Characterization of Quantum Workloads on SIMD Architectures

Robert Risque and Adwait Jog
Department of Computer Science
College of William and Mary, Williamsburg, VA
rcrisque@email.wm.edu, adwait@cs.wm.edu

Abstract*— There is a growing research interest in quantum computing because of its promise to provide significant performance speedups over classical computers at specialized tasks. While there have been many advances in building more capable, robust, and useful quantum algorithms and software, it is not clear how a scalable, high-performance, and area-efficient quantum architecture should be designed for efficient execution of various quantum workloads. This paper attempts to fill this gap by performing a detailed characterization of many real-world quantum algorithms on SIMD-style quantum architectures. Specifically, we characterize the effect of size and the number of SIMD regions of a quantum SIMD architecture on various metrics such as performance, utilization, qubit (data) movement etc. We hope that the presented insights on the trade-offs and relationships between aforementioned evaluation metrics will be useful towards designing an efficient and scalable quantum architecture.

I. Introduction

Recent advances in the quantum computing research has significantly increased the possibility of adopting quantum computers in the near future [1]. Many leading computer industries such as Google, IBM, Intel, and Microsoft are making significant investments in quantum computing technology [1]–[4] because of its potential to provide very impressive performance speedups and reduction in computational complexity over traditional/classical super computing clusters for specific tasks such as large-scale factoring and searching operations [5], [6]. These improvements are possible because of the core differences between quantum and classical computing. In contrast to classical computing, quantum computing algorithms traverse through large solution spaces exponentially fast by taking advantage of *qubits* [6]. A qubit, or quantum-bit, is the medium of information storage in a quantum computer, which is a superposition of 0 and 1 states. Therefore, a string of only n qubits can simultaneously represent 2^n different solution states of a problem and thereby facilitating in searching the solution space exponentially faster.

While many of the previous works [7] have focused on making quantum algorithms more robust, realizable, and error-resilient, significant work is required to architect a performance and energy-efficient quantum computer. This issue is even more important than it is in classical computers, because of the fact that quantum bits are highly error-prone and require very expensive Quantum Error Correcting Code (QECC) operations for reliable computation [6]. If the computation times can be reduced by avoiding the addition of extra quantum hardware, the overall probability of incurring errors can potentially be reduced as well.

Currently, the most prevalent quantum computer architecture design will employ ion-trap technology [6], [8] for qubits and quantum operations. Analogous to the well-studied single-instruction multiple data (SIMD) classical computers, Heckey et al. [9] proposed ion-trap technology based multi-SIMD architecture for efficient quantum computation. This architecture consists of multiple SIMD regions (say k) with each region capable of executing a quantum gate operation on multiple qubits (say d) at the same time. Drawing parallels to the classical SIMD computer, k is the number of SIMD cores and d is the width of each core. All these cores communicate to the global memory, where qubits are stored. These qubits are moved to cores as when required for computation. If sufficient amount of inter-region parallelism (to exploit quantum logic gate (instruction)-level parallelism) and intra-region parallelism (to exploit qubit (data)-level parallelism across multiple qubits within a region) is present in the quantum applications, the proposed architecture is capable of concurrently working on $k \times d$ qubits leading to very high-performance speedups. Although this architecture is promising, there is no prior work that performs a deep analysis of the factors that affect different architecture-specific metrics such as performance, utilization, qubit movement, performance/area etc. Our **goal** is to deeply characterize various quantum workloads across different quantum architecture choices to understand the trade-offs and relationships between aforementioned evaluation metrics. We hope that our observations would be helpful towards designing an efficient and scalable quantum architecture. Specifically, this paper makes the following **contributions:**

- We show that many different quantum workloads have varied inter- and intra-SIMD region parallelism requirements. Therefore, a multi-SIMD quantum architecture can either provide inferior performance or be severely underutilized if the parallelism characteristics of a quantum application are not considered during the design of the architecture.
- We analyze the qubit movements for different multi-SIMD architecture choices and show that the number of moves is not only related to the number of regions but also on the width of each region. Therefore, it is imperative to carefully architect a multi-SIMD architecture if qubit movements need to be minimized.
- Finally, we show that there is significant variation in

parallelism and movement characteristics even within a single quantum application at different instances of its execution. Therefore, such variations should also be considered while designing a more efficient quantum architecture.

To our knowledge, this is the first work that comprehensively analyzes multi-SIMD quantum architectures and quantifies the effect of important design parameters (i.e., width and number of SIMD regions) on various metrics such as performance, utilization, qubit movement, and performance/area. The rest of the paper is structured as follows. Section II provides background on quantum computing and quantum SIMD architectures. Section III describes the considered evaluation metrics in the context of quantum architectures along with an illustrative example. Evaluation methodology is described in Section IV followed by experimental results and analysis in Section V. Section VI describes the prior work related to this paper. Finally, we summarize our key conclusions in Section VII.

II. Background

In this section, we provide a brief background on quantum computing, the quantum SIMD architecture, and the evaluated quantum workloads.

A. Quantum Computing

Quantum computing boasts impressive speedups at specific tasks when compared to classical computers. These tasks include database search and integer factoring algorithms [5], [6], [10]; processes that may take a current super computing cluster an unreasonably long amount of time to accomplish. The core difference between quantum computing and classical computing lies in the quantum mechanical properties of qubits. A qubit, or quantum bit, stores the information in a quantum computer. While a traditional bit can only exist in either the 0 or 1 state at any given instance, a qubit can exist in a quantum superposition of both states simultaneously. A qubit state can be represented as:

$$\psi = \alpha|0> + \beta|1>$$

where α^2 and β^2 represent the probability amplitude of the qubit state to collapse into the 0 or 1 state, respectively. A string of N qubits is now capable of representing a superposition of 2^N binary strings simultaneously. A quantum operation performed on this qubit string is therefore intrinsically parallel, equivalent to a classical operation being performed on each binary string constructing the superposition, individually.

Qubits have other special properties that require attention when dealing with quantum information. The exact state of a quantum system cannot be copied from one qubit to another without destroying the original quantum state. This is known as the *no cloning theorem* [6]. Thus, moving information or data within a quantum computer requires either physically moving the qubits via ballistic motion, or transmitting the quantum state via quantum teleportation [6], [9], [11], [12]. Because of the delicacy of quantum superposition in regards to decoherence (i.e., the state of the quantum bit can only be maintained for a short window of time before it decoheres), the interaction between qubits and their environment must be minimized in order to reduce computation errors. The primary source of qubit decoherence is qubit motion required for communication. Thus, as will be discussed in further sections, new techniques need to be developed to reduce the amount of qubit movement.

The most prominent physically realizable quantum computer is the ion-trapped quantum computer [6], [8], [9]. Trapped ions are promising qubit candidates because of their relatively long decoherence times (compared to their operation times) and the ability to confine ions to a small point in space via ion traps (as opposed to linear optical quantum computing (LOQC), which uses photons as qubits, and therefore cannot be trapped as easily) [6]. The two states of the qubits, 0 and 1, can be realized in a trapped ion qubit as either the ground state and excited state, or as two hyperfine states. There also exists experimental protocol for the quantum teleportation of ion quantum states, which as discussed previously is used for the communication of data within a quantum computer.

B. Quantum SIMD Architecture

In this paper, we assume a Single-Instruction-Multiple-Data (SIMD) quantum architecture initially proposed by Heckey et al. [9]. Figure 1 shows the schematic of a SIMD quantum architecture consisting of four (4) SIMD operating regions. If the architecture has more than one SIMD operating region, the architecture is also called as multi-SIMD quantum architecture [9]. However, without the loss of generality, we use multi-SIMD quantum architecture and SIMD quantum architecture terminology interchangeably in the paper. Each SIMD operating region is capable of storing and working on multiple qubits at a time. In Figure 1, we assume that each region can store and work on sixteen (16) qubits at a time. We call this the capacity or the width of the SIMD region.

The actual computations performed on qubits take place in a SIMD operating region. If a particular quantum logic gate operation (e.g., H, CNOT [13] etc.) needs to be performed on qubits, those particular qubits need to be placed in a particular SIMD region. Making analogy to a classical computer, quantum logic gate operation can be considered as a particular *instruction* and qubits in the same region can be considered as *data*. Therefore, a particular instruction (quantum logic gate operation) can be applied to multiple data (qubits) placed in the same SIMD operating region, thereby leveraging data-level parallelism. A quantum logic gate operation is applied to a particular SIMD region with the help of microwave pulse as shown by SIMD control signal in Figure 1. A peculiar property of this microwave pulse is that it is applied to *all* the qubits present in the particular region and hence cannot be selectively applied to a subset of qubits present in the region. Therefore, if a particular gate operation needs to be applied to only a subset of qubits present in the region, the remaining qubits have to be moved to a *shielded* place such as global memory. The qubits stored in the global memory do not get affected by any SIMD control signals (e.g., gate operations) triggered

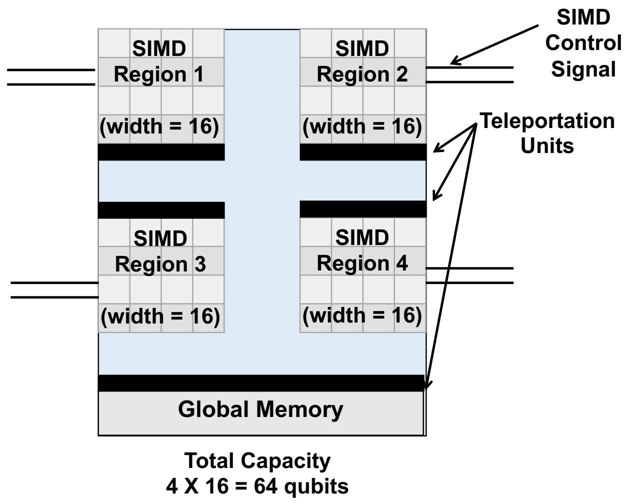

Fig. 1: A SIMD quantum architecture showing four SIMD regions with each region capable or working on sixteen qubits at a time. Therefore, the total capacity of the shown architecture is 64 (4×16) qubits. Note that these numbers are only for illustrative purposes and we have evaluated many different configurations of the SIMD quantum architectures.

at any of the regions. It is important to note that multiple *different* SIMD regions can concurrently apply *different* quantum logic gate operations thereby supporting quantum logic gate (instruction)-level parallelism. To assist qubit movement between regions and global memory, teleportation units are employed (as shown in Figure 1). Please refer to the previous literature on quantum teleportation [6] for understanding the mechanics of the actual physical communication process.

C. Evaluated Quantum Algorithms

In order to evaluate various quantum computing architectures, we study eight quantum algorithms. Some brief notes on each benchmark are as follows:

- ***Binary Welded Tree (BWT):*** Traversal of two balanced binary trees based on a continuous time quantum walk, as opposed to quantum algorithms that employ Quantum Fourier transform (QFT). Finds path between entry and exit node of the graph [10].
- ***Class Number (CN):*** Polynomial-time quantum algorithm for the computation of the unit group and class group of a number field. This quantum algorithm is faster than classical algorithms as it uses Quantum Fourier transform to solve Pell's equation [14].
- ***Ground State Estimation (GSE):*** Quantum chemistry algorithm that utilizes quantum phase estimation to obtain the energy of a molecular system with fixed nuclear geometry, with the result stored in a qubit string corresponding to spin-orbital states. Provides means to simulate a chemical Hamiltonian on a quantum computer [15].
- ***Ising Model (IM):*** Quantum computing implementation of a mathematical model of ferro-magnetism used in both statistical mechanics and computer science. The Ising Model is a model consisting of two discrete spin variables, but can be applied in a variety of focuses, including magnetic materials, lattice gases, binary allows, neural systems, and economic models [16].
- ***Quantum Fourier Transform (QFT):*** Linear transformation on qubits; quantum analog of discrete Fourier transform. Used to computing discrete logarithm and quantum phase estimation [17].
- ***Square Root (SR):*** Quantum algorithm that computes the square root in a polynomial ring. Used in Grover's search algorithm, a quantum algorithm which employs amplitude amplification to search a database of elements [9].
- ***Toffoli (TOFF):*** Quantum logic gate, also known as controlled-controlled-NOT gate (CCNOT gate). We have included the Toffoli gate in our group of benchmarks as an example to explain our evaluation metrics of quantum computation performance [18].
- ***Triangle Finding Problem (TFP):*** Finds a triangle (clique of size 3) within a dense undirected graph using a quantum random walk [19].

III. Evaluation Metrics for Quantum Architectures

In this section, we define and describe various metrics to evaluate quantum SIMD architectures, followed by discussions of these metrics in the context of Toffoli quantum algorithm.

A. Evaluation Metrics: Definitions and Goals

(I) Quantum logic gate and qubit operations. We primarily focus on two architectural parameters of the SIMD quantum architecture (Section II): number of SIMD operating regions (k) and qubit capacity or width of the SIMD operating regions (d). We assume that each SIMD region can perform one quantum logic gate operation in one cycle, where one cycle is defined as the amount of time required for the quantum logic gate operation to be applied to all qubits present in the SIMD operating region. We define such an operation on a qubit as a *qubit operation*. We further assume that all the quantum logic gate operations present in the considered quantum algorithms take the same amount of time if all the desired qubits are available in the SIMD region. Considering these assumptions, the maximum number of *qubit operations* that can be performed in one cycle is equal to the sum of the maximum capacity of all regions, which is equal to the product of number of SIMD operating regions (k) and qubit capacity of each SIMD region (d). For example, a SIMD architecture with k = 4 and d = 16, will have a maximum capacity of 64 qubits (also see Section II). This system can apply a maximum of 4 different quantum logic gate operations (1 operation per SIMD operating region) in a cycle, operating on a maximum of 16 qubits per region, leading to 64 qubit operations per cycle.

(II) Performance. A SIMD architecture presents the opportunity to execute many qubit operations in parallel. In order to quantify a quantum algorithm's performance, we measure qubit operations per cycle (QOPC), when it is executed on a quantum architecture. In order to achieve the greatest algorithm speedup, the architecture must maximize QOPC as it is directly related to the time requirements to run a quantum program. The maximum value of QOPC for an architecture is the product of k and d. If the application has many data-dependency chains and is serial in nature, it will not be able to take advantage of the available intra- and inter-region parallelism in the architecture. This is true for applications such as GSE and QFT, whose QOPC value is far away from the maximum achievable value of it. On the other hand, other algorithms such as CN and IM have ample amount of parallelism. We discuss these trends in more detail in Section V.

(III) Performance/Capacity. One may assume that in order to maximize quantum algorithm performance, k and d should be as high as possible. This is true when considering algorithm performance alone, however when considering a quantum architecture efficiency, one must examine other metrics such as the performance per capacity, where capacity is considered to be directly related to the area, cost, and power consumption of the architecture. The higher the capacity of the architecture, the higher its area, cost, and power consumption. Because each operating region takes up both power and physical space, a quantum architecture needs to be designed to optimize QOPC while not wasting resources and capacity. In other words, QOPC needs to be optimized while keeping the utilization of the architecture as high as possible. To capture this trend, we consider the Performance/Capacity (P/C) metric, which has the similar notion as that of the well-known metric used for classical computers: Performance/Area.

(IV) Qubit (Data) Movement. The most prominent cause of qubit interaction with the external environment comes from qubit (data) movement between global memory and different SIMD regions. Such movements can cause the loss in quantum information and induce errors. However, such qubits movements from SIMD regions to global memory are required due to the fact that the qubits that do not participate in a gate operation applied on a SIMD region must be moved to the global memory, as gate operations will affect all qubits present in the SIMD region. In summary, a quantum architecture must minimize the amount of qubit (data) movement in order to reduce the overall probability of qubit errors as a result of qubit state collapse. In addition, the qubit movement can also incur performance overheads due to non-zero communication latencies (see next).

(V) Communication Cost. Qubit (Data) Movement can also cause loss in performance (QOPC) due to the cost of communication (in terms of cycles) involved in bringing the essential qubits from global memory to the correct SIMD operating region, as well moving qubits from different SIMD regions back to the global memory. A realistic communication cost of moving a qubit between two regions is four cycles [9]. Therefore, a qubit operation (one cycle) which requires a qubit to be moved from some other region to the correct SIMD region in which a quantum operation is to be applied will take five cycles to complete. The time cycles spent waiting for qubit states to be transported will increase the total cycle requirements of the quantum program.

TABLE I: An Illustrative example showing the step by step logic gate operation scheduling of Toffoli quantum algorithm on an architecture with only one SIMD operating region (SIMD1). The average QOPC, P/C and the total number of moves are shown in bold. The working is shown for three qubits (a0, a1, a2).

Time	Global	SIMD1	QOPC	P/C	Moves
1	a1, a2	H(a0)	1	0.5	1
2	a0, a2	$T^\dagger$(a1)	1	0.5	2
3	a1	T(a2), T(a0)	2	1	3
4	a0	CNOT(a2, a1)	2	1	2
5	a2	CNOT(a1, a0)	2	1	2
6	a0, a2	$T^\dagger$(a1)	1	0.5	1
7	a1	CNOT(a0, a2)	2	1	3
8	a0	CNOT(a1,a2)	2	1	2
9	a0	$T^\dagger$(a1), $T^\dagger$(a2)	2	1	0
10	a1, a2	T(a0)	1	0.5	3
11	a2	CNOT(a1,a0)	2	1	1
12	a0, a2	S(a1)	1	0.5	1
13	a1	CNOT(a0,a2)	2	1	3
14	a1, a2	H(a0)	1	0.5	1
15	a0	CNOT(a2,a1)	2	1	3
			1.6	**0.8**	**28**

B. Understanding Metrics via an Illustrative Example

In this section, we illustrate the previously described metrics using Toffoli quantum logic gate, also known as the CCNOT gate (controlled-controlled-NOT gate) [18]. We consider two scenarios in which the algorithm is mapped to two different quantum architectures. Table I shows the first scenario, in which the Toffoli gate is mapped to a SIMD architecture consisting of only one SIMD region ($k = 1$) with a capacity of two qubits ($d = 2$). As the width of the single SIMD region is two qubits, the maximum number of qubits that can be operated on during the same cycle is also two. For the sake of brevity of the schedule, we assume that the cost of qubit communication is zero. The Toffoli gate is constructed by five different quantum logic gates: H gate, T gate, $T^\dagger$ gate, CNOT gate, and S gate. The exact functionality of these logic gates is described in the technical report [13]. As evident by the scheduling diagram (Table I), the H, T, $T^\dagger$ and S gates take only one qubit as an input, while the CNOT requires two qubits as input. In Table I, for each time-step (or cycle) we show: a) the qubits stored in global memory, b) the current logic gate operation at the SIMD operating region, c) value of QOPC (quantum operations per cycle), d) value of Performance/Capacity (P/C), and e) amount of qubit movement (moves) per cycle. We assume that all qubits required by the Toffoli quantum algorithm are initially kept in the global memory.

At time-step 1, H gate needs to be applied to only qubit a0. Therefore, the other qubits (a1 and a2) remain shielded in the global memory. The qubit movement of a0 is considered as one move. As H gate is applied to only one of the qubit in the first time step, the value of qubit operation is also one leading to QOPC of 1. The P/C value indicates how much of the resources are wasted when the particular operation is operated. As the maximum capacity of this architecture is 2 ($k = 1$ x $d = 2$), the maximum possible value of P/C is 1 if both qubits of the region are operated in the same cycle of the region leading to two qubit operations. At time step 1, this value is 0.5 (1/2). At time step 2, $T^\dagger$ needs to be applied to qubit a1. In order to do so, two moves are necessary. First, we need to move the existing qubit (a0) in the SIMD region. This step is necessary because without that $T^\dagger$ would also apply to this qubit (a0) even if is not legal as per the program order. The second step is to move qubit (a1) to the SIMD region. Therefore, the global memory contains two qubits: a0 and a2. As only one qubit operation is completed at time-step 2, QOPC is 1. At time step 3, two separate T gates are applied to a2 and a0. As the gate operation is the same across both the qubits, both T gates can be applied at the same time in the SIMD operating region. However, three moves will be necessary: one to move a1 back to global memory and two moves related to bring a0 and a2 to the operating region. As we observe that, because of only one SIMD region, there are many back and forth qubits moves which can lead to high qubit error rate and also cause performance loss (not shown) because of the involved communication cost.

Table II demonstrates the effect on different metrics by mapping the same Toffoli quantum algorithm to a SIMD architecture with two operating regions. If no operation is applied to a qubit in a particular SIMD region, the qubit is only shown to represent that it is currently shielded in that SIMD region. Due to increase in the number of regions (thereby supporting quantum logic gate (instruction)-level parallelism) two quantum logic gate operations can be executed simultaneously, whenever possible (see Table II). Two observations are in order in comparison to the previous case when the Toffoli algorithm was mapped to only one SIMD operating region. First, we observe that on average more qubits operations are executed per cycle because of the increased parallelism. However, the P/C is reduced, which demonstrates that the increase in QOPC is not proportional to the increase in architectural resources (i.e., in terms of capacity). Second, we observe a decrease in the total number of qubit moves. This reduction seems to be in contrast to the classical SIMD computing which usually sees an increase in data movement with the number of cores. This key difference between classical and quantum computing is a result of the previously discussed intra-region parallelism. Quantum multi-SIMD computing must move qubits between the various SIMD operating regions, however, when a quantum logic gate is applied to a SIMD region as a microwave pulse, it will effect all qubits within the region at that cycle. Therefore, in order to shield a qubit from some quantum operation, it must be moved into either the global memory or another idle SIMD region. The sensitivity of qubit movement to both the k and d values of an architecture presents an optimization problem for designing a quantum computing architecture. We discuss the related trade-offs in Section V.

IV. Evaluation Methodology

In order to analyze large-scale quantum algorithms that cannot be scheduled by hand, we utilize a number of previously contributed software packages. This includes both a quantum programming language used to write quantum algorithms, as well as a quantum algorithm compiler. Scaffold is a quantum programming language developed by the Scaffold Compiler Working Group [18]. Scaffold is very similar to C, with the included support for quantum information, data structures, and quantum logic gates. Quantum algorithms and programs can be written in Scaffold code, then compiled by the Scaffold compiler ScaffCC [7]. This framework maps quantum algorithms onto targeted multi-SIMD hardware with the number of SIMD regions k and SIMD region qubit capacity d. ScaffCC is capable of executing many scheduling algorithms. For our benchmark test analysis, we use the Standard Scheduler (SS) included in the ScaffCC package. This scheduler, as well as other scheduling algorithms available in the package, are described in detail in previous work [7], [9], [18]. Once the algorithm has been mapped to a specified architecture, ScaffCC reports various metrics concerning this algorithm's performance, such as resource estimation (number of qubits and quantum operation logic gates), qubit operations (the total number of instances that a qubit is operated on), qubit movement, and the number of total time cycles required for the quantum algorithm runtime [7], [9], [18], [20]. These metrics can be used to analyze a system's performance for various quantum benchmark tests.

ScaffCC analysis metrics can be used to report our performance metrics of QOPC, qubit moves, and performance per area as discussed in the previous section. As our results demonstrate, these three performance metrics are highly dependent on both the k and d parameters of the multi-SIMD architecture. In order to analyze the performance of the different quantum benchmarks, we schedule benchmarks while evaluating through a range of k and d values. The ranges of these parameters are $k = [1, 2, 4, 6, 8]$ and $d = [2, 4, 16, 32, 64]$, therefore we analyze the quantum benchmarks for 25 different multi-SIMD architectures. Each quantum benchmark is defined by a constant number of qubit operations that is independent of the quantum computing architecture. For each combination of k and d parameters, we record the reported total time cycle count as well as the number of qubit moves. Using these numbers, we can calculate the average QOPC and Performance per area metrics for each k and d architecture combination.

In order to look at the time-wise results for the quantum benchmarks, we perform a full schedule regression via ScaffCC, which reports a detailed communication-free scheduling of the quantum algorithm. These schedules list, for each cycle, which quantum logic gates and qubit movements that occur, as well as the location of all qubits within the system. These time-wise results demonstrate how average QOPC and qubit movement intensity vary over time for each quantum benchmark, giving further insight into the specific system

TABLE II: An Illustrative example showing the step by step logic gate operation scheduling of Toffoli quantum algorithm on an architecture with two SIMD operating regions (SIMD1 and SIMD2). The average QOPC, P/C and the total number of moves are shown in bold. The working is shown for three qubits (a0, a1, a2).

Time	Global	SIMD1	SIMD2	QOPC	P/C	Moves
1	a2	H(a0)	$T^\dagger$(a1)	2	0.5	2
2		T(a0), T(a2)	a1	2	0.5	1
3	a0	CNOT(a2,a1)		2	0.5	2
4	a2	CNOT(a1,a0)		2	0.5	2
5		$T^\dagger$(a1)	CNOT(a0,a2)	2	0.75	2
6		CNOT(a1,a2)	T(a0)	3	0.75	1
7		$T^\dagger$(a1), $T^\dagger$(a2)	a0	2	0.5	0
8	a2	CNOT(a1,a0)		2	0.5	2
9		S(a1)	CNOT(a0, a2)	3	0.75	2
10		H(a0)	CNOT(a2,a1)	3	0.75	2
				2.4	**0.6**	**16**

requirements (k and d parameters) demanded by different quantum benchmarks.

V. Experimental Results and Analysis

In this section, we analyze the trade-offs and relationships between different metrics previously described in Section III. We specifically focus on performance (QOPC), performance/capacity, qubit movement (moves) and communication cost. We have evaluated these metrics across eight real-world quantum workloads on 25 different choices of SIMD quantum architectures. These architectures have varying degrees of capabilities to support quantum logic gate (instruction)-level parallelism (via different regions) and qubit (data)-level parallelism (via capacity or width of each region). All the presented data is normalized to that of the metrics obtained on an SIMD quantum architecture consisting of one region (k = 1) with a capacity of two qubits (d = 2), unless otherwise specified. We choose this baseline because this is the smallest-sized architecture that is able to execute all gate operations in the considered quantum algorithms. Note that an architecture with d = 1 would not be able to execute operations such as CNOT. Further, these results assume that qubit movement only incurs error overheads and not runtime overheads. This assumption helps us to focus on the trends in the considered evaluation metrics while decoupling the effects of the communication costs. We will also consider runtime overheads in Section V-D.

A. Performance (QOPC) Analysis

Figure 2 shows the trends in QOPC with different SIMD architecture choices for different quantum algorithms. The x-axis lists the choices of k for the architecture on which the corresponding algorithm is executed. We observe that the trends in QOPC are different across different algorithms. On one hand, where algorithms such as BWT, QFT, and SR do not benefit from more than two regions, on the other hand, algorithms such as CN and IM scale well with the number of regions. We find that the overall effect of d on QOPC is less prominent than the effect of k. However, at smaller k values ($k = 1$), the effect of d is more noticeable. This implies that a system with a small k value can still achieve an increase in QOPC by increasing the capacity d of the SIMD operating regions. However, at high k values ($k = 8$), all QOPC lines overlap, suggesting the low impact of d on QOPC after certain k value.

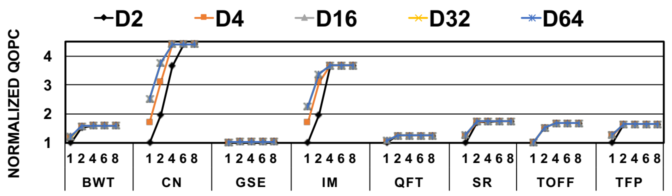

Fig. 2: Trends observed in Performance (QOPC) with different SIMD architecture choices.

For better understanding, Figure 3 shows the QOPC of different algorithms with three different SIMD architectures each with the same total qubit capacity of 32. We clearly see that the configuration with the highest number of regions (8×4) has the highest value of QOPC across all the evaluated algorithms. We also plot the values of QOPC over time for QFT application to understand if the effect of k and d varies over time. Figure 4 shows the changes in QOPC for QFT for two SIMD architectures with different number of regions but each with the same qubit capacity. We observe that for most of the time windows, the architecture with higher number of regions (2×16) has a higher or the same value as that of the other architecture choice with a lower number of regions (1×16). On the other hand, Figure 5 shows the changes in QOPC for QFT for two SIMD architectures with the same number of regions but each with different qubit capacity. We find that higher value of d has not much impact after time window number 17. However, before that, it has mixed trends. In the beginning of the execution, the architecture with higher d value also leads to higher capacity and hence the reason for very high speedup. However, because of such speedup it finishes the *compute heavy* instructions of that particular phase quickly and is left with instructions with low qubit utilization. On the other hand, the architecture with lower d has more steady QOPC for that phase but on average has low QOPC to the other architecture with a higher d value.

Conclusion. The key idea taken away from this analysis is that the number of regions (k) has much higher positive

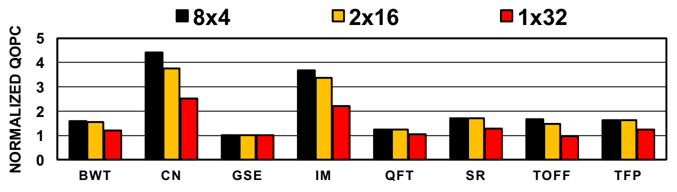

Fig. 3: Trends observed in Performance (QOPC) with different SIMD architecture choices each with the same qubit capacity (iso-capacity).

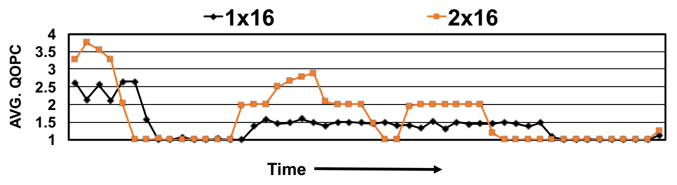

Fig. 4: Trends observed in Performance (QOPC) *over time* for QFT for two SIMD architectures with different number of regions but each with the same qubit capacity.

impact on QOPC compared to their capacity (d). Therefore, it is important to leverage quantum logic gate (instruction)-level or inter-region parallelism in quantum applications for better performance.

B. Performance/Capacity Analysis

As per our previous analysis, the average QOPC increases or remains the same with increase in total qubit capacity of the architecture. However, supporting higher qubit capacity architecture is very expensive in terms of cost, area, power etc. Therefore, it is imperative to consider performance in conjunction with the capacity of the architecture so that the architecture is also utilized well. In this context, Figure 6 shows the trends observed for Performance/Capacity (P/C) metric across different algorithms.

We observe that for many algorithms applications (except CN and IM), the value of $d = 2$ is fairly a good choice to optimize for P/C. However, for the effect of k on P/C is very significant *only* at smaller d values (less than or equal to 16). Consequently, the value of k needs to be more carefully selected when the value of d is low because of the sharp decline in P/C after a certain k value. For example, algorithms such as GSE and QFT experience the best P/C values at $k = 1$; BWT, SR, TOFF and TFP at $k = 2$; and CN and IM at $k = 4$.

Conclusion. The key insight from this study is that both number of regions (k) and size of each region (d) can have impact on P/C. It is because of the fact that increasing k beyond a particular value (which is different for different applications) can lead to sharp decrease in P/C if a low (d) value is chosen. Therefore, in order to design an architecture that is both fast but efficient (highly utilized), one must consider both k and d carefully.

C. Qubit (Data) Movement Analysis

Figure 7 shows the trends observed in Qubit (Data) Movement with different SIMD architecture choices. We observe that, in general, the qubit movement decreases with the increase in k value. This is in direct agreement with our discussions in Section III. However, the QFT benchmark shows a slight increase in total qubit movement between $k = 2$ and $k = 4$. This effect is likely because of the scheduler that optimizes for QOPC instead of qubit movement. The effect of d varies across algorithms. At low k values, increasing d decreases the number of moves because fewer qubits are required to be shielded in the global memory (Section III) However, at high k values, increasing d will either have no effect or actually increase the number of moves. This effect is again because of the scheduler that optimizes for QOPC instead of qubit movement.

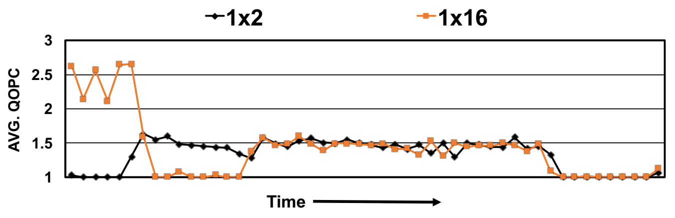

Fig. 5: Trends observed in Performance (QOPC) *over time* for QFT for two SIMD architectures with the same number of regions but each with different qubit capacity.

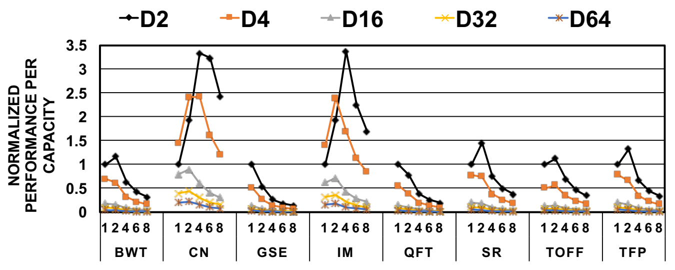

Fig. 6: Trends observed in Performance/Capacity (P/C) with different SIMD architecture choices.

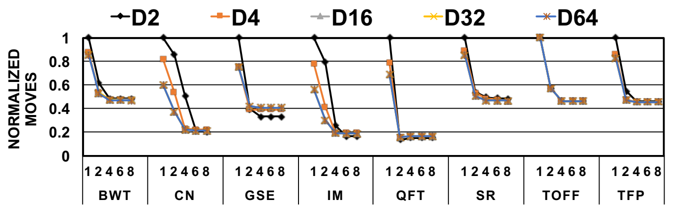

Fig. 7: Trends observed in Qubit (Data) Movement with different SIMD architecture choices

For a deeper understanding of the above trends, we looked at the trends in qubit movement in iso-capacity architectures (as shown in Figure 8). We clearly observe that the architecture with the lowest k value has the highest moves. It is because of the fact that in order to operate different logic gate operations in fewer regions, more qubits need to be moved to global memory so that they are not affected by undesirable gate operations. We also looked at the qubit movement trend over time for QFT. In Figure 9, the value of d is kept constant across both architecture options. We observe again that for most of the time windows, the $2{\times}16$ configuration leads to fewer qubits moves. In Figure 10, the value of k is kept constant across both architecture options. As expected, we observe that the architecture with higher total qubit capacity ($1{\times}16$) leads to fewer qubits moves for almost all time windows.

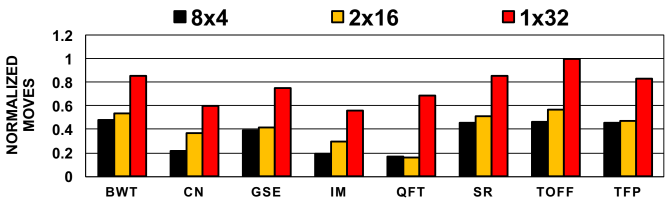

Fig. 8: Trends observed in Qubit (Data) Movement with different SIMD architecture choices each with the same qubit capacity (iso-capacity).

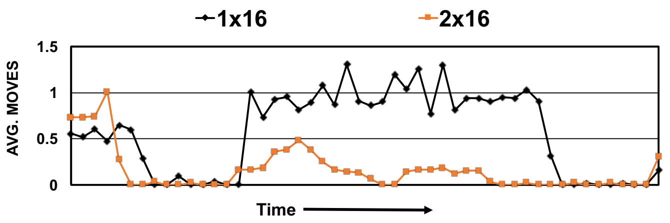

Fig. 9: Trends observed in Qubit (Data) Movement *over time* for QFT for two SIMD architectures with different number of regions but each with the same qubit capacity.

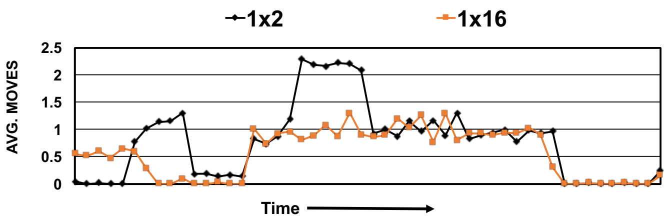

Fig. 10: Trends observed in Qubit (Data) Movement *over time* for QFT for two SIMD architectures with the same number of regions but each with different qubit capacity.

Conclusion. The key insight from this study is that, in general, lower total qubit capacity (k x d) architectures tend to have higher qubit movement because the qubits that do not participate in the logic gate operation need to be moved to global memory often in order to be shielded from undesirable quantum logic gate operations.

D. Communication Cost Analysis

The previous work assumed that the communication overhead for quantum teleportation is 4 cycles [9]. However, this overhead is likely to be changed with the research advances in the quantum communication technology. In this section, we evaluate the impact of four different communication latencies (cl) on performance for two different quantum algorithms: GSE (Figure 11 and QFT (Figure 12). The x-axis shows the trends for various cl values, where $cl = 0$ means no communication cost and $cl = 4$ means each move takes 4 cycles. We observe that an increase in cl results in a decrease in QOPC for both benchmarks. This result is intuitive because an increase in the number of cycles required for qubit movement will increase the overall cycle requirement, therefore decreasing the average QOPC.

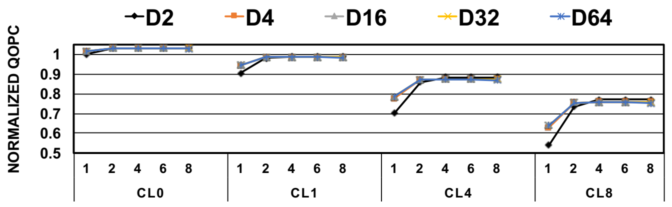

Fig. 11: Effect of communication cost on the performance of GSE.

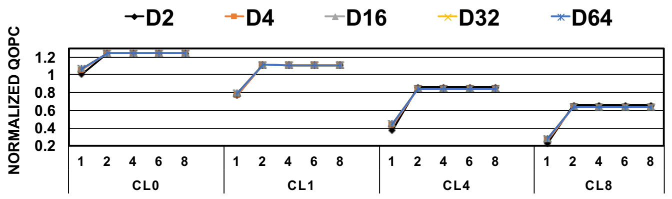

Fig. 12: Effect of communication cost on the performance of QFT.

Conclusion. Qubit movement is one of the primary causes of decoherence, which collapses the qubit superposition and lead to quantum state collapse (computation error). Therefore, the number of moves as well as the time for which the qubits are under motion should always be minimized in any given algorithm.

VI. Related Work

To the best of our knowledge, this is the first work that performs a deep characterization of various quantum algorithms on a variety of SIMD architectures and show that there is a significant difficulty in optimizing different architectural parameters in order to achieve high performance and efficiency. In this section, we briefly describe the previous works closest to ours.

Quantum Computing. Quantum algorithms have been formulated and written by a number of authors [5], [7], [10], [14]–[17], [19], [21], [22]. Previous work demonstrates the use of QPE (quantum phase estimation) in various quantum benchmarks, as well as various methods used to manage the expensive computational requirements of QPE [23]. In this paper, we leverage the previously proposed quantum benchmarks to evaluate different designs of quantum computers.

Quantum Compiler Framework and Tools. Previous work on quantum computing has focused on both high-level programming languages [18] and quantum program compilers and analysis [7]. In this context, the ScaffCC framework was proposed to analyze various large-scale quantum programs written in the Scaffold programming language [7]. This framework is able to generate various quantum code metrics and schedules of different operations dependent on the simulated quantum architecture.

Quantum Architectures. The proposition of Multi-SIMD quantum computing architecture design is based on previous quantum computing architecture and quantum memory hierarchy research [8], [11], [24]–[29]. Heckey et al. [9] proposed an ion trap multi-SIMD architecture composed of k SIMD operating regions each with a width or qubit capacity of d [8], [9] to execute many quantum applications written in Scaffold programming language. These programs are scalable and can be mapped to different SIMD architecture with different architectural properties (k, d, communication latency weights [6], [9], [12], [30]). In this work, we leverage the aforementioned tools and framework to collectively analyze multiple quantum architectural parameters on different evaluation metrics.

VII. Conclusions

Although scalable quantum computers that can process large amounts of qubits are not yet available, there have been recent research advances on the software and scheduling side of quantum computing. Our analysis of the simulation of quantum benchmarks on multi-SIMD architectures provides insights into the design of a quantum architecture that will be necessary for a scalable quantum computer. In our work, we show that there is no single most efficient quantum computing architecture for all quantum algorithms or for all the considered evaluation metrics, which is a result of the varied architectural needs of the different real-world quantum algorithms. In order to develop an optimized quantum architecture, one should consider the effects of different architectural parameters such as the number and size of SIMD regions, and communication overheads between SIMD regions and the global memory.

Acknowledgments

We thank the anonymous reviewers for their feedback. Special thanks to Ali JavadiAbhari for answering our questions on the ScaffCC framework and Irina Novikova for providing insights into the problems related to quantum computing. This work is supported by a start-up grant from the College of William and Mary.

References

[1] "IBM Makes Quantum Computing Available on IBM Cloud to Accelerate Innovation," http://www-03.ibm.com/press/us/en/pressrelease/49661.wss.

[2] "Intel Invests 50 Million USD to Advance Quantum Computing," https://newsroom.intel.com/news-releases/intel-invests-us50-million-to-advance-quantum-computing/.

[3] T. Simonite, "Googles Quantum Dream Machine," https://www.technologyreview.com/s/544421/googles-quantum-dream-machine/.

[4] B. Darrow, "Microsoft Simulator Brings Quantum Computing One Step Closer to the Masses," http://fortune.com/2015/11/13/microsoft-quantum-computing-simulator/.

[5] M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz, "A fault tolerant, area efficient architecture for Shor's factoring algorithm," in *ISCA*, 2009.

[6] T. S. Metodi, A. I. Faruque, and F. T. Chong, *Quantum Computing for Computer Architects, Second Edition*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2011.

[7] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "ScaffCC: a framework for compilation and analysis of quantum computing programs," in *CF*, 2014.

[8] D. D. Thaker, T. S. Metodi, and F. T. Chong, "A Realizable Distributed Ion-Trap Quantum Computer," in *HiPC*, 2006.

[9] J. Heckey, S. Patil, A. JavadiAbhari, A. Holmes, D. Kudrow, K. R. Brown, D. Franklin, F. T. Chong, and M. Martonosi, "Compiler Management of Communication and Parallelism for Quantum Computation," in *ASPLOS*, 2015.

[10] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by a quantum walk," in *STOC*, 2003.

[11] E. Chi, S. A. Lyon, and M. Martonosi, "Tailoring Quantum Architectures to Implementation Style: A Quantum Computer for Mobile and Persistent Qubits," in *ISCA*, 2007.

[12] D. Copsey, M. Oskin, T. S. Metodi, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "The effect of communication costs in solid-state quantum computing architectures," in *SPAA*, 2003.

[13] A. Muthukrishnan, "Classical and quantum logic gates: An introduction to quantum computing," in *Quantum Information Seminar*. RCQI, 1999.

[14] S. Hallgren, "Fast quantum algorithms for computing the unit group and class group of a number field," in *Symposium on Theory of Computing*. ACM, 2005.

[15] A. A.-G. James D. Whitfield, Jacob Biamonte, "Simulation of electronic structure hamiltonians using quantum computers," in *Molecular Physics*, 2011.

[16] N. I. Akira Matsuo, Keisuke Fujii, "A quantum algorithm for additive approximation of ising partition functions," in *Physical Review Letters*, 2014.

[17] D. G. C. Yaakov S. Weinstein, Seth Lloyd, "Implementation of the quantum fourier transform," in *Physical Review Letters*, 1999.

[18] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "Scaffold: Quantum Programming Language," in *Princeton University Technical Report*, 2012.

[19] M. S. Frederic Magniez, Miklos Santha, "Quantum algorithms for the triangle problem," in *Journal on Computing*. SIAM, 2005.

[20] M. Suchara, J. Kubiatowicz, A. I. Faruque, F. T. Chong, C. Lai, and G. Paz, "QuRE: The Quantum Resource Estimator toolbox," in *ICCD*, 2013.

[21] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Foundations of Computer Science*. IEEE, 1994.

[22] D. Kudrow, K. Bier, Z. Deng, D. Franklin, Y. Tomita, K. R. Brown, and F. T. Chong, "Quantum rotations: a case study in static and dynamic machine-code generation for quantum computers," in *ISCA*, 2013.

[23] S. Patil, A. JavadiAbhari, C. Chiang, J. Heckey, M. Martonosi, and F. T. Chong, "Characterizing the performance effect of trials and rotations in applications that use Quantum Phase Estimation," in *IISWC*, 2014.

[24] T. S. Metodi, D. D. Thaker, A. W. Cross, I. L. Chuang, and F. T. Chong, "High-level interconnect model for the quantum logic array architecture," in *JETC*, 2008.

[25] 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," in *ISCA*, 2006.

[26] N. Isailovic, M. Whitney, Y. Patel, J. Kubiatowicz, D. Copsey, F. T. Chong, I. L. Chuang, and M. Oskin, "Datapath and control for quantum wires," in *TACO*, 2004.

[27] M. Oskin, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "Building Quantum Wires: The Long and the Short of It," in *ISCA*, 2003.

[28] M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz, "Automated generation of layout and control for quantum circuits," in *CF*, 2007.

[29] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, "Interconnection Networks for Scalable Quantum Computers," in *ISCA*, 2006.

[30] N. Isailovic, M. Whitney, Y. Patel, and J. Kubiatowicz, "Running a Quantum Circuit at the Speed of Data," in *ISCA*, 2008.