Maria Zmpainou
mzmpainou@student.tudelft.nl

Responsible professor:
Tim Coopmans

Daily supervisor:
Juul Sanders

Delft University of Technology

BEYOND GENERAL-PURPOSE HASHING: CIRCUIT-AWARE UNIQUE TABLES FOR QMDD SIMULATION

Reducing node-lookup comparisons in QFT Entangled QMDD simulation

Motivation & Problem

- Current quantum hardware is limited and only returns measurement outcomes.
- Classical simulation gives access to the full state vector → circuit debugging, noise studies, controlled tests before hardware execution.
- Direct simulation is expensive: every added qubit doubles the state size.
- Quantum Multiple-Valued Decision Diagrams (QMDDs) compress the state by representing it as a graph
- For the merging, nodes are looked up in a table storing all unique nodes.

Problem:
Unique-table lookup cost is usually treated as a general hash-table problem.

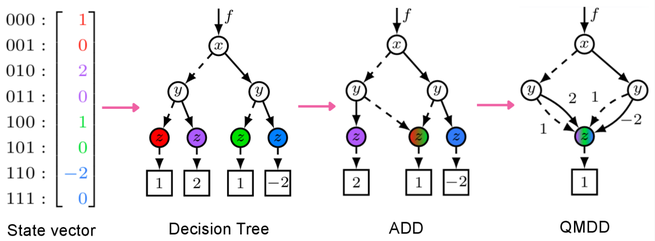
Idea:
For structured circuits, lookup cost also depends on the circuit: gates change the quantum state → this changes the QMDD → this determines which node requests appear.

Goal:
Use the known QFT-GHZ gate sequence to predict unique-table calls and reduce unnecessary lookup comparisons.

Background: Compression of state vector

An n-qubit state is a vector of 2^n complex values.

A state gets compressed by representing it as a graph:
1. Recursively split by qubit value → Decision Tree
2. Merge equivalent subgraphs → Algebraic Decision Diagram (ADD)
3. Merge subgraphs that differ only by a scaling factor → QMDD

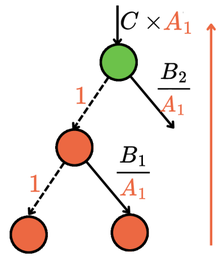


Depending on where the scaling factor is stored, the same substate can be described in different ways.

Background: Normalization and Unique-Tables

Normalization fixes this:
- scalar factor moved to incoming edge
- outgoing edges get fixed form
- scalar-equivalent subgraphs get the same description

→ Equivalent nodes recognized and merged.



Unique tables store one representative of each normalized node. If a node is found it gets reused, otherwise a fresh one gets created and inserted.

The unique table is implemented as a **hash table**. Using **separate chaining**, a node is mapped to one bucket; each non-matching entry is a failed comparison.

Baseline: Global hash table

Hash combines all fields defining a node: children, zero-low flag, complex high-edge label

This gives a reasonable bucket distribution

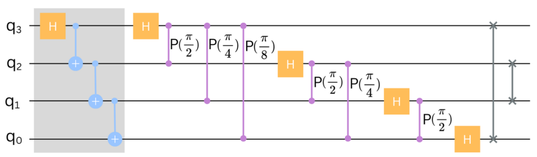
Qubits	Entries	Used buckets	Buckets ≥ 4	Buckets ≥ 8	Hits	Failed checks
6	3,070	2,762	11	0	2,643	878
7	6,438	5,161	34	2	5,808	2,592
8	13,670	9,136	205	2	12,727	9,659
9	27,768	13,145	1,658	13	26,185	37,193
10	57,960	15,780	7,531	658	55,871	131,170

Table 5: Global hash-table baseline on QFT-GHZ circuits with 16,384 buckets.

Many failed checks → Hashing not enough → circuit-aware design

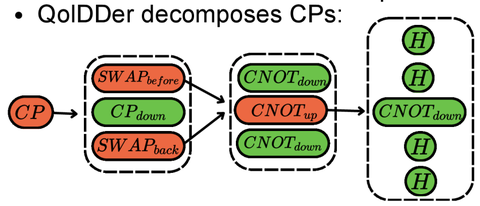
QFT-GHZ Circuit

The benchmark is a Greenberger–Horne–Zeilinger (GHZ)-entangled input state followed by the Quantum Fourier Transform (QFT).



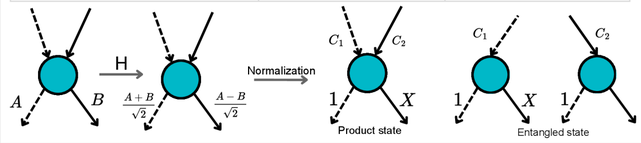
QoIDDer Gate Decomposition

- QFT: Hadamards + CPs + Swaps
- QoIDDer decomposes CPs:



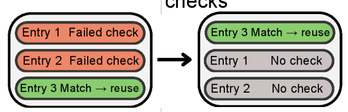
1. Temporary nodes → Node Removal

QMDD Effect	Lookup pattern	Table design
H mixes low/high labels of each node at that qubit level In product state, one shared node In entagled state split into 2 nodes Lower-level Hs split these nodes again: 2 → 4 → 8 ... Swap Hs create many fresh nodes. They encode the phase labels present at that moment	Later CPs add new phases → previous intermediate nodes remain on the table Swap-back → no request to these nodes Swap-before →requested by the next same-control CP Only add failed comparisons	Skip unique-table insertion for swap-back nodes Temporarily insert swap-before nodes and remove them after their predicted reuse



2. Frequently reused nodes → Bucket Promotion

QMDD Effect	Lookup pattern	Table design
Phases top-down. Lower GHZ nodes keep 1/0 structure until phases reach them → New parents are rebuilt above same lower nodes Once a subgraph contains phases and H-mixing, rare exact same subgraphs Level-1 nodes with both children terminal → After last phase same node reappears	Repeatedly request lower GHZ nodes longest-distance CP creates a stable level-1 node repeatedly requested Last-CP swap-before Hs create similar level-1 frequently requested nodes	Mark frequently reused nodes Move marked entries to the bucket front Keep new entries behind marked hot entries Repeated requests avoid repeated failed checks

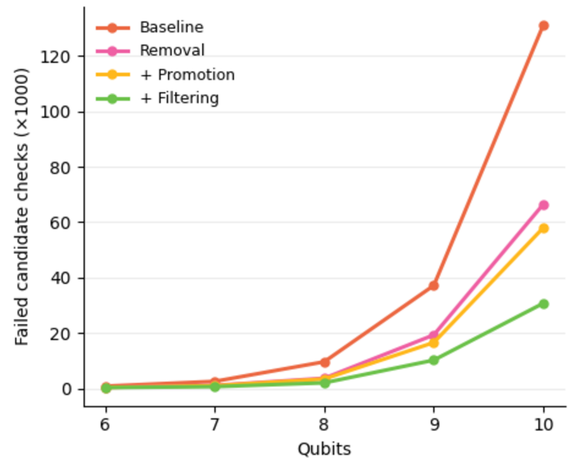


3. Impossible matches → Lookup filtering

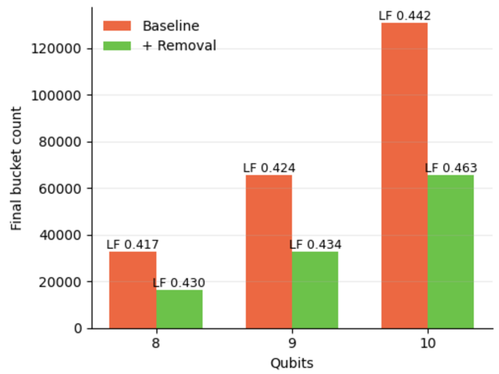
QMDD Effect	Lookup pattern	Table design
Swap Hs create many intermediate nodes. Later CPs add new phases creating nodes that didn't exist before	Swaps make requests predicted to lead to misses A requested parent node uses a child created in the current swap Older table entries cannot contain it, so their comparisons must fail	Mark swap nodes. Check requested low/high child IDs. If one child created in current swap, skip bucket entries before swap. Compare only with entries created in current swap

Results & Discussion

- Failed candidate checks decrease with each circuit-aware design
- Full design: 76.54% fewer failed checks



- From 8 qubits onward, removal avoids one resize and reaches a similar load factor with half buckets



- Failed checks follow circuit patterns, not only hash distribution:
- removal → temporary nodes
- promotion → repeated requests
- filtering → impossible matches

Conclusion & Future Work

- Studying the circuit makes lookup calls predictable.
- Predicted calls can guide unique-table decisions.
- Circuit-aware design avoids comparisons a general hash table still performs.
- Future work: test whether the same idea works for other circuits and improves full simulator runtime.