Just Like the Real Thing: Fast Weak Simulation of Quantum Computation

Stefan Hillmich[1], Igor L. Markov[2], and Robert Wille[1]
[1]Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
[2]Department of EECS, University of Michigan, USA
stefan.hillmich@jku.at, imarkov@eecs.umich.edu, robert.wille@jku.at
http://iic.jku.at/eda/research/quantum/

***Abstract*—Quantum computers promise significant speedups in solving problems intractable for conventional computers but, despite recent progress, remain limited in scaling and availability. Therefore, quantum software and hardware development heavily rely on simulation that runs on conventional computers. Most such approaches perform *strong simulation* in that they explicitly compute amplitudes of quantum states. However, such information is not directly observable from a physical quantum computer because quantum measurements produce random samples from probability distributions defined by those amplitudes. In this work, we focus on *weak simulation* that aims to produce outputs which are statistically indistinguishable from those of error-free quantum computers. We develop algorithms for weak simulation based on quantum state representation in terms of decision diagrams. We compare them to using state-vector arrays and binary search on prefix sums to perform sampling. Empirical validation shows, for the first time, that this enables mimicking of physical quantum computers of significant scale.**

***Index Terms*—quantum computing, simulation, weak simulation, sampling**

I. Introduction

Quantum computing [1] promises to fundamentally change the field of computing and its applications. For example, Shor's algorithm [2] performs integer factorization in low polynomial time and poses a severe threat to modern cryptography which relies on the hardness of integer factorization. Applications proposed more recently include search for better catalysts in quantum chemistry [3], as well as machine learning, cryptography, quantum simulation, and solving systems of linear equations [4]–[6]. Their potential has been recognized by Google, IBM, Microsoft as well as start-ups such as Rigetti and IonQ which heavily invest in this technology.

Despite initial optimism, the construction of quantum computers and the implementation of quantum algorithms turned out exceptionally challenging. Quantum computers available today are expensive, error-prone, limited in their scalability, and inaccessible to most researchers. Therefore, simulation methods which faithfully mimic the behavior of a quantum computer on conventional hardware are essential to the design, optimization, verification, and performance evaluation of quantum algorithms and their applications. However, state-of-the-art techniques for quantum circuit simulation [7]–[13] remain somewhat disconnected from this goal because they primarily focus on so-called *strong simulation*, i.e., explicitly compute some or all of the amplitudes of the final quantum state produced by a given circuit. Notably, such amplitudes cannot be directly observed from a quantum computer. Instead, every run of a quantum computer produces nondeterministic outputs of quantum measurements which can be interpreted as indices (represented in binary) of amplitudes and sampled according to probabilities computed as squared norms of these amplitudes. In contrast to *strong simulation*, the task of mimicking such nondeterministic output, possibly with some error, is called *weak simulation*. The two tasks are not equivalent, but weak simulation has not yet received extensive coverage [14], [15].

In this work, we develop fast methods for weak simulation. Since there often is no need to return exponential numbers of amplitudes, we (1) represent quantum states using a compressed data structure called *edge-weighted decision diagram*, and (2) develop novel algorithms to simulate measurements in terms of this data structure in *linear time* with respect to its size. To validate our approach and provide a performance baseline, we also show how to simulate measurement on an explicit array of all amplitudes by using prefix sums and binary search. Empirical validation confirms that the reduced memory needs of decision diagrams help making sampling from quantum states more practical. In comparison, state representations that use full arrays require exorbitant amounts of memory and often cannot perform weak simulation in practice. Our empirical results demonstrate, for the first time, that physical quantum computers running several well-known quantum algorithms on a significant scale can be mimicked faithfully and efficiently by simulators running on modest conventional computers.

The rest of the paper is structured as follows: Section II reviews the background on quantum states and operations, so as to make the paper accessible to readers without a keen understanding of quantum computing. Section III outlines the main idea of using weak simulation to mimic a quantum computer and also describes algorithms that rely on exponentially-sized arrays. All concepts and algorithms are illustrated by detailed examples. Section IV introduces our approach to weak simulation without exponentially-sized arrays. This section reviews how decision diagrams represent quantum states, then describes our algorithm for weak simulation and an enhancement for decision diagrams in this context. Empirical validation of our implementation is described in Section V. Section VI concludes the paper.

II. Background: Quantum Computing

In the realm of quantum computing, classical bits are generalized to *quantum bits* or *qubits*. While the former can be either in state 0 or in state 1, qubits may assume one of two basis states (denoted $|0\rangle$ and $|1\rangle$) and also any linear combination of them. This is described by $|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$ with *amplitudes* $\alpha_0, \alpha_1 \in \mathbb{C}$ which have to satisfy the normalizing constraint $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Qubits with $\alpha_0 \neq 0$ and $\alpha_1 \neq 0$ are said to be in *superposition*.[1]

For multi-qubit quantum systems, the description is extended accordingly to represent the exponential number of basis states the systems can assume. For example, a system with two qubits has four basis states, i.e., $|\psi\rangle = \alpha_{00} \cdot |00\rangle + \alpha_{01} \cdot |01\rangle + \alpha_{10} \cdot |10\rangle + \alpha_{11} \cdot |11\rangle$. Since the amplitudes dictate the probabilities, the normalizing constraint is extended as well: $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$. Commonly, the description of a quantum state is shortened to a vector containing only the amplitudes, e.g., $|\psi\rangle = [\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}]^{\mathrm{T}}$.

Example 1. *Consider an arbitrary quantum system composed of two qubits, which is in the state* $|\psi\rangle = 1/\sqrt{2} \cdot |00\rangle + 0 \cdot |01\rangle + 0 \cdot |10\rangle + 1/\sqrt{2} \cdot |11\rangle$. *This represents a valid state, since* $|1/\sqrt{2}|^2 + 0^2 + 0^2 + |1/\sqrt{2}|^2 = 1$. *The corresponding state vector is* $|\psi\rangle = [1/\sqrt{2}, 0, 0, 1/\sqrt{2}]^{\mathrm{T}}$.

Before quantum measurement is applied, the state of a quantum system can be manipulated using unitary quantum operations. Such operations are defined through unitary matrices, i.e., square matrices whose inverse is their conjugate transposed [1]. Examples of important single-qubit quantum operations are

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{ and } Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

where X negates the state of the qubit, H sets the qubit into superposition, and Z shifts the phase of the qubit. To couple multiple qubits, one can use, e.g., the $CNOT$ (controlled-NOT) operation, which negates a *target qubit*, iff the chosen *control qubit* is in the state $|1\rangle$. This is defined through the matrix

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The action of a quantum operation represented by a matrix on a quantum state represented by a vector can be described through matrix-vector multiplication as illustrated next:

Example 2. *Consider a quantum system composed of two qubits which is currently in state* $|\psi\rangle = |00\rangle$. *Performing an* H *operation on the first qubit and a* $CNOT$ *operation (with control on the first qubit and target on the second) yields a new state* $|\psi'\rangle$ *determined by multiplying the vector* $|\psi\rangle$ *with the matrices of these two operations, i.e.,*

$$\underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}}_{H \text{ on } 1^{st} \text{ qubit}} \times \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{CNOT} \times \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{|\psi\rangle} = \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{|\psi'\rangle}.$$

Unfortunately, the amplitudes α_{00}, α_{01}, α_{10}, and α_{11} of the resulting output state cannot be observed directly on a quantum computer. Instead, these amplitudes dictate the probability of certain outcomes of a measurement with respect to the corresponding basis states. More precisely, measuring a single qubit in state $|\psi\rangle = \alpha_{00} \cdot |00\rangle + \alpha_{01} \cdot |01\rangle + \alpha_{10} \cdot |10\rangle + \alpha_{11} \cdot |11\rangle$ yields the output $|00\rangle$ with probability $|\alpha_{00}|^2$, the output $|01\rangle$ with probability $|\alpha_{01}|^2$, etc. After the measurement, the qubits will lose any superposition, i.e., they collapse into a basis state.

Example 3. *Consider again the state* $|\psi'\rangle = [1/\sqrt{2}, 0, 0, 1/\sqrt{2}]^{\mathrm{T}}$ *produced in Example 2. Measuring its qubits does* not *provide the amplitudes of this state, but only yields one of the possible basis states (here:* $|00\rangle$ *or* $|11\rangle$*; both with probability of* $|\alpha_{00}|^2 = |\alpha_{11}|^2 = |1/\sqrt{2}|^2 = 1/2$*). After the measurement, both qubits will lose their superposition, i.e., the state collapses to the resulting basis state.*

A common way to represent quantum computations is with quantum circuit diagrams [1]. Here, the qubits are represented by horizontal lines, while quantum operations are placed on the qubits and are applied from left to right.

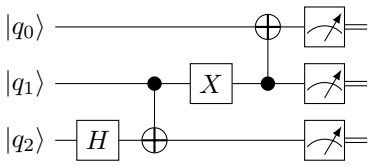


Fig. 1. A quantum circuit diagram

Example 4. *Fig. 1 shows a quantum circuit diagram with four operations:* H, $CNOT$ ($\bullet$ *as control,* $\oplus$ *as target),* X, *and another* $CNOT$, *followed by a measurement on each qubit. The double lines after the measurements indicate that the qubit is in a basis state here (as a result of the measurement).*

III. Weak Simulation: Mimicking Physical Quantum Computers

In this work, we explore the simulation of quantum computers on conventional hardware that produces the same kind of output as the physical quantum computers. Moreover, we hope to produce outputs that are statistically indistinguishable from those of (error-free) physical quantum computers. Given the well-defined model of quantum computation covered in Section II, this task seems straightforward at a first glance: We are given an input basis state and a sequence of quantum operations, and need to (nondeterministically) produce bitstrings that represent measurement outcomes. However, performing this kind of simulation without unnecessary overhead has been elusive, and the literature focuses on explicitly describing the

[1] Another important phenomenon is *entanglement*, where the measurement of a single qubit may influence the measured result of another qubit.

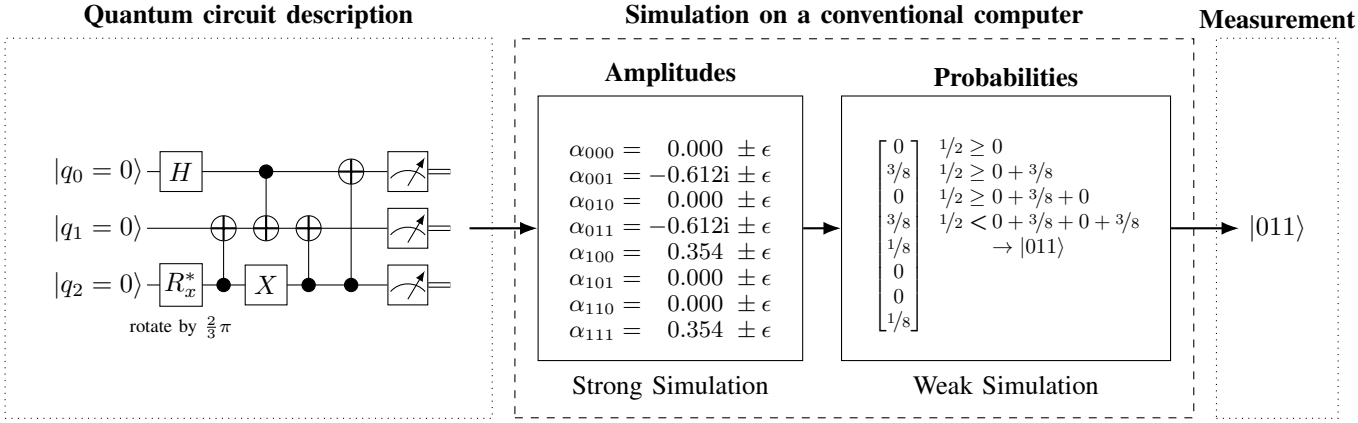


Fig. 2. Mimicking a physical quantum computer by generating individual output samples after strong simulation

output distribution rather than producing individual output samples efficiently [14], [15]. In this section, we describe and illustrate relevant challenges using the following example:

Example 5. *Consider Fig. 2 (left) which illustrates the running example used in the remainder of this work. The dotted box specifies the input basis state* $|000\rangle$ *along with a sequence of quantum operations (given as a quantum circuit diagram) to be simulated.*

Based on the n-qubit input state and circuit operations, the 2^n amplitudes α_i of the corresponding output state can be calculated by matrix-vector multiplication (see Section II).

Example 6. *Using the vector representation of the input basis state* $|000\rangle$ *and the matrix representations of circuit operations, we can determine the output state vector through a series of matrix-vector multiplications by calculating the amplitudes* $\alpha_{000}, \alpha_{001}, \alpha_{010}, \ldots, \alpha_{111}$*, which cannot be produced by one run of a quantum computer. As indicated in Fig. 2 (middle), computing those amplitudes with a small error is also acceptable.*

Finally, to simulate a quantum measurement in the computational basis, we sample an output-basis state from the probability distribution (p_i) where $p_i = \alpha_i^* \alpha_i = |\alpha|^2$.

Example 7. *Fig. 2 (right) illustrates a probability distribution of measurement outcomes. Sampling from that distribution may yield* $|001\rangle$.

To faithfully mimic a physical quantum computer, one can first capture the output distribution (p_i) via strong simulation and, then, sample from it, as outlined in Fig. 2. If the output probability distribution is described explicitly by a vector of probabilities p_i, sampling can be performed using a standard *biased random selection* routine. The idea is to generate a random number $\hat{p} \in [0, 1)$ and, then, determine the largest index i such that $\Sigma_{k=0}^{i} p_k \leq \hat{p}$.

Example 8. *Continuing our running example in Fig. 2 (right), we find the probability* p_i *of each basis state. Now, assuming that* $\hat{p} = 1/2$ *was generated randomly, the fourth prefix sum exceeds* $\hat{p}$ *since* $0 + 3/8 + 0 + 3/8 > 1/2$*. Therefore,* $i = 3$ *and the resulting sample is* $|011\rangle$.

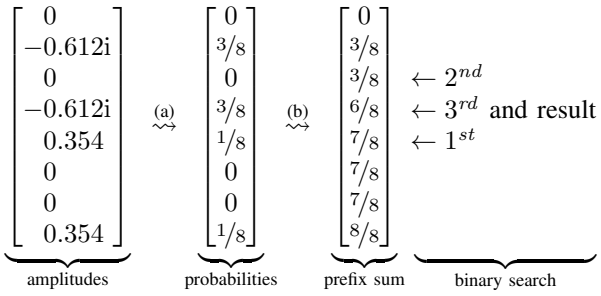


Fig. 3. Biased random selection via binary search on a prefix array. The precomputation in (a) is performed once to facilitate efficient repeated sampling in (b).

The index i can be found by a direct (linear) traversal, which takes 2^{n-1} steps on average. To accelerate *repeated sampling*, one can first compute the prefix values $r_i = \Sigma_{k=0}^{i} p_k$ and store them in an array, noting that $r_i \leq r_{i+1}$ for all i. Then, for each newly-generated $\hat{p}$, find i using binary search in $\mathcal{O}(\log 2^n) = \mathcal{O}(n)$ time. Fig. 3 illustrates the computation of the prefix array and subsequent sampling by binary search in this array.

The use of precomputation and repeated binary search makes it possible to draw a large number of samples much more efficiently than with linear traversals. However, the use of binary search requires random access to the prefix array, which must be loaded in memory. In contrast, linear traversals can be performed on large vectors stored in out-of-memory files, with only small blocks loaded to memory at any given time. Both techniques are limited in scaling by their use of exponentially-sized arrays and the need to read each amplitude at least once.

In the field of quantum circuit simulation, an overwhelming majority of prior work focuses on computing all amplitudes through strong simulation. Here, the exponential complexity is tackled either by supercomputers (by distributing both the storage and the processing to multiple processors [7]–[10]) or by dedicated data structures such as decision diagrams [11], [12] and Matrix Product States [13]. However, of equal importance is efficient storage of resulting state/probability vectors and efficient output sampling with such data representations. Thus, prior literature leaves a gap between strong simulation and the task of mimicking a physical quantum computer.

IV. Advanced Weak Simulation

In this section, we develop a method for weak simulation that does not rely on exponentially-large arrays and reduces memory blow-up in important cases. Moreover, this method generates samples quickly after an initial precomputation.

As outlined in Section III, weak simulation can be approached by first computing *all* amplitudes of a state vector to determine the probability of each output basis state. Storing amplitudes and/or probabilities in exponentially-large arrays is often prohibitively expensive, but in many important cases *decision diagrams* can capture quantum states and support more memory-efficient representation [11], [16], [17]. In the following, we show how such data structures can be extended to perform weak simulation. First, we provide the necessary background material on decision diagrams and, then, describe our algorithm for weak simulation. With an eye on efficient implementation, we also develop a normalization scheme for decision diagrams and combine the several ideas presented into a faithful and efficient weak simulation technique.

A. Background on Decision Diagrams

Decision diagrams have been successfully utilized with quantum circuits for tasks such as simulation [11], [12], [18], synthesis [19]–[21], and verification [22], [23], since they often drastically reduce the memory needed to represent state vectors and operation matrices. Strong simulation approaches based on decision diagrams have recently moved into the spotlight because they can significantly outperform vector-based simulators in cases where data redundancies can be exploited—in extreme cases leading to an improvement in runtime from 30 days to 2 minutes [12].

The main idea behind decision diagrams is to dynamically identify data redundancies and provide compaction by sharing sub-structures. Conceptually, the state vector is split into two equal-sized sub-vectors. Given that the number of amplitudes is a power of two, this process is repeated until the sub-vectors contain a single element only, i.e., one split for every qubit. Where identical sub-vectors occur in this process, this redundancy is exploited by re-using (sharing) the same sub-structure in the resulting decision diagram. In practice, the identification of repeated sub-vectors is performed by hashing and bottom-up rather than top-down. Unlike the early uses of decision diagrams in quantum circuit simulation [11], we leverage edge-weighted decision diagrams [12] which provide a more powerful compression mechanism and need significantly less memory in important cases. In the following, we outline key technical details of relevant edge-valued decision diagrams to represent quantum state-vectors.

Consider a quantum system with qubits $q_{n-1}, q_{n-2}, \ldots, q_0$, whereby q_{n-1} represents the most significant qubit. Then, the first 2^{n-1} entries of the corresponding state vector represent the amplitudes for the basis states with q_{n-1} set to $|0\rangle$; the other entries represent the amplitudes for states with q_{n-1} set to $|1\rangle$. This decomposition is represented in a decision diagram structure by a node labeled q_{n-1} and two successors leading to nodes representing the two sub-vectors. By convention, the left (right) edge indicates the 0-successor (1-successor). The sub-vectors are recursively decomposed until vectors of size 1 (i.e., complex numbers) result. During this decomposition, equivalent sub-vectors can be represented by the same nodes—reducing the complexity of the representation. Then, instead of having a terminal node for every distinct value in the state vector, common factors of the amplitudes are stored in the edge weights. Each amplitude in the state vector is represented by a directed path in the decision diagram, and its complex value can be reconstructed by multiplying the edge weights along the path. The approach provides exponential data compression when a compact graph exhibits a large number of directed paths.

Example 9. *Consider the state vector in Fig. 4a—the annotations sketch how the vector is decomposed (left) and which base state corresponds to each entry in the vector (right). Fig. 4b shows the corresponding decision diagram. Here, e.g., the amplitude of the state* $|111\rangle$ *is accessed by following the path in the decision diagram for* $q_2 = 1$, $q_1 = 1$, $q_0 = 1$ *and multiplying the edge weights along the path, i.e.,* $-0.612\mathrm{i} \cdot 0.578\mathrm{i} \cdot 1 \cdot 1 = 0.354$.

B. Decision Diagrams for Weak Simulation

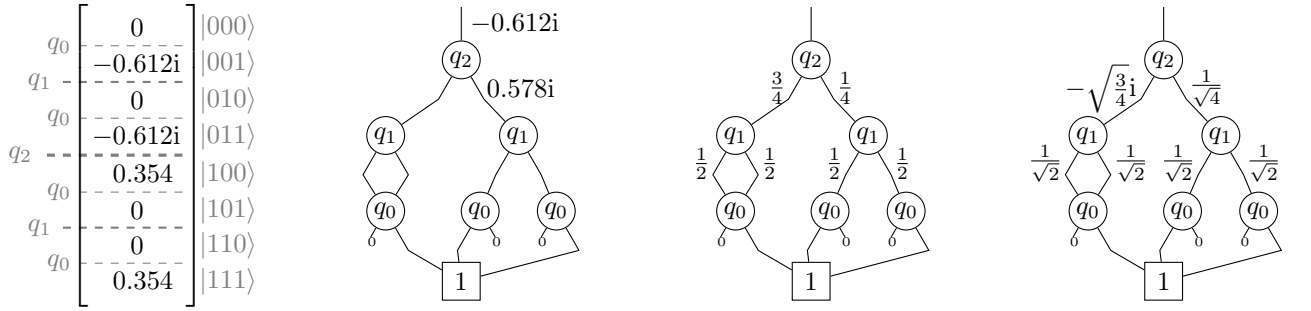
Decision diagrams can compactly represent all amplitudes of a quantum state, given sufficient data redundancy. The key idea is that instead of explicitly storing an exponential number of probabilities, we encode them in a decision diagram with the same structure as the decision diagram for amplitudes. This precomputation step is performed by full traversals of the decision diagram, where we compute new weights—the probabilities that the left and right successor of a node should be followed when drawing an output sample. The idea behind the subsequent sampling algorithm is to perform a randomized single-path traversal of the decision diagram from the top node, deciding randomly at each node whether to descend to the left or right child node, based on precomputed probabilities. While physical quantum states get destroyed by quantum measurement, simulated measurement is a read-only operation that can be repeated.

We now specify two types of probabilities computed at each node and the decision diagram traversals that calculate them:

1) The *downstream probabilities* are sums of probabilities of all half-paths from the current node to any terminal node. They are calculated by a *depth-first traversal*.
2) The *upstream probabilities* are sums of probabilities of all half-paths from the root to the current node. They are calculated by a *breadth-first traversal*.

The runtime of the traversals is linear in the number of nodes in the decision diagram. Once both probabilities for a node are computed, the probability to follow the left (right) successor is the product of the upstream and downstream probabilities weighted by the squared magnitude of the left (right) edge weight. Finally, the probability of an individual basis state is the product of all probabilities along the path.

Example 10. *Fig. 4c continues our running example and shows the same decision diagram with the edge weights giving the probability of choosing either successor node. From the*



(a) Vector representation (b) Decision diagram (c) Probabilities on edges (d) Proposed normalization scheme

Fig. 4. Representations of the quantum state as a vector and a decision diagram

outgoing edge weight of q_2, it can be seen that the left successor will be chosen 3 out of 4 times, whereas the right successor will only be chosen 1 out of 4 times. That is, the resulting basis state will have $q_2 = 0$ in $75\,\%$ of the samples and $q_2 = 1$ in $25\,\%$ of the samples.

C. Efficient Sampling

The method described above enables sampling with reduced memory complexity in important cases, exploiting on the compactness of the representation. Each sample is produced by traversing a root-to-terminal path in the decision diagram. Given that for n-qubit states, such paths have at most $n + 2$ nodes, each output sample is generated in $\mathcal{O}(n)$ time. Still, the runtime of the sampling process can be improved using a new normalization scheme for outgoing edges.

The outgoing edges of a node are often normalized by dividing both weights by the weight of the left-most edge (when $\neq 0$), and multiplying this factor to the incoming edges as illustrated in Fig. 4b. However, we found that in the context of sampling, it is more effective to divide by the norm of the vector containing both edge weights and adjust the incoming edges accordingly. This normalizes the sum of the squared magnitudes of the outgoing edge weights to 1 and is consistent with the quantum semantics, where basis states $|0\rangle$ and $|1\rangle$ are observed after measurement with probabilities that are squared magnitudes of the respective weights.

Example 11. *Applying the proposed normalization scheme to Fig. 4b results in the decision diagram shown in Fig. 4d.*

V. Empirical Validation

Our method for weak simulation avoids explicit exponential-sized vectors in important cases and generates output samples quickly to mimic physical quantum computers. To evaluate it, we use a dedicated implementation of edge-valued decision diagrams with adaptations to quantum circuit simulation, such as the use of high-precision complex numbers, based on [24] (common software packages for decision diagrams are generally not usable in this context). Our empirical validation focuses on output sampling for quantum circuits that implement well-known quantum algorithms, algorithmic blocks, and applications:

- The *Quantum Fourier Transformation* (QFT) (denoted "qft_A" for A qubits),
- Grover's search [25] with a random oracle (denoted "grover_A" for A qubits),
- Shor's algorithm to factorize integers [2] (denoted "shor_A_B" for factorizing A with the coprime value B),
- quantum circuits simulating the uniform electron gas [26] (denoted "jellium_AxA" for size of the system $A \times A$ on a grid), and
- quantum circuits provided by researchers from Google [27] as candidates to establish quantum-computational supremacy using controlled-Z gates (denoted "supremacy_AxB_C", representing a circuit on an $A \times B$ surface with depth C).

Following the overall flow outlined in Section III, we performed strong simulation of these circuits to find the final quantum state in the form of a decision diagram. Subsequently, we completed weak simulation by (1) calculating the prefix sum and, then, conducting the sampling through binary search as described in Section III (*vector-based*) and by (2) using decision diagrams (*DD-based*) and the normalization scheme described in Section IV. We generated one million samples, which is common for benchmarking quantum computers today. All runs were performed on an Intel i7-7700K CPU (4.2 GHz) with 32 GiB main memory (and an additional 32 GiB of swap space) under GNU/Linux.

The results are presented in Table I. The first two columns contain the name of the benchmark and the number of qubits. The following two columns contain the size of the state vector (number of entries) that we sample from as well as the time it took to compute the prefix sum and to draw one million samples (in CPU seconds). For DD-based sampling, similar columns show the size (number of nodes) of the state-vector decision diagram and the time it took to draw one million samples from this decision diagram. Both approaches support faithful (error-free) weak simulation. Moreover, the results also confirm what one might expect from complexity analysis: If the (exponentially large) state vector can be stored in memory[2], then samples can be generated using prefix sums, as described in Section III. Otherwise (e.g., for the quantum algorithms qft_32, qft_48, and grover_35), a *memory out* (MO) occurs and this method reaches its limits. In contrast, decision diagrams facilitate much more compact representations. Together with the sampling method proposed in Section IV,

[2]For vector-based sampling, sizes above 2^{30} incurred swapping, the DD-based sampling did not require swapping in the evaluation.

TABLE I
RUNTIME AND MEMORY FOR ERROR-FREE SAMPLING OF 1M BITSTRINGS

benchmarks		vector-based		DD-based	
name	qubits	size	t [s]	size	t [s]
qft_16	16	2^{16}	0.12	$16 \approx 2^{4.0}$	**0.22**
qft_32	32	2^{32}	MO	$32 \approx 2^{5.0}$	**0.43**
qft_48	48	2^{48}	MO	$48 \approx 2^{5.5}$	**0.63**
grover_20	21	2^{21}	0.70	$40 \approx 2^{5.3}$	**0.23**
grover_25	26	2^{26}	17.91	$50 \approx 2^{5.6}$	**0.27**
grover_30	31	2^{31}	993.99	$60 \approx 2^{5.9}$	**0.29**
grover_35	36	2^{36}	MO	$70 \approx 2^{6.1}$	**0.43**
shor_33_2	18	2^{18}	0.15	$48\,793 \approx 2^{15.5}$	**0.20**
shor_55_2	18	2^{18}	0.16	$93\,478 \approx 2^{16.5}$	**0.21**
shor_69_4	21	2^{21}	0.62	$196\,382 \approx 2^{17.5}$	**0.26**
shor_221_4	24	2^{24}	3.72	$1\,048\,574 \approx 2^{20.0}$	**0.27**
shor_247_4	24	2^{24}	3.81	$1\,376\,221 \approx 2^{20.3}$	**0.31**
jellium_2x2	8	2^{8}	0.04	$117 \approx 2^{6.8}$	**0.09**
jellium_3x3	18	2^{18}	0.17	$59\,475 \approx 2^{15.8}$	**0.22**
supremacy_4x4_10	16	2^{16}	0.11	$65\,070 \approx 2^{15.9}$	**0.39**
supremacy_5x4_10	20	2^{20}	0.66	$486\,503 \approx 2^{18.8}$	**0.82**
supremacy_5x5_10	25	2^{25}	12.04	$16\,779\,617 \approx 2^{24.0}$	**4.28**

this supports faithful and efficient weak simulation in these cases as well. Overall, our results demonstrate, for the first time, that the outputs of physical quantum computers running several well-known quantum algorithms can be mimicked faithfully and efficiently by simulators running on conventional computers without relying on exponentially-sized arrays.

VI. CONCLUSIONS

We proposed efficient methods for *weak simulation*, whose output cannot be statistically distinguished from the output of (error-free) physical quantum computers. Prior literature in the field focuses on *strong simulation*, i.e., computing the amplitudes of the final quantum state. However, generating output samples is a distinctively different task. Hence, we outline sampling for amplitudes in an exponentially-large array and develop an alternative method that works directly with state vectors compactly represented as decision diagrams. Both techniques (1) rely on precomputations performed in time linear in the size of input data, and (2) generate n-qubit output samples in $\mathcal{O}(n)$ time. Sampling from decision diagrams is a little slower in practice, but scales much better in terms of memory for quantum states produced by important quantum circuits. Empirical validation confirms the feasibility of our techniques and demonstrates, for the first time, a faithful mimicking of quantum computers by a simulator running on a conventional computer at scales where exponentially-sized arrays are infeasible.

ACKNOWLEDGMENTS

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

REFERENCES

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

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.

[3] Y. a. Cao, "Quantum chemistry in the age of quantum computing," *arXiv:1812.09976*, 2018.

[4] A. Montanaro, "Quantum algorithms: An overview," *npj Quantum Information*, vol. 2, p. 15023, 2016.

[5] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[6] P. J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev *et al.*, "Quantum algorithm implementations for beginners," *arXiv:1804.03719*, 2018.

[7] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "QuEST and high performance simulation of quantum computers," *Scientific Reports*, vol. 9, no. 1, pp. 1–11, 2019.

[8] D. Wecker and K. M. Svore, "LIQUi|>: A software design architecture and domain-specific language for quantum computing," *CoRR*, vol. abs/1402.4467, 2014.

[9] N. Khammassi, I. Ashraf, X. Fu, C. Almudever, and K. Bertels, "QX: A high-performance quantum computer simulation platform," in *Design, Automation and Test in Europe*, 2017.

[10] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik, "qHiPSTER: The quantum high performance software testing environment," *CoRR*, vol. abs/1601.07195, 2016.

[11] G. F. Viamontes, I. L. Markov, and J. P. Hayes, *Quantum Circuit Simulation*. Springer, 2009.

[12] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019.

[13] D. S. Wang, C. D. Hill, and L. C. L. Hollenberg, "Simulations of Shor's algorithm using matrix product states," *Quantum Information Processing*, vol. 16, no. 7, p. 176, 2017.

[14] M. V. den Nest, "Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond," *arXiv:0811.0898*, 2008.

[15] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, "Simulation of quantum circuits by low-rank stabilizer decompositions," *Quantum*, vol. 3, p. 181, 2019.

[16] V. Samoladas, "Improved BDD algorithms for the simulation of quantum circuits," in *European Symposium on Algorithms*, 2008, pp. 720–731.

[17] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 86–99, 2016.

[18] A. Zulehner and R. Wille, "Matrix-vector vs. matrix-matrix multiplication: Potential in DD-based simulation of quantum computations," in *Design, Automation and Test in Europe*, 2019.

[19] A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Design, Automation and Test in Europe*, 2006, pp. 317–322.

[20] P. Niemann, R. Wille, and R. Drechsler, "Improved synthesis of Clifford+T quantum functionality," *Design, Automation and Test in Europe*, pp. 597–600, 2018.

[21] A. Zulehner and R. Wille, "One-pass design of reversible circuits: Combining embedding and synthesis for reversible logic," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 996–1008, 2018.

[22] S.-A. Wang, C.-Y. Lu, I.-M. Tsai, and S.-Y. Kuo, "An XQDD-based verification method for quantum circuits," *IEICE Trans. Fundamentals*, vol. 91-A, no. 2, pp. 584–594, 2008.

[23] L. Burgholzer and R. Wille, "Improved DD-based equivalence checking of quantum circuits," in *Asia and South Pacific Design Automation Conf.*, 2020.

[24] A. Zulehner, S. Hillmich, and R. Wille, "How to efficiently handle complex values? implementing decision diagrams for quantum computation," in *Int'l Conf. on CAD*, 2019, pp. 1–7.

[25] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Theory of computing*, 1996, pp. 212–219.

[26] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, "Low-depth quantum simulation of materials," *Phys. Rev. X*, vol. 8, p. 011044, 2018.

[27] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, p. 595, 2018, Benchmarks available at https://github.com/sboixo/GRCS.