# An Overview of the Development of a Quantum Computer Simulator

**Jonathan Law - lj048**

**00611905**

Supervisors:

**Simon Scola**

**Alex Fedorec**

April 2014

Project URL: `http://stuweb.cms.gre.ac.uk/~lj048/QuantumSimulator/index.html`

Submitted in partial fulfillment of the University of Greenwich's

**BEng (Hons) Software Engineering**

**Word Count: 14,678**

**Abstract**

Since the conception of the Quantum Computer (QC), their potential for greater magnitudes in speed of processing calculations has been realised by the vast majority of interested parties. This has led to a recent push in research and development in creating a stable QC which can perform quantum algorithms.

To aid the creation of a stable and highly available quantum computer, simulators have been developed as a tool to as the thought processes. This report discusses details of a project that aims to investigate around the subject area, and produce an application to allow users to simulate quantum algorithms on a QC.

# 1 Acknowledgments

The owner of the project would like to take this opportunity to thank those who assisted throughout the duration of this project, and contributed to its success.

First, to project supervisors Simon Scola and Alex Fedorec who guided the project owner through the project's development. Second, to Kevin McManus who co-ordinated the programme and helped to structure the year of university. Finally, to friends and family, who supported the project owner personally, both within and outside of the project.

Also to Hugh Brace. *"Hugh were right."*

# Contents

# 2 List of Tables

## List of Tables

# 3 List of Figures

## List of Figures

# 4 Introduction

## 4.1 Background Information

Modern computers have been leveraged for their capabilities in terms of instruction processing, data storage, calculation speed and much more. Computers are capable of completing these tasks through the usage of bits, which are in a deterministic state of either 0 or 1.

Quantum computers make use of phenomena found in quantum mechanics such as superposition and entanglement, to allow qubits (quantum bits) to take a state of being simultaneously 0 and 1. Quantum computers use qubits to perform their calculations, which then returns results which may or may not be deterministic. Due to quantum computers being able to leverage the usage of qubits, they are capable of performing tasks such as integer factorization and execute algorithms with several orders of magnitudes increase in execution speed compared to classical algorithms.

It is for the previously stated reasons that quantum computers are highly desirable. However, one of the greatest challenges facing the field of quantum computing is quantum decoherence; superpositions of atoms are highly volatile and are susceptible to decohering at the slightest interaction with its environment. Decoherence of an atom is an irreversible action, and as a result needs to be highly controlled in a quantum computer's environment, lest it be incapable of maintaining qubits' and their states.

In WIRED's interview (2007) with David Deutsch, Deutsch was recorded stating that the next significant advancement in the field of quantum computing would be the pursued interest of simulating a quantum computer. Developing algorithms in software will help researchers to identify and conceptualise the requirements of the hardware. The project outlined in this report looks at the development of a quantum computer from conception to live roll out of the system.

## 4.2 Scope of the Project

The project aims to cover the development of the application from conception, requirements gathering, design, implementation, testing and deployment. This means providing relevant documentation at significant milestones, such as design of the system or testing documentation. The application itself should allow users to input data such as the number of qubits to simulate, their initial starting state and which quantum logic gates should act upon qubits in a desired order.

There are several methods or factors to be looked at when evaluating the success of the system. The developer will be looking at how accurately the system can simulate results from logic gates

and quantum algorithms, but also how quickly it can perform these algorithms. This is due to the fact it is only a simulation, not a real quantum system and thus should not be able to match the speeds predicted for a real quantum system. The expectation is that the simulation shall be many magnitudes slower than a real system, though the results should be accurate if the product is to be successful.

The system should be easy to use, and allow the user to quickly set up the circuit to be simulated. This is especially important in setting up large and complex systems to be simulated, as the number of logical objects to be managed can become very large. In addition to this, the results should be clear, and easy to understand, even for novice users.

A feature which may or may not be implemented due to time constraints will be the ability to store results, and save desired set ups of the system. This feature could be useful to researchers who would like to perform analytics on the results retrieved, or to go back to previous set ups to modify or review the set up. As such, this feature will be kept in the middle ranges in terms of priority for completion as it adds a high amount of conceivable value to the product, but is not critical in its usability or success.

Another constraint will be the inability for the user to create their own gate. While this may be a useful feature to the user, and add some future proofing to the system, it is again considered outside the scope of this system. This is due to the complexity involved, and a lack of vision for a successful, robust, and usable implementation of this feature.

The project will be deemed successful if it can accurately simulate quantum logic gates acting on qubits, and quantum algorithms. Another factor to rate the project's success will be the speed at which the circuit can be simulated. Greater speeds of evaluation will be considered more succesful than slower evaluation speeds.

## 4.3 Project Aims & Objectives

The aim of this project is to develop an application to allow users to simulate a quantum computer. This involves research into the field of quantum computing to further understand the field of quantum computing and quantum algorithms, how best to simulate them, and developing such an application from conception to deployment.

A summary of objectives to be completed can be seen as follows:

- Conduct research on literature to gather insights about the current state of the field and related fields.
- Evaluate technologies in order to base an informed decision about how best to implement the application.
- Critically evaluate existing products which achieve the same objectives in order to see how to improve the field.
- Create design documentation to implement the QC Simulator (covering both GUI design, and implementation of functionality).

- Test the application, documenting testing processes, results and changes made to the system.
- Deploy and evaluate the final product and management of the project, including choice of methodology, time scheduling, risk analysis, etc.

## 4.4 Methodology

Evolutionary prototyping allows for regular prototyping of the product, enabling feedback and increased changeability in the product. Feedback is useful when considering GUIs due to its nature of being hard to evaluate until it is developed. While design documentation will still be developed, it is hard to truly experience an interface until it is being used. Evolutionary prototyping allows the developer to quickly develop a UI, gather feedback and make suitable changes in a small time frame. It also suits the low manpower and timeliness available for the project.

The iterative nature of evolutionary prototyping also allows for changes to be made to the system during development with a greater degree of swiftness when compared to staged development methodologies such as the Waterfall methodology or the V-Model. This is an important factor in this project, as there is an exploratory element to this project and a small time frame to commit to. While the aim and objective of this project is clearly defined, the application to be developed in order to achieve such objectives have not been fully specified, with several elements to be decided upon later in development.

Unit testing will be useful as it allows for automated testing, and is suitable to test the product's accuracy, as the expected output will be the result of a mathematical function. However the availability and complexity of integrating unit testing into the project will depend on the programming language chosen to implement the system. Newer languages will not have had unit testing libraries built into them as the more popular and defined languages have, but it will be likely that there is a 3rd party unit testing library available, or failing that, being able to code in such a way that will return test results, though not strictly a unit test.

# 5 Literature Review

## 5.1 Potential of Quantum Computers

In 1994, an algorithm devised by Peter Shor, which has come to be known as Shor's Algorithm, inspired research into whether a quantum computer could be built. Shor's algorithm, a quantum algorithm which can only be performed on quantum systems, could efficiently factor natural numbers and compute discrete logarithms. Lack of an efficient algorithm to compute these has been the core of the security of public key encryption systems, notably RSA encryption, which is widely used around the world. Such an algorithm then, if successfully implemented, could break a vast majority of security protocols implemented in a reasonable time (Bacon and Van Dam, 2010).

Another algorithm which promises vast improvements in magnitudes of execution times when compared to that of its classical counterpart is Grover's Algorithm which has been described as a search algorithm. Though the example of a database is often given when discussing Grover's algorithm, it is important to understand that it actually identifies a target within an unstructured data set. Grover's algorithm has also been proven to be the best attainable improvement of performance possible by any quantum algorithm (Arikan, 2003).

While it is widely agreed algorithms such as Shor's Algorithm will allow for efficiency to the capabilities of factoring large numbers in polynomial time, Van Meter and Horsman (2013) warns that it is difficult to discuss the actual prospective performance of a system based purely on algorithms implemented alone. Other factors such as the physical and logical clock used in the system, the architecture of the system and the number of qubits available to the system to name a few.

Van Meter and Horsman go on to note that the architecture and hardware has a dramatic impact on the viability of a quantum computer. It could be the difference between a useful proof of concept, to an immediate threat to security systems world wide. Figure 1 illustrates the difference in time for Shor's algorithm to complete, depending on the architecture and clock speed used, also comparing to that of the classical number field sieve.

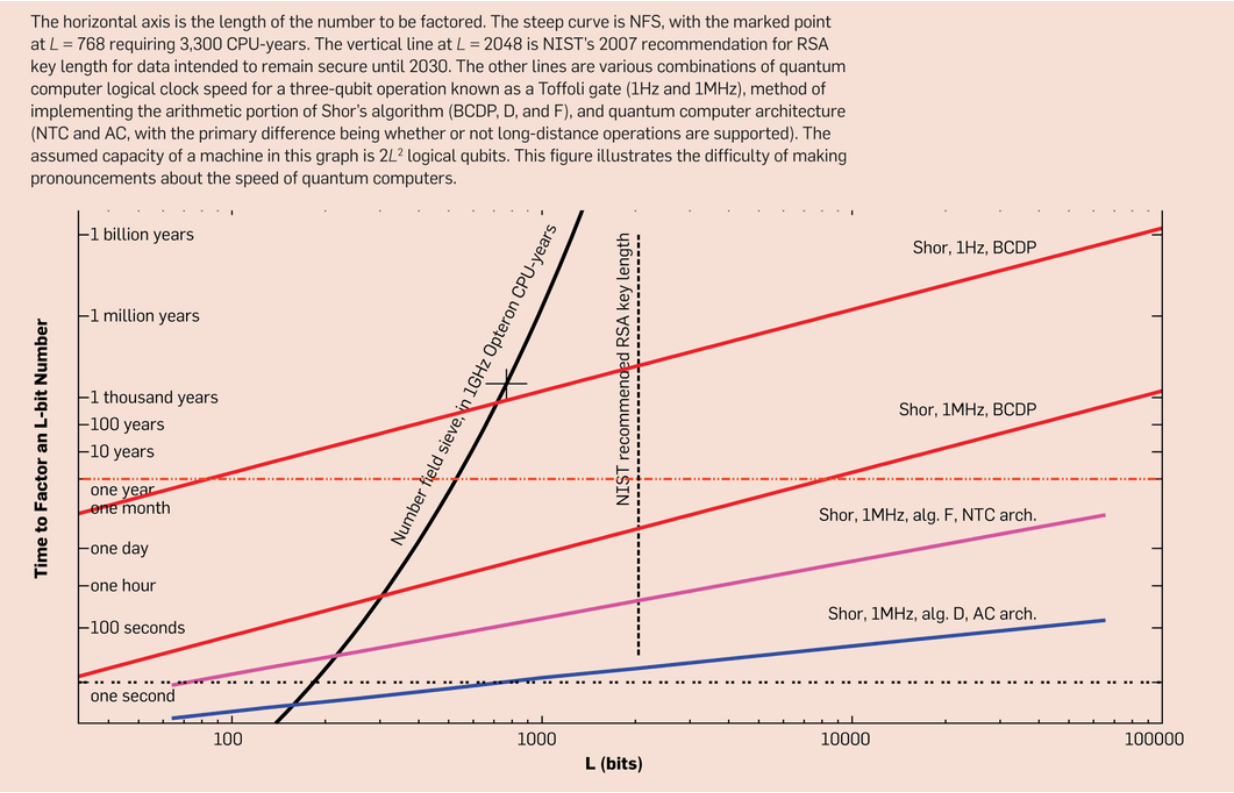


Figure 1: Comparison of Shor's algorithm run on different architectures and number field sieve (Source: Van Meter and Horsman, 2013)

On the other side of the coin of decryption, is quantum computer system's enhanced capabilities in terms of encryption. Development on systems which rely quantum cryptography which is said to be a potentially unhackable form of communication has been done by scientists. ID Quantique, a Swiss firm, has already used quantum cryptography in order to secure transactions from point-to-point within some institutes.

As previously mentioned, current cryptography relies on the difficulty in factorising integers to create encryption. Without the correct keys to decipher the encrypted message, the only other alternative is to attempt to decode the encryption, which takes a significant amount of time on classical systems.

Quantum computer have the ability to encrypt messages in a different way, using laws of physics found in the quantum world. The two computers on each end of the point-to-point communication will generate a shared but secret key which is as long as the message itself, as opposed to a set character limit in current implementations, will only be used once. First by the sender to encrypt the message, then by the recipient in order to decrypt the message. In the most common implementation, prepare and measure, the sender sends photons of light to the recipient. Photons can assume a number of different states to represent bits. The recipient then measures the received bit, translates it using the shared key, and measure the result with the sender.

The security in this system is to capture or measure photons will create a disturbance which can be detected almost immediately. Hence, there will be a discrepancy in the results between the two parties if there are any unauthorised members trying to intercept the photons being sent. If there are too many discrepancies in a given transaction in a message, the involved parties can assume the line is insecure and cease communication (Mone, 2013).

## 5.2 Current State of Field

While there is a lot of potential that comes with the usage of quantum computers, it is important to be able to construct a viable quantum computer, which can stably hold qubits for a reasonable length of time without decoherence. Different architectures must be explored so that their advantages and disadvantages can be evaluated, as well as processes or protocols. Once a live system has been rolled out, security flaws and performance leaks can be addressed, leading to a more viable machine.

While the potential for realising an efficient algorithm to factorise integers in a reasonable amount of time has been theorised, it has not seen much groundbreaking work executed in the real world. IBM reported that they had successfully implemented Shor's algorithm on a quantum system and had factorised 15 into its prime factors of 3 and 5 (IBM, 2001). While IBM's experiment had been criticised as its use of entanglement had not been demonstrated, other experiments have since taken place which do show evidence of entanglement (Lu, et al., 2007).

Other issues in implementing quantum computer systems include decoherence; A quantum computer must be able to complete all of its calculations and instructions before decoherence changes the state of the qubit irreversibly. Currently, the longest reported record of keeping a quantum memory state stable at room temperature has been reported at 39 minutes, extending to three hours at cryogenic temperatures. This record is claimed by an international team who worked at Simon Fraser University, Canada (BBC, 2013).

Issues in current existing implementations of quantum computers have been noted. In 2010, a group of hackers announced they had successfully hacked a quantum cryptography system (similar to the system detailed previously) by blinding the system with a large amount of bright lights. The group of hackers, who were working with the Swiss firm, ID Quantique, expresses beliefs that this is not a weakness with quantum cryptography as a concept, but rather the current state of implementation itself (Mone, 2013).

Many quantum algorithms have been theorised; All of which revolve around the idea of enhancing the efficiency of previous tasks by several orders of magnitude. Shor's algorithm can factorise numbers exponentially faster than the general number sieve, the quickest performing classical algorithm which achieves the same objective. Grover's algorithm runs quadratically faster than its classical pair, which is used to search through an unordered database. While these algorithms have been mathematically proven to be correct, and widely accepted to be improvement on its classical counterparts, more algorithms will be theorised as time goes on.

Previously mentioned, it was reported that Deutsch stated that a simulator for a quantum computer will be the next significant step in advancing the field (WIRED, 2007). Currently, there

are a wide variety of simulators which simulate the lower level circuitry level of the quantum computer. On this level it is possible to implement quantum algorithms, execute the simulation and retrieve an outcome.

Another level to simulate would be the particle level, in order to be able to simulate decoherence and entanglement of photons or implemented atomic level particle. While this would make the implementation of quantum computers simpler, as we would have the ability to implement an architecture and simulate its viability, it would require many algorithms that classical computers would take an exponential time to complete.

D-Wave are currently boasting the first commercially available quantum computer call the D-Wave Two system (D-Wave, 2014). The D-Wave Two system as advertised on their websites boasts factors such as 'exploits quantum mechanical effects' and 'enables quantum algorithms to solve very hard problems.' These quantum effects are achieved by chilling a lattice of 512 'tiny superconducting circuits' to absolute zero.

The first customer, Lockheed Martin, bought the first D-Wave system in 2011 and installed it in a new 'Quanutum Computation Center' at the University of Southern California. This machine was bought at roughly US$10 million (Nature, 2013). The second customer of a D-Wave system is a collaboration of customers between Google, NASA and the Universities Space Research Association. Together they have purchased and installed a system in NASA's Ames Research Center.

Despite the success enjoyed from large organisations, D-Wave were the victims of skepticism after the company's 2007 press event. This was due to their marketing approach with skeptics claiming D-Wave systems did not really leverage quantum mechanics at all. These problems arose from the lack of academic publishings being publicly available prior to their press releases, hence it was difficult to reinforce their claims. However, in 2011, D-Wave published evidence for the usage of quantum mechanics implemented in its 8-bit chip (Johnson et al, 2011). Other issues in assessing D-Wave's systems' performance involve the lack of measure which to compare it against. It is reported not even experts are completely sure how to judge it. John Martinis, a physicist at the University of California has been quoted: "[D-Wave] do these demonstrations, and how do you know if it's any more significant than factoring 15?" (Nature, 2013).

Despite all this, D-Wave has some strong backing and evidence to reinforce their claims of quantum computing being an improvement on its classical counterpart. Google has published papers indicating the D-Wave system excelled in terms of artificial intelligence and machine learning (Neven, et al, 2009). In 2012, Harvard University researchers claimed to be able to use a D-Wave machine to find the lowest-energy folding configuration for a protein with six amino acids. Although they did not have enough qubits to code the problem correctly, it is claimed at 13 correct results out of 10,000 runs is still a good turnover (Perdomo-Ortiz, et al., 2012).

Quantum programming languages have been implemented, though the hardware is not currently in place to support it (NewScientist, 2013). For example, Quipper is implemented on the basis that it runs on a gate model machine, as opposed the existing adiabatic models. These languages are currently executed on a simulation of a quantum computer. Developing software for quantum computers may reveal insights or knowledge which may help to shape or design the

physical implementation or the hardware architecture of the quantum computer itself.

## 5.3 Gate Model vs Adiabatic Model

The two leading models in regards to the architecture of quantum computers are the gate and adiabatic model. Both have advantages and disadvantages which should be taken into consideration before developing the simulator application in question.

The gate model is widely used and documented due to its ease of communication and use. The gate model is highly graphical with functionality being displayed as graphical gates, which represent a transformation matrix to be applied. They are placed onto qubits to be operated upon. Hence, quantum algorithms in the gate model can be documented in a quantum circuit which contains the qubits initial starting state, and the gates or functions to be operated on specific qubits in order to carry out the algorithm.

It also has strong foundations in its logic, as opposed to its hardware. Gate model algorithms have been theorised mainly with the intention of factorisation, teleportation or entanglement. To support this, it also has support from fully fledged theories of error corrections.

However, the gate model has weak hardware basis. Gate models are highly susceptible to decoherence as different energy levels are required to represent 0 and 1. This is problematic as users of a physical gate model must then take care to ensure the phase coherence between the two energy levels. The phase coherence must be maintained tightly in order for the system to operate.

For this reason, it is difficult to create elaborate gate model systems as the complexity of maintaining a stable state for all qubits involved becomes exponentially more difficult with each qubit introduced. With each new qubit there is the difficulty of maintaining the phase coherence between all qubits within the system.

D-Wave claim the adiabatic model that they have adopted in their systems have the potential to scale much more rapidly in terms of hardware, as well as being more applicable to industry. Colin Williams from D-Wave, explains gate model systems tend to focus on the stabilisation of qubits at a slow rate, giving the example of one extra perfect qubit per year. However, Williams claims that the number of qubits used within their D-Wave systems have been doubling each year, dwarfing the growth rate of gate model systems. He goes on to draw comparisons against this growth rate and Moore's Law which concerns itself with classical computers (D-Wave, 2013).

This is due to the lack of complexity involved with decoherence. 0 and 1 states are not encoded through use of energy levels in the adiabatic model. The system is entirely encoded as the ground state of the Hamiltonian and thus decoherence has a much lesser affect on the adiabatic model.

## 5.4 Conclusion

Comparing between gate model and adiabatic model, it would seem the gate model lends itself further to the purposes of simulation. The strong graphical basis of its notation allows for a means of simple input to the application without a strong background knowledge of the workings of the system required. This allows users with less academic knowledge in the field, to be able to construct a circuit using modular gates which can be chained together to create a system.

In addition, the gate model has strong theoretical support with named algorithms theorised for deployment and implementation on gate model quantum computers, whereas adiabatic algorithms can vary depending on their specific implementation, as well as the problem to be solved. As such, there is a greater community and academic gathering when discussing gate model circuits as opposed to adiabatic models.

Gate models are also simpler to expand or build upon, due to the modular nature of the logic gates in use. This allows ideas and concepts to be explained in the form of applying logic gates to a system. Each logic gate performs a function, which can be chained together to create a circuit. These logic gates are applied in order, transforming the overall system based on the gate's function, then finally returning a final state.

The downside of the gate model is its difficult implementation in hardware, due to quantum phenomena such as decoherence. As the product will be a simulated system on a classical computer, decoherence is not an issue in the simulation and will not be accounted for.

As a result of the findings from this research, for the purposes of the simulation application a gate model quantum system shall be used as the model of choice.

# 6 Technical Review

## 6.1 Programming Languages Considered

### 6.1.1 Java

Java brings many advantages, especially as it is the language which the developer believes he is most knowledgeable in, meaning there will be little to no learning required to implement the project. As with most modern programming languages, Java has strong object oriented programming capabilities; a programming style which the developer is comfortable in working with.

JUnit is a widely supported unit testing library available for Java; so heavily supported it is often bundled with IDEs natively. If not bundled natively, the installation process is often managed by the IDE through native installation tools.

Java also has the support of large corporations such as Google and IBM, and strong IDEs which have free of charge options such as Eclipse. Java is currently very widely used and does not look to be losing support soon, hence the risk of the language becoming obsolete and unused is not likely in the near future. Java is platform free, which adds to its popularity and wide usage. Once Java is installed on a machine, it will be capable of executing Java programs, independent of the platform.

In terms of deployment, it is possible to compile the application into an executable JAR file which users can obtain and execute locally. This has implications on the method of distribution though. If developer wishes for the application to be widely available, considerations may involve uploading an install package to a file hosting service or through physical storage means such as removable storage devices.

Alternatively, the application can be written as an applet, which means it can be embedded as part of a web page. However, the two are not interchangeable meaning it is not possible to write a Java application as a desktop application, then run it as an applet later in its development. It must either be developed from the beginning for its intended use, or undergo code base changes to support its new deployment realisation.

Downsides to Java as the choice programming language involve Java's lack of support of strong mathematical functions which are the core of quantum computing, and the necessity to install Java's library before execution of Java programs is possible. On top of this, Java must be updated if the target application uses a higher version of Java than installed on the client's machine. This again increases the amount of friction required for its use. Java's lack of strong mathematical functions can be supplemented through the use of libraries.

The requirement of a Java installation cannot be mitigated and adds friction to the system's usage. A user who does not have the correct version of Java installed will be notified, and the user will be prompted to upgrade before usage of the program is possible.

### 6.1.2 JavaScript

JavaScript is supported by widely used web browsers such as Google Chrome, Mozilla Firefox or Safari and further extends to browsers aimed at smaller devices such as Android Browser and iOS Safari. This causes minimal friction on the user's side as there is no installation of the application required.

A web server will be required to host the web page and serve the client when requested. This can be the cause of downtime if the remote machine itself encounters problems such as hardware failure, or networking issues. There will need to be considerations into how to deploy the web server, which can be affected by the programming language of choice. The clients will also need to access the application through a URL which may add friction if the URL is vague, difficult to remember or difficult to input.

Users can configure their web browser to enable or disable JavaScript execution so it is possible a user may not wish to enable to JavaScript and block the execution of the application. This can be offset by displaying a message requesting the user to enable JavaScript. Most users will generally comply with this request if an explanation can be given as to why it is needed. However, the decision ultimately lies with the client.

JavaScript can be considered to be an extremely flexible language supporting object oriented, imperative and functional programming. Although, it lacks support of a widely popular IDE, to the point where debugging tools have been deferred to the responsibility of browsers for provision, such as Mozilla Firefox's Firebug extension or Google Chrome's native Developer Tools set.

In addition to this, a framework for unit testing has not been decisively selected by either the community or the developers, with QUnit looking to be the most influential, boasting claims that it is the unit test runner for jQuery followed by FireUnit, part of the FireBug extension.

While JavaScript does not natively support the use of matrices and complex numbers, it is possible to supplement this issue with the use of libraries. This simply requires a download and a call to import them. Alternatively, it is possible to import a library by referencing an online location where the library is hosted. While this eases the import of libraries, it is important to consider the affects of the location of the library being moved, removed or otherwise inaccessible. Downloading the library means there will always be a local version to be referenced.

### 6.1.3 MatLab

To create an application in MatLab, the developer must build function and script files which are chained together to create a usable application, optionally creating a GUI pane to aid

the users accessibility to these functions. The GUI pane is also ideal for displaying graphical representation of the data involved. MatLab uses a high-level language which is based on object oriented principles, much like Java or C#.

MatLab's core functionality revolves around its heavy focus on strong mathematical capabilities such as functions for linear algebra, statistics etc as well as built-in graphics allowing for easy representation of graphical data. Features such as these makes it ideal for the application to be produced which relies heavily on the usage of matrices and complex numbers without the use of external libraries to extend the capabilities of this language.

It is also capable of producing unit tests similar to other languages, in that you create a set of tests which aim to test the output of functions when given certain inputs. MatLab makes use of its own native library set to handle unit testing as opposed to other languages which defer to 3rd party libraries.

Break points for testing and debugging purposes are enabled in MatLab. Debugging is also supported through MatLab's IDE with the usage of setting break points then being able to manually step through the code to examine the values of variables and the functions being performed within the application. This allows the developer to understand what the current state of the application is, and help to identify why it may be behaving in an unintended manner.

However, accessibility will be considerably more limited than previously mentioned programming languages. MatLab applications cannot be hosted on a web page and so must be downloaded and executed locally on the user's own environment. In addition, MatLab applications cannot be executed without MatLab installed which cannot be attained freely. This causes a high amount of friction when considering usage.

### 6.1.4 Conclusion

JavaScript will be used as the language of choice, as it provides high levels of availability and the lowest user end friction for its golden path of usage. In addition, a web server will be supplied by the University of Greenwich to host and serve end users.

While a Java applet could be delivered in a similar process, the user will still require to have the right level of Java installed, which can cause high amounts of friction upon set up. Especially as some users could view this as an unforeseen friction which can unsettle users.

Although MatLab provides some strong mathematical basis as a programming language, this weakness of JavaScript can be supplemented with the use of external libraries which will also reduce the amount of implementation required of the developer. The high amount of friction provided by MatLab cannot be ignored, and the developer's personal skill level in MatLab is low which is a contributing factor to its rejection as a choice of language.

## 6.2 Development Software

### 6.2.1 EditPlus

EditPlus is a text editor which offers FTP/SFTP capabilities, allowing changes to be made directly on the server. This removes the arduous cycle of working locally, uploading and testing by replacing it with the cycle of modify and test. This speeds up development and reduces the likelihood of errors. It also offers colour coding of code for ease of view and supports indentation for readability.

However, as a text editor it does not provide the wholeness of functionality fuller IDEs often provide such as refactoring of code, auto-completions, and preview of code.

### 6.2.2 Aptana Studio

As a heavier weight IDE, Aptana development works by creating a local project area (which can be cloned from a web server's directory) which can then synchronize with a web server through an FTP/SFTP connection. As an IDE, it also supports refactoring, auto-completion with suggestions and preview of code.

Upon quick evaluation, the synchronisation is only supported with text based files, multimedia content appears to require a manual push to server. In addition, an error occurred during cloning of the University of Greenwich's server, reporting a stack overflow error in its attempt to do so.

### 6.2.3 Cloud 9

Cloud 9 offers similar functionality to Aptana in that it too is an IDE aimed at web development. Being a web application, without the need to locally install files is its main advantage over other software. It works directly from the FTP/SFTP server with the requirement of creating a local area, although this does not mean it is copying the files to its own servers.

Similar to Aptana however, it reports an unknown error in its attempt to clone the University of Greenwich's web server. Once past the initial error, it appears the workspace has successfully cloned the server's file system and offers a fully functional FTP/SFTP connection to work directly on the files.

In addition, it offers support to GitHub and Bitbucket which may be used as a repository for the usage of backing up files, versioning and managing change sets.

### 6.2.4 Conclusion

To conclude, Cloud 9 will be the software of choice to drive the implementation of the web application. Reasons for this decision include the functionality offered by an IDE such as colour

coded code syntax, auto-completions and suggestions, as well as its FTP/SFTP connection capability. In addition, as a web application, it offers high levels of portability with low friction set up from machine to machine.

## 6.3 Back Up Method

### 6.3.1 Dropbox

Dropbox can be interfaced through a web application or a locally installed application on an end user machine. Looking at the locally installed application, Dropbox offers cloud based storage which will synchronise the file system in a given directory on the local system, to the cloud, with either a single user, or multiple white listed users. Navigating to the web application will allow users to view a change list showing the time and user who committed the last change.

Dropbox offers a minimum 2GB of storage free of charge, with the capability to upgrade to a subscription service allowing for more space and more users (Dropbox, 2014). However, change sets are not leveraged and so versioning can become difficult across a range of files.

As part of these findings, it is concluded Dropbox will be useful to back up the written essay and documentation surrounding the project, but will not be relied on to control the application itself.

### 6.3.2 University of Greenwich Servers

As the developer is a student of the University of Greenwich, a system to store files, and a web server to serve files to clients is available for usage. Students of the CMS department are allocated 250MB on the Unix system, which can be accessed via FTP/SFTP, as well as access through remote terminal or virtualisation.

It is stated the servers are under maintenance from 7am - 9am on Tuesdays (University of Greenwich, 2013). It is further stated this is when maintenance and essential updates are performed, and this is assumed to extend to include back up of current file storage.

In terms of file management, while the servers do not offer versioning, change sets or builds, it is a service that requires no additional cost in resources for usage, and provides a web server which will allow the developer to serve the produced application to the web. In addition, the University of Greenwich will be liable for any downtime or data loss within their system. This presents a transfer of risk, lowering the risk for the developer of the project. As such, the system will be employed for usage throughout the project.

## 6.4 HTML5 Canvas vs SVG

HTML5 Canvas and SVG are technologies which handle the drawing of dynamic graphical elements on web pages, often through the use of JavaScript. Canvas and SVG handle their

drawing in different methods which brings about different advantages and disadvantages in their usage. This is an important factor in their success; as both are powerful but better suited to different applications as neither technology is superior all around.

HTML5 describes Canvas as a web page embedded element which "provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, art, or other visual images on the fly" (W3C, 2014). This description retains the essence of enabling the capabilities of creating graphical elements dynamically.

Typical implementation of HTML5 canvas element is to create a HTML5 canvas element and through JavaScript it is possible to use the Canvas API to manipulate pixels within the canvas itself. The API has been developed to support utility functions such as drawing lines, shapes, rendering text as well as rendering images onto the canvas. In addition, it is possible to add event listeners to the canvas which can then call functions to be performed. The graphical representation is abstracted away under the canvas element, which holds the event listener. The developer must then implement a way of translating the event which occurred on the canvas, to the user's intention.

SVG describes SVG (1.1 Second Edition) as "a language for describing two-dimensional graphics in XML [XML10]" (W3C, 2011). SVG allows creation of vector graphic shapes, images and text which can be animated. As SVG elements are described through XML, it is possible for browsers to parse the XML, then create objects embedded in the DOM to display the graphical element. Otherwise, it is possible to save the image in an image editing software as an SVG file which can be embedded into the web page using the `<img>` tag. As the graphic becomes embedded into the mark up itself, it retains robust capabilities available to other elements. This includes attaching an ID attribute to reference it through the DOM, applying CSS styles to elements, and adding event listeners.

Both Canvas and SVG are widely compatible with a range of browsers, supported by popular desktop browsers like Firefox and Chrome, as well as browsers designed for physically smaller devices such as iOS Safari and Android Browser. However, SVG has made further optimisations for mobile devices. The W3C recommendation for SVG Mobile is to use a version of SVG called SVG Tiny (1.2). Canvas has not made differentiating versions intended for optimisations on other devices.

As SVG uses XML in order to describe a set of vectors to create an image or a graphic, SVG elements are highly scalable. They can be resized with little loss of quality, as the vectors are not coupled to the positioning of pixels themselves. Instead it draws the image from scratch on each resize. Canvas relies on manipulating pixels directly, which means on a resize, it has high loss of quality due to the attempt in positioning pixels relative to one another, without accounting for space generated or lost in a resize.

With the gained advantage of scaling well with resized dimensions, comes loss of efficiency in terms of resource usage. Using XML to describe an image, which is then embedded into the DOM for future manipulation may result in slower rendering of images when comparing to Canvas' method of manipulating pixels directly. This cost is further exacerbated when a large portion of the elements must be redrawn. This is due to SVG's need to modify the XML

structure, and reapply it to the DOM before being rendered, whereas Canvas can begin rendering almost immediately.

This loss of efficiency may be offset by the amount that must be rendered on an update. As SVG elements can act independently of one another, updating a single SVG element in a large set of elements may be quick to process. In the case of Canvas, the whole canvas is treated as a single image, and to update any part of this image, requires an entire redraw of the canvas element. If the graphic to be updated can be isolated and identified it may be more prudent to use SVG. In the case where it is expected a large amount of the image will be updated on a very frequent basis, Canvas will be better suited to perform the large amount of quick updates.

Looking at Canvas, as the contents of the image itself is abstracted away from the DOM, any accessibility text or meta-data can only be bound to the canvas element itself, rather than its contents. In addition, the canvas element itself does not specify any additional attributes to support accessibility such as title or descriptions. While the use of a `<noscript>` tag may aid in the cases where JavaScript is not enabled on the client's browser, it does not aid in cases of alternative usage such as those who are hard of hearing or sight.

After reviewing the advantages of both technologies, Canvas and SVG, it is decided SVG will be used to approach the graphical concerns of development. The ease of binding events to graphical objects plays the largest factor in its choice, with the lack of large amounts of frequent updates also being a consideration in its choice. From here, it may be useful to begin research into libraries which aim to ease the implementation of SVG with the ideals of simplifying the implementation as much as possible.

# 7 Legal, Ethical, Social, and Professional Issues

The application to be produced does not currently have any design or implementation decisions which requirement capture of user's personal data. This alleviates many issues such as conforming to the Data Protection Act 1998, which touches upon many aspects such as security, allowing user control over data, feedback systems, and relevancy and accuracy of data obtained.

Though copyright would be a bigger concern if the programming language of choice would be JavaScript, it is still of concern in the usage of the other mentioned languages. To prevent others from claiming the application as their own, an initial and low effort mitigation would be to add a visual copyright disclaimer to the product which would state the ownership of the application. This would protect the rightful owner from legal contact and could be used against a party accused of theft.

Professionally, the results from the simulator should be accurate, which relies on accurate values being obtained and used during development. This means retrieving details about mathematics at the core of quantum computing from a credible and reliable source, in both the functionality of the gates and the application of the transformation.

Additionally, it would be prudent to ensure the application has an appropriate amount of stability. A highly unstable application may frustrate users, and perhaps interfere with the usage of other applications. For example, if it were deployed as a web application and it were to critically fail in the browser, it may affect other tabs or perhaps the whole browser in some cases.

Allowing users to input data for usage will have both security and ethical implications. There will need to be considerations into validation of the data to ensure it is not malicious to the program, as well as considerations into the suitability of the data to be entered. To take the example of a text string which may be taken from a user as input, then later displayed in the application, there will need to be considerations on how to avoid explicit or offensive material being displayed to other users. This may be an automated or manual process.

If a means of contact is available through the application, through an implemented process or visible contact details, then a process should be declared and visible to the user. This should address what details are needed, what they are needed for, in addition to what details will be stored. This touches upon the Data Protection Act, and as such the developer will attempt to avoid the usage of personal data where possible.

While outside of the scope of this project, if bugs will be fixed, then a controlled method of deploying fix patches will be necessary. This means keeping a backlog previous versions, should rolling back be necessary (as well as keeping them available for download) as well as keeping a maintained list of changes in each patch so the user understands what has changed.

# 8 Existing Product Review

## 8.1 Similar Existing Products

The main purpose of the application to be developed is to allow the user to build and simulate quantum circuits; Outside of this, any features are supportive and while may add value, is not a critical success factor. As the method of delivery or implementation method has not been finalised, it would also be useful to look at varying implementations of applications which allow for the simulation of quantum circuitry.

The applications to be evaluated are as follows:

- jQuantum (jQuantum, 2010)
- Quantum Circuit Simulator (Wybiral, 2013)
- Zeno (Federal University of Campina Grande)

It may also be wise to evaluate quantum programming languages in their usage. It is important to understand quantum programming languages are in fact, not programming languages designed for execution on a quantum machine. Instead, they are languages which simulate common quantum functions into a library. It is also important to understand that these languages are often used for exploratory and experimental purposes as opposed to the intent of creating a functional program.

The quantum programming languages to be evaluated are as follows:

- QCL (Ömer, 2014)
- QML (University of Nottingham, 2014)

## 8.2 Evaluation Method

In evaluating existing products, using Likert Scale as an evaluation technique will provide a quantitative result against factors of the evaluators choosing. Therefore, factors to be evaluated will derive from a subset of ISO/IEC 9126, as it is believed not all items are relevant to this type of application. The result can be analysed mathematically, hence we can provide an overall average for each product, based on its scoring on each category.

In addition, features identified in each product, which the evaluator believes adds high value will be noted and aggregated for consideration for implementation in the application to be developed. In the same vein, features which are not present in products which would increase its effectiveness will be noted for the same reason.

The ISO/IEC 9126 factors to use in the evaluation of the applications are as follows:

- Functionality
  - Suitability
  - Accuracy
- Usability
  - Learnability
  - Attractiveness
- Efficiency
  - Time Behaviour

When evaluating the quantum programming languages are as follows:

- Functionality
  - Suitability
  - Accuracy
- Usability
  - Learnability
  - Understandability

## 8.3 Evaluation of Existing Products

### 8.3.1 jQuantum

This product is a Java application, and although alternative methods of accessibility were provided such as applets to enable in browser usage, these alternatives appear to be non-functional and unsupported. As such, a download of the application was a necessity.

The interface is somewhat cluttered and difficult to understand, with little in the way of prompts or feedback to aid the user in learning the application. The user must first initialise the qubits within both the x and y registers, which then updates graphical interface with a visual representation of the initialised qubits. In order to change the starting states of the qubits, or add gates to the circuit, the user must click a button with the appropriate control which brings up a text form to fill in. For example, to place a hadamard gate on qubit 4 of the x register, the user must click the hadamard gate and enter '4' into the x field.

To measure the output of the circuit, the user can move along the gate sequentially using the arrows to move step by step through each row of gates. The results display themselves as squares which are colour coded depending on the value of the complex number z, which can be mapped on a colour map of a complex plane.

This process is lengthy, accident prone and unintuitive to the user as it abstracts away the direct control of the circuit from the user, as well as providing difficult to understand controls and feedback.

| | | |
|---|---|---|
| Functionality | Suitability | 3.0 |
| | Accuracy | 5.0 |
| Usability | Learnability | 2.0 |
| | Attractiveness | 2.0 |
| Efficiency | Time Behaviour | 2.0 |
| | **Overall** | **2.8** |

Figure 2: Likert Scale results for jQuantum

### 8.3.2 Quantum Circuit Simulator (Wybiral)

This product is a JavaScript application, hence installation was not a necessary phase. It provides an easy to learn and use UI, with quantum gates marked by a symbol, and additional information on hover. The menu abstracts less used functionality such as importing and exporting of existing circuit set ups, or modifying the current set up. Draw backs to this application is the limited number of gates you can apply in a single circuit, as well as the limited number of qubits (only 9 per circuit).

| | | |
|---|---|---|
| Functionality | Suitability | 3.0 |
| | Accuracy | 5.0 |
| Usability | Learnability | 4.0 |
| | Attractiveness | 4.0 |
| Efficiency | Time Behaviour | 5.0 |
| | **Overall** | **4.2** |

Figure 3: Likert Scale results for Quantum Circuit Simulator (Wybiral)

### 8.3.3 Zeno (Federal University of Campina Grande)

Implemented with Java, it does require an installation but the application itself is easy to use and feature rich, similar to Wybiral's Quantum Circuit Simulator. However, much of the commmonly used controls such as selecting quantum gates are hidden away which adds extra time in changing and setting up a circuit. In addition, a gate is placed one at a time, instead of multiple gates per selection. To its advantage, it allows for an unlimited number of qubits and spaces for quantum gates to be allocated, though these settings must be pre-specified and the user must begin again if the settings are incorrect.

| | | |
|---|---|---|
| Functionality | Suitability | 5.0 |
| | Accuracy | 5.0 |
| Usability | Learnability | 4.0 |
| | Attractiveness | 3.0 |
| Efficiency | Time Behaviour | 5.0 |
| | **Overall** | **4.4** |

Figure 4: Likert Scale results for Zeno (Federal University of Campina Grande)

### 8.3.4 QCL

As a programming language, it is considerably more difficult to learn and use than a produced application. In addition, there is little in the way of support and documentation to aid the user in the usage of the language. The evaluator of the product struggled to produce anything of value with QCL.

It is possible once the user is learned, they may be capable of producing compiled algorithms of great value. The inclusion of programmable functions allows for flexible and reusble parts of code. In addition, QCL allows for the inspection quantum machine during the execution of the program.

| | | |
|---|---|---|
| Functionality | Suitability | 4.0 |
| | Accuracy | 5.0 |
| Usability | Learnability | 1.0 |
| | Understandability | 1.0 |
| | **Overall** | **2.75** |

Figure 5: Likert Scale results for QCL

### 8.3.5 QML

Like QCL, QML is a compiler for the programming lanugage Haskell, which creates high amount of friction in its usage as it becomes more difficult for a user to understand its usage. The low amount of documentation not only obscures the potential of the technology in its value and usage, but frustrated the evaluator to the point of aborting the attempt of the installation. As such, QML will ultimately not be considered in its evaluation, but the experience of the difficult install and minimal learnability of the technology will be drawn upon.

## 8.4 Conclusion

Accuracy played a highly important factor throughout all applications reviewed; understandably so as the application would be of little use if the results gathered from a simulation were inaccurate. While the other factors examined also affected its level of quality as a product, they

were less critical to the application on the whole. As such, other factors amongst the reviewed applications had a greater variety and range in its perceived ratings. From this, it is important to take home the necessity of developing an accurate application.

jQuantum is a powerful tool which provides a lot of feedback and has the potential to simulate a large system. However, timeliness is an issue with this application, as the interface is unintuitive and difficult to view. This is further exacerbated by the difficulty of finding the intended tool through the cluttered interface. The evaluation of the circuit is difficult to understand with the colour coded system.

Wybiral's Quantum Circuit Simulator is extremely easy to use and learn. Commonly used tools were exposed at its lowest level. For example, all the quantum gates were available to select immediately. Less commonly used tools, such as import and export of data, were abstracted away into sub-menus. Finally, the process of setting up of a quantum circuit was streamlined in terms of time due to the ability to re-use the last tool, shortcut keys and changes in the circuits properties did not reset the state of the circuit completely.

The Zeno quantum circuit simulator is similar to Wybiral's in many aspects, but differed on few key aspects. Its lack of tool reuse made the set up of circuits less timely, and the reset of circuit state if any properties were to change. However, it had the potential to be a much more powerful simulator as it allowed the user to specify the number of qubits and spaces to allocate quantum gates to, whereas Wybiral's was limited in both of these regards.

Both quantum programming languages were difficult to install and use, due to the minimal documentation and support available to the user. The lack of a graphical interface also heightened the difficulty, and there is little in the way of user feedback and discovery of features.

Strong features to take into consideration during development will be the accuracy of the simulation, size of the simulation (in both qubits and quantum gates), the ease of the circuitry set up, and the abstraction of tools to appropriate levels. It will also be important to consider the difficulty encountered in the attempt of evaluating quantum programming languages.

# 9 Design Documentation

## 9.1 Statement of Requirements

Requirements the product aims to achieve, in terms of both functional and non-functional behaviours or constraints will be detailed here. Requirements have been prioritised and grouped using the MoSCoW method, categorising into Must have, Should have, Could have, Won't have.

### 9.1.1 Functional Requirements

**Must have:**

- Allow users to set up a quantum circuit, using quantum wires and quantum gates
- Allow users to alter initial state of qubits
- Evaluate a given quantum circuit, returning probabilistic values

**Should have:**

- Allow users to adjust properties of a circuit, such as number of quantum wires and steps to evaluate
- Display of time taken to perform evaluation

**Could have:**

- Export of current quantum circuit set up
- Import of a pre-defined quantum circuit set up

**Won't have:**

- Creation of custom quantum gate
- Bug report feature

### 9.1.2 Non-Functional Requirements

**Should have:**

- Declaration of external libraries used
- Hyperlinks to home page or reference page of external libraries used

- Prompts to feedback to user about the system
- Warn users of actions which are irreversable

## 9.2 Functional Design

### 9.2.1 Use Case Diagram

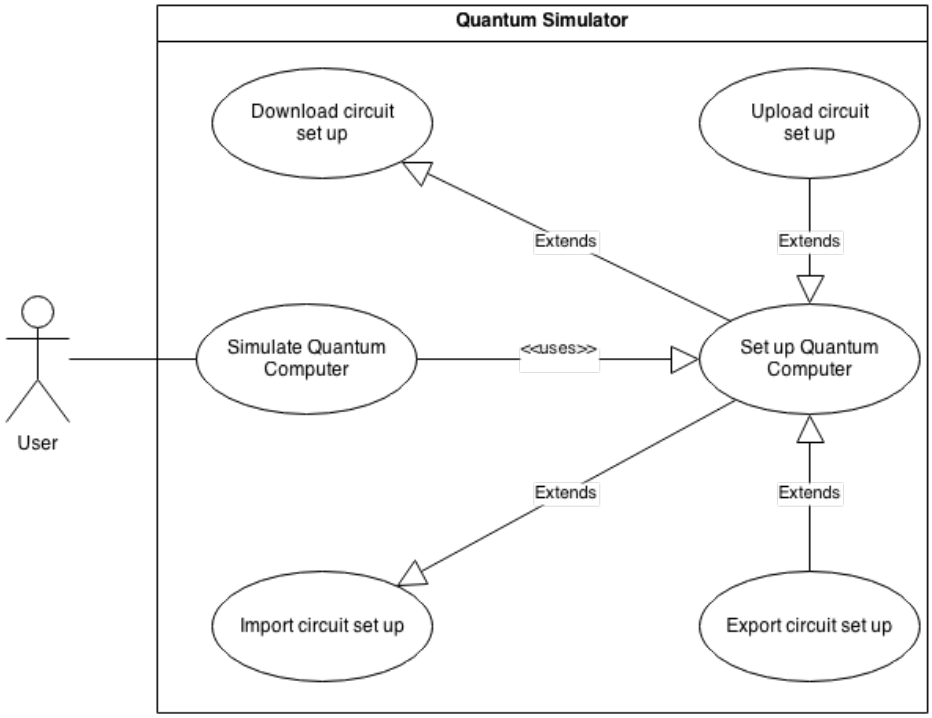


Figure 6: Use case diagram (version 2) for the proposed application

There is only one stereotype of user within this system named **User**, and it is expected the only real use case is to **Simulate QC (quantum computer)**, however doing this requires **Set up QC (quantum computer)** which the user may optionally choose to either **Import** or **Export** a set up. The user may also choose to **download** or **upload** their circuit during the usage of the application.

### 9.2.2 Class Diagram

Figure 7 shows the diagram which has been taken into the development phase and will guide the implementation. Earlier versions of class diagrams developed can be seen in the Appendices. This should allow the developer the quickly identify the objects required, their attributes and the expected functions from each object.

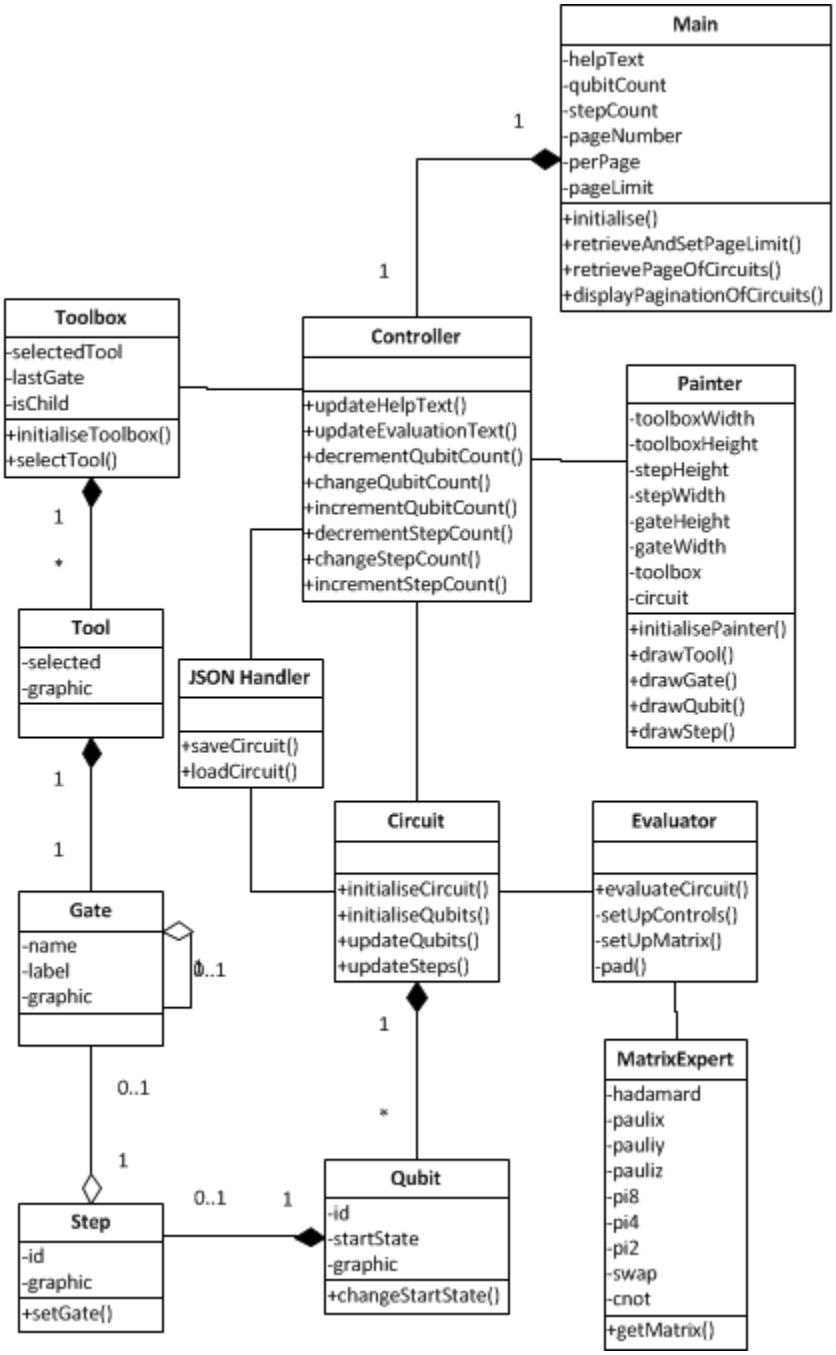


Figure 7: Class diagram (version 3) for the proposed application

This design attempts to create highly cohesive components which work together to create a working application. The user interacts either with the **Main** or the **Painter** class, depending on whether the DOM or the graphical circuit was accessed. Both classes must communicate through the **Controller** to access specific function calls such as evaluating the circuit, or adding a gate to a step.

A design which is highly cohesive should aid factorisation of the code base, maintainability and testability. In this example, it is possible to construct a circuit, modify the set up and add gates to the system, then finally evaluate the system without the need of coupling with a graphical interface. This allows us to create unit tests to test the accuracy of the evaluator.

## 9.3 User Interface Design

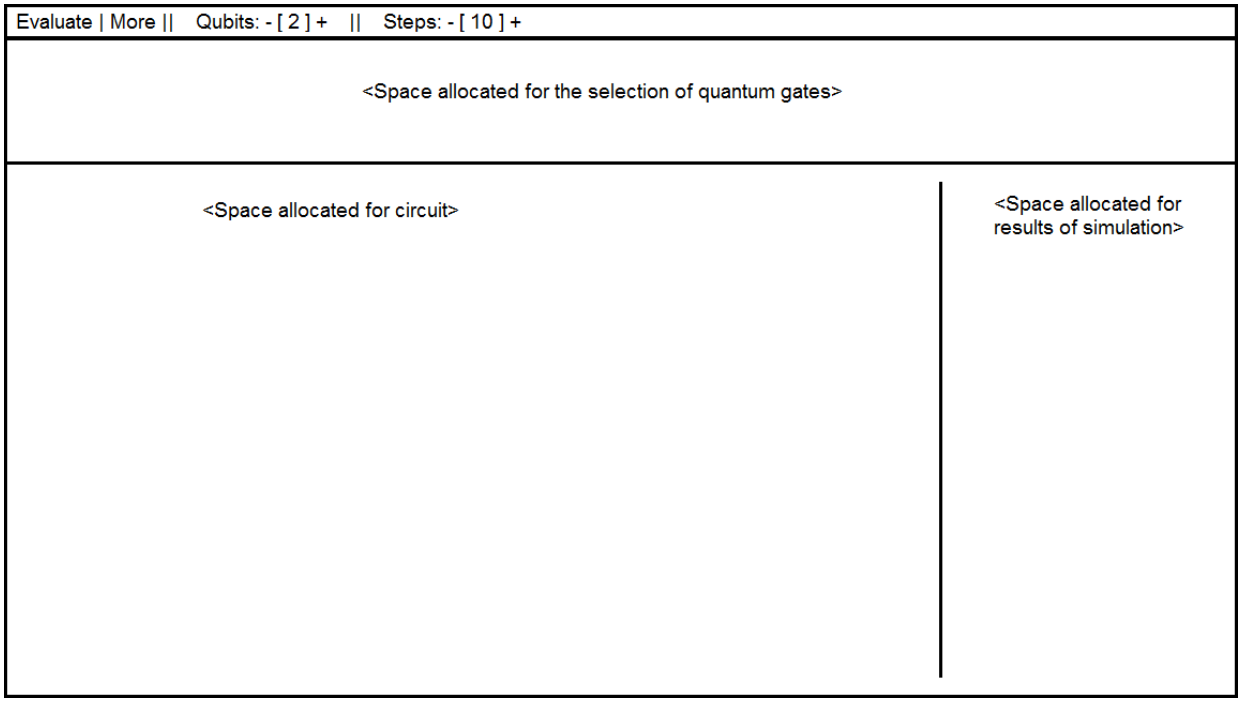


Figure 8: Layout design for the proposed application

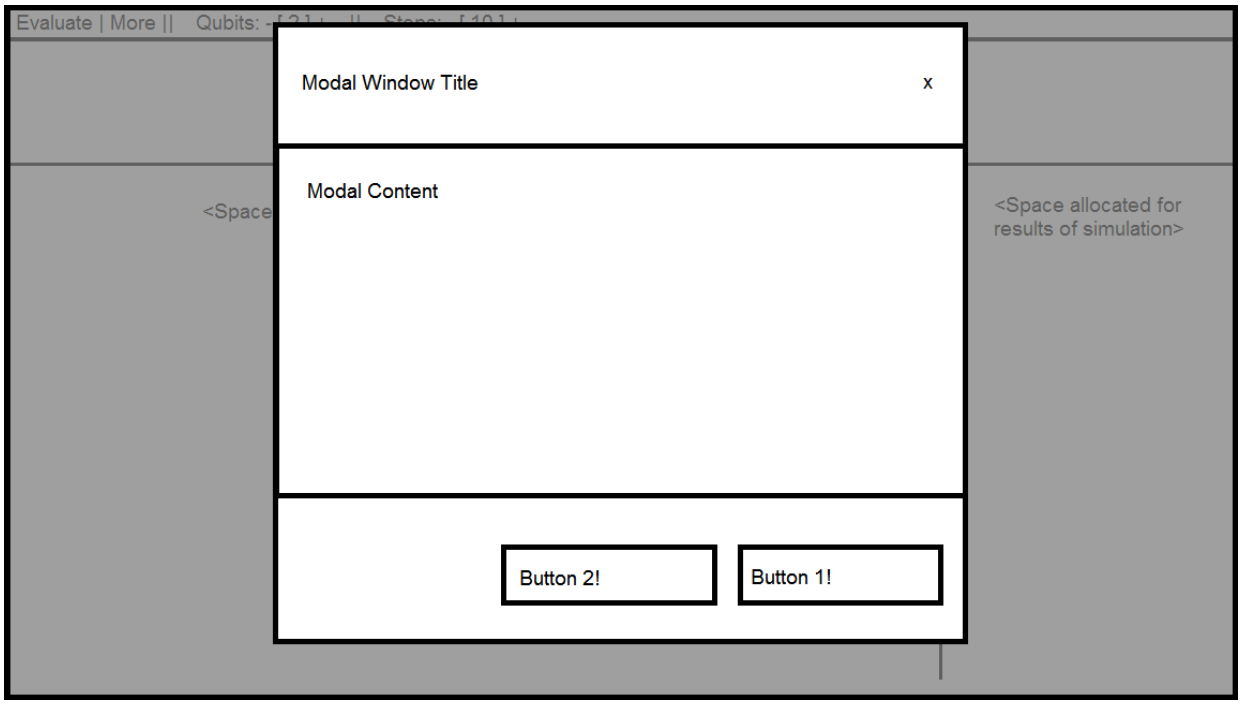


Figure 9: Layout design (with modal) for the proposed application

The user's main purpose of using this application will be to set up a quantum circuit, which will then be simulated and return results. The majority of the screen allocation should revolve mainly around the view on the quantum circuit, and enough space to comfortably understand the results returned. Other space allocated will be for the selection of the quantum gates which should be large enough so the user understands at all times which tool has been selected. Finally, there will be a menu bar which will give access to additional functions and adjustment of properties, but as these are expected to have a low frequency of use, space allocation will be less of a priority.

Usage of modal windows also aids in this abstraction of information, as it hides away views until the user requests them. This allows for features which require a large amount of space to display to be a part of the system without cluttering the main UI, by hiding and showing the view when it is needed.

The elements described have been incorporated into the layout design found in Figure 8. Should additional menu buttons or sub menus be required, they may be appended to the already existing menu bar without the need for redesign, as this design captures the essence of the layout.

# 10 Implementation Documentation

## 10.1 Prototype One

The first version dealt mainly with the user interface. This involved the usage of small amounts of HTML and CSS in order to layout and style the web page correctly, as well as using JavaScript to draw the graphical elements of the circuit and toolbox.

Initially, the developer used JavaScript to draw directly directly onto canvas elements embedded within the page. This worked well for the period of time as the developer had managed to graphically draw a toolbox and implemented functions which allowed users to select tools from the toolbox itself. This required the developer to capture the position of the mouse on the canvas on the click event, and transpose the co-ordinates of the click to the graphical element being seen. For example, to select the first gate in the toolbox, the click event would have been somewhere between 1-40 on the X-axis and 1-40 on the Y-axis which the developer would then translate into being the first tool in the row of tools.

While this was arduous, it worked for a time as it was simplified by the static nature of the toolbox itself. Converting co-ordinates to functions was simpler on the toolbox as the elements within the toolbox did not move. This meant a range could be specified, and a function call could be derived from this. The difficulty arose when implementing the circuit itself. Within the circuit, there were several differing elements that could be clicked on, such as the qubits themselves in order to change the initial state, the wire associated with the qubits to place a gate on, or a gate to replace with a new one. The dynamic nature of the circuit complicated the handling of the click events as it was difficult to understand what element was being clicked on based on the co-ordinates of the clicking.

This drew the developer to halt the implementation momentarily as the handling of graphics on a canvas was becoming complex, arduous and frustrating. This prompted some research into the technical aspects of graphical interface development, as it was certain there was a simpler approach to this issue. This research then took the developer into reading discussions between two leading technologies for displaying and manipulating dynamic graphical elements on a browser; HTML5 Canvas and SVG. After some research, the developer decided to proceed using SVG, for the benefits of being able to bind events to the elements themselves, abstracting the need for translating click co-ordinates.

This research also extended into looking at libraries which aimed to further abstract and simplify the workings of SVG creation. The result of this research was to use Raphael to handle the drawing. Working with Raphael did ease the drawing of the graphics on the developer's side but the developer still struggled to get to grips with the concepts behind working with graphics. For example, to begin with, the developer simply bound on-click events to every element within

a graphic. This included elements such as the outlines of gates, the background, the lettering, etc.

A problem arose when the user was unable to place a logic gate onto the circuit, due to the placement of another gate in its place. The source of this problem was due to the layering of graphical objects. When a gate was placed on top of a wire within the circuit, the gate took priority over the wire when clicked. Since the gate did not have a click function, the function to replace the gate was not called. One solution was to bind the gate's graphical elements with the same function as the wire in order for the function to be called on click.

This quickly became lengthy and repetitive. To remedy this, the developer stopped development for a period of time and looked to some online tutorials and examples, and consulting the documentation in order to discover the most effective way of dealing with binding functions to events concerning graphical elements. The results of this was to declare the function to be called on the event's occurrence separately, then to group up all similar elements together into a single logical object before binding a function to be called on the event occurrence to the group.

As a result of this, the circuit became much more flexible in its usage as users could click the area surrounding the wire to place a gate, even if a previous gate had been placed on top.

## 10.2 Prototype Two

With the completion of the graphical interface, allowing the user to set up a circuit to be simulated, prototype two oversaw the implementation and testing of the algorithms used to simulate the circuit generated by the user. This was not immediately successful as the first implementation of the algorithm did not consider correctly the whole range of the system to be simulated.

The first attempt at implementing the algorithm ran as such:

- Gather the qubits and set initial state as specified by user input
- Collect all the gates in the order of earliest execution to the latest
- Iterate through the list of gates, applying the matrix transformation on the specific qubit
- Gather the possible output states, derived from the number of qubits within the system (length of $2^n$ where n is the number of qubits)
- Iterate through the list of possible states, and multiply each qubit's respective amplitude to find the probability
- Display all possibilities with the probability of occurrence

This worked well for single qubit gates, however it did not correctly take into account gates which extended over more than one qubit. In addition, some gates were hard coded such as the Swap gate. This took the developer to research into existing algorithms. This took the developer to implementing an algorithm as outlined in Lee Spector's book Automatic Quantum

Computer Programming: A Genetic Programming Approach (2006). This algorithm took into consideration quantum logic gates which operated on multiple qubits. The algorithm is as follows:

- Begin timer
- Gather all possible states, derived from the number of qubits within the system (length of $2^n$ where n is the number of qubits)
- Assign probability of each state independently, derived from the initial set up of the system by the user
- Collect all the gates in the order of earliest execution to the latest
- Iterate through the list of gates, doing the following for each gate found (Lee Spector's Explicit Matrix Expansion):
  - Let *G* be the quantum gate's matrix representation of its applied transformation
  - Expand *G* to account for any attached controllers to the gate
  - Let *M* be the matrix of dimensions $2^n$ x $2^n$ where n is the number of qubits in the system to be simulated
  - Let *Q* be the set of qubits which the gate affects, and *Q'* be the gate which it does not
  - *M[i][j] = 0* where *i* and *j* differ in their binary representations, in any of the positions which apply to *Q'*
  - If they do not differ, concatenate bits from the binary representation of *i* in the positions which apply to *Q* to produce *i**. Do the same for *j**, then set *M[i][j] = G[i*][j*]*
  - Apply matrix *M* to the vectors of amplitude for each possible state
- Calculate all possibilities with the probability of occurrence
- Stop timer
- Display all probabilities along with time of execution

The implemented algorithm works well as it accounts for the whole system, rather than applying each gate to their applied qubits in isolation. However, it runs slowly as it requires the expansion of each gate when applying the transformation to the vectors of amplitudes. Future improvements may involve streamlining the algorithm to reduce the time taken to execute the algorithm, perhaps by storing previously expanded matrices, to prevent the repetition of the algorithm when the application is already aware of its expanded state.

## 10.3 Prototype Three

As a problem identified within prototype two during testing, the application becomes slow and unwieldy to use when working with moderate to large scale systems, while sometimes resulting in a complete failure of the application and requiring a restart, losing the current circuit setup the user had achieved. This becomes more apparent as more objects are added to the working system; this may mean qubits, length of wire or logic gates themselves. As a result, prototype three looks into performance issues and potential for optimisation within the code base, in order to make the application more usable, scale better with larger systems and less frustrating for the user to achieve their objectives through usage of the application.

This saw the execution of the first phase of performance tests. The results of the tests can be found in the Appendix section (tests ran at this phase have been numbered Test Run 1-1 to 1-3). The result set shows the functions (which were written by the developer, as opposed to native JavaScript functions or imported library functions) called which were allocated the largest amount of processing time in order to reach its evaluation of the simulation are as follows:

- updateCircuit - circuitry.js
- drawCircuit - circuitry.js
- draw - qubit.js
- setGate - step.js
- draw - step.js

Analysing the functions identified as lengthy in terms of processing time revealed these functions mainly involve creating or manipulating, and displaying SVG elements. To add to the processing time, the handling of the drawing has been revealed to be inefficient. This is due to an earlier implementation decision to simply redraw the whole circuit whenever a graphical element requires updating, or there has been an addition/removal from the circuit entirely. This was simpler in terms of logical processing (i.e. conceiving the algorithm to deal with drawing the circuit), as well as its implementation itself. However, it clear has had an adverse affect on the performance of the application, as it redraws elements which have had no changes. This is an unnecessary function call which wastes processing time.

One function call which does not deal with graphic management *only* is the `setGate()` function; while this does call for an update to the circuit drawing, it also changes the gate which the step is holding in its place. Looking at this, it also instantiates the logical objects to do with its algorithmic evaluation, such as creating a set of matrices to apply transformations to. This is lengthy as it means the creation of logical objects which are not immediately necessary, and at a worse case scenario, unused if the user replaces the gate.

The proposed solutions to improve the required processing in setting up a circuit are as follows will involve two major changes to the code base in general. The first of which is to refine the drawing algorithms of the circuit. Instead of simply calling a redraw on every update, it would be more efficient to pinpoint the elements that require updating, and update those in isolation. This should remove a large amount of redundant drawing functions. The second, is to refactor

the code, and separate the drawing from the simulation elements. This way it will be possible to defer the creation of components until they are necessary to complete a higher function, and will remove the risk of the object itself being redundant in the case that they are never used.

During the refactoring of the code base, it was also discovered a large amount of graphical elements were not being removed on the redraw function. As such, the frequent number of calls to the redraw function resulted in a large amount of graphical objects being alive at once, though behind the most updated graphical circuit. This used a significant portion of memory to maintain, and is likely to the largest contributor to the reduction in responsiveness and likelihood to critically fail after extended use. Addressing this issue relieved the application of poorly managed resources, and allowed the system to be more responsive.

Below compares the processing time used on these functions on prototype two and three. Note, the updateCircuit function and drawCircuit function is not compared as it no longer exists in prototype three. Additionally, we have drawQubit and drawStep functions, instead of draw functions which belong to respective objects. Instead, drawing functions have been refactored into a painter class.

| | Avg. Time (ms) | | |
|---|---|---|---|
| Function | Prototype Two | Prototype Three | Time Difference (%) |
| draw/drawQubit | 19,529.33 | 45.43 | 0.23 |
| setGate | 19,461.27 | 21.87 | 0.11 |
| draw/drawStep | 17,668.87 | 204.87 | 1.16 |

Figure 10: Comparison of processing time between Prototype Two and Three

## 10.4 Prototype Four

Prototype four involved work on the persistence of created circuit set ups. Work on this feature brought about the inclusion of allowing the user to upload to a communally shared database, as well as downloading works of the individual as well as others who have uploaded. JSON is used to convert the circuit set up to a textual representation for storage and transfer whereas MongoDB is used to host the data; MongoDB is suitable as a document database suitable for storing large text strings. jQuery has also been pulled in to help with the binding of events to web control elements, as well as aiding with the AJAX calls to send and retrieve data from the database.

To begin, the developer began implementing the functionality with very basic interface support. This involved the usage of JavaScript's native alerts and prompts in order to receive input from the user such as the desired name to give to a circuit. Once the functionality was developed, work moved towards providing an interface to streamline the user's usage of interacting with the persistence options. For this, Bootstrap was used as well as their example of modal windows to provide views which may be shown or hidden using web controls. This abstracts the functionality away from the main use case of simulating circuits. It was also realised this would be a good

way to declare external libraries used to create the system, without cluttering the interface, and so this was also implemented in this prototype.

The developed feature allows the user to save or load the system locally; this involves converting the circuit to a JSON string and the user entering a JSON string respectively. If opting to leverage the available database, the user must enter a circuit name to save the circuit, and upon retrieval the user may select from a list of circuits to load into the workspace.

The JSON string constructed is an array containing only the bare minimum information required to construct a circuit, which has been converted into JSON. This has the benefits of minimising the length of the JSON string generated, as well as avoiding the need to resolve any issues with the circuit object itself, such as unnecessary information and recursive structures. In Figure 11, a JSON representation of a circuit with two qubits, each with an initial state of zero, with a Hadamard gate placed in the upper most left corner of the circuit (first qubit, first step) is shown.

```
["00",["Hadamard",0,0,[]]]
```

Figure 11: Example of a JSON representation of a simple circuit.

```
>db.circuits.find(name: "Hadamard")
"_id" : ObjectId("534c1d607f8f95bf670009dd"), "name" : "Hadamard", "json" :
"[\"00\",[\"Hadamard\",0,0,[]]]"
```

Figure 12: MongoDB Record of "Hadamard" Circuit.

# 11 Testing

## 11.1 Manual Testing

Manual testing was used in the early prototypes when unit tests had not yet been implemented, and is still used to test cases which have not yet had unit tests created for them, or cannot be or is hard to test through unit tests. An example of something which cannot be unit tested is if the correct graphic is drawn when a gate is placed. This must be verified by a human. Things which are difficult to unit test is the check if the pagination is working correctly for the downloadable circuit listing, as the listing on the page may change. A manual testing plan and results can be found in the appendix.

Further improvements to be made in the testing suite is to migrate tests which can be unit tested, for example the conversion of a circuit to a JSON string, or the opposite which is to construct a circuit from the JSON string.

## 11.2 Unit Testing

Unit testing has been leveraged in this project to ensure the results obtained from evaluating the simulated system is accurate. To perform unit testing, QUnit was used to implement the unit test suite. The first iteration of test suites began from adapting an example version which served to ease developers into unit testing with JavaScript, by implementing a test suite which used only JavaScript functions and the JavaScript console as an output log.

This was a gentle introduction as well as a powerful tool, as it allowed the developer to check if any modifications to the code base had affected the results of the evaluator on a wide range of inputs each time the application was executed. Hence, errors could be noticed immediately, and corrections be made much swifter comparing to the exclusion of unit testing.

| | |
|---|---|
| Beginning unit tests... | unit-test-runner.js:1 |
| Of 23 tests, 0 failed, 23 passed. | unit-test-runner.js:31 |

Figure 13: Passing output log from version one of unit test suite

## 11.3 Performance Testing

Performance testing was seen as a necessity in this project, due to the wasteful management of resources seen in Prototype Two. To obtain a set of consistent and semantic results from performance testing, the same machine will be used in each iteration of test runs. This helps to

control the results, leaving the performance improvements to be the main reason for the change in results. For the same reason, the same browser will be used under the same conditions. For completeness, information on the machine and browser to be used are as follows:

| Test Machine | |
|---|---|
| Operating System | Windows 7 Professional 64-bit (6.1, Build 7601) |
| Processor | Intel Core i7 CPU - 2.8GHz (8 CPUs) |
| Memory | 4096MB RAM |
| Video Card | Nvidia GeForce GTX 460 - Approx. 2748MB Memory |

Figure 14: Hardware listing of test machine to be used to test application performance

| Web Browser | |
|---|---|
| Name | Google Chrome |
| Version | 33.0.1750.154 m |
| Extensions | ActiveX for Chrome - 1.5.0.7<br>AdBlock - 2.6.18<br>Google Docs - 0.5<br>Hola Better Internet - 1.3.31<br>JavaScript Errors Notifier - 2.1.5<br>Web Developer - 0.4.5 |

Figure 15: Detail listing of browser to be used to test application performance

Obtaining tangible data as a result of any test performed is important, as it allows the tester concrete data to base their findings and analysis upon. With a strong foundation of these findings, it simplifies the resulting action of constructing an action plan to follow the results, fixing any problems identified during the process. Without tangible data, the interpretation of the results can be much looser, meaning individuals may draw different findings from the same result set, and thus react differently.

To obtain this data, the testing process will make use of Google Chrome's natively built Developer Tools (DevTools). The tool to be used from the Developer Tools will be the JavaScript CPU Profiler, allowing recording of the CPU profile, breaking the processing time into the specific JavaScript function being processed. This will provide a break down of the proportion of time spent performing JavaScript functions, even if they are called multiple times. It will allow the tester to identify which functions are the most time costly in normal application usage, as a quick function may be called frequently, whereas a slower function may only be called once.

The circuit shown in Figure 16 shows the circuit to be constructed while the CPU profiler is recording. Previous attempts to formalise a larger system for testing have caused the program to be unstable and likely to critically fail beyond this point. It would be prudent to formalise the method of construction; the steps devised are as follows:

- Begin the CPU Profiler

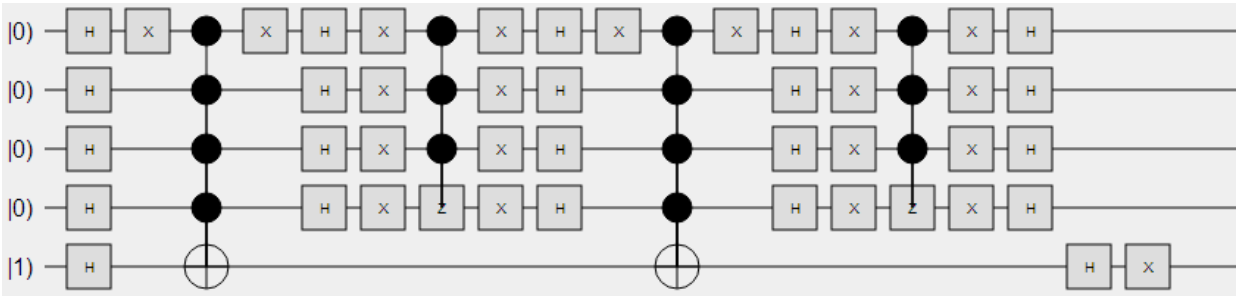


Figure 16: Performance Testing test circuit

- Increment the qubit counter to *5* using the buttons (instead of manual typing)
- Increment the step counter to *20* using the buttons (instead of manual typing)
- Change the bottom most qubit initial state to *1*
- From left to right, top to bottom - Add gates to the circuit until complete
- Evaluate the circuit
- Stop the CPU Profiler

# 12 Evaluation of Product

As mentioned in the introduction of this section, the product shall be evaluated against a subset of ISO/IEC 9126 to analyse quality. The subset of ISO/IEC 9126 factors to be used in evaluating the quality of the product are as follows:

- Functionality
  - Suitability
  - Accuracy
- Reliability
  - Fault Tolerance
- Usabillity
  - Learnability
  - Operability
  - Attractiveness
- Efficiency
  - Time Behaviour
- Maintainability
  - Testability

The product developed delivers perhaps the most important functionality required of a simulator which is to evaluate a user generated quantum circuit. The user may use the controls to specify properties of the circuit such as the number of qubits required and the possible number of steps to simulate. Manipulation of the circuit itself is possible by changing the start state of qubits as well as chaining together logic gates. Once the user is satisfied with the set up of the circuit, an evaluation allows the user to view results derived from the set up.

Suitability can be further improved in this product by adding additional features which may prove useful to the user such as exporting the results of a simulation to other formats allowing for simpler embedding in other environments. Another feature may be to export a visual representation of the circuit, such as a PNG format for similar reasons; embedding in other environments.

Although unit testing has allowed the developer to ensure the accuracy of the simulation is consistent, it requires accurate values as a comparator to be used. Values to be used for comparison were obtained through a combination of manual calculation and academic resources

(such as books and videos). The combination of both unit testing and accurate values to be used for the purposes of comparison has provided a basis of confidence when determining the accuracy of the results provided by the simulator.

The system developed has a considerable amount of fault tolerance, part of which is a result of having chosen JavaScript as the language of implementation. JavaScript has a considerable amount of fault tolerance as it omits a function call if it encounters an exception, instead of halting execution of the application as a whole. Other factors which increase the fault tolerance of the system include validation of input. Checks are in place to ensure the qubit number and step number do not drop below one as it would not make sense to attempt to simulate a circuit where either of these values are zero.

To a user who has little knowledge of the workings of a quantum system may find the system difficult to use, with minimal prompts to inform the user of the proper usage of the system. The user may not fully understand the function of the gate, or perhaps even the purpose of the logic gates on the whole, as little explanation of how the results have been derived from the simulated system.

To a user who is knowledgeable in the field of quantum computing and understands the notation of the gates involved, as well as the underlying mathematics to obtain the values, the product should be easy to learn. The graphical prompts give feedback and aid the usage of the system. For example, highlighting the selected gate or highlighting where the gate will be placed on click.

To improve learnability, semantic and contextual prompts could be added. Potential areas for these include the toolbox, so the user may understand the usage of the logic gates. Another potential feature may be to add a step-by-step evaluation, to show the change of qubit states over the process of the simulation. This breakdown should help the user to learn how the simulated circuit has transformed the initial qubit states into their final results.

In terms of operability, the product is a stand alone product and does not provide or require functions which call other external systems aside from external libraries. These have been downloaded and referenced locally so operability is not a major concern when looking at its external calls. Internally, the system operates well under small scale simulated systems, functioning in a timely manner and remaining stable. On a larger scale of simulated system, the application does not utilize available resources well and is prone to critically failing. A short term fix would be to apply a hard coded limit on the size of the system to be evaluated, in order to restrict the amount of resources required to perform an evaluation. A more suitable fix would be to analyse the evaluation algorithm, and identify areas of the code base to be optimised.

The attractiveness of the product changes considerably with the size of the window. When the window is large enough, the system displays as intended with information easily accessible and to identify. Upon shrinking of the window, the web page tries to re-organise the elements in order to fit into the new window size, disjointing some elements from their intended location. One approach to improving this would be to reconsider positioning of elements. It may be more flexible to display the results on the left to match flow layout which web pages try to implement. Other approaches will involve researching possible methods to assist in maintaining

a layout.

Facets concerning time behaviour have been previously mentioned; it operates in a timely manner when dealing with small scale systems, but suffers on larger simulations. Especially so with a large number of logic gates to evaluate. Drawing of graphical elements is fairly optimised and timely, as the system only draws the element which requires an update, instead of redrawing the whole system. However, when performing an evaluation, the algorithm must be run over all logic gates devised in the system and is currently the largest threat to the system's performance. In the future, it may be effective to analyse the algorithm which performs the evaluation and identify points of optimisation in order to increase the timeliness of the system.

Outside of the code base, design decisions may have factored into the product's timeliness. The user must place one gate at a time, with no alternative option to put down multiple gates concurrently. An improvement in this respect may be to add a feature which allows the user to somehow place multiple gates in quick succession. This could be realised as drawing an area to place the gates, or using a textbox prompt to specify a range to place the logic gates in.

When evaluating testability, it is important to consider the range of tests to be performed, as different groupings of tests must be performed in different methods. The factors to be tested within the product involve the graphical representation of the constructed circuit, the data representation of the circuit constructed (through the graphical interface), and the accuracy of the evaluation of the circuit.

Testability of the graphical interface is currently a manual process in which the tester must execute the application and construct a circuit using a variety of tools available, and changing the properties of the circuit set up. The nature of the product makes it difficult to automate to test the graphical interface, as many test suites available test graphical web controls such positioning of textboxes or buttons, as opposed to dynamically drawn elements. The data representation of the circuit when constructed through the use of the graphical interface is also currently performed as manual tests. Again, this is due to the nature of the graphical implementation of the product.

The evaluation of the circuit has a higher level of testability, currently realised as a set of unit tests which can be run on execution of the application. The unit test suite tests the accuracy of the quantum logic gates, both in isolation and some cases where it makes sense to test the gates in combination such as testing entanglement or controlled gates. The speed of testing and error detection is increased by many magnitudes. The tests are completed in a small time period, and returns results immediately upon completion. The developer can detect faulty code almost immediately and these tests can be extended to cover new implementations as the unit tests have been refactored. The developer simply needs to call the test with the appropriate parameters.

From the evaluation the developer believes the product is of a fairly high quality, though there is much room for improvement. The product meets all requirements as laid out in the specification, and considerable work has gone into improving quality factors concerning the product, such as timeliness, stability and maintainability. Further work will go into the product to further improve its quality.

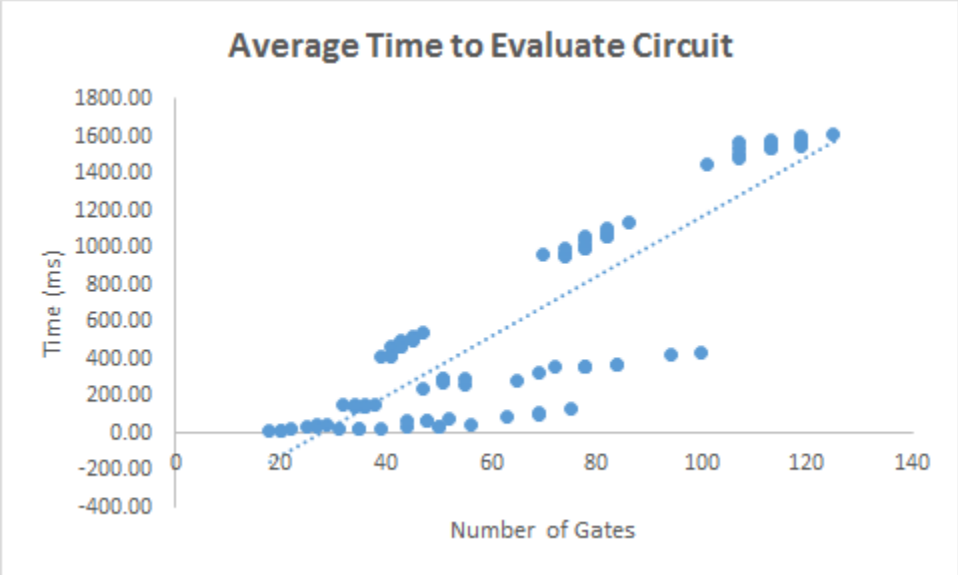


Figure 17: Execution times of evaluator against number of logic gates

To further evaluate the product, the performance of the evaluator shall be analysed. It is expected the simulator will not be able to match the predicted processing speeds of quantum algorithms. To perform the analysis, Grover's algorithm will be simulated multiple times, with an increasing flag bit to be searched for. It is expected larger circuits with higher number of logic gates will take longer to evaluate. A comprehensive tabulation of the results can be found in the appendix.

While the results are not exactly as predicted, they still show an increase in execution time correlating to the number of logic gates to be simulated, with a larger magnitude than the polynomial execution times expected of quantum algorithms. An explanation for the staged groupings of results seen in Figure 17 may be due to the number of qubits required to simulate Grover's Algorithm on that certain flag bit. With more qubits, the algorithm implemented must expand the logic gate's matrix to match the size of the qubits' possible states, hence a lengthier execution time.

The algorithm currently implemented is of complexity $O(n)$ as the algorithm scales linearly, in two respects; The number of gates or the number of qubits. When either of these parameters increase, so does the execution time. To improve on this, it would be worth analysing the algorithm itself to find points of optimisation, or perhaps to implement a faster algorithm. For example, Spector's Implicit Expansion Algorithm may be faster as it does not require expanding the logic gate's matrix.

# 13 Conclusions

Throughout the project, development leveraged the usage of evolutionary prototypes to deliver functionality iteratively. This allowed the developer to stage the development, which created conceptual milestones to achieve to drive the development. Following this ideal allowed for a more robust and functional application at the end of each iteration, with each iteration bringing more and more features.

This worked well, as the abstraction of functionality to be achieved allowed the developer to better focus on the current objective to be met. For example, during the development of the first prototype, the developer needed to worry only about the graphical elements which built up the user interface, without worryingh about implementing the algorithms to evaluate the circuit. The testing phase at the end of each prototype allowed the developer to decide which feature to work on during the next prototype. Hence, why prototype three saw the optimisation of the graphical elements.

To improve upon the prototyping methodology used, it may have been useful to formalise time periods to deliver prototypes on. With the irregular prototype completion frequency which came as a result of the informalised evolutionary process, the proportions of time spent on each aspect or phase of the development cycle varied from prototype to prototype. While it is expected larger applications may take longer to design upon or test upon, the product developed saw a wide range of time portions contributing to each prototype, seemingly with little to no correlation. This had the knock on affect of the developer sometimes being in the incorrect mind set. For example, feeling he should still be in the testing phase due to the length of the testing phase in the previous iteration. Formalising the time periods may have allowed for a more regular process and improved the flow of the development.

Further improvements would be to make use of the resources available, extending this definition to include personnel and time. Along with the evolutionary prototypes it would have been useful to meet with supervisors to gather feedback on how to progress and further the product with additional features and academic insights. This will be a good process to carry forward into future projects.

The performance of the literature, technical review, as well as evaluation of existing product allowed the developer to garner insights to the current state of the field. This was useful as it allowed the developer to understand the quality required of the product to be developed, as well as any academic or technical interests involved with the product to be developed. Both online and physical sources were consulted, ranging from a wide variety of media, from academic journals, textbooks and short online videos giving a large scope into the research.

Although this process was performed to what the developer believes is an adequate level, it may have been hindered by the developer's desire to build a specific application. In the future, it will be better to take a more objective and neutral view upon the findings of the research, before

beginning any design or implementation of the product. This would allow for the project to be directed toward building a product with a higher level of interests from both the academic and technical areas of the involved field, and thus deliver a product of higher all around value.

Throughout the duration of the project the developer has gained an increase in competency and technical ability having worked with mostly technologies previously not known to the developer such as JavaScript and MongoDB, as well as the inclusion of widely used libraries such as jQuery and Bootstrap. Abilities were also improved in technologies already known such as PHP, HTML and CSS. Perhaps the greatest achievement is the implementation of AJAX to submit asynchronous requests and retrieve data from the database.

Abilities the developer will endeavour to work on in the future is regulation of formal processes. The project undertaken was carried out in a particularly informal manner which worked well as it suited the nature of a project performed by an individual as opposed to a team, but also created periods of times when the developer was not fully confident in the next step or the direction to take the project into. This could be aided by formalisation, such as organising regular meetings and utilising the full range of available resources.

Overall, the developer is content with the product created as a deliverable of this project as it meets the initial requirements laid out in the specification, utilises fairly new technologies which are of interest and further work is planned to continue to improve and increase the quality of the product.

# 14 References

WIRED, 2007. *The Father of Quantum Computing*. [online] Available at: http://www.wired.com/science/discoveries/news/2007/02/72734 [Accessed 6 January 2014]

Bacon, D. and Van Dam, W., 2010. Recent Progress in Quantum Algorithms. *Communications of the ACM*, [e-journal] 53(2). Available through: Communications of the ACM website at: <http://cacm.acm.org/> [Accessed 6 January 2014]

Arikan, E., 2003. *An information-theoretic analysis of Grover's algorithm.* [pdf] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1228418> [Accessed 12 April 2014]

Van Meter, R., and Horsman, C., 2013. A Blueprint for Building a Quantum Computer. *Communications of the ACM*, [e-journal] 58(10) Available through: Communications of the ACM website at: <http://cacm.acm.org/> [Accessed 6 January 2014]

Mone, G., 2013. Future-Proof Encryption. *Communications of the ACM*, [e-journal] 56(11). Available through: Communications of the ACM website at: <http://cacm.acm.org/> [Accessed 6 January 2014]

D-Wave, 2013. *Quantum Computing - Gate Model v Adiabatic*. [video online] Available at: <https://www.youtube.com/watch?v=4tAG-qyikFc> IBM, 2001. *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance.* [pdf] IBM Almaden Research Center: Macmillan Magazines. Available at: <http://cryptome.org/shor-nature.pdf> [Accessed 6 January 2014]

Lu, C., et al., 2007. *Demonstration of Shor's quantum factoring algorithm using photonic qubits.* [pdf] Available at: http://arxiv.org/pdf/0705.1684v3.pdf [Accessed 6 January 2014]

BBC, 2013. Quantum memory 'world record' smashed. *BBC News*, [online] 15 November 2013. Available at: <http://www.bbc.co.uk/news/science-environment-24934786> [Accessed 6 January 2014]

D-Wave, 2014. The D-Wave Two System - The first commercial quantum computer. *D-Wave*, [online] 2014. Available at: <http://www.dwavesys.com/d-wave-two-system> [Accessed 01 April 2014]

Nature, 2013. Google and NASA snap up quantum computer. *Nature*, [online] 16 May 2013. Available at: <http://www.nature.com/news/google-and-nasa-snap-up-quantum-computer-1.12999> [Accessed 01 April 2014]

Nature, 2013. Computing: The quantum company. *Nature*, [online] 19 June 2013. Available at: <http://www.nature.com/news/computing-the-quantum-company-1.13212> [Accessed 01

April 2014]

Johnson, M. W. et al. *Nature* 473, 194-198 (2011)

Neven, H., et al., 2009. *Training a Large Scale Classifier with the Quantum Adiabatic Algorithm.* [pdf] Available at: <http://arxiv.org/pdf/0912.0779v1.pdf> [Accessed 01 April 2014]

Perdomo-Ortiz, A., et al., 2012. Finding low-energy conformations of lattice protein models by quantum annealing. *Nature* [online] Available at: <http://www.nature.com/srep/2012/120813/srep00571/full/srep00571.html> [Accessed 01 April 2014]

NewScientist, 2013. New language helps quantum coder build killer apps. *NewScientist*, [online] 05 July 2013. Available at: <http://www.newscientist.com/article/dn23820-new-language-helps-quantum-coders-build-killer-apps.html#.U0mZKPldVP8> [Accessed 12 April 2014]

Dropbox, 2014. Pricing. *Dropbox* [online] Available at: <https://www.dropbox.com/pricing> [Accessed 18 January 2014]

University of Greenwich, 2013. School of Computing & Mathematical Sciences, Student Technical Support. *University of Greenwich*, [online] 16 October 2013. Available at: <http://portal.gre.ac.uk/tag.dbc05d659fb70c4e.render.userLayoutRootNode.uP?uP_root=root&uP_sparam=activeTab&activeTab=u12l1s22&uP_tparam=frm&frm=frame> [Accessed 18 January 2014]

W3C, 2014. HTML5 - W3C Candidate Recommendation. *W3C* [online] Available at: <http://www.w3.org/TR/html5/scripting-1.html#the-canvas-element> [Accessed 05 April 2014]

W3C, 2011. Scalable Vector Graphics (SVG) 1.1 (Second Edition) - W3C Recommendation *(*W3C [online] Available at: <http://www.w3.org/TR/SVG11/intro.html> [Accessed 05 April 2014]

Wybiral, D., 2013. *Quantum Circuit Simulator* [online] Available at: <http://www.davyw.com/quantum/> [Accessed 9 January 2014]

Federal University of Campina Grande. *Zeno* [online] Available at: <http://dsc.ufcg.edu.br/~iquanta/zeno/index_en.html> [Accessed 9 January 2014]

Spector, L. (2006). *Automatic Quantum Computer Programming: A Genetic Programming Approach* New York, Springer Science+Business Media, LLC

# 15 Appendices

Figure 18: Project Proposal

## An Overview of the Development of a Quantum Computer Simulator

**Jonathan Law - lj048**
**BEng Software Engineering**
**000611905**

## 15.1 Project Proposal

### 15.1.1 Overview

The objective of the project is to develop a simulator with the capability of supporting the computations and evaluations of a quantum computer at the circuitry level. The software will allow users to specify an amount of qubits, their input states and the logic gates to operate on the qubits. Once the set-up of the circuitry is completed, the software will be capable of analysing the output and give a probability of each qubit's corresponding 0 and 1 states. Achieving these functionality should also aid users in the simulation of quantum algorithms; Algorithms which can only be completed specifically by quantum computers due to their use of superpositions. The focus of this project is the functionality and accuracy of the simulator, over the ease of use of factors such as the UI or responsiveness, though these factors will still be considered.

Keywords: [simulator, quantum, computer, superpositions]

### 15.1.2 Aim

The aim is to provide a simulator which allows the user to simulate the circuitry level of a quantum computer. The means of deployment will depend on the finFdings of the research, as well as the requirements which will be finalised at a later date.

### 15.1.3 Objectives

Objectives are described using nouns, either concrete nouns or abstract nouns. Some objectives are big and some objectives are small. All objectives must be SMART. Objectives are created by activities. Activities are described using verbs. An objective may be created by a single activity

or by several activities. Activities take time. Provide a time estimate in days for each activity (as shown below in square brackets).

At the highest level your objectives are; a research report, design documentation, an implementation, an evaluation report. These high level objectives must be broken down into smaller objectives according to your individual project and the deliverables expected by your supervisor. All required project deliverables should be included here as an objective. Objectives should be ordered in approximate chronological order of creation, so 'Statement of Requirements' will come before 'Paper Prototype' even though there may be iteration between these two objectives.

Provided below is a product breakdown structure; Listed will be the products which should be developed in order for the project to be successful, and each product will detail tasks required for that specific product. See the GANTT Chart attached in the appendix to understand the time scheduling associated with tasks.

1. Compiled Report
    (a) Literature Review
        i. Read peer review sources around the subject area to gain better understand of the current field.
        ii. Create document to record and discuss findings.
    (b) Technical Review
        i. Research and choose from available programming languages to implement the application.
        ii. Create document to record and discuss findings.
    (c) Existing Product Review
        i. Research and evaluate similar existing applications which achieves similar objectives.
        ii. Create document to record and discuss findings.
    (d) Design Documentation
        i. Develop design documents to graphically show the intended layout of the application and its typical use case scenario.
        ii. Develop functionality documents to diagrammatically show aspects of the implementation, such as use cases, behaviours, etc.
    (e) Implementation Documentation
        i. Document progress through each prototype and discuss problems.
    (f) Testing Documentation
        i. Plan and prepare documentation to allow outlining of test plans and results.
        ii. Perform manual tests outlined against documentation against application.

2. QC Simulator Application
    (a) Design Documentation
        i. (Identical to the Design Documentation product outlined in the Compiled Report)
    (b) Implementation Documentation
        i. (Identical to the Implementation Documentation product outlined in the Compiled Report)
    (c) Testing Documentation
        i. (Identical to the Testing Documentation product outlined in the Compiled Report)
    (d) Application
        i. Prototype One
            A. Create basic web controls
            B. Develop toolbox to allow users to select gates
            C. Create graphical circuit to allow users to place gates
            D. Allow space to display evaluation content
        ii. Prototype Two
            A. Implement algorithm to calculate results of simulated circuit
            B. Display evaluation
        iii. Prototype Three
            A. Add saving of circuit design
            B. Add loading of circuit design

### 15.1.4 Legal, Social, Ethical and Professional

As a professional, the system should be accurate when evaluating the output of a simulated circuit. Users may be using the product as a tool to aid in the evaluation of quantum algorithms, and incorrect results could possibly lead to an incorrect conclusion about the work they are conducting.

Research and testing will help to mitigate against this, but to allow for retroactive correct after the system is completed, users should be able to contact the developer or give feedback flagging any inaccuracies. They will be checked and amended, available in an update of the product.

### 15.1.5 Planning

For the purposes of the design documentation, UML will be used to convey the design decisions taken. This will affect the design documentation for the library. The UI will be designed as a series of images, outlining the placement of objects within the window.

During development, using RAD methodology will allow for regular prototyping of the product, enabling feedback and increased changeability in the product. It also suits the low manpower and timeliness available for the project. JUnit testing will be useful as it allows for automated testing, and is suitable to test the library's accuracy, as the expected output will be the result of a mathematical function.

To manage the project on the whole, tasks will be flagged for completion during an iteration. Any leftover or uncompleted tasks will carry over to the next iteration. This may result in the need for using contingency time.

### 15.1.6 Initial References

Wikipedia, 2013, *Quantum Computer.* [online] Available at <http://en.wikipedia.org/wiki/Quantum_computing> [Accessed 04 November 2013]

Norton Q, 2007, The Father of Quantum Computing. *WIRED*, [online] 15 February, Available at: <http://www.wired.com/science/discoveries/news/2007/02/72734> [Accessed 04 November 2013]

Wikipedia, 2013, *Quantum Gate*. [online] Available at <http://en.wikipedia.org/wiki/Quantum_gate> [Accessed 04 November 2013]

Van Meter, R and Horsman, C, 2013. A Blueprint for Building a Quantum Computer. *Communications of the ACM*, 58(10), pp.84-93.

| ID | Name | Duration | Start | Finish | Predecessor |
|---|---|---|---|---|---|
| 1 | Literature Review | 14 days | 21/11/2013 | 05/12/2013 | |
| 2 | Technical Review | 14 days | 05/12/2013 | 19/12/2013 | 1 |
| 3 | Existing Product Review | 14 days | 19/12/2013 | 02/01/2014 | 2 |
| 4 | Prototype 1 | 31 days | 02/01/2014 | 02/02/2014 | 3 |
| 5 | Create basic web controls | 2 days | 16/01/2014 | 18/01/2014 | |
| 6 | Develop toolbox to allow users to select gates | 7 days | 18/01/2014 | 25/01/2014 | 5 |
| 7 | Create graphical circuit to allow users to place gates | 7 days | 25/01/2014 | 01/02/2014 | 6 |
| 8 | Allow space to display evaluation content | 1 day | 01/02/2014 | 02/02/2014 | 7 |
| 9 | Prototype 2 | 31 days | 03/02/2014 | 06/03/2014 | 4 |
| 10 | Implement algorithm to calculate results of simulated circuit | 21 days | 12/02/2014 | 05/03/2014 | |
| 11 | Display evaluation | 1 day | 05/03/2014 | 06/03/2014 | 10 |
| 12 | Prototype 3 | 31 days | 07/03/2014 | 07/04/2014 | 9 |
| 13 | Add saving of circuit design | 7 days | 24/03/2014 | 31/03/2014 | |
| 14 | Add loading of circuit design | 7 days | 31/03/2014 | 07/04/2014 | 13 |
| 15 | Compilation and Final Checks of Report | 7 days | 07/04/2014 | 14/04/2014 | 12 |

Figure 19: Table showing proposed schedule

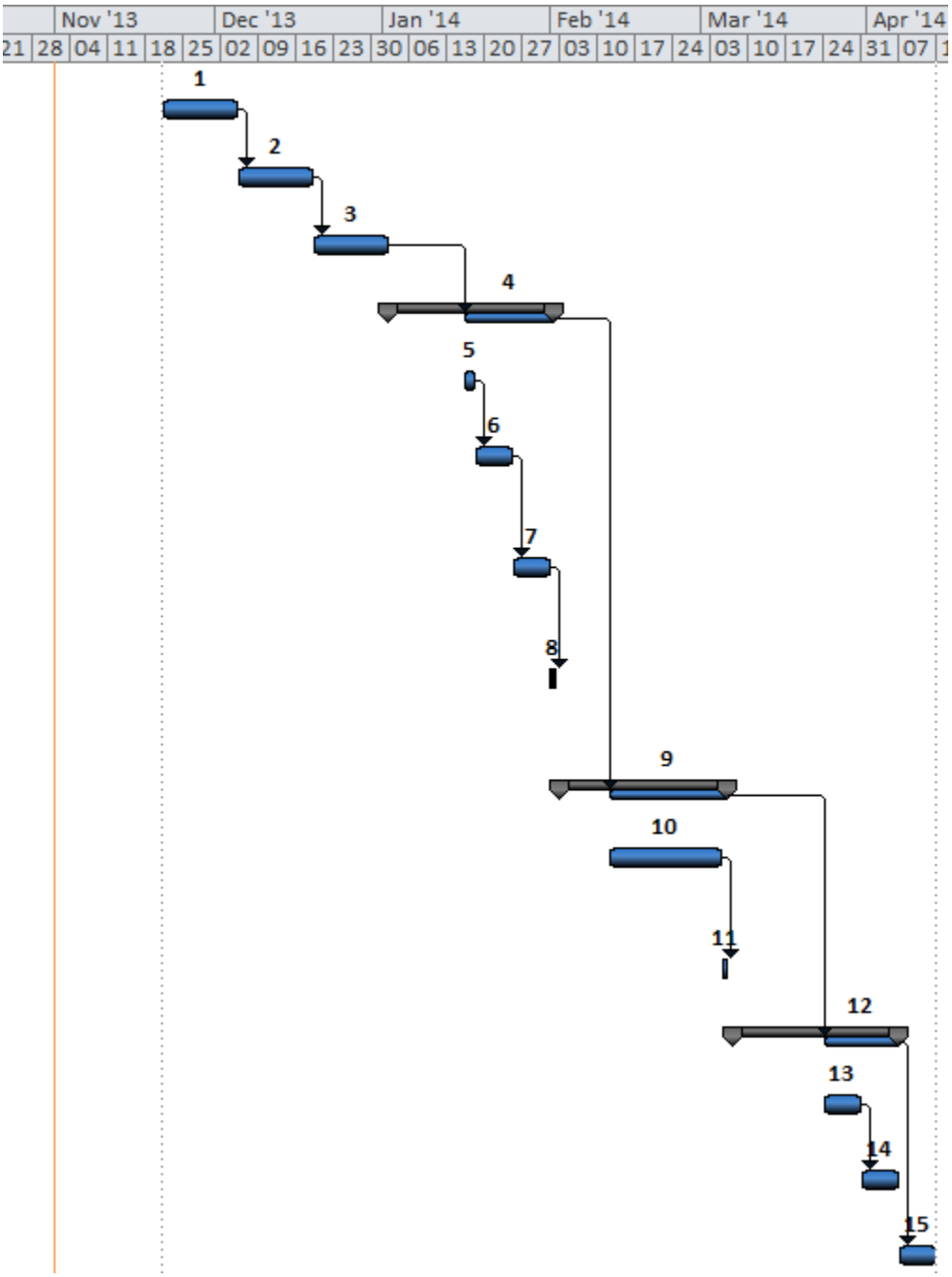


Figure 20: GANTT chart of proposed schedule

## 15.2 Design Documentation

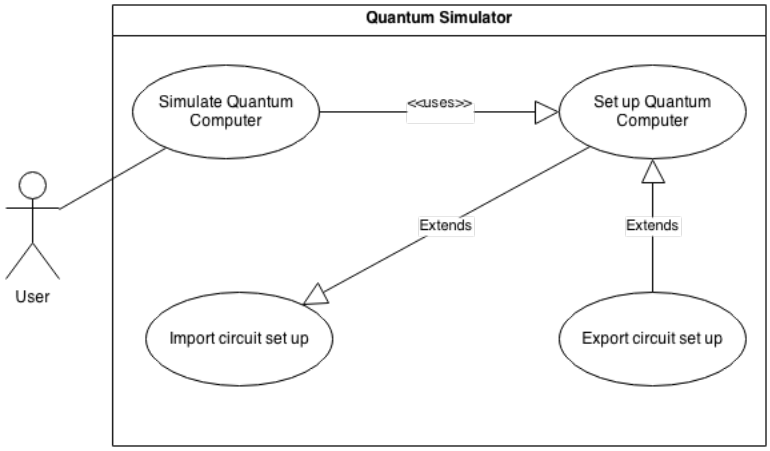


Figure 21: Use case diagram (version 1) for the proposed application

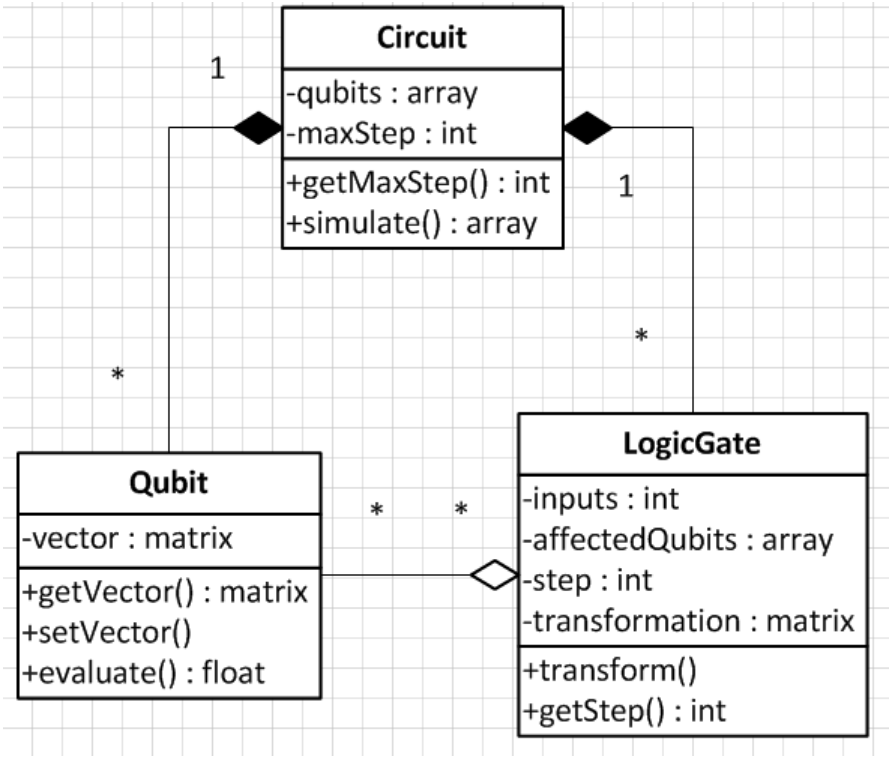


Figure 22: Class diagram (version 1) for the proposed application

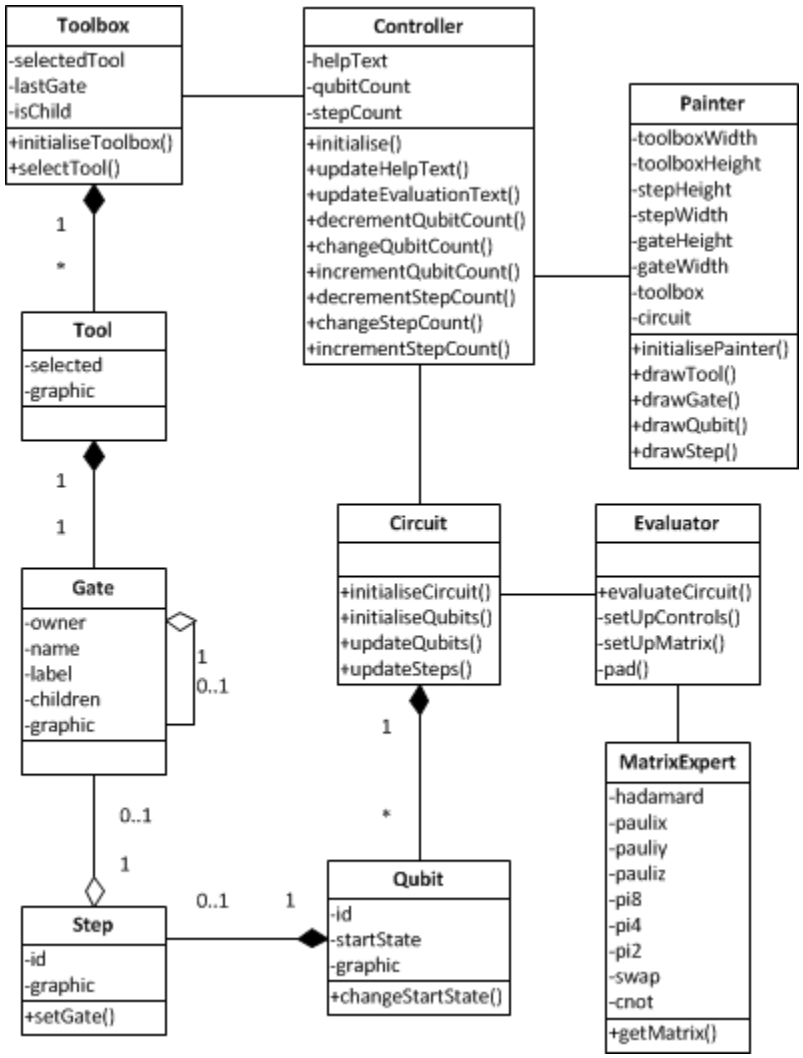


Figure 23: Class diagram (version 2) for the proposed application

## 15.3 Screenshots of Program

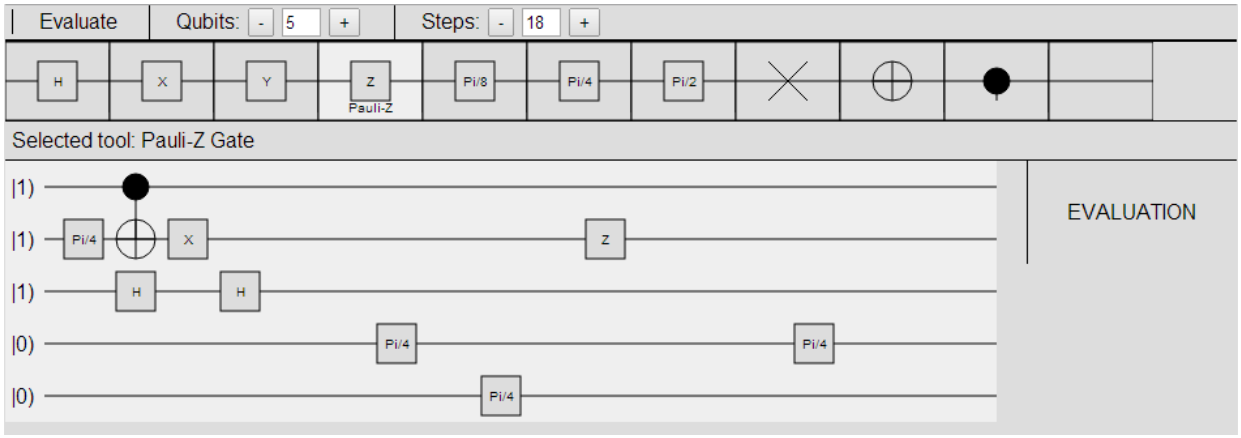


Figure 24: Screenshot of prototype version 1

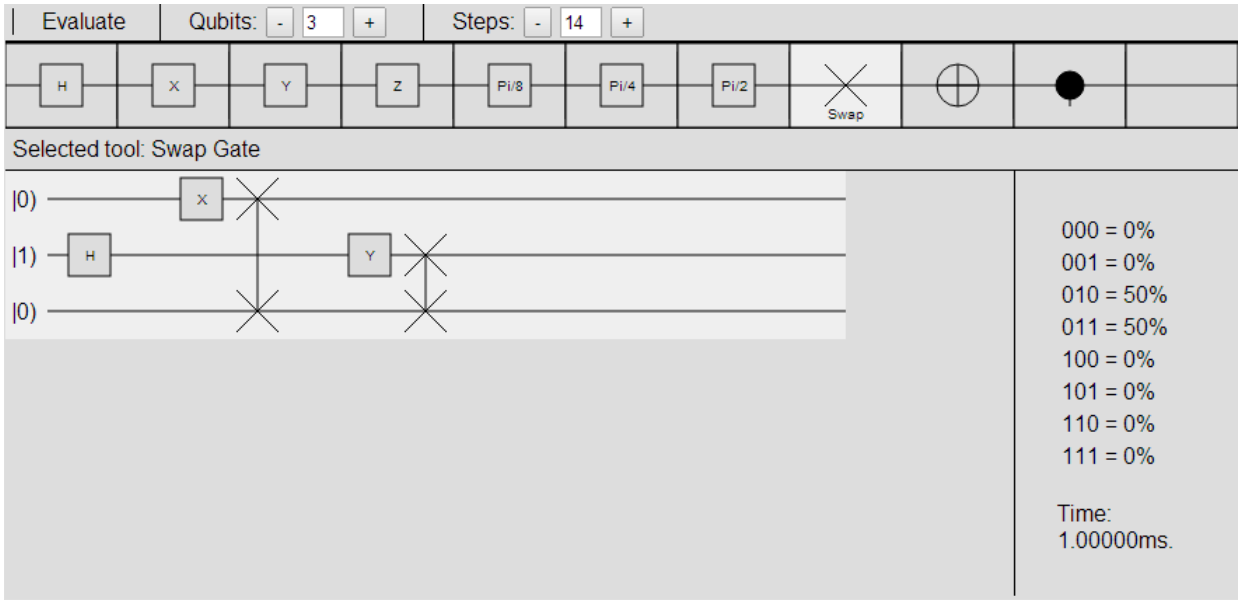


Figure 25: Screenshot of prototype version 2

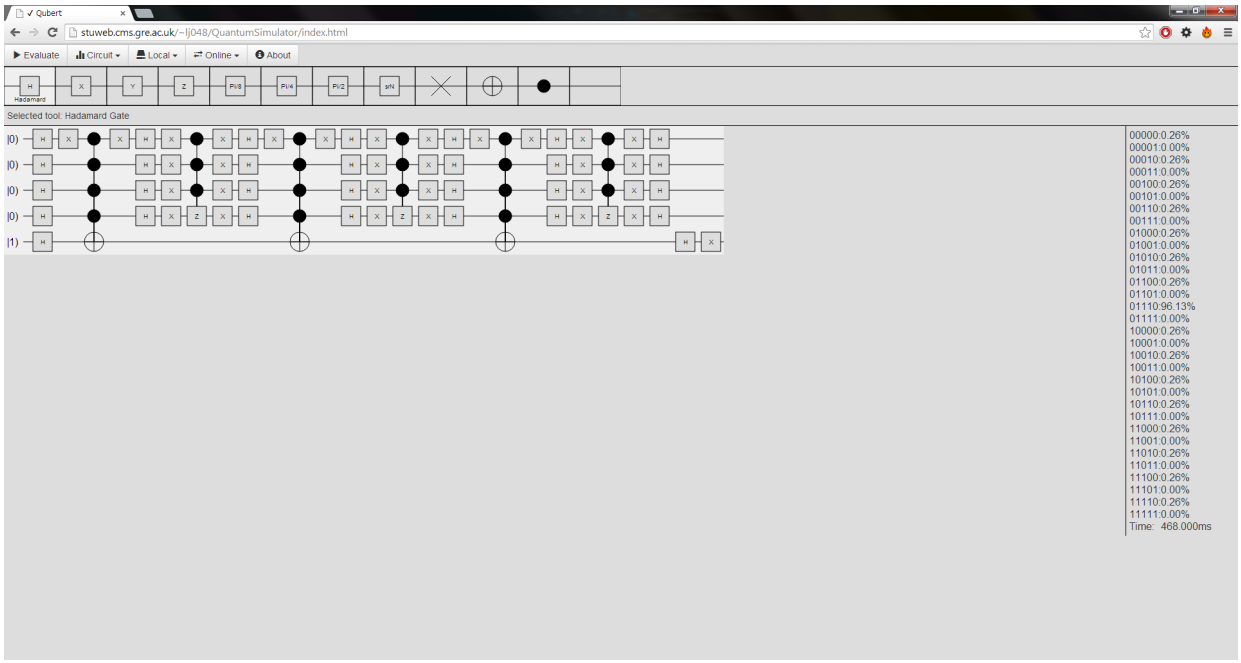


Figure 26: Screenshot of prototype version 4

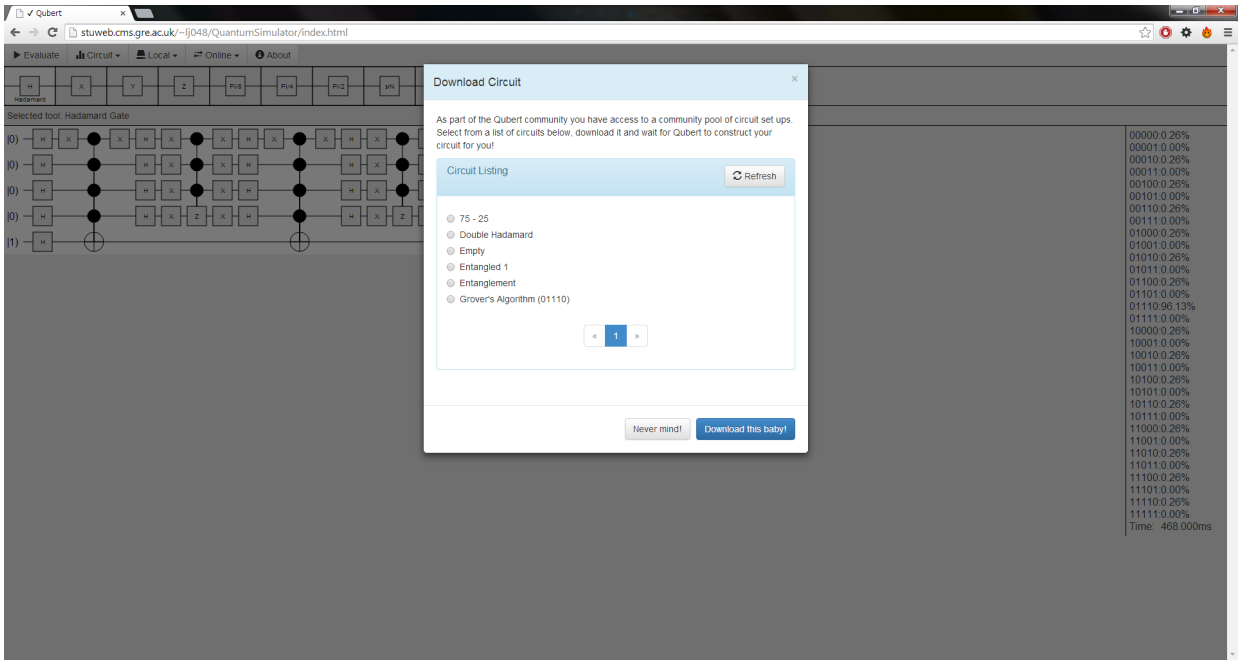


Figure 27: Screenshot of prototype version 4 with a modal window

## 15.4 Testing Documentation

Table 1: Manual Test Results.

| ID | Test | Expectation | Result | Action Taken |
|---|---|---|---|---|
| 1 | Press Evaluate button | Results appear in evaluation pane. | Results appear in evaluation pane. | None taken. |
| 2 | Select tool | Tool is highlighted and help text updates. | Help text did not update. | Called function to update help text after tool selection. |
| 3 | Place gate | Gate is placed on circuit in correct area. | Gate is placed on circuit in correct area. | None taken. |
| 4 | Place gate on top of another | Gate is replaced over old gate. | Gate is replaced over old gate. | None taken. |
| 5 | Change qubit state | Qubit changes state. | Qubit changes state. | None taken. |
| 6 | Increase qubit count | Another qubit is added to circuit. | Qubit was added but not drawn. | Called draw function after qubit count changes. |
| 7 | Decrease qubit count | Last qubit to be removed from circuit. | Last qubit to be removed from circuit. | None taken. |
| 8 | Increase step count | Length of wire to increase by one step. | Length of wire to increase by one step. | None taken. |
| 9 | Decrease step count | Length of wire to decrease by one step. | Drawing area shrunk, removing step graphic, but did not splice the logical object. | Correctly spliced the step on step decrease. |
| 10 | Press reset button and accept | Circuit returns to original state. | Circuit returns to original state. | None taken. |
| 11 | Press reset button and decline | No change to circuit. | Modal window did not face out. | Added dismiss call on the decline button. |
| 12 | Save circuit | Window to appear with JSON in textbox. | Window to appear with JSON in textbox. | None taken. |
| 13 | Load good circuit | Circuit to change to new configuration. | Alert informing user of rejected JSON string. | Adjusted JSON string obtained from save to correctly match accepted input. |
| 14 | Load bad circuit | Alert informing user of rejected JSON string. | Alert informing user of rejected JSON string. | None taken. |

Continued on next page

**Table 1 – continued from previous page**

| ID | Test | Expectation | Result | Action Taken |
|---|---|---|---|---|
| 15 | Upload circuit | Circuit to be uploaded after entering circuit name. | Alert informing user of failed upload. | Modified PHP to use the correct method calls to insert into MongoDB. |
| 16 | Download circuit | Circuit to change to new configuration. | Circuit to change to new configuration. | None taken. |
| 17 | Increment page (on download view) | Page counter increments, and new page of circuits displayed. | Page counter did not increment. No new page shown. | Correctly incremented page counter, and applied fix to decrement page counter. |
| 18 | Decrement page (on download view) | Page counter decrements, and new page of circuits displayed. | Page counter decrements, and new page of circuits displayed. | None taken. |

Figure 28: Manual Test Results

Table 2: Performance Test Run 1-1 Results.

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 66415.3 | 66415.3 | (idle) | |
| 2.0 | 20325.1 | updateCircuit | circuitry.js:35 |
| 6.1 | 20095.8 | f | raphael-2.1.2.js:3098 |
| 4.0 | 19854.4 | drawCircuit | circuitry.js:19 |
| 34.3 | 19735.2 | draw | qubit.js:34 |
| 0 | 19648.3 | (anonymous function) | step.js:141 |
| 1.0 | 19648.3 | setGate | step.js:20 |
| 131.3 | 18093.6 | draw | step.js:124 |
| 163.7 | 12611.3 | (anonymous function) | raphael-2.1.2.js:5233 |
| 202.0 | 12572.9 | setproto.forEach | raphael-2.1.2.js:5223 |
| 1637.5 | 12397.1 | (anonymous function) | raphael-2.1.2.js:5235 |
| 1564.8 | 10486.8 | R.(anonymous function). elproto.(anonymous function) | raphael-2.1.2.js:3431 |
| 5362.1 | 8926.1 | e | raphael-2.1.2.js:3097 |
| 1575.9 | 4702.5 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 4509.5 | 4509.5 | (garbage collector) | |
| 3564.0 | 3564.0 | addEventListener | |
| 2919.5 | 2919.5 | (program) | |
| 72.7 | 2228.5 | paperproto.path | raphael-2.1.2.js:3746 |

Continued on next page

**Table 2 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 79.8 | 2109.3 | R._engine.path | raphael-2.1.2.js:6444 |
| 32.3 | 1986.0 | drawGate | gate.js:113 |
| 123.2 | 1913.3 | elproto.attr | raphael-2.1.2.js:6750 |
| 44.4 | 1797.1 | paperproto.text | raphael-2.1.2.js:3791 |
| 488.9 | 1744.6 | $ | raphael-2.1.2.js:5792 |
| 1573.9 | 1577.9 | setAttribute | |
| 44.4 | 1228.4 | tuneText | raphael-2.1.2.js:6319 |
| 74.8 | 1161.7 | paperproto.rect | raphael-2.1.2.js:3687 |
| 7.1 | 980.9 | elproto._getBBox | raphael-2.1.2.js:6653 |
| 973.8 | 973.8 | getBBox | |
| 41.4 | 824.3 | R._pathToAbsolute | raphael-2.1.2.js:2104 |
| 0 | 597.0 | incrementStepCount | main.js:72 |
| 0 | 597.0 | onclick | index.html:27 |
| 0 | 597.0 | updateSteps | circuitry.js:52 |
| 109.1 | 558.6 | R.parsePathString | raphael-2.1.2.js:1465 |
| 34.3 | 468.7 | R.clear | raphael-2.1.2.js:7088 |
| 183.9 | 423.3 | eve | raphael-2.1.2.js:54 |
| 416.2 | 416.2 | removeChild | |
| 385.9 | 389.9 | appendChild | |
| 346.5 | 346.5 | (anonymous function) | raphael-2.1.2.js:1548 |
| 1.0 | 290.9 | onclick | index.html:25 |
| 3.0 | 289.9 | evaluateCircuit | circuitry.js:65 |
| 2.0 | 285.9 | (anonymous function) | circuitry.js:101 |
| 237.4 | 266.7 | (anonymous function) | raphael-2.1.2.js:1480 |
| 249.5 | 249.5 | clone | raphael-2.1.2.js:795 |
| 239.4 | 239.4 | eve.listeners | raphael-2.1.2.js:129 |
| 16.2 | 227.3 | setUpMatrix | circuitry.js:154 |
| 135.4 | 199.0 | newf | raphael-2.1.2.js:1210 |
| 65.7 | 152.5 | paths | raphael-2.1.2.js:1539 |
| 1.0 | 145.5 | (anonymous function) | step.js:153 |
| 2.0 | 144.5 | unhighlight | step.js:167 |
| 1.0 | 133.3 | R._engine.text | raphael-2.1.2.js:6947 |
| 1.0 | 131.3 | drawConnections | gate.js:208 |
| 2.0 | 130.3 | (anonymous function) | gate.js:210 |
| 108.1 | 108.1 | R.is | raphael-2.1.2.js:780 |
| 25.3 | 103.0 | e.index | math.min.js:29 |
| 100.0 | 100.0 | createElementNS | |
| 93.9 | 93.9 | get scrollTop | |
| 0 | 90.9 | updateQubits | circuitry.js:41 |
| 0 | 90.9 | onclick | index.html:26 |
| 22.2 | 90.9 | t.subset | math.min.js:27 |

Continued on next page

**Table 2 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 0 | 90.9 | incrementQubitCount | main.js:45 |
| 86.9 | 86.9 | setTimeout | |
| 1.0 | 82.8 | (anonymous function) | step.js:147 |
| 2.0 | 81.8 | highlight | step.js:160 |
| 79.8 | 79.8 | apply | |
| 1.0 | 79.8 | getEventPosition | raphael-2.1.2.js:3086 |
| 78.8 | 78.8 | R._path2string | raphael-2.1.2.js:1201 |
| 73.7 | 75.8 | Element | raphael-2.1.2.js:6356 |
| 64.7 | 67.7 | t | math.min.js:27 |
| 63.6 | 63.6 | repush | raphael-2.1.2.js:1204 |
| 7.1 | 61.6 | paperproto.circle | raphael-2.1.2.js:3661 |
| 5.1 | 55.6 | y | math.min.js:28 |
| 1.0 | 50.5 | unhighlight | gate.js:194 |
| 0 | 47.5 | (unresolved function) | |
| 22.2 | 46.5 | n | math.min.js:28 |
| 1.0 | 46.5 | a | math.min.js:27 |
| 45.5 | 45.5 | get style | |
| 43.4 | 43.4 | call | |
| 3.0 | 42.4 | highlight | gate.js:182 |
| 0 | 40.4 | (anonymous function) | gate.js:101 |
| 29.3 | 29.3 | (anonymous function) | raphael-2.1.2.js:1483 |
| 29.3 | 29.3 | getPropertyValue | |
| 2.0 | 28.3 | pathClone | raphael-2.1.2.js:2020 |
| 8.1 | 28.3 | t.set | math.min.js:27 |
| 0 | 28.3 | (anonymous function) | gate.js:106 |
| 3.0 | 27.3 | selectGate | toolbox.js:33 |
| 0 | 27.3 | (anonymous function) | gate.js:96 |
| 24.2 | 26.3 | r | math.min.js:27 |
| 19.2 | 21.2 | t.get | math.min.js:27 |
| 1.0 | 19.2 | e.zeros | math.min.js:29 |
| 7.1 | 18.2 | g | math.min.js:28 |
| 14.1 | 18.2 | paperproto.set | raphael-2.1.2.js:3813 |
| 12.1 | 16.2 | R.format | raphael-2.1.2.js:5643 |
| 0 | 16.2 | changeStartState | qubit.js:90 |
| 0 | 16.2 | (anonymous function) | qubit.js:65 |
| 14.1 | 14.1 | n.isNumber | math.min.js:31 |
| 1.0 | 14.1 | n.resize | math.min.js:31 |
| 1.0 | 13.1 | setUpControls | circuitry.js:138 |
| 4.0 | 13.1 | o | math.min.js:31 |
| 2.0 | 12.1 | R._engine.rect | raphael-2.1.2.js:6919 |
| 1.0 | 11.1 | R._engine.circle | raphael-2.1.2.js:6910 |

Continued on next page

**Table 2 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 8.1 | 8.1 | onSubtreeModified | inject_actions.js:11 |
| 8.1 | 8.1 | l | math.min.js:27 |
| 1.0 | 7.1 | u | math.min.js:27 |
| 7.1 | 7.1 | t.min | math.min.js:27 |
| 3.0 | 7.1 | n.size | math.min.js:31 |
| 7.1 | 7.1 | setproto.push | raphael-2.1.2.js:5185 |
| 6.1 | 6.1 | get firstChild | |
| 4.0 | 4.0 | slice | |
| 0 | 4.0 | (anonymous function) | math.min.js:27 |
| 4.0 | 4.0 | i | math.min.js:31 |
| 4.0 | 4.0 | getComputedStyle | |
| 3.0 | 3.0 | f | math.min.js:28 |
| 3.0 | 3.0 | t.isScalar | math.min.js:27 |
| 3.0 | 3.0 | t | math.min.js:27 |
| 0 | 3.0 | unhighlight | qubit.js:83 |
| 0 | 3.0 | (anonymous function) | qubit.js:59 |
| 0 | 3.0 | i | math.min.js:27 |
| 0 | 3.0 | t | math.min.js:27 |
| 1.0 | 3.0 | PauliXGate | gate.js:226 |
| 0 | 3.0 | HadamardGate | gate.js:217 |
| 2.0 | 2.0 | (anonymous function) | math.min.js:29 |
| 2.0 | 2.0 | math.min.js:28 | |
| 2.0 | 2.0 | | math.min.js:28 |
| 2.0 | 2.0 | Matrix | raphael-2.1.2.js:2804 |
| 2.0 | 2.0 | (anonymous function) | raphael-2.1.2.js:6304 |
| 0 | 2.0 | Gate | gate.js:8 |
| 1.0 | 1.0 | getElementsByTagName | |
| 1.0 | 1.0 | splice | |
| 1.0 | 1.0 | e.eye | math.min.js:29 |
| 1.0 | 1.0 | get documentElement | |
| 1.0 | 1.0 | f | math.min.js:27 |
| 0 | 1.0 | updateHelpText | main.js:11 |
| 1.0 | 1.0 | set innerHTML | |
| 1.0 | 1.0 | get y | |
| 0 | 1.0 | highlight | qubit.js:76 |
| 0 | 1.0 | Step | step.js:4 |
| 1.0 | 1.0 | get x | |
| 0 | 1.0 | i | math.min.js:27 |
| 0 | 1.0 | (anonymous function) | qubit.js:53 |
| 0 | 1.0 | PauliZGate | gate.js:244 |
| 0 | 1.0 | Qubit | qubit.js:7 |

Figure 29: Performance Test Run 1-1 Results

Table 3: Performance Test Run 1-2 Results.

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 61541.5 | 61541.5 | (idle) | |
| 8.1 | 19923.2 | updateCircuit | circuitry.js:35 |
| 6.1 | 19646.2 | f | raphael-2.1.2.js:3098 |
| 3.0 | 19417.8 | drawCircuit | circuitry.js:19 |
| 38.4 | 19303.6 | draw | qubit.js:34 |
| 1.0 | 19246.0 | setGate | step.js:20 |
| 120.3 | 17047.6 | draw | step.js:124 |
| 0 | 15322.2 | (anonymous function) | step.js:141 |
| 156.7 | 12312.1 | (anonymous function) | raphael-2.1.2.js:5233 |
| 305.3 | 12242.4 | setproto.forEach | raphael-2.1.2.js:5223 |
| 9781.2 | 9781.2 | (garbage collector) | |
| 950.1 | 9734.7 | R.(anonymous function). elproto.(anonymous function) | raphael-2.1.2.js:3431 |
| 4664.7 | 8319.6 | e | raphael-2.1.2.js:3097 |
| 803.6 | 7547.4 | (anonymous function) | raphael-2.1.2.js:5235 |
| 1190.7 | 4422.1 | (anonymous function) | raphael-2.1.2.js:5235 |
| 1084.6 | 4277.6 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 0 | 3923.8 | (anonymous function) | step.js:141 |
| 3654.9 | 3654.9 | addEventListener | |
| 2669.4 | 2669.4 | (program) | |
| 100.1 | 2192.3 | paperproto.path | raphael-2.1.2.js:3746 |
| 22.2 | 2122.6 | drawGate | gate.js:113 |
| 84.9 | 2042.8 | R._engine.path | raphael-2.1.2.js:6444 |
| 28.3 | 1951.8 | paperproto.text | raphael-2.1.2.js:3791 |
| 522.6 | 1830.5 | $ | raphael-2.1.2.js:5792 |
| 1611.2 | 1613.2 | setAttribute | |
| 81.9 | 1463.6 | paperproto.rect | raphael-2.1.2.js:3687 |
| 131.4 | 1442.4 | elproto.attr | raphael-2.1.2.js:6750 |
| 35.4 | 1359.5 | tuneText | raphael-2.1.2.js:6319 |
| 8.1 | 1121.9 | elproto._getBBox | raphael-2.1.2.js:6653 |
| 1113.9 | 1113.9 | getBBox | |
| 26.3 | 768.2 | R._pathToAbsolute | raphael-2.1.2.js:2104 |
| 0 | 592.3 | updateSteps | circuitry.js:52 |
| 0 | 592.3 | onclick | index.html:27 |
| 0 | 592.3 | incrementStepCount | main.js:72 |
| 571.1 | 571.1 | R.is | raphael-2.1.2.js:780 |
| 149.6 | 532.7 | R.parsePathString | raphael-2.1.2.js:1465 |
| 35.4 | 496.3 | R.clear | raphael-2.1.2.js:7088 |

Continued on next page

**Table 3 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 190.0 | 459.9 | eve | raphael-2.1.2.js:54 |
| 443.7 | 445.7 | removeChild | |
| 422.5 | 425.5 | appendChild | |
| 9.1 | 410.4 | R._engine.text | raphael-2.1.2.js:6947 |
| 346.7 | 346.7 | (anonymous function) | raphael-2.1.2.js:1548 |
| 1.0 | 299.2 | incrementQubitCount | main.js:45 |
| 0 | 299.2 | onclick | index.html:26 |
| 268.9 | 269.9 | eve.listeners | raphael-2.1.2.js:129 |
| 192.0 | 214.3 | (anonymous function) | raphael-2.1.2.js:1480 |
| 205.2 | 205.2 | set value | |
| 193.1 | 193.1 | clone | raphael-2.1.2.js:795 |
| 56.6 | 171.8 | paths | raphael-2.1.2.js:1539 |
| 115.2 | 162.7 | newf | raphael-2.1.2.js:1210 |
| 129.4 | 129.4 | createElementNS | |
| 0 | 127.4 | drawConnections | gate.js:208 |
| 2.0 | 127.4 | (anonymous function) | gate.js:210 |
| 117.2 | 117.2 | get scrollTop | |
| 115.2 | 115.2 | setTimeout | |
| 1.0 | 95.0 | unhighlight | step.js:167 |
| 0 | 93.0 | updateQubits | circuitry.js:41 |
| 3.0 | 93.0 | getEventPosition | raphael-2.1.2.js:3086 |
| 1.0 | 86.9 | (anonymous function) | step.js:153 |
| 86.9 | 86.9 | apply | |
| 82.9 | 82.9 | R._path2string | raphael-2.1.2.js:1201 |
| 7.1 | 72.8 | paperproto.circle | raphael-2.1.2.js:3661 |
| 65.7 | 65.7 | Element | raphael-2.1.2.js:6356 |
| 0 | 63.7 | (unresolved function) | |
| 56.6 | 56.6 | call | |
| 0 | 55.6 | unhighlight | gate.js:194 |
| 3.0 | 54.6 | highlight | step.js:160 |
| 2.0 | 53.6 | (anonymous function) | step.js:147 |
| 51.5 | 51.5 | get style | |
| 47.5 | 47.5 | repush | raphael-2.1.2.js:1204 |
| 46.5 | 46.5 | getPropertyValue | |
| 3.0 | 40.4 | pathClone | raphael-2.1.2.js:2020 |
| 1.0 | 40.4 | (anonymous function) | gate.js:101 |
| 0 | 40.4 | highlight | gate.js:182 |
| 1.0 | 32.3 | (anonymous function) | gate.js:106 |
| 1.0 | 30.3 | (anonymous function) | gate.js:96 |
| 1.0 | 29.3 | selectGate | toolbox.js:33 |
| 1.0 | 28.3 | R._engine.circle | raphael-2.1.2.js:6910 |

Continued on next page

**Table 3 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 22.2 | 22.2 | (anonymous function) | raphael-2.1.2.js:1483 |
| 2.0 | 22.2 | R._engine.rect | raphael-2.1.2.js:6919 |
| 20.2 | 21.2 | paperproto.set | raphael-2.1.2.js:3813 |
| 0 | 16.2 | (anonymous function) | qubit.js:65 |
| 0 | 16.2 | changeStartState | qubit.js:90 |
| 7.1 | 14.2 | R.format | raphael-2.1.2.js:5643 |
| 13.1 | 13.1 | setproto.push | raphael-2.1.2.js:5185 |
| 0 | 9.1 | (anonymous function) | step.js:153 |
| 7.1 | 7.1 | onSubtreeModified | inject_actions.js:11 |
| 3.0 | 5.1 | (anonymous function) | raphael-2.1.2.js:1480 |
| 4.0 | 4.0 | set innerHTML | |
| 4.0 | 4.0 | get firstChild | |
| 0 | 4.0 | updateHelpText | main.js:11 |
| 0 | 4.0 | HadamardGate | gate.js:217 |
| 3.0 | 3.0 | (anonymous function) | raphael-2.1.2.js:6304 |
| 3.0 | 3.0 | slice | |
| 3.0 | 3.0 | (anonymous function) | raphael-2.1.2.js:1548 |
| 3.0 | 3.0 | getElementsByTagName | |
| 0 | 3.0 | (anonymous function) | step.js:147 |
| 2.0 | 2.0 | createTextNode | |
| 2.0 | 2.0 | (anonymous function) | raphael-2.1.2.js:1483 |
| 0 | 2.0 | e.zeros | math.min.js:29 |
| 1.0 | 2.0 | paperproto.setSize | raphael-2.1.2.js:3862 |
| 1.0 | 1.0 | push | |
| 1.0 | 1.0 | f | math.min.js:28 |
| 1.0 | 1.0 | f | math.min.js:27 |
| 1.0 | 1.0 | n.resize | math.min.js:31 |
| 1.0 | 1.0 | getComputedStyle | |
| 1.0 | 1.0 | get defaultView | |
| 0 | 1.0 | R._engine.setSize | raphael-2.1.2.js:6964 |
| 1.0 | 1.0 | get y | |
| 0 | 1.0 | unhighlight | qubit.js:83 |
| 0 | 1.0 | Gate | gate.js:8 |
| 0 | 1.0 | t | math.min.js:27 |
| 0 | 1.0 | highlight | qubit.js:76 |
| 0 | 1.0 | (anonymous function) | qubit.js:53 |
| 0 | 1.0 | Step | step.js:4 |
| 0 | 1.0 | (anonymous function) | qubit.js:59 |
| 0 | 1.0 | y | math.min.js:28 |

Figure 30: Performance Test Run 1-2 Results

Table 4: Performance Test Run 1-3 Results.

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 52481.2 | 52481.2 | (idle) | |
| 6.1 | 20163.4 | updateCircuit | circuitry.js:35 |
| 9.1 | 19875.0 | f | raphael-2.1.2.js:3098 |
| 9.1 | 19670.6 | drawCircuit | circuitry.js:19 |
| 37.4 | 19549.2 | draw | qubit.js:34 |
| 0 | 19489.5 | setGate | step.js:20 |
| 118.4 | 17865.4 | draw | step.js:124 |
| 2.0 | 13520.5 | (anonymous function) | step.js:141 |
| 123.4 | 12470.2 | (anonymous function) | raphael-2.1.2.js:5233 |
| 249.9 | 12424.6 | setproto.forEach | raphael-2.1.2.js:5223 |
| 946.1 | 9829.2 | R.(anonymous function). elproto.(anonymous function) | raphael-2.1.2.js:3431 |
| 4698.1 | 8474.3 | e | raphael-2.1.2.js:3097 |
| 1646.3 | 8455.1 | (anonymous function) | raphael-2.1.2.js:5235 |
| 8393.4 | 8393.4 | (garbage collector) | |
| 0 | 5971.0 | (anonymous function) | step.js:141 |
| 1110.0 | 4321.7 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 3776.3 | 3776.3 | addEventListener | |
| 483.7 | 3756.0 | (anonymous function) | raphael-2.1.2.js:5235 |
| 2603.5 | 2603.5 | (program) | |
| 94.1 | 2190.7 | paperproto.path | raphael-2.1.2.js:3746 |
| 51.6 | 2152.2 | drawGate | gate.js:113 |
| 84.0 | 2060.1 | R._engine.path | raphael-2.1.2.js:6444 |
| 21.2 | 1919.5 | paperproto.text | raphael-2.1.2.js:3791 |
| 500.9 | 1841.6 | $ | raphael-2.1.2.js:5792 |
| 1683.7 | 1686.8 | setAttribute | |
| 135.6 | 1563.3 | elproto.attr | raphael-2.1.2.js:6750 |
| 86.0 | 1422.7 | paperproto.rect | raphael-2.1.2.js:3687 |
| 44.5 | 1365.0 | tuneText | raphael-2.1.2.js:6319 |
| 4.0 | 1116.1 | elproto._getBBox | raphael-2.1.2.js:6653 |
| 1111.0 | 1111.0 | getBBox | |
| 31.4 | 736.6 | R._pathToAbsolute | raphael-2.1.2.js:2104 |
| 0 | 587.9 | updateSteps | circuitry.js:52 |
| 0 | 587.9 | incrementStepCount | main.js:72 |
| 0 | 587.9 | onclick | index.html:27 |
| 243.9 | 520.1 | eve | raphael-2.1.2.js:54 |
| 513.0 | 513.0 | R.is | raphael-2.1.2.js:780 |
| 116.4 | 491.8 | R.parsePathString | raphael-2.1.2.js:1465 |
| 45.5 | 485.7 | R.clear | raphael-2.1.2.js:7088 |

Continued on next page

**Table 4 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 424.0 | 424.0 | removeChild | |
| 9.1 | 413.9 | R._engine.text | raphael-2.1.2.js:6947 |
| 397.7 | 402.7 | appendChild | |
| 351.1 | 351.1 | (anonymous function) | raphael-2.1.2.js:1548 |
| 10.1 | 328.9 | evaluateCircuit | circuitry.js:65 |
| 0 | 328.9 | onclick | index.html:25 |
| 2.0 | 307.6 | (anonymous function) | circuitry.js:101 |
| 276.2 | 276.2 | eve.listeners | raphael-2.1.2.js:129 |
| 181.1 | 212.5 | (anonymous function) | raphael-2.1.2.js:1480 |
| 208.4 | 208.4 | clone | raphael-2.1.2.js:795 |
| 22.3 | 208.4 | setUpMatrix | circuitry.js:154 |
| 135.6 | 179.1 | newf | raphael-2.1.2.js:1210 |
| 51.6 | 149.8 | paths | raphael-2.1.2.js:1539 |
| 2.0 | 127.5 | drawConnections | gate.js:208 |
| 0 | 125.5 | (anonymous function) | gate.js:210 |
| 38.5 | 113.3 | e.index | math.min.js:29 |
| 106.2 | 106.2 | createElementNS | |
| 101.2 | 101.2 | apply | |
| 98.2 | 98.2 | setTimeout | |
| 1.0 | 96.1 | onclick | index.html:26 |
| 1.0 | 96.1 | unhighlight | step.js:167 |
| 0 | 95.1 | updateQubits | circuitry.js:41 |
| 0 | 95.1 | incrementQubitCount | main.js:45 |
| 13.2 | 94.1 | y | math.min.js:28 |
| 90.1 | 90.1 | get scrollTop | |
| 89.0 | 89.0 | R._path2string | raphael-2.1.2.js:1201 |
| 3.0 | 84.0 | paperproto.circle | raphael-2.1.2.js:3661 |
| 0 | 83.0 | (anonymous function) | step.js:153 |
| 2.0 | 78.9 | getEventPosition | raphael-2.1.2.js:3086 |
| 1.0 | 78.9 | highlight | step.js:160 |
| 23.3 | 75.9 | n | math.min.js:28 |
| 75.9 | 75.9 | Element | raphael-2.1.2.js:6356 |
| 5.1 | 73.9 | R._engine.rect | raphael-2.1.2.js:6919 |
| 71.8 | 73.9 | t | math.min.js:27 |
| 1.0 | 71.8 | (anonymous function) | step.js:147 |
| 0 | 64.8 | (unresolved function) | |
| 62.7 | 62.7 | get style | |
| 15.2 | 61.7 | t.subset | math.min.js:27 |
| 59.7 | 59.7 | call | |
| 2.0 | 51.6 | unhighlight | gate.js:194 |
| 5.1 | 48.6 | pathClone | raphael-2.1.2.js:2020 |

Continued on next page

**Table 4 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 43.5 | 43.5 | repush | raphael-2.1.2.js:1204 |
| 4.0 | 43.5 | a | math.min.js:27 |
| 29.3 | 41.5 | g | math.min.js:28 |
| 37.4 | 37.4 | getPropertyValue | |
| 2.0 | 35.4 | R._engine.circle | raphael-2.1.2.js:6910 |
| 1.0 | 32.4 | highlight | gate.js:182 |
| 0 | 32.4 | (anonymous function) | gate.js:101 |
| 31.4 | 31.4 | (anonymous function) | raphael-2.1.2.js:1483 |
| 0 | 29.3 | (anonymous function) | gate.js:106 |
| 1.0 | 23.3 | (anonymous function) | gate.js:96 |
| 0 | 22.3 | selectGate | toolbox.js:33 |
| 0 | 20.2 | e.zeros | math.min.js:29 |
| 10.1 | 17.2 | t.set | math.min.js:27 |
| 4.0 | 17.2 | n.size | math.min.js:31 |
| 10.1 | 16.2 | n.resize | math.min.js:31 |
| 0 | 15.2 | changeStartState | qubit.js:90 |
| 0 | 15.2 | setUpControls | circuitry.js:138 |
| 0 | 15.2 | (anonymous function) | qubit.js:65 |
| 13.2 | 15.2 | paperproto.set | raphael-2.1.2.js:3813 |
| 15.2 | 15.2 | n.isNumber | math.min.js:31 |
| 0 | 13.2 | (anonymous function) | step.js:153 |
| 0 | 12.1 | n.validate | math.min.js:31 |
| 2.0 | 12.1 | i | math.min.js:31 |
| 11.1 | 11.1 | i | math.min.js:31 |
| 6.1 | 11.1 | R.format | raphael-2.1.2.js:5643 |
| 11.1 | 11.1 | setproto.push | raphael-2.1.2.js:5185 |
| 8.1 | 10.1 | t.min | math.min.js:27 |
| 10.1 | 10.1 | r | math.min.js:27 |
| 8.1 | 8.1 | onSubtreeModified | inject_actions.js:11 |
| 0 | 8.1 | (anonymous function) | step.js:147 |
| 7.1 | 7.1 | l | math.min.js:27 |
| 7.1 | 7.1 | (anonymous function) | raphael-2.1.2.js:1480 |
| 4.0 | 6.1 | t.isScalar | math.min.js:27 |
| 3.0 | 6.1 | o | math.min.js:31 |
| 0 | 5.1 | t | math.min.js:27 |
| 0 | 4.0 | HadamardGate | gate.js:217 |
| 3.0 | 4.0 | f | math.min.js:28 |
| 0 | 4.0 | PauliXGate | gate.js:226 |
| 3.0 | 3.0 | n.argsToArray | math.min.js:28 |
| 3.0 | 3.0 | get documentElement | |
| 1.0 | 3.0 | h | math.min.js:28 |

Continued on next page

**Table 4 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 3.0 | 3.0 | get y | |
| 3.0 | 3.0 | f | math.min.js:27 |
| 3.0 | 3.0 | createTextNode | |
| 0 | 3.0 | Gate | gate.js:8 |
| 0 | 2.0 | e.eye | math.min.js:29 |
| 0 | 2.0 | i | math.min.js:27 |
| 1.0 | 2.0 | i | math.min.js:27 |
| 2.0 | 2.0 | t.get | math.min.js:27 |
| 2.0 | 2.0 | set innerHTML | |
| 0 | 2.0 | u | math.min.js:27 |
| 2.0 | 2.0 | t.min | math.min.js:27 |
| 0 | 2.0 | e.matrix | math.min.js:29 |
| 0 | 1.0 | paperproto.setSize | raphael-2.1.2.js:3862 |
| 0 | 1.0 | e.min | math.min.js:29 |
| 0 | 1.0 | highlight | qubit.js:76 |
| 0 | 1.0 | t | math.min.js:29 |
| 1.0 | 1.0 | c | math.min.js:28 |
| 1.0 | 1.0 | splice | |
| 0 | 1.0 | (anonymous function) | qubit.js:53 |
| 0 | 1.0 | t.resize | math.min.js:27 |
| 1.0 | 1.0 | get scrollLeft | |
| 0 | 1.0 | R._engine.setSize | raphael-2.1.2.js:6964 |
| 1.0 | 1.0 | get body | |
| 0 | 1.0 | updateEvaluationText | main.js:19 |
| 1.0 | 1.0 | pad | circuitry.js:131 |
| 1.0 | 1.0 | (anonymous function) | math.min.js:29 |
| 1.0 | 1.0 | getElementsByTagName | |
| 1.0 | 1.0 | get firstChild | |
| 0 | 1.0 | updateHelpText | main.js:11 |
| 1.0 | 1.0 | Set | raphael-2.1.2.js:5164 |
| 1.0 | 1.0 | (anonymous function) | raphael-2.1.2.js:6304 |
| 1.0 | 1.0 | R._engine.rect | raphael-2.1.2.js:6919 |
| 1.0 | 1.0 | getComputedStyle | |

Figure 31: Performance Test Run 1-3 Results

Table 5: Performance Test Run 2-1 Results.

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| | | | Continued on next page |

**Table 5 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 70099.9 | 70099.9 | (idle) | |
| 2967.8 | 2967.8 | (program) | |
| 6.1 | 714.7 | f | raphael-2.1.2.js:3098 |
| 28.3 | 429.0 | elproto.attr | raphael-2.1.2.js:6750 |
| 265.5 | 385.6 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 0 | 336.1 | evaluate | controller.js:44 |
| 0 | 335.1 | evaluateCircuit | evaluator.js:1 |
| 5.0 | 334.1 | (anonymous function) | evaluator.js:4 |
| 24.2 | 254.4 | setUpMatrix | evaluator.js:49 |
| 1.0 | 203.9 | drawStep | painter.js:204 |
| 1.0 | 166.6 | (anonymous function) | painter.js:239 |
| 1.0 | 165.5 | unhighlight | step.js:136 |
| 0 | 162.5 | (anonymous function) | painter.js:226 |
| 2.0 | 126.2 | (anonymous function) | painter.js:233 |
| 46.4 | 124.2 | e.exports.e.index | math.min.js:29 |
| 0 | 124.2 | highlight | step.js:129 |
| 116.1 | 116.1 | get scrollTop | get scrollTop |
| 1.0 | 90.9 | drawGate | painter.js:71 |
| 21.2 | 89.8 | eve | raphael-2.1.2.js:54 |
| 26.2 | 87.8 | t.subset | math.min.js:27 |
| 72.7 | 75.7 | t | math.min.js:27 |
| 20.2 | 71.7 | y | math.min.js:28 |
| 67.6 | 67.6 | eve.listeners | raphael-2.1.2.js:129 |
| 22.2 | 66.6 | n | math.min.js:28 |
| 0 | 65.6 | paperproto.path | raphael-2.1.2.js:3746 |
| 1.0 | 64.6 | R._engine.path | raphael-2.1.2.js:6444 |
| 1.0 | 64.6 | getEventPosition | raphael-2.1.2.js:3086 |
| 2.0 | 62.6 | unhighlight | tool.js:26 |
| 61.6 | 61.6 | setAttribute | |
| 0 | 55.5 | setproto.forEach | raphael-2.1.2.js:5223 |
| 6.1 | 54.5 | a | math.min.js:27 |
| 20.2 | 52.5 | $ | raphael-2.1.2.js:5792 |
| 0 | 49.5 | highlight | tool.js:16 |
| 0 | 48.5 | drawQubit | painter.js:153 |
| 0 | 47.4 | paperproto.text | raphael-2.1.2.js:3791 |
| 0 | 47.4 | R._engine.text | raphael-2.1.2.js:6947 |
| 0 | 46.4 | (anonymous function) | painter.js:55 |
| 30.3 | 46.4 | elproto.remove | raphael-2.1.2.js:6632 |
| 0 | 38.4 | (anonymous function) | painter.js:60 |
| 3.0 | 37.3 | paperproto.rect | raphael-2.1.2.js:3687 |
| 0 | 36.3 | (anonymous function) | painter.js:206 |

Continued on next page

**Table 5 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 0 | 34.3 | (anonymous function) | painter.js:194 |
| 2.0 | 33.3 | tuneText | raphael-2.1.2.js:6319 |
| 32.3 | 32.3 | r | math.min.js:27 |
| 2.0 | 32.3 | incrementStepCount | controller.js:145 |
| 31.3 | 31.3 | (garbage collector) | |
| 0 | 31.3 | setUpControls | evaluator.js:29 |
| 0 | 30.3 | (anonymous function) | controller.js:87 |
| 17.2 | 30.3 | t.set | math.min.js:27 |
| 1.0 | 29.3 | incrementQubitCount | controller.js:118 |
| 0 | 28.3 | (anonymous function) | controller.js:69 |
| 0 | 28.3 | selectTool | toolbox.js:30 |
| 0 | 28.3 | (anonymous function) | painter.js:49 |
| 0 | 28.3 | updateQubitCount | controller.js:65 |
| 0 | 24.2 | elproto._getBBox | raphael-2.1.2.js:6653 |
| 24.2 | 24.2 | getBBox | |
| 0 | 23.2 | setGate | step.js:9 |
| 0 | 20.2 | (anonymous function) | painter.js:189 |
| 9.1 | 19.2 | g | math.min.js:28 |
| 15.1 | 19.2 | R.is | raphael-2.1.2.js:780 |
| 16.2 | 16.2 | appendChild | |
| 0 | 16.2 | e.exports.e.zeros | math.min.js:29 |
| 0 | 15.1 | R._pathToAbsolute | raphael-2.1.2.js:2104 |
| 15.1 | 15.1 | n.isNumber | math.min.js:31 |
| 9.1 | 12.1 | newf | raphael-2.1.2.js:1210 |
| 0 | 11.1 | n.resize | math.min.js:31 |
| 1.0 | 11.1 | o | math.min.js:31 |
| 3.0 | 10.1 | R.parsePathString | raphael-2.1.2.js:1465 |
| 4.0 | 9.1 | (anonymous function) | raphael-2.1.2.js:5235 |
| 1.0 | 9.1 | (anonymous function) | raphael-2.1.2.js:5233 |
| 0 | 9.1 | (anonymous function) | painter.js:73 |
| 9.1 | 9.1 | removeChild | |
| 0 | 9.1 | t.get | math.min.js:27 |
| 0 | 8.1 | t | math.min.js:27 |
| 0 | 8.1 | R._engine.rect | raphael-2.1.2.js:6919 |
| 8.1 | 8.1 | (anonymous function) | raphael-2.1.2.js:1548 |
| 6.1 | 7.1 | t.isScalar | math.min.js:27 |
| 3.0 | 7.1 | n.size | math.min.js:31 |
| 0 | 7.1 | u | math.min.js:27 |
| 7.1 | 7.1 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 0 | 6.1 | (anonymous function) | painter.js:141 |
| 6.1 | 6.1 | l | math.min.js:27 |

Continued on next page

**Table 5 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 6.1 | 6.1 | t.min | math.min.js:27 |
| 0 | 6.1 | updateStepCount | controller.js:84 |
| 6.1 | 6.1 | get style | |
| 5.0 | 6.1 | (anonymous function) | raphael-2.1.2.js:1480 |
| 4.0 | 4.0 | i | math.min.js:31 |
| 1.0 | 4.0 | R._engine.circle | raphael-2.1.2.js:6910 |
| 2.0 | 4.0 | R.(anonymous function). elproto.(anonymous function) | raphael-2.1.2.js:3431 |
| 0 | 4.0 | paperproto.circle | raphael-2.1.2.js:3661 |
| 3.0 | 3.0 | repush | raphael-2.1.2.js:1204 |
| 0 | 3.0 | (anonymous function) | raphael-2.1.2.js:7115 |
| 3.0 | 3.0 | R._path2string | raphael-2.1.2.js:1201 |
| 2.0 | 3.0 | i | math.min.js:27 |
| 2.0 | 3.0 | paths | raphael-2.1.2.js:1539 |
| 0 | 3.0 | (anonymous function) | raphael-2.1.2.js:7117 |
| 1.0 | 3.0 | pathClone | raphael-2.1.2.js:2020 |
| 3.0 | 3.0 | eve.off.eve.unbind | raphael-2.1.2.js:288 |
| 3.0 | 3.0 | R._removedFactory | raphael-2.1.2.js:1946 |
| 0 | 3.0 | (anonymous function) | math.min.js:27 |
| 3.0 | 3.0 | f | math.min.js:28 |
| 3.0 | 3.0 | f | math.min.js:27 |
| 2.0 | 2.0 | f | raphael-2.1.2.js:3098 |
| 2.0 | 2.0 | e | raphael-2.1.2.js:3097 |
| 2.0 | 2.0 | clone | raphael-2.1.2.js:795 |
| 2.0 | 2.0 | createElementNS | |
| 1.0 | 1.0 | getComputedStyle | |
| 1.0 | 1.0 | drawQubit | painter.js:153 |
| 1.0 | 1.0 | Element | raphael-2.1.2.js:6356 |
| 1.0 | 1.0 | getElementById | |
| 0 | 1.0 | h | math.min.js:28 |
| 1.0 | 1.0 | repush | raphael-2.1.2.js:1204 |
| 1.0 | 1.0 | call | |
| 0 | 1.0 | e.exports.e.eye | math.min.js:29 |
| 1.0 | 1.0 | setTimeout | |
| 1.0 | 1.0 | (anonymous function) | circuit.js:79 |
| 1.0 | 1.0 | (anonymous function) | raphael-2.1.2.js:1483 |
| 1.0 | 1.0 | get documentElement | |
| 1.0 | 1.0 | apply | |
| 1.0 | 1.0 | Element | raphael-2.1.2.js:6356 |
| 1.0 | 1.0 | get x | |
| 1.0 | 1.0 | t.resize | math.min.js:27 |

Continued on next page

**Table 5 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 0 | 1.0 | updateHelpText | controller.js:36 |
| 1.0 | 1.0 | slice | |
| 0 | 1.0 | getInitialAmplitudes | circuit.js:70 |
| 0 | 1.0 | (unresolved function) | |
| 1.0 | 1.0 | concat | |
| 0 | 1.0 | n.validate | math.min.js:31 |
| 1.0 | 1.0 | t | math.min.js:27 |
| 1.0 | 1.0 | (anonymous function) | math.min.js:29 |
| 1.0 | 1.0 | setproto.push | raphael-2.1.2.js:5185 |
| 0 | 1.0 | highlight | qubit.js:32 |
| 1.0 | 1.0 | get tagName | |
| 0 | 1.0 | (anonymous function) | painter.js:177 |
| 0 | 1.0 | (anonymous function) | painter.js:155 |

Figure 32: Performance Test Run 2-1 Results

Table 6: Performance Test Run 2-2 Results.

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 71110.7 | 71110.7 | (idle) | |
| 2845.4 | 2845.4 | (program) | |
| 5.0 | 688.9 | f | raphael-2.1.2.js:3098 |
| 21.2 | 399.4 | elproto.attr | raphael-2.1.2.js:6750 |
| 245.1 | 369.2 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 0 | 287.5 | evaluate | controller.js:44 |
| 0 | 286.5 | evaluateCircuit | evaluator.js:1 |
| 2.0 | 285.4 | (anonymous function) | evaluator.js:4 |
| 16.1 | 213.8 | setUpMatrix | evaluator.js:49 |
| 1.0 | 200.7 | drawStep | painter.js:204 |
| 2.0 | 161.4 | (anonymous function) | painter.js:226 |
| 1.0 | 153.3 | (anonymous function) | painter.js:239 |
| 2.0 | 152.3 | unhighlight | step.js:136 |
| 114.0 | 114.0 | get scrollTop | |
| 1.0 | 108.9 | (anonymous function) | painter.js:233 |
| 1.0 | 107.9 | highlight | step.js:129 |
| 37.3 | 101.9 | e.index | math.min.js:29 |
| 28.2 | 87.8 | eve | raphael-2.1.2.js:54 |
| 5.0 | 87.8 | drawGate | painter.js:71 |
| 16.1 | 72.6 | t.subset | math.min.js:27 |
| 11.1 | 68.6 | y | math.min.js:28 |

Continued on next page

**Table 6 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 1.0 | 64.6 | unhighlight | tool.js:26 |
| 21.2 | 62.5 | n | math.min.js:28 |
| 58.5 | 59.5 | t | math.min.js:27 |
| 1.0 | 59.5 | getEventPosition | raphael-2.1.2.js:3086 |
| 59.5 | 59.5 | eve.listeners | raphael-2.1.2.js:129 |
| 3.0 | 58.5 | paperproto.path | raphael-2.1.2.js:3746 |
| 57.5 | 57.5 | setAttribute | |
| 0 | 55.5 | setproto.forEach | raphael-2.1.2.js:5223 |
| 3.0 | 54.5 | R._engine.path | raphael-2.1.2.js:6444 |
| 3.0 | 54.5 | a | math.min.js:27 |
| 1.0 | 53.5 | highlight | tool.js:16 |
| 1.0 | 49.4 | (anonymous function) | painter.js:55 |
| 47.4 | 47.4 | (garbage collector) | |
| 26.2 | 45.4 | elproto.remove | raphael-2.1.2.js:6632 |
| 2.0 | 45.4 | drawQubit | painter.js:153 |
| 14.1 | 45.4 | $ | raphael-2.1.2.js:5792 |
| 0 | 44.4 | (anonymous function) | painter.js:60 |
| 0 | 43.4 | paperproto.text | raphael-2.1.2.js:3791 |
| 1.0 | 43.4 | R._engine.text | raphael-2.1.2.js:6947 |
| 0 | 37.3 | (anonymous function) | painter.js:206 |
| 5.0 | 36.3 | paperproto.rect | raphael-2.1.2.js:3687 |
| 0 | 34.3 | (anonymous function) | painter.js:194 |
| 0 | 33.3 | updateStepCount | controller.js:84 |
| 1.0 | 32.3 | incrementStepCount | controller.js:145 |
| 30.3 | 32.3 | r | math.min.js:27 |
| 15.1 | 31.3 | t.set | math.min.js:27 |
| 0 | 30.3 | (anonymous function) | controller.js:87 |
| 4.0 | 30.3 | tuneText | raphael-2.1.2.js:6319 |
| 1.0 | 27.2 | (anonymous function) | painter.js:49 |
| 0 | 26.2 | selectTool | toolbox.js:30 |
| 0 | 25.2 | (anonymous function) | controller.js:69 |
| 0 | 25.2 | incrementQubitCount | controller.js:118 |
| 0 | 25.2 | updateQubitCount | controller.js:65 |
| 1.0 | 23.2 | setGate | step.js:9 |
| 12.1 | 23.2 | g | math.min.js:28 |
| 0 | 23.2 | e.zeros | math.min.js:29 |
| 21.2 | 21.2 | n.isNumber | math.min.js:31 |
| 0 | 21.2 | elproto._getBBox | raphael-2.1.2.js:6653 |
| 21.2 | 21.2 | getBBox | |
| 0 | 20.2 | (anonymous function) | painter.js:189 |
| 1.0 | 19.2 | R._pathToAbsolute | raphael-2.1.2.js:2104 |

Continued on next page

**Table 6 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 0 | 17.1 | n.resize | math.min.js:31 |
| 6.1 | 17.1 | newf | raphael-2.1.2.js:1210 |
| 2.0 | 16.1 | o | math.min.js:31 |
| 3.0 | 13.1 | R.parsePathString | raphael-2.1.2.js:1465 |
| 12.1 | 12.1 | repush | raphael-2.1.2.js:1204 |
| 0 | 12.1 | setUpControls | evaluator.js:29 |
| 11.1 | 11.1 | R.is | raphael-2.1.2.js:780 |
| 11.1 | 11.1 | appendChild | |
| 2.0 | 10.1 | R._engine.rect | raphael-2.1.2.js:6919 |
| 3.0 | 9.1 | (anonymous function) | raphael-2.1.2.js:5235 |
| 9.1 | 9.1 | eve.off.eve.unbind | raphael-2.1.2.js:288 |
| 0 | 9.1 | (anonymous function) | raphael-2.1.2.js:5233 |
| 2.0 | 8.1 | t | math.min.js:27 |
| 7.1 | 7.1 | removeChild | |
| 7.1 | 7.1 | t.min | math.min.js:27 |
| 6.1 | 7.1 | t.isScalar | math.min.js:27 |
| 1.0 | 7.1 | (anonymous function) | painter.js:141 |
| 5.0 | 7.1 | l | math.min.js:27 |
| 7.1 | 7.1 | (anonymous function) | raphael-2.1.2.js:1548 |
| 0 | 7.1 | (anonymous function) | painter.js:73 |
| 2.0 | 6.1 | (anonymous function) | math.min.js:27 |
| 6.1 | 6.1 | createElementNS | |
| 0 | 6.1 | paperproto.circle | raphael-2.1.2.js:3661 |
| 2.0 | 6.1 | R._engine.circle | raphael-2.1.2.js:6910 |
| 2.0 | 6.1 | R.(anonymous function).<br>elproto.(anonymous function) | raphael-2.1.2.js:3431 |
| 0 | 5.0 | (anonymous function) | raphael-2.1.2.js:7115 |
| 5.0 | 5.0 | i | math.min.js:31 |
| 1.0 | 5.0 | (anonymous function) | raphael-2.1.2.js:7117 |
| 5.0 | 5.0 | clone | raphael-2.1.2.js:795 |
| 0 | 5.0 | pathClone | raphael-2.1.2.js:2020 |
| 1.0 | 5.0 | paths | raphael-2.1.2.js:1539 |
| 4.0 | 5.0 | (anonymous function) | raphael-2.1.2.js:1480 |
| 0 | 5.0 | n.size | math.min.js:31 |
| 5.0 | 5.0 | get style | |
| 1.0 | 4.0 | e | raphael-2.1.2.js:3097 |
| 4.0 | 4.0 | f | math.min.js:27 |
| 4.0 | 4.0 | setTimeout | |
| 4.0 | 4.0 | R._path2string | raphael-2.1.2.js:1201 |
| 1.0 | 4.0 | u | math.min.js:27 |
| 0 | 3.0 | (anonymous function) | painter.js:183 |

Continued on next page

**Table 6 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 3.0 | 3.0 | addEventListener | |
| 0 | 3.0 | unhighlight | qubit.js:39 |
| 3.0 | 3.0 | set innerHTML | |
| 0 | 3.0 | updateHelpText | controller.js:36 |
| 3.0 | 3.0 | apply | |
| 2.0 | 2.0 | call | |
| 0 | 2.0 | decrementStepCount | controller.js:124 |
| 0 | 2.0 | (anonymous function) | controller.js:90 |
| 2.0 | 2.0 | R._removedFactory | raphael-2.1.2.js:1946 |
| 0 | 2.0 | (anonymous function) | controller.js:91 |
| 0 | 2.0 | t.get | math.min.js:27 |
| 2.0 | 2.0 | t.size | math.min.js:27 |
| 0 | 2.0 | (unresolved function) | |
| 2.0 | 2.0 | (anonymous function) | math.min.js:29 |
| 2.0 | 2.0 | paperproto.set | raphael-2.1.2.js:3813 |
| 1.0 | 1.0 | get parentNode | |
| 0 | 1.0 | getMatrix | matrix-expert.js:57 |
| 0 | 1.0 | e.eye | math.min.js:29 |
| 1.0 | 1.0 | (anonymous function) | raphael-2.1.2.js:1483 |
| 0 | 1.0 | R._engine.setSize | raphael-2.1.2.js:6964 |
| 0 | 1.0 | (anonymous function) | painter.js:177 |
| 1.0 | 1.0 | f | math.min.js:28 |
| 0 | 1.0 | t.resize | math.min.js:27 |
| 0 | 1.0 | paperproto.setSize | raphael-2.1.2.js:3862 |
| 0 | 1.0 | e.clone | math.min.js:29 |
| 1.0 | 1.0 | getPropertyValue | |
| 1.0 | 1.0 | toString | |
| 1.0 | 1.0 | Element | raphael-2.1.2.js:6356 |
| 0 | 1.0 | getInitialAmplitudes | circuit.js:70 |
| 1.0 | 1.0 | (anonymous function) | math.min.js:31 |
| 0 | 1.0 | highlight | qubit.js:32 |

Figure 33: Performance Test Run 2-2 Results

Table 7: Performance Test Run 2-3 Results.

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 65788.8 | 65788.8 | (idle) | |
| 2930.3 | 2930.3 | (program) | |
| 6.1 | 709.6 | f | raphael-2.1.2.js:3098 |

Continued on next page

**Table 7 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 24.2 | 421.9 | elproto.attr | raphael-2.1.2.js:6750 |
| 270.5 | 386.6 | setFillAndStroke | raphael-2.1.2.js:6065 |
| 1.0 | 306.9 | evaluate | controller.js:44 |
| 3.0 | 305.8 | (anonymous function) | evaluator.js:4 |
| 0 | 305.8 | evaluateCircuit | evaluator.js:1 |
| 18.2 | 230.1 | setUpMatrix | evaluator.js:49 |
| 4.0 | 210.0 | drawStep | painter.js:204 |
| 0 | 166.5 | (anonymous function) | painter.js:226 |
| 2.0 | 150.4 | unhighlight | step.js:136 |
| 0 | 150.4 | (anonymous function) | painter.js:239 |
| 47.4 | 116.1 | e.index | math.min.js:29 |
| 115.1 | 115.1 | get scrollTop | |
| 1.0 | 107.0 | (anonymous function) | painter.js:233 |
| 2.0 | 106.0 | highlight | step.js:129 |
| 0 | 84.8 | drawGate | painter.js:71 |
| 22.2 | 82.8 | eve | raphael-2.1.2.js:54 |
| 11.1 | 73.7 | t.subset | math.min.js:27 |
| 1.0 | 70.7 | getEventPosition | raphael-2.1.2.js:3086 |
| 1.0 | 68.6 | unhighlight | tool.js:26 |
| 13.1 | 66.6 | y | math.min.js:28 |
| 63.6 | 66.6 | t | math.min.js:27 |
| 21.2 | 61.6 | n | math.min.js:28 |
| 60.6 | 60.6 | eve.listeners | raphael-2.1.2.js:129 |
| 0 | 59.6 | highlight | tool.js:16 |
| 59.6 | 59.6 | setAttribute | |
| 0 | 57.5 | (anonymous function) | painter.js:55 |
| 0 | 57.5 | setproto.forEach | raphael-2.1.2.js:5223 |
| 3.0 | 55.5 | a | math.min.js:27 |
| 4.0 | 54.5 | paperproto.path | raphael-2.1.2.js:3746 |
| 1.0 | 49.5 | R._engine.path | raphael-2.1.2.js:6444 |
| 0 | 49.5 | (anonymous function) | painter.js:60 |
| 3.0 | 48.5 | R._engine.text | raphael-2.1.2.js:6947 |
| 0 | 48.5 | paperproto.text | raphael-2.1.2.js:3791 |
| 27.3 | 46.4 | elproto.remove | raphael-2.1.2.js:6632 |
| 13.1 | 45.4 | $ | raphael-2.1.2.js:5792 |
| 0 | 42.4 | drawQubit | painter.js:153 |
| 40.4 | 40.4 | (garbage collector) | |
| 0 | 38.4 | (anonymous function) | painter.js:206 |
| 3.0 | 38.4 | paperproto.rect | raphael-2.1.2.js:3687 |
| 37.3 | 37.3 | r | math.min.js:27 |
| 0 | 33.3 | updateStepCount | controller.js:84 |

Continued on next page

**Table 7 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 4.0 | 32.3 | tuneText | raphael-2.1.2.js:6319 |
| 0 | 32.3 | (anonymous function) | painter.js:194 |
| 14.1 | 32.3 | t.set | math.min.js:27 |
| 1.0 | 32.3 | incrementStepCount | controller.js:145 |
| 0 | 30.3 | (anonymous function) | controller.js:87 |
| 26.2 | 26.2 | R.is | raphael-2.1.2.js:780 |
| 15.1 | 25.2 | g | math.min.js:28 |
| 1.0 | 24.2 | elproto._getBBox | raphael-2.1.2.js:6653 |
| 0 | 24.2 | updateQubitCount | controller.js:65 |
| 0 | 24.2 | incrementQubitCount | controller.js:118 |
| 23.2 | 23.2 | getBBox | |
| 0 | 23.2 | (anonymous function) | controller.js:69 |
| 0 | 22.2 | selectTool | toolbox.js:30 |
| 0 | 22.2 | (anonymous function) | painter.js:49 |
| 1.0 | 21.2 | e.zeros | math.min.js:29 |
| 0 | 19.2 | setGate | step.js:9 |
| 0 | 19.2 | (anonymous function) | painter.js:189 |
| 2.0 | 18.2 | o | math.min.js:31 |
| 0 | 18.2 | n.resize | math.min.js:31 |
| 0 | 17.2 | R._pathToAbsolute | raphael-2.1.2.js:2104 |
| 16.2 | 16.2 | n.isNumber | math.min.js:31 |
| 7.1 | 12.1 | newf | raphael-2.1.2.js:1210 |
| 0 | 12.1 | setUpControls | evaluator.js:29 |
| 2.0 | 11.1 | (anonymous function) | raphael-2.1.2.js:5235 |
| 0 | 11.1 | (anonymous function) | raphael-2.1.2.js:5233 |
| 3.0 | 11.1 | R.parsePathString | raphael-2.1.2.js:1465 |
| 10.1 | 10.1 | (anonymous function) | raphael-2.1.2.js:1548 |
| 10.1 | 10.1 | removeChild | |
| 10.1 | 10.1 | appendChild | |
| 2.0 | 9.1 | R.(anonymous function).<br>elproto.(anonymous function) | raphael-2.1.2.js:3431 |
| 0 | 8.1 | pathClone | raphael-2.1.2.js:2020 |
| 8.1 | 8.1 | clone | raphael-2.1.2.js:795 |
| 1.0 | 8.1 | unhighlight | qubit.js:39 |
| 4.0 | 8.1 | t.isScalar | math.min.js:27 |
| 0 | 8.1 | (anonymous function) | painter.js:183 |
| 8.1 | 8.1 | l | math.min.js:27 |
| 0 | 7.1 | R._engine.rect | raphael-2.1.2.js:6919 |
| 0 | 7.1 | paperproto.circle | raphael-2.1.2.js:3661 |
| 7.1 | 7.1 | eve.off.eve.unbind | raphael-2.1.2.js:288 |
| 3.0 | 7.1 | e | raphael-2.1.2.js:3097 |

Continued on next page

**Table 7 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 2.0 | 7.1 | R._engine.circle | raphael-2.1.2.js:6910 |
| 7.1 | 7.1 | t.min | math.min.js:27 |
| 3.0 | 7.1 | t.get | math.min.js:27 |
| 6.1 | 6.1 | get style | |
| 0 | 6.1 | (anonymous function) | painter.js:73 |
| 6.1 | 6.1 | f | math.min.js:28 |
| 0 | 6.1 | (anonymous function) | painter.js:177 |
| 0 | 6.1 | highlight | qubit.js:32 |
| 5.0 | 5.0 | repush | raphael-2.1.2.js:1204 |
| 0 | 5.0 | (anonymous function) | painter.js:141 |
| 4.0 | 4.0 | addEventListener | |
| 0 | 4.0 | (anonymous function) | raphael-2.1.2.js:7115 |
| 0 | 4.0 | (anonymous function) | math.min.js:27 |
| 1.0 | 4.0 | t | math.min.js:27 |
| 0 | 4.0 | (anonymous function) | raphael-2.1.2.js:7117 |
| 1.0 | 4.0 | n.size | math.min.js:31 |
| 4.0 | 4.0 | (anonymous function) | raphael-2.1.2.js:1480 |
| 4.0 | 4.0 | R._path2string | raphael-2.1.2.js:1201 |
| 3.0 | 3.0 | i | math.min.js:31 |
| 3.0 | 3.0 | createElementNS | |
| 0 | 3.0 | h | math.min.js:28 |
| 0 | 2.0 | R._engine.setSize | raphael-2.1.2.js:6964 |
| 0 | 2.0 | decrementStepCount | controller.js:124 |
| 0 | 2.0 | paperproto.setSize | raphael-2.1.2.js:3862 |
| 0 | 2.0 | (anonymous function) | controller.js:90 |
| 0 | 2.0 | (anonymous function) | controller.js:91 |
| 2.0 | 2.0 | R._removedFactory | raphael-2.1.2.js:1946 |
| 0 | 2.0 | u | math.min.js:27 |
| 1.0 | 2.0 | paths | raphael-2.1.2.js:1539 |
| 2.0 | 2.0 | (anonymous function) | math.min.js:29 |
| 1.0 | 1.0 | call | |
| 0 | 1.0 | e.eye | math.min.js:29 |
| 0 | 1.0 | R.format | raphael-2.1.2.js:5643 |
| 0 | 1.0 | updateHelpText | controller.js:36 |
| 0 | 1.0 | (unresolved function) | |
| 0 | 1.0 | Element | raphael-2.1.2.js:6356 |
| 1.0 | 1.0 | setproto.push | raphael-2.1.2.js:5185 |
| 1.0 | 1.0 | (anonymous function) | raphael-2.1.2.js:6238 |
| 1.0 | 1.0 | n.toNumber | math.min.js:31 |
| 1.0 | 1.0 | setTimeout | |
| 0 | 1.0 | t.resize | math.min.js:27 |

Continued on next page

**Table 7 – continued from previous page**

| Self (ms) | Total (ms) | Function | Source |
|---|---|---|---|
| 1.0 | 1.0 | paperproto.set | raphael-2.1.2.js:3813 |
| 1.0 | 1.0 | R.matrix | raphael-2.1.2.js:2801 |
| 1.0 | 1.0 | get x | |
| 1.0 | 1.0 | (anonymous function) | math.min.js:27 |
| 1.0 | 1.0 | f | math.min.js:27 |
| 1.0 | 1.0 | set innerHTML | |

Figure 34: Performance Test Run 2-3 Results

## 15.5 Analysis Documentation

Table 8: Evaluator Analysis.

| | | | Time (ms) | | | |
|---|---|---|---|---|---|---|
| **Flag** | **Iterations** | **Gates** | **1** | **2** | **3** | **Avg.** |
| 0 | 1 | 22 | 20 | 12 | 18 | 16.67 |
| | 2 | 39 | 29 | 24 | 23 | 25.33 |
| | 3 | 56 | 37 | 65 | 33 | 45.00 |
| 1 | 1 | 20 | 17 | 12 | 11 | 13.33 |
| | 2 | 35 | 21 | 35 | 22 | 26.00 |
| | 3 | 50 | 28 | 27 | 41 | 32.00 |
| 2 | 1 | 20 | 12 | 16 | 12 | 13.33 |
| | 2 | 35 | 19 | 18 | 16 | 17.67 |
| | 3 | 50 | 28 | 27 | 41 | 32.00 |
| 3 | 1 | 18 | 12 | 10 | 12 | 11.33 |
| | 2 | 31 | 19 | 18 | 16 | 17.67 |
| | 3 | 44 | 38 | 26 | 28 | 30.67 |
| 4 | 1 | 29 | 59 | 44 | 40 | 47.67 |
| | 2 | 52 | 80 | 75 | 70 | 75.00 |
| | 3 | 75 | 137 | 102 | 137 | 125.33 |
| 5 | 1 | 27 | 40 | 36 | 67 | 47.67 |
| | 2 | 48 | 69 | 66 | 68 | 67.67 |
| | 3 | 69 | 97 | 97 | 99 | 97.67 |
| 6 | 1 | 27 | 40 | 36 | 35 | 37.00 |
| | 2 | 48 | 67 | 66 | 63 | 65.33 |
| | 3 | 69 | 125 | 100 | 99 | 108.00 |
| 7 | 1 | 25 | 36 | 33 | 32 | 33.67 |
| | 2 | 44 | 89 | 61 | 62 | 70.67 |

Continued on next page

**Table 8 – continued from previous page**

| | | | Time (ms) | | | |
|---|---|---|---|---|---|---|
| **Flag** | **Iterations** | **Gates** | **1** | **2** | **3** | **Avg.** |
| | 3 | 63 | 93 | 92 | 91 | 92.00 |
| 8 | 1 | 38 | 152 | 178 | 137 | 155.67 |
| | 2 | 69 | 330 | 334 | 312 | 325.33 |
| | 3 | 100 | 438 | 429 | 437 | 434.67 |
| 9 | 1 | 36 | 132 | 130 | 160 | 140.67 |
| | 2 | 65 | 270 | 267 | 301 | 279.33 |
| | 3 | 94 | 418 | 430 | 427 | 425.00 |
| 10 | 1 | 36 | 134 | 131 | 151 | 138.67 |
| | 2 | 55 | 283 | 232 | 258 | 257.67 |
| | 3 | 84 | 383 | 372 | 342 | 365.67 |
| 11 | 1 | 34 | 140 | 139 | 155 | 144.67 |
| | 2 | 51 | 290 | 270 | 260 | 273.33 |
| | 3 | 78 | 375 | 359 | 326 | 353.33 |
| 12 | 1 | 36 | 145 | 150 | 157 | 150.67 |
| | 2 | 55 | 300 | 290 | 280 | 290.00 |
| | 3 | 84 | 345 | 372 | 377 | 364.67 |
| 13 | 1 | 34 | 140 | 145 | 147 | 144.00 |
| | 2 | 51 | 305 | 295 | 280 | 293.33 |
| | 3 | 78 | 362 | 357 | 355 | 358.00 |
| 14 | 1 | 34 | 150 | 143 | 147 | 146.67 |
| | 2 | 51 | 310 | 279 | 285 | 291.33 |
| | 3 | 78 | 363 | 350 | 349 | 354.00 |
| 15 | 1 | 32 | 155 | 145 | 149 | 149.67 |
| | 2 | 47 | 256 | 231 | 242 | 243.00 |
| | 3 | 72 | 367 | 348 | 352 | 355.67 |
| 16 | 1 | 47 | 564 | 535 | 529 | 542.67 |
| | 2 | 86 | 1122 | 1137 | 1158 | 1139.00 |
| | 3 | 125 | 1620 | 1622 | 1596 | 1612.67 |
| 17 | 1 | 45 | 510 | 505 | 514 | 509.67 |
| | 2 | 82 | 1050 | 1080 | 1056 | 1062.00 |
| | 3 | 119 | 1580 | 1530 | 1525 | 1545.00 |
| 18 | 1 | 45 | 503 | 504 | 500 | 502.33 |
| | 2 | 82 | 1065 | 1070 | 1055 | 1063.33 |
| | 3 | 119 | 1588 | 1535 | 1540 | 1554.33 |
| 19 | 1 | 43 | 499 | 490 | 505 | 498.00 |
| | 2 | 78 | 990 | 1002 | 1005 | 999.00 |
| | 3 | 113 | 1550 | 1560 | 1500 | 1536.67 |
| 20 | 1 | 45 | 507 | 510 | 512 | 509.67 |
| | 2 | 82 | 1080 | 1075 | 1083 | 1079.33 |

Continued on next page

**Table 8 – continued from previous page**

| | | | Time (ms) | | | |
|---|---|---|---|---|---|---|
| **Flag** | **Iterations** | **Gates** | **1** | **2** | **3** | **Avg.** |
| | 3 | 119 | 1600 | 1558 | 1569 | 1575.67 |
| 21 | 1 | 43 | 485 | 487 | 491 | 487.67 |
| | 2 | 78 | 996 | 999 | 1001 | 998.67 |
| | 3 | 113 | 1543 | 1540 | 1570 | 1551.00 |
| 22 | 1 | 43 | 493 | 453 | 450 | 465.33 |
| | 2 | 78 | 1010 | 1003 | 989 | 1000.67 |
| | 3 | 113 | 1551 | 1564 | 1533 | 1549.33 |
| 23 | 1 | 41 | 501 | 453 | 448 | 467.33 |
| | 2 | 74 | 943 | 956 | 947 | 948.67 |
| | 3 | 107 | 1503 | 1506 | 1497 | 1502.00 |
| 24 | 1 | 45 | 516 | 519 | 507 | 514.00 |
| | 2 | 82 | 1101 | 1114 | 1092 | 1102.33 |
| | 3 | 119 | 1624 | 1579 | 1588 | 1597.00 |
| 25 | 1 | 43 | 475 | 481 | 492 | 482.67 |
| | 2 | 78 | 1015 | 1026 | 991 | 1010.67 |
| | 3 | 113 | 1576 | 1586 | 1534 | 1565.33 |
| 26 | 1 | 43 | 460 | 471 | 499 | 476.67 |
| | 2 | 78 | 1054 | 1037 | 1021 | 1037.33 |
| | 3 | 113 | 1568 | 1549 | 1503 | 1540.00 |
| 27 | 1 | 41 | 403 | 434 | 426 | 421.00 |
| | 2 | 74 | 957 | 963 | 971 | 963.67 |
| | 3 | 107 | 1504 | 1483 | 1470 | 1485.67 |
| 28 | 1 | 43 | 464 | 457 | 471 | 464.00 |
| | 2 | 78 | 1064 | 1072 | 1052 | 1062.67 |
| | 3 | 113 | 1601 | 1589 | 1530 | 1573.33 |
| 29 | 1 | 41 | 405 | 416 | 413 | 411.33 |
| | 2 | 74 | 977 | 989 | 919 | 961.67 |
| | 3 | 107 | 1548 | 1543 | 1501 | 1530.67 |
| 30 | 1 | 41 | 411 | 426 | 439 | 425.33 |
| | 2 | 74 | 1016 | 975 | 989 | 993.33 |
| | 3 | 107 | 1573 | 1590 | 1552 | 1571.67 |
| 31 | 1 | 39 | 401 | 415 | 423 | 413.00 |
| | 2 | 70 | 978 | 963 | 947 | 962.67 |
| | 3 | 101 | 1462 | 1434 | 1447 | 1447.67 |

Figure 35: Evaluator Analysis