# Implementing QVMP using QROM

Elton Pinto

April 12, 2022

# Motivation

- Grover search: popular quantum search algorithm
- Depends on a black-box oracle to perform the search
- Offers quadratic speedup over classical linear search with a runtime of $O(\sqrt{N})$

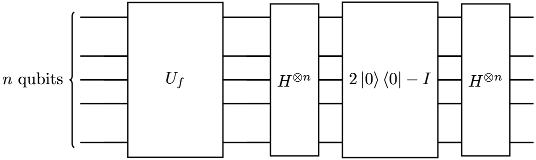



Figure 1: Grover operator circuit

# Motivation (contd)

- Core of Grover search straightforward to implement
- Main challenge: encoding the oracle as a quantum circuit
- QVMP
  - Quantum Verification of Matrix Products
  - Offers quadratic speed-up over classical VMP
  - Used in HPC applications
  - Algorithm uses Grover search as a sub-routine

**Goal**

Implement QVMP to better understand these challenges, determine feasibility of use, and investigate enhancements into oracle encoding

# Main contributions

- Working implementation of QVMP in Qiskit
- Circuit metrics (gate count, circuit depth)
- Transpilation and simulation times

# QVMP

- Quantum Verification of Matrix Products
- Given $n \times n$ matrices $A$, $B$ and $C$, check if $AB = C$
- Two quantum algorithms:
  - Grover search based: $O(n^{\frac{7}{4}})$
  - Quantum random walk based: $O(n^{\frac{5}{3}})$

# QVMP Algorithm

**Algorithm 1** Quantum VMP using Grover Search

**Input:** $n \times n$ matrices $A, B, C$
**Output:** 1 if $AB = C$ and 0 otherwise
**Procedure:**

1. Partition $B$ and $C$ into sub-matrices of size $n \times \sqrt{n}$
2. Perform amplitude amplification for $n^{\frac{1}{4}}$ iterations using this subroutine:
   2.1 Pick a random vector $x$ of size $\sqrt{n}$
   2.2 Classically compute $y = B_i x$ and $z = C_i x$
   2.3 Using Grover search with $\sqrt{n}$ iterations, find a row of index $j$ such that $(Ay \neq z)_j$
3. XOR the sub-results

# QVMP Implementation

```
# QVMP oracle described using a classical function

def find_row_mismatch(A, y, z):
  z_prime = A * y
  for j, value in enumerate(z_prime):
    if value != z[j]:
      return j
  return -1
```

- The above snippet is encoded as a quantum circuit and constitutes the oracle
- QROM is used to efficiently encode the matrix
- Out-of-place inner product performs the row-vector multiplication

# QROM - Quantum Read-only Memory

- Encodes an $n \times m$ binary matrix using only $n + \log_2(n)$ qubits
- Outputs the value of the $j$th row indexed using address qubits
- Can use superposition to extract multiple rows

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

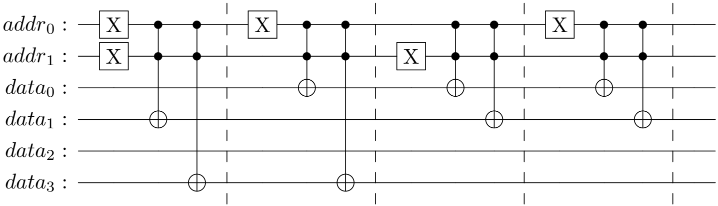



Figure 2: QROM encoding of a $4 \times 4$ matrix $A$

# QVMP oracle

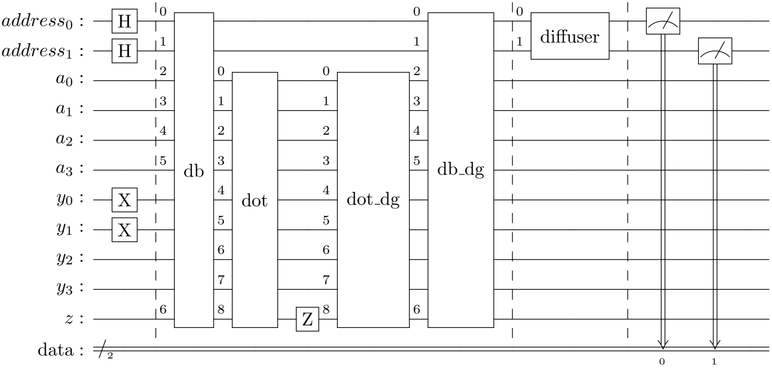



Figure 3: QVMP oracle for a $4 \times 4$ matrix $A$ performing one iteration

# Evaluation

- Aer simulator provided by Qiskit
- Rudimentary noise model
- Testbench specs
  - AMD EPYC 7502 32-Core Processor, 1498.333 MHz
  - 128 CPUs
  - x86_64 architecture
- Simulation methods
  - Statevector: Dense statevector simulation, limited by size
  - Matrix product state (MPS): Tensor-network statevector simulator, doesn't model entire quantum state

# Evaluation - Functionality

- **Input:** $16 \times 16$ matrix $A$ and two vectors $y$ and $z$ with $(Ay \neq z)_j$ for $j \in \{0, 5, 4\}$

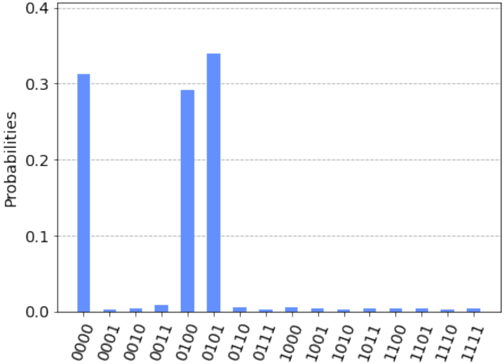



Figure 4: Probability of measuring the row-index $j$ after running the QVMP oracle

# Evaluation - Circuit metrics

| Dimension | Row mismatches | ccx | cx | x | h | z | Circuit Depth | Qubit count | Gate count |
|---|---|---|---|---|---|---|---|---|---|
| (4,4) | 1 | 30 | 1 | 11 | 2 | 1 | 44 | 11 | 49 |
| (16,8) | 2 | 32 | 10108 | 69 | 284 | 2 | 16993 | 21 | 21494 |
| (32,4) | 2 | 24 | 42060 | 204 | 461 | 3 | 69510 | 14 | 85299 |
| (64,8) | 3 | 48 | 300324 | 401 | 1602 | 3 | 497172 | 23 | 604329 |

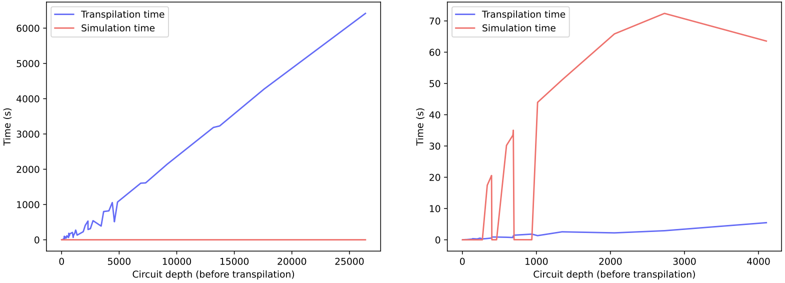
(a) MPS

| Dimension | Row mismatches | ccx | cx | x | h | z | Circuit Depth | Qubit count | Gate count |
|---|---|---|---|---|---|---|---|---|---|
| (4,4) | 1 | 30 | 1 | 11 | 2 | 1 | 44 | 11 | 49 |
| (16,8) | 2 | 32 | 0 | 76 | 4 | 2 | 385 | 21 | 130 |
| (32,4) | 2 | 24 | 0 | 208 | 5 | 3 | 684 | 14 | 270 |
| (64,8) | 3 | 48 | 0 | 405 | 6 | 3 | 2040 | 23 | 498 |

(b) Statevector

Table 1: Circuit metrics for MPS and statevector simulation methods on select dimensions

# Evaluation - Transpilation vs Simulation




(a) MPS

(b) Statevector

Figure 5: Circuit depth vs Transpilation/Simulation time

# Conclusion

- QVMP can be simulated on moderately-sized inputs, but not large enough to observe quantum advantage
- Transpilation time and circuit depth can be a bottleneck when scaling to larger circuits
- Choice of simulation method can alter the size of the transpiled circuit
- Tooling for automated oracle synthesis is limited

# Future work

**Automated synthesis of oracles**

Extend existing work on reversible compilers to support higher-level programming constructs like lists, records, multi-dimensional arrays

**Better encoding of matrices**

Investigate more efficient encodings of matrices and related operations

**Transpilation time bottlenecks**

Investigate bottle-necks in transpilation

# End of talk

Questions?