Arrays vs. Decision Diagrams: A Case Study on Quantum Circuit Simulators

Thomas Grurl[*†] Jürgen Fuß[*] Stefan Hillmich[†] Lukas Burgholzer[†] Robert Wille[†]
[*]Secure Information Systems, University of Applied Sciences Upper Austria, Austria
[†]Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
{thomas.grurl, juergen.fuss}@fh-hagenberg.at {stefan.hillmich, lukas.burgholzer, robert.wille}@jku.at
http://iic.jku.at/eda/research/quantum/

***Abstract*—Despite the recent progress in the physical implementation of quantum computers, a significant amount of research still depends on the use of quantum circuit simulators running on classical hardware. While there are several techniques for quantum circuit simulation, many state-of-the-art simulators rely on an array-based simulation approach. However, this array-based approach has exponential memory complexity with respect to the number of simulated qubits. To address this drawback, complementary approaches based on decision diagrams have been proposed. While these approaches allow simulating circuits that could not be simulated before, they come with their own drawbacks. Unfortunately, no detailed case study has been conducted to date, which compares those complementary approaches and their respective strengths and weaknesses. In this work, we are addressing this by providing a survey on both approaches as well as a detailed case study on their respective performances.**

I. Introduction

Quantum computers can solve specific problems significantly faster than classical computers by utilizing quantum mechanical effects. First examples of quantum algorithms are Shor's factorization algorithm [1] and Grover's search algorithm [2]. Recently, other relevant applications for quantum algorithms have been found in the areas of chemistry, finance, machine learning and mathematics [3]–[7]. In addition to the theoretical work, there have also been impressive accomplishments towards the physical realization of quantum computers. Most notable is Google's recent claim of having achieved quantum supremacy by calculating a task within two minutes on their quantum computer for which they estimate a state-of-the-art super-computer would require approximately ten thousand years [8]. Also other big players such as Intel, Rigetti, Microsoft or Alibaba are all heavily investing in this emerging technology.

Nevertheless, the development of quantum computers still is in its research phase and available realizations still suffer from a limited number of qubits, a relatively high error rate and a short coherence time. Therefore, a significant amount of research still depends on the use of quantum circuit simulators running on classical hardware.

From a mathematical point of view, simulating a quantum circuit boils down to matrix-vector multiplication. Accordingly, many state-of-the-art quantum circuit simulators use arrays to represent vectors and matrices and conduct simulation by matrix-vector multiplication on these arrays (e.g. [9]–[15]). However, array-based simulators are severely limited by the inherent exponential size of the involved vectors and matrices with respect to the number of simulated qubits.

To overcome the restrictions of array-based quantum circuit simulators, several other simulation approaches have been developed, e.g. based on the *stabilizer formalism* [16], *matrix product states* [17], tensor network contractions [18] and *decision diagrams* [19]–[23]. These simulation styles have their own limitations, e.g. restrictions on the employable quantum operations, exponential growth with respect to the degree of entanglement or feasibility only for low-depth circuits.

In the following, we focus on array- and decision diagram-based quantum circuit simulation, because both are common-purpose simulators (i.e. they have no restrictions in their supported quantum operations or the circuits' depth) and conduct simulation by matrix-vector multiplication. While array-based simulation is widely adopted and well understood, decision diagram-based simulation offers potential for further improvement. However, no detailed case study has been conducted to date, which compares the strengths and weaknesses of these two complementary approaches.

In this work, we address the lack of evaluation considering these different simulation approaches. To this end, we first provide a survey on array- and decision diagram-based quantum circuit simulation including a discussion of their presumed strengths and weaknesses. Afterwards, we show the results of a detailed case study evaluating the respective performances. For this case study, we used our access to state-of-the-art implementations and hardware infrastructure, namely the commercial *Atos Quantum Learning Machine* (QLM) [15] coming with 96 cores. To the best of our knowledge, this is the first comparison of both considered approaches on such a high-end computer. From the results obtained by this case study, we are able to confirm previously made discussions and provide new insights into the strengths and weaknesses of array- and decision diagram-based quantum circuit simulation.

The remainder of this paper is structured as follows: In Section II, we review the basics of quantum computing. Section III introduces how array- and decision diagram-based quantum simulation is conducted from a conceptional point of view. In Section IV, we present the setup of the conducted case study, followed by the presentation and discussion of the results. Finally, Section V concludes the paper.

II. Background

In order to keep this work self-contained, this section briefly reviews the concepts of quantum computing. We refer the interested reader to [24] for a thorough introduction.

A. Quantum States

While in classical computing the state of a system can be described using bits which are either 0 or 1, in quantum computation the state is described by so-called *quantum bits* (qubits). Those qubits can be in the states 0 or 1, which are called basis states and – using Dirac notation – are written as $|0\rangle$ and $|1\rangle$. However, they can also be in an (almost) arbitrary superposition of these two basis states. More precisely, the state $|\psi\rangle$ of a qubit is described by $|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$. The two complex-valued factors α_0 and α_1 are called *amplitudes* and denote how much the qubit is related to each of the two basis states. The amplitudes must satisfy the normalization constraint $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

While measuring a classical bit returns its exact state, the measurement of a qubit results in its collapse to one of the basis states $|0\rangle$ or $|1\rangle$, with probability $|\alpha_0|^2$ and $|\alpha_1|^2$, respectively. Thus, the exact state of a quantum system cannot be directly observed.

These concepts can be generalized to describe states composed of multiple qubits—also called *quantum registers*. Since every qubit has exactly two basis states, a quantum register composed of n qubits has 2^n basis states. Each basis state of the quantum register is represented by an $|x\rangle$, where $x \in \{1,0\}^n$. This generalization leads to the following definition of a quantum state:

Definition 1. *All possible states of a quantum system composed of n qubits are defined by*

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x \cdot |x\rangle, \textit{ where } \sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1,\ \alpha_x \in \mathbb{C}.$$

The state $|\psi\rangle$ of a quantum register of size n can also be written as a complex column vector of size 2^n, whose entries correspond to the amplitudes α_x for $x \in \{0,1\}^n$.

Example 1. *Consider a two qubit register in the state*

$$|\psi\rangle = \frac{1}{\sqrt{2}} \cdot |00\rangle + 0 \cdot |01\rangle + 0 \cdot |10\rangle + \frac{1}{\sqrt{2}} \cdot |11\rangle,$$

which is represented as $1/\sqrt{2} \cdot \begin{bmatrix}1&0&0&1\end{bmatrix}^\top$. This constitutes a valid state since $|1/\sqrt{2}|^2 + 0^2 + 0^2 + |1/\sqrt{2}|^2 = 1$. Measuring the system yields either $|00\rangle$ or $|11\rangle$ – both with probability $|1/\sqrt{2}|^2 = 1/2$. Note that the measurement outcome of one qubit affects the other one as well – an essential concept in quantum computing referred to as entanglement.

B. Quantum Operations

The state of a quantum system can be manipulated using quantum operations. With the exception of measurements, all quantum operations are inherently reversible and represented by unitary matrices, i.e. complex-valued square matrices whose inverse is given by their conjugate transpose. The size of the matrix depends on the number of qubits the operation is applied to. Examples for important single qubit operations include the NOT $= \begin{bmatrix}0&1\\1&0\end{bmatrix}$ operation, which negates the state of a qubit, and the Hadamard operation $\mathrm{H} = 1/\sqrt{2}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$, which transform a qubit from a basis state into a superposition. Additionally, there are also multi-qubit operations. A prominent two-qubit operation is the controlled-NOT operation (CNOT), which negates the state of its target qubit if the designated control qubit is in state $|1\rangle$. The functionality of a CNOT operation with control on the first and target on the second qubit is represented by the matrix

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

Quantum operations applied to quantum states are evaluated by employing matrix-vector multiplication. More precisely:

Definition 2. *Applying a quantum operation $U \equiv [u_{i,j}]$ (with $0 \le i,j < 2^n$), to a quantum state $|\psi\rangle \equiv [\alpha_x]$ (with $0 \le x < 2^n$), yields an output state $|\psi'\rangle \equiv [\alpha'_i]$ defined by $|\psi'\rangle = U \cdot |\psi\rangle$, i.e.*

$$|\alpha'_i\rangle = \sum_{x\in\{0,1\}^n} u_{i,x} \cdot |\alpha_x\rangle, \textit{ for } 0 \le i < 2^n.$$

Example 2. *Consider a quantum register $|\psi\rangle$ composed of two qubits q_0 and q_1 in the state*[1] *$1/\sqrt{2} \cdot \begin{bmatrix}1&0&1&0\end{bmatrix}^\top$. Applying a* CNOT *operation to the state yields*

$$\underbrace{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}}_{\mathrm{CNOT}} \cdot \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix}}_{|\psi\rangle} = \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}}_{|\psi'\rangle}.$$

Measuring the final state $|\psi'\rangle$ leaves the system in state $|00\rangle$ or $|11\rangle$, each with probability $|1/\sqrt{2}|^2 = 1/2$.

III. Quantum Circuit Simulation

This section reviews the main concepts of array- and decision diagram-based quantum circuit simulation and discusses their respective strengths and weaknesses. Based on that, a case study on their performance has been conducted whose results are summarized in the next section.

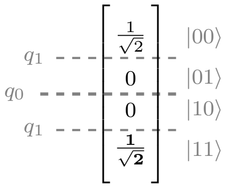
A. Array-based Simulation

The array-based simulation approach realizes the concepts reviewed in Section II in a straightforward fashion: Vectors and matrices are described in terms of 1-dimensional and 2-dimensional arrays, respectively. Then, simulation is conducted by matrix-vector multiplication as illustrated in Example 2, which can easily be implemented on top of those arrays.

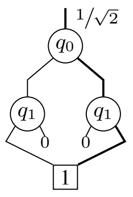
One obvious drawback is that this representation incurs a huge memory footprint, since all arrays representing the vectors and matrices are exponentially large. As a consequence, each additional qubit effectively doubles or quadruples the required memory to represent the states and operations, respectively. These memory requirements limit array-based simulation methods to rather small/moderate quantum computations (today's practical limit is less than 50 qubits [8], [25]).

In contrast, the underlying direct realization of matrix-vector multiplication allows for huge potential with respect to concurrent executions. In fact, the multiplication of a matrix

[1]This state can be constructed by applying a Hadamard operation to q_0 when the state is initialized to $|00\rangle$.

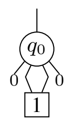


(a) Vector representation

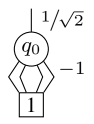


(b) DD representation

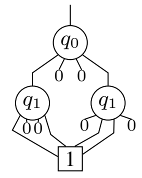
Fig. 1. Decision diagram-based state vector representation



(a) NOT



(b) H



(c) CNOT

Fig. 2. Decision diagram-based operation representation (cf. Sec. II-B)

with a vector can easily be decomposed into a series of multiplications and additions of (smaller) sub-matrices and sub-vectors, i.e.

$$\begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} M_{00} \cdot V_0 + M_{01} \cdot V_1 \\ M_{10} \cdot V_0 + M_{11} \cdot V_1 \end{bmatrix}.$$

These can further be decomposed in a recursive fashion—eventually leading to a large set of intermediate steps which can be executed concurrently with little synchronization overhead. State-of-the-art methods (such as [9]–[15]) make heavy use of that and do not only use all the cores available in modern desktop computers, but employ super-computers which allow huge degrees of concurrency and implement sophisticated parallelization schemes.

B. Decision Diagram-based Simulation

In order to tackle the exponential memory complexity of array-based simulation approaches, decision diagrams (also called DDs) have been proposed as one possibility for compactly representing quantum states and operations [19]–[23]. Using decision diagrams allows for a much more compact representation of the exponentially large matrices and vectors and already led to substantial improvements e.g. for synthesis [26]–[28] or verification [29], [30]. In the remainder of this subsection, we provide a survey as to how decision diagram-based simulation works and how it differs from array-based simulation.

1) Representation of Quantum States: The general idea of DD-based quantum simulation is that quantum states and operations are not represented in the form of vectors and matrices, but rather as decision diagrams. More precisely, consider a quantum register composed of n qubits $q_0, q_1, \ldots, q_{n-1}$, where q_0 represents the most significant qubit. The first 2^{n-1} entries of the corresponding state vector represent entries in which q_0 is $|0\rangle$ and the other half of the state vector represents entries in which q_0 is $|1\rangle$. This corresponds to a decision diagram node labeled q_0 connected to two successor nodes labeled q_1, representing the zero- and one-successor, respectively. This splitting process is conducted recursively until the nodes representing q_{n-1} are eventually connected to terminal nodes which contain the individual amplitudes. During the decomposition, equivalent sub-vectors can be represented by the same node, reducing the size of the generated decision diagram. Additionally, instead of having terminal nodes for all distinct amplitudes, edge weights are utilized to extract common factors of sub-parts and, thus, allow for even more compaction. The amplitude corresponding to a specific state can then be obtained by multiplying the edge weights along the desired path in the decision diagram. In order to improve the readability of decision diagrams, edge weights of 1 are typically omitted from the visualization and nodes with an incoming edge weight of zero are shown as 0-stubs to indicate that the whole sub-part is zero.

Example 3. *Fig. 1 shows a quantum state in both the vector as well as the decision diagram representation. The annotations of the state vector in Fig. 1a indicate how the corresponding decision diagram is constructed. In order to reconstruct the amplitude for a specific state from the decision diagram, the edge weights of the corresponding path need to be multiplied. For example, reconstructing the amplitude of the state $|11\rangle$ (bold lines in the figure), requires multiplying the edge weight of the root edge ($1/\sqrt{2}$) with the right edge of q_0 (1) as well as q_1 (1), i.e. $1/\sqrt{2} \cdot 1 \cdot 1 = 1/\sqrt{2}$.*

2) Representation of Quantum Operations: Quantum operations are represented in a similar way to state vectors. However, due to their square structure, matrices are split into four equally sized sub-parts as opposed to two in the case of vectors. Specifically, consider a quantum register composed of n qubits $q_0, q_1, \ldots, q_{n-1}$ and a quantum operation represented by the unitary matrix U of size $2^n \times 2^n$. Then, the matrix U is split into four sub-matrices, each of size $2^{n-1} \times 2^{n-1}$. These represent the transformation of q_0 by the quantum operation. Accordingly, a node labeled q_0 is drawn in the decision diagram and four sub-nodes are attached to it, representing the four sub-matrices, i.e. the sub-matrix of the upper left corner is represented by the first node, the upper right corner by the second node, the lower left sub-matrix by the third node and the lower right sub-matrix by the fourth node. As in the case of vectors, this process is repeated recursively until sub-matrices only consisting of single matrix entries remain. During the decomposition equivalent sub-matrices are again represented by the same node and edge weights are utilized to allow the extraction of common factors.

Example 4. *The decision diagram representations for the quantum operations from Section II-B are shown in Fig. 2.*

3) Decision-Diagram-based Simulation: Decision diagram-based quantum circuit simulation is conducted in a similar fashion as array-based simulation. A quantum operation U is applied to a state $|\psi\rangle$ by multiplication as defined in Definition 2. The multiplication needs to be decomposed with respect to the most significant qubit, leading to

$$\begin{aligned} |\psi_i'\rangle &= \sum_{x=0}^{2^n-1} u_{i,x} \cdot |\psi_x\rangle \\ &= \sum_{x=0}^{2^{n-1}-1} u_{i,x} \cdot |\psi_x\rangle + \sum_{x=2^{n-1}}^{2^n-1} u_{i,x} \cdot |\psi_x\rangle , \end{aligned}$$

or, using matrix notation,

$$U \cdot |\psi\rangle = \begin{bmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{bmatrix} \cdot \begin{bmatrix} \psi_0 \\ \psi_1 \end{bmatrix} = \begin{bmatrix} U_{00} \cdot \psi_0 + U_{01} \cdot \psi_1 \\ U_{10} \cdot \psi_0 + U_{11} \cdot \psi_1 \end{bmatrix}.$$

Thus, the resulting decision diagram consists of a top node with the zero-successor representing $U_{00} \cdot \psi_0 + U_{01} \cdot \psi_1$ and the one-successor representing $U_{10} \cdot \psi_0 + U_{11} \cdot \psi_1$. These successors are recursively decomposed in a similar fashion until only operations on complex numbers remain. Afterwards, the obtained sub-results are added together accordingly in order to obtain the resulting state vector. Hence, decision diagram-based simulation mainly involves recursive traversals of the involved decision diagrams. On top of that, further optimizations are possible with respect to the precision of the simulation [31] or the run-time performance [32].

IV. Case Study

In our case study, we compare the runtime as well as the required memory of state-of-the-art array- and DD-based simulators. Based on that, we gain detailed insights into their performance as well as the strengths and weaknesses of both approaches. In this section, the obtained results and, more importantly, the conclusions which can be drawn from them are summarized. To this end, we first describe the setup of the case study in detail; afterwards, the results and conclusions are provided.

A. Setup of the Case Study

As a state-of-the-art representative of an array-based simulator, we used the *Linalg* simulator from the commercial *Atos Quantum Learning Machine* (QLM) [15]. The QLM is an environment for quantum computing, which has been developed since 2016. Among other things, it comes with its own quantum simulators, circuit optimizers, dedicated hardware and a customized Linux kernel. In 2018, the QLM was the first simulator which allowed to simulate quantum circuits under consideration of noise effects [33]. The particular QLM instance we are using utilizes 96 cores running at a clock frequency of 2.2 GHz and 1.5 TB of RAM.

As a state-of-the-art representative of a DD-based simulator, we used the implementation taken from [34] (based on the principles of [21]). To allow for a fair comparison, we ported the DD-based simulator onto the QLM using Docker [35]. We decided to use Docker since its virtualization overhead is negligible [36]. Therefore, both simulators are running on the same hardware and have the same resources available. In practice however, this DD-based simulator currently does not make use of parallel executions (using concurrency for DD-based simulation is still an active field of research [37]), while the array-based simulator uses the full potential of the hardware resources (this way having 96 times more processor power at its disposal).

Finally, in order to properly evaluate the performance of both simulators, different types of benchmark circuits have been considered. This includes benchmarks representing certain quantum characteristics (namely the *Entanglement*-benchmarks, which construct entangled states), benchmarks realizing certain quantum algorithms (namely the quantum Fourier transform, i.e. QFT [24], Grover's algorithm for database search [2] and Shor's factorization algorithm [1]), as well as certain random circuits, designed by Google in their development towards *quantum supremacy* [38], which are engineered to be exceptionally hard to simulate classically.[2] All simulations were conducted using a variable number of qubits, allowing for a detailed evaluation of the scalability of both simulation approaches.

B. Obtained Results

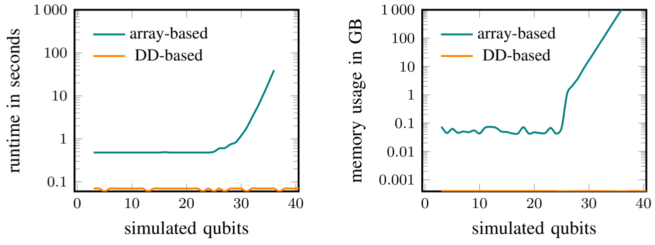
All obtained results are summarized in Figs. 3–7. The respective plots provide the runtime (left-hand side) as well as the memory consumption (right-hand side) for both approaches and for each benchmark (scaling with respect to the number of qubits). Simulations, which could not be conducted within one hour are omitted (i.e. the respective curve stops at the largest number of qubits that could be simulated within this time). In the following, we first discuss the obtained results for each benchmark separately. This provides the basis for our conclusions which are summarized afterwards.

We have decided to present our data in the form of logarithmic plots, since they give a more immediate impression of how the simulators behave for specific benchmarks. In the remainder of this section we describe the behavior of both simulators for each benchmark.

Considering the Entanglement-benchmarks (see Fig. 3) as well as the QFT-benchmarks (see Fig. 4), almost identical performance can be observed: The runtime and the memory consumption of the array-based simulator grows exponentially with respect to the number of simulated qubits. In contrast, the DD-based simulator shows linear growth in runtime and memory consumption. This limits the array-based approach to a maximum of 36 qubits, while the DD-based approach can actually be scaled to hundreds of qubits in these cases.

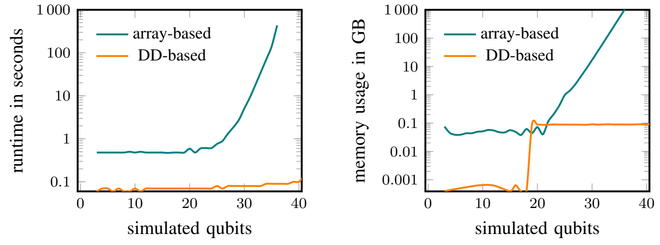
Considering Shor's algorithm (see Fig. 5), we observed fluctuations in the runtime of the DD-based simulator. On the one hand, this is due to the fact that multiple instances of Shor's algorithm with the same number of qubits but different integers to be factored were used during the evaluations. Naturally, the factorization of different integers leads to different computations during the simulation, which in turn result in decision diagrams of varying sizes. On the other hand, the implementation used in our evaluations contains intermediate measurements. While the array-based simulator does not make use of these measurements (it still stores the whole state vector), the decision diagram-based simulator exploits the post-measurement collapse of the quantum state to achieve compaction. Since the measurement process is inherently probabilistic (see Section II), its execution during the simulation depends on a given random seed—further explaining the varying runtimes. To properly reflect that in our results, we plot both, the best and worst, runs for the DD-based simulator in Fig. 5. Independently of this, we observe an exponential growth in all cases. The DD-simulator manages to simulate circuits composed of 30 qubits in its worst runs, which is also the limit of the array-based simulator. In the best case, the DD-based simulator can simulate Shor's algorithm with up to 34 qubits, making the DD-based approach the better choice for simulating this important algorithm.

[2] All benchmarks used in the simulation are provided at https://github.com/iic-jku/quantum_circuits.



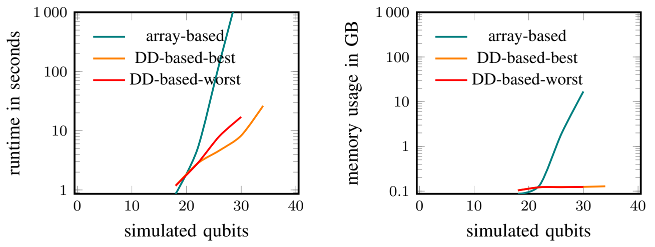
(a) Simulation time per qubit (b) Memory usage per qubit

Fig. 3. Entanglement circuit simulation



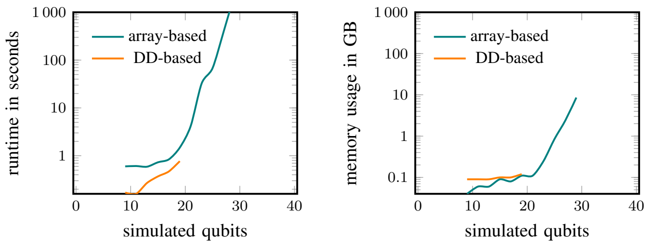
(a) Simulation time per qubit (b) Memory usage per qubit

Fig. 4. Quantum Fourier transform circuit simulation



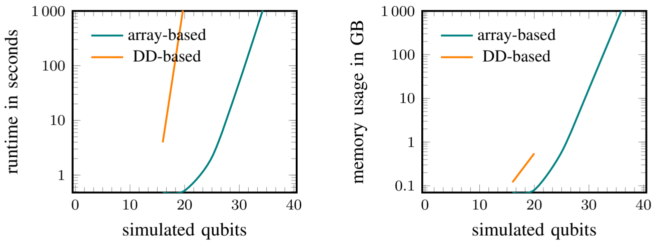
(a) Simulation time per qubit (b) Memory usage per qubit

Fig. 5. Shor's algorithm circuit simulation



(a) Simulation time per qubit (b) Memory usage per qubit

Fig. 6. Grover's algorithm circuit simulation



(a) Simulation time per qubit (b) Memory usage per qubit

Fig. 7. Quantum supremacy circuit simulation

Considering Grover's algorithm (see Fig. 6), both simulators exhibit exponential growth as well. But here, the array-based simulator manages to simulate more qubits than the decision diagram-based simulator (namely up to 29 qubits in the case of array-based simulation and up to 19 qubits in the case of DD-based simulation).

Finally, considering Google's quantum supremacy benchmarks (see Fig. 7), the array-based simulator performs significantly better than the DD-based simulator[3]. This is due to the specific structure of these benchmarks. They are specifically designed in a way so that almost no redundancies can be exploited. Thus, the overhead in DD-based simulation, incurred by deriving more compact representations through extracting common factors and sharing nodes, does not pay off in these cases. Instead, the array-based approach keeps following its straightforward scheme which allows for simulation up to a limit similar to the other benchmarks.

C. Resulting Conclusions

Based on the results discussed above, we draw the following conclusions:

1) Runtime Performance: The array-based simulator always exhibits an exponential runtime with respect to the number of simulated qubits, i.e. each additional qubit effectively doubles the runtime for the simulation. Compared to this effect, the depth of the simulated circuit, as well as the quantum operations, only play a minor role. In contrast, the runtime of the DD-based simulator strongly depends on whether it is possible to obtain a compact representation. If this is possible (such as in the Entanglement, QFT and the Shor benchmarks), simulation can be conducted very efficiently. Otherwise (such as in the Grover and the supremacy benchmarks), limits are reached quickly (in fact, in these cases, limits are reached faster than with the array-based approach).

2) Memory consumption: With respect to memory consumption, a similar performance as for the runtime can be observed: The array-based approach always exhibits an exponential memory consumption, whereas the memory consumption of the DD-based approach strongly depends on the benchmark and whether the quantum state can be compactly represented.

3) Dependency between Runtime and Memory: The array-based simulator displayed a strong correlation between the runtime and the memory consumption for all circuits we simulated. In fact, the curve progression of the runtime graph and the memory consumption graph is roughly the same for all benchmarks. The runtime of the DD-based simulator also corresponds with its memory, but not in such a direct relation. The DD-based simulator reacts more sensitively when the memory footprint (i.e. complexity of the quantum state) increases. Especially if the complexity exceeds some critical threshold, the runtime "skyrockets".

4) Predictability of Runtime and Memory Consumption: As we have discussed above, the memory consumption of the array-based simulator can be predicted reliably by the number of simulated qubits. Furthermore, the runtime of the array-based simulator strongly correlates with the memory consumption. Those two characteristics lead to the conclusion that the required resources of the array-based simulator can be predicted quite accurately prior to the simulation. However, this does not hold for the DD-based simulator. As long as the complexity of the simulated circuit does not cross some critical line, the DD-based simulator is very fast. Being able to forecast the complexity of DD-based simulation (e.g. the amount of available redundancy) prior to the simulation is non-trivial, but might allow for a good estimate of the DD-based

[3]In the plots, results are only provided for 16, 20, 25, 30 and 36 qubits, where the DD-based simulator was only able to simulate the first two instances.

simulator's performance.

V. Conclusion

In this paper, we conducted a case study comparing array- and DD-based quantum circuit simulation. By this, we shed light on the performance of these complementary simulation schemes. We observed that the memory consumption of the array-based simulator always growths exponentially with the number of simulated qubits. Since the runtime strongly correlates with the memory footprint, the resource consumption of the array-based simulator can be predicted reliably prior to the execution. However, this predictability paired with the exponential growth, severely limits the number of qubits array-based simulators can handle. By using a potentially more compact representation of the quantum state, DD-based simulation frequently allows to simulate more qubits using less memory. In contrast to array-based simulation the performance does not directly depend on the number of simulated qubits, but rather on the structure of the simulated circuit. Specifically, there is a direct correlation between the degree of compaction achieved during the simulation and its resource consumption. Gaining a better understanding on how this quantity may be estimated prior to the simulation is crucial for the effective utilization of this simulation approach in the future. Besides that, also questions on whether hybrid approaches are possible or whether concurrent approaches as well as approximation schemes can be exploited remain open issues for future work.[4]

Acknowledgments

This work has been supported by the University of Applied Sciences PhD program of the government of Upper Austria (executed by the FFG) as well as the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," vol. 26, no. 5, pp. 1484–1509, 1997.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996, pp. 212–219.

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

[4] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik, "Polynomial-time quantum algorithm for the simulation of chemical dynamics," *Proc. of the National Academy of Sciences*, vol. 105, no. 48, pp. 18 681–18 686, 2008.

[5] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte Carlo pricing of financial derivatives," *Phys. Rev. A*, vol. 98, 2018.

[6] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, "q-means: A quantum algorithm for unsupervised machine learning," *Proc. of the Neural Information Processing Systems*, 2019.

[7] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv:1411.4028*, 2014.

[8] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

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

[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] 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.

[12] G. Aleksandrowicz *et al.*, "Qiskit: An Open-source Framework for Quantum Computing," *Zenodo*, 2019.

[13] D. S. Steiger, T. Häner, and M. Troyer, "ProjectQ: an open source software framework for quantum computing," *Quantum*, vol. 2, p. 49, Jan. 2018.

[14] "Cirq: A python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits." 2019, github.com/quantumlib/Cirq.

[15] Atos SE, "Quantum learning machine," atos.net/en/products/quantum-learning-machine, 2016, Accessed: 2019-11-20.

[16] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *CoRR*, vol. quant-ph/0406196, 2004.

[17] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Physical review letters*, vol. 91, no. 14, p. 147902, 2003.

[18] B. Villalonga, S. Boixo, B. Nelson *et al.*, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Information*, vol. 5, no. 1, p. 86, 2019.

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

[20] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "High-performance QuIDD-based simulation of quantum circuits," in *Design, Automation and Test in Europe*, 2004, p. 21354.

[21] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2018.

[22] D. M. Miller, M. A. Thornton, and D. Goodman, "A decision diagram package for reversible and quantum circuit simulation," in *IEEE World Congress on Computational Intelligence*, 2006, pp. 8597–8604.

[23] 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.

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

[25] H. De Raedt *et al.*, "Massively parallel quantum computer simulator, eleven years later," *Computer Physics Communications*, vol. 237, pp. 47–61, 2019.

[26] P. Niemann, R. Wille, and R. Drechsler, "Efficient synthesis of quantum circuits implementing clifford group operations," in *Asia and South Pacific Design Automation Conf.*, 2014, pp. 483–488.

[27] 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.

[28] P. Niemann, R. Wille, and R. Drechsler, "Improved synthesis of Clifford+T quantum functionality," 2018.

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

[30] P. Niemann, R. Wille, and R. Drechsler, "Equivalence checking in multi-level quantum systems," in *Int'l Conf. of Reversible Computation*, 2014, pp. 201–215.

[31] A. Zulehner, P. Niemann, R. Drechsler, and R. Wille, "Accuracy and Compactness in Decision Diagrams for Quantum Computation," in *Design, Automation and Test in Europe*, 2019, pp. 280–283.

[32] 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, pp. 90–95.

[33] Atos SE, "Atos announces world first in quantum computing," atos.net/en/2018/press-release/general-press-releases_2018_04_09/atos-announces-world-first-quantum-computing, Accessed: 2019-11-20.

[34] A. Zulehner, S. Hillmich, and R. Wille, "How to efficiently handle complex values? Implementing decision diagrams for quantum computing," in *Int'l Conf. on CAD*, 2019.

[35] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux Jour.*, vol. 2014, no. 239, 2014.

[36] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software*, March 2015, pp. 171–172.

[37] S. Hillmich, A. Zulehner, and R. Wille, "Concurrency in DD-based quantum circuit simulation," in *Asia and South Pacific Design Automation Conf.*, 2020.

[38] S. Boixo, S. V. Isakov, V. N. Smelyanskiy *et al.*, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, 2018.

[39] A. Zulehner, S. Hillmich, I. Markov, and R. Wille, "Approximation of quantum states using decision diagrams," in *Asia and South Pacific Design Automation Conf.*, 2020.

[4] First results towards these questions are provided in [37], [39].