Optimization of Quantum Computer Architecture using a Resource-Performance Simulator

Muhammad Ahsan
Department of Computer Science
Duke University
Durham, North Carolina 27705
Email: ahsan@cs.duke.edu

Jungsang Kim
Department of Electrical and Computer Engineering,
Physics and Computer Science
Duke University
Durham, North Carolina 27705
Email: jungsang@duke.edu

***Abstract*—The hardware technology characterized by the device parameters (DPs) often drives the architectural optimization in a novel computer design such as the quantum computer. We highlight the role of DPs by quantifying the performance of a fully error-corrected 1024-bit quantum carry look-ahead adder on a modular, reconfigurable architecture based on trapped ions. We develop a simulation tool that estimates the performance and resource requirements for running a quantum circuit on various quantum architectures as a function of the underlying DPs. Using this tool, we found that (1) the latency of the adder circuit execution due to slow entanglement generation process for qubit communication can be adequately eliminated with a small increase in entangling qubits, and (2) the failure probability of the circuit is ultimately determined by the qubit coherence time, which needs to be improved in order to reliably execute the adders comprising core of the Shor's algorithm.**

***Index Terms*—quantum architecture, device parameter, resource performance trade-off, performance simulation**

I. Introduction

Performance of computing systems in the early stages of development is dictated by the quality of component devices in the underlying hardware. Current quality of available quantum bits (qubits), and the logic operations and interconnects between them calls for quantum error correction (QEC) and fault-tolerant construction, which requires substantial resource overhead in implementing a functional quantum computer (QC). A number of scalable architecture designs have been proposed [1]–[7], but a process of optimizing the architectural choices for the best system performance based on rapidly evolving device parameters (DPs) of the hardware platform remains a challenging problem.

The performance of a QC architecture can be quantified using adequate simulation tools [3], [8]–[10]: these tools generally (1) map application circuits on the quantum hardware, (2) generate the sequence of quantum logic gates operating on the qubits, (3) simulate the scheduling and execution of these gates, and (4) estimate the performance metrics such as total execution time and failure probability. The accuracy of the estimated performance hinges on the ability of the tool to simulate realistic constraints of the hardware architecture, such as the availability of scarce resources and the ability to transport qubits from one place to another within the QC [11].

In this paper, we present a novel performance simulation tool with the added flexibility adequate for use in architectural optimization as the parameters of the component devices change. In particular, this simulator is designed to

- handle *reconfigurable quantum hardware*, where device parameter (DP) values can be adjusted
- handle *reconfigurable quantum architecture*, where resource investment scenarios can be studied
- output concrete *breakdown of performance metrics* (e.g. total execution time and failure probability) and *contents of resource overhead* (such as qubits used as ancilla or for communication purposes). This feature allows the user to readily identify performance bottlenecks.

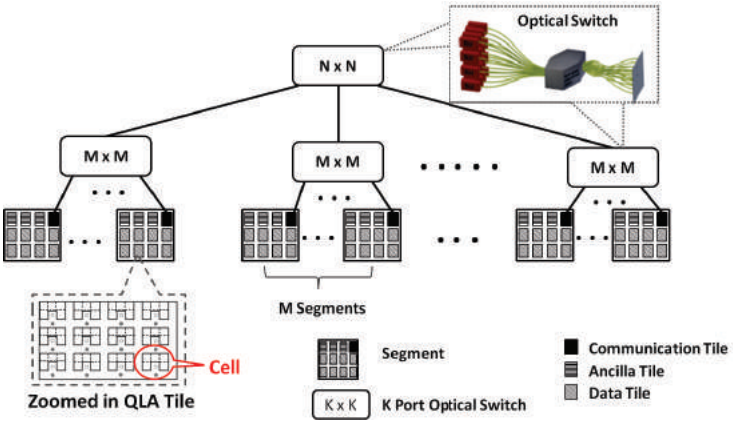
Using this tool, we analyze the performance characteristics of a trapped-ion-based distributed modular QC architecture [5]. Our tool allows us to quantify the system performance for a given QC architecture as a function of the DPs. Given a set of DPs, the architecture space is searched by varying different types of resource investments to improve the performance for a benchmark algorithm, a quantum carry look-ahead adder (QCLA) [12] in our example. This resource-performance trade-off is measured by and optimized for the qubit-delay product (QD), a metric which captures the effect of performance gain in execution time per unit resource investment. Once we extract the best performance from the optimized architecture, the result naturally exposes the DPs that impose bottleneck on the system performance. This approach provides concrete guidelines for the experimental research to improve the critical DPs towards the construction of large scale QCs.

We organize the rest of the paper as follows: Section II describes the underlying quantum hardware technology and the modular QC architecture used in our simulation. Section III briefly details our toolset, and the results and relevant discussions are given in Section IV. Section V concludes the study with possible directions for the future work.

II. Quantum Hardware and Architecture Models

In this paper, we refer to *hardware* as the underlying physical device technology to perform quantum computation, such as trapped ions and superconducting circuits. On the other hand, the term *architecture* describes the allocation and

978-3-9815370-4-8/DATE15/©2015 EDAA



Physical Quantum Operation	Time	Failure Rate
Single Qubit	1μs	10^{-7}
Two Qubit	10μs	10^{-7}
Three Qubit	100μs	10^{-7}
Measurement	100μs	10^{-7}
Shuttling Time/cell	1μs	10^{-7}
EPR generation	5000μs	10^{-4}
Qubit Memory	10s	$1\text{-}e^{-0.17}$

Device Parameter Table

Fig. 1. Overview of the reconfigurable quantum computer architecture analyzed in our performance simulation tool.

arrangement of various quantum resources as larger QC systems are considered. For example, the size and the bandwidth of quantum communication channel and amount of qubits dedicated to fault tolerant quantum operations are parameters of the architecture.

A. Quantum Hardware Model

Our model of the quantum hardware is based on trapped ion technology, in which the qubit is stored in two internal states ($|0\rangle$ or $|1\rangle$) of the atomic ion (*e.g.*, $^{171}Yb^{+}$ ion [13]). The ion qubits are assumed to be individually addressable, and a single-qubit quantum gate is accomplished by simple application of laser pulse(s) on the qubit in its original location. The two-qubit gate, on the other hand, requires that both ions are brought in proximity before the laser pulses are applied. This proximity can be achieved either by transporting actual ions through dedicated physical channels called *ballistic shuttling channels* (BSCs), or by transporting quantum information stored in the ion using *entanglement links* (ELs) [5] generated by entangled qubit pair (also known as EPR pair). The fidelity degradation of a BSC as a function of distance tends to limit its utility to short distance communication only. In contrast, EL can be realized with distance-independent fidelity, which makes it a suitable candidate for long distance communication. We note that the generation time of reliable EPR pairs is currently a slow process due to technology limitation, but can be reduced by using *pipelined EPR Generation* (PEG) at the expense of additional qubit resources [5].

B. Quantum Architecture Model

Fig.1 shows the hierarchical model of the modular scalable universal ion-trap QC (MUSIQC) architecture chosen for our analysis [5]. At the bottom of the hierarchy, architecture contains **Tiles** made of memory cells and BSC regions. Memory cells provide storage and manipulation of qubits for quantum gates, whereas BSCs allow rapid transportation of qubits to facilitate multi-qubit gates between memory cells. The physical qubits in the Tile are encoded into a logical qubit using a quantum error correcting code (QECC) of choice (the Steane [[7,1,3]] code [14] is used in our model), and the error correction is performed at regular intervals to preserve the logical qubit from degrading. Multiple layers of encoding can be concatenated to dramatically improve the protection against errors at the expense of increased number of qubits and circuit complexity used for the error correction. For example a Tile supporting two layers of Steane encoding (L2 Tile) has lower failure probability compared to single layer of encoding (L1 Tile) but contains up to seven times more qubits. In addition to the error correction, Tile is equipped to perform other necessary operation which defines its architectural functionality: state preparation (*Ancilla* Tile), basic fault tolerant quantum gate operations (*Data* Tile) and long distance communication between two Tiles (*Communication* Tile). These Tiles are sufficient to construct a scalable QC in our architecture.

At the next level of the hierarchy, multiple Tiles are arranged to form a **Segment** that performs complex fault tolerant circuit level gate operations. For example, a Segment groups necessary set of Tiles to execute a fault tolerant Controlled-Controlled-NOT (CCNOT, or Toffoli) gate, widely used in quantum arithmetic circuits [12]. The Tiles within the Segment communicate through BSC. As more Tiles are assembled into a Segment, the communication among Tiles within a Segment will require constituent qubits to move over longer distances, where shuttling becomes costly. Therefore at the the third level of hierarchy, Segments are connected to each other through a reconfigurable optical cross-connect switch [15]. The entangling ions comprising the *Communication* Tiles from any pair of Segments within the system can be connected by establishing an EPR pair between them using the photonic channel. This feature makes the cost of inter-Segment communication (and gates between logical qubits in the corresponding Segments) independent of distance, a unique feature of the MUSIQC architecture that provides significant advantages in implementing useful quantum circuits [5].

The performance of the QC for executing a chosen application will depend on the size and composition of the Segments used in the architecture. By varying the architecture parameters (resources) such as number of Segments (*NSeg*),

total *Ancilla* Tiles (*NAnc*), and *Communication* Tiles per Segment (*NComm*), we can minimize the impact of DPs on the performance metrics: execution time T_{exec} and failure probability $1 - P_{succ}$. Identifying specific DPs that become performance bottlenecks for various architectures is the main contribution of this work.

C. Error Model and Baseline Device Parameters

For the simulations, we assume the following as the baseline DPs. A depolarizing channel (equal probability of bit flip, phase flip and bit-and-phase flip errors) is assumed as the underlying error model [16], with fixed physical gate error probability of 10^{-7} for both single-qubit and two-qubit gates. Furthermore, qubit memory error is modeled as an exponential decay in its fidelity $F \sim \exp(-at)$, where a (=$1/T_{coh}$) is determined by the coherence time of the qubit assumed to be T_{coh}=10s, and t is the time between quantum gates over which qubit sits idle (*No-Op*). It is assumed that a single-qubit gate takes $1\mu s$, a two-qubit gate $10\mu s$, and a three-qubit Toffoli gate and measurement each takes $100\mu s$. For BSC, since the dimensions of the state of the art ion-trap cell described in [17] fall in the ∼mm size range, we use $T_{shutt} = 1\mu s$ as the time it takes for an ion to be shuttled through the cell. For a L2 Tile which contains 600 cells arranged in a 2-D grid, the time to shuttle logical qubit through the Tile is taken as $60\mu s$.

For EL, we assume that the entanglement generation time T_{gen} between two Segments using one optical switch is $5,000\mu s$. The size of the optical switch is restricted to 1,000 ports [15] which can connect up to 20 Segments using 49 optical fibers/L2 *Communication* Tile. This means that as the number of *NSeg* or *NComm* increases, multiple optical switch ports will be needed to connect the Segments, which are arranged in a tree-like network topology, as shown in Fig.1. The time cost of establishing an entangled pair through such a network will depend on the the height H of the switch in the tree hierarchy, as the photons will have to propagate through additional switches incurring more photon loss. We adjust new T'_{gen} as $2^H \times T_{gen}$. The resulting EPR pair has non-zero infidelity characterized by EPR_{inf} which will be taken as 10^{-4}, unless otherwise specified. These DP values are optimistic, but most of them are realistically achievable through aggressive technology development in the near future.

D. Benchmark Application Algorithm

We select a 1,024-bit logarithmic depth QCLA as the benchmark algorithm [12]. Such adder circuit forms the elementary building block of a modular exponentiation circuit that underlies the execution time of Shor's prime factorization algorithm [18]. The bulk of the QCLA circuit contains Toffoli gates that can be reliably and concurrently executed if sufficient *Ancilla* and *Communication* Tiles are available. Since the performance of this benchmark is a strong function of architectural resources, it is a suitable candidate for studying resource-performance trade-offs in QC architecture design.

III. Tool Description

Our tool consists of two main components: Tile Designer and Performance Analyzer (TDPA), and Architecture Designer and Performance Analyzer (ADPA). TDPA maps the fault tolerant circuit blocks to the Tile whose performance is then computed and stored as functions parametrized by DPs similar to that described in [11]. This allows TDPA to produce a spectrum of Tiles for ADPA, with different number of physical qubits having different performance levels depending on DP values. For baseline DPs in Fig.1, TDPA computes the performance of a unified L2 Tile (which can act as *Data*, *Ancilla* and *Communication* Tile) in Table I.

ADPA components are designed to map application circuits onto the reconfigurable architecture shown in Fig. 1, described by user-provided architecture parameters and DPs. Based on these parameters, initial map of available Tiles (obtained from TDPA) to be assembled in the Segments is first generated. During this process, the Assembler maximizes the locality among *Data* Tiles (computed from the connectivity of logical qubits in the circuit) by solving *optimal linear arrangement problem* using efficient graph-theoretic algorithm [19]. The Scheduler of ADPA generates the sequence of gates for QCLA execution by solving the standard *resource-constraint scheduling problem* using *ALAP* (as-late-as-possible) heuristic. T_{exec} is minimized by reducing the *circuit critical path* through maximum utilization of available resources. The Scheduler also maximizes P_{succ} by scheduling error correction on *Data* Tiles at regular intervals when they sit idle. The overall P_{succ} is computed by Error Analyzer as $\prod_{i=1}^{i=N}(1-P_{Li})$, where P_{Li} is the failure probability of the i-th logical gate and N is the total number of logical operations in the circuit. The failure probability for each logical gate was pre-computed using *fault path counting method* [20]. Error Analyzer keeps track of the operational source of each P_{Li} [such as shuttling, teleportation, memory (or *No-Op*) or gate] while scheduling is in progress. This way, $1 - P_{succ}$ is considered as a sum of errors for shuttling, teleportation, memory and gate operations.

ADPA also performs breakdown of T_{exec} using Critical Path Analyzer, which (1) splits the circuit critical path into transportation latency overhead incurred due to the movement of qubits across the hardware (this includes Shuttling overhead T_{ovhd}^{SHUT} and Teleportation overhead and T_{ovhd}^{TEL}), and (2) calculates total number of logical gates comprising critical path N_{ovhd}^{CRT} that captures the magnitude of serialized execution of parallel operations, enforced by fewer shared resources. The splitting of the performance variables is a unique feature of our tool, which allows us to identify critical DPs and architecture parameters to drive QC architecture design optimization.

IV. Simulation Results

Our analysis strategy systematically divides architecture space into subspaces characterized by performance-limiting resource and DPs. Within each subspace, we select a suitable architecture that minimizes qubit-delay (QD) product and improve the performance by providing more resources or tuning

TABLE I
L2 TILE PERFORMANCE NUMBERS

Logical Operation	$T_{exec}(\mu s)$	$1 - P_{succ}$
Pauli gate (X, Z)	1	7.4×10^{-24}
Hadamard gate	4	7.4×10^{-24}
CNOT gate	18	4.74×10^{-22}
Toffoli gate	108	2.9×10^{-21}
7-qubit cat-state prep.	780	2.67×10^{-22}
Measurement	814	1.32×10^{-17}
Error Correction	4830	6.86×10^{-16}
State prep. ($\|\overline{0}\rangle$,$\|\overline{+}\rangle$)	5644	6.99×10^{-16}
EPR pair connection	$T_{gen} + 5662$	6.3×10^{-11}

DPs. We found that T_{exec} can be reduced to its minimum value by investing in the architecture resources, whereas the failure probability $1-P_{succ}$ can be minimized by tuning specific DPs.

A. Searching in the Architecture Space

Architecture space is explored by analyzing the performance metrics as functions of *Ancilla* and *Communication* Tiles provided to the hardware (Fig. 2 and Fig. 3), equally distributed among Segments. Figure 2(a) shows T_{exec} as a function of *NAnc*, for various architectures where the QC is divided into *NSeg* segments. This result slices large combinatoric space spanned by the architecture parameters into two regimes: (1) *SA-Regime* where the performance continues to increase as more *Ancilla* Tiles are provided, and (2) *Tel-Regime* where the performance improvement saturates beyond certain *NAnc*. A large number of *Data* Tiles are clustered in fewer Segments (small *NSeg*) in *SA-Regime* architectures, while *Tel-Regime* architectures consist of large number of Segments (large *NSeg*) each containing small number of *Data* Tiles.

For the *SA-Regime* curves (*NSeg* $<$ 16), T_{exec} is bottlenecked by *NAnc*. Smaller number of *Ancilla* Tiles shared among large number of *Data* Tiles in the Segment leads to serialized scheduling of the Toffoli gates that are otherwise concurrently executable, resulting in the increased T_{exec}. Furthermore, since most of the communication is accomplished through BSC within the Segment, shuttling error is the main source of the failure probability as shown in Table II. As *NAnc* increases, the expansion in Segment size causes longer shuttling distance through which qubits are transported. Longer waiting times for the memory qubits raises the failure probability with increasing *NAnc* (Fig. 3).

For *Tel-Regime* curves (*NSeg* $\geq$16), T_{exec} remains almost flat with *NAnc* but strongly depends on *NComm* as shown in Fig. 2(b). In this regime, increase in T_{exec} caused by slow EPR pair generation rate can be compensated by adding *NComm* to generate multiple EPR pairs in parallel. However, increasing *NComm* beyond certain point (*NComm* $>$ 6 in Fig. 2(b)) results in increased height *H* of the optical switch hierarchy since more switches are needed to connect added *Communication* Tiles. In this case, the time overhead of EPR pair generation using top-tier optical switches outweighs the

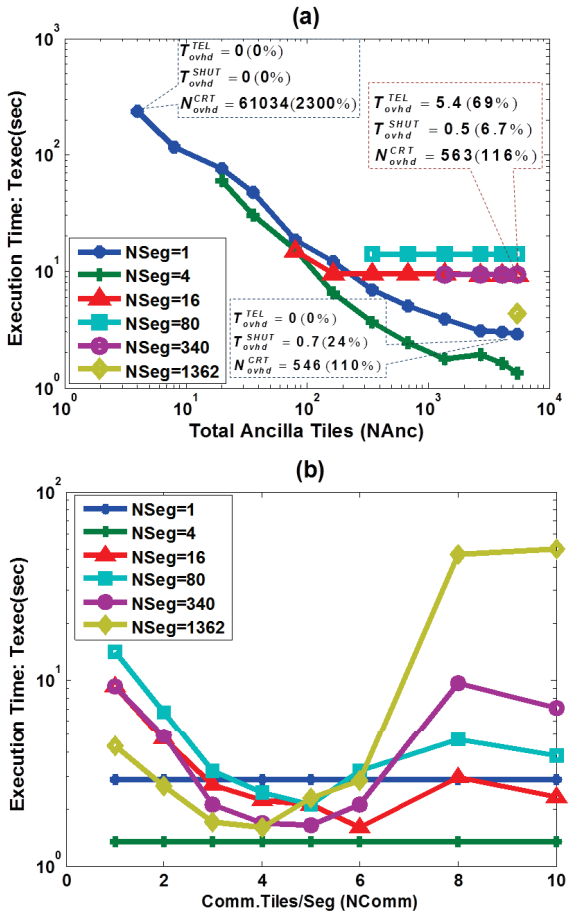


Fig. 2. Execution time plotted as a function of (a) *NAnc* and (b) *NComm*. (a) shows curves for both SA-and Tel- regimes and sample breakdown of T_{exec} for certain points for *NComm* = 1. Each Segment needs to have minimum of 4 *Ancilla* Tiles for preparation of ancilla state needed for Toffoli gate, so minimum *NAnc* increases with *NSeg* and reduces the number of data points for larger *NSeg*. (b) shows T_{exec} as a function of *NComm* for maximum *NAnc* = 5448. Note that *SA-Regime* curves (*NSeg*$<$ 16) remain flat with *NComm* since T_{ovhd}^{TEL} does not contribute to the execution time.

benefit of added *NComm*, thus increasing T_{exec}. In *Tel-Regime*, the failure probability that remains constant with *NAnc* (shown in Fig. 3) is mainly contributed by (1) teleportation errors arising from the EPR pair generation, and (2) memory errors while waiting for the slow EPR pair generation process. The relevant contributions of these errors are shown in Table II.

B. Selecting an Architecture

Once resource bottlenecks are identified for the *SA-* and *Tel-regimes*, we identify optimal architectures in each regime by considering both resource budget and performance captured by the qubit-delay (QD) product. Figure 4 shows the performance (T_{exec} on the left axis) and resources (total number of physical qubits on the right axis) as a function of relevant resource bottlenecks for the *SA-Regime* [*NAnc*, Fig. 4(a)] and *Tel-Regime* [*NComm*, Fig. 4(b)]. For clarity, we present two suitable examples from each regime for comparison (among all the architectures analyzed) that yield smallest QD metric.

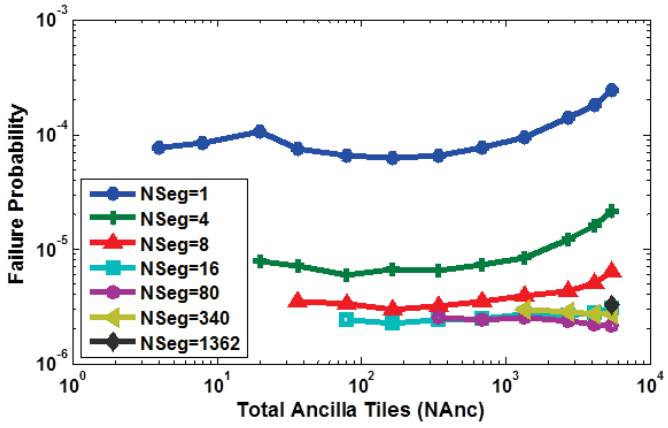


Fig. 3. The failure probability ($1 - P_{succ}$) as a function of *NAnc*. The failure probability shows an overall increasing trend for the curves in *SA-Regime*, while it remains almost flat in *Tel-Regime*.

TABLE II
BREAKDOWN OF THE $1 - P_{succ}$ FOR DIFFERENT *NSeg*. THE AVERAGE VALUE IS COMPUTED OVER COMBINATIONS OF *NAnc* AND *NComm*.

Segments (NSeg)	% Shuttling Error	%Teleportation Error	% Memory Error	Average $1\text{-}P_{succ}$
1	≥ 96.9%	= 0 %	≤ 3.1%	1.05×10^{-4}
4	≥ 66.4%	≤ 0.8%	≤ 33.4%	9.9×10^{-6}
8	≥ 34.4%	≥ 1.33%	≥ 25%	6.04×10^{-6}
16	≤ 48.6%	≥ 2.86%	≥ 48.7%	3.1×10^{-6}
80	≤ 4.3%	≥ 12.5%	≥ 77.7%	2.05×10^{-6}
340	≤ 2.7%	≥ 25%	≥ 69%	2.54×10^{-6}
1362	≤ 0.2%	≥ 36.7%	≥ 45.3%	2.7×10^{-6}

For the architectures in the *SA-Regime*, we show two representative simulations for *NSeg* = 4 and 8 in Fig. 4(a). As *NAnc* increases, T_{exec} for the less distributed case (*NSeg*=4) continues to decrease, whereas the more distributed case saturates as the computation becomes limited by T_{gen}. Since the total number of qubits in both cases are the same, we see that (*NSeg*=4, *NComm*=1, *NAnc*=1,362) is an optimal case that minimizes QD metric in the *SA-Regime*. This is confirmed by comparing the QD metric for other *NSeg* values in this regime. The performance metric for running the benchmark algorithm is ($T_{exec} = 2.22s$, $1 - P_{succ} = 8.86 \times 10^{-6}$) for this case.

For the architectures in the *Tel-Regime*, we show two representative simulations for *NSeg* = 16 and 1,362 in Fig. 4(b), when sufficient *NAnc* is provided (5,448). From the comparison of T_{exec}, we see that both architectures reach the minimum computation time of about 1.3s, although at different values of *NComm* (4 per Segment for *NSeg* = 1,362 case, and 6 per Segment for *NSeg* = 16 case). However, the total number of qubits is much less when *NSeg* = 16, so this is a better architecture in terms of QD metric. For this architecture, T_{exec} decreases about 6 folds for nominal (4.2 %) increase in total number of qubits. By comparing a range of *NComm* and *NAnc* values in this regime, we conclude that (*NSeg*=16, *NComm*=6, *NAnc*=1,362) is the optimal design in the *Tel-Regime*. The performance metric for running the benchmark

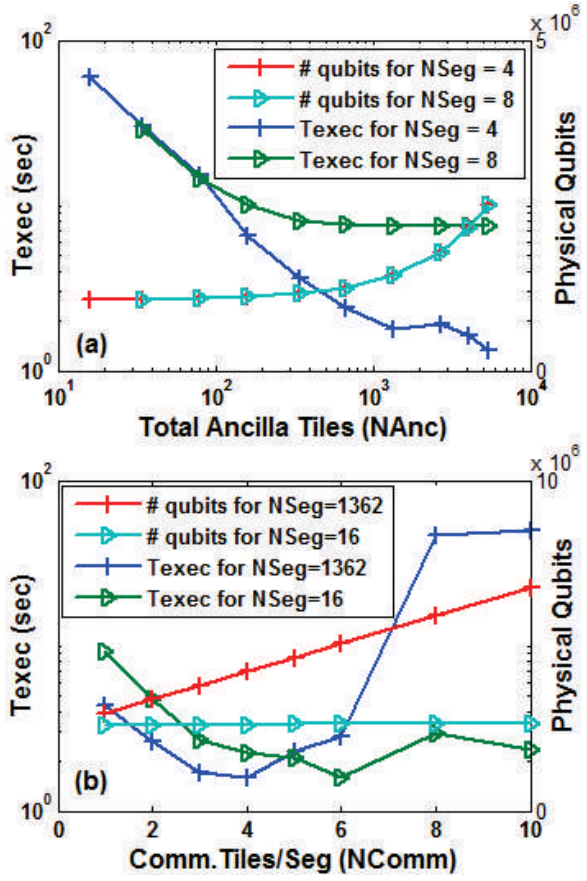


Fig. 4. Comparison of candidate architectures in each regime. Explicit variation of T_{exec} and total number of physical qubits for selected architectures in *SA-Regime* is shown in (a) while those for *Tel-Regime* is shown in (b).

TABLE III
FAILURE PROBABILITY VS. MEMORY ERROR RATE AND SHUTTLING TIME. $T_{gen} = 5000\mu$S. *NComm* = 1, *NAnc* = 1,362. #PHYS. QUBITS = 1,439,788.

Memory Error Rate: $1 - e^{-\alpha T}$	Shuttling Time : T_{shutt}			
	1us	0.7us	0.4us	0.1us
$\alpha = 0.1$	8.86×10^{-6}	3.7×10^{-6}	2.62×10^{-6}	2.46×10^{-6}
$\alpha = 0.07$	2.01×10^{-6}	9.07×10^{-7}	6.35×10^{-7}	5.98×10^{-7}
$\alpha = 0.04$	2.21×10^{-7}	1.02×10^{-7}	7.35×10^{-8}	6.96×10^{-8}
$\alpha = 0.01$	7.59×10^{-9}	6.78×10^{-9}	6.67×10^{-9}	6.65×10^{-9}
$\alpha = 0.001$	6.75×10^{-9}	6.41×10^{-9}	6.41×10^{-9}	6.41×10^{-9}
T_{exec}	2.22s	1.71s	1.71s	1.71s

algorithm is ($T_{exec} = 1.93s$, $1 - P_{succ} = 2.77 \times 10^{-6}$) for this case. Note that in both cases T_{exec} is within a factor of 2 from its minimum value 1.05s, which can be computed by scheduling the benchmark algorithm under no hardware constraints. Adding more qubits to these architectures cannot provide noticeable performance gains, especially for P_{succ}.

C. Improving Performance through Device Parameters

Failure probability for selected architecture can be reduced by improving the dominant sources of errors in respective

TABLE IV
FAILURE PROBABILITY VS. MEMORY ERROR RATE AND PHYSICAL EPR PAIR INFIDELITY. T_{gen} REDUCED TO 250μS USING PEG AT THE EXPENSE OF ADDITIONAL COMMUNICATION QUBITS PER PORT. THE T_{shutt} IS TUNED TO 0.1μS. *NComm* = 6, *NAnc* = 5,448. #PHYS. QUBITS = 2,733,432.

Memory Error Rate: $1 - e^{-aT}$	Infidelity of physical EPR pair (EPR_{inf})			
	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$a = 0.1$	1.47×10^{-6}	1.38×10^{-6}	1.38×10^{-6}	1.38×10^{-6}
$a=0.07$	4.19×10^{-7}	3.3×10^{-7}	3.3×10^{-7}	3.3×10^{-7}
$a=0.04$	1.23×10^{-7}	3.62×10^{-8}	3.62×10^{-8}	3.62×10^{-8}
$a=0.01$	8.77×10^{-8}	9.58×10^{-10}	9.43×10^{-10}	9.43×10^{-10}
$a=0.001$	8.76×10^{-8}	8.18×10^{-10}	8.03×10^{-10}	8.03×10^{-10}

regimes which can be identified from Table II. In the *SA-Regime*, the dominant source of error is memory error while waiting for shuttling operation to complete. By either speeding up the shuttling speed or increasing coherence time of the memory qubits, $1 - P_{such}$ can be reduced dramatically, as shown in Table III. For example, when T_{coh} is increased 10 fold and T_{shut} is accelerated slightly from 1μs to 0.7μs, $1 - P_{succ}$ falls drastically by 10^3 fold (from 8.86×10^{-6} to 6.78×10^{-9}). Further improvements in the shuttling time improves neither P_{such} nor T_{exec}, indicating that the shuttling time is no longer the performance bottleneck in this limit.

In the *Tel-Regime*, EPR pair fidelity and memory errors are the main contributors to the failure probability. In order to further reduce the memory errors arising from waiting for the EPR pair generation, one must dramatically reduce T_{gen}, which can be achieved by pipelined EPR generation (PEG) process [5]. For the numbers shown in Table IV, we assume a 20-fold reduction in T_{gen} using the PEG technique, as well as sufficient ancilla qubits (*NAnc* = 5,448). A ten fold increase in T_{coh} provides more than 10^3 fold reduction in $1 - P_{such}$ (from 1.38×10^{-6} to 9.58×10^{-10}) if the infidelity of the EPR pairs is improved to 10^{-5}.

In order to execute 1,024-bit modular exponentiation circuit necessary for the Shor's algorithm, about 4 million calls to adder is required [21]. Thus we need each 1,024 bit adder to be successfully executed with failure probability of $\ll (4 \times 10^6)^{-1} = 2.5 \times 10^{-7}$. The shaded regions in Table III and Table IV provides combination of memory error rates with shuttling time or EPR infidelity which achieve $1 - P_{succ}$ required to reliably execute 1,024 bit modular exponentiation.

V. CONCLUSION

We highlighted the impact of DPs and architecture choices on the performance of QC systems using a resource-performance simulation tool. This analysis shows that for baseline DPs, the execution time of the benchmark circuit was limited by the number of ancilla qubits for the *SA-Regime* and the communication qubits for the *Tel-Regime*. On the other hand the failure probability of circuit was mainly limited by the coherence time of the memory qubits. Improving this DP by an order of magnitude dramatically enhances the error performance, ensuring successful execution of the modular exponentiation circuit for Shor's algorithm.

The authors thank Rod Van Meter for reviewing the manuscript and the Office of the Director of National Intelligence (ODNI) and Intelligence Advanced Research Projects Activity (IARPA) for supporting this work through the Army Research Office.

REFERENCES

[1] T. Metodi, D. Thaker, A. Cross, F. Chong, and I. Chuang, "A quantum logic array microarchitecture: Scalable quantum data movement and computation," in *Proc. 38th Annual IEEE/ACM Internat. Symp. on Microarchitecture (MICRO-38)*, 2005, pp. 12–23.

[2] R. V. Meter, W. J. Munro, K. Nemoto, and K. M. Itoh, "Arithmetic on a distributed-memory quantum multicomputer," *J. Emerg. Technol. Comput. Syst.*, vol. 3, pp. 2:1–2:23, 2008.

[3] M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz, "A fault tolerant, area efficient architecture for shor's factoring algorithm," *ACM SIGARCH Computer Architecture News*, vol. 37, pp. 383–394, 2009.

[4] J. Kim and C. Kim, "Integrated optical approach to trapped ion quantum computation," *Quantum Inf. & Comput.*, vol. 9, pp. 181–202, 2009.

[5] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large scale modular quantum computer architecture with atomic memory and photonic interconnects," *Phys. Rev. A*, vol. 89, p. 022317, 2014.

[6] A. Galiautdinov, A. N. Korotkov, and J. M. Martinis, "Resonator-zero-qubit architecture for superconducting qubits," *Phys. Rev. A*, vol. 85, p. 042321, 2012.

[7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, p. 032324, 2012.

[8] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov, "A layered software architecture for quantum computing design tools," *Computer*, vol. 39, pp. 74–83, 2006.

[9] M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz, "Automated generation of layout and control for quantum circuits," in *Proc. of the 4th Internat. Conf. on Computing Frontiers*, 2007, pp. 83–94.

[10] S. Balensiefer, L. Kreger-Stickles, and M. Oskin, "Quale: quantum architecture layout evaluator," in *Proc. of the SPIE*, vol. 5815, 2005, pp. 103–114.

[11] M. Ahsan, B.-S. Choi, and J. Kim, "Performance simulator based on hardware resources constraints for ion trap quantum computer," in *IEEE 31st Internat. Conf. on Computer Design (ICCD)*, 2013, pp. 411–418.

[12] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Inf. & Comput.*, vol. 6, pp. 351–369, 2006.

[13] S. Olmschenk, K. C. Younge, D. L. Moehring, D. N. Matsukevich, P. Maunz, and C. Monroe, "Manipulation and detection of a trapped yb^+ hyperfine qubit," *Phys. Rev. A*, vol. 76, p. 052314, 2007.

[14] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, pp. 793–797, 1996.

[15] J. Kim, C. Nuzman, B. Kumar, D. Lieuwen, J. Kraus, A. Weiss, C. Lichtenwalner, A. Papazian, R. Frahm, N. Basavanhally, D. Ramsey, V. Aksyuk, F. Pardo, M. Simon, V. Lifton, H. Chan, M. Haueis, A. Gasparyan, H. Shea, S. Arney, C. Bolle, P. Kolodner, R. Ryf, D. Neilson, and J. Gates, "1100 x 1100 port mems-based optical crossconnect with 4-db maximum loss," *IEEE Photon. Technol. Lett.*, vol. 15, pp. 1537 –1539, 2003.

[16] M. A.Nielsen and I. L.Chuang, *Quantum computation and quantum information*. Cambridge University Press, 2000.

[17] C. Monroe and J. Kim, "Scaling the ion trap quantum processor," *Science*, vol. 339, p. 1164, 2013.

[18] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A*, vol. 54, p. 147, 1996.

[19] M. Juvan and B. Mohar, "Optimal linear labelings and eigenvalues of graphs," *Discrete Appl. Math.*, vol. 36, pp. 153–168, 1992.

[20] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *arXiv:quant-ph/0504218*, 2005.

[21] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Phys. Rev. A*, vol. 71, p. 052320, May 2005.