Project Proposal: Enhancing a simulator for quantum stabilizer circuits

October 21, 2024

1 Introduction

Quantum programs are written using logical qubits. However, due to limitations on physical quantum computers, in order to achieve a single logical qubit, several data qubits are required. This is because during execution the individual qubits are subject to lots of different types of noise. Furthermore, if a gate is miscalibrated, due to the linear nature of most quantum gates (since the operations have to be unitary), the error accumulates over time. Quantum error correction involves periodically measuring certain ancillary qubits and then using the outcomes to detect errors and then correct them without compromising any existing entanglement.

To ensure that the error correction code works correctly simulation is done. Stim is a simulator for quantum error correction code that runs faster than other simulators.

1.1 Description

The main aim of the project is the development and implementation of an algorithm that translates graphically represented error codes into Stim input text strings. The project aims to design and write an application that can receive as input a quantum error correction code that has been represented graphically, and, given the input, outputs the corresponding Stim python script. The produced application should act as an abstraction for being able to use a high speed stabiliser circuit simulator without understanding how Stim works directly. Understanding the dissertation will not require the reader to have an in-depth understanding of quantum error correction.

2 Starting Point

I am in the process of familiarising myself with quantum error correction as well as learning more about quantum computation in general. No design decisions have been made nor code written.

3 Project Implementation

1. Familiarise myself with different error correction codes and how Stim works
    - General overview: *Quantum Computer Science: An Introduction* by David Mermin [Mer07]
    - Steane and Shor codes: [HBC23]
    - Surface area codes: [Ste14] [Hor+12] [GMB19]
    - Color code lattices: [LR14] [LAR11]
    - Other codes: [PaB24]
    - About Stim: [Gid21]
2. Manually write out the Stim code for around five different error correction codes.
3. Develop an algorithm for how to automate going from the graphical representation of the error correcting code to the Stim code for it.
4. Design a very simple user interface for drawing out the error correction circuits, this could be using an existing circuit drawing library, or potentially with representing them as graphs.
5. Implement the front end and the main body of the code. The aim of the program is that given an input of an arbitrary error correction code it outputs the corresponding Stim code. This will be a separate application that can work with Stim.

3.1 Extensions

Currently, the project involves constructing a very simple user interface tool in order to get the input of an arbitrary error correction code. An extension would be to do proper user research when designing the user interface.
An alternative extension, would be to research what the main file type is that researchers are using to create the error correction codes and adapt the application so that it uses that filetype as an input.
A further extension, would be to post-process the Stim output so that the results could be represented through the application developed. This would mean that the user would not need to use Stim directly and the results could be manipulated so that they correspond with the user's graphical input.

4 Project Evaluation

There are papers (for example [Ste14], [PaB24]) where given an error correction code it specifies the threshold and other properties of it. I could construct the error correction code in my designed user interface, and then after running it

in Stim, I could check if I get the same results for the threshold. I could set aside an evaluation set of error correction codes (which I did not do manually initially) for this testing.

5 Resources Required

I will use my personal laptop for both code development and dissertation writing. I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure. I will back up my code base in GitHub and write my dissertation in Overleaf. Any other software or libraries that I use will be open source. Stim software has an Apache license so is open source.

6 Timetable and Milestones

1. Michaelmas weeks 1–2 (10/10 - 23/10)
   Planned work
   - Further research into quantum error correction and different types of quantum error correction codes, including Shor code and Steane code.
   - Familiarise myself with how Stim inputs work.
2. Michaelmas weeks 3–4 (24/10 - 06/11)
   Planned work
   - Write out the Stim implementations for five different error correction codes.

   Milestones
   - Implemented five different error correction codes.
3. Michaelmas weeks 5–6 (07/11 - 20/11)
   Planned work
   - Devise an algorithm for this conversion for the conversion from graphical representation to Stim code.
4. Michaelmas weeks 7–8 (21/11 - 04/12)
   Planned work
   - Design a simple user interface for error correction code representation.
   - Start developing the front end.

   Milestones

- Design decisions made.

5. Christmas Holidays (05/12 - 22/01)
   Planned work

   - Finish developing the front end.
   - Start developing the implementation of the algorithm and integrating it with the front end.

   Milestones

   - Front end of the application is complete.

6. Lent weeks 1–2 (23/01 - 05/02)
   Planned work

   - Finish developing the core application.
   - Test the implementation against error correction codes and ensure that it works correctly.
   - Correct the code, and repeat testing.
   - Choose an extension and start working on it.

   Milestones

   - Complete draft code base.

7. Lent weeks 3–4 (06/02 - 19/02)
   Planned work

   - Continue developing extension.
   - Prepare the progress report and presentation.

8. Lent weeks 5–6 (20/02 - 05/03)
   Planned work

   - Test the application with at least seven different error correction codes.
   - Finish extension development.

   Milestones

   - Data collected for evaluation section.

9. Lent weeks 7–8 (06/03 - 19/03)
   Planned work

   - Begin the dissertation: write the first two chapters.

10. Easter Holidays (20/03 - 09/04)
    Planned work

- Write dissertation

Milestones

- Full dissertation draft sent for review

11. Easter Holidays (10/04 - 30/04)
Planned work

- Make changes to dissertation based on the feedback

Milestones

- Final dissertation ready

12. Easter weeks 1–3 (01/05 - 16/05)
Planned work

- Submit dissertation
- Submit source code

Milestones

- Final dissertation and source code submitted.

References

[Mer07] N. David Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007.

[LAR11] Andrew J. Landahl, Jonas T. Anderson, and Patrick R. Rice. *Fault-tolerant quantum computing with color codes*. 2011. arXiv: 1108.5738 [quant-ph]. URL: https://arxiv.org/abs/1108.5738.

[Hor+12] Dominic Horsman et al. "Surface code quantum computing by lattice surgery". In: *New Journal of Physics* 14.12 (2012), p. 123011. DOI: 10.1088/1367-2630/14/12/123011. URL: https://dx.doi.org/10.1088/1367-2630/14/12/123011.

[LR14] Andrew J. Landahl and Ciaran Ryan-Anderson. *Quantum computing by color-code lattice surgery*. 2014. arXiv: 1407.5103 [quant-ph]. URL: https://arxiv.org/abs/1407.5103.

[Ste14] Ashley M. Stephens. "Fault-tolerant thresholds for quantum error correction with the surface code". In: *Phys. Rev. A* 89 (2 2014), p. 022321. DOI: 10.1103/PhysRevA.89.022321. URL: https://link.aps.org/doi/10.1103/PhysRevA.89.022321.

[GMB19] M. Gutiérrez, M. Müller, and A. Bermúdez. "Transversality and lattice surgery: Exploring realistic routes toward coupled logical qubits with trapped-ion quantum processors". In: *Phys. Rev. A* 99 (2 2019), p. 022330. DOI: 10.1103/PhysRevA.99.022330. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.022330.

[Gid21] Craig Gidney. "Stim: a fast stabilizer circuit simulator". In: *Quantum* 5 (July 2021), p. 497. ISSN: 2521-327X. DOI: 10.22331/q-2021-07-06-497. URL: https://doi.org/10.22331/q-2021-07-06-497.

[HBC23] Shilin Huang, Kenneth R. Brown, and Marko Cetina. *Comparing Shor and Steane Error Correction Using the Bacon-Shor Code*. 2023. arXiv: 2312.10851 [quant-ph]. URL: https://arxiv.org/abs/2312.10851.

[PaB24] Balint Pato, Judd Will Staples Jr. au2, and Kenneth R. Brown. *Logical coherence in 2D compass codes*. 2024. arXiv: 2405.09287 [quant-ph]. URL: https://arxiv.org/abs/2405.09287.