Retrieval-Augmented Generation for Cirq Quantum Code Generation: A Multi-Agent, Tool-Augmented, RL-Enhanced LLM Approach

Umer Farooq, Hussain Waseem Syed, and Muhammad Irtaza Khan

FAST NUCES Islamabad, Department of Computer Science
i220891@nu.edu.pk, i220893@nu.edu.pk, i220911@nu.edu.pk

Abstract. We propose a specialized Retrieval-Augmented Generation (RAG) system for generating accurate Cirq quantum computing code from natural language descriptions. Our approach combines a curated knowledge base of Cirq implementations with large language models to produce executable quantum circuits with comprehensive explanations. The system addresses the growing need for accessible quantum programming tools by providing an AI-powered assistant specifically tailored for Google's Cirq framework. We will evaluate our approach using standard quantum algorithms including VQE, QAOA, and quantum teleportation, with the goal of improving code accuracy and educational value compared to baseline LLM-only approaches. Our goal is to achieve over 90% syntactically correct code generation with comprehensive explanations, making quantum programming more accessible to students and researchers.

Keywords: Retrieval-Augmented Generation · Cirq · Quantum Computing · Code Generation · Educational AI · Multi-Agent Systems · Tool-Augmented LLMs · Agentic Reinforcement Learning

1 Introduction

Quantum computing represents a paradigm shift in computational capabilities, promising exponential speedups for specific problems. However, the steep learning curve associated with quantum programming frameworks presents a significant barrier to adoption. Google's Cirq framework, while powerful, requires deep understanding of quantum mechanics and Python programming, making it challenging for newcomers to the field.

The complexity of quantum programming is compounded by the need to understand both quantum mechanical principles and framework-specific syntax. Traditional approaches to learning quantum programming rely heavily on documentation and examples, which can be overwhelming for beginners. This creates a significant gap between theoretical understanding and practical implementation.

Recent advances in Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) systems offer promising solutions to this challenge [7]. RAG

systems combine the generative capabilities of LLMs with external knowledge bases, enabling more accurate and contextually relevant responses. When applied to quantum code generation, RAG systems can provide not only correct code but also educational explanations and best practices, as demonstrated by recent work in quantum code transpilation [6].

Our work introduces a specialized RAG system for Cirq quantum code generation that addresses these challenges through a curated knowledge base and intelligent retrieval mechanisms. The system is designed to serve both educational and practical purposes, aiming to help users learn quantum programming while generating production-ready code.

2 Related Work

2.1 Quantum Code Generation

Recent work has explored the application of AI to quantum programming. Several studies have demonstrated the potential of LLMs for generating quantum code, with varying degrees of success across different frameworks. However, most existing approaches focus on either code generation, explanation generation, or code optimization not all three together.

2.2 Retrieval-Augmented Generation

RAG systems have shown significant promise in improving the accuracy and reliability of LLM outputs by incorporating external knowledge. The approach has been successfully applied to various domains, including code generation, but has not been extensively explored in the context of quantum computing.

2.3 Educational AI Systems

AI-powered educational tools have gained traction in computer science education, with systems designed to provide personalized learning experiences and automated code generation. However, the unique challenges of quantum computing education require specialized approaches that combine domain expertise with AI capabilities.

2.4 Quantum Computing Education

The field of quantum computing education has seen growing interest, with various tools and platforms designed to make quantum concepts more accessible. However, there remains a significant gap in AI-powered tools specifically designed for quantum programming education.

2.5 Foundation Research and Building Blocks

Our work builds upon several recent advances in RAG systems and quantum programming assistance. Siavash and Moin [7] present a comprehensive model-driven approach to quantum code generation using RAG, demonstrating significant improvements in code accuracy through sophisticated retrieval mechanisms and context-aware generation strategies. Their work establishes the foundation for domain-specific RAG applications in quantum computing, providing crucial insights into retrieval optimization and knowledge base construction for quantum programming tasks.

Building on this foundation, Siavash and Moin [6] introduce advanced techniques for improving low-level quantum code quality using LLMs. Their approach demonstrates the importance of framework-specific knowledge in quantum programming assistance, which informs our focus on Cirq-specific optimization.

Dupuis et al. [3] present the Qiskit Code Assistant, a specialized LLM system for generating quantum computing code. Their work demonstrates the effectiveness of framework-specific training and fine-tuning approaches, providing valuable insights into how to adapt general-purpose LLMs for quantum programming tasks. Their evaluation methodology and training strategies directly inform our approach to developing Cirq-specific code generation capabilities.

Jiménez-Navajas et al. [5] explore code generation for classical-quantum software systems using UML modeling approaches. Their work provides crucial insights into systematic approaches for AI-powered code generation, emphasizing the importance of robust evaluation frameworks and quality assessment methodologies. Their systematic approach provides essential guidelines for implementing reliable and maintainable AI code generation systems in quantum computing contexts.

Basit et al. [1] introduce PennyLang, a pioneering LLM-based quantum code generation system with a novel PennyLane-centric dataset. Their work addresses the critical challenge of dataset curation and quality assessment for quantum programming education, providing methodologies that we adapt for our Cirq-specific knowledge base construction. Their approach to creating framework-specific datasets directly informs our methodology for ensuring high-quality training data and evaluation metrics.

Finally, Campbell et al. [2] present comprehensive multi-agent optimization and quantum error correction approaches for LLM-based quantum code generation. Complementary to this, QUASAR by Yu et al. [8] demonstrates tool-augmented LLMs with agentic reinforcement learning (RL) for low-level quantum code generation, and Agent-Q by Jern et al. [4] explores fine-tuning LLMs specifically for quantum circuit generation and optimization. Together, these works inform our approach to combining strong circuit priors (Agent-Q-style fine-tuning) with in-loop tool-augmented optimization (QUASAR-style) within a multi-agent framework.

2.6 Our Contribution and Innovation

While these foundational works provide excellent starting points, our contribution lies in creating the first specialized hybrid RAG + Multi-Agent system specifically designed for Google's Cirq framework. We extend the model-driven RAG approach from [7] by implementing a multi-agent architecture that combines specialized agents for different quantum programming tasks. Our system focuses on Cirq-specific optimization and educational assistance.

We adapt the framework-specific training methodologies from [3] for Cirq development, implementing specialized fine-tuning approaches and evaluation strategies tailored to Cirq's unique characteristics. Our system architecture follows the systematic code generation approaches outlined in [5], ensuring robust evaluation frameworks and quality assessment methodologies throughout the development process.

The dataset curation and quality assessment methods from [1] directly inform our approach to constructing a comprehensive Cirq-specific knowledge base, ensuring high-quality training data and educational content. We extend the multi-agent optimization framework from [2] to create specialized agents for circuit design, optimization, validation, and education, establishing new benchmarks for quantum programming assistance systems.

Our key innovation lies in the integration of these approaches into a unified, Cirq-focused hybrid system that addresses the specific challenges of quantum programming education while maintaining the rigor and quality standards established by these foundational works. The system combines the strengths of RAG-based knowledge retrieval with multi-agent coordination to provide comprehensive quantum programming assistance.

3 Theoretical Foundations Used

- Transformer Architecture
- Large Language Models (LLMs)
- Retrieval-Augmented Generation (RAG)
- Semantic Vector Search
- Contrastive Sentence Embeddings
- Multi-Agent Systems
- Tool-Augmented Reasoning and Function Calling
- Agentic Reinforcement Learning (RL)
- Prompt Engineering
- In-Context Learning (Few-Shot)
- Supervised Fine-Tuning
- Quantum Circuit Model
- Variational Quantum Algorithms (VQE, QAOA)
- Quantum Measurement and Simulation

4 Methodology

4.1 System Architecture

Our hybrid RAG + Multi-Agent system for Cirq code generation consists of five main components: a knowledge base, a retrieval system, an orchestration layer, specialized agents, and a generation system. The knowledge base contains curated Cirq implementations, documentation, and educational content, following the dataset curation methodologies from [1]. The retrieval system uses semantic search to find relevant information for user queries, implementing the advanced RAG techniques from [7]. The orchestration layer coordinates between the RAG system and specialized agents, while the multi-agent system includes Circuit Designer, Optimizer, Validator, and Educational agents, extending the multi-agent framework from [2]. The generation system combines retrieved information with LLM capabilities to produce accurate code and explanations.

4.2 Knowledge Base Construction

We construct a comprehensive knowledge base by curating Cirq implementations from various sources, including official documentation, tutorials, and research papers. Following the dataset curation methodologies proposed by Basit et al. [1], we implement systematic quality assessment frameworks to ensure high-quality training data. Their approach demonstrated significant improvements in code generation accuracy, achieving 58.2% accuracy with GraphRAG-enhanced pipeline compared to 20.5% without RAG, using a dataset of 3,347 PennyLane-specific quantum code samples.

Our Cirq-centric dataset construction follows their structured methodology for data curation, annotation, and formatting to enhance LLM usability. Each entry includes the code implementation, natural language description, and educational explanations, following the multi-modal approach outlined in their PennyLane-centric dataset construction. The knowledge base is organized to support both specific algorithm implementations and general quantum programming concepts, incorporating the advanced RAG techniques from Siavash and Moin [7] for optimal retrieval performance. We apply the quality assessment criteria from their work to ensure that each knowledge base entry meets high standards for accuracy, completeness, and educational value.

The dataset construction process involves three main phases: (1) Data Collection from official Cirq documentation, GitHub repositories, and quantum computing textbooks, (2) Data Annotation with contextual descriptions and educational explanations, and (3) Quality Assessment using automated validation and human expert review to ensure code correctness and educational value.

4.3 Retrieval Mechanism

The retrieval system uses semantic search to identify relevant knowledge base entries for user queries, implementing the advanced RAG techniques from Siavash

and Moin [7]. We employ sentence transformers to encode both queries and knowledge base entries, enabling efficient similarity search with context-aware retrieval strategies. The system retrieves the most relevant entries and provides them as context for code generation, following the scientific computing methodologies adapted for quantum programming contexts.

4.4 Multi-Agent Code Generation

The generation system employs a multi-agent architecture to produce Cirq code based on user queries and retrieved context, extending the multi-agent optimization framework from Campbell et al. [2]. We pair an Agent-Q–style fine-tuned LLM for higher-quality initial drafts [4] with QUASAR-style tool-augmented agentic RL [8] to iteratively refine circuits using compile/simulate metrics as rewards. This yields better first-pass validity and fewer optimization iterations.

Our system includes four specialized agents, each with distinct responsibilities:

Circuit Designer Agent: Creates quantum circuits based on user specifications, implementing the model-driven approach from Siavash and Moin [7]. This agent translates natural language descriptions into Cirq circuit implementations, leveraging the retrieved context from the RAG system.

Optimizer Agent: Optimizes generated circuits for performance and resource efficiency, incorporating the optimization strategies from Campbell et al. [2]. This agent reduces circuit depth, minimizes gate count, and optimizes for specific quantum hardware constraints.

Validator Agent: Validates generated code for syntax correctness, logical consistency, and quantum mechanical principles, following the systematic validation approaches from Jiménez-Navajas et al. [5]. This agent ensures that all generated code is syntactically correct and follows quantum computing best practices.

Educational Agent: Provides comprehensive explanations, learning content, and step-by-step breakdowns of quantum algorithms, incorporating the educational methodologies from Basit et al. [1]. This agent enhances the learning experience by providing detailed explanations and educational context.

Each agent is fine-tuned using the framework-specific training methodologies from Dupuis et al. [3], adapted for Cirq development. The system employs advanced prompt engineering techniques and agent coordination protocols to ensure consistent and high-quality outputs, with agents collaborating through a centralized orchestration layer.

4.5 Evaluation Framework

We evaluate our system using a comprehensive set of metrics including code accuracy, educational value, and user satisfaction, following the evaluation framework proposed by Campbell et al. [2]. Their multi-agent optimization approach demonstrated significant improvements in quantum code generation quality through specialized agent coordination and quantum error correction mechanisms.

Our evaluation methodology incorporates multiple assessment dimensions:

Technical Accuracy Metrics: Following the framework-specific evaluation strategies from Dupuis et al. [3], we assess code generation accuracy using metrics such as syntax correctness (targeting ¿90%), functional correctness, and execution success rates. We implement automated validation using Cirq's built-in circuit validation and quantum simulator testing.

Educational Value Assessment: Incorporating the quality metrics from Basit et al. [1], we evaluate the educational content quality through expert review, user feedback, and learning outcome assessments. This includes measuring explanation clarity, step-by-step breakdown completeness, and conceptual understanding improvement.

Multi-Agent Performance: We assess individual agent performance and overall system coordination using the multi-agent evaluation approaches from Campbell et al. [2]. This includes measuring agent collaboration efficiency, task completion rates, and error reduction through agent coordination.

We implement both automated metrics and human evaluation to ensure comprehensive assessment of system performance, incorporating the software engineering quality assessment methodologies from Jiménez-Navajas et al. [5]. Our evaluation methodology extends the benchmarking approaches from their work while adapting them specifically for Cirq code generation tasks, ensuring rigorous assessment of both technical accuracy and educational value.

5 Experimental Setup

5.1 Dataset

We construct a comprehensive dataset of quantum algorithms and implementations for evaluation, following the dataset construction methodology from Basit et al. [1]. Our Cirq-centric dataset includes standard algorithms such as VQE, QAOA, quantum teleportation, Grover's algorithm, and quantum Fourier transform, with implementations ranging from simple single-qubit operations to complex multi-qubit algorithms.

The dataset construction follows a three-tier approach: (1) **Basic Operations** including single-qubit gates, two-qubit gates, and measurement operations, (2) **Standard Algorithms** covering well-known quantum algorithms with varying complexity levels, and (3) **Advanced Applications** including quantum machine learning, optimization, and error correction implementations. Each entry includes the Cirq code implementation, natural language description, educational explanations, and expected outputs, following the multi-modal approach from the PennyLang dataset construction methodology.

We target a dataset size of approximately 2,500-3,000 Cirq-specific implementations, comparable to the 3,347 samples used in the PennyLang study, ensuring comprehensive coverage of quantum programming concepts and maintaining high quality through automated validation and expert review processes.

5.2 Baseline Methods

We compare our hybrid RAG + Multi-Agent approach against several baseline methods to demonstrate the effectiveness of our integrated system. Following the evaluation methodology from Basit et al. [1], we establish the following baselines:

Direct LLM Generation: We compare against direct LLM generation without retrieval augmentation, similar to the baseline approach that achieved 20.5% accuracy in the PennyLang study. This baseline demonstrates the improvement gained through RAG integration.

Traditional RAG System: We evaluate against a standard RAG system without multi-agent coordination, implementing the basic retrieval-augmented generation approach to isolate the benefits of our multi-agent architecture.

Existing Quantum Programming Tools: We compare against existing quantum programming assistance tools and frameworks, including the Qiskit Code Assistant approach from Dupuis et al. [3], adapted for Cirq to ensure fair comparison.

Single-Agent Systems: We evaluate individual agents in isolation to demonstrate the collaborative benefits of our multi-agent approach, following the multi-agent evaluation methodology from Campbell et al. [2].

The comparison includes both quantitative metrics (code accuracy, execution success rates, response times) and qualitative assessment of generated code quality, educational value, and user satisfaction, ensuring comprehensive evaluation of our system's performance improvements.

5.3 Ablation Studies

We will run a focused set of ablations to capture the largest effects with minimal variants:

- Retrieval: *No-RAG* (prompt-only) vs *RAG* (semantic retrieval).
- Orchestration: *Single-agent* (Designer only) vs *Multi-agent (no RL)* (Designer, Optimizer heuristic, Validator, Educational).
- Model initialization: base LLM vs *Agent-Q–style* fine-tuned LLM for circuit drafting.
- Optimization: heuristic/deterministic passes vs *QUASAR-style tool-augmented RL* (compile/simulate metrics as rewards).
- Key role ablation: *Multi-agent (no RL)* with and without the Optimizer to quantify optimization impact.

For each variant, we will report: compile success rate, functional correctness (unit/equivalence tests), circuit depth, two-qubit gate count, simulated fidelity under noise, number of tool calls/iterations, and wall-clock time to reach a passing solution. Where applicable, we will also report explanation quality (expert rating).

5.4 Evaluation Metrics

Our evaluation includes multiple metrics to assess different aspects of system performance:

- Code accuracy: Percentage of syntactically correct and executable code
- Functional correctness: Percentage of code that produces expected results
- Educational value: Quality of explanations and learning content
- User satisfaction: Feedback from users testing the system

6 Results

6.1 Code Generation Performance

We will evaluate the RAG system against baseline methods to measure improvements in code generation accuracy. The target is to achieve over 90% syntactically correct code generation, alongside comprehensive explanations and educational content.

6.2 Educational Impact

We will assess educational value through detailed explanations and step-by-step breakdowns of quantum algorithms. We aim to improve users' understanding of quantum programming concepts and will measure learning outcomes through user studies.

6.3 User Experience

We will conduct user studies to evaluate satisfaction with the system's ability to generate accurate code and provide educational explanations, and to assess how well the system bridges the gap between theoretical understanding and practical implementation.

7 Discussion

7.1 Limitations

While we aim for significant improvements, there are several limitations to consider. The system's performance will depend on the quality and comprehensiveness of the knowledge base. Additionally, the system may struggle with highly novel or complex quantum algorithms not well-represented in the knowledge base.

7.2 Future Work

Future work could explore expanding the system to support multiple quantum frameworks, improving the knowledge base with more diverse implementations, and developing more sophisticated retrieval mechanisms. Additionally, the system could be enhanced with interactive features and personalized learning capabilities.

7.3 Implications

Our work demonstrates the potential of RAG systems for quantum programming education and assistance. The approach could be extended to other quantum frameworks and adapted for different educational contexts.

8 Conclusion

We propose a specialized RAG system for Cirq quantum code generation that aims to address the challenges of quantum programming education and assistance. We will evaluate whether the system improves code accuracy and educational value compared to baseline approaches. This work is intended to contribute to the growing field of AI-powered educational tools and to provide a foundation for future research in quantum computing education.

We aim for the system to generate accurate code and provide educational explanations, making it a valuable tool for students and researchers learning quantum programming. Future work will focus on expanding the system's capabilities and improving its performance across a wider range of quantum algorithms and use cases.

Acknowledgments

We thank the quantum computing community for their contributions to open-source tools and educational resources that made this work possible.

References

1. Basit, A., Innan, N., Asif, M.H., Shao, M., Kashif, M., Marchisio, A., Shafique, M.: Pennylang: Pioneering llm-based quantum code generation with a novel pennylane-centric dataset (2025), https://arxiv.org/abs/2503.02497
2. Campbell, C., Chen, H.M., Luk, W., Fan, H.: Enhancing llm-based quantum code generation with multi-agent optimization and quantum error correction (2025), https://arxiv.org/abs/2504.14557
3. Dupuis, N., Buratti, L., Vishwakarma, S., Forrat, A.V., Kremer, D., Faro, I., Puri, R., Cruz-Benito, J.: Qiskit code assistant: Training llms for generating quantum computing code. In: 2024 IEEE LLM Aided Design Workshop (LAD). pp. 1–4 (2024). https://doi.org/10.1109/LAD62341.2024.10691762
4. Jern, L., Uotila, V., Yu, C., Zhao, B.: Agent-q: Fine-tuning large language models for quantum circuit generation and optimization (2025), https://arxiv.org/abs/2504.11109
5. Jiménez-Navajas, L., Pérez-Castillo, R., Piattini, M.: Code generation for classical-quantum software systems modeled in uml. Software and Systems Modeling **24**(3), 795–821 (2025). https://doi.org/10.1007/s10270-024-01259-w
6. Siavash, N., Moin, A.: Llm-powered quantum code transpilation (2025), https://arxiv.org/abs/2507.12480
7. Siavash, N., Moin, A.: Model-driven quantum code generation using large language models and retrieval-augmented generation (2025), https://arxiv.org/abs/2508.21097
8. Yu, C., Uotila, V., Deng, S., Wu, Q., Shi, T., Jiang, S., You, L., Zhao, B.: Quasar: Quantum assembly code generation using tool-augmented llms via agentic rl (2025), https://arxiv.org/abs/2510.00967