A Design Overview for a Simulation Infrastructure for Exploring Quantum Architectures

Dean Copsey[‡], Mark Oskin[†], Andrew Cross[⋄], Tzvetan Metodiev[‡], Frederic T. Chong[‡], Isaac Chuang[⋄], and John Kubiatowicz[∘]
[‡] University of California at Davis, [†] University of Washington,
[⋄] Massachusetts Institute of Technology, [∘] University of California, Berkeley

ABSTRACT

Quantum architectural research has been hindered by the lack of scalable simulation tools. This deficiency is understandable, since the amount of information required to accurately simulate a quantum system grows exponentially. However, it is possible to obtain accurate statistical information about a quantum architecture, without fully simulating a quantum algorithm. Some interesting questions for architects are: how fast does it run? how many operations are performed? how reliable is the result? To answer these questions, a quantum simulator doesn't require complete state information. This paper is a proposal for a scalable quantum architectural simulator, based on a hierarchy of modeling abstractions, that works across a few different proposed silicon-based quantum device technologies.

1. INTRODUCTION

There are many different packages that simulate quantum computations [8], by simulating quantum state transitions. While quite useful for verifying algorithms on a small number of quantum bits, the overhead of storing and manipulating quantum state makes such languages impractical for modeling systems of more than a dozen entangled qubits. This makes the straightforward approach impractical; we cannot utilize the same methods of classical architecture research in the quantum domain.

Fortunately, for architectural studies we can make an important observation: simulating a quantum state requires potentially exponential resources, but simulating the *reliability* of a quantum state requires only polynomial overhead. Hence, if we divorce of ourselves of the actual state (i.e. the classical analog of the simulated memory and register values), and instead focus only on estimating the reliability of this state, then a quantitative investigation of quantum architectures becomes practical.

In this paper we outline the necessary abstractions for a quantum architectural simulator, and how these models can be linked together for polynomial time evaluation of architecture proposals. Classically, architects are generally interested in the following question: how fast is a given architecture on a given application. The ultimate goal is the optimization of the cost in terms of area and/or power. In the quantum world, the view is similar except that one tries to optimize for area (least number of quantum bits), while still maintaining a necessary level of reliability.

The next section sets the stage with a discussion of classical architectural simulation. Section 3 gives an overview of error in quantum systems, and a rationale for being able to estimate quantum error in polynomial time and space. Section 4 describes an approach to stateless simulation of quantum devices, circuits, architecture and software *in polynomial time and space*. Section 5 describes the actual steps necessary to perform simulation from software down to the circuit and gate levels. Conclusions are given in Section 6, and for those readers unfamiliar with quantum computation, or this paper's notation, Section 7 gives a brief introduction and overview.

2. CLASSICAL SIMULATION

Research into conventional systems is a well established field. Over time, it has specialized itself into various sub-disciplines such as devices, circuits, architectures and software support systems, with the latter split much finer still. The field of quantum computation is still too young have reached a similar degree of specialization. Hence, we need to consider a much broader form of "simulation", than is typical in the classical world. To put the remainder of this paper in perspective we outline the various tools and research methods used in each of these classical specialties prior to discussing their quantum analog in subsequent sections.

Physical layer: The bottom-most layer of classical technology is the actual device physics that controls the interactions of electrons in silicon. Here basic *n*-body simulations are used to determine device operation. Just a small step up from these are gates, whereby basic abstractions such as transistors, are used to form complete logic gates.

SPICE simulation tools are used to model the input-output characteristics to answers questions such as delay and drive capability.

Circuits: Just above the physical layer in complexity is the circuit layer. Here, standard building blocks such as adders, memories, content-addressable memories and other components are put together to form working circuits. A typical approach to work at this layer is to implement a design in a hardware description language (such as Verilog or VHDL). The measurements undertaken at this layer usually involve executing precisely designed test vectors through simulation tools to test functionality and estimate performance (cycle time, power). The design can also be compiled by synthesis tools to produce an ASIC. Some or all components can be manually optimized to form a full-custom device.

Architecture: Above the circuit implementation stands the architectural level. Here, larger building blocks, such as execution units, issue-windows, and instruction fetch stages are modeled using custom designed software simulations. Typically these simulations are far removed from circuit implementations and performance is estimated by functionally simulating an application and measuring instructions per cycle (IPC).

Software support: At the upper most layer of classical research stands the software support system. This usually includes a compiler, the performance of which is typically gauged on a real system. It may also include a dynamic optimizer, whose quality is determined either on a real system or on an architectural level simulator. As with the architectural layer, the performance is measured in either wall-clock time or IPC depending upon the specifics of the research.

At the moment, all four of these layers are of interest to quantum architects. Prior to discussing their quantum analogs, we first describe the intuition behind polynomial time simulation of reliability, which is one of the fundamental criteria of quantum architectures.

3. RELIABILITY

A quantum operator (gate) G with input ρ and output ρ' *fails* if $D(G(\rho),\rho') \neq 0$ for metric D such as the trace distance. The action of a quantum operator G that fails with probability p can be represented as

$$\rho' = (1-p)G(\rho) + pE(\rho)$$

where $E(\rho)$ in an arbitrary quantum error operator, and p is less than a particular bound [3]. In general, if several such gates are composed, the error bound grows linearly in p, to the first order.

Considering a coarser set of gates, h_i, each composed of gates from $\{g_i\}$, gives a set of corresponding probabilities, q_i, that can be bounded by applying the quantum circuit model, specifically, by simulating the action of a circuit C in the presence of physical noise processes such as control inaccuracy and decoherence. The true probability of failure is less than the composition of the q_i.

3.1 Failure Bounds

An important problem in engineering reliable systems built from faulty components is the accurate determination of bounds on failure probabilities. For classical circuits, excellent bounds on the failure probability of a system can be established by bounding the failure probability of its components. However, for quantum circuits, that is not the case. This section gives some simple examples of this important and non-intuitive fact.

3.1.1 Error propagation in classical systems

Let us begin by considering the classical case. Suppose we have a noisy NOT gate, G, which fails with probability p. It then follows that prob $\Big[G(0)=1\Big] \geq 1-p$, that is, acting on an input value of 0, the output is the correct value of 1 with probability at least $1-p$. Furthermore, for a circuit $C = G \circ G$ of two such noisy gates cascaded together, prob $\Big[C(0)=0\Big] \geq (1-p)^2$, since the composite circuit will give the correct output if both gates work successfully.

This elementary idea, that the probability of success of the whole can be bounded below by the probability of success of its components, is crucial to the study of complex systems and architectures, and how such systems can be made reliable. Of course, there are a variety of errors that a classical gate can have: it may be a completely random error, or its output could be stuck at one value, for example. Generally, however, the likelihood that errors cancel each other out by chance is very small, particularly in large circuits, and never for all possible inputs (except in trivial instances). Thus, it is a good approximation that the system will operate properly only when all of its components do.

3.1.2 Quantum gates with systematic errors

The failure probability of a quantum circuit can similarly be bounded below by the probability of failure of its component quantum gates. However, in some important situations, this bound is far from being tight, meaning that it is *not* a good assumption that the system will operate properly *only* when all of its components do so perfectly. That is because there are many more ways for a quantum gate to fail than for a classical gate to fail.

For example, consider a noisy quantum NOT gate, $U_\varepsilon = R_x(\pi + 2\varepsilon)$, written using standard notation [6], where ε is a fixed, small quantity. This gate suffers from a systematic error, an over-rotation of the qubit, known to be a common error in actual physical implementations, where the area of a pulse determines the qubit rotation angle; when such pulses are too long, the qubit is over-rotated. For an input state $|0\rangle$, the output becomes $U_\varepsilon|0\rangle \approx |1\rangle - \varepsilon|0\rangle$, to first order in ε. When measured, this output state gives the wrong answer, 0, with probability $|\varepsilon|^2$.

When these faulty gates are cascaded, errors of this form can be made to partially cancel, even for all possible inputs, with specially constructed circuits. For example, consider the four gate quantum circuit $V_\varepsilon = R_x(\pi/2+\varepsilon)R_{-y}(\pi+2\varepsilon)R_x(\pi/2+\varepsilon)R_{-z}(\pi+2\varepsilon)$. This is a sequence of four rotations that accomplish the same action as U when $\varepsilon = 0$. However, for small ε, $V|0\rangle \approx |1\rangle + \varepsilon^2|0\rangle$. When measured, this output state gives the wrong answer, 0, with probability $|\varepsilon|^4$, which is much less than before! In fact as long as ε is sufficiently small, the failure probability of this circuit is *less* than the failure probability of any of the individual gates, for *all* inputs.

Let us analyze this example in more detail by introducing the concept of fidelity. The *entanglement fidelity* of a single-qubit unitary quantum gate acting on an entangled two-qubit pure state $|\psi\rangle_{AB}$ is

$$F(|\psi\rangle_{AB}, U) = F(|\psi\rangle_{AB}, (U\otimes I)|\psi\rangle_{AB})^2. \qquad (1)$$

The entanglement fidelity is a lower bound on the square of the static fidelity of U acting on a single-qubit state [6], which can be interpreted as the probability that U succeeds given $|\psi\rangle_A$. By minimizing the entanglement fidelity over all two-qubit entangled pure states, one can define a *gate fidelity*

$$F(U_1,U_2) \equiv \min_{|\psi\rangle_{AB}} F(|\psi\rangle_{AB}, U_1^\dagger U_2) \qquad (2)$$

that can be written [1]

$$F(U_1,U_2) = \frac{|\mathrm{tr}\,(U_1^\dagger U_2)|^2}{4}. \qquad (3)$$

In this sense, $F(U_1,U_2)$ is a lower bound on the probability that the noisy gate U_1 succeeds given the ideal gate U_2, or, as we will prefer, $1-F(U_1,U_2)$ is an upper bound on the probability of failure.

The failure probability of U_ε acting on an arbitrary single-qubit state $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$ is $p_{\mathrm{fail}}(U_\varepsilon, |\psi\rangle) = 1 - \langle\psi|U_0^\dagger U_\varepsilon|\psi\rangle$. In the specific case where $U_\varepsilon = R_x(\pi+2\varepsilon)$, $p_{\mathrm{fail}} = 1-\cos^2\varepsilon - \sin^2\varepsilon\sin^2\theta\cos^2\phi$. The gate fidelity of the noisy U_ε gate, $F(U_\varepsilon,U_0) = \cos^2\varepsilon$, gives a clear upper bound $p_{\mathrm{fail}} \le 1-\cos^2\varepsilon$ which is met by the states $|0\rangle$, $|1\rangle$, and states that pass through $|0\rangle \pm i|1\rangle$ as θ varies (i.e. U_ε gives the worst performance for these states).

Consider now the error bound on the V gate given by the gate fidelity,

$$p_{\mathrm{fail}} \le 1 - F(V_\varepsilon, V_0) = 1-\cos^2\varepsilon|\cos\varepsilon + \sin^2\varepsilon|^2. \qquad (4)$$

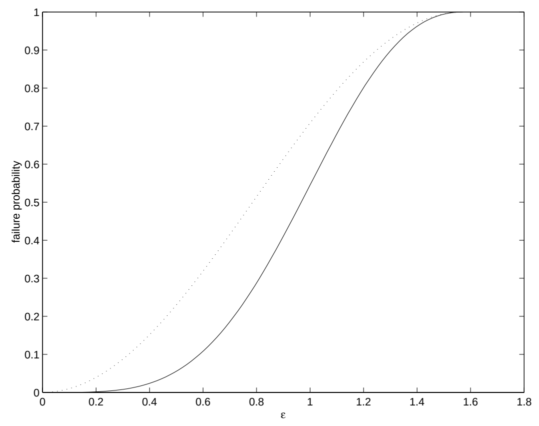

Figure 1: Upper bounds on gate failure probability implied by fidelity. The solid line is $1-F(V_\varepsilon,-iX)$ and the dashed line is $1-F(U_\varepsilon,-iX)$. These failure probabilities are very close to those obtained when U_ε and V_ε act on $|0\rangle$ (i.e. $\sin^2\varepsilon$ and $\sin^4\varepsilon$ respectively).

Surprisingly, this error bound is better than the bound on U, even though two of the four gates in V implement U. In fact, $F(V_\varepsilon,-iX) \ge F(U_\varepsilon,-iX)$ for $0<\varepsilon<\pi/2$, noting that $V_0 = U_0 = -iX$ (see Figure 1). States very close to $|0\rangle$ or $|1\rangle$ fail with probability close to the upper bound given by the gate fidelity.

Calculating the failure probability of V to full order, $p_{\mathrm{fail}}(V) \le p_{\mathrm{fail}}(U)$ for all $|\psi\rangle$ some distance $\delta(\varepsilon)$ from the X eigenstates $|0\rangle \pm |1\rangle$, where $\delta(\varepsilon)$ is on the order of ε.

This example shows how gates with *systematic* errors can be used to construct circuits which perform with lower failure probability than its components would imply, if we had used classical methods for bounding failure probability. This technique is an art known as *composite pulses* in the field of NMR [5], and is one of the most important tools in practical quantum computation today.

3.1.3 Quantum gates with random errors

The failure probability of quantum circuits also depends highly on the specific inputs fed to it; often, even for gates which fail *randomly*, there are wide classes of inputs which may or may not be sensitive to the error induced by the gate.

For example, consider the single qubit gate $U = R_x(2\eta)$, where η is a normally distributed random variable, with zero mean and variance σ. This faulty gate produces significant error when fed a $|0\rangle$ or $|1\rangle$ input, since $U|0\rangle = \cos(\eta)|0\rangle + i\sin(\eta)|1\rangle$ and $U|1\rangle = \cos(\eta)|1\rangle + i\sin(\eta)|0\rangle$; ideally, the output should be the same as the input (because the mean value of the noise is zero), but the wrong answer is measured at the output with probability

$$\frac{1}{\sqrt{2\pi\sigma}}\int_{-\infty}^{\infty}\sin^2(\eta)e^{-\eta^2/2\sigma}d\eta = 1-e^{-2\sigma}. \qquad (5)$$

However, now consider the circuit $V = HR_x(2\eta)H$, where H is the Hadamard gate. This circuit works *flawlessly* on inputs $|0\rangle$ and $|1\rangle$, since $V|0\rangle = |0\rangle$ and $V|1\rangle = |1\rangle$. In fact, any single qubit gate $R_{\hat{n}}(2\eta)$ has the same two eigenstates as $\hat{n}\cdot\vec{\sigma}$ and will work flawlessly on these states. This example makes the point that quantum circuits with random errors may also have significantly different failure probabilities than the naive bound given by combining the failure probability of its component gates.

3.2 Polynomial Time Simulation

Clearly, it is desirable to achieve a tight lower bound on reliability. However, to do so requires an accurate description of the complete quantum state. Since the amount of state information grows exponentially with system size, this limits the size of the system being simulated.

One simplifying assumption is that errors accrued by the system are, to first order, independent. This doesn't help much if the system is viewed as an amorphous group of qubits, but it becomes very helpful when circuit-level structure is imposed on the qubits. In particular, certain structures are deemed essential for scalable quantum computation. These include blocks for error-correction codes (linear codes, decoherence free subspaces, etc.), circuits for entropy removal, and circuits to *purify* known states with a quantifiable amount of error. All of these structures have the purpose of removing error from a subset of qubits in the system. Furthermore, these structures are generally small enough to simulate precisely. The precise simulation results in a known aggregate behavior that can be modeled much more efficiently.

Aggregating substructures that have been precisely modeled can be done in polynomial time, given the assumption that errors between substructures are not correlated. If error accrues in the system more rapidly than it is removed, however, this can be detected during simulation, resulting in a simulation failure.

4. QUANTUM SIMULATORS

In Section 2 we described how research into classical systems has already specialized to the degree that dedicated tools and even research communities focus on individual abstraction layers from device technology through software. Research into quantum systems has not undergone such a specialization yet and thus quantum architecture research needs to work across the conventional layers of abstraction. In this section we describe the approach to quantum devices, circuits, architecture and software simulation that we are undertaking.

Operator	Time	Error
ROTX	1.0e-7	1.0e-8
ROTZ	1.0e-7	1.0e-8
SWAP	5.7e-7	2.1e-7

Table 1: Operations summary for the Skinner-Kane technology.

4.1 Physical layer

The concept of a qubit is naturally at the core of our physical device simulation. We abstract from the actual device technology, instead characterizing qubits by the following properties:

Static decoherence: quantum systems are subject to noise from the environment, leading to decoherence. The amount of decoherence, or introduced error, is a function of time.

Realizable operators: each quantum device technology seems to implement a different set of underlying quantum operations. Thus, at the core of a qubit model is what operations are supported, how long they require in real time execute, how much error is induced while performing them, and how the canonical set of universal quantum operations are synthesized from them. Table 1 depicts these attributes for the Skinner-Kane [11] technology.

Measurement operators: measurement operators are applied to the system for two reasons. The first is to read out the final state of the system. The second, and more common, reason is for error mitigation. Some technologies allow for measurement on any qubit, while others have special devices for measurement. Measurement almost always has the side effect of reducing the overall number of states, and so cannot be completely modeled at the physical level. Hence, the description of a measurement operator contains a "call" to an error-mitigation function, which is implemented by the circuit-layer.

Implementation dependent parameters: some implementations may parameters that are not general. An example is the speed of ballistic transport–physically moving qubits, rather than states. Not all qubits are mobile.

Combined these properties are used to control the simulation of a quantum architecture. By exploiting the ideas from the previous section, such a device simulator returns the reliability of the operation as well as the total amount of time and resources required.

4.2 Circuit layer

To manage the complexity of classical systems, designers form basic abstractions for collections of electrical components. These abstractions include gates, adders, memories, etc. These components are then composed into circuits using high level description languages or schematic capture. On occasion, full-custom layout is used. We expect the same approach to abstraction will be used for building large quantum circuits.

Some basic building blocks (quantum cad cells) we have already discussed in previous work [9] and include teleportation, entropy-exchange, and purification units. Obvious other ones include error correction blocks and analogs of classical components (adders, memories, etc). As with the quantum cellular automata (QCA) work [7], the wire-level pipelining of quantum technologies will likely lead to some unique circuit constructions and design rules.

As with the physical layer simulation, components of the circuit layer taken on a set of properties. These properties are very much like those a component would have within a standard cell library.

Interconnectivity: Certain proposed device technologies, such as ion-traps [2, 4] are more mobile than others (such as [11]). This means than instead of having to always use a swapping or teleportation channel for communication [9], the qubit itself can be ballistically transported across a physical space. This interconnection ability changes the reliability of transport, just as with conventional communication techniques.

Spatial extent: As with classical standard cells, the spatial dimensions are an important property of a component. The distance between adjacent quantum bits drives a number of architectural parameters. For instance, an extremely fine spacing implies a sparser classical control, leading to potential layout constraints [9], or even a SIMD style approach to interconnect [13].

Input / output characteristics: Similar to classical components, what inputs and outputs are available is part of the description of a quantum cell. While quantum operations are reversible by nature, it often makes sense to have non-reversible cells, such as for entropy exchange or measurement. Hence, the number of inputs may not match the number of outputs on a quantum component, even though it may perform a reversible operation (e.g., a reliable *cnot* gate may have two inputs and two outputs, and a third input as a source of cold zero qubits from an entropy exchange cell).

Reliability function: Unlike the low-level error probability calculation of the physical layer simulation, at the component level we abstract errors from state, and provide an aggregate error behavior model.

4.3 Architecture layer

The architectural level simulation of a quantum computer is currently an extension of the circuit level simulator. In the classical case, one simulates the architecture of a computer with the appropriate Verilog or VHDL circuit description. However, quantum hardware description languages [12] are only now being developed. In the combined simulation approach required for quantum systems at the moment, it makes little sense to make a distinction between an architecture layer simulator and simulation of ever more complex quantum CAD cells.

In the future, however, as particular quantum device technologies prove more promising than others and we can abstract ever more away from them, then a more architecturesque simulator will be called for. As is the case now, such a simulator is focused on obtaining two results: efficiency of execution on a given algorithm, and reliability.

4.4 Software support

The final layer of abstraction for a quantum simulator is software support, or the quantum instruction set architecture. Our work here is only just beginning, but two things seem evident:

Parallel languages will almost certainly be used. Quantum devices are so unreliable, and it will be extremely costly to make large ones, that the typical cost/benefit of making the programmers task more difficult is quite different than on a classical system. It seems likely that anything that reduces the number of qubits required, or speeds up the calculation by exploiting parallelism will be worth it, even with a great deal of programmer intervention.

Dynamic compilation is a certainty. Currently, quantum algorithms are expressed at the bit-level. Similar to classical bit-level calculations, a significant amount of circuit specialization opportunities are available. For example, it is well known that specialized multipliers are far smaller and faster than fully general ones. At the heart of Shor's algorithm for factoring [10] lies a multiply that is ripe for such specialization.

With these speculations about the software support in mind, it seems likely that any quantum architecture simulator will also tightly couple the software support system. Indeed, we imagine that in the first revision of our work this software support will be done "off line" but with full knowledge of the architecture and software being simulated.

5. SIMULATOR OPERATION

Execution of the quantum architecture simulator proceeds as follows. First a pre-execution phase is performed:

1. Dynamically compile the application by specializing the operations as much as possible to remove

redundant or useless primitives.

2. Instantiate operations by realizing them into the underlying physical gates provided by the modeled device technology. The physical level should provide a set of operators that can be composed to form any arbitrary operator. For example, the phosphorus-silicon model allows for arbitrary rotations around the $\hat{x}$- and $\hat{z}$-axes, and partial SWAP "rotations." Any arbitrary single-qubit operator can be composed from three rotation operators. In addition, if the error model is known *a priori*, then it is possible to compose multiple operators with some error ε so that the error cancels in large part to ε^n:

3. Optimize the operators to reduce the overall operator count. This is done because often adjacent logical quantum operations are decomposed into a set of physical gates, that when such gates are placed back-to-back an even more efficient underlying physical implementation is possible. For example, if an operator is composed of an $\hat{x}$-rotation, a $\hat{z}$-rotation, and a second $\hat{x}$-rotation, applying a second operator would juxtapose two $\hat{x}$-rotations, that can be combined.

4. Verify that all constraints at the physical and structural levels are met, such as supply of cold states, space for classical control circuitry, etc.

Next, a loop is performed with the body of the loop being something like an n-body simulation. Each element of this simulation is a physical quantum bit, and may have an operator applied at any time step. All operators that can be currently applied are applied:

1. Check for circuit-layer constraints, such as adjacency. If a constraint is unmet, raise an error.

2. Accrue error. As discussed in Section 3, to a first order approximation, error accrues linearly with time (static decoherence) and operator application.

3. Remove any error due to an error correction process, by applying circuit-level error-mitigation functions associated with measurement operators.

6. CONCLUSION

Quantum simulation has been limited to small systems due to the exponentially expanding state required to simulate larger systems. However, results that are useful to quantum architecture research do not require the outcome of a state simulation. Rather, statistics about the speed and reliability of an algorithm are significant. This paper puts forward a proposal to ignore the majority of state information, precisely calculate the run-time and pessimistically estimate the reliability, in order to be able to run larger, more meaningful simulations in polynomial time. This pragmatic approach to quantitative research into quantum systems, side-steps the difficult challenge of exponential resource consumption, while still providing useful insight to the architect.

7. TUTORIAL ON QUANTUM COMPUTATION

Quantum computation is computing on data encoded in quantum states. A detailed description of quantum computation is beyond the scope of this paper, and the reader is encouraged to consult the literature [6].

In general, quantum computation is performed on binary quantum states, or qubits, much like classical computation is performed on binary digits. The details of the underlying quantum mechanism are orthogonal to what calculations can be performed–all quantum computers can efficiently compute the same class of problems, and many systems have been demonstrated or proposed for quantum computation. A few examples are vibrational energy states of "trapped" ions, polarization of photons, bulk nuclear magnetic resonance of specially crafted molecules, spins of phosphorous-31 nuclei embedded in isotopically pure silicon-28, quantum dots, and Josephson junctions. The only requirements are that a minimum set of quantum operators exist to manipulate single qubits, interact multiple qubits, and read (measure) the result.

Analogous to the classical case, the two quantum states are labeled 0 and 1, or more precisely, $|0\rangle$ and $|1\rangle$. Unlike the classical case, where a bit is either in the 0 state or the 1 state, a qubit's state is described by $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex valued amplitudes; the probability of the qubit being observed in the state $|0\rangle$ is $|\alpha|^2$, and in state $|1\rangle$ is $|\beta|^2$. If the qubit is not in a pure $|0\rangle$ or $|1\rangle$ state, it is said to be in a *superposition*, where the waveforms for each of the states are added together. Observation, or measurement, collapses the superposition to a single state.

Just as the number of states that can be represented by a classical set of bits doubles with each additional bit, each additional qubit doubles the number of states that is represented by a quantum system. The difference is that the system itself can be in a superposition of any or all of the states, and, more importantly, any operator applied to an n-qubit system can impact the amplitudes of all 2^n states simultaneously, as opposed to the single state transition allowed for a classical system. This exponential speed-up in operation is balanced by the fact that only a single state will ever be read: measuring the system collapses all of the states in the superposition to the single state observed, and the amplitudes of the remaining states are set to zero. As a concrete example, it is possible to perform a quantum Fourier transform (QFT) on the amplitudes of a quantum system. The result of the transform is the frequencies associated with the discrete sample encoded in the original amplitudes. However, measurement will only return one state. The probability of a particular state being returned is proportional to the square of the modulus of the amplitude for that "frequency," so the measured state is likely to represent a stronger frequency. But this is clearly not useful for most common applications of a Fourier transform[1]!

In our classical analogy, an entire system that can perform any algorithm can be built from a small *universal* set of gates, such as NOT, AND, and OR, or even more simply, from NAND gates. Quantum gates take a system from one complex state to another, so one would expect the gates to be more complex, and indeed, they are. However, it is possible to form a universal set of quantum gates from just three operators: CNOT, H, and T. But it is easier to understand quantum operators from the basics. The quantum equivalent of a NOT gate is the X gate, which takes $\alpha|0\rangle + \beta|1\rangle$ to $\beta|0\rangle + \alpha|1\rangle$. Similarly, the *phase* or Z gate takes $\alpha|0\rangle + \beta|1\rangle$ to $\alpha|0\rangle - \beta|1\rangle$, changing the relative phase of the states by π. The Z gate has no classical analogue, since there is no concept of phase in a singly-valued bit. The Hadamard operator, or H gate, is slightly more complex. Applying H to $|0\rangle$ gives $H|0\rangle = \frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle$, while $H|1\rangle = \frac{\sqrt{2}}{2}|0\rangle - \frac{\sqrt{2}}{2}|1\rangle$. A little bit of thought should convince the reader that these three gates are each their own inverse: $HH|0\rangle = |0\rangle$. The T gate is similar to the Z gate in that it changes the relative phase of the states, but the T gate only changes the phase by $\pi/4$. The last of the operators required for a universal set is the CNOT gate, which takes a pair of qubits and inverts the second (target) qubit if the first (control) qubit is $|1\rangle$. Hence, $\text{CNOT}(\frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|10\rangle) = \frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$.

The state $\frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$ is an important enough state to have several names. It is called an EPR pair, after the Einstein-Podolsky-Rosen paradox, a cat state after Schrödinger's thought experiment, and is also one of the Bell states. The reason the state is important is that the qubits are *entangled*: they contain information as a pair that neither qubit alone has. Taken separately, each qubit has an equal probability of being measured as $|0\rangle$ or $|1\rangle$. However, since the two states where the qubits have different values have zero amplitude, the pair of qubits must measure to the same value of either $|0\rangle$ or $|1\rangle$.

Unlike classical bits stored in modern computers, qubits are relatively fragile. Interactions with the environment can cause unwanted entanglement, or *decoherence*, introducing erroneous states into the computation. One of the greatest challenges in designing quantum computers is how to reduce decoherence. Operators, being external fields, are also a significant source of decoherence, since the precision of the operator being applied is limited by the physical process that implements it. One way to battle decoherence is by using error correction. While the qubits cannot be directly measured without destroying their state, it is possible to encode the quantum state in such a way that errors can be greatly reduced. For example, the simplest classical error correction code is triple redundancy: 0 is encoded as 000, and 1 is encoded as 111. Applying this to quantum states, we can encode $\alpha|0\rangle + \beta|1\rangle$ as the three-qubit state $\alpha|000\rangle + \beta|111\rangle$. Decoherence introduces states where the bits have different values, by flipping the value of one of the three encoding qubits. Measuring the pairwise parity of the qubits (using $|0\rangle$ *ancilla*) collapses the system back to two states. In addition, the measurements indicate which, if any, of the qubits was flipped. Applying an X operator to the flipped qubit flips it back. There are many quantum *block codes* known that protect against both bit-flips and phase-flips, and also allow operators to be applied to the encoded qubits relatively simply.

[1]However, QFT is an essential component of Shor's algorithm, which factors an n-bit number in time $O(n^3)$.

8. ACKNOWLEDGMENTS

This work is supported in part by the DARPA Quantum Information, Science and Technology Program, by NSF CAREER grants to Mark Oskin and Fred Chong, an NSF NER grant and a UC Davis Chancellor's Fellowship to Fred Chong.

9. REFERENCES

[1] A. Ac. Statistical distinguishability between unitary operators. *quant-ph/0102064*.

[2] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. 404:597, 2000.

[3] A. W. Harrow and M. A. Nielsen. How robust is a quantum gate in the presence of noise? *quant-ph/0301108*, 2003.

[4] D. Kielpinsky, C. Monroe, and D. Wineland. Architecture for a large-scale ion trap quantum computer. *Nature*, 417:709, 2002.

[5] M. Levitt. Composite pulses. *18*, pages 61–122, 1986.

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

[7] M. T. Niemier. A potentially implementable fpga for quantum dot cellular automata. *1st Workshop*

on Non-Silicon Computation, 2002.

[8] B. Ömer. Quantum programming in qcl. Master's thesis, Technical University of Vienna, 2000.

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

[10] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comp.*, 26(5):1484–1509, 1997.

[11] A. Skinner et al. Hydrogenic spin quantum computing in silicon: a digital approach. *quant-ph/0206159*, 2002.

[12] M. Stillerman, D. Guaspari, and W. Polak. Final report–a design language for quantum computing. Technical report, DARPA, 2003.

[13] M. Whitney, Y. Patel, N. Isailovic, and J. Kubiatowicz. Can we build classical control circuits for silicon quantum computers? *2nd Workshop on Non-Silicon Computation*, 2003.