5 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, given an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

What units are being tested? How? Tools?

- **Qubits and Ion traps are being tested individually through a C++ simulation.**
- **Quantum component with ion traps through the same C++ code, though it would have to be upgraded.**
- **Structures through a 3D design software.**
- **Quantum program testing through algassert.com/quirk.**
- **Laser addressing simulation through whatever software we can find to simulate it.**

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

There are no interfaces within our project as we are making a quantum computer. The programming of this computer would be classical, and as such, outside of the scope of this project.

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

- **The ion traps will interact with each other. We will simulate ion trap units together.**
- **The quantum component will be in the traps.**
- **The ion traps will be assembled into a larger structure.**
- **The quantum programs will run on the computer. This is important because some problems can be solved with programs.**
- **Lasers will interact on the ion traps.**

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

The primary testing will be the overall structure and how it works together, as it is the most important part we are innovating. The 3D testing, quantum component of ion traps, and the laser addressing will be enough to be considered system testing.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

We will test as we go, at each step to ensure that no addition of components of code breaks the system. Whenever one change will impact others, we will retest. This shouldn't happen too frequently as most tests are relatively contained from the rest. Many components test an individual aspect of the design, and given correct testing in the previous iterations, we should be able to iterate conveniently.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

- **If ions can be moved between ion traps, that would be a measure of proper function**
- **If ion traps can be organized in a way conducive to the tradeoff of ions between traps and nodes, that would be a measure of proper function**
- **If we can fit the ion traps within a reasonable form factors conducive to efficient computing, that would be a non-functional requirement met**
- **The designation of these things is not only based on the client's desire, but on physical properties (space, electro-dynamics)**

5.7 Security Testing (if applicable)

There will not be a security component to our computer within the scope of our project. While all physical systems of this manner will have a cybersecurity component, the security component of this computer will end up being outside the scope of our program.

5.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

We hope to have a working simulation of our quantum computer in c++ and have an accurate paper about it. The testing will verify whether or not our physical guesses translate well when simulated. Our requirements are designated by us, so our compliance with the requirements will be designated by us. Here is an example of the code we are using for rudimentary testing:

ion_trap_sim.cpp 9+

```cpp
//I'm generating this with mathematica code rn
//This should be first the mins, then the maxes in x y z order
//x is the axis for the ions, z is perpindicular to the trap surface
//so Controlm31 is in the -y direction
std::multimap<std::string, std::array<float, 6>> trapGeom{
  {"RF1", {{-800.0f, 245.0f, -2.0f, 800.0f, 500.0f, 0.0f}}},
  {"RF1", {{-800.0f, -500.0f, -2.0f, 800.0f, -245.0f, 0.0f}}},
  {"RFouter1", {{-800.0f, 150.0f, -2.0f, 800.0f, 245.0f, 0.0f}}},
  {"RFouter1", {{-800.0f, -245.0f, -2.0f, 800.0f, -150.0f, 0.0f}}},
  {"Ground1", {{-800.0f, 55.0f, -2.0f, 800.0f, 150.0f, 0.0f}}},
  {"Ground1", {{-800.0f, -150.0f, -2.0f, 800.0f, -55.0f, 0.0f}}},
  {"Controlm11", {{-100.0f, 30.0f, -14.0f, 0.0f, 55.0f, -12.0f}}},
  {"Controlm11", {{-100.0f, -55.0f, -14.0f, 0.0f, -30.0f, -12.0f}}},
  {"Controlm21", {{-200.0f, 30.0f, -14.0f, -100.0f, 55.0f, -12.0f}}},
  {"Controlm21", {{-200.0f, -55.0f, -14.0f, -100.0f, -30.0f, -12.0f}}},
  {"Controlm31", {{-300.0f, 30.0f, -14.0f, -200.0f, 55.0f, -12.0f}}},
  {"Controlm31", {{-300.0f, -55.0f, -14.0f, -200.0f, -30.0f, -12.0f}}},
  {"Control11", {{0.0f, 30.0f, -14.0f, 100.0f, 55.0f, -12.0f}}},
  {"Control11", {{0.0f, -55.0f, -14.0f, 100.0f, -30.0f, -12.0f}}},
  {"Control21", {{100.0f, 30.0f, -14.0f, 200.0f, 55.0f, -12.0f}}},
  {"Control21", {{100.0f, -55.0f, -14.0f, 200.0f, -30.0f, -12.0f}}},
  {"Control31", {{200.0f, 30.0f, -14.0f, 300.0f, 55.0f, -12.0f}}},
  {"Control31", {{200.0f, -55.0f, -14.0f, 300.0f, -30.0f, -12.0f}}},
  {"RF2", {{-245.0f, 800.0f, 122.0f, -500.0f, -800.0f, 120.0f}}},
  {"RF2", {{245.0f, 800.0f, 122.0f, 500.0f, -800.0f, 120.0f}}},
  {"RFouter2", {{-150.0f, 800.0f, 122.0f, -245.0f, -800.0f, 120.0f}}},
  {"RFouter2", {{150.0f, 800.0f, 122.0f, 245.0f, -800.0f, 120.0f}}},
  {"Ground2", {{-55.0f, 800.0f, 122.0f, -150.0f, -800.0f, 120.0f}}},
  {"Ground2", {{55.0f, 800.0f, 122.0f, 150.0f, -800.0f, 120.0f}}},
  {"Controlm12", {{-30.0f, 100.0f, 134.0f, -55.0f, 0.0f, 132.0f}}},
  {"Controlm12", {{55.0f, 100.0f, 134.0f, 30.0f, 0.0f, 132.0f}}},
  {"Controlm22", {{-30.0f, 200.0f, 134.0f, -55.0f, 100.0f, 132.0f}}},
  {"Controlm22", {{55.0f, 200.0f, 134.0f, 30.0f, 100.0f, 132.0f}}},
  {"Controlm32", {{-30.0f, 300.0f, 134.0f, -55.0f, 200.0f, 132.0f}}},
  {"Controlm32", {{55.0f, 300.0f, 134.0f, 30.0f, 200.0f, 132.0f}}},
  {"Control12", {{-30.0f, 0.0f, 134.0f, -55.0f, -100.0f, 132.0f}}},
  {"Control12", {{55.0f, 0.0f, 134.0f, 30.0f, -100.0f, 132.0f}}},
  {"Control22", {{-30.0f, -100.0f, 134.0f, -55.0f, -200.0f, 132.0f}}},
  {"Control22", {{55.0f, -100.0f, 134.0f, 30.0f, -200.0f, 132.0f}}},
  {"Control32", {{-30.0f, -200.0f, 134.0f, -55.0f, -300.0f, 132.0f}}},
  {"Control32", {{55.0f, -200.0f, 134.0f, 30.0f, -300.0f, 132.0f}}} };



//the first float is the intensity, the second is frequency for cos. 0 means cos(0) = 1
//since a linear program is possible, I don't need to specify any more control
std::map<std::string, std::array<float, 2>> program{
    {"RF1", {{40.0f,20.0f}}},
    {"RFouter1", {{0.114995f,0.0f}}},//0.1197f
    {"Ground1", {{-4.2f,20.0f}}},
    {"Controlm11", {{0.0f,0.0f}}},
    {"Controlm21", {{0.0f,0.0f}}},
    {"Controlm31", {{-2.0f,0.0f}}},
    {"Control11", {{0.0f,0.0f}}},
    {"Control21", {{0.0f,0.0f}}},
    {"Control31", {{-2.0f,0.0f}}},
```