Mixed-Dimensional Quantum Circuit Simulation with Decision Diagrams

Kevin Mato* Stefan Hillmich†‡ Robert Wille*†
*Chair for Design Automation, Technical University of Munich, Munich, Germany
†Software Competence Center Hagenberg (SCCH) GmbH, Hagenberg, Austria
‡Institute for Integrated Circuits, Johannes Kepler University Linz, Linz, Austria
kevin.mato@tum.de, stefan.hillmich@scch.at, robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/

***Abstract*—Quantum computers promise to solve several categories of problems faster than classical computers ever could. Current research mostly focuses on qu*b*its, i.e., systems where the unit of information can assume only two levels. However, the underlying physics of most (if not all) of the technological platforms supports more than two levels, commonly referred to as qu*d*its. Performing computations with qudits increases the overall complexity while, at the same time, reducing the number of operations and providing a lower error rate. Furthermore, qudits with different number of levels can be mixed in one system to ease the experimental control and keep representations as compact as possible. Exploiting these capabilities requires dedicated software support to tackle the increased complexity in an automated and efficient fashion. In this paper, we present a qudit simulator that handles mixed-dimensional systems based on *Decision Diagrams* (DDs). More precisely, we discuss the type of decision diagram introduced as underlying data structure as well as the resulting implementation. Experimental evaluations demonstrate that the proposed solution is capable of efficiently simulating mixed-dimensional quantum circuits, with specific use cases including more than 100 qudits in one circuit. The source code of the simulator is available via github.com/cda-tdum/MiSiM under the MIT license.**

***Index Terms*—quantum computing, qudits, simulation**

I. Introduction

Quantum computing utilizes a different computing paradigm that promises to solve several categories of problems compared to classical computers. Examples include Shor's algorithm [1] to factorize integers, Grover's search [2] for unstructured data, and evaluating possible materials as catalysts in quantum chemistry [3]. Around the globe, many research groups in academia and industry (such as Google, IBM, and Microsoft) try realize this potential and push on what technology can achieve today.

However, thus far, the considered applications where mostly limited to systems composed of two-dimensional qubits. This neglects a large potential available through systems of higher dimensions, inherently possessed by almost any underlying technology of physical realizations of quantum computers. The usage of higher dimensions increases the overall complexity of building circuits but it also enables compression of previously costly non-local gates between qubits into local gates on a single qu*d*it—increasing the fidelity of the resulting state [4].

The abstract idea for higher-dimensional systems and corresponding theory has been around for quite some time [5]. Fundamentally, qudits enable denser storage of information and provide a much larger set of possible operations compared to qubits. Given these advantages, basic control has been demonstrated in physical platforms such as trapped ions [4], [6], to photonic systems [7]–[10], superconducting circuits [11], [12], Rydberg atoms [13], nuclear spins [14], cold atoms [15], nuclear magnetic resonance systems [16] and molecular spin [17].

Recent developments in quantum algorithms have shown that multi-level logic is a more natural architecture for implementing complex applications [18], [19]. Simulations of models representing fermion-boson interactions on mixed-dimensional quantum computers could enable real-time simulations of quantum electrodynamics and other field theories with continuous or larger symmetry groups [20]–[22]. Gate decompositions on mixed-dimensional systems offer reduced complexity due to the temporary expansion of the Hilbert space [23]. This leads to smaller circuits and a higher chance of success due to less noise accumulation. Optimizing the usage of qudits of different dimensions further improves circuit compactness and error rate, as seen in recent research.

However, given these recent breakthroughs from physicists, there is the risk of an emerging *design gap*, where powerful multi-dimensional quantum computers are available but we do not have means to utilize their power. To avoid this situation, dedicated methods and software support are required to keep up in the design automation domain before the point is reached, where manually designing systems, circuits, and controls is not tractable anymore.

We contribute to that by presenting a classical simulator for mixed-dimensional quantum circuits, i.e., circuits where each qudit may have a different dimensionality. To this end, we propose an extension to edge-weighted Decision Diagrams [24]–[26] which serve as the main data-structure to cope with the (exponential) complexity of simulation. In the past, decision diagrams have been proven to be a suitable data structure to compactly represent exponentially-sized data in many cases [24], [25], [27]–[30]. The type of decision diagram proposed in this work dynamically captures the dimensionality of each qudit in the mixed-dimensional system to minimize the required memory. Further, it elegantly visualizes the individual dimensionalities of the qudits. The experimental evaluation on a set of benchmarks confirms the efficacy of the simulator (publicly available at github.com/cda-tum/MiSiM) as tool for design automation in mixed-dimensional systems.

The remainder of this paper is structured as follows. Section II provides the necessary background on quantum states and operations for higher-dimensional systems, i.e., qudits.

Section III motivates the problem of quantum circuit simulation and details the contributions of this paper. Section IV gives an overview of the state of the art in quantum circuit simulation. Section V describes the proposed type of decision diagram for mixed-dimensional system and Section VI details the corresponding implementation. Section VII summarizes the experimental evaluation. Finally, Section VIII concludes the paper.

II. Background

In this section, we briefly review the basics of quantum information processing with a focus on mixed-dimensional quantum logic and how these concepts scale to the abstraction of quantum circuits.

A. Quantum Information Processing

In classical computing, *bit* (binary digits) are the primary unit of information, which can only exist in either the 0 or 1 state. In quantum computing, *qubits* (quantum bits) are the corresponding unit of information. The key difference from classical computing is that qubits can exist in any linear combination of $|0\rangle$ and $|1\rangle$ (using Dirac's bra-ket notation [31]). However, constructing qubits involves restricting the natural multi-level structure of the underlying physical carriers of quantum information.

Therefore, these systems natively support multi-level logic with the fundamental unit of information termed a *qudit* (quantum digit). A qudit is the quantum equivalent of a d-ary digit with $d \geq 2$, whose state can be described as a vector in the d-dimensional Hilbert space $\mathcal{H}_d$. The state of a qudit can thus be written as a linear combination $|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle + \ldots + \alpha_{d-1} \cdot |d-1\rangle$, or simplified as vector $|\psi\rangle = \begin{bmatrix} \alpha_0 & \alpha_1 & \ldots & \alpha_{d-1} \end{bmatrix}^{\mathrm{T}}$, where $\alpha_i \in \mathbb{C}$ are the amplitudes relative to the orthonormal basis of the Hilbert space—given by the vectors $|0\rangle, |1\rangle, |2\rangle, ..., |d-1\rangle$.

The squared magnitude of an amplitude $|\alpha_i|^2$ defines the probability with which the corresponding basis state i will be observed when measuring the qudit. Since the probabilities have to add up to 1, the amplitudes have to satisfy $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$.

Two key properties that distinguish quantum computing from classical computing are superposition and entanglement. A qudit is said to be in a *superposition* of states in a given basis when at least two amplitudes are non-zero relative to this basis. *Entanglement*, on the other hand, describes a form of superposition born from interactions in multi-qudit systems. Entanglement is a powerful form of quantum correlation, where the quantum information is encoded in the state of the whole system and cannot be extracted from the individual qudits anymore.

Example 1. *Consider a system of one qudit with only three energy levels (also referred to as* qutrit*). The quantum state* $|\psi\rangle = \sqrt{1/3} \cdot |0\rangle + \sqrt{1/3} \cdot |1\rangle + \sqrt{1/3} \cdot |2\rangle$ *is a valid state with equal probability of measuring each basis. Equivalently, the quantum state may be represented as vector* $\sqrt{1/3} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$.

In a similar fashion, quantum systems of mixed dimensions *can be constructed. Extending the previous qutrit state by a qubit enables representation of the following entangled state* $|\psi'\rangle = \sqrt{1/3} \cdot |0\rangle_3 |0\rangle_2 + \sqrt{1/3} \cdot |1\rangle_3 |1\rangle_2 + \sqrt{1/3} \cdot |2\rangle_3 |0\rangle_2$*—equivalently represented by the vector* $\sqrt{1/3} \cdot \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$.

B. Quantum Operations

The state of a single d-level qudit system can be manipulated by operations which are represented in terms of $d \times d$-dimensional unitary matrices U, i.e., matrices that satisfy $U^\dagger U = U U^\dagger = I$. This property makes quantum operations logically reversible. Quantum operations can be divided in two categories: local and non-local–entangling operations. Common examples of local operations are the Pauli operations

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & \omega^2 \end{bmatrix}, \tag{1}$$

with $\omega = e^{2\pi i/d}$ and d being the dimension of the single qudit. The local operations shown in Eq. (1) are the generalization of the qubit operations to a multi-level systems, in this particular case a *qutrit*.

Qudit systems can be entangled and, for this reason, they support a set of non-local operations, although these last operations cannot be simply derived by generalizing entangling qubit operations. Realizing entangling operations in qubit systems is comparatively simple and rather unclear in qudit systems. More precisely, for qubit systems it is sufficient to compile the *CNOT* gate to the native operations of the quantum hardware, because all qubit entangling operations can be implemented by the controlled-*NOT* (*CNOT*) gate and appropriate local operations on the subsystems [31]. In contrast, for qudit systems this does not hold anymore and entanglement can be generated in many in-equivalent ways. Consequently, while any single entangling gate is sufficient for universal quantum computation [32], not all entangling gates are equally useful for any given application.

Example 2. *Consider, the controlled-exchange gate CEX [4] applied on two qudits and defined by*

$$\mathit{CEX}_{c,t_1,t_2} : \begin{cases} \mathit{Swap}(t_1, t_2) & \mathit{if\ control\ is}\ c \\ \mathit{Identity} & \mathit{otherwise} \end{cases}. \tag{2}$$

This qudit-embedded version of the CNOT gate generates qubit-level entanglement in a high-dimensional Hilbert space. However, there are gates that directly generate qudit entanglement, such as the controlled-SUM gate defined by

$$\mathit{CSUM} : |c, j\rangle \mapsto |i, i \oplus j\rangle, \tag{3}$$

where $\oplus$ *denotes addition modulo the dimension* d*.*

These are just two examples of a more general theme in qudit systems, where entangling gates differ in their *entangling power*. The reasons why non-local qudit gates produce more entanglement than qubit ones are presented in depth in Ref. [33].

C. Quantum Circuit

After discussing how operations are represented for qudits and mixed-dimensional systems, it is relevant to discuss how to apply a quantum operation, called gates in the quantum circuit

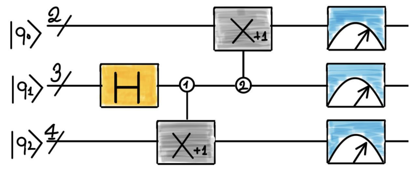


Figure 1: A mixed-dimensional circuit with three qudits

representation, and how algorithms for quantum systems can be represented as well. The matrices representing quantum operations describe the evolution of a quantum system. For this reason, the application of a U operator can be determined by multiplying the corresponding input state from the left with the matrix U, and the output state is the final state of the evolution imposed by U.

Example 3. *Consider a three-level qudit (i.e., a qutrit) initially in the state $|0\rangle$. Applying the Hadamard operation H_3 to it yields the output state shown before in Example 1, i.e.,*

$$H_3 \cdot |0\rangle = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{2\pi}{3}} & e^{\frac{-2\pi}{3}} \\ 1 & e^{\frac{-2\pi}{3}} & e^{\frac{2\pi}{3}} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Given multiple qudits of different dimensions and several operations applied to them according to an algorithm, we can illustrate the sequence as a *quantum circuit*. For a quantum circuit consisting of multiple mixed-dimensional qudits, the unitary matrix will be of dimension $\prod_i d_i \times \prod_i d_i$ with d_i denoting the dimension of each qudit. The application of the operation will require an input state of the size of $\prod_i d_i$ entries. Later on, adjustments to the matrices to match the size of the system are going to be discussed in more depth.

Example 4. *The quantum circuit in Figure 1 shows three lines, in order: a qubit, a qutrit (3 levels), and a ququart (4 levels), as well as three different types of operations, a local operation, two controlled operations, and measurements. The controlled operations on qudits apply a unitary to a target line only if the control qudit is in a $|i\rangle$ between 0 and $d-1$, with d dimensions of the control qudit. The control state is marked inside the circle on the control line. The first operation is a Hadamard applied to the qutrit. Afterwards a controlled-on-1 Pauli X (normally addressed as CNOT) is applied to the ququart and lastly to the qubit. Finally, the three units of information are measured in order to derive the outcomes of the circuit. The state of the system after the application of each operation can be viewed as* intermediate*, as the output state of one is the input state of another.*

III. Motivation

In this work, we investigate how to efficiently simulate quantum circuits where the qudits can have different dimensions, referred to as mixed-dimensional systems. To this end, the previous section provided the basis for understanding quantum information, with a particular focus on multi-level quantum logic, i.e., systems with both qubits and qudits. In this section, we review the design automation task of quantum simulation that is going to enable the utilization of mixed-dimensional systems. Following that, we discuss the challenges of simulating quantum computations and the benefits of simulating quantum circuits of mixed dimensionality.

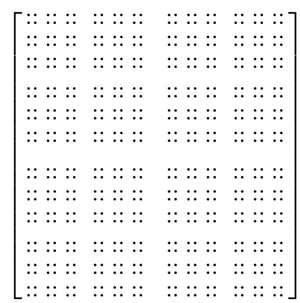
A. Simulation

Simulation of quantum circuits is an important task in the domain of design automation. In essence, the simulation of a quantum circuit is conducted by successively applying operations in sequence to an initial quantum state, i.e., multiplying matrices and vectors. While quantum circuit simulation is simple, the exponential memory requirements in the number of qubits make it hard in practice. With qudits of higher dimensions, this memory requirement becomes even higher.

More precisely, given a system with N qudits of possibly mixed dimensions, a vector describing this system will have length $D = \prod_{i=0}^{N-1} d_i$, with d_i denoting the dimension of each qudit. Correspondingly, the matrices representing the operations on this system will have dimension $D \times D$. In fact, even for operations that affect only a subset of qudits, they have to be "padded" with the appropriate identity operations to ensure the correct (large) size (using the Kronecker product [31]).

Example 5. *Consider the Hadamard gate on a qutrit as shown in Figure 1, also denoted H_3, affecting the second of the three qudits. Applying this operation leaves the first and third qudit untouched, i.e., it applies the identity operation of appropriate dimension. This is achieved by combining the Hadamard operation and identity operations via the Kronecker product as in the following illustration:*

$I_2 \otimes H_3 \otimes I_4 =$



Here, each dot represents an element of the combined operation. The result is a matrix with 24×24 entries.

Example 5 should convey an intuition about the rapid increase in size for state vector and operation matrices with respect to the number of qudits and their respective dimensionality. Consequently, the simulation of quantum computations requires an asymptotically exponential amount of memory with respect to the number of qudits. Single measurements of a quantum state suffer from the same complexity, but repeated measurements can be done in amortized linear time [34].

The memory complexity can be lowered quite significantly for many cases, especially for common single-qudit operations. In essence, the operation can be applied to the state vector directly, e.g., by swapping rows for the exchange-operation. Further, quantum states with many zero entries may be represented by sparse vectors [35]. However, the efficacy of this approach diminishes with operations affecting an increased number of qudits and quantum states that are not sparse.

Example 6. *Consider the* GHZ state *of two qutrits, i.e., qudits of dimension three, as an example of an entangled state, given as* $|\psi\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |11\rangle + |22\rangle)$. *The corresponding state vector is* $\psi = \frac{1}{\sqrt{3}}[1\,0\,0\,0\,1\,0\,0\,0\,1]$. *Here, only three out of nine entries are non-zero.*

B. The Case for Quantum Circuit Simulation

As highlighted in the previous section, the simulation of quantum circuits is conceptually simple but not easy in its execution. This where *design automation* comes into play to enable realizing the impact of high-mixed-dimensional systems. There is a great need for new automated methods, software frameworks, and theory, where only preliminary results have been achieved [36]–[40]. In this section, we make the case for utilizing the domain knowledge of design automation to overcome the exponential complexity in many cases. In fact, the need for better solutions to quantum simulation lies not only in the capability of understanding quantum computations but in enabling the study of new applications and new features created by using mixed-dimensional systems.

There are three immediate advantages in having an appropriate simulator for mixed-dimensional quantum systems.

1) Getting otherwise opaque information about the quantum state.
2) Enabling design exploration.
3) Aiding in verification.
4) Identifying potential for compression.

The inner processes of a quantum computer are fundamentally opaque. Trying to "look at" a quantum state will inadvertently make it decay into a basis state – immediately destroying any superposition or entanglement. However, we can accurately calculate the exact information of a quantum state by classical means. Of course, this is only possible for smaller systems due to the exponential overhead in the number of qudits. Nevertheless, classical simulation is an important tool for algorithm development and debugging. This becomes even more important when targeting higher- and mixed-dimensional systems. Their increased complexity also effects the developers of these system, making automated software support a necessity to reduce errors and speed up the design.

The second motivation is that simulation can take on the role of a tool for design exploration. In fact, in qudit systems, it is overall possible to explore new trade-offs between using higher-levels in the single qudit and using ancilla lines. The first one is similar to the effect of using ancilla qubits, but with a simpler circuit complexity and hardware design [23]. On mixed-dimensional systems this can be achieved with a temporal expansion of the Hilbert space. This leads to smaller circuits that have a higher chance of succeeding due to the smaller noise accumulated. However, just increasing the dimension of all qudits in a quantum circuit still leaves room for improvement [41], [42]. A simulator enables the study of different circuits, composed of more or less ancilla lines, depending on the limitations and potential of the implementing platform. This can lead to a more accurate study of suitable circuits for dedicated platforms and applications.

Classical simulation of quantum circuits also aids in verification schemes. On the one hand, it helps circuit equivalence checking by simulating two circuits with identical initial states and comparing the output. This is an easier task than constructing the matrices and comparing these. Such schemes that incorporate quantum circuit simulation have been proposed [43], [44]

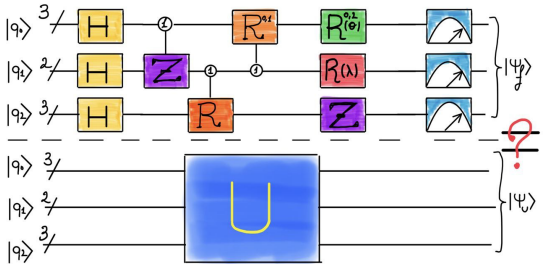


Figure 2: Verification of a quantum circuit given the original algorithm, or the unitary representing the evolution.

Finally, it is possible to see through simulations that the computation may happen only on specific levels of a qudit. This could allow the compression of a single qudit's computational space from a higher dimension to a lower dimension restricted to only the useful levels [45]. By this, the representations become more compact, reflecting into a competitive error rate and experimental control, typical of the state of art of qudit systems [4]. The simulation of mixed-dimensional systems opens new possibilities in the field of circuit compression. Two examples illustrate the ideas.

Example 7. *The use of a simulator could be of use for verifying that a compiled quantum circuit, a new decomposition, or a particular encoding preserves the intended original functionality defined in the algorithm. In Figure 2, this is illustrated using a simple use case, where a unitary has been compiled to a mixed-dimensional system and the simulation can tell us if, given an initial input state, the final states of the two are identical. If true, the circuit performs correctly the intended algorithm.*

Example 8. *Consider the following operation* U*:*

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & e^{\frac{2\pi}{3}} & e^{\frac{-2\pi}{3}} & 0 & 0 \\ 0 & e^{\frac{-2\pi}{3}} & e^{\frac{2\pi}{3}} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

It is possible to note that the local operation is operating only on two levels of the original five-level system (i.e., of a ququint). Taken into account that many operations are applied to only a subset of the available levels, we can consider the original system to be of only two dimensions, and consequently remap the logic levels to $|0\rangle$ *and* $|1\rangle$*. This results in a more compact operation of dimension* 2×2*, namely*

$$U' = \begin{bmatrix} e^{\frac{2\pi}{3}} & e^{\frac{-2\pi}{3}} \\ e^{\frac{-2\pi}{3}} & e^{\frac{2\pi}{3}} \end{bmatrix}. \tag{5}$$

C. Contribution

Motivated from the above, we present a mixed-dimensional quantum circuit simulator, based on decision diagrams, with a publicly available implementation. The aim is to investigate a new method for an efficient simulation of circuits with qudits of arbitrary and mixed dimensionality.

The proposed approach is motivated by the success of the decomposition schemes used in [24], [26], that fall under the family of *Decision Diagrams* (DDs; [24], [25], [27]–[30]) used in various fields of design automation, especially in simulation, verification, and synthesis. Further, the structure of directed acyclic graphs, which resemble tree, can elegantly illustrate the dimensionalities, interactions, and entanglement in mixed-dimensional systems.

This will unlock the potential benefits of this generalized type of quantum circuit by exploiting more compact representations for quantum states and operations on mixed-dimensional systems. In the following sections, we provide a review of the current state of art in quantum circuit simulation, introduce decision diagrams and their benefits, and describe in detail our proposed representations and required manipulation algorithms.

IV. State of the Art

There has been an ongoing and significant interest from researchers and engineers in developing solutions for simulating quantum computations. There are several available solutions that individually make use of different data structures, e.g., arrays [46]–[49], decision diagrams [26], [29], tensor networks [50], [51], and matrix-product states [51]. However, the vast majority of these simulators have been developed so far with the focus on qubit systems and their related circuits, while qudit systems have not been provided with the same software support. Qubit simulators can be divided in two main categories.

The first category of simulators is referred to as *array-based*. These simulators have limitations due to their reliance on the straightforward representation of quantum states and operations. Typically, simple 1-dimensional and 2-dimensional arrays are used. To simulate larger quantum systems, they require massive hardware power, such as HPC infrastructures composed of thousands of nodes, and petabytes of distributed memory. The utilization of accelerators and dedicated hardware can improve the run-time of the matrix-vector multiplications.

The second category of simulators relies on specialized data structures. For example, solutions based on decision diagrams [26], [29], have been proposed to exploit redundancies to gain a more compact representation of state vectors and matrices. Simulators based on tensor networks, such as those found in popular software packages [52], [53], share a similar goal of simulating quantum systems. However, they achieve this goal by compressing the information using a network of tensors that are contracted together in a specific way. One example of such a simulation method is the *Matrix Product States* (MPS), which efficiently calculates physical properties of one-dimensional quantum systems [51]. There are also examples of simulators using a combination of matrix-vector multiplication and tensor networks as qsim [50], which supports a variety of circuit models.

However, the available implementations offer preliminary support of higher-dimensional systems at best, if at all. There are early examples of trials to qudit simulations of quantum circuits of homogeneous dimensions, that peaked with a proof-of-concept prototype of QMDDs [24]. More recent attempts come from the established framework for quantum circuits manipulation and simulation, Google's Cirq [50]. It provides the core functionalities for the simulation of quantum circuits but requires the scientists to explicitly integrate the software with the desired gate set and special circuit constructions. Hence, unfortunately, Cirq supports mixed-dimensional quantum circuit simulation only in principle. The recently published qudit simulator QuDiet [54] provides a qubit-qudit quantum simulator package with quantum circuit templates of well-known qubit quantum algorithms for fast prototyping and simulation. QuDiet shares several similarities with Cirq and qsim [50] in terms of software structure and philosophy. So far, the existing simulators are focused on providing interfaces for the development of specific use cases rather than being ready-to-use qudit simulators. This lack of availability of the current software platforms and subsequent lack of automated methods is a major obstacle in development for higher- and mixed-dimensional systems.

V. Decision Diagrams

A key aspect of developing an efficient simulator is to conquer the correspondingly resulting exponential complexity for as many cases as possible. In the past, decision diagrams have been proven to enable efficient representation of exponentially-sized data in many cases [24]–[30], [55], [56]. In essence, they achieve their compactness by exploiting redundancy in data they represent. This section introduces the decision diagrams for representing quantum states and operations of mixed-dimensional systems.

A *Decision Diagram* (DD) is a *Directed Acyclic Graph* (DAG), composed of nodes and directed edges. The nodes are organized in levels, which each level representing one qudit. The edges represent the connection between the qudits and have additional information annotated to them. This construction can represent quantum states and quantum operations as detailed in the following sections.

A. Quantum States

The general idea of using decision diagrams to represent quantum states is based on repeated decompositions of the corresponding vectors. To this end, consider an n-qudit quantum state $q_{n-1}, q_{n-2}, \ldots, q_0$, where q_{n-1} is arbitrarily designated the "most sigificant qudit". This state is split into equally-sized parts based on the dimensionality of qudit q_{n-1}. Each part is represented by a successor node which also encodes the decision for the value of q_{n-1}. The splitting procedure is then repeated for each part until the individual complex entries in the original vector are reached. During this process, the complex amplitudes of each basis state are stored in the edge weights and equal sub-vectors (up to a complex factor) are represented by the same nodes. To guarantee canonicity, the nodes are normalized such that the sum of squared magnitudes of out-edges of a node add up to one. The resulting decision

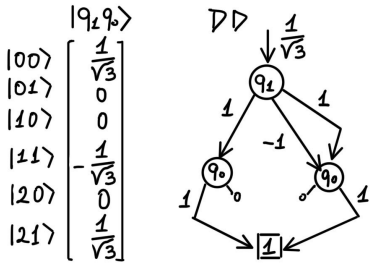


Figure 3: A state vector of a qutrit-qubit entangled state and the corresponding DD representation.

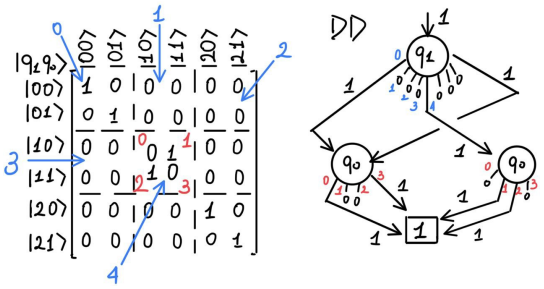


Figure 4: A unitary matrix applying the a *CEX*, controlled on the level $|1\rangle$ of a qutrit applied to $|0\rangle$ and $|1\rangle$ of a qubit.

diagram has a *root node* q_{n-1}, at least one node for each further q_i, and finally a single *terminal node*, which does not have any successor. The decisions on the value of the qudits is recorded so it can be reconstructed. Reconstructing the amplitude for individual basis states is achieved by accordingly traversing the decision diagram and multiplying all edge weights along the path. An example illustrates the idea.

Example 9. *Consider the quantum state* $\frac{1}{\sqrt{3}}(|00\rangle - |11\rangle + |21\rangle)$ *in a qutrit-qubit system. Figure 3 gives both, the representation as vector and as decision diagram. The vector has 6 entries due to the product of the local dimensionalities:* 3 *for the qutrit and* 2 *for the qubit. The root node* q_1 *has 3 outgoing edges, one for each possible level in qutrit. The nodes in the second level represent the qubit and have 2 edges each. The second and the third edge of the root node point to the same qubit node, due to the exploitation of redundancy. The* q_0 *nodes point to the* terminal node.

Finally, in order to retrieve an amplitude of a basis state, we start by multiplying the weights along the path composed of the basis state we want to retrieve. In the of case $|00\rangle$*, we compose* $1/\sqrt{3} \cdot 1 \cdot 1$*, corresponding to the weights of the root node, the edge* 0 *of* q_1*, and the edge* 0 *of* q_0*. In case of the bitstring* $|11\rangle$*, we multiply* $1/\sqrt{3} \cdot -1 \cdot 1$*, corresponding to the weights of the root node, the edge* 1 *of* q_1 *and the edge* 1 *of* q_0*.*

B. Quantum Operations

The representation of quantum operations as decision diagram follows a similar scheme to that of quantum states. Instead of splitting the vector, for quantum operations the corresponding matrix is split into equal parts depending on the dimenionality of the "most significant qudit", e.g., $3 \times 3 = 9$ parts in case of a qutrit. This scheme is again applied until the individual complex numbers in the matrix are reached. Equal sub-parts up to a complex constant are recognized and stored by the same node in the decision diagram to achieve compactness and canonicity. Again, an example illustrates the idea.

Example 10. *Figure 4 depicts the matrix* U *for a controlled-on-1-NOT between a qutrit and a qubit, consequently the matrix has dimension* 6×6*. This operation, like most controlled single-qudit operations, is sparse and has high redundancy leading to a very compact decision diagram. Further, the all-zero sub-matrices are immediately represented by* zero stubs *from the root node. Therefore, only three out of the nine edges of the root node representing the qutrit point to a non-zero successor. The first and last edge of the root node, point to the same node that represents the pattern for the identity matrix, while the other non-zero edge in between point to a node that represents the qubit Pauli-X.*

Similarly to the decision diagrams for quantum states, the individual elements of the matrix are retrieved by following a path and multiplying the complex edge weights on this path. Considering the element for mapping $|00\rangle$ *to itself (i.e., the element in the upper left corner), we multiply* $1 \cdot 1 \cdot 1$*, corresponding to the weights of the edge to the root node and following the left-most edges from the following nodes.*

C. Rationale for Choosing DDs

Decision diagrams have proven to be an efficient and compact data structure for exponentially-sized information in many cases in classical design automation and they have shown promising result for design automation in the quantum domain thus far. They exhibit multiple advantages, especially for quantum circuit simulation, such as the following:

- Sub-matrices that contain all-zero elements are represented with a zero-stub, i.e., a single edge of weight zero.
- Decision diagrams are canonical and, therefore, guarantee exploiting redundancies in the data structure.
- During operations like multiplication and addition shared nodes might be encountered more than once. This enables caching of results and, therefore, an improved run time.
- The Kronecker product between two decision diagrams requires only the reassignment of pointers from the terminal node of the first DD to the root node of the second (possibly followed by renormalization).
- Decision diagrams nicely model the locality of a single qudit. The number of edges at each levels matches the dimensionality of the qudit.

The aforementioned advantages of decision diagrams make them promising candidates for quantum circuit simulation as considered in this paper. There exist other data structures, most notably tensor networks, that are commonly employed in simulation. However, they serve a different purpose. For tensor networks, the strength is calculating a single observable rather than the full quantum state, where they decay to exponentially sized vectors anyway [57].

VI. Implementation of Mixed-Dimensional Decision Diagrams

This section details important aspects of the implementation of the previously presented decision diagrams. More precisely, we cover construction of decision diagrams and illustrate how the data structure is exploited for performing the basic operations for simulating quantum operations, i.e., multiplication and state measurements. The full source code is available at github.com/cda-tum/MiSiM.

A. Representation of States and Operations

As discussed previously, decision diagrams promise to compactly represent quantum states and quantum operations in many cases. This, so far, rather abstract discussion has to be accompanied by an efficient implementation that can handle the representations without intermediate steps that involve the full vectors or matrices.

Instead of constructing decision diagrams representing quantum states from the corresponding vector (which require exponential memory), their direct construction is limited to basis states [31]. In the presented implementation we consider the basis states. This kind of quantum state can be described linearly in the number of qudits. For the construction of the decision diagram, one node for each qudit is required. Building the state from a bottom-up approach will encode the state of each qudit in a single node, connected via the edges (whose edge weights encode the concrete state). By convention, the initial quantum state in simulation is the all-zero state, which has the nodes in the decision diagram only connected by the left-most edges.

The second key ingredient is the efficient construction of decision diagrams representing quantum operations: the procedure for the construction is shown in Algorithm 1. For the proposed simulator, we considered local operations with arbitrary controls where the matrix for the local operation is the only one that is ever actually kept in memory. In the construction, each qudit is again considered in a bottom-up fashion, i.e., from q_0 to q_{n-1}. Each qudit is either (i) the target qudit, (ii) a control qudit, or (iii) not part of the considered operation. For the target, a node with edges holding the values of the local operation is created. For control qudits, if the edges that represent a mapping where the operation is applied are pointing towards the operations, the remaining edges will point to the identity operation. This is known in the process of the construction. However, unlike qubit systems where controls are limited to $|0\rangle$ and $|1\rangle$, the controls can be on arbitrary levels, therefore constructing controlled operations becomes increasingly complex. For qudits that are not part of the operation, the corresponding edges represent the identity operation on these qudits.

To further optimize the construction process, special patterns in the sub-matrices are identified. These patterns include the identity matrix, symmetric, and transposed matrices. Especially for the identity matrix, sub-graphs are stored in a lookup table and referenced within the decision diagram when needed. This approach avoids the repeated construction of identity operations and speeds up normalization routines. Additionally, caching identity patterns helps to efficiently construct decision diagrams for rotations of sub-spaces within the Hilbert space, where matrices mostly consist of ones with only a few entries representing the actual operation.

Algorithm 1 Make Gate DD

Require: Matrix Operation U, lines L, target line t, controls $c \subset C$
Ensure: root edge e of Gate DD
  e_U = U entries as edges
  for l in L, $l < t$ **do**
    Allocate e_L = $l.size()$ zero edges
    for e_i in e_U **do**
      if l is ctrl, w/ ctrl c and e_i is diag **then**
        $e_{L_c} \leftarrow e_i$
        for e_{L_j} in e_L, $j \neq c$ **do**
          if e_{L_j} is diag **then** $e_{L_j} \leftarrow I$
          else$e_{L_j} \leftarrow$ zero
      else if l is ctrl, w/ ctrl c and e_i not diag **then**
        $e_{L_c} \leftarrow e_i$
      else ▷ Line is not a control
        for e_{L_j} in e_L **do**
          $e_{L_j} \leftarrow e_i$
    $e_i \leftarrow e$ to norm node w/ successors e_L
  $e_U \leftarrow e$ to norm node w/ successors e_i ▷ Target line
  for l in L, $l > t$ **do** ▷ Variables above the target
    Allocate e_L = $l.size()$ zero edges
    for e_j in e_L **do**
      if l is ctrl, w/ ctrl c and e_j is diag **then**
        if $j = c$ **then** $e_j \leftarrow e_U$
        else$e_j \leftarrow I$
      else ▷ Line is not a control
        if e_j is diag **then** $e_j \leftarrow e_U$
    $e_U \leftarrow e$ to norm node w/ successors e_L

The key difference to previous implementations of decision diagrams is the added capability of dynamically creating decision diagrams with different dimensions for each qudit. More precisely from the implementation perspective, the number of out-edges of a node can be changed to accommodate the given dimensionality. This marks a significant departure from the fixed and static structures of previous types of decision diagrams [24], [25], [27]–[30], making it adaptable for future use cases where temporary expansion of the Hilbert space of a qudit could be a crucial component for shorter circuits. The implemented data structure is closer to the behavior of nature than ever before.

B. Applying Quantum Operations

Given the construction of quantum states and operations detailed in the previous section, this section covers the part of the implementation that applies the operations to the states. To this end, we consider the Kronecker product, multiplication of decision diagrams, and measurement as the essential parts of the implementation.

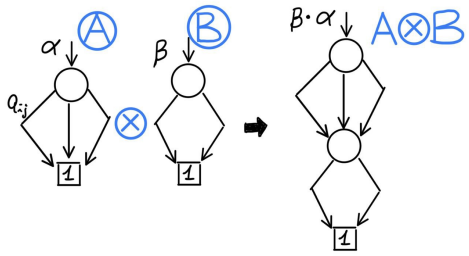


Figure 5: Kronecker product for decision diagrams

Algorithm 2 Kronecker Product

Require: root edge x, root edge y
Ensure: root edge e of $x \otimes y$
  if x or y is zero **then return** zero
  if x is scalar **then return** $y.weight \times x.weight$
  if $LookUp(x, y)$ **then return** $cached(x, y)$
  Allocate e_N = $x.node.size()$ zero edges
  for e_i in e_N **do**
    $e_i \leftarrow$ *Kron*$(x.node.e_i, y)$
  $e_K \leftarrow e$ to norm node w/ successors e_N
  $e_K.weight \leftarrow e_K.weight \times x.weight \times y.weight$
  return e_K

1) Kronecker Product: The first part we consider for applying quantum operations to states is the Kronecker product for decision diagrams. For matrices, it is defined as

$$A \otimes B = \begin{bmatrix} a_{0,0} \cdot B & \cdots & a_{0,D-1} \cdot B \\ \vdots & \ddots & \vdots \\ a_{D-1,0} \cdot B & \cdots & a_{D-1,D-1} \cdot B \end{bmatrix}.$$

This means that each element of A is replaced by the element $a_{i,j} \cdot B$. Moreover, on matrices the Kronecker product is computationally expensive, since each of the elements of the exponentially-sized matrices has to be accessed at least once.

However, for decision diagrams, the Kronecker product is efficient. The terminal node of a decision diagram representing A is replaced by the root node of B by means of changing the corresponding pointers. Afterwards, for normalization, the weight of the previous edge pointing to the root node of B is multiplied to the root edge to A. This process is detailed in Algorithm 2 and illustrated in Figure 5.

The implementation of the Kronecker product uses a recursive approach that leverages the decision diagram; the computational complexity of the operation is linear in the nodes of the left-hand side operand.

2) Multiplication and Addition: As discussed in Section III-A, multiplication and addition are the quintessential operations necessary to conduct quantum circuit simulation. Given a quantum state $|\psi\rangle$ (with vector representation ψ) and a unitary operation U, the operation is applied by:

$$\psi'_i = \sum_{k=0}^{D-1} u_{i,k} \cdot \psi_k,$$

where D is the dimension of the vector and $u_{i,k}$ (ψ_k) represents an element in the matrix (vector). We can use the recursive conceptual decomposition of the matrix (vector) represented by the respective DD, but this time to perform the multiplication. We use

$$\psi' = U \cdot \psi = \begin{bmatrix} U_{0,0} & \cdots & U_{0,D-1} \\ \vdots & \ddots & \vdots \\ U_{D-1,0} & \cdots & U_{D-1,D-1} \end{bmatrix} \cdot \begin{bmatrix} \psi_0 \\ \vdots \\ \psi_{D-1} \end{bmatrix}$$
$$= \begin{bmatrix} \sigma_0 \\ \vdots \\ \lambda_0 \end{bmatrix} + \cdots + \begin{bmatrix} \sigma_{D-1} \\ \vdots \\ \lambda_{D-1} \end{bmatrix}$$

The algorithm will recursively follow corresponding paths in the DDs for the matrix and the vector, and apply the operation of multiplication between the sub-matrices and sub-vectors, as detailed in Algorithm 3 and Algorithm 4. All the intermediate state vectors can be added recursively, i.e.,

$$\psi' = \begin{bmatrix} \sigma_0 + \cdots + \sigma_{D-1} \\ \vdots \\ \lambda_0 + \cdots + \lambda_{D-1} \end{bmatrix}$$

The order and structure of the sub-multiplications and sub-additions follow the structure of the decision diagram. Consequently, the complexity is bound to the decision diagram in terms of number of nodes. However, the reduction of the nodes allows to cache intermediate results. Finally, the reduction of DDs potentially delivers an exponential improvement in terms of memory complexity, as well as in terms of time complexity by terminating early sub-computations already performed. The final step of multiplication is, naturally, the normalization of the decision diagram. The procedure is recursive and included in the steps described in Algorithm 4.

The multiplication and addition operations have been optimized to account for the dynamic structure of the decision diagrams. Unlike previous versions of the data structure where the number of edges was fixed and known beforehand (i.e., pointers in an array), the current version uses a vector with variable-sized edges (pointers), which allows for greater flexibility. Despite the added overhead of checking the size and iterating over each edge, the implementation remains relatively efficient compared to previous fixed versions of the data structure.

3) Measurement: The measurement can be interpreted as a global measurement since the outcome of the simulation is the state vector of the system. Decision diagrams enable a measurement procedure that is linear in the number of qudits.

The process of measuring (or sampling) from the decision diagram requires one traversal of the decision diagram in a top-to-bottom manner, starting from the root node. At each node, the squared magnitude of each edge weight are the probability that this edge should be followed. After following the edge to the next node, this procedure is repeated until the target node is reached. The corresponding basis state is then given by the path through the decision diagram (the index of each edge gives the corresponding level, starting with zero) and the product of the edge weights on the path gives the corresponding amplitude.

Algorithm 3 Multiplication

Require: root edge x, root edge y
Ensure: root edge e of $x \times y$
if x or y is NULL or $x.weight$ or $y.weight = 0$ **then return** zero
if $Terminal(x)$ and $Terminal(y)$ **then return** Terminal e, w/ $e.weight = x.weight \times y.weight$
if $LookUp(x, y)$ **then return** $cached(x, y)$
if x is I **then return** y
if y is I **then return** x
Allocate $e_N[R]$ zero edges ▷ Edges for Results
for i in R **do** ▷ Rows
  for j in C **do** ▷ Columns
    for e_k in e_N, $k = R * i + j$; **do**
      $e_k \leftarrow e_k + Mult(e_x[R \cdot i + k], e_y[j + C \cdot k])$
$Cache\ e_K \leftarrow e$ to norm node w/ successors e_N
$e_K.weight \leftarrow e_K.weight \times x.weight \times y.weight$
return e_K

Algorithm 4 Addition

Require: root edge x, root edge y
Ensure: root edge e of $x + y$
if $Terminal(x)$ **then return** y
if $Terminal(y)$ **then return** x
if $x.node = y.node$ **then return** e, w/ $e.weight = x.weight + y.weight$ and $e.node = x.node$
if $LookUp(x, y)$ **then return** $cached(x, y)$
Create e_N edges with $e_N.size() = x.nodes.edges.size$
for $i = 0 \dots e_N.size() - 1$ **do**
  $e_{y,next} \leftarrow e_{yi.node}; e_{y,next.weight} \leftarrow y_{weight} \times e_{y_i.weight}$
  $e_{x,next} \leftarrow e_{xi.node}; e_{x,next.weight} \leftarrow x_{weight} \times e_{x_i.weight}$
  $e_j = add(e_{x,next}, e_{y,next}), i = j$
return e to norm node w/ successors e_N

VII. Experimental Evaluation

We implemented the proposed approach and evaluated its applicability. The implementation is publicly available at github.com/cda-tum/MiSiM as part of the Munich Quantum Toolkit (MQT, [58]).

The evaluation was performed on a server running GNU/Linux with an Intel Xeon W-1370P (running at 3.6 GHz) and 128 GiB main memory. The implementation is written in C++ and was compiled with GCC 9.3.0. To demonstrate the capabilities, a set of three different algorithms with multiple instances was considered more precisely:

- Mixed W-States [59] given as states $\frac{1}{\sqrt{n}}(|0\dots01\rangle + |0\dots010\rangle + |10\dots0\rangle)$. Qu*b*it W-states embedded in prime-dimensional qu*d*its.
- *GHZ States* [60] were considered for n qutrits as $\frac{1}{\sqrt{3}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n} + |2\rangle^{\otimes n})$.
- Random Circuits built from randomly selected local operations (Hadamard and Givens rotations) and entangling operations (*CEX* and controlled Clifford operations).

The results are summarized in Table I. The first column lists the name of the benchmark. The second column gives the total number of qudits where as the following four columns give the number of qudits for each level, i.e., the number of qubits (2), qutrits (3), ququads (4), and ququints (5). In the following three columns, the number of operations, number of nodes, and number of distinct complex numbers in the decision diagram of the final state after the simulation are listed. Finally, the last column lists the run time of each instance of the benchmark.

For the algorithms Mixed W-State and GHZ state, Table I nicely confirms the efficiency of the proposed approach and the corresponding implementation. The simulation for these algorithms completes in less than a second with a low number of nodes and distinct complex numbers, as one might expect. In contrast, the random circuits contain little to no redundancy to be exploited by the decision diagrams and, therefore, can be considered to show the worst-case behavior. Still, the implementation can simulate these circuits with, for example, 8000 operations on 8 qudits within around 4 h, which, given the individual number of levels, are comparable to 14 qubits. The simulator is remarkable in its ability to deliver reliable results for randomized circuits, also when those are comparable to a larger 15-qubit system. Although the simulation process largely terminates within two days, this time frame is a testament to the efficiency and accuracy of the simulator's algorithms. These results confirm the efficacy of the simulator as a design tool for mixed-dimensional quantum circuits.

VIII. Conclusions

Design automation is an essential component in the development of quantum computers and a key factor in unlocking their full potential. In this paper, we proposed a classical simulator for mixed-dimensional qudit systems. To this end, we introduced a type of decision diagram capable of handling the simulation of mixed-dimensional quantum circuits and presented a corresponding implementation (publicly available at github.com/cda-tum/MiSiM). More precisely, the proposed decision diagrams encode the dimensionality of a qudit in the number of out-going edges of a node, enabling a dynamic handling of the dimension and fast adaption to the restrictions imposed, e.g., after calibration of a qudit system. This compact representation is the basis for the corresponding implementation, which enables utilization of the method in real world context. The experimental evaluation confirms the efficacy for the selected set of benchmarks, ranging from easy circuits such as the GHZ state to random circuits, which show the worst-case behavior.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the ERC Consolidator Grant (agreement No 101001318) and the NeQST Grant (agreement No 101080086). It is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus and was partially supported by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

Table I: Evaluations

Benchmark	#Qudits	#Qudits with Dimension				#Operations	#Nodes	#Distinct $\mathbb{C}$	Run time
		2	3	4	5				
Mixed W-State	4	2	2	0	0	15	12	9	0.000
	30	15	3	0	12	96	60	64	0.017
	54	2	52	0	0	159	108	112	0.027
	60	8	4	0	48	213	120	140	0.169
	90	15	63	0	12	276	180	198	0.103
	90	2	80	0	8	273	180	195	0.101
	102	75	2	0	25	315	200	207	0.111
	108	8	100	0	0	321	216	229	0.102
GHZ State	5	0	5	0	0	8	14	5	0.000
	10	0	10	0	0	18	29	5	0.000
	30	0	30	0	0	58	89	5	0.003
	60	0	60	0	0	118	179	5	0.011
	120	0	120	0	0	238	359	5	0.043
	128	0	128	0	0	254	383	5	0.049
Random	2	0	0	2	0	2000	6	68 903	0.021
	2	0	2	0	0	2000	5	40 835	0.015
	2	0	0	0	2	2000	7	112 918	0.068
	2	0	1	1	0	2000	5	52 451	0.016
	2	0	1	1	0	2000	5	52 451	0.016
	3	2	1	0	0	3000	10	66 575	0.021
	3	0	0	2	1	3000	27	531 995	0.172
	3	0	0	2	1	3000	27	545 173	0.184
	3	1	0	2	0	3000	12	168 915	0.048
	3	1	0	2	0	3000	14	227 513	0.069
	4	1	2	0	1	4000	67	774 865	0.339
	4	1	1	1	1	4000	44	870 383	0.380
	4	2	2	0	0	4000	22	239 247	0.069
	4	0	1	1	2	4000	107	2 651 118	4.172
	4	2	1	0	1	4000	41	429 614	0.153
	4	0	0	2	2	4000	106	3 141 654	6.564
	4	0	2	1	1	4000	53	1 299 120	0.828
	5	0	0	3	2	5000	507	16 771 562	231.361
	5	1	1	2	1	5000	214	4 578 367	13.622
	5	1	1	2	1	5000	161	3 287 282	6.592
	5	2	0	1	2	5000	137	3 181 180	6.258
	5	2	1	0	2	5000	101	2 319 028	3.094
	5	2	0	1	2	5000	164	3 435 001	7.242
	5	1	1	1	2	5000	217	5 732 896	22.778
	6	2	1	1	2	6000	434	11 781 434	105.238
	6	2	2	1	1	6000	486	7 518 639	38.225
	6	2	1	0	3	6000	437	13 889 217	159.516
	6	4	2	0	0	6000	137	1 324 892	0.959
	7	3	1	2	1	7000	1308	23 540 630	390.539
	7	3	1	0	3	7000	2447	42 457 590	1275.100
	7	3	2	0	2	7000	577	16 793 344	227.600
	7	2	2	1	2	7000	1301	41 679 061	1407.100
	8	5	2	1	0	8000	572	11 114 401	85.309
	8	2	3	1	2	8000	3902	135 137 166	14 972.200
	8	2	1	2	3	8000	19886	432 117 869	170 405.000

The column "#Distinct $\mathbb{C}$" shows the number of different complex numbers in the decision diagram.

References

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

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Symp. on Theory of Computing*, G. L. Miller, Ed., ACM, 1996, pp. 212–219.

[3] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[4] M. Ringbauer, M. Meth, L. Postler, R. Stricker, R. Blatt, P. Schindler, and T. Monz, "A universal qudit quantum processor with trapped ions," *Nature Physics*, vol. 18, no. 9, pp. 1053–1057, 2022.

[5] Y. Wang, Z. Hu, B. C. Sanders, and S. Kais, "Qudits and high-dimensional quantum computing," *Frontiers in Physics*, vol. 8, p. 589 504, 2020.

[6] X. Zhang, M. Um, J. Zhang, S. An, Y. Wang, D.-l. Deng, C. Shen, L.-M. Duan, and K. Kim, "State-independent experimental test of quantum contextuality with a single trapped ion," *Phys. Rev. Lett.*, vol. 110, p. 070 401, 2013.

[7] B. P. Lanyon, M. Barbieri, M. P. Almeida, T. Jennewein, T. C. Ralph, K. J. Resch, G. J. Pryde, J. L. O'Brien, A. Gilchrist, *et al.*, "Simplifying quantum logic using higher-dimensional Hilbert spaces," *Nature Physics*, vol. 5, pp. 134–140, 2008.

[8] M. Ringbauer, T. R. Bromley, M. Cianciaruso, L. Lami, W. Y. S. Lau, G. Adesso, A. G. White, A. Fedrizzi, and M. Piani, "Certification and quantification of multilevel quantum coherence," *Phys. Rev. X*, vol. 8, p. 041 007, 2018.

[9] X.-M. Hu, Y. Guo, B.-H. Liu, Y.-F. Huang, C.-F. Li, and G.-C. Guo, "Beating the channel capacity limit for superdense coding with entangled ququarts," *Sci. Adv.*, vol. 4, no. 7, 2018.

[10] M. Malik, M. Erhard, M. Huber, M. Krenn, R. Fickler, and A. Zeilinger, "Multi-photon entanglement in high dimensions," *Nature Photonics*, vol. 10, pp. 248–252, 2016.

[11] M. Kononenko, M. Yurtalan, S. Ren, J. Shi, S. Ashhab, and A. Lupascu, "Characterization of control in a superconducting qutrit using randomized benchmarking," *Phys. Rev. Res.*, vol. 3, no. 4, p. L042007, 2021.

[12] A. Morvan, V. V. Ramasesh, M. S. Blok, J. M. Kreikebaum, K. O'Brien, L. Chen, B. K. Mitchell, R. K. Naik, D. I. Santiago, *et al.*, "Qutrit randomized benchmarking," *Phys. Rev. Lett.*, vol. 126, p. 210 504, 2021.

[13] J. Ahn, T. Weinacht, and P. Bucksbaum, "Information storage and retrieval through quantum phase," *Science*, vol. 287, no. 5452, pp. 463–465, 2000.

[14] C. Godfrin, A. Ferhat, R. Ballou, S. Klyatskaya, M. Ruben, W. Wernsdorfer, and F. Balestro, "Operating quantum states in single magnetic molecules: Implementation of Grover's quantum algorithm," *Phys. Rev. Lett.*, vol. 119, p. 187 702, 2017.

[15] B. E. Anderson, H. Sosa-Martinez, C. A. Riofrío, I. H. Deutsch, and P. S. Jessen, "Accurate and robust unitary transformations of a high-dimensional quantum system," *Phys. Rev. Lett.*, vol. 114, p. 240 401, 2015.

[16] Z. Gedik, I. A. Silva, B. Çakmak, G. Karpat, E. L. G. Vidoto, D. O. Soares-Pinto, E. R. DeAzevedo, and F. F. Fanchini, "Computational speed-up with a single qudit," *Sci. Rep.*, vol. 5, p. 14 671, 2015.

[17] F. Tacchino, A. Chiesa, R. Sessoli, I. Tavernelli, and S. Carretta, "A proposal for using molecular spin qudits as quantum simulators of light–matter interactions," *J. Mater. Chem. C*, vol. 9, pp. 10 266–10 275, 32 2021.

[18] Y. Deller, S. Schmitt, M. Lewenstein, S. Lenk, M. Federer, F. Jendrzejewski, P. Hauke, and V. Kasper, *Quantum approximate optimization algorithm for qudit systems with long-range interactions*, 2022. arXiv: 2204.00340.

[19] D. Gonzá lez-Cuadra, T. V. Zache, J. Carrasco, B. Kraus, and P. Zoller, "Hardware efficient quantum simulation of non-abelian gauge theories with qudits on Rydberg platforms," *Physical Review Letters*, vol. 129, no. 16, 2022.

[20] E. J. Gustafson, "Prospects for simulating a qudit-based model of $(1+1)$D scalar qed," *Phys. Rev. D*, vol. 103, p. 114 505, 11 2021.

[21] D. M. Kurkcuoglu, M. S. Alam, A. C. Li, A. Macridin, and G. N. Perdue, "Quantum simulation of ϕ^4 theories in qudit systems," *TBD*, 2021.

[22] L. Lamata, A. Mezzacapo, J. Casanova, and E. Solano, "Efficient quantum simulation of fermionic and bosonic models in trapped ions," *EPJ Quantum Technology*, vol. 1, p. 9, 2014.

[23] B. Lanyon, M. Barbieri, M. Almeida, T. Jennewein, T. Ralph, K. Resch, G. Pryde, J. O'Brien, A. Gilchrist, *et al.*, "Quantum computing using shortcuts through higher dimensions," *Nature Physics*, vol. 5, 2008.

[24] D. Miller and M. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, IEEE, 2006.

[25] 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*, ACM, 2019, pp. 1–7.

[26] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 848–859, 2019.

[27] 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.

[28] S. Wang, C. Lu, I. Tsai, and S. Kuo, "An XQDD-based verification method for quantum circuits," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 91-A, no. 2, pp. 584–594, 2008.

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

[30] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 35, no. 1, pp. 86–99, 2016.

[31] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[32] G. K. Brennen, S. S. Bullock, and D. P. O'Leary, "Efficient circuits for exact-universal computationwith qudits," *Quantum Info. Comput.*, vol. 6, no. 4, pp. 436–454, 2006.

[33] P. Hrmo, B. Wilhelm, L. Gerster, M. W. van Mourik, M. Huber, R. Blatt, P. Schindler, T. Monz, and M. Ringbauer, "Native qudit entanglement in a trapped ion quantum processor," *Nature Communications*, vol. 14, no. 1, p. 2242, 2023.

[34] S. Hillmich, I. L. Markov, and R. Wille, "Just like the real thing: Fast weak simulation of quantum computation," in *Design Automation Conf.*, IEEE, 2020, pp. 1–6.

[35] S. Jaques and T. Häner, "Leveraging state sparsity for more efficient quantum simulations," *ACM Trans. on Quantum Computing*, vol. 3, no. 3, 2022.

[36] T. J. Stavenger, E. Crane, K. C. Smith, C. T. Kang, S. M. Girvin, and N. Wiebe, "C2QA - bosonic qiskit," in *High Performance Extreme Computing Conf.*, 2022.

[37] K. Mato, M. Ringbauer, S. Hillmich, and R. Wille, "Compilation of entangling gates for high-dimensional quantum systems," in *Asia and South Pacific Design Automation Conf.*, ACM, 2023, pp. 202–208.

[38] K. Mato, M. Ringbauer, S. Hillmich, and R. Wille, "Adaptive compilation of multi-level quantum operations," in *Int'l Conf. on Quantum Computing and Engineering*, IEEE, 2022, pp. 484–491.

[39] B. Poór, Q. Wang, R. A. Shaikh, L. Yeh, R. Yeung, and B. Coecke, "Completeness for arbitrary finite dimensions of ZXW-calculus, a unifying calculus," 2023.

[40] K. Mato, S. Hillmich, and R. Wille, "Compression of Qubit Circuits: Mapping to Mixed-Dimensional Quantum Systems," in *Int'l Conf. on Quantum Software*, 2023.

[41] A. Litteken, J. M. Baker, and F. T. Chong, "Communication trade offs in intermediate qudit circuits," in *Int'l Symp. on Multi-Valued Logic*, 2022, pp. 43–49.

[42] L. Seifert, J. Chadwick, A. Litteken, F. T. Chong, and J. M. Baker, "Time-efficient qudit gates through incremental pulse re-seeding," in *Int'l Conf. on Quantum Computing and Engineering*, 2022, pp. 304–313.

[43] L. Burgholzer and R. Wille, "The power of simulation for equivalence checking in quantum computing," in *Design Automation Conf.*, IEEE, 2020, pp. 1–6.

[44] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1810–1824, 2021.

[45] D. Volya and P. Mishra, "Quantum data compression for efficient generation of control pulses," in *Asia and South Pacific Design Automation Conf.*, ACM, 2023, pp. 216–221.

[46] H. Abraham, AduOffei, R. Agarwal, G. Agliardi, M. Aharoni, V. Ajith, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, *et al.*, *Qiskit: An open-source framework for quantum computing*, 2021.

[47] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "QuEST and high performance simulation of quantum computers," *Sci. Rep.*, vol. 9, no. 1, 2019.

[48] S. Efthymiou, S. Ramos-Calderer, C. Bravo-Prieto, A. Pérez-Salinas, D. García-Martín, A. Garcia-Saez, J. I. Latorre, and S. Carrazza, "Qibo: A framework for quantum simulation with hardware acceleration," *Quantum Science and Technology*, vol. 7, no. 1, p. 015 018, 2021.

[49] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, "Yao.jl: Extensible, efficient framework for quantum algorithm design," *Quantum*, vol. 4, p. 341, 2020.

[50] S. V. Isakov, D. Kafri, O. Martin, C. V. Heidweiller, W. Mruczkiewicz, M. P. Harrigan, N. C. Rubin, R. Thomson, M. Broughton, *et al.*, *Simulations of quantum circuits with approximate noise using qsim and Cirq*, 2021. arXiv: 2111.02396.

[51] J. Hauschild and F. Pollmann, "Efficient numerical simulations with tensor networks: Tensor network python (TeNPy)," *SciPost Phys. Lecture Notes*, 2018.

[52] J. Gray, "Quimb: A python package for quantum information and many-body calculations," *Journal of Open Source Software*, vol. 3, no. 29, p. 819, 2018.

[53] M. Fishman, S. R. White, and E. M. Stoudenmire, "The ITensor Software Library for Tensor Network Calculations," *SciPost Phys. Codebases*, p. 4, 2022.

[54] T. Chatterjee, A. Das, S. K. Bala, A. Saha, A. Chattopadhyay, and A. Chakrabarti, *QuDiet: A classical simulation platform for qubit-qudit hybrid quantum systems*, 2022. arXiv: 2211.07918.

[55] S. Hillmich, C. Hadfield, R. Raymond, A. Mezzacapo, and R. Wille, "Decision diagrams for quantum measurements with shallow circuits," in *Int'l Conf. on Quantum Computing and Engineering*, IEEE, 2021, pp. 24–34.

[56] R. Wille, S. Hillmich, and L. Burgholzer, "Decision diagrams for quantum computing," in *Design Automation of Quantum Computers*. Springer, 2023, pp. 1–23.

[57] L. Burgholzer, A. Ploier, and R. Wille, *Tensor networks or decision diagrams? Guidelines for classical quantum circuit simulation*, 2023. arXiv: 2302.06616.

[58] R. Wille, S. Hillmich, and L. Burgholzer, "MQT: The Munich Quantum Toolkit," in *Gesellschaft für Informatik Quantum Computing Workshop*, 2022.

[59] L. Yeh, *Scaling W state circuits in the qudit Clifford hierarchy*, 2023. arXiv: 2304.12504.

[60] D. M. Greenberger, M. A. Horne, and A. Zeilinger, "Going beyond bell's theorem," in *Bell's Theorem, Quantum Theory and Conceptions of the Universe*, M. Kafatos, Ed. Springer, 1989, pp. 69–72.