AI-Driven Quantum Error Correction Circuit Synthesis

Bridging Stabilizer Theory and Fault-Tolerant Implementation

01 PROJECT OVERVIEW

The Challenge

Quantum error correction codes are traditionally defined using stabilizers, but translating these abstract mathematical structures into fault-tolerant quantum circuits is complex and hardware-dependent.

Our Goal

Develop an AI agent that automatically synthesizes optimized quantum circuits for error correction, tailored to specific hardware architectures and noise models.

02 FOUNDATIONAL CONCEPTS

Quantum Building Blocks

- **Quantum Gates:** Single-qubit (H, X, Z) and two-qubit (CNOT) operations
- **Bell States:** Maximally entangled two-qubit states ($|\Phi^+\rangle$, $|\Phi^-\rangle$, $|\Psi^+\rangle$, $|\Psi^-\rangle$)
- **CAT States:** Superposition states of macroscopic quantum systems
- **Quantum Circuits:** Sequential gate operations for quantum computation

Error Correction Fundamentals

- **Stabilizer Formalism:** Uses commuting Pauli operators to define protected quantum subspaces where information is encoded and errors can be detected
- **Parity Measurement Circuits:** Process of measuring error patterns without destroying the encoded quantum state, enabling error identification and correction
- **Fault-Tolerant State Preparation:** Operations designed so single errors don't cascade into multiple uncorrectable errors, maintaining code distance throughout computation

03 CIRCUIT EXAMPLES

Five-Qubit CAT State Preparation

Prepares the state $|00000\rangle + |11111\rangle$

```
q0: -H-@-@-@-@-
       | | | |
q1: ---X-|-|-|-
         | | |
q2: -----X-|-|-
           | |
q3: -------X-|-
             |
q4: ---------X-
```

Custom Flag Circuit

Flag-based fault-tolerant circuit

```
q0: -X-----X-
     |     |
q1: -@-@-@-@-
       | |
q2: ---|-X---
       |
q3: ---X-----
```

04 PROBLEM STATEMENT & APPROACH

The Core Challenge

Quantum error correction exists in a difficult middle ground: stabilizers provide an elegant mathematical description of error-correcting codes, but hardware requires explicit circuit implementations. This translation is non-trivial because:

- **Hardware Heterogeneity**
- **Fault-Tolerance Requirements**
- **Optimization Trade-offs**
- **The Compilation Gap**

Current Limitations

Manual Design Approach:

- Experts hand-craft circuits for each error correction code
- Process takes weeks to months per code
- Requires deep expertise in both QEC theory and hardware constraints
- Not scalable as codes and hardware evolve

Existing Compilation Tools:

- General-purpose compilers optimize for gate count but ignore fault-tolerance
- Focus on noise-free compilation, failing in realistic NISQ settings
- Don't account for architecture-specific constraints during synthesis
- Cannot explore the vast space of functionally equivalent circuits

Our LLM/AI Agent Approach

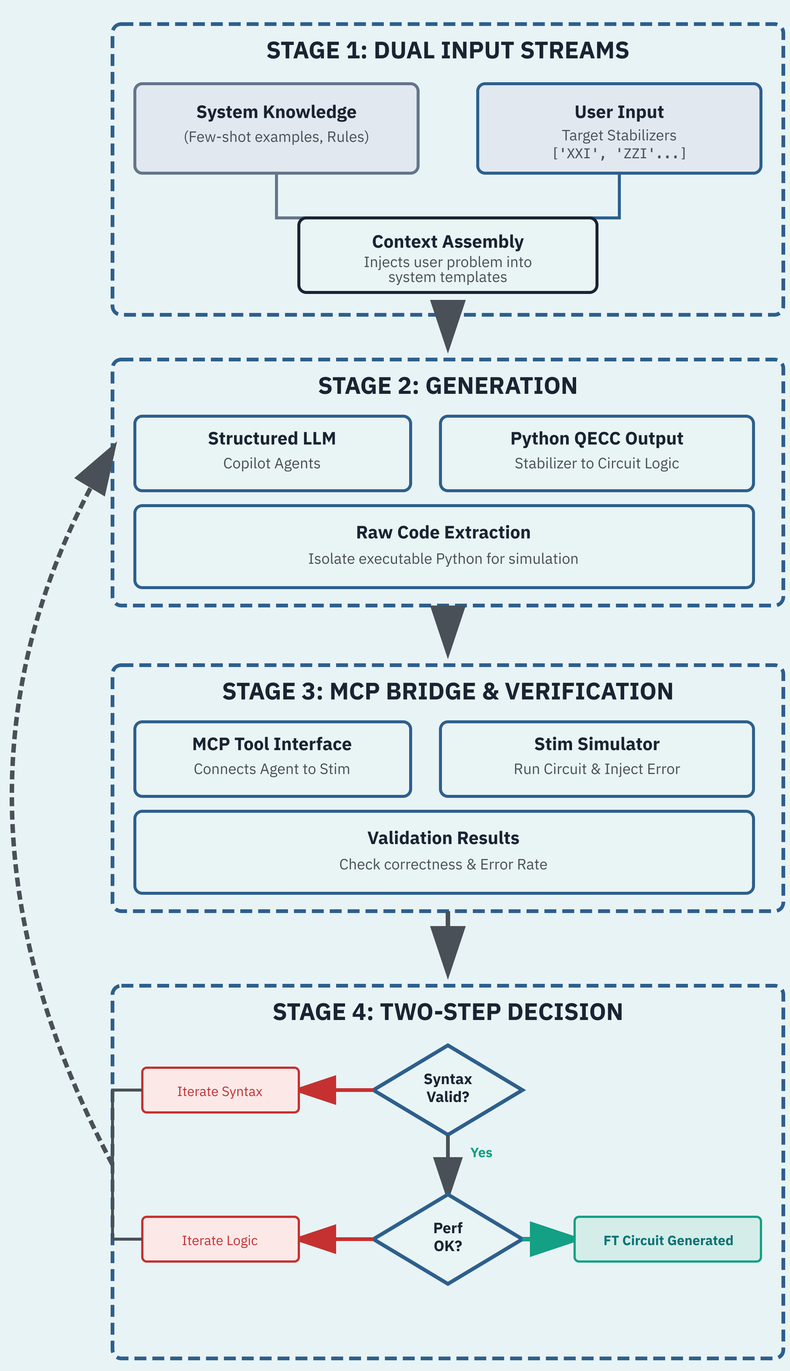
Why This Works:

- **Pattern Learning:** LLMs can learn the implicit rules of fault-tolerant circuit design from examples
- **Contextual Adaptation:** Agents can incorporate hardware constraints, noise models, and architectural details directly
- **Iterative Refinement:** Through validation loops with simulators (STIM), agents learn which patterns succeed and fail
- **Exploration at Scale:** AI can rapidly explore millions of candidate circuits, finding non-obvious solutions

Key Advantages:

1. **Automated Discovery:** Generate novel circuit implementations without human intervention
2. **Hardware-Aware:** Tailor circuits to specific platform constraints (connectivity graphs, gate fidelities, coherence times)
3. **Noise-Conscious:** Optimize for realistic error rates rather than idealized assumptions
4. **Rapid Iteration:** Test multiple approaches in hours rather than weeks
5. **Knowledge Transfer:** Learn from one code/platform and apply insights to new ones

05 CURRENT PIPELINE



06 RESEARCH ROADMAP

Near-Term Objectives

- **Prompt Optimization:**
- **Validation Metrics Expansion:**
- **Noise Integration:**
- **Fault-Tolerance Verification:**

Medium-Term Goals

- **Cross-Architecture Validation:**
- **End-to-End Toolchain:**
- **Benchmarking Study:**

Members: Christian Tarta Mayee Sun, Sarju Patel, Sylvie Lausier (Mentor: Andres Paz)