Stripping Quantum Decision Diagrams of their Identity

Aaron Sander* Ioan-Albert Florea* Lukas Burgholzer* Robert Wille*§
*Chair for Design Automation, Technical University of Munich, Munich, Germany
§Software Competence Center Hagenberg (SCCH) GmbH, Hagenberg, Austria
aaron.sander@tum.de, lukas.burgholzer@tum.de, robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/

Abstract*—Classical representations of quantum states and operations as vectors and matrices are plagued by an exponential growth in memory and runtime requirements for increasing system sizes. Based on their use in classical computing, an alternative data structure known as *Decision Diagrams* (DDs) has been proposed, which, in many cases, provides both a more compact representation and more efficient computation. In the classical realm, decades of research have been conducted on DDs and numerous variations tailored for specific applications exist. However, DDs for quantum computing are just in their infancy and there is still room for tailoring them to this new technology. In particular, existing representations of DDs require extending all operations in a quantum circuit to the full system size through extension by nodes representing identity matrices. In this work, we make an important step forward for quantum DDs by stripping these identity structures from quantum operations. This significantly reduces the number of nodes required to represent them as well as eases the pressure on key building blocks of their implementation. As a result, we obtain a structure that is more natural for quantum computing and significantly speeds up computations—with a runtime improvement of up to $70\times$ compared to the state-of-the-art.

***Index Terms*—Decision Diagrams, Quantum Computing, Quantum Circuit Simulation**

I. Introduction

Quantum computing is a promising new technology that is step-by-step becoming closer to reality and has the chance to propel our computational abilities forward to solve currently intractable problems. Similar to classical computing, algorithms on quantum computers can be decomposed into smaller operations known as *gates*. These gates form a *quantum circuit* analogous to digital circuits in which their combined operation on *quantum bits* (qubits) performs some computation. However, currently it is still necessary to use classical methods to simulate, verify, and compile these circuits. Unfortunately, these tasks become increasingly difficult due to the standard representation of quantum states and operations (namely vectors and matrices) grows exponentially relative to the number of qubits in the algorithm. Storing and manipulating these large vectors and matrices quickly becomes infeasible for classical computers—motivating both the need for a quantum computer to perform these computations as well as the need for further development of sophisticated classical methods to represent and work with states and operations in quantum circuits. Eventually, the capabilities of the most powerful classical simulators define the boundary of quantum advantage.

In the classical realm, the design automation community has spent decades to successfully develop solutions for tackling excessive memory requirements. One of these solutions is to use *decision diagrams* to represent information. Over the last decades, a plethora of different types tailored for different application scenarios has emerged [1]–[4]. Inspired by their success in the classical realm, decision diagrams have been adapted to the quantum realm in order to exploit both sparsity as well as redundancy in the underlying structures they represent—leading to significant compression in memory requirements and reduction in the runtime necessary to perform calculations [5]–[13]. However, these methods do not fully exploit that the underlying representations originate from a quantum context, which leaves a huge potential untapped.

In this work, we focus on quantum decision diagrams as defined in [12]. This type of DDs always requires the operands of any DD operation (such as multiplication or addition) to act on the same number of qubits—a reasonable assumption considering that, e.g., plain matrix addition also requires both matrices to have the same dimensions. This is accomplished by blowing up the DD representations of operations with identity nodes for any qubit that is not acted on. Since most quantum operations only feature a low number of qubits (typically one or two), this incurs a substantial overhead—not only in the sheer number of nodes but also in the pressure that is being put on key data structures such as compute and unique tables within the respective DD package. Additionally, these identity structures inherently do not play a role in the circuit as they do not perform any action.

Motivated by this fact, this work proposes a new DD structure that strips away these identities—leading to a significantly more compact representation that, simultaneously, better mimics the quantum gates that it represents. Experimental evaluations on DD-based statevector and unitary simulation demonstrate that this seemingly simple change has profound implications on the performance of the resulting DD package—resulting in an average speed-up of $7.7\times$ and up to an $70\times$ improvement compared to the state of the art. The resulting implementation is publicly available at https://github.com/cda-tum/mqt-ddsim as part of the *Munich Quantum Toolkit* (MQT;[14]).

The rest of this paper is structured as follows: Section II reviews the basics of quantum computing and decision diagrams. Based on that, Section III motivates the proposed idea of an identity-less DD structure—with detailed implementation changes described in Section IV and implications of this change discussed in Section V. Afterwards, Section VI presents and discusses the obtained experimental results, before Section VII concludes the paper.

II. Background

In order to keep this paper self-contained, this section briefly covers the basics of quantum computing used in the remainder of this work and reviews decision diagrams for representing and manipulating quantum states and operations.

A. Quantum Computing

While classical computing relies on bits (that can either be 0 or 1), quantum computing relies on *quantum bits* (or *qubits*) that can also be $|0\rangle$ or $|1\rangle$, but additionally in an arbitrary *superposition* of both computational basis states. The state $|\Psi\rangle$ of a single qubit is described as $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, with complex-valued *amplitudes* α_0 and α_1 such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. In general, the state $|\Psi\rangle$ of an n-qubit system is described by 2^n complex-valued amplitudes α_i that describe a linear combination of the computational basis states $|i\rangle$ for $i = 0, \dots, 2^n - 1$. Here, $|i\rangle$ can be thought of as the (classical) state corresponding to the bitstring of size n that is given by the binary expansion of the integer i. Again, these amplitudes are normalized such that $\sum_i |\alpha_i|^2 = 1$. A quantum state $|\Psi\rangle$ is typically represented by a complex-valued vector containing its amplitudes that is frequently referred to as the *statevector*.

Example 1. *Consider the following two-qubit quantum state* $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. *Then, the corresponding statevector is given by*

$$|\Psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}^T. \tag{1}$$

This state is known as a Bell state *and is the smallest example of an* entangled *quantum state, where the state of the individual qubits cannot be described separately any more.*

Similar to quantum states being represented by vectors, *quantum operations* (also called *quantum gates*) are represented by matrices that are unitary, i.e., $U^\dagger U = I$ with $U^\dagger$ denoting the conjugate transpose of U and I denoting the identity matrix. While n-qubit states require 2^n complex values, n-qubit operations require $2^n \times 2^n$ entries.

Example 2. *Common examples of single-qubit gates are the Pauli-X and H (Hadamard) gates. The X gate is analogous to a bit-flip, while the H gate is used to generate a superposition from a computational basis state. These, along with the identity operation, are defined in matrix form as follows*

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \text{ and } I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2}$$

One of the most common two-qubit gates is the Controlled-NOT *(CNOT) gate, which applies an X gate to a* target *qubit conditioned on a* control *qubit being $|1\rangle$. This corresponding unitary matrix is given by*

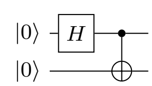
$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix}, \tag{3}$$

which, as shown, is equivalent to smaller 2×2 blocks corresponding to the identity I and X gate.

In general, a *quantum algorithm* is a unitary transformation and, hence, can itself be represented as a unitary matrix U that encodes the full functionality of the algorithm. The application of a quantum algorithm to a certain initial state is then conceptually equivalent to the matrix-vector multiplication of the algorithm's unitary U and the quantum state's statevector $|\Psi\rangle$, i.e., $|\Psi'\rangle = U |\Psi\rangle$ with $\alpha'_i = \sum_j u_{ij} \alpha_j |j\rangle$.

Since the size of these unitaries grows exponentially with the system size, it is hardly feasible and practicable to represent quantum algorithms in this form. Even more so, since actual quantum computers only offer a limited (yet universal) set of natively available gates that is typically limited to single- and two-qubit operations. As a consequence, quantum algorithms are predominantly described as sequences of smaller quantum gates that form a *quantum circuit*. A quantum algorithm and its circuit representation can be thought of as a direct analogue to high-level classical computations (such as addition) and the logic circuits representing them (such as an adder circuit).

Example 3. *Consider the following quantum circuit G*



(4)

that acts on two qubits and consists of two gates—a Hadamard gate applied to the top qubit and a CNOT gate controlled by the top qubit and targeted at the bottom qubit. Then, the functionality U of G is described by $U = (CNOT)(H \otimes I)$, where $\otimes$ corresponds to the Kronecker product*, used here to expand the H operation to the full system size with the identity matrix I so that the matrix-matrix multiplication can be applied. This results in the unitary*

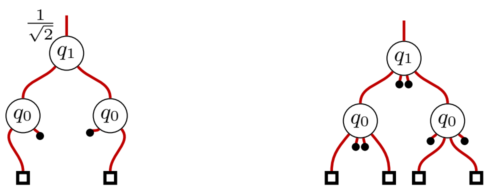
$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}. \tag{5}$$

Applying this unitary to the all-zero initial state $|00\rangle$, i.e., computing

$$U |00\rangle = (CNOT)(H \otimes I) |00\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{6}$$

yields the Bell state from Example 1.

While vectors and matrices are perfectly suitable for representing small-scale quantum systems on classical computers, the inherent exponential complexity quickly becomes prohibitive for larger system sizes. This motivates the need for



(a) Bell State (b) CNOT

Fig. 1: DDs for Example 1 and Eq. (3)

alternative methods to efficiently represent and manipulate quantum states and operations on classical computers, thus continuously pushing the boundary of what can currently be simulated and understood without an actual quantum computer.

B. Decision Diagrams

Decision Diagrams (DDs) have been proposed as one such alternative data structure [5]–[10], [12], [13], [15]. Inspired by their classical counterparts, commonly used to represent and manipulate Boolean functions in classical circuit design [1]–[4], the use of the underlying principles has recently been introduced as a tool for classical simulation, verification, and compilation of quantum circuits [5]–[8], [16]–[22]. In the following, we explicitly focus on DDs as described in [12] as the basis of this work[1]. Thereby, the main principle is the recursive subdivision of the underlying representations into subcomponents corresponding to individual qubits and explicitly exploiting sparsity and redundancy throughout this subdivision in conjunction with suitable normalization criteria.

For quantum states, this amounts to recursively halving the corresponding statevector until only scalar numbers remain. At each division, a node with two successors is introduced—the left successor representing the top half of the (sub)vector and the right successor representing the bottom half of the (sub)vector. In this splitting, the left successor always leads to an amplitude where the qubit corresponding to the current level in the DD is $|0\rangle$, whereas the right successor leads to amplitudes where that qubits is $|1\rangle$. Whenever a node is solely composed of zero-entries, it is removed and replaced by a *so-called zero-stub* indicating that anything along the respective path will lead to 0. DDs are a canonic data structure, so whenever two nodes have an identical structure, only one of them is ever actually represented and shared within a DD—reducing the overall resources required to represent the state.

Example 4. *The Bell state described in Example 1 is represented as the decision diagram Fig. 1a Herein, each level corresponds to a qubit in the system. Individual amplitudes are obtained by multiplying the edge weights throughout the tree along the path of a given computational state. For example, the amplitude of the $|00\rangle$ state is reconstructed starting at the top of the above DD and going left twice, leading to the computation $1 \times \frac{1}{\sqrt{2}} \times 1 = \frac{1}{\sqrt{2}}$.*

[1]Due to page limitations, the following descriptions had to be kept rather brief. We refer the interested reader to [12] and the references therein for an in-depth introduction to decision diagrams.

The decomposition of matrices follows a similar scheme in that the underlying unitary matrix is recursively quartered and nodes with four successors are created at each division. Here, the left-most successor corresponds to the top-left, the second to the top-right, the third to the bottom-left, and the right-most to the bottom-right quadrant.

Example 5. *The CNOT gate is represented by the decision diagram in Fig. 1b Again, each level corresponds to an interaction with a given qubit in the system. This is equivalent to the block decomposition as seen in Eq. (3).*

The unique selling point of DDs is that, instead of scaling with the number of entries in the underlying vectors or matrices, operations on decision diagrams (such as addition and multiplication) scale with the number of nodes in the respective DDs—a direct consequence of their recursive definition. That is, as long as these representations stay compact, DDs not only allow to compactly represent, but also to efficiently manipulate components relevant for classically conducting quantum computations.

III. Motivation and General Idea

The above description might make it seem that DDs for quantum computing are a rather mature data structure where everything is solved. However, compared to the decades of research on variations of classical DDs for dedicated classes of problems, theoretical bounds on their growth, as well as highly engineered software implementations, DDs for quantum computing are still in their infancy with many unanswered questions and lots of potential to improve the underlying concepts. Especially, since these methods do not fully exploit that the underlying representations originate from a quantum context. An example illustrates the untapped potential.

Example 6. *Say that we have a 100-qubit system and want to generate a Bell state between the first and the last qubit such that we generate the state*

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\big(|0\underbrace{\ldots 0 \ldots}_{98\ times}0\rangle + |1\underbrace{\ldots 0 \ldots}_{98\ times}1\rangle\big) \qquad (7)$$

The circuit used to generate this state as well as the DD representing every entity in it are shown in Fig. 2.

Observe how each gate DD contains identity nodes at levels which are not affected by the operation (drawn as wires). This is a direct consequence of the extension to the full system size previously observed in Example 3 that is used to make the dimensions of the respective quantities fit. As a consequence, the DD for the single-qubit Hadamard gate consists of 100 nodes while the DD for the CNOT gate is blown up to a total of 199 nodes (one at the control, 2 identities at each intermediate level, plus an identity and an X gate at the target level).

As the above example has shown, the extension to the full system size introduces a severe overhead for representing quantities that per-definition do not affect the circuit

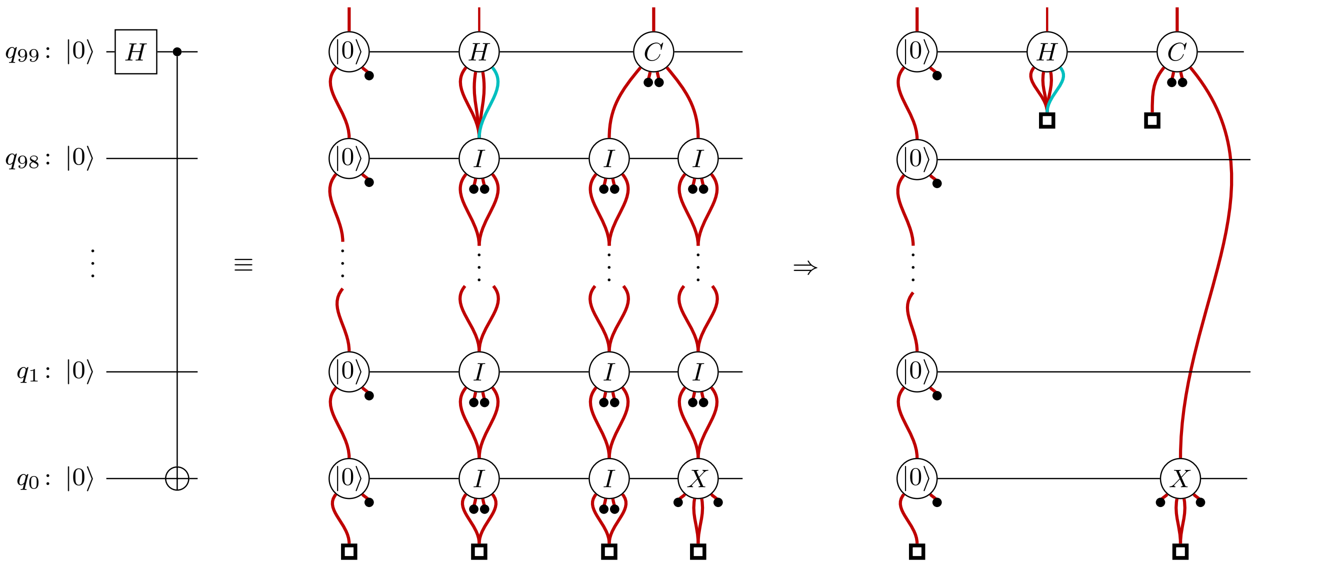


Fig. 2: Bell State between 0th and 99th qubit as described in Example 6 represented as a circuit, the current DD structure, and the new identity-less structure

functionality at all. For that reason, many sophisticated implementations of other techniques for statevector simulation directly manipulate the amplitudes of the state vector that are affected by an operation and never construct the full operation matrix (which, in many cases, could not even be feasibly represented due to its exponential size) [23]–[25]. To further advance the state of the art in decision diagrams for quantum computing, it is necessary to strip decision diagrams representing quantum operations of their identity nodes. This makes them both a more natural representation of quantum circuits as well as significantly reduces the node count needed in their implementation—especially for large systems. The following example demonstrates the profound implications that this proposed change brings along.

Example 7. *Say that we want to recreate the situation in Example 6, but completely strip away the identity nodes. Then, the operations become significantly more compact as illustrated in Fig. 2. Now, the single-qubit Hadamard gate is only represented by a single-level DD, while the two-qubit CNOT gate is represented by a two-level DD. Overall, this reduces the node count for these operations from* 100 *and* 199 *to* 1 *and* 2*, respectively.*

Algorithm 1 Gate DD Creation (simplified)

procedure GateDD(n, c, t, U)
  $[\, e_{00}\ e_{01}\ e_{10}\ e_{11} \,] \leftarrow$ Terminal($[\, u_{00}\ u_{01}\ u_{10}\ u_{11} \,]$)
  for l= 0,…,t-1 **do** ▷ Identities below t
    $e_{ij} \leftarrow$ Node(l, $[\, e_{ij}\ 0\ 0\ e_{ij} \,]$)
  $e \leftarrow$ Node(t, $[\, e_{00}\ e_{01}\ e_{10}\ e_{11} \,]$)
  for l= t+1,…,c-1 **do** ▷ Identities between t and c
    $e \leftarrow$ Node(l, $[\, e\ 0\ 0\ e \,]$)
  $e \leftarrow$ Node(c, $[\, \mathbb{I}_{c-1}1\ 0\ 0\ e \,]$)
  for l= c+1,…,n-1 **do** ▷ Identities above c
    $e \leftarrow$ Node(l, $[\, e\ 0\ 0\ e \,]$)
  return e

Algorithm 2 DD Node Creation (simplified)

procedure Node(l, $[\, e_{00}\ e_{01}\ e_{10}\ e_{11} \,]$)
  $e := (e_{\text{node}}, e_{\text{weight}}) \leftarrow$ (GetNode(), 1)
  $e_{\text{node}}.l \leftarrow l$
  e_{node}.edges $\leftarrow [\, e_{00}\ e_{01}\ e_{10}\ e_{11} \,]$
  $e \leftarrow$ Normalize(e)
  if ResemblesIdentity(e) **then**
    $s \leftarrow$ Successor(e, 0)
    Free(e_{node})
    return s
  return UTLookup(e)

IV. Implementation

While the change proposed above is seemingly small, the underlying assumption that two interacting DDs always have to act on the same number of levels is deeply rooted in all of the methods present in state-of-the-art realizations. Hence, this section discusses the key changes required to deliver on the promise of stripping DDs of their identity—specifically in (1) the creation of DDs for quantum operations, (2) the creation of DD nodes themselves, and (3) the DD operations such as multiplication and addition. For each of these, a simplified pseudo-code diff is provided to illustrate the change. To this end, additions will be marked in green while deletions will be marked in red. The interested reader is welcome to check out the open source implementation at https://github.com/cda-tum/mqt-core for further details.

The most obvious change lies in the method used to create DDs for quantum gates which, previously, had to be extended to the full system size by explicitly inserting identities for levels not acted on by the gate. For simplicity, we only consider the case of a two-qubit controlled-U gate (U being specified as a 2×2 unitary matrix) with control qubit c and target qubit t in an n-qubit system. We additionally assume that the control qubit comes before the target qubit in the variable order of the DD, i.e., $t < c < n$. This is not required by the implementation, but is simpler to illustrate the algorithm. Then, Algorithm 1 sketches the corresponding method and how it was adapted to not even create the identities. In the new implementation, the method only ever touches the levels the operation acts on—regardless of system size n.

However, it is not sufficient to simply avoid creating identity nodes during gate construction. Such nodes may naturally occur as the result of a computation, e.g., in lines 5 or 8 in Algorithm 1. Hence, it is also required to adapt the method for creating DD nodes given a level l and a list of successor DDs $[\, e_{00}\ e_{01}\ e_{10}\ e_{11} \,]$. This is shown in Algorithm 2. Compared to the original implementation, an additional check is introduced after the normalization that triggers if the normalized DD node resembles the identity, i.e., its first and last successor point to the same node with an edge weight of one, while the other successors are zero. If so, the newly created node is freed (as it is not needed) and the first successor is returned as a result of the call—effectively skipping the identity node.

Finally, the addition and multiplication routines were modified to account for the fact that two DDs that act as operands in these routines can no longer be assumed to always act on the same level due to potentially skipped nodes. For illustrative purposes, we consider the multiplication algorithm here that takes a matrix DD U, a vector DD v, as well as a level l and recursively computes the matrix-vector product Uv. The resulting algorithm is shown in Algorithm 3. Compared to the original version, it checks whether the matrix DD is at the correct level and implicitly treats the DD as an identity if it is not. In the following, the implications of the above changes on the overall methodology are discussed—both in theoretical limits as well as practical performance.

Algorithm 3 Matrix-Vector Multiplication (simplified)

procedure MULTIPLY(U, v, l)
  if ISZERO(U) or ISZERO(v) **then return** 0
  if ISIDENTITY(U) **then return** v
  if success, r $\leftarrow$ CTLOOKUP(U, v) **then return** r
  edges $\leftarrow [\, 0\ 0\ 0\ 0 \,]$
  for i, $j = 0, 1$ **do**
    if $U.l = l$ **then**
      $e_1 \leftarrow$ SUCCESSOR(U, $2i + j$)
    else $e_1 \leftarrow i = j \mathbin{?} x : 0$
    $e_2 \leftarrow$ SUCCESSOR(y, j)
    $m \leftarrow$ MULTIPLY(e_1, e_2, $l - 1$)
    edges[i] $\leftarrow$ADD(edges[i], m)
  $r \leftarrow$ NODE(l, edges)
  CTINSERT(U, v, r)
  return r

V. IMPLICATIONS

When developing data structures for quantum computing, the scalability with the number of qubits is one of the crucial criteria for determining a method's viability. In the previous iterations of decision diagrams, all gates on an n-qubit system must be scaled to n-level DDs. This is objectively a significant bottleneck for scalability, as infinite-level operations, i.e. ($n \to \infty$) would be required in the asymptotic limit. However, removal of the identity nodes means that the DDs representing gates are no longer connected to the overall number of qubits, but rather the number of qubits the gate acts upon. This localization gives credence to the theoretical capabilities of DDs in representing very large systems.

Stripping identity nodes in DDs for operations also directly leads to a reduction in the overall node count and, consequently, the memory required to represent the necessary quantities for a particular task. Instead of scaling with the overall system size, the memory requirements to store operations are entirely based on the number of qubits upon which they operate. As confirmed by the experimental evaluations, which are summarized in Section VI, this leads to a drastic reduction in the overall number of allocated nodes.

The reduction in the number of nodes also has direct consequences on the performance of key data structures within the DD package. This most significantly affects the *unique table*, which is used to check whether two DD nodes represent the same functionality and, thus, to ensure canonicity of the data structure. By reducing the number of nodes, there are significantly fewer lookups and inserts into this unique table (which is commonly implemented as a hash table for each variable). This implies that less cleanup (so-called *garbage collection*) is required to get rid of superfluous entries and guarantee the amortized $O(1)$ complexity for lookup and insertion. Since garbage collection is quite costly when it comes to runtime, this reduction in frequency constitutes a major performance improvement. A similar impact applies to the compute table.

VI. EXPERIMENTAL EVALUATIONS

In order to evaluate the impact of the newly proposed type of decision diagrams, we implemented the removal of identity nodes on top of the state-of-the-art decision diagram package publicly available as part of the MQT Core library (https://github.com/cda-tum/mqt-core). The resulting simulator is available as part of the *Munich Quantum Toolkit* (MQT; [14]) at https://github.com/cda-tum/mqt-ddsim. Then, we benchmarked the resulting implementation against the original package by performing statevector simulations (in which the gates are applied sequentially to the initial state) and unitary simulations (in which the functionality of the quantum algorithm is constructed directly) for a wide range of benchmarks—including generating the GHZ state, generating the W state, the Bernstein-Vazirani (BV) algorithm, Quantum

TABLE I: Experimental Results

	Circuit Simulation								Unitary Simulation							
			Old			New					Old			New		
	n	$\|G\|$	t [s]	$\|V\|$	$\|GC\|$	t [s]	$\|V\|$	$\|GC\|$	n	$\|G\|$	t [s]	$\|V\|$	$\|GC\|$	t [s]	$\|V\|$	$\|GC\|$
BV	256	636	<0.1	49 506	0	<0.1	382	0	256	636	0.1	180 645	1	<0.1	81 753	0
	512	1 271	0.3	328 054	2	0.1	761	0	512	1 271	0.4	788 888	6	0.2	327 378	2
	1 024	2 546	1.3	1 311 591	9	0.5	1 524	0	1 024	2 546	1.7	6 442 227	24	0.7	4 595 427	10
	2 048	5 104	6.9	5 252 986	39	2.7	3 058	0	2 048	5 104	8.7	25 854 080	97	3.7	7 420 378	40
	4 096	10 221	40.6	21 213 345	160	17.5	6 127	0	4 096	10 221	52.3	51 188 288	399	21.9	21 018 096	164
GHZ State	256	256	<0.1	33 151	0	<0.1	511	0	256	256	<0.1	125 792	0	<0.1	33 151	0
	512	512	0.1	132 350	1	<0.1	10 23	0	512	512	0.1	265 600	2	0.1	131 839	1
	1 024	1 024	0.5	529 915	4	0.2	2 047	0	1 024	1 024	0.5	1 049 600	8	0.3	525 823	4
	2 048	2 048	2.6	2 132 975	16	1.0	4 095	0	2 048	2 048	2.7	4 196 352	32	1.4	2 100 223	16
	4 096	4 096	15.6	8 660 926	65	6.6	8 191	0	4 096	4 096	16.3	17 875 872	131	8.0	8 394 751	64
W State	256	1 021	0.1	132 093	1	0.1	1 786	0	256	1 021	0.3	651 639	4	0.2	520 971	3
	512	2 045	0.6	527 156	4	0.3	3 578	0	512	2 045	1.2	2 614 852	19	0.9	2 090 512	15
	1 024	4 093	2.7	2 106 893	16	1.3	7 162	0	1 024	4 093	5.8	10 477 875	80	4.5	8 375 330	64
	2 048	8 189	14.4	8 440 874	64	7.1	14 330	0	2 048	8 189	32.0	41 970 068	322	24.5	33 527 949	257
	4 096	16 381	85.6	33 923 956	256	46.2	28 666	0	4 096	16 381	200.1	168 183 059	1299	159.6	134 164 915	1 041
Grover	28	1 019 143	0.1	45 146	0	0.1	42 668	0	28	1 019 143	0.2	9 580	0	0.2	7 345	0
	32	4 658 751	0.3	84 913	0	0.3	81 583	0	32	4 658 751	0.8	151 349	0	0.7	147 866	0
	36	20 964 167	1.2	156 491	1	1.1	152 108	1	36	20 964 167	2.6	266 061	1	2.5	260 593	1
	40	93 174 159	1.1	272 885	3	0.9	268 040	3	40	93 174 159	3.1	429 992	3	2.6	427 986	3
	42	195 665 483	1.5	238 364	1	1.3	233 159	1	42	195 665 483	1.6	336 721	2	1.4	328 534	2
QFT	256	33 024	1.0	1 435 826	10	0.1	20 380	0	18	180	8.4	527 414	4	0.3	524 591	4
	512	131 584	7.5	5 785 838	44	0.7	42 652	0	19	199	49.1	1 052 288	7	1.1	1 048 915	7
	1 024	525 312	67.1	23 251 301	176	3.5	87 196	0	20	220	338.8	2 101 530	11	6.4	2 097 529	11
	2 048	2 099 200	566.9	93 733 455	711	21.3	176 284	1	21	241	2 456.5	4 199 433	16	38.6	4 194 721	16
	4 096	8 392 704	4 683.8	380 202 897	2 870	134.5	354 460	2	22	264	18 855.5	8 394 578	22	267.8	8 389 067	22
QPE	15	134	0.2	1 254	0	0.2	225	0	8	43	<0.1	10 676	0	<0.1	10 507	0
	16	151	1.7	1 527	2	1.2	257	2	9	53	<0.1	41 760	1	<0.1	41 519	1
	17	169	27.3	1 835	3	9.1	291	3	10	64	0.3	165 727	2	0.2	165 387	2
	18	188	333.8	2 180	4	57.0	327	4	11	76	13.4	661 130	5	3.7	660 649	5
	19	208	2 239.5	2 564	5	332.8	365	5	12	89	667.8	2 643 492	8	82.9	2 641 229	8

n: Number of qubits $|G|$: Gate count t [s]: Runtime $|V|$: Node Count $|GC|$: Garbage Collection Runs

Fourier Transform (QFT), Quantum Phase Estimation (QPE), and Grover's algorithm. For both types of simulation, each algorithm is simulated for several different qubit counts n, while the gate count $|G|$, runtime t, overall count of matrix DD nodes $|V|$, as well as number of garbage collection runs $|GC|$ of the old and new implementation are recorded. These results are summarized in Table I.

The results align well with the theoretical and practical expectations laid out in Section V. As expected, removing the identity nodes yields an increase in performance across all benchmarks—resulting in an average speed-up of $7.7\times$ and up to a $70\times$ improvement compared to the state of the art. Furthermore, the overall node count required for the simulation is reduced, on average, by a factor of $218\times$ and up to $3462\times$. This reduction in the number of nodes allows the new implementation to perform fewer garbage collections, which, along with the reduction in node count and required operations, is a major factor for the drastic runtime decrease that can be observed for both types of simulation.

VII. Conclusion

By stripping quantum decision diagrams of their identity nodes, decision diagrams have been brought closer to a natural, more efficient representation of quantum operations. In this work, it has been shown that this change is not only theoretically motivated towards allowing DDs to push towards larger and larger systems, but also sees significant practical advantage compared to previous state-of-the-art implemenations. Due to the significant runtime and storage benefits presented here, it is expected that the structure of the decision diagrams outlined in this work will supplement previous implementations to become the de facto representation used to simulate, verify, and compile quantum circuits.

Acknowledgments

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 101001318), was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus, and has been supported by the BMWK on the basis of a decision by the German Bundestag through project QuaST.

References

[1] Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[2] S. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," in *Design Automation Conf.*, 1993, pp. 272–277.

[3] Randal E. Bryant and Yirng-An Chen, "Verification of arithmetic circuits using binary moment diagrams," *Software Tools for Technology Transfer*, vol. 3, no. 2, pp. 137–155, 2001.

[4] Tom van Dijk, Robert Wille, and Robert Meolic, "Tagged BDDs: Combining reduction rules from different decision diagram types," in *Int'l Conf. on Formal Methods in CAD*, 2017, pp. 108–115. DOI: 10.23919/FMCAD.2017.8102248.
[5] S.-A. Wang *et al.*, "An XQDD-based verification method for quantum circuits," in *IEICE Trans. Fundamentals*, 2008, pp. 584–594. DOI: 10.1093/ietfec/e91-a.2.584.
[6] A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Design, Automation and Test in Europe*, 2006.
[7] George F. Viamontes, Igor L. Markov, and John P. Hayes, "High-performance QuIDD-Based simulation of quantum circuits," in *Design, Automation and Test in Europe*, 2004.
[8] Yuan-Hung Tsai, Jie-Hong R. Jiang, and Chiao-Shan Jhang, "Bit-slicing the Hilbert space: Scaling up accurate quantum circuit simulation," in *Design Automation Conf.*, 2021, pp. 439–444. DOI: 10.1109/DAC18074.2021.9586191.
[9] D.M. Miller and M.A. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, 2006.
[10] Philipp Niemann *et al.*, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2016.
[11] Xin Hong *et al.*, "A Tensor Network based Decision Diagram for Representation of Quantum Circuits," *ACM Transactions on Design Automation of Electronic Systems*, vol. 27, no. 6, 60:1–60:30, 2022. DOI: 10.1145/3514355.
[12] Robert Wille, Stefan Hillmich, and Lukas Burgholzer, "Decision Diagrams for Quantum Computing," in *Design Automation of Quantum Computers*, 2023.
[13] Lieuwe Vinkhuijzen *et al.*, "LIMDD: A Decision Diagram for Simulation of Quantum Computing Including Stabilizer States," en-GB, *Quantum*, vol. 7, p. 1108, 2023, Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. DOI: 10.22331/q-2023-09-11-1108.
[14] Robert Wille *et al.*, "The MQT Handbook: A Summary of Design Automation Tools and Software for Quantum Computing," in *IEEE QSW*, 2024.
[15] Xin Hong *et al.*, *A tensor network based decision diagram for representation of quantum circuits*, 2020. arXiv: 2009.02618.
[16] Alwin Zulehner and Robert Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019. DOI: 10.1109/TCAD.2018.2834427.
[17] Lukas Burgholzer and Robert Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2021. DOI: 10.1109/TCAD.2020.3032630.
[18] Lukas Burgholzer, Hartwig Bauer, and Robert Wille, "Hybrid Schrödinger-Feynman simulation of quantum circuits with decision diagrams," in *Int'l Conf. on Quantum Computing and Engineering*, 2021. DOI: 10.1109/QCE52317.2021.00037.
[19] Kaitlin N. Smith and Mitchell A. Thornton, "Quantum logic synthesis with formal verification," *IEEE Int. Midwest Symp. on Circuits and Systems*, 2019. DOI: 10.1109/MWSCAS.2019.8885132.
[20] Robert Wille *et al.*, "The basis of design tools for quantum computing," in *Design Automation Conf.*, 2022.
[21] Thomas Grurl, Jurgen Fuß, and Robert Wille, "Noise-aware quantum circuit simulation with decision diagrams," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 42, no. 3, pp. 860–873, 2023. DOI: 10.1109/TCAD.2022.3182628.
[22] Aaron Sander, Lukas Burgholzer, and Robert Wille, "Towards Hamiltonian Simulation with Decision Diagrams," *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 283–294, 2023, Conference Name: 2023 IEEE International Conference on Quantum Computing and Engineering (QCE) ISBN: 9798350343236 Place: Bellevue, WA, USA Publisher: IEEE. DOI: 10.1109/QCE57702.2023.00039.
[23] Hans De Raedt *et al.*, "Massively parallel quantum computer simulator, eleven years later," *Computer Physics Communications*, vol. 237, pp. 47–61, 2019. DOI: 10.1016/j.cpc.2018.11.005.
[24] Thomas Häner and Damian S. Steiger, "0.5 petabyte simulation of a 45-Qubit quantum circuit," in *Int'l Conf. for High Performance Computing, Networking, Storage and Analysis*, 2017.
[25] Gian Giacomo Guerreschi *et al.*, "Intel Quantum Simulator: A cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, vol. 5, no. 3, p. 034 007, 2020. DOI: 10.1088/2058-9565/ab8505.