Experimenting with Fat-Tree QRAM

Mid-term Report

Alex Newsham (amn57), Oliver Fogelin (of284)

Problem

We are reproducing and validating the core performance claims of the Fat-Tree QRAM paper by Xu et al.[1]. Quantum Random Access Memory (QRAM) provides quantum computing algorithms with a memory primitive that can be used to query a classical database in superposition[2]. Many algorithms, including Grover search, HHL, Hamiltonian simulation and quantum machine learning schemes, assume QRAM as an ideal primitive and count only the number of queries[3, 4, 5, 6].

The prevailing architecture for QRAM is bucket-brigade (BB) QRAM[2, 7], which uses a binary tree of quantum routers to route address and bus qubits, achieving $O(\log N)$ latency for a database of size N with $O(N)$ qubits. However, in a shared-memory setting, a single query occupies the entire router path from root to leaves, so concurrent queries from multiple QPUs must effectively serialize. This pushes effective latency to $O(p \log N)$ for p clients and makes QRAM a bandwidth bottleneck for parallel workloads.

Fat-Tree QRAM is proposed as a high-bandwidth shared QRAM that retains BB's $O(N)$ space and error scaling while enabling query-level parallelism[1]. By duplicating routers and interleaving controlled-SWAP and SWAP layers according to a specific schedule, the architecture claims to support $O(\log N)$ concurrent queries and $O(\log N)$ latency, giving bandwidth that is asymptotically independent of N.

Our problem is to test these claims at the circuit level. We will (i) construct explicit quantum circuits for BB and Fat-Tree QRAM from the architectural description and scheduling pseudocode, and verify that both implement the same logical QRAM operations (including pipelined multi-query behaviour), and (ii) measure depth, gate-count, qubit-count, bandwidth and space-time volume as functions of N and query parallelism, and compare to the analytic estimates in Xu et al.[1]. This lets us assess how well the asymptotic advantages of Fat-Tree translate into concrete resource savings in realistic circuit models.

Background

A QRAM exposes a query operation of the form

$$\sum_{i=0}^{N-1} \alpha_i \left|i\right\rangle_A \left|0\right\rangle_B \mapsto \sum_{i=0}^{N-1} \alpha_i \left|i\right\rangle_A \left|x_i\right\rangle_B ,$$

where $\left|i\right\rangle_A$ is an $n = \log N$-qubit address register, $\left|x_i\right\rangle_B$ is the bus register containing the data bit (in this project we primarily consider a single-qubit bus), and $\{x_i\}$ is a classical database.

Bucket-brigade (BB) QRAM arranges quantum "routers" in a binary tree above the classical memory[2, 7]. Each router has a three-state control (wait, route-left, route-right) and uses controlled-SWAP operations to direct address and bus qubits down the

tree. A query is carried out in three stages: loading the address bits into successive tree levels, routing a delocalised bus qubit to the leaves and applying data-controlled gates in parallel, and unloading the address to restore all routers to the idle state. Bit-level pipelining allows different address bits and the bus to be in flight simultaneously, giving $O(\log N)$ latency for a single query and favourable error propagation properties[8]. However, in a shared-memory setting a single query occupies all routers from root to leaves, so concurrent clients must serialize their accesses and effective latency grows as $O(p \log N)$ for p clients.

Fat-Tree QRAM adapts this tree structure by duplicating routers along a new index k and increasing the number of wires between levels, so that each tree level consists of multiple copies of the BB router arranged in a "fat" node[1]. One interpretation suggested by the paper is as a stack of sub-QRAMs of different effective address widths (parameterised by k). Queries move between these sub-QRAMs via local SWAP steps interleaved with standard BB-style gate steps. By carefully scheduling these gate and swap layers (Algorithm 1 in the paper), the architecture realises query-level pipelining: up to $\Theta(\log N)$ distinct queries can coexist in different k-slices of the tree without conflicting on any physical router.

Approach

Our approach is to build circuit-level implementations of BB and Fat-Tree QRAM for capacities $N = 2^n$, validate their behaviour against an ideal classical model for small N, and then use the resulting circuits to calculate depth, gate-count and qubit-count metrics for larger N where full state simulation is infeasible.

For Fat-Tree QRAM, the core challenge is instantiating the three-dimensional routing structure indexed by (i, j, k) and reproducing the scheduling pattern of alternating gate steps and SWAP-I / SWAP-II layers. We have built data structures that map each logical router (level, node index, copy index) to simulator qubits, and implemented gate steps for BB-style address loading and unloading within each fixed-k slice. Swap layers locally exchange input and router qubits between neighbouring k values, following Algorithm 1 in Xu et al.[1]. We have an initial implementation of this full scheduler from the paper and are currently debugging it for small instances by comparing pipelined execution of multiple queries against sequential execution of the same queries (and comparing to ground truth within numerical tolerance).

BB QRAM can be generated as Fat-Tree instances where each node contains a single router (the $k = n - 1$ slice), so both architectures are produced by the same generator and share the validation framework.

Evaluation

We will first validate correctness for small N by comparing BB QRAM outputs against an ideal classical lookup, and Fat-Tree outputs (with a single query in flight) against BB. We will then test the Fat-Tree pipeline by running several logically independent queries with different address superpositions through the shared QRAM and checking that the joint output distribution matches sequential execution. For selected small instances we also plan to run the circuits on IBM's cloud hardware to cross-check simulator behaviour with empirical results.

Beyond correctness, we will use the generated circuits to extract microarchitectural and algorithm-level metrics and compare them to the analytic formulas in Xu et al.[1]. For a range of N we will measure qubit counts, logical circuit depth for single and multiple queries, amortised per-query latency, query parallelism and derived bandwidth

/ space-time volume, and embed both BB and Fat-Tree QRAM into simple workloads (e.g. alternating "processing" blocks and QRAM queries, and a small parallel Grover-style search) to compare end-to-end depth and QRAM utilisation. We will then contrast these results with the values and trends reported in Tables 1 and 2 and the figures of Xu et al.[1], and analyse any discrepancies in terms of gate decompositions, layer-counting conventions, and simplifications in our simulator (such as treating all layers as equal cost). Using gate-type counts from our circuits and the error model of Xu et al. Sec. 8.1, we will also verify that our implementation reproduces the predicted $O(\epsilon \log^2 N)$ infidelity scaling for both architectures.

Finally, we also aim to evaluate how well-suited this architecture is to different quantum platforms (superconducting, trapped ion & neutral atom). For example, neutral atom qubits allow us to move the qubits physically which will remove much of the cost caused by many long-distance swaps when moving a query to the next router tree in the QRAM.

References

[1] Sifan Xu et al. "Fat-Tree QRAM: A High-Bandwidth Shared Quantum Random Access Memory for Parallel Queries". In: *Proceedings of the 2025 ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2025.

[2] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum random access memory". In: *Physical Review Letters* 100.16 (2008), p. 160501.

[3] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. ACM, 1996, pp. 212–219.

[4] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical Review Letters* 103.15 (2009), p. 150502.

[5] Guang Hao Low and Isaac L. Chuang. "Hamiltonian simulation by qubitization". In: *Quantum* 3 (2019), p. 163.

[6] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202.

[7] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Architectures for a quantum random access memory". In: *Physical Review A* 78.5 (2008), p. 052310.

[8] Srinivasan Arunachalam et al. "On the robustness of bucket brigade quantum RAM". In: *New Journal of Physics* 17.12 (2015), p. 123010.