# High-Performance QuIDD-based Simulation of Quantum Circuits

George F. Viamontes, Igor L. Markov, and John P. Hayes
The University of Michigan
Advanced Computer Architecture Laboratory
Ann Arbor, MI 48109-2122, USA
{gviamont, imarkov, jhayes}@eecs.umich.edu

## Abstract

*Simulating quantum computation on a classical computer is a difficult problem. The matrices representing quantum gates, and vectors modeling qubit states grow exponentially with the number of qubits. It has been shown experimentally that the QuIDD (Quantum Information Decision Diagram) datastructure greatly facilitates simulations using memory and runtime that are* polynomial *in the number of qubits. In this paper, we present a complexity analysis which formally describes this class of matrices and vectors. We also present an improved implementation of QuIDDs which can simulate Grover's algorithm for quantum search with the asymptotic runtime complexity of an ideal quantum computer up to negligible overhead.*

## 1. Introduction

Richard Feynman observed in the 1980s that simulating quantum-mechanical processes on a *classical* computer seems to require super-polynomial memory and time [5]. For instance, a complex vector of size $2^n$ is needed to represent all the information in $n$ quantum bits, and modeling (simulating) the time evolution of the states calls for square matrices of size $2^{2n}$ [6]. Consequently, Feynman proposed *quantum computing* which uses the quantum mechanical states themselves to simulate quantum processes. A fundamental postulate of quantum mechanics dictates that these individual state vectors can be combined, via the tensor product, with other state vectors to represent multiple qubits in a composite state [6]. With composite states of qubits, quantum computers can operate directly on exponentially more data than a classical computer with a similar number of operations and information units.

In our previous work [10], we described the *Quantum Information Decision Diagram* (QuIDD) and several approaches for simulating quantum computation, including qubit-wise multiplication [7], the Heisenberg representation [2], and QDD [3]. We noted that the Heisenberg representation was incapable of simulation with a universal gate set and that QDD could only capture a limited set of qubit states. We showed experimentally that our QuIDD-based simulator QuIDDPro scales much better in both runtime and memory [10]. One may observe that QuIDDPro's runtime exhibited a complexity of $\Theta(1.66^n)$, while an ideal quantum computer requires $\Omega((\sqrt{2})^n)$ time. In trying to explain the performance gap by theoretical analysis, we discovered that QuIDDPro *should* run as fast as an ideal quantum computer in the simulations we performed. Indeed, a re-engineering of our implementation led to significantly improved runtimes and memory usage, as reported below.

## 2. Complexity Analysis

Below we prove that a significant subset of the QuIDD matrices and vectors used in quantum circuit simulation require runtime and memory that are *linear* in the number of qubits. Proofs of the theorems can be found in [9].

**Theorem 1** *Given QuIDDs* $\{Q_i\}_{i=1}^n$, *the tensor-product QuIDD* $\otimes_{i=1}^n Q_i$ *contains* $|In(Q_1)| + \Sigma_{i=2}^n |In(Q_i)||Term(\otimes_{j=1}^{i-1} Q_j) + |Term(\otimes_{i=1}^n Q_i)|$ *nodes.*[1]

It follows [9] that the number of nodes in the tensor product of QuIDDs grows linearly in $n$ if the number of terminals in $\otimes_{i=1} Q_i$ is constant; otherwise it grows exponentially in $n$. Thus, the growth depends on matrix entries because terminals of $A \otimes B$ are products of terminal values of $A$ by terminal values of $B$, and repeated products are merged. If all QuIDDs $Q_i$ have terminal values from a set $\Gamma$, the terminal values of $\otimes_{i=1} Q_i$ are products of elements from $\Gamma$.

Consider finite non-empty sets of complex numbers $\Gamma_1$ and $\Gamma_2$, and define their *all-pairs product* as $\{xy \mid x \in \Gamma_1, y \in \Gamma_2\}$. This operation is associative, making the set $\Gamma^m$ of *all m-element products* well-defined for $m > 0$. We then call a finite non-empty set $\Gamma \subset \mathbb{C}$ *persistent* iff for all $m > 0$, $|\Gamma^m|$ is constant. An important example is the set consisting of 0 and all $n$-th degree roots of unity $\mathbb{U}_n = \{e^{2\pi ik/n} | k = 0..n-1\}$. Every persistent set is either $c\mathbb{U}_n$ for some $n$ and $c \neq 0$, or $\{0\} \cup c\mathbb{U}_n$ [9].

[1] $|In(A)|$ denotes the number of internal nodes in $A$, while $|Term(A)|$ denotes the number of terminal nodes in $A$.

1530-1591/04 $20.00 (c) 2004 IEEE

**Theorem 2** *Given a persistent set* $\Gamma$ *and a constant C, consider n QuIDDs with at most C nodes each and terminal values from a persistent set* $\Gamma$*. The tensor product of those QuIDDs has* $O(n)$ *nodes and can be computed in* $O(n)$ *time.*

For $\Gamma = \{0\} \cup c\mathbb{U}$, the tensor product can be computed in linear time and memory for any equal superposition of *n* qubits, any sequence of *n* qubits in the computational basis states, *n*-qubit Pauli matrices, *n*-qubit Hadamard matrices, and other matrices and vectors with terminals in $\Gamma$. Since the time and memory complexity of matrix multiplication and measurement are polynomial in the size of the operand QuIDDs [1, 9], the above results suggest that there exists a polynomial-sized QuIDD representation of any *n*-qubit quantum circuit with a constant number of such gates.

## 3. Simulation of Grover's Algorithm

To demonstrate the power of QuIDDs, we used QuIDDPro to simulate Grover's algorithm [4], one of the two major quantum algorithms that have been discovered to date. Grover's algorithm searches for a subset of items in an unordered database of *N* items. Allowed selection criteria are black-box predicates, called oracles, that can be evaluated on any database record. The overall complexity analysis is performed by counting queries. In the classical domain, any algorithm for such an unordered search must query the predicate $\Omega(N)$ times. However, Grover's algorithm can perform the search with quantum query complexity $O(\sqrt{N})$.

**Theorem 3** *[9] The memory complexity of simulating Grover's algorithm using QuIDDs is polynomial in the size of the oracle QuIDD and the number of qubits.*

Theorem 3 implies that simulation with QuIDDs can offer *polynomial* memory complexity depending on the oracle. Empirically, only linear memory is required in many specific cases. This memory complexity is the same as an actual, ideal quantum computer.

Figures 1 and 2 describe simulations of Grover's algorithm with an oracle that searches for one item out of $2^n$. All experiments are performed on a 1.2GHz AMD Athlon with 1GB RAM running Linux. As shown, the runtime of the improved QuIDDPro is much closer to the lower bound $\propto (\sqrt{2})^n$ for an ideal quantum computer, while the original QuIDDPro has performance $\propto 1.66^n$. Similar improvements were observed for several other oracles [9].

## 4. Future Work

Representing certain quantum operators on a classical computer, even with QuIDDs, requires super-polynomial resources. Examples include the Hidden-Weighted Bit [11] oracle, the quantum Fourier transform and its inverse. Algorithmic improvements directed at specific useful operators could further improve QuIDDPro's performance. We are also studying the simulation of quantum noise and decoherence. Error simulation is the key to modeling realistic quantum-computational devices.

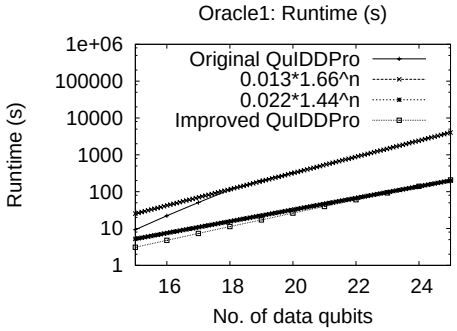


**Figure 1. Logscale runtime for the original [10] and improved QuIDDPro (oracle 1).**

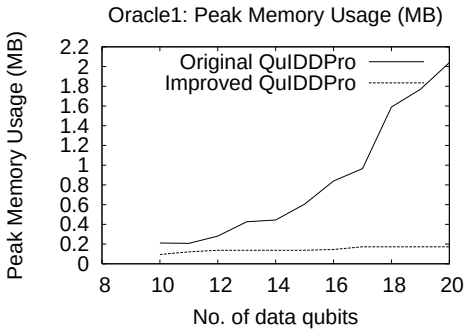


**Figure 2. Memory usage for the original [10] and improved QuIDDPro (oracle 1).**

## References


[1] R. I. Bahar et al., "Algebraic decision diagrams and their applications," *Journal of Formal Methods in System Design*, Vol. 10, no. 2/3, April/May 1997.

[2] D. Gottesman, "The Heisenberg representation of quantum computers," *Plenary speech at the 1998 International Conference on Group Theoretic Methods in Physics*, `http://xxx.lanl.gov/abs/quant-ph/9807006`.

[3] D. Greve, "QDD: a quantum computer emulation library," `http://thegreves.com/david/QDD/qdd.html`

[4] L. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett.* (79), pp. 325-8, 1997.

[5] A. J. G. Hey, ed., *Feynman and Computation: Exploring the Limits of Computers*, Perseus Books, 1999.

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

[7] K. M. Obenland and A. M. Despain, "A parallel quantum computer simulator," *High Performance Computing*, 1998.

[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. of Computing*, Vol. 26, p. 1484, 1997.

[9] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Improving gate-level simulation of quantum circuits", *Los Alamos Quantum Physics Archive*, Sept. 2003 `http://xxx.lanl.gov/abs/quant-ph/0309060`

[10] G. F. Viamontes, M. Rajagopolan, I. L. Markov, and J. P. Hayes, "Gate-level simulation of quantum circuits", *Proc. of ACM/IEEE Asia and South-Pacific Design Automation Conf.* (ASPDAC), pp. 295-301, Kitakyushu, Japan, January 2003.

[11] I. Wegener, *Branching Programs and Binary Decision Diagrams: Theory and Applications*, SIAM, 2000.