BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026
http://excel.fit.vutbr.cz

A Tool for Quantum Circuits Analysis

Filip Novák*

Abstract

Quantum circuit analysis is essential while large-scale quantum hardware remains unavailable and further performance improvements are required. We present an MTBDD-based package with alternative amplitude representations that accelerates simulation. We also provide a unified OpenQASM frontend for multiple backends.

*xnovakf00@stud.fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Quantum circuit analysis and simulation face a fundamental obstacle: the state vector grows exponentially with every additional quantum bit—*qubit*. Compact representations and efficient data structure manipulation are therefore essential for simulating larger circuits on classical hardware.

One such tool is `MEDUSA` [1], a decision diagram-based simulator using the `Sylvan` [2] library as its backend. While effective, Sylvan's multi-core parallelism introduces overhead that hurts performance on single-simulation workloads, and the original amplitude representation limits the set of quantum gates.

We address both issues by extending the `BuDDy` [3] library to support multi-terminal decision diagrams with configurable amplitude representations, and integrating it as a new `MEDUSA` backend. Additionally, we develop `qFront`, a unified `OpenQASM 3` frontend that emits a backend-independent intermediate representation suitable for `MEDUSA` and other simulators.

2. Quantum Simulation

Simulating a quantum circuit means applying a sequence of gates to an initial state vector, where each gate is represented as a unitary matrix acting on one or more qubits. The state of an n-qubit system is described by 2^n complex amplitudes, whose squared magnitudes must sum to 1 at all times. Repeated matrix-vector multiplication over this exponentially large space is what makes classical simulation fundamentally expensive. Many simulators therefore, opt for more concise representations. One such representation is a multi-terminal decision diagram (MTBDD), described further in Section 3.

Quantum circuits exploit phenomena such as superposition and entanglement to encode and process information in ways that have no classical counterpart. Certain algorithms, such as Grover's search, achieve provably better complexity than their best-known classical equivalents. Classical simulation of these circuits is therefore not just an engineering challenge, but also a tool for verifying correctness, and analyzing behavior.

3. MTBDDs

Multi-Terminal Binary Decision Diagrams (MTBDDs) are an extension of Binary Decision Diagrams (BDDs) in which terminal nodes can take values from an arbitrary countable domain $\mathbb{D}$, rather than just $\{0, 1\}$. Formally, an MTBDD represents a function

$$f\colon \{0,1\}^k \to \mathbb{D}.$$

In the context of quantum simulation, the k binary variables encode the index of a basis state, and the terminal values are the corresponding complex amplitudes. Shared sub-graphs allow states with repeated amplitude values to be stored compactly, and standard recursive algorithms over the diagram structure replace explicit matrix-vector multiplication. Diagrams can be found in **Figure 3**.

4. MoToBuDDy

`BuDDy` is a lightweight C library for Binary Decision Diagram manipulation, originally developed by J. Lind-Nielsen [3]. Compared to `Sylvan` [2], it carries signifi-

cantly less overhead, as it operates sequentially and requires no synchronisation primitives. Its main limitation is the lack of support for multi-terminal diagrams.

Our extension, `MoToBuDDy`, adds MTBDD support while keeping the internal node structure unchanged. Instead of storing two successor indices for high and low edges, leaf nodes repurpose these fields as an index into a dedicated terminal value table. Terminal values are stored in a hash table, ensuring $O(1)$ average-case lookup and deduplication, so identical amplitude values are stored only once, preserving the compactness that makes decision diagrams attractive for quantum simulation.

4.1 MoToBuDDy as MEDUSA Backend

We successfully integrated `MoToBuDDy` as a backend for `MEDUSA`. The performance improvement over the original Sylvan-based backend is visible in **Figure 7** as *Medusa-MoToBuDDy [loop]*.

Originally, complex amplitudes were represented in algebraic form using `MPZ` multi-precision integers from the GMP library [4]. While exact, this representation introduces non-trivial overhead and restricts the supported gate set — gates with arbitrary rotation angles, such as R_X and R_Y, cannot be expressed as algebraic integers.

We therefore supplemented the backend with a floating-point representation, storing amplitudes as real and imaginary parts. This significantly accelerated simulation, as shown in **Figure 7**. To quantify the precision loss, we measured the deviation of $\sum_i |\alpha_i|^2$ from 1 after each gate application or loop summarization and report the maximum observed deviation in **Table 1**. Based on these results, we adopt *quadruple (128-bit)* floating-point precision for all further experiments, as it provides sufficient numerical accuracy across the tested benchmarks.

5. OpenQASM Frontend

Tools for quantum circuit simulation and analysis typically accept circuits described in the `OpenQASM` 3 language. We developed `qFront`, a frontend tool that parses such circuits and transforms them into formats suitable for downstream simulators. The internal representation can be modified to suit the target backend — supported transformations include loop unrolling, gate and subroutine inlining, and register grouping.

One output format is a custom tree manipulation language that encodes gate application as a sequence of decision diagram passes. Rather than repeatedly evaluating a general complex update formula, the entire gate can be applied in a single pass over the diagram. An example of the CX gate representation in this format is shown in **Figure 4**.

6. Benchmarks

We evaluated multiple `MEDUSA` implementations against state-of-the-art simulators: `Quokka#`, `Quasimodo`, `SliQSim`, `DDSIM`, and `Q-Sylvan`. For each competing tool, only its best-performing configuration is reported. All `MEDUSA` variants outperform the competition on amplitude amplification algorithm circuits due to loop summarisation. The `MoToBuDDy` backend with floating-point amplitudes achieves further speedups, as discussed in Section 4.1.

Notably, the double-precision (f64) variant achieves a speedup of $\mathbf{6916}\times$ over the original `MEDUSA` on the Grover benchmark (0.454 s vs. 3140 s).

Figure 8 shows a per-circuit scatter plot comparing the `Sylvan`-based and f128 `MoToBuDDy` backends. The new implementation is faster on the majority of circuits, however, for circuits from the *Random* family it underperforms.

7. Conclusions

We presented `MoToBuDDy`, a lightweight MTBDD package and an alternative backend for the `MEDUSA` quantum circuit simulator. Replacing `Sylvan` with `MoToBuDDy` alone yields a notable speedup due to the elimination of parallelism overhead. Supplementing the exact `MPZ` algebraic amplitude representation with floating-point arithmetic brings further significant speedups, up to $6916\times$ over the original MEDUSA, at the cost of some numerical precision. Quadruple (128-bit) precision retains sufficient accuracy across all tested benchmarks, making it the recommended configuration.

Additionally, we contributed `qFront`, a unified `OpenQASM` 3 frontend that performs semantic validation and transforms circuits into backend-specific representations, lowering the integration barrier for future simulators and analysis tools.

Acknowledgements

I would like to thank my supervisor doc. Ing. Ondřej Lengál Ph.D for their help and valued insights.

References

[1] Tian-Fu Chen, Yu-Fang Chen, Jie-Hong Roland Jiang, Sára Jobranová, and Ondřej Lengál. *Accelerating Quantum Circuit Simulation with Symbolic Execution and Loop Summarization*. Association for Computing Machinery, New York, NY, USA, 2025.

[2] Tom van Dijk and Jaco van de Pol. Sylvan: Multi-core decision diagrams. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 677–691. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[3] Jørn Lind-Nielsen. BuDDy: A Binary Decision Diagram Package. Technical report, Department of Information Technology, Technical University of Denmark, 1999.

[4] Torbjörn Granlund and GMP development team. GNU MP: The GNU multiple precision arithmetic library. online, 2020.