# Architecture Optimization for Quantum Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Physics from the College of William and Mary.

by


## Robert Risque


Advisor: Adwait Jog

Senior Research Coordinator: Henry Krakauer

Date: May 8th, 2016

## Introduction

Promising research in quantum computing on both the hardware and software sides have increased speculation of physically realizable quantum computers in the near future. While previous work has diverged between computer scientists and physicists, recent advances have required the cooperation between both sides. Current problems are focusing on how to efficiently map quantum computing algorithms onto quantum computing architectures once physically realized quantum computers become available. This consideration of what kind of resources quantum algorithms will require is one of the major driving factors behind current quantum computing developments.

Currently, the most prevalent quantum computer architecture design will employ ion trap technology for qubits and quantum operations. These trapped ion quantum computers will utilize specialized multi-SIMD architectures and scheduling algorithms which will schedule quantum algorithms most efficiently to quantum computing architectures in order to increase speedup while also reducing computation errors due to qubit communication and quantum state decoherence. The following paper presents an introduction to quantum computing and current quantum programming language compilers, and how the development of specialized architectures and schedulers can significantly increase quantum computation efficiency.

## Quantum Computing

Quantum computing boasts impressive speedups at specific tasks when compared to classical computers. These tasks include database search and integer factoring algorithms; processes that may take a classical super computing cluster many years to accomplish. The core difference between quantum computing and classical computing lies in the quantum mechanical properties of qubits. A qubit, or *quantum-bit*, is the medium of information storage in a quantum computer. Analogous to classical bits, a qubit is a two state system used to represent binary information. However, while classical bits can only exist in either the 0 or 1 state at any given instance, a qubit can exist in both states simultaneously. At any point in time, a qubit will exist in some superposition of the 0 and 1 binary states. A qubit state $\Psi$[1] can be represented as:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $|\alpha|^2$ and $|\beta|^2$ are equal to the probability of the qubit being in state 0 or 1, 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

[1] Qubit state $\Psi$ written in Dirac notation.

classical binary string that makes up the superposition, individually. Qubits have other special properties as a quantum mechanical system that require attention when dealing with quantum computing architectures. The exact state of a quantum system cannot be copied from one qubit to another without destroying the original state. 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. Because of the delicacy of quantum superposition in regards to decoherence, however, interaction between qubits and their environment must be minimized in order to reduce computation errors. The primary source of qubit decoherence is ballistic motion required for communication. Thus, as will be discussed in further sections, schedulers must optimize their schedules to require the smallest amount of qubit movement.

## Scaffold

Scaffold is a quantum programming language developed by the Scaffold Compiler Working Group, a joint project between Princeton, UCSB, and IBM. Scaffold is very similar to the C language, with the included support for quantum information, data structures, and operations. Quantum algorithms and programs can be written in Scaffold code, then compiled by the Scaffold compiler ScaffCC. This compiler maps algorithms onto targeted hardware that can be specified in the ScaffCC options. Once the algorithm has been mapped and scheduled, ScaffCC will return various metrics on this algorithms theoretical runtime, such as resource estimation (number of qubits and gates, qubit operations, and cycles required for computation). These metrics can be used to analyze a system's performance for various quantum benchmark tests, discussed further in the Optimization section.

## Architecture

The most prominent physically realizable quantum computer is the trapped ion quantum computer. Trapped ions are promising qubit candidates because of their relatively long decoherence times (compared to their operation time) and the ability to confine ions to a small point in space via ion traps (as opposed to photonic qubits, a close runner-up, which cannot be trapped as easily). The two states of the qubits, 0 and 1, are 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.

The actual computations performed on qubits will take place in SIMD regions (Single-Instruction-Multiple-Data). These SIMD operating regions are capable of holding multiple qubits at a time. A quantum logic gate in the form of a microwave pulse (single instruction) will be applied to the

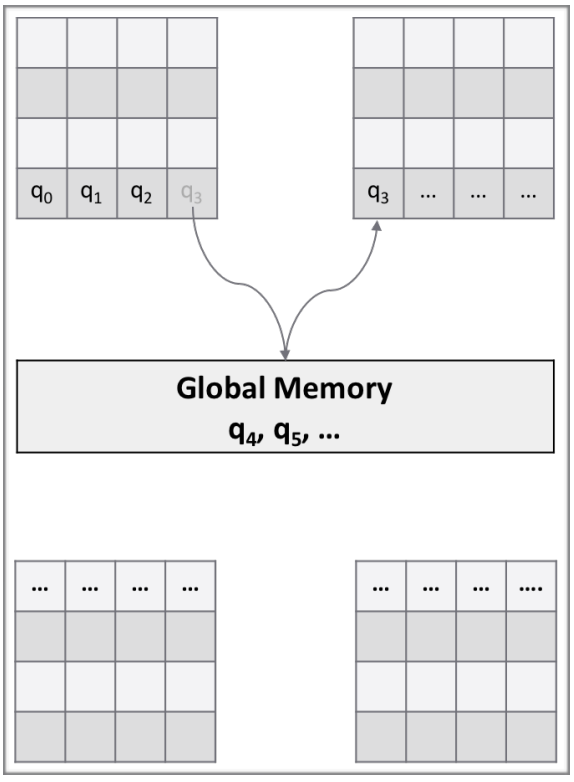
entire SIMD region, affecting all qubits stored within the region (multiple data). Qubits that require measurement or storage which is shielded from the microwave pulse are moved to the global memory via quantum teleportation.

These SIMD architecture models are defined by multiple parameters. *K* is the number of SIMD operating regions and *D* is the qubit capacity of one of these SIMD regions. Each SIMD region can undergo one quantum operation per cycle, where one cycle is defined as the amount of time required for the system to apply the longest quantum logic gate. This means that the max number of operations that can be performed per cycle is the product of *K* and *D*. The parameter *W* is defined as the cycle weight applied towards communication.

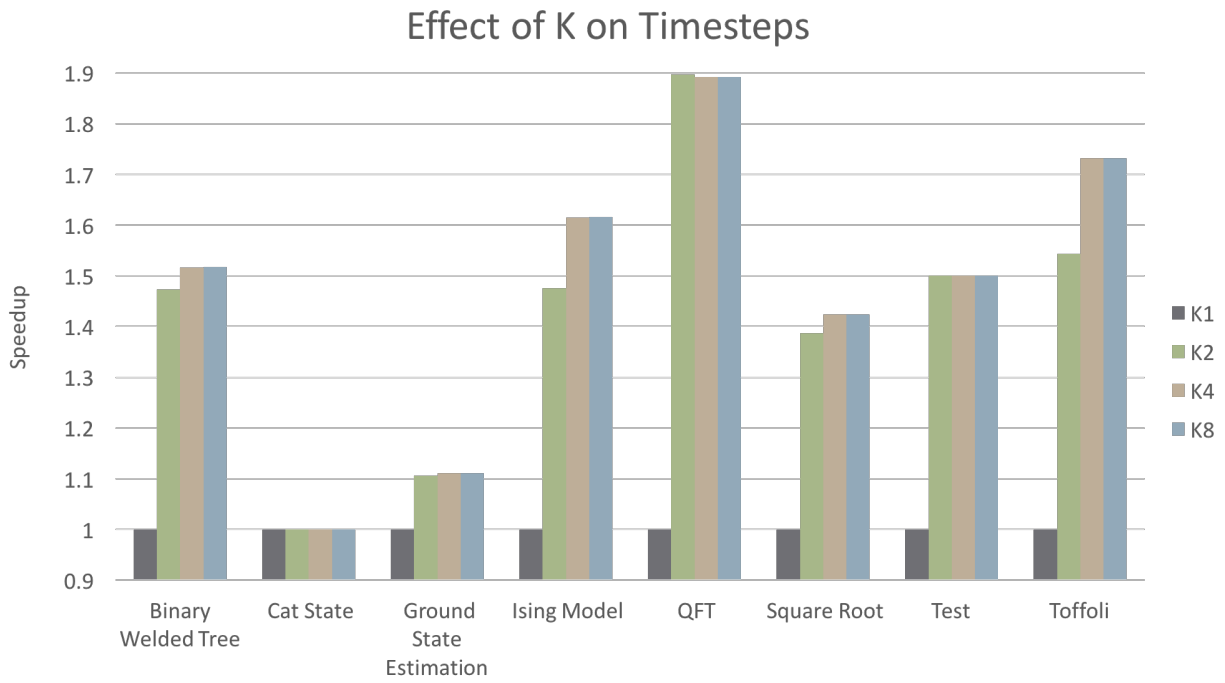
Our results discussed in the following section show that this proposed SIMD architecture brings further parallelism to quantum computations, allowing for greater benchmark speedup. However, there exist specific *K, D,* and *W* parameter combinations which will optimize quantum benchmarks for greatest efficiency.

## Optimization

In order to optimize computation efficiency, we must look at different metrics of the quantum algorithm schedules. The first metric is IPC (instructions-per-cycle). IPC is defined as the number of qubit operations per cycle. As mentioned previously, the max IPC is equal to *KxD*. The next metric, Utilization, is concerned with how much of the computational resources are being used per second. Utilization is defined as *IPC/max capacity*. These two metrics are dependent on both K and D parameters, making efficient scheduling a multivariable optimization problem. IPC is important when considering an algorithm's speedup, as it is directly related to the number of cycles required to run an quantum program. Utilization allows us to determine how much of the architecture is actually being used at any point of time while running a program. This allows us to consider turning off certain SIMD regions that are unnecessary for a given quantum program in order to save power that is usually required to run an ion trap. An Efficiency metric is defined as (IPC**$\alpha$)(Utilization**$\beta$). Both $\alpha$ and $\beta$ are tuning parameters which can prioritize either




Example quantum architecture with *K=4* SIMD regions with qubit capacity *D=16*.




The effect of *K* on benchmark speedup. Speedup is defined as the percent decrease in time cycles when compared to the base case (*K=1*).

IPC or Utilization when analyzing an algorithms performance[2].

The different quantum benchmarks are listed below. These algorithms vary in size and present different reactions to various *K* and *D* combinations.

- Binary Welded Tree
- Cat State
- Ground State Estimation
- Ising Model
- Shor's Algorithm
- Quantum Fourier Transform
- Toffoli gate

These algorithms were compiled originally compiled with 4 different *K* values; 1, 2, 4, and 8. In this simulation, *D* was set to an arbitrarily high constant value of 1024. Increasing *K* from 1 to 2 always shows an impressive algorithm speedup[3]. However, the increase in speedup between *K=2* and *K=4* is smaller, and the difference between *K=4* and *K=8* is almost always negligible. This indicates that there is an optimal *K* value which ensures max IPC while not wasting resources and

[2] *Efficiency* is a working title for this metric. Depending on the available resources and time constraints for a quantum computation, one may wish to prioritize either fast computation speed or reduced resource requirements. One would then prioritize either IPC or Utilization, respectively, when optimizing for maximum efficiency.

[3] The Cat State benchmark does not see any speedup due to an increase in the number of SIMD regions, because its schedule is highly serial and therefore unable to be operated in parallel.

reducing the utilization of the system. The plateau that speedup reaches varies between benchmarks as well, suggesting that there is not one optimal *K* value to be used. A heterogenous architecture model will be discussed in the last section.

These graphs[4] show the combined effect of both *K* and *D* on Efficiency, which is a combined metric of both IPC and Utilization. In these examples, both $\alpha$ and $\beta$ are equal to 1.

As shown by these results, there are specific *K* and *D* values which maximize benchmark efficiency. These parameter combinations vary between benchmarks, however their exist general trends which can be exploited. Increasing *K* will always increase the benchmark IPC, to a point (plateau). Increasing *D* will also sometimes help IPC, however the effect of increased capacity is less than that of increasing *K*. Increasing either of these parameters will negatively impact SIMD region utilization.

## Communication Aware Scheduling

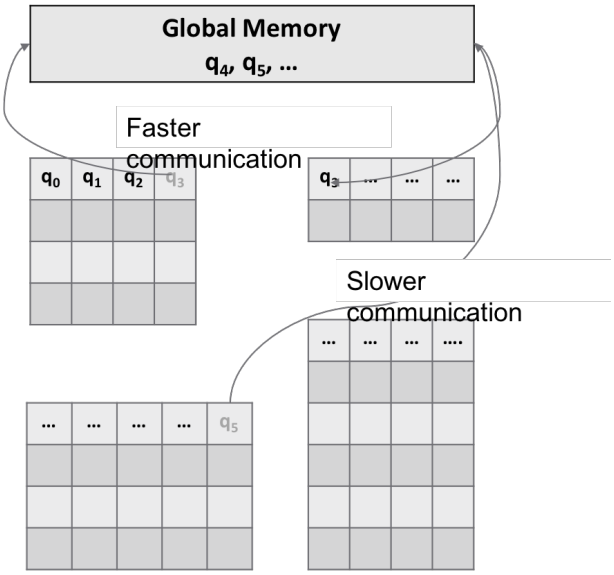
All previously reported results have ignored communication costs. However, as mentioned before, data transmission happens via either ballistic qubit movement or quantum teleportation, both of which require a non-trivial amount of cycles. The theorized cycle count for one quantum teleportation is 4 cycles. Standard scheduling does not account for this communication cost, which presents a few issues.

First, qubit movement (ballistic or quantum teleportation) is the primary cause of decoherence, which will collapse the qubit superposition and lead to quantum state collapse (computation error). Therefore, the number of moves in any given algorithm should always be minimized. Second, a scheduler that does not minimize the number of moves will greatly increase the cycle requirements for an algorithm. This communication latency can be masked by scheduling tasks in parallel, however failing to do so will cause the SIMD regions to be starved for qubits. Currently, increasing *K* and *D* values with the ScaffCC SS (Standard Scheduler) while including communication latencies will not always guarantee speedup. The SS will sometimes over utilize SIMD regions, reducing communication free cycle-count calculations but ultimately increasing the total cycle-count for realistic communication cycle-counts.

## NUMA Architecture

NUMA (Non-Uniform-Memory-Access) is a concept in classical computing which involves either sectioning the global memory of a system into multiple parts. This architectural design can also be applied to quantum computers. In previously proposed quantum computers, all SIMD regions must communicate qubits via quantum teleportation with a single global

[4] Graphs representing the effect of *K* and *D* on efficiency with $\alpha = \beta = 1$ are located on the last page.




Example section of a NUMA architecture with heterogenous *K* and *D* values. These specialized SIMD operating regions will have different communication weights *W* depending on their proximity to to the global memory. This image shows only a small section of what the entire architecture will be.

memory. However, it may actually be more beneficial to quantum computers to utilize non-uniform global memories. Smaller, more localized memories can then be placed in closer proximity to SIMD regions, thus reducing both communication latency and distance, and overall reducing the potential for computation errors due to quantum state decoherence.

There also exists motivation to explore non-uniform SIMD operating region setups. This type of architecture would use heterogenous *K* and *D* values for its SIMD regions, thus allowing a scheduler to map quantum algorithms onto specialized SIMD regions which optimize both IPC and resource utilization. The motivation for this comes from the fact that different benchmarks perform best when using different combinations of *K* and *D* parameters. A quantum computing architecture could accommodate various SIMD regions of different sizes and distances from global memory, utilizing only the regions which are best suited for a given quantum program.

## Conclusions

Scalable quantum computers will most likely be realized within the next few decades. Although the hardware side has a lot of work left, there has been significant work on the software and scheduling side of quantum computing. Analyzing the simulation of quantum benchmarks on multi-SIMD architectures provides valuable insights into what sort of hardware will be necessary for a quantum computer. Our results show that the most efficient quantum computing architecture will involve trapped-ion qubits transported between multiple global memories and specialized non-uniform SIMD operating regions.

Future work in this area will involve the creation of a new scheduling algorithm capable of considering heterogenous *K*, *D*, and *W* architecture parameters in order to most efficiently map quantum algorithms onto these targeted hardware models. The development of this type of scheduler will require time-wise analysis of quantum benchmarks in regards to IPC and utilization metrics.

## Related Work and References


1. J. Heckey, S. Patil, A. JavadiAbhari, A. Holmes, D. Kudrow, K. R. Brown, D. Franklin, F. T. Chong, and M. Martonosi, “Compiler Man- agement of Communication and Parallelism for Quantum Computation,” in *ASPLOS*, 2015.

2. 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.

3. 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

4. E. Chi, S. A. Lyon, and M. Martonosi, “Tailoring Quantum Architec- tures to Implementation Style: A Quantum Computer for Mobile and Persistent Qubits,” in *ISCA*, 2007.
5. M.Whitney,N.Isailovic,Y.Patel,andJ.Kubiatowic z,“Afaulttolerant, area efficient architecture for Shor’s factoring algorithm,” in *ISCA*, 2009. [6] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, “Interconnection Networks for Scalable Quantum Computers,” in *ISCA*, 2006.

6. M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz, “Automated generation of layout and control for quantum circuits,” in *CF*, 2007.

7. N. Isailovic, M. Whitney, Y. Patel, and J. Kubiatowicz, “Running a Quantum Circuit at the Speed of Data,” in *ISCA*, 2008.

8. 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.

9. M. Oskin, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, “Building Quantum Wires: The Long and the Short of It,” in *ISCA*, 2003.

10. 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.

11. D.D.Thaker,T.S.Metodi,A.W.Cross,I.L.Chuang, andF.T.Chong, “Quantum Memory Hierarchies: Efficient Designs to Match Available Parallelism in Quantum Computing,” in *ISCA*, 2006.

12. D. D. Thaker, T. S. Metodi, and F. T. Chong, “A Realizable Distributed Ion-Trap Quantum Computer,” in *HiPC*, 2006.

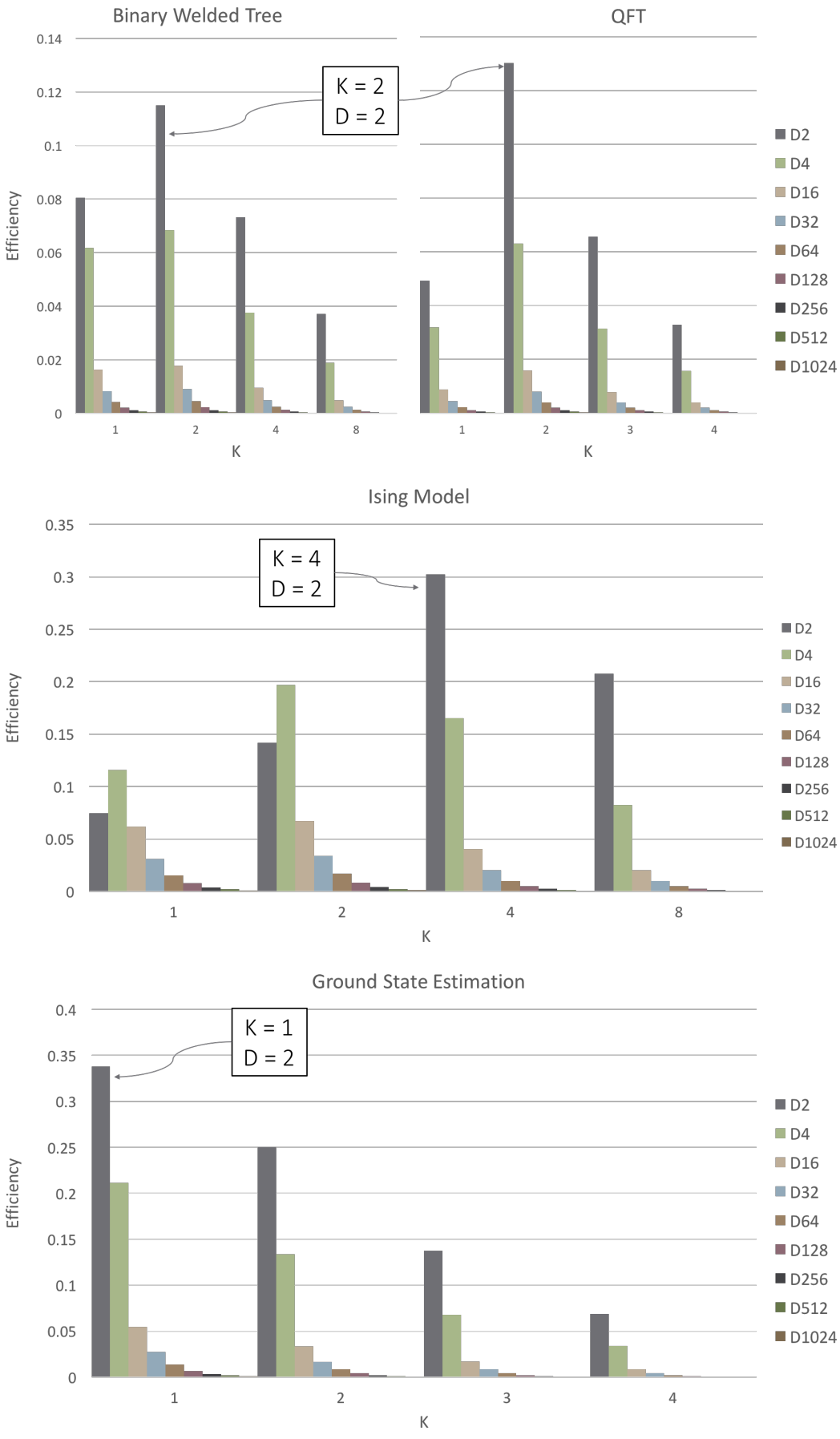
13. T. S. Metodi, D. D. Thaker, A. W. Cross, I. L. Chuang, and F. T. Chong, “High-level interconnect model for the quantum logic array architecture,” 2008.

14. 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.

15. 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.

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

## Sample Results




The effect of *K* and *D* on the efficiency of multiple benchmarks. Both the Binary Welded Tree and the QFT benchmarks experience optimal performance with the parameter combination of *K=2* and *D=2*. The Ising Model benchmark performs best with *K = 4* and *D = 2*, whereas the Ground State Estimation benchmark is best with *K=1* and *D=2*.