in the architecture community are the QLA [13] and follow-on CLQA [20]. This is unfortunate since our data indicates that architectural design has a far greater impact on system performance and fidelity than control operations. In fact, through careful construction, we are able to utilize

[1] We use this term to describe the use of the same hardware resource by data bits for which correlated errors would negate fault tolerance. Avoiding such reuse isolates the fault tolerance scheme from the impact of especially faulty components.

the same basic algorithm for software fault tolerance as prior work [18] but with an architecture and microcode which is $\frac{1}{2}$ the size, 3 times as fast, and an order of magnitude more reliable. Moreover, this design is superior to all others we considered across a wide range of reasonable technology parameters. This indicates it will continue to be effective as the technology matures: serving today as the basis for experimentation and later as the basis for large-scale computation.

Our design space exploration has revealed several important lessons pertaining to software error correction on an ion-trap architecture which generalize across specific technology assumptions.

- A phenomenon known as *storage* or *memory* errors cannot, as prior research [13, 20] has suggested, be ignored even when using the most optimistic projections for quantum computing device technology. We find our best design, given the most optimistic technology parameters, has over 3500 potential points of failure related to storage, compared to ≈ 50 for all other potential fault points.

- Due to the high cost in time and fidelity of communication given even the most optimistic projections [21], algorithms and designs must utilize locality and minimize communication to achieve high reliability. This is true even if the number of communication related operations is small compared to all other potential sources of error. In fact, using optimistic projections for the technology, we find that for many designs nearly 50% of systemic failures occur due to errors in communication.

- Designs that are nearby in the microarchitecture design space vary *significantly* in resulting performance and fidelity.

- The microarchitecture of a quantum computer can have a greater impact on fidelity and performance than algorithmic advancements. In particular, algorithms that appear efficient due to fewer control operations can be substantially slower and less reliable than others once mapped onto hardware. In one instance a design which utilizes a seemingly more efficient algorithm has over 4 times the number of potential points of failure than another, seemingly less efficient design.

- Designing for robust fault tolerance requires considerations at the architectural level as well as the software level. Prior work [13, 20, 2] implemented the software error correction process correctly but did not consider the impact of correlated hardware errors. In this paper we address the issue.

The rest of this paper is structured as follows: Section 2 briefly covers the state of the art of ion-trap systems and the basics of software-based error correction schemes for quantum computation. Section 3 addresses the methods we use to explore the design space and evaluate the results in a largely technology parameter agnostic way. Sections 4 and 5 present the designs we explored and their evaluation. Section 6 discusses avenues for future research and presents some preliminary results of that work. Section 7 concludes.

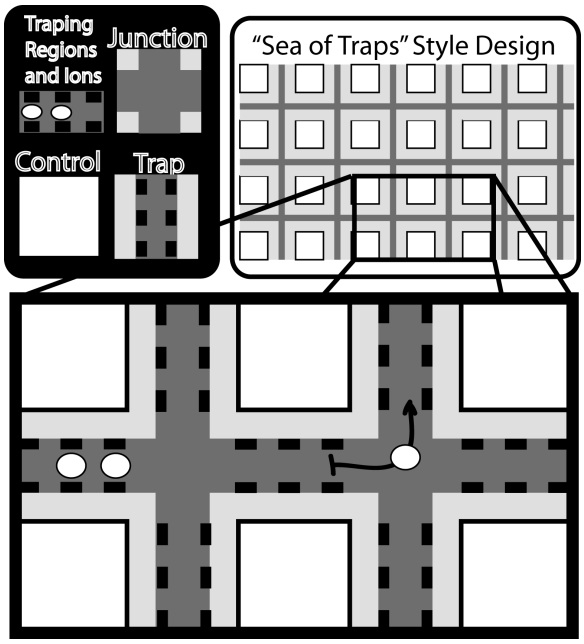

Figure 1. The Basic Building Blocks of an Ion-Trap Quantum Computer showing an inset from a "Sea of Traps" style homogonous microarchitecture as propossed in [12, 3]. **Traps** contain **trapping regions** which hold and manipulate **ions** which are the physical manifestation of **qubits** and store quantum state. **Ions** move around the computer by moving through traps and turning corners via **junctions**.

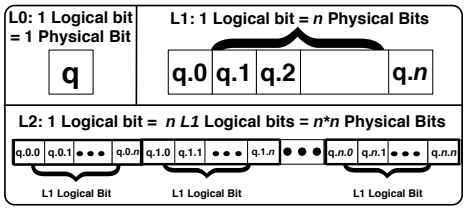

Figure 2. Recursive Logical Encodings Showing logical qubits encoded at 0,1, and 2 levels of error correction and utilizing 1,n and n^2 physical bits respectively.

2. Ion-Trap Computers and Quantum Error Correction

At the highest level, ion-trap technology consists of 4 basic components, shown in Figure 1. **Ions** are the physical manifestation of a quantum bit or **qubit**. An ion's state is manipulated through the use of lasers which reside outside the quantum computer[2]. Since quantum state cannot be copied [22], ions must be physically relocated in order to interact with one another. Because the ions have a charge this can be done through the use of magnetic fields which propel ions around corner **junctions**, between and through **traps**, hold them in place in **trapping regions**, and **split** them apart after interaction. In addition to these communication operations, there are a number of operations which manipulate a

[2]We do not, nor has prior art, considered the lasers as part of the architecture design since their placement is highly flexible and largely orthogonal to the rest of system design.

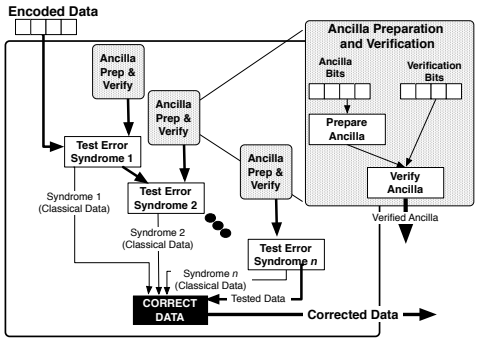

Figure 3. Highlevel Overview of the Error Detection and Correction Process, including ancilla preparation and verification

qubit's state.

Unfortunately, all of these operations are highly error prone and it is unlikely that component fidelity rates will ever be high enough to support large scale quantum computation natively [14, 2]. Fortunately, software techniques have been developed to make quantum computers fault tolerant. These software techniques work by operating on **logical** qubits which are encoded redundantly across many **physical data qubits**. In order to ascertain what physical bits, if any, have become corrupted a set of additional qubits (called an **ancilla**) is prepared into a well known state. Since this preparation is also error prone, yet another set of qubits (called **verification** qubits) are used to verify the ancilla. A single ancilla cannot check for all correctable errors, thus multiple ancillas are required. If one layer of error correction is insufficent to make a computation reliable, the process can be recursively applied (error correcting the error correction process!). Figure 3 illustrates this process. While this very high level of understanding is sufficient to appreciate the results of this paper, we encourage the interested reader to explore fundamental literature on the subject [19, 14, 1].

3. Methods

3.1. Design Space Exploration

The goal of our experimental setup and evaluation methodology is to explore the space of software fault tolerant ion-trap architectures in a way that is flexible enough to cope with the wide range of experimental and projected technology parameters in Table 1. The first step to realizing this goal lies in our decision to perform a systematic design space evaluation which is largely independent of specific technology parameters. The next step is to develop a measure of system fidelity which lets us abstract component fidelity.

Our exploration of the design space addresses both the layout and sizing of the hardware components and the microcode which utilizes these resources to provide an abstraction of a fault-tolerant quantum computer to the program. User software should not have to concern itself with the error correction process. Our microcode utilizes the Steane code [17] for software fault tolerance. The Steane code encodes a logical qubit across 7 physical qubits and can recover from a single error on any one of the encoding bits. The primary reason for this choice was for ready and fair comparison against the body of prior work [13, 20, 2] which utilizes it.

Microarchitecturally, many of the designs we consider appear similar – they are all "sea of traps" designs; the differences are one layer down. Each has different microcode and different allocations of ions to traps and tasks and thus will utilize the physical resources differently. These differences will enable builders to specialize areas of the machine, for instance by removing control logic.

We start our domain space exploration (**DSE**) by considering 4 methods for preparing ancillas for use with the Steane code. We then consider 3 hardware structures on which to map the ancilla processes. Moving up to the full level one (L1) error correction process, we consider whether to perform the aforementioned ancilla preparations adjacent to the encoded data or offset from it. Next, we explore the impact of varying the number of ancillas for a subset of the most promising designs. This exploration reveals one universally superior design for L1 across all technology parameters. Figure 4 provides an overview of how a particular ancilla preparation method maps onto the hardware for the process of error correction at L1.

Our exploration of the design space concludes with the full L1 error correction process. Since no quantum computer has been built which is large enough to support computation at L1 we have focused our work on making this process as efficient and reliable as possible. Furthermore, since L2 is built on top of L1, any improvements in the L1 process will have immediate impact at L2. Though space does not permit inclusion of a full exploration of the L2 design space, Section 6 (Future Work) provides an overview of novel directions our research is headed and some preliminary results.

3.2. Evaluation Methodology

Our evaluation methodology relies on potential fault point counting (**PFPC**). With this technique every operation is recorded as a potential point of failure. Then, given an estimate of operation fidelities one can compute the probability that more errors occur than the EC scheme can accommodate, yielding an estimate of system fidelity after error correction. This method of evaluation has broad precedent [14, 19, 5].

The complicating factor in this setup is the impact of storage errors. Since storage errors are more likely to occur the longer a qubit sits idle, our PFPC is directly dependent on component timings. Fortunately, as shown in Table 1, the range of potential timings is much smaller than the range of potential fidelity numbers. Furthermore, timing parameters do not impact the other sources of error. Therefore, we can construct a PFPC for all other sources of error that will remain constant across all timings and augment this with storage PFPCs based on a small set of exemplary timing conditions. These timing scenarios are displayed in Table 1 and are labeled LOW, MED and HIGH.

To gather data we built a behavioral simulator which takes as input a candidate architecture, a timing scenario, and a microcode. The microcode is a hand-scheduled program for the error correction process which represents the assignments of qubits to traps, interactions of qubits, and the paths they take to interact with one another. Thus the microcode encodes how to efficiently utilize the underlying machine resources for fault-tolerant quantum computation. This schedule is used by the simulator to track the flow and source of potential fault points (PFPs), counting all points which have the

Timing and Error Ranges							
Operation	**Current Time**	**Optimistic Projection**	**Current Fail Rate**	**Optimistic Projection**	**LOW**	**MED**	**HIGH**
Unary Op	$1\mu s$	$1\mu s$	$1 * 10^{-4}$	$1 * 10^{-8}$	$1\mu s$	$1\mu s$	$1\mu s$
Binary Op	$10\mu s$	$10\mu s$	$3 * 10^{-2}$	$1 * 10^{-7}$	$10\mu s$	$10\mu s$	$10\mu s$
Measure	$200\mu s$	$10\mu s$	$1 * 10^{-2}$	$1 * 10^{-8}$	$10\mu s$	$10\mu s$	$10\mu s$
Split	$200\mu s$	$.1\mu s$	$\star$	$1 * 10^{-8}$	$1\mu s$	$10\mu s$	$10\mu s$
Move	$20\mu s$	$.1\mu s$	$\star$	$1 * 10^{-6}$	$1\mu s$	$1\mu s$	$10\mu s$
Turn	$20\mu s$-80ms	$10\mu s$	$\star$	$2 * 10^{-6}$	$10\mu s$	$100\mu s$	$100\mu s$
Store	Measured in Pr_{fail} per $1\mu s$		$7 * 10^{-8}$	$7 * 10^{-9}$	$1\mu s$	$1\mu s$	$1\mu s$

Table 1. Timing and Error Rates for Operations in an Ion-Trap Computer: Numbers given represent the current state of the art as well as optimistic projections into the future. LOW, MED, and HIGH indicate the timings used in our behavioral simulator. $\star$ Represents parameters for which no good experimental results currently exist because experiments have measured fidelity in terms of how many ions flew out of the trap as opposed to how many ions maintained their state after the operation. These numbers are derived from [6, 13, 20, 10, 21]

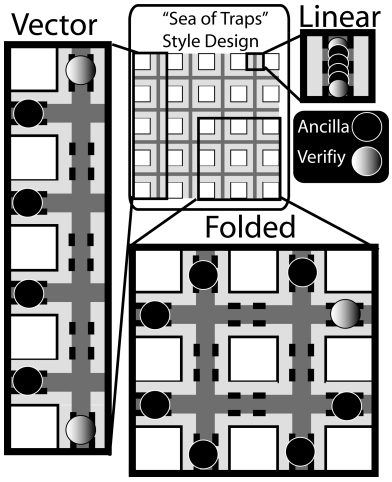

Figure 5. Ancilla Preparation Designs for L1 Error Correction shown in the context of a "Sea of Traps" style micro architecture and annotated with initial "home" placements for qubits.

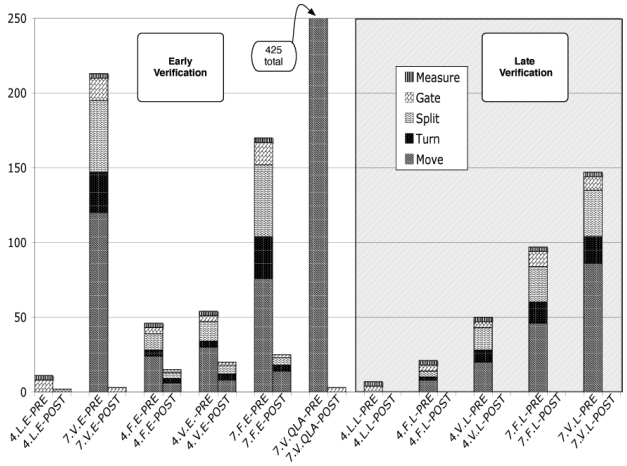

Figure 6. Ancilla Level Control and Communication Potential Fault Point Counts in increasing order of post verification fault counts, seperated by early and late verification techniques. (Shorter is better)

potential to impact encoded data.[3]

Post simulation we have a set of PFPs for each operation type along with a count of storage PFPs related to a variety of timing scenarios for each candidate architecture. This data is then used to infer characteristics of each architecture and their relative sensitivity to each type of error. For illustrative purposes, we also provide specific fidelity estimates at L1 which were calculated by using a modified version of the error estimator for PFPCs described in [19]. For comparison we use the fidelity parameters from prior work [13] as input. This corresponds to the optimistic column in Table 1.

4. Ancilla Exploration

We begin our DSE by exploring two algorithms for preparing and utilizing ancillas. The simplest method involves 4 qubits per ancilla and requires that a total of 6 states be measured [14]. For fault tolerance, each of these states must be measured at least twice [19] for a grand total of 12 ancilla preparations and interactions. A more complicated method exists which utilizes 7 qubit ancillas and employs more complicated conventional computation to ascertain the nature of an error on the data with just 2 state measurements (for a total of 4 ancilla preparations and interactions) [19].

Each of these methods has an alternate implementation which we consider: For the 7-bit method, 4 verification bits are used to ensure that the ancilla has been properly prepared. There exists a more compact implementation [19] which verifies as it goes using only one verification bit. While there are fewer control operations in this method, the disadvantage is that bits spend more time waiting after verification (which can be a problem if storage errors are substantial), and it artificially serializes the verification process leading to a longer critical path. The 4 bit method typically employs either 2 or 3 additional verification bits but a technique exists by which all preparation and verification can be done by interacting only those bits which are nearest neighbors on a linear array. The advantage is a potentially substantial reduction in communication costs. The disadvantage is additional control operations and a limit on the number of verification bits which can be simultaneously employed, leading to a longer critical path. Statistics regarding all four of these methods can be found in Table 2. We separate those operations which

[3]Not all PFPs have the potential to impact data. For instance, after an ancilla has been verified the verfication bit sits idle but any storage errors on that bit have no way to impact encoded data.

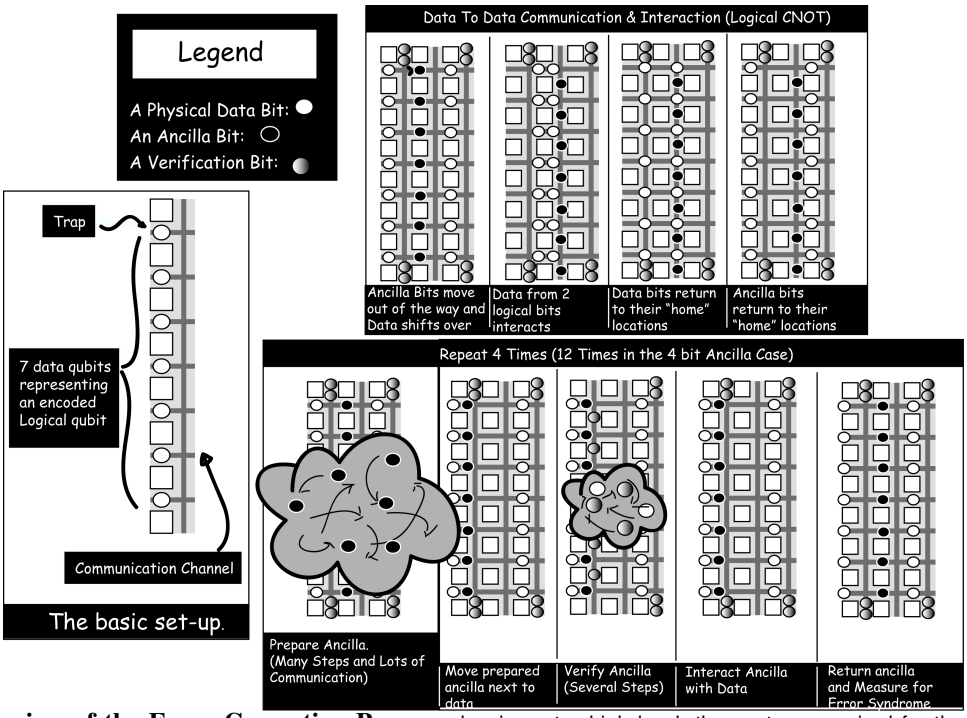

Figure 4. Overview of the Error Correction Process showing, at a high level, those steps required for the 7 bit, adjacent methods.

Method	Num. Ancillas Required	Unary Ops (Pre/Post-Verify)	Binary Ops (Pre/Post-Verify)	Measures (Pre/Post-Verify)	Critical Path Depth
7-bit [19]	**4**	10 (3/7)	31 (17/14)	11 (11/0)	15
7-bit Compact [19]	**4**	10 (1/9)	25 (7/18)	11 (11/0)	19
4-bit [14]	**12**	5 (1/4)	13 (5/8)	7 (7/0)	5
4-bit Nearest Neighbor	**12**	5 (1/4)	16 (8/8)	7 (7/0)	10

Table 2. Control Overhead for Various Steane-Code Ancilla Preparation Methods. Note that since the 4 bit methods must each be performed 3 times as often as the 7 bit methods, they involve a greater total number of control operations.

occur before and after verification since the odds of an error occurring in the ancilla which escapes detection by the verification process is orders of magnitude lower than the odds of all other sources of error. In fact, there is a precedent for ignoring pre-verification sources of error all together when estimating system fidelity [19, 5].

Next, we consider 3 hardware layouts for implementing these algorithms, shown in Figure 5. The first is based on prior work [13] and arranges the ancilla bits in a linear series of traps with an adjacent set of unallocated traps for communication and routing (**Vector**). The second is a set of uninterrupted trapping regions laid out linearly within a single trap (**Linear**). Because this design doesn't feature any means for routing one qubit around another, it's only appropriate for algorithms which have a linear nearest neighbor implementation such as the preparation of the 4-bit ancilla [8, 11]. The third design folds the ancilla and verification bits around a crossbar for maximum interconnect and decreased communication and latency at a cost of larger size (**Folded**). Finally, we consider scheduling each of the algorithms on the hardware in 2 ways: with verification occurring as quickly as possible or with delayed verification which does not occur until the ancilla is in place next to the data qubits it will interact

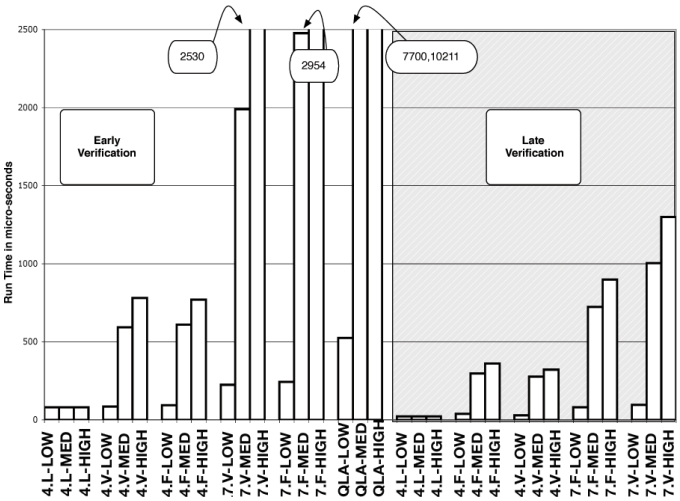

Figure 7. Ancilla Preparation Runtimes in increasing order, separated by early and late verification schemes

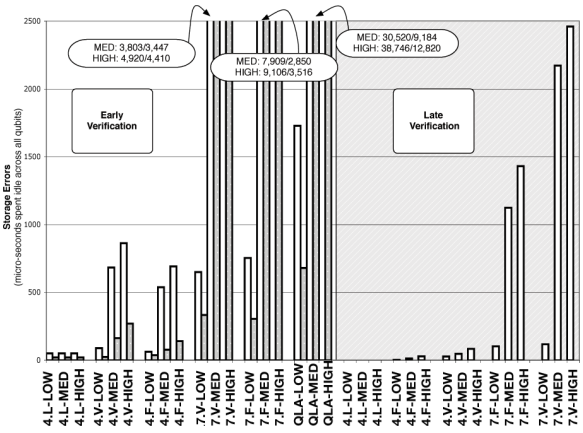

Figure 8. Ancilla Preparation Storage Error PFPs in increasing order of post-verification PFPs, separated by early and late verification methods.

with. In total, this yields 10 legal mappings. In addition, we include results for the 7 bit vector early verification method as implemented in prior work [13]. When implementing the schedule from prior work in our microcode it became clear that the microarchitecture from prior art could be better utilized. So as not to unfairly skew the results against this microarchitecure, we include results for both implementations (labeled 7.V.QLA and 7.V.E respectively).

In Figures 6, 7, 8, and 9 we provide and analyze data related to these ancilla preparation methods[4]. While we highlight some interesting statistics and trends, it is important to note that what really matters isn't the ancilla preparation process itself, but how that process impacts the speed and fidelity of the error correction process as a whole. Furthermore, comparisons across categories are not always valid at this stage: for instance, the 4-bit methods requires 3 times as many ancillas as the 7 bit methods, and the "late verification" methods have not yet incurred the cost of the verification process.

First, we draw the reader's attention to Figures 7 and 8 and those entries relating the the 4-bit linear nearest neighbor methods (labeled 4.L), performed with both early and late verification. Because these methods involve no movement or turns, their performance and fidelity remain invariant across timing scenarios.

Next, we compare the entries labeled 7.V.E and 7.V.QLA. There are three points to be made with this data: First, we can see the importance of a well tuned microcode. Figure 6 shows that our microcode (7.V.E) reduces the communication overhead of this method by more than *half*. Second, because of this decreased communication, coupled with better utilization of resources (resulting in lower numbers of idle bits), the run time of the improved method is less than $\frac{1}{3}$ that of the original (Figure 7), and the number of pre- and post-verification storage PFPs are $\approx \frac{1}{8}$ and $\approx \frac{1}{3}$ that of prior work respectively. Third, the data makes clear that, despite substantial improvements in the microcode, this microarchitecure is by far the least efficient in terms of both runtime and PFPs across timing scenarios.

We now draw the reader's attention to Figure 9 which shows the combined PFPCs for both communication and control (which are invariant across timing scenarios) and storage errors corresponding to the *medium* communication scenario[5]. The most striking result in this chart is the overwhelming number of storage related potential fault points in the 7-bit methods. For these methods the number of storage related PFPs exceeds all other sources of error by at least a factor of 3, and in the method proposed by prior work [13] a factor of more than 10. This means that if storage is within an order of magnitude as error prone as other operations, storage will be a primary source of errors in the system. The 4-bit methods do not feature the same number of storage related PFPs because they execute more quickly and most operations can occur in parallel, leading to fewer idle qubits.

In summary, our ancilla-level exploration has revealed:

- **Linear Nearest Neighbor Methods Perform Well Across A Wide Range of Timing Scenarios.**
- **A Well Tuned Microcode Can Have a Dramatic Impact on Performance and Fidelity.** We were able to write microcode (7.V.E) for the same design used in prior art [13](7.V.QLA) which reduces the run time and post verification PFPs by a factor of 3.
- **Storage Related PFPs Dominate the Ancilla Preparation Process For 7 bit methods,** suggesting that 4 bit methods will ultimately be more efficient when the entire L1 correction process is considered.

5. L1 Exploration

Having seen the wide variance in performance and PFPs in the previous section, we now explore placing ancilla preparation in the wider context of error correction. Of particular interest is understanding how the choice of ancilla preparation and placement affects ancilla-to-data communication, data-to-data communication, the ability to hide ancilla preparation latency, and ultimately system performance and fidelity.

To understand these interactions we explore two basic arrangements of data and ancilla: one where the data and ancilla are placed ADJACENT to one another, and one where the ancilla is OFFSET from the data.[6] Combining these with the 10 ancilla mappings yields a total of 16 legal[7] L1 design-

[4]Our shorthand naming convention lists: [num bits **7 or 4**].[microarchitecture (**V**ector, **L**inear, or **F**olded].[**E**arly or **L**ate Verification]. In L1 we add **A**djacent and **O**ffset. So the 4 bit linear design with late verification would be 4.L.L.

[5]For readability we omit the LOW and HIGH scenarios from the summary since the relative performance of the architectures was similar. The data for these scenarios can be found in Figures 6 and 8.

[6]In the interest of architectural fault tolerance we only consider arranging the L1 data linearly; in this configuration all of the data bits move along one axis in parallel, eliminating the need to share a location in time. Moving along the other axis can be accomplished through the use of a cross-bar such as that shown in Figure 5. Changing axis should be a very rare occurance due to the existence of a number of efficient linear-nearest-neighbor implementations of the algorithms of interest for quantum computation [4, 9] which means that such a cross bar would only be necessary on the edges of a device.

[7]The Linear 4 bit method does not make sense adjacent and the 7 bit folded method results in so much wasted space when offset that it is not worth considering.

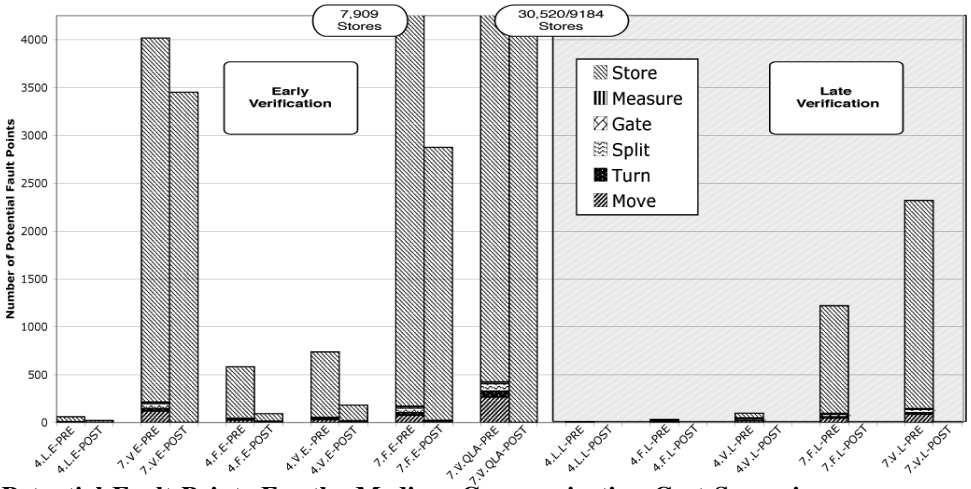

Figure 9. All Potential Fault Points For the Medium Communication Cost Scenario in the same order as previous Figures for readability and separated by early and late verification methods. Note the extremely high incidence of storage related PFPs as compared to all other potential sources of error in the 7-bit methods.

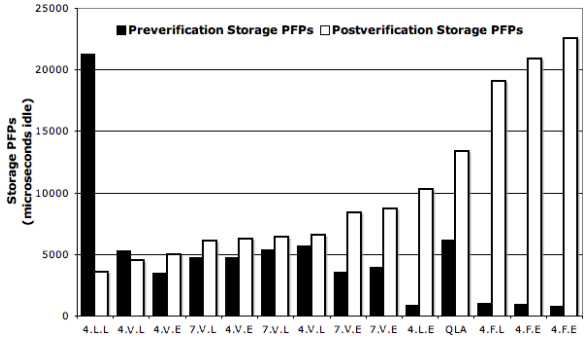

Figure 12. Storage Related PFPs, LOW communication cost scenario

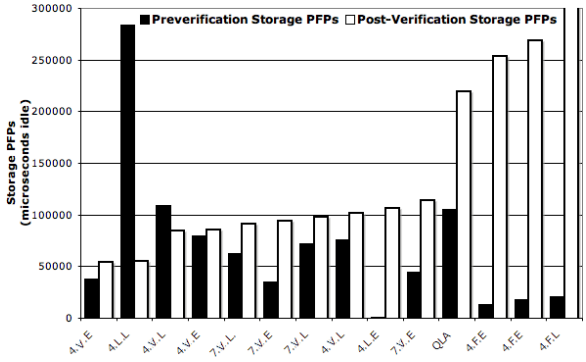

Figure 13. Storage Related PFPs, HIGH communication cost scenario

schedules, on 10 circuits[8] as shown in Figure 10. Later, we will explore the impact of additional ancillas and the resulting area/performance tradeoffs.

We simulated the designs with a binary operation on two logical qubits, followed by the error detection and correction process. This setup forces data-to-data interaction and ensures that the results reflect the impact of the additional data-to-data spacing in the adjacent configurations. The results of our L1 exploration appear in Figures 11, 12, 13, and 14.

Perhaps the most striking aspect of this data is how much the performance and error point gaps between methods shrink in the context of the whole L1 error correction process. The reason is that the overhead of moving the ancilla in and out of place for each error check can be quite high as compared to ancilla preparation, particularly in the simpler 4-bit cases.

Next, we note the wide variance in performance and PFPCs across methods, indicating the importance of a well-tuned design and microcode.

The data also demonstrates the high incidence of storage PFPs across timing scenarios and designs. Even in the LOW communication cost scenario the design with the lowest number of post verification storage PFPs has $\approx$ 3500 post-verify storage PFPs compared to $\approx$ 50 post-verify PFPs from all other sources. This suggests that unless storage is $\approx$ 2 orders of magnitude more reliable than all other operations it will dominate as the cause of error. In contrast, prior art [13, 20] has suggested that storage errors are of secondary importance and can be ignored for the purposes of architectural design.[9] Clearly this is not the case.

Figure 14 shows fidelity estimates broken down by source of error and based on the optimistic projections of prior art [13] found in Table 1. The results are as expected, since moves and turns are the only potential sources of error which are substantially more error prone than stores (10^{-6} vs. 10^{-9} respectively) they are the only sources of error which contend with stores for substantial systemic impact. Figure 14 shows that for the 8 most reliable designs, 60% of failures are due to storage or movement. These are errors that computer

[8] Due to the need for verification bits in the early verification scheme the 4 bit folded circuits differ in size (the other circuits have sufficient unused locations to accommodate them.)

[9] This contention is based on a subtle misunderstanding of experimental results in the field. The QLA and CQLA papers [13, 20] utilize optimistic fidelity projections for their design process. Recent work [10] has demonstrated qubits which maintain their state for an *average* of $\approx 10sec.$. This fact is interpreted incorrectly in [13, 20] to mean that qubits are *guaranteed* not to corrupt for $10sec.$. Given this misunderstanding those papers show the error correction process to take less than 10 sec. and contend that is sufficient to ignore storage related errors.

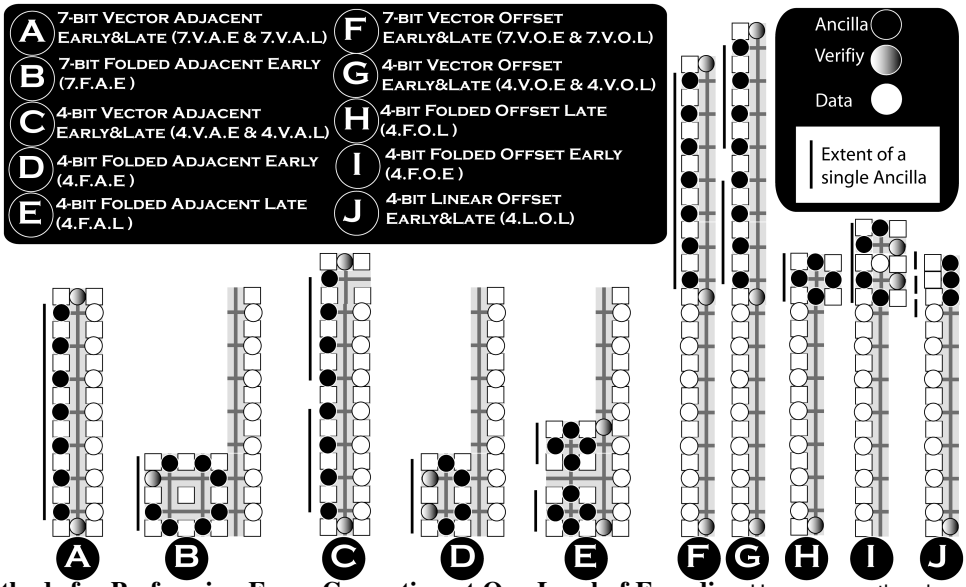

Figure 10. Methods for Performing Error Correction at One Level of Encoding Here we see the placement of various ancilla methods relative to the encoded data. Note that while most designs feature unique microcode mappings on a homogenous"Sea of Traps" microarchitecture, the three uninterrupted traps of (J) necessitate a different micro-architectural design.

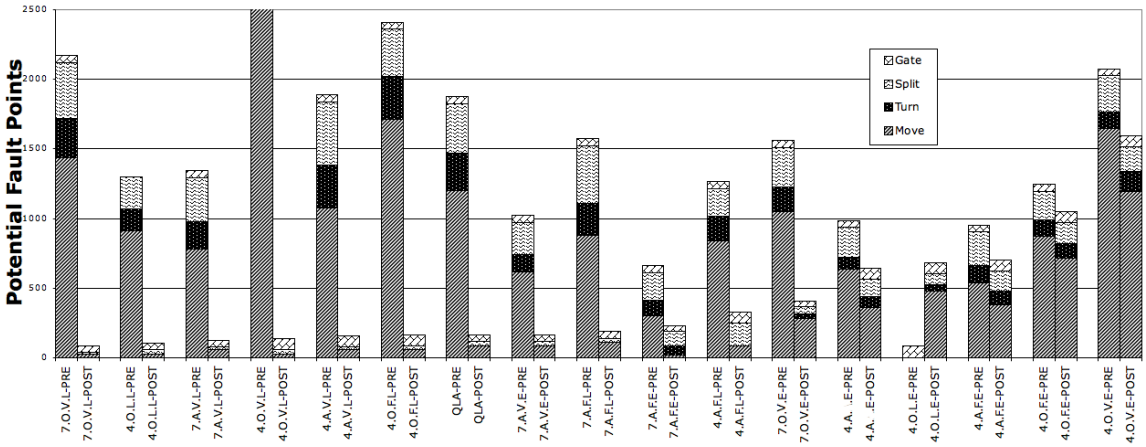

Figure 11. Control and Communication Related PFPs for L1 Methods separated between pre- and post-verification sources and sorted in increasing order of total post verification PFPs.

architects are in the best position to address as we move forward into larger scale quantum systems. Furthermore, those designs which do not feature predominately storage related errors have the worst system fidelity. This is because they have poor locality, forcing the bits to move. In other words, the reason the bits aren't idle is because they are doing something *worse* for system fidelity, they are in transit. Even given different fidelity scenarios, it should be clear that due to the much higher incidence of storage PFPs across designs, minimizing the impact of storage errors will be one of the issues of first order concern for quantum architects in the years ahead.

Finally, it is interesting to compare the data in Figures 12 and 11 with the estimated PFPC given in [19]. In that work an analytical approach is used to estimate the number of control and storage fault points. The model for storage errors is that every bit not involved in an operation incurs a storage PFP. However, since this model does not account for communication costs, this results in a substantial undercount of storage and communication PFPs. In fact, that model would not indicate *any* difference in expected fidelity between any of the 7 bit methods we consider. In actuality, the best 7 bit method has over 5000 storage PFPs in the LOW communication cost scenario. In contrast, the analytical model predicts $\approx$ 600. This is an order of magnitude gap in the projected number of PFPs for storage fault points alone and indicates that if theoreticians desire an accurate estimate of system fidelity they must develop models more closely tied to the hardware architecture.

Finally, these results show the universal value of the 4 bit-offset-linear nearest neighbor method. It is the smallest, fastest, and most reliable design across all the timing and fidelity scenarios we consider. Furthermore, because no other design has fewer total PFPs and the 4.O.L.L method has the lowest incidence of communication related PFPs, it is virtually guaranteed that this design will be the most robust for any reasonable set of technology parameters.

In summary, our L1 exploration shows:

- **There is a Wide Variance in Performance and PFPCs Across Methods.**
- **At L1 the incidence of Storage Related Errors is Highly Pronounced.**
- **At Times, Architecture Matters More than Theoretical Efficiency.** Even with highly tuned designs and optimistic projections for component fidelity, when systems fail it is predominately due to faults relating to communication or memory.
- **An Architecture and Microcode Based on An Offset 4 bit Linear Nearest Neighbor Scheme is the Best Choice Across a Wide Range of Technology Scenarios.** This design occupies less than $\frac{1}{2}$ the space of the design proposed in prior work and operates with an expected fidelity that is over an order of magnitude more reliable given prior art's technology assumptions.

5.1. Varied Ancilla Resources

To get a sense of how these architectures respond to and utilize additional ancillary resources we now examine two test cases: The 4-bit offset linear nearest neightbor design

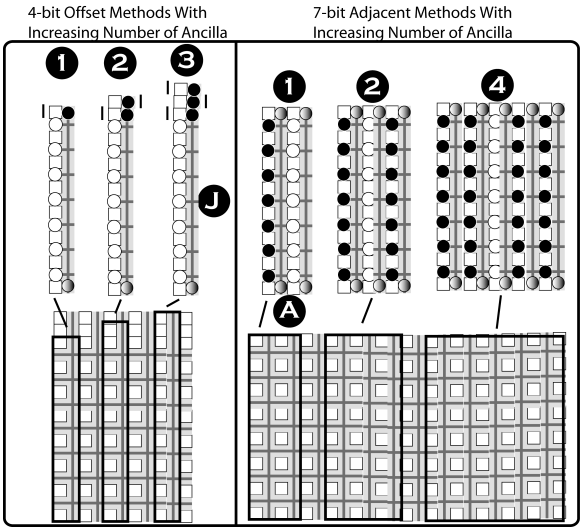

Figure 15. 4 bit Linear Offset and 7 bit Array Adjacent Microarchitectures With Increasing Numbers of Ancillary Resources and Communication Channels

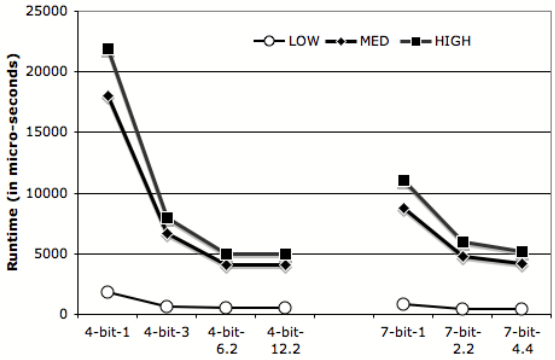

Figure 16. L1 Error Correction Runtimes with increasing number of physical Ancillas (shorter is better)

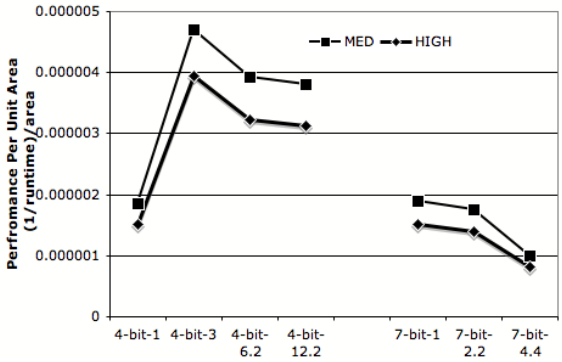

Figure 17. L1 Error Correction Performance/Area. With increasing number of physical ancillas.(taller is better, LOW has been omitted for readibility, Performance = $\frac{1}{runtime}$)

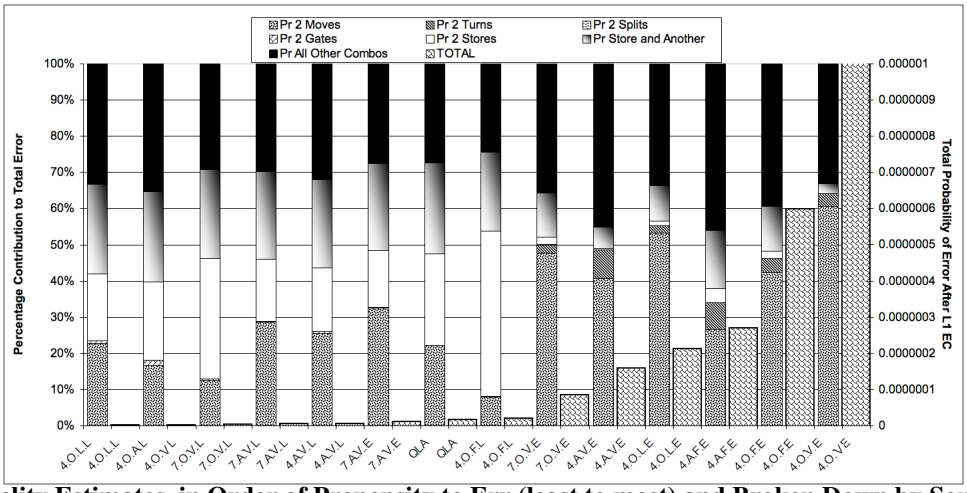

Figure 14. Fidelity Estimates, in Order of Propensity to Err (least to most) and Broken Down by Source of Error. Note that the late verification methods almost universally outperform early methods in terms of fidelity, and that communication related PFPs (such as move) and storage related PFPs dominate sources of error.

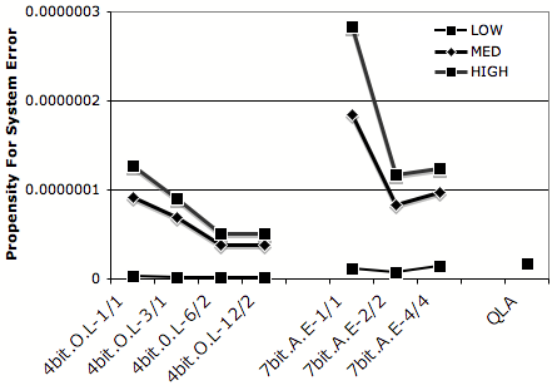

Figure 18. Expected L1 Error Correction Fail Rates indicating how likely it is that the encoded data is still in error *after* the QEC process has completed (shorter is better)

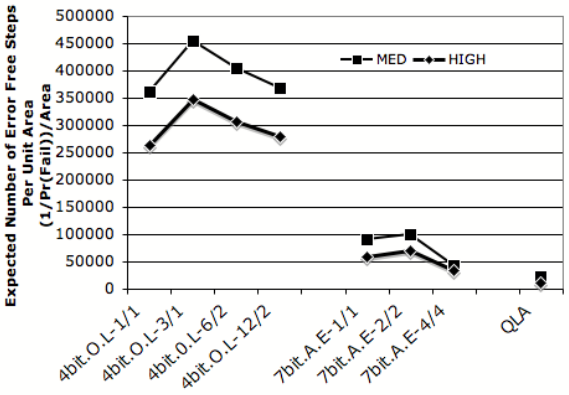

Figure 19. Expected Number of Error Free Computational Steps on Encoded Data. Calcluated as $\frac{1}{Pr_{fail}}/AREA$ (taller is better).

and the 7-bit vector adjacent design. For each of these we increase the number of ancillas and the number of traps devoted to communication (**communication channels**). Figure 15 shows the different architectures we consider. For 4-bit offset we utilize 1 and 3 ancillas on a single communication channel and then 6 ancilla divided between two communication channels. In the 7-bit case we explore 1, 2 and 4 ancillas and communication channels. The results of this exploration appear in Figures 16, 17, 18, and 19.

The data indicates that the 4-bit method makes good use of additional ancilla on a single channel. Figures 17 and 19 show a maximal value for the 4 bit method with 3 ancilla on a single channel. In contrast, the 7 bit method has strictly decreasing performance per area with additional ancilla (Figure 17) and only slightly better fidelity over area with the addition of a second ancilla (Figure 19).

Next, we draw the reader's attention to Figure 18. Here we can see the 4 bit method strictly improving in fidelity as we increase ancilla (though not all ancilla can be well utilized so there are diminishing returns). In contrast the 7-bit method actually suffers an *increase* in propensity to err with the addition of 4 ancilla. The reason for this difference is that the 4 bit method is an offset type, so increasing the number of ancilla does not increase data to data communication costs. In contrast, the 7-bit method is an adjacent type so each additional ancilla pushes logical qubits further apart.

Also of note is that the 7 bit method increases in performance while decreasing in fidelity with 4 ancilla. This shows that the relationship between runtime and fidelity is not one to one.

We note that utilizing additional ancilla in parallel will result in increased reliability only if the impact of storage errors is substantial. With more ancillas the same number of control operations must be carried out, and in fact, the additional structures can increase the distance between encoded data words or ancillas and encoded data thus *increasing* the number of communication related PFPs. The only PFPs that additional ancillas can address are storage related because the additional ancillas decrease error correction time and thus decrease the time that bits sit idle. If storage errors could be ignored, then any additional ancillas would result in a net *decrease* in system reliability.

Figure 18 shows that, based on the optimistic fidelity numbers from prior work [13], the 4-bit method outperforms the 7-bit method and is more reliable, even as the number of ancillas are increased. Furthermore, the small size of the 4 bit ancillas means that performance and reliability per unit area are higher across the board for the 4 bit methods. Since area considerations will be a major issue when scaling ion traps up to large systems [14], we believe this is an important metric to consider.

In general our exploration of the impact of additional ancillas has shown:

- **The 4 bit Offset Method makes Good Use of Additional Ancilla on a Single Channel**
- **The 7 bit Adjacent Method Suffers a Decrease in System Fidelity with 4 Ancilla** This is due to the additional overhead of data to data communication that occurs when L1 data is pushed further apart by each additional ancilla.
- **Increased Fidelity from Increasing Ancilla is Due Exclusively to the Impact of Storage PFPs** In the absence of storage errors additional ancillas may decrease runtime but they will not improve fidelity (and may worsen it).
- **The 4 bit method is Superior to the 7-bit Method given Comparable Numbers of Ancilla** and in much less area.

6. Future Work

Looking ahead, there are several avenues of future work that we are starting to consider:

- **Grouping Pre-Error Correction Steps:** Currently, all the overhead of the error correction process, 10's of thousands of operations, is used to error correct a single operation. The actual error of the corrected operation then is minimal as compared to all other PFPs. Therefore, we are exploring the impact of grouping together more operations before performing the error correction process.
- **Automatic Construction of the Microcode:** While we contend that a hand scheduled microcode is most likely to achieve near optimal results, we are developing automated tools for scheduling so as to provide a means for faster first pass evaluations of fault tolerant schemes and micro-architectures.
- **More Fault Tolerant Schemes:** While the Steane code has been the most widely adopted code in the architecture community, many other error correcting codes exist. We would like to combine some sort of automated microcode process with an exploration of more error correcting codes.
- **A Similar Design Space Exploration For Two Levels of Error Correction (L2):** While the results of this work will have immediate impact on any L2 implementation (since L2 must necessarily be composed of L1 error correction schemes) we believe there is value in performing a similar DSE for the L2 design space. Figure 20 shows one of the axes of exploration that is unique to L2. In this diagram we see two means for error correcting the error correction process. On the left is the most straight forward design which is similar to that proposed in prior work [13, 20]. In this design, the structure of the error correction process at L1 is directly mirrored at L2. We are particularly excited about the prospects for our novel design on the right. We call this design "striped" recursion because the data is arranged in stripes with the n^{th} bit of all the L1 logical bits aligned in a single column. There are several advantages to this design. First, by aligning data in this way no two bits from the same L1 logical bit need ever utilize the same hardware resource when performing the L2 error correction process. This is not true of the design on the left since the construction of the L2 ancilla requires interacting encoded ancilla bits which must move vertically in a single file column, thus violating architectural fault tolerance for the encoded ancilla bits. We have generated some preliminary performance numbers for these designs with increasing number of encoded ancilla and find that in addition to addressing architectural fault tolerance, the striped design is 3 times faster than the array method when both employ the improved L1 designs from this study.

7. Conclusion

This paper has explored how to design and microcode ion-trap quantum computers. We have examined a wide range of designs for ancilla preparation and L1 error correction. Our data reveals several important lessons for physicists, theoreticians and architects.

For physicists, our data demonstrates the huge importance of reliable quantum storage and communication. Ion trap technology has made tremendous strides in regards to storage, with average stable qubit lifetimes up to 10 seconds. However, moving and turning ions remain the most error prone operations by far, transforming the relativly few instances of these operations into major sources of systemic error.

For theoreticians, our results are more ominous. When algorithms are mapped onto actual microarchitectures, the constraints of communication and storage can make algorithmic techniques which look good on "paper" inefficient in practice. In addition, we find that even when using highly optimistic projections for communication timings, theoretical models such as [19] undercount the number of PFPs by nearly an order of magnitude. What we can conclude from this is that quantum theoreticians working in the error correction space need a more technology-driven model with which to design algorithms.

For architects, our results illustrate the value of exploring this space now, as opposed to when device technology is completely stable. The reasons for this are twofold. First, we have demonstrated the huge impact architects can have on system fidelity and performance. This implies that we can accelerate the arrival of large scale quantum systems. Second, since we have found a single architecture and microcode which outperforms all others across a range of technology parameters, we can have confidence that as the technology space evolves, the architectural contributions of this work will hold.

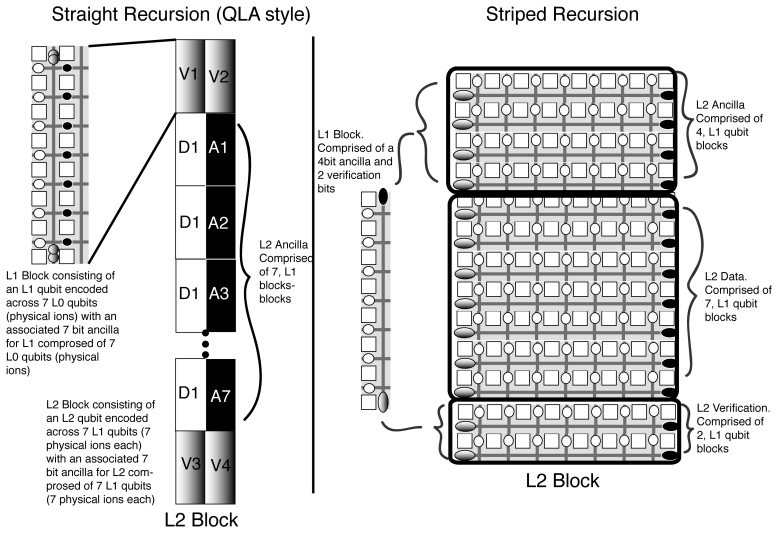

Figure 20. Methods for Constructing an L2 Error Correction Block out of L1 Blocks

Acknowledgements

We would like to thank David Bacon for the invaluable information and expertise he provided as we worked on this paper. This work is made possible with generous support from NSF CCF-0523359 and CCF-0621621 and the Sloan Foundation.

References

[1] D. Bacon. Operator quantum error correcting subsystems for self-correcting quantum memories, 2005.

[2] S. Balensiefer, L. Kreger-Stickles, and M. Oskin. An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures. In *Proc. International Symposium on Computer Architecture (ISCA 2005)*, New York, 2005. ACM Press.

[3] C. M. D. Kielpinski and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. *nature*, 417:709–711, June 2002.

[4] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg. Implementation of shor's algorithm on a linear nearest neighbour qubit array. *QUANT.INFO.COMPUT.*, 4:237, 2004.

[5] D. Gottesman. Fault-tolerant quantum computation with local gates. *Journal of Modern Optics*, 47:333, 2000.

[6] W. K. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, C. Monroe, and J. Rabchuk. T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation. *Applied Physics Letters*, 88:4101–+, Jan. 2006.

[7] L. C. L. Hollenberg, A. D. Greentree, A. G. Fowler, and C. J. Wellard. Two-dimensional architectures for donor-based quantum computing. pages 045311–+, 2006.

[8] D. Kielpinski, A. Ben-Kish, J. Britton, V. Meyer, M. A. Rowe, C. A. Sackett, W. M. Itano, C. Monroe, and D. J. Wineland. Recent results in trapped-ion quantum computing, 2001.

[9] S. A. Kutin. Shor's algorithm on a nearest-neighbor machine, 2006.

[10] C. Langer, R. Ozeri, J. D. Jost, J. Chiaverini, B. DeMarco, A. Ben-Kish, R. B. Blakestad, J. Britton, D. B. Hume, W. M. Itano, D. Leibfried, R. Reichle, T. Rosenband, T. Schaetz, P. O. Schmidt, and D. J. Wineland. Long-lived qubit memory using atomic ions. *Physical Review Letters*, 95(6):060502, 2005.

[11] D. Leibfried, E. Knill, S. Seidelin, J. Britton, R. B. Blakestad, J. Chiaverini, D. B. Hume, W. M. Itano, J. D. Jost, C. Langer, R. Ozeri, R. Reichle, and D. J. Wineland. Creation of a six-atom 'Schrödinger cat' state. *nature*, 438:639–642, Dec. 2005.

[12] S. Lloyd. Quantum mechanical computers. *Scientific American*, october 1995.

[13] 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, 2005.

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

[15] M. Oskin, F. T. Chong, I. Chuang, , and J. Kubiatowicz. Building quantum wires, the long and short of it. In *Proc. International Symposium on Computer Architecture (ISCA 2003)*. ACM Press, 2003.

[16] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J.SCI.STATIST.COMPUT.*, 26:1484, 1997.

[17] A. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77, 1996.

[18] A. Steane. Multiple particle interference and quantum error correction. *PROC.ROY.SOC.LOND.A*, 452:2551, 1996.

[19] A. Steane. Space, time, parallelism and noise requirements for reliable quantum computing. *ArXiv Quantum Physics e-prints*, Aug. 1997.

[20] D. D. Thaker, T. S. Metodi, A. W. Cross, I. L. Chuang, and F. T. Chong. Quantum memory hierarchies: Efficient designs to match available parallelism in quantum computing. *SIGARCH Comput. Archit. News*, 34(2):378–390, 2006.

[21] D. Wineland and T. Heinrich. Ion trap approaches to quantum information processing and quantum computation. 2004.

[22] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–+, Oct. 1982.