The University Of Leeds

School of Physics and Astronomy

Ph.D. Thesis

Using the qubus for quantum computing

Katherine Louise Brown

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy.
April 2011

The candidate confirms that the work submitted is her own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others. The contributions of other authors to this work has been made explicitly clear throughout the thesis, by the use of their name. All other work, has been contributed by me, even when the personal pronoun 'we' has been used in the thesis.

Chapter 2 has been extended, and published as a review article on quantum simulation. This was jointly authored by me, and Viv Kendon, with Bill Munro making significant contributions to the section on state initialisation, particularly on creating thermal equilibrium states, and direct state preparation. The research for the paper was primarily conducted by me.
Paper: K.L. Brown, W.J. Munro, V.M. Kendon, *'Using Quantum Computers for Quantum Simulation'*, Entropy **12** 2268

Chapters 3 through to 6 have been submitted as a paper on simulating the BCS Hamiltonian. The majority of this work, was carried out by me, supervised by Viv Kendon, and Bill Munro. Suva De assisted with the work on performing the QFT, and the optimal gate sequence for that was joint work. However, the work on creating CNOT gates, was again carried out by me alone.
Paper: K.L.Brown, S. De, V.M. Kendon, W.J. Munro, *'Ancilla-based quantum simulation'*, arXiv:1011.2984 (submitted to New Journal of Physics)

Chapter 8 will form a publication, that is in preparation. This will looked at a layered technique for generating cluster states. This paper will contain work conducted by me, under supervision of Viv Kendon, and Bill Munro, with input from Clare Horsman.

Chapter 9 has been published, in a paper on robust cluster state generation. The bulk of the combinatorial work for this section has been carried out by myself, with Vivien Kendon, and Clare Horsman contributing ideas. The proofs in the paper contain material from chapters 7 and 8, and were worked out by Clare Horsman. The error models in this paper are provided by Bill Munro.
Paper: C. Horsman, K.L. Brown, W.J. Munro, V.M. Kendon, *'Reduce, reuse, recycle, for robust cluster state generation'*, Phys. Rev. A **83** 042327

Acknowledgements

First of all, thanks to my supervisors Viv Kendon, and Bill Munro, for all the help and support they have provided throughout my PhD. In particular, Bill Munro, has provided significant help in understanding the qubus system, and potential uses, as well as contributing the error model for this thesis. Viv Kendon, has made significant contributions to the work on building cluster states, in particular when considering the brick-based, and column based schemes. I'd also like to thank Viv, and Bill, for the significant emotional support, and encouragement they have given me, throughout my time as a PhD student. This wouldn't have been possible without them.

I'd also like to thank Clare Horsman, and Suvabrata De for their significant contributions to the work in this thesis. Clare, in particular contributed significantly to the work on robustness within cluster states, and provided the proofs of optimality for cluster states. This work significantly improved the work on cluster state generation, and lead to the development of the spiral-based scheme, and the column based scheme, proposed by Viv. I'd also like to thank Clare for the emotional support she provided me with, and for putting up with my moods. She helped me through some tough times, and ensured I sought the help I needed. I am grateful for this. Suvabrata De made significant contributions to work on performing the quantum Fourier transform, demonstrating that the circuit we came up with jointly, was a suitable scheme for implementation.

Mark Everitt, provided the LaTeX template for this document, and significant technical help throughout my PhD. The template, has simplified the processing of writing the PhD, and the extensive preamble, means all the necessary fonts and packages work. Without Mark's technical help, this PhD would have been significantly more difficult, so I'm grateful to him for this help. I'd also like to thank him for the enjoyable times we spent together.

My parent's next door neighbour, Margaret, provided significant help with the proof reading of this work. I am grateful for her contributions, and all the time she put in. I believe her work has made this document significantly more readable, and helped compensate for my own poor grammar. Any sections, which still read poorly, are those which I hadn't finished in time to send her the document.

I had very useful discussions with Kae Nemoto about the qubus architecture, and with Tim

Spiller about the BCS Hamiltonian. These helped increase my understanding of the subjects, and I am very grateful to them. Janet Anders, and Dan Browne, provided many useful discussions, some of which have contributed to ideas for this thesis, but most of which have focused on work that is planned for the future. I'm grateful to Jonathon Dowling for offering me a job, and for the people at LSU for help, and useful discussions. I'd also like to thank all those in my office past and present; Mireia Alòs, Martin Aulbach, Tony Blake, Katie Barr, Jessica Cooper, Neil Lovett, Veiko Palge, Adam Stokes, Giovanni Vacanti and Rob Wagner, for useful discussions, and putting up with my mess, and bad moods. They have helped me with this work, and let me rant when necessary. I'd also like to thank, all those friends I have made through conferences, while too numerous to list here, they are what has made doing a PhD fun, and it is always great to see them again. The same goes for the numerous friends I made at Leeds.

I'm grateful to Faith Bonner for administrative support, and sorting my pay when things went wrong. I'd also like to thank HP and EPSRC for providing me the funding to do this research, and attend numerous conferences. It was this funding which made my PhD financially viable, and put food on my table everyday.

I owe a lot to my teachers from the Holt School, Wokingham, in particular, but not limited to Mrs Weaver, Mrs Hancock, and Miss Bell. It was their enthusiasm, and encouragement that led to me going into physics. I was very lucky to have had good, enthusiastic specialist physics teachers. Without them, I don't think I would be here today. I am also grateful to Mr Hayter, Ms Nayer, Mr Childs, and Dr Salvison, all of whom contributed to my interest in the physical sciences. I think the fact that a disproportionate number of people from our school went into both the physical, and biological sciences was a testimony to how good these teachers were, and I am grateful for it.

Finally, I'd like to thank those closest to me for all the emotional support they have provided during this turbulent time. In particular, my mum and dad have provided me with a large amount of both emotional, and financial support. I am grateful that they let me choose my own direction in life, and for the level of support they have provided me with. I have been supported in everything I ever wanted to do, even if this was difficult for them. I am insanely grateful for that, and don't think I can ever repay them. Without their support I don't think I could have achieved half the things I have today. I'd also like to thank my friends from home, and from Durham, who have helped me through difficult times. In particular I am grateful to Danielle Miller, and Michelle Clark, for sticking with me, even when I have been difficult to get hold of, or busy working. They have been good friends through thick and thin. From Durham I would like to thank Robert Moss, Emma Nicholls, Jason Raj, Kristen Jones, and Anthony Bailey for their continued emotional support. I know I have been difficult at times, and I thank you for putting up with me. I'd also like to thank, and apologise to anyone I have missed off this list.

Abstract

Katherine Louise Brown "Using the qubus for quantum computing", Ph.D. thesis, University of Leeds, April 2011.

In this thesis I explore using the qubus for quantum computing. The qubus is an architecture of quantum computing, where a continuous variable ancilla is used to generate operations between matter qubits. I concentrate on using the qubus for two purposes - quantum simulation, and generating cluster states.

Quantum simulation is the idea of using a quantum computer to simulate a quantum system. I focus on conducting a simulation of the BCS Hamiltonian. I demonstrate how to perform the necessary two qubit operations in a controlled fashion using the qubus. In particular I demonstrate an O(N^3) saving over an implementation on an NMR computer, and a factor of 2 saving over a naïve technique. I also discuss how to perform the quantum Fourier transform on the qubus quantum computer. I show that it is possible to perform the quantum Fourier transform using just, $24\lfloor N/2 \rfloor + 7N - 6$, this is an O$(N)$ saving over a naïve method.

In the second part of the thesis, I move on, and consider generating cluster states using the qubus. A cluster state, is a universal resource for one-way or measurement-based computation. In one-way computation, the pre-generated, entangled resource is used to perform calculations, which only require local corrections and measurement. I demonstrate that the qubus can generate cluster states deterministically, and in a relatively short time. I discuss several techniques of cluster state generation, one of which is optimal, given the physical architecture we are using. This can generate an $n \times m$ cluster in only $3nm - 2n - 2m + 4$ operations. The alternative techniques look at generating a cluster using layers or columns, allowing it to be built dynamically, while the cluster is used to perform calculations. I then move on, and discuss problems with error accumulation in the generation process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Quantum computing has the potential to revolutionise computation as we know it. Algorithms for quantum computing exist which give an exponential improvement over classical computation. The most famous example is Shor's algorithm [1], which can find the prime factors of a large number in a time that scales polynomially with the number of bits in the system. The best available classical algorithm, requires a time which scales exponentially with the size of the system. However implementing quantum algorithms, and building quantum computers are difficult problems. Numerous architectures have been proposed for quantum computing, and as yet one has not won out, as a dominant architecture. In this thesis I will concentrate on using the qubus for quantum computing, because of the advantages it presents for generating interactions between distant qubits.

The qubus is a form of ancilla-based quantum computing, where one system is used to generate interactions within another. The main advantage of ancilla based quantum computing is that it allows us to generate interactions between distant qubits without the need for a large number of swap gates. It also allows qubits to be placed far enough apart to get individual addressability, something which can be problematic otherwise. The qubus is also a form of hybrid quantum computing, where two different forms of quantum system are used within the computing process. In this case a continuous variable field is used to generate interactions between matter qubits [2, 3]. This system has significant advantages as we do not need to be able to generate then detect single photons, in order to generate entanglement. It is possible to make significant savings in the number of operations required for performing an operation by rearranging the order in which they appear. This allows us to generate multiple-instances of two body entanglement between sets of qubits in a single set of operations.

In my thesis, I will consider using the qubus for two specific purposes - quantum simulation, and generating cluster states. In both cases, we show how to get significant savings in the total number of operations required by rearranging the order in which we perform them. In both cases we demonstrate at least a factor of 2 improvement over a naïve method, with both the improved and naïve case being efficient. These significant improvements demonstrate the advantages of the qubus quantum computer. As part of our work on simulation we show how to perform some standard building blocks of quantum computing. This includes the quantum Fourier transform, and making operations controlled on an ancilla qubit. This shows that our work is applicable to a far larger range of cases than just the two we covered.

1.1 Quantum simulation

We chose to look at quantum simulation, because it is one example of where we can obtain an exponential speed up by using a quantum system, compared to a classical system. Quantum simulation was first proposed by Feynman in 1982 [4], since simulating an arbitrary quantum system on a classical computer is a hard problem. Feynamn's argument centred on the fact that even storing a quantum state, required resources which scale exponentially with the size of the system. His argument was developed by Lloyd [5], who demonstrated that it was possible to perform the necessary time evolutions in an efficient fashion. The difficulties with simulating an arbitrary quantum system on a classical computer demonstrate one of the reasons why quantum simulation is so interesting. One of the largest completely general simulations of a quantum system, on a classical computer, looked at simulating only 36 qubits [6]. Even this required a significant amount of computational power. To perform even slightly larger simulations would be beyond the capabilities of today's computers. We can therefore see that the break even point for quantum simulation, the point where quantum algorithms give advantages over their classical counterparts, is going to be significantly lower than for other quantum algorithms. At the same time, quantum simulation has the potential to explore interesting physical systems. Algorithms have already been proposed exploring things such as ground states of molecules [7], details about bond formation [8], and eigenvalues and eigenvectors [9] of many-body quantum Hamiltonians; for a recent review, see Brown et al. [10].

Our work on quantum simulation concentrates on one particular Hamiltonian, the BCS Hamiltonian. This is a description of superconductivity that was first proposed by Bardeen, Cooper and Schrieffer in 1957 [11]. We consider this model for two primary reasons. The first, is that a large amount of previous work has looked at simulating this system on a quantum computer [12–17]. The second, is that it is in itself an interesting problem. For large scale systems, the BCS ansatz holds true, while for small scale systems a complete classical simulation is possible. This leaves a problem with medium sized system where it is uncertain whether the BCS ansatz will hold, and where it is not possible to conduct a full classical simulation. An example of such a system is small metallic grains, these systems typically consist of a few hundred relevant states. A quantum computation of these systems would reveal new information, and would confirm whether the BCS ansatz does hold in these scenarios.

1.2 Cluster state computation

The circuit model is not the only model of quantum computing. One of the alternative but directly equivalent models is measurement-based computation [18]. In this case, a large highly entangled resource is created, then measurements and local corrections are performed on this resource to drive the calculation. This removes many of the difficulties for performing calculations. However generating the initial resource is not trivial. The most commonly used resource is a cluster state, which is a lattice graph of qubits with C-Phase gates between nearest neighbours. Cluster states are universal for quantum computing, and save having to design a specific generation procedure for the specific graph state required to implement a particular calculation. Pushing the entanglement generation off-line in this fashion is not trivial, and turns out to be expensive in terms of the total entanglement required. Therefore, the obvious question is - why do it?

There are several reasons. The first is that it is possible to generate a cluster state efficiently while using a heralded entangling gate with a low success probability [19]. This is a significant advantage, because while it is possible to turn probabilistic gates into deterministic gates through repeat applications, this is resource intensive [20]. Therefore, this probabilistic growth is more efficient. The second reason is that generating all the entanglement first pushes the most difficult stages of the calculation off-line. Therefore, any necessary corrections to our system can be done before we begin the calculations. We are also considering using only a single entangling gate, without requiring a large number of local unitaries to turn it into the necessary form of entanglement. This has the potential to simplify the necessary algorithms considerably, and hopefully make implementation more feasible. It also considerably simplifies the range of entanglement required, since to perform any quantum operation we only need nearest-neighbour entanglement. When we consider cases such as quantum simulation, we often need to entangle a qubit to both near-by and distant qubits. It is not always possible to rearrange our qubit system in such a way that all the entanglement is physically local. Once again this should make cluster state computation easier to perform. Finally, there are small improvements in the computational efficiency of measurement-based quantum computation compared to the circuit model. In particular a measurement-based computer is equivalent to the circuit model with the addition of the highly theoretical fan-out (copy) gate [21]. This reduces the circuit-depth of certain operations [22] although it does not give any changes in complexity class.

We can see therefore, that the cluster state offers numerous potential advantages. In this work, we do not consider how to use a cluster state, and instead simply focus on cluster state generation. While it is possible to generate a cluster state in a probabilistic fashion, this is expensive and can lead to problems with routing, therefore it is worth considering a deterministic technique for generating cluster states. The qubus seems a suitable system to use for this, because it can be used to generate deterministic C-Phase gates. It also fits well with other developments on deterministically generating cluster states, in the fact that it is an ancilla system. In our work we look at several techniques for generating cluster states, including one which is optimal given the system we are using. One significant issue with cluster states is decoherence. If too much of the cluster is generated in one go, then it is likely to decohere before operations are performed on it. This suggests that it is best to create the cluster state in a dynamical fashion, where operations are performed on the early parts of the cluster, while the later parts are being generated. We therefore, introduce a technique for generating the cluster in a dynamic fashion that generates the rows of the cluster very quickly. This should minimise problems with decoherence.

1.3 Composition of thesis

In chapter 2 we introduce the qubus architecture of quantum computing. We outline previous work conducted on this architecture, including basic entangling gates, and saving operations by reordering our bus interactions. This chapter provides a basis for the other work in the thesis. For completeness, we also show how to use the qubus quantum computer to generate local unitaries. This is something which has not been looked at previously, due to the assumption that each qubit would have individual addressability. We also briefly discuss some experiments, which have shown the possibilities of entangling a continuous variable with a qubit, and thus demonstrate that a qubus scheme is feasible.

Chapter 3 introduces the idea of using a quantum computer for simulating quantum systems. We explore the foundational results for the three processes of quantum simulation, state initialisation, time evolution, and data extraction, and demonstrate that all three can be performed efficiently in certain cases. In this chapter we also introduce the BCS Hamiltonian and discuss in full the algorithm presented by Wu et al. [12] for simulating this on a quantum computer.

Chapters 4, 5, 6 and 7 contain our work on simulating the BCS Hamiltonian on the qubus quantum computer. We begin in chapter 4 by concentrating on how to perform the necessary two qubit unitaries. In particular we show how to perform these on a qubus quantum computer while achieving up to $\mathrm{O}(N)$ saving over a naïve technique. Even in the worst cases, we demonstrate a factor of 2 improvement. This shows the significant advantages of our qubus for generating long range interactions. However, the data extraction procedure we use - the phase estimation algorithm [23], requires that our operations are implemented in a controlled fashion. We explore how to do this in chapter 5, where we introduce some techniques for making our operations controlled. These techniques are adaptations of standard techniques to make a single qubit local unitary controlled [24]. In chapter 6 we discuss extracting data from our system. In particular, we show how to perform the quantum Fourier transform (QFT) in only $\mathrm{O}(N)$ operations. We find that the dominant term in performing the QFT is the need for swap gates, and that even with these we only require $24\lfloor N/2\rfloor + 7N - 6$ operations. Finally, in chapter 7 we put the results from the previous chapters together to work out the total number of operations required to simulate the BCS Hamiltonian. We find that for a precision given by δ, only $T_{\text{qubus}} \approx \frac{0.1\pi}{\delta}(122N^2 + 1683N - 956)$ operations are required. Therefore, we see that we require fewer operations than for a simulation on an NMR quantum computer provided $N \geq 6$ in the nearest neighbour case, and $N \geq 7$ in the general case.

We then move on and look at cluster state computation. Chapter 8 introduces the concept of cluster state computation. The main focus is on previous work that has looked at creating cluster states. We briefly outline some previous techniques for generating cluster states, including work that has been conducted on the qubus by Louis et al. [25]. In new work, we adapt this scheme to make it optimal when we consider connecting only chains of qubits. Chapters 9, and 10 contain our new work on cluster state computation. In chapter 9 we introduce a technique where we build our cluster layer by layer. We begin considering each layer individually, then look at ways we can save operations by keeping our bus attached to the qubit system between layers. In this chapter we also derive a lower bound on the number of operations required, and show that our layered generation technique can not meet this bound, even in the ideal case. In particular, we note that the minimum number of operations required given our layered technique, appears to be $3nm - 2n - \frac{3}{2}m + 3$. As a result in chapter 10 we go on to consider other techniques for generating cluster states. This work includes using a spiral path, which we demonstrates meets our lower bound of $3nm - 2n - 2m + 4$ operations. We also consider how to make our cluster state generation more resistant to errors, and introduce techniques for creating our cluster using multiple buses.

Finally we conclude in chapter 11, which contains a summary of the results we have obtained in our previous chapters, and some suggestions for further work. For completeness we have an appendix at the end of our work which contains further information for our chapters on cluster state generation.

Chapter 2

The qubus quantum computer

2.1 Introduction

In this chapter we will introduce the qubus quantum computer, which will be the main focus of work in this thesis. The qubus is a form of hybrid quantum computer, where a continuous variable field is used to generate interactions in a main processing unit made of qubits [2, 3]. Hybrid quantum computers use two (or more) different types of quantum system to perform calculations. In the majority of cases, hybrid quantum computers are a form of ancilla based quantum computing, where one system is used to generate interactions between the units of the second. The qubus fits into this category of ancilla systems.

Ancilla based schemes are useful for quantum computation, because they remove the need for qubits to interact directly to generate entanglement. They have two principal advantages. The first is that they allow interactions to be generated between distant qubits without the need for swap gates: the second, is that they remove the difficulties that come from needing to have both individual addressability, and the ability to directly generate entanglement. While qubits need to be placed close together to allow direct entanglement to be generated, distance is required to obtain individual addressability. We therefore find that, there is a trade off between these two requirements, for example in optical lattices containing neutral atoms it is often difficult to get individual addressability [26] although entangling and global operations are possible. In other systems, such as photons, the reverse is true, and generating entanglement can require a large amount of additional resources.

In this chapter we introduce techniques for performing two qubit gates using the qubus quantum computer. In section 2.2 we will look at schemes which focus on using controlled rotations to generate the necessary operations. We start by introducing a parity gate which is near-deterministic with repeats, or with the availability of a photon-number resolving detector.

Next, we briefly introduce a deterministic sequence of operations for producing a maximally entangling C-Phase or CNOT gate. We do not discuss this gate in detail, rather we simply observe that such a gate is not error free, even in the ideal case. As a result, in section 2.3 we introduce several techniques for producing gates using controlled displacement operators. Once again, we begin by discussing the parity gate, before moving on and showing some deterministic schemes for performing the C-Phase and CNOT gates. We also show a method for generating our controlled displacements from controlled rotations and deterministic displacement operations. Finally in section 2.4 we briefly introduce some of the experimental schemes which might be used to make a physical qubus architecture.

2.2 Using controlled rotations

The earliest gates developed for the qubus quantum computer used a combination of controlled rotations, and deterministic displacement operations to generate entanglement [2, 27, 28]. In most cases these gates were probabilistic, or inherently imperfect. However, in certain cases the addition of a photon number resolving detector could make these gates deterministic. Unfortunately, a photon number resolving detector is still a physically unrealistic requirement, although this may change in the next few years. In this section we will outline some of these gates, as an introduction to the operation of the qubus. In reality, the gates outlined in the next section are more promising. However these early gates show the development of the qubus as an architecture, and concentrate on more physically realisable interactions.

Following the work of Nemoto et al. [2, 27, 28] we consider using an interaction Hamiltonian of the form

$$H_{int} = \hbar\chi\sigma_z a^\dagger a \tag{2.1}$$

where the bus mode interaction is taking the effective form of a cross-Kerr non-linearity. Here we have $a^\dagger$ and a representing the creation and annihilation operators, σ_z as the Pauli Z operation on a qubit, and χ as the strength of the non-linearity. Such an operation produces a rotation on the phase space of the bus of the form $\pm\phi$, where $\phi = \chi t$ with the sign dependent upon the state of the qubit.

If we also incorporate a deterministic displacement operator of $D(-\kappa)$, where D takes the form $D(\kappa) = \exp(\kappa a^\dagger - \kappa^* a)$, it is possible to use this to generate a two qubit parity gate. The circuit shown in figure 2.1 transforms an initial state given by

$$|\Psi_i\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)|\kappa\rangle \tag{2.2}$$

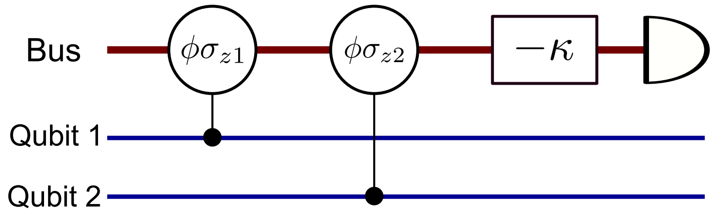


Figure 2.1: This figure shows a parity gate built of controlled rotations (show as circular gates), and uncontrolled displacements (shown as rectangular gates). If the measurement is a photon number resolving detector then this gate always returns a Bell state

to the final state,

$$|\Psi_f\rangle = \frac{1}{2}\Big(|00\rangle|\kappa(e^{2i\phi}-1)\rangle + (|01\rangle+|10\rangle)|0\rangle + |11\rangle|\kappa(e^{-2i\phi}-1)\rangle\Big) \tag{2.3}$$

A photon number resolving measurement results in two states, when $n = 0$ we get the state

$$|\Psi_{f_{n=0}}\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \tag{2.4}$$

while when $n \neq 0$ we end up in the state

$$|\Psi_{f_{n\neq 0}}\rangle = \frac{1}{\sqrt{2}}\left(i^n|00\rangle + (-i)^n|11\rangle\right) . \tag{2.5}$$

Provided this photon resolving measurement is available, both cases result in a Bell State. We choose a value for κ that minimises the error, so that we can near deterministically distinguish the two states. While this photon number resolving detector can give us a deterministic result, it is physically unrealistic. We therefore consider the result of a measurement on the position quadrature of the bus. Performing this measurement projects the state in equation 2.3 into the result shown in equation (2.4) a total of 50% of the time. This results in an entangled state. However, the other 50% of the time, since n is unknown, the result of the measurement is a state with no entanglement. Spiller et al. [2] describe a third approach for generating the necessary gate which involves repeating measurements, and Hadamard operations in order to reduce the component of the state which is in the unentangled state shown in equation (2.5). This would allow us to create a near deterministic Bell state. Spiller et al. [2] also present an alternative gate, which allows a near deterministic Bell state to be generated at the cost of extra operations. However, in either case, without the photon number resolving measurement it is impossible to create a theoretically ideal parity gate.

While we have a circuit that allows us to create a Bell state, we also want to consider how to perform a standard two qubit gate. Bell state generation is not commonly used to generate

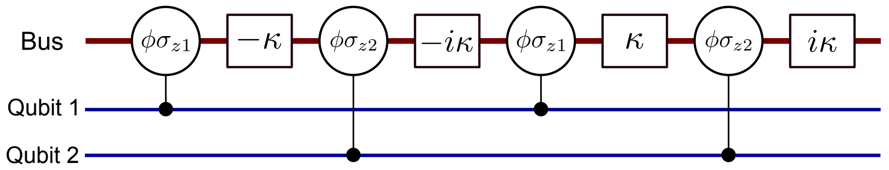


Figure 2.2: This figure shows how to perform a two-qubit gate using controlled rotation operations (shown as circular gates) and displacement operators (shown as rectangular gates).

quantum gates in the circuit model. We therefore want to consider how to make a universal maximally entangling gate such as a C-Phase, or CNOT gate. A suitable circuit is shown in figure 2.2. The details of this gate are outlined in the paper by Spiller et al. [2]. We do not discuss this gate here, but simply note that it is imperfect due to the nature of the controlled rotations.

Unfortunately, while we can produce a deterministic two qubit gate, it is imperfect. Even with ideal operations there will be an error in the resultant gate, and although it should be theoretically possible to correct for this using error correction protocols, it would be beneficial to consider a gate that is perfect in the ideal case. We explore how to do this in the next section where we consider using conditional displacement operators as the main component of our gates.

2.3 Using controlled displacements

One of the most significant advantages of the qubus quantum computer is the ability to generate useful entangling gates, both perfectly, and deterministically in the ideal case. We will concentrate on how to generate two entangling gates deterministically - the C-Phase gate and the CNOT gate [2, 3]. Both of these gates are maximally entangling, and therefore when combined with general local unitaries, are universal for quantum computation [29]. We will consider two schemes for generating our gates, one which requires controlled displacements and measurement, and one which requires only controlled displacements.

An important aspect of both these schemes is being able to perform a controlled displacement operator on our bus. This controlled displacement operator is performed based on the state of the qubit with which the bus is entangling. Spiller et al. [2] show how to do this using an interaction Hamiltonian of the form

$$H_{int} = \hbar\chi\sigma_z X(\theta) \tag{2.6}$$

where χ is the strength of the coupling between the field and the qubit, σ_z is the Pauli Z operator acting on the qubit, and $X(\theta)$ is a field quadrature operator of the form $(a^\dagger e^{i\theta} + ae^{-i\theta})$. We take $a^\dagger$ and a to be the field creation and annihilation operators. Implementing such a Hamiltonian for time t results in a displacement operation on the bus of the form $D(\sigma_z\beta)$ where $\beta = \chi t e^{i\left(\theta - \frac{\pi}{2}\right)}$. The displacement operator D takes the form $D(\sigma_z\beta) = \exp(\sigma_z\beta a^\dagger - \sigma_z\beta^* a)$. We use θ to represent the quadrature of the bus which the qubit of interest is coupling to. If a qubit couples to the momentum quadrature then we require that $\theta = \pi/2$, and hence $\beta = \chi t$. While if we couple a qubit to the position quadrature we require $\theta = 0$, and hence $\beta = i\chi t$. The net result of this is that we can perform displacements on our bus in two orthogonal directions.

We now consider what happens when we combine multiple displacement operators. To do this we use the Baker, Campbell, Hausdorff (BCH) formula to combine non-commuting exponentials [30–32]. In particular we note that to third order the BCH formula is given by

$$\exp A \exp B = \exp(A+B) \exp\left(\frac{1}{2}[A,B]\right) \exp\left(\frac{1}{12}[(A-B),[A,B]]\right). \qquad (2.7)$$

We consider combining two displacements of the form $D(\sigma_1\alpha_1) = \exp(\sigma_1\alpha_1 a^\dagger - \sigma_1\alpha_1^* a)$, and $D(\sigma_2\alpha_2) = \exp(\sigma_2\alpha_2 a^\dagger - \sigma_2\alpha_2^* a)$, where σ is an unspecified Pauli operator on an unspecified qubit, and α_1 and α_2 are complex constants. As σ is a Pauli operator, it is its own Hermitian transpose. We substitute these operations into equation (2.7), taking $A = (\sigma_1\alpha_1 a^\dagger - \sigma_1\alpha_1^* a)$ and $B = (\sigma_2\alpha_2 a^\dagger - \sigma_2\alpha_2^* a)$. It is easy to see that

$$\exp(A+B) = \exp\Big((\sigma_1\alpha_1 + \sigma_2\alpha_2)a^\dagger - (\sigma_1\alpha_1^* + \sigma_2\alpha_2^*)a\Big) = D(\sigma_1\alpha_1 + \sigma_2\alpha_2) \qquad (2.8)$$

The second order term gives us

$$\exp\left(\frac{1}{2}[A,B]\right) = \exp\begin{pmatrix} \sigma_1\sigma_2\left(\alpha_1\alpha_2 a^\dagger a^\dagger - \alpha_1\alpha_2^* a^\dagger a - \alpha_1^*\alpha_2 a a^\dagger + \alpha_1^*\alpha_2^* aa\right)/2 \\ +\sigma_2\sigma_1\left(-\alpha_1\alpha_2 a^\dagger a^\dagger + \alpha_1^*\alpha_2 a^\dagger a + \alpha_1\alpha_2^* a a^\dagger - \alpha_1^*\alpha_2^* aa\right)/2 \end{pmatrix}. \qquad (2.9)$$

If our terms σ_1 and σ_2 commute this can be reduced to

$$\exp\left(\frac{1}{2}[A,B]\right) = \exp\left(\frac{(\alpha_1\alpha_2^* - \alpha_1^*\alpha_2)\sigma_1\sigma_2}{2}\right) \qquad (2.10)$$

Therefore our third order term $\exp([(A-B),[A,B]]/12) = 0$ when σ_1 and σ_2 commute. Since our third order terms are 0 all further terms will also be 0. Provided σ_1 and σ_2 commute we can see that

$$D(\sigma_1\alpha_1)D(\sigma_2\alpha_2) = \exp\left(\frac{(\alpha_1\alpha_2^* - \alpha_1^*\alpha_2)\sigma_1\sigma_2}{2}\right) D(\sigma_1\alpha_1 + \sigma_2\alpha_2). \qquad (2.11)$$

This expression gives us a phase factor which is the imaginary part of $\alpha_1\alpha_2\sigma_1\sigma_2$. Since χ and

t are both real numbers, β is real when we connect a qubit to the momentum quadrature of the bus, and imaginary when we connect a qubit to the position quadrate of the bus. This means that a phase factor is generated when we connect two qubits to opposing quadratures of the bus, but that no phase factor is generated when we connect two qubits to the same quadrature of the bus. It is this phase factor which we use to entangle our qubits.

In the case where σ_1 and σ_2 do not commute this relation is not valid, and the result is a complicated expression containing creation and annihilation operators. It is therefore important that we consider only commuting Pauli operators in each sequence of operations. Since the three Pauli operators do not commute, we are limited to performing only one of the three, (although we can perform it multiple times), on each qubit before we disentangle the bus completely from our system.

For simplicity in explanation, we will now consider a slight change in notation. We will give our β as χt making it always real. The total displacement operator will now be $D\left(\sigma_z \beta e^{i\left(\theta - \frac{\pi}{2}\right)}\right)$. We thus use a displacement operator of the form $D(i\beta\sigma)$ when we connect to the position quadrature, and $D(\beta\sigma)$ when we connect to the momentum quadrature. This is mathematically equivalent, it simply makes the result of a sequence of displacements easier to visualise.

2.3.1 The parity gate

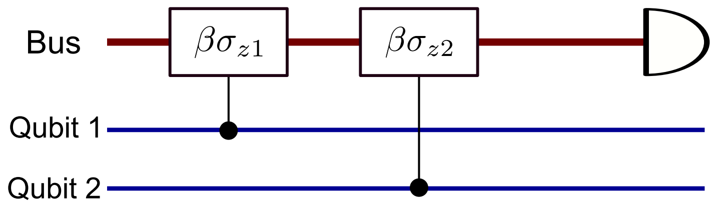


Figure 2.3: This figure shows a parity gate made entirely from controlled displacement operators. The rectangular boxes correspond to displacement operators on the momentum quadrature of the bus.

For completeness, we will demonstrate that it is possible to use our controlled displacement operations to create a parity gate, in a similar fashion to what we can do using controlled rotations [2]. This gate is illustrated in figure 2.3, and takes an initial state of the form

$$\Phi_i = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)|\kappa\rangle \tag{2.12}$$

to the state

$$|\Phi_f\rangle = \frac{1}{2}(|00\rangle|2\beta\rangle + |01\rangle|0\rangle + |10\rangle|0\rangle + |11\rangle| - 2\beta\rangle) \tag{2.13}$$

Similarly to the gate illustrated in section 2.2 provided we can perform a photon number resolving measurement, we end up with two different states. When $n = 0$ we get the state

$$|\Psi_{f_{n=0}}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.14}$$

while when $n \neq 0$ we end up in the state

$$|\Psi_{f_{n\neq 0}}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + (-i)^n|11\rangle) \ . \tag{2.15}$$

As before Spiller et al. [2] show that by repeat applications of the gate, along with measurement, it is possible to turn this gate into a near deterministic version of the parity gate.

2.3.2 A maximally entangling gate without measurement

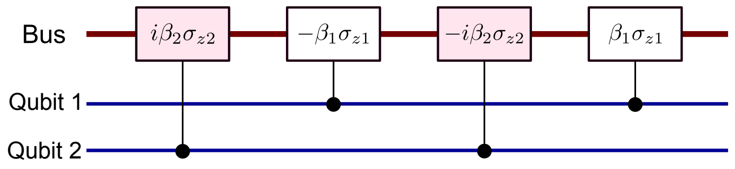


Figure 2.4: This circuit shows how we can create a gate on our qubus, to make this gate maximally entangling we require $2\beta_1\beta_2 = \pi/4$. The operations shown are displacement gates on the bus performed dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus, and unshaded boxes represent operations on the momentum quadrature of the bus.

The first gate we will consider is a maximally entangling gate between two qubits created without using measurement [2, 3]. This gate can be performed using four displacement operators, and is illustrated in figure 2.4. We will work through the results of the gates and demonstrate that they give a maximally entangling operation, which could be used for a CNOT or C-Phase gate.

$$\begin{aligned} U &= D(\beta_1\sigma_{z1})D(-i\beta_2\sigma_{z2})D(-\beta_1\sigma_{z1})D(i\beta_2\sigma_{z2}) \\ &= \exp(i\beta_1\beta_2\sigma_{z1}\sigma_{z2})D(\beta_1\sigma_{z1} - i\beta_2\sigma_{z2})D(-\beta_1\sigma_{z1})D(i\beta_2\sigma_{z2}) \\ &= \exp(2i\beta_1\beta_2\sigma_{z1}\sigma_{z2})D(-i\beta_2\sigma_{z2})D(i\beta_2\sigma_{z2}) = \exp(2i\beta_1\beta_2\sigma_{z1}\sigma_{z2})\,. \end{aligned} \tag{2.16}$$

The result of our sequence of displacements is an operator on two qubits, with no net operations on the bus. Therefore the bus starts and ends in the same state. At the end of our sequence of operations the bus is completely disentangled from the system. The resultant gate, $\exp(i\beta_1\beta_2\sigma_{z1}\sigma_{z2})$ is maximally entangling when $2\beta_1\beta_2 = \pi/4$. We can see this by considering the result of the gate on the input state

$$|\Psi_i\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle\,. \tag{2.17}$$

The net result is

$$|\Psi_f\rangle = U\left(\frac{\pi}{4}\right)|\psi_i\rangle = \exp\left(\frac{i\pi}{4}\right)[\alpha_1|00\rangle + \alpha_4|11\rangle] + \exp\left(\frac{-i\pi}{4}\right)[\alpha_2|01\rangle + \alpha_3|10\rangle] \quad (2.18)$$

we now apply a local unitary C_z to each qubit, where

$$C_z = \exp(i\pi/8)(\mathbb{1} - i\sigma_z)/\sqrt{2} \quad (2.19)$$

this results in the state

$$C_{z1}C_{z2}|\Psi_f\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle - \alpha_4|11\rangle\,. \quad (2.20)$$

We can therefore see that our unitary U is directly equivalent to a maximally entangling gate, and have demonstrated how to use local unitaries to turn this gate into a C-Phase gate.

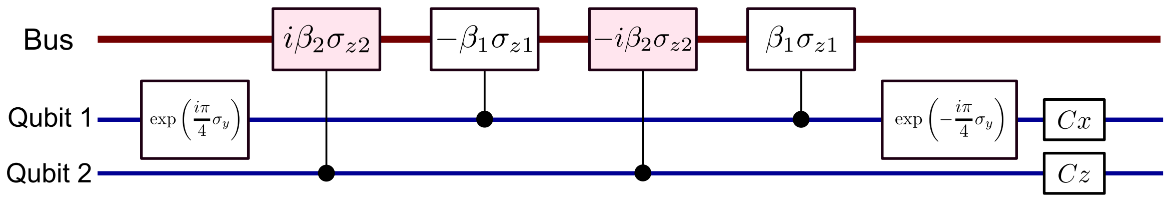


Figure 2.5: A diagrams showing how to perform the CNOT gate using 4 interactions with the bus, and 4 local unitaries. Cx and Cz are defined in equations (2.21) and (2.19) respectively. Shaded boxes represent an operation acting on the position quadrature of the bus, while unshaded boxes represent operations on the momentum quadrature.

Since both the C-Phase and the CNOT can be used for universal quantum computing when combined with local unitaries, we can easily see that they should be equivalent under local transformations. The CNOT gate is a particular useful gate in our work on the quantum Fourier transform, and in making gates controlled. We will therefore demonstrate here how to perform it using our qubus entangling operation. Spiller et al. [2] show that this involves using local operations to change the bus displacement to be based on a qubit in the form σ_x, rather than σ_z. The local operations that provide this change are given by $\exp(i\pi\sigma_y/4)$ and $\exp(-i\pi\sigma_y/4)$. They are performed either side of our sequence of operations, as illustrated in figure 2.5. The circuit shown will flip qubit 1 if qubit 2 is in the state $|1\rangle$. By performing our basis change on qubit 2 instead of qubit 1, we could generate the operation where we flip qubit 2 when qubit 1 is in $|1\rangle$. We note that our qubit which has the Pauli Y operator acting on it, requires a local correction of the form

$$Cx = \exp(i\pi/8)(\mathbb{1} - i\sigma_x)/\sqrt{2}\,. \quad (2.21)$$

This is directly equivalent to our previous correction but performed in the X basis instead of the Z basis.

We can perform a CNOT that flips the target qubit when the control qubit is in $|0\rangle$, by changing the form of our local correction on the target qubit to

$$Cx_0 = \exp(-3i\pi/8)(\mathbb{1} + i\sigma_x)/\sqrt{2} \tag{2.22}$$

Therefore, both forms of our CNOT are equally easy to perform on the qubus, and each gate requires only two operations more than the standard C-Phase gate.

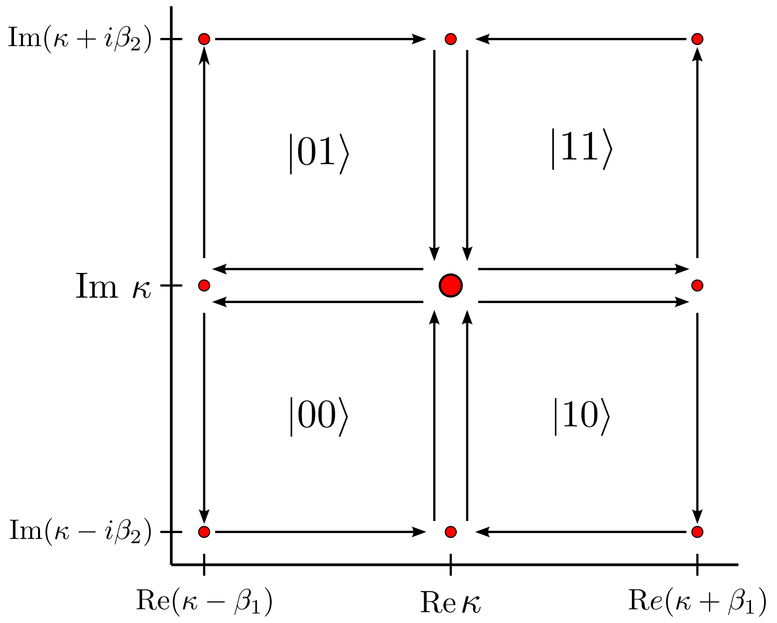

$\mathrm{Im}(\kappa + i\beta_2)$
$|01\rangle$
$|11\rangle$
$\mathrm{Im}\ \kappa$
$|00\rangle$
$|10\rangle$
$\mathrm{Im}(\kappa - i\beta_2)$
$\mathrm{Re}(\kappa - \beta_1)$
$\mathrm{Re}\,\kappa$
$\mathrm{Re}(\kappa + \beta_1)$

Figure 2.6: This diagram shows the movement of the bus through phase space for generating a maximally entangling gate without measurement.

It is worth considering here, what is happening in phase space to generate our maximally entangling gate. This allows us to easily visualise why we do not need measurement in this case. We illustrate the path the bus takes through phase space in figure 2.6. Dependent upon the state of the qubits it takes one of the four possible paths, ending up back in the central initial position. We can therefore easily see that the bus is disentangled from the qubits at the end of a sequence of operations, since it ends at the same point independent of the state of the qubits. Spiller et al. [2] note that it is the area traced out in the path through phase space that determines the phase acquired. Each path traces out the same total area, but in a different position within the phase space. As a result each path acquires a different phase, which all have the same magnitude. This is an elegant representation of our gate.

2.3.3 A maximally entangling gate with measurement

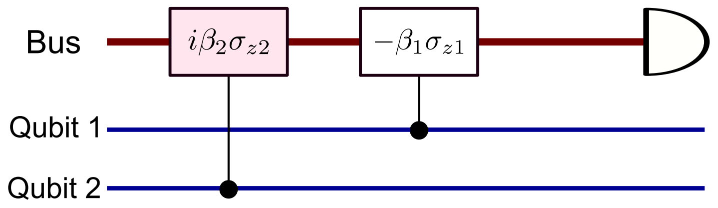


Figure 2.7: A figure showing how to create a deterministic gate between two qubits using displacement operators on the bus and measurement. The operations shown are displacement gates on the bus performed dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus, and unshaded boxes represent operations on the momentum quadrature of the bus.

Above we have discussed how to create a deterministic maximally entangling gate, without the need for measurement. We will now discuss how we can use a similar method but incorporate a measurement to reduce the total number of operations required. An example circuit is shown in figure 2.7. Our net operator is given by

$$U = D(-\beta_1\sigma_{z1})D(i\beta_2\sigma_{z2}) = \exp(i\beta_{z1}\beta_{z2}\sigma_{z1}\sigma_{z2})D(-\beta_1\sigma_{z1} + i\beta_2\sigma_{z2})\,. \tag{2.23}$$

If we consider our qubus and qubit system to be in the initial state

$$|\Psi_i\rangle = |\Psi_q\rangle|\kappa\rangle \tag{2.24}$$

where $|\Psi_q\rangle$ is the state of the qubits, and $|\kappa\rangle$ the state of the continuous variable field, the net result of U is

$$U|\Psi_i\rangle = \exp(i\beta_{z1}\beta_{z2}\sigma_{z1}\sigma_{z2})|\Psi_q\rangle|\kappa - \beta_1\sigma_{z1} + i\beta_2\sigma_{z2}\rangle\,. \tag{2.25}$$

Therefore, we can see that the bus is still entangled to the qubits.

To disentangle the bus from the qubits, we perform a photon number resolving measurement on the bus. The result of this measurement would not allow us to find any information about the state, since the net result of our displacement operations gives the same number state in all four cases. We can see this by looking at figure 2.8, where the resultant states are all on the same circle which represents a photon number resolving measurement. This scheme of generation allows us to disentangle the bus from the system without the need for extra operations. However, while choosing a suitable measurement is relatively simple in this case, it becomes more complicated when we consider combining operations. Measurements are also difficult to perform accurately, particularly when considering photon number resolving measurements, something which is still an experimental challenge. For most of our work on the qubus we will

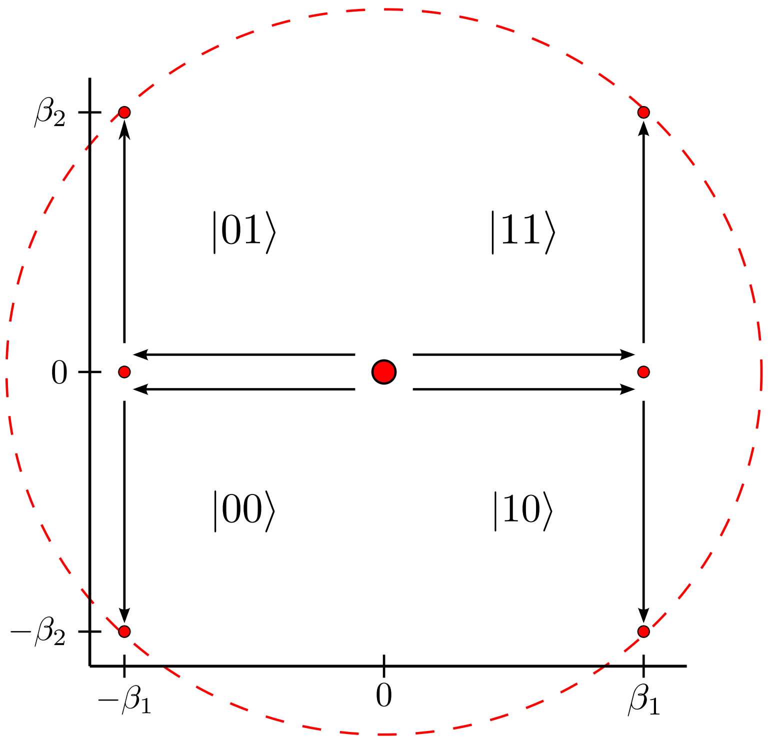


Figure 2.8: This figure shows the movement of the bus through phase space if we perform only the operations in figure 2.7. We see that the resultant states are all on the circle shown in red, which represents states with the same result for a photon number resolving measurement.

consider the deterministic measurement-free scheme. However in some of our work in section 4.4 we will also consider a scheme that uses measurement.

2.3.4 Using controlled rotations to generate controlled displacements

Since the controlled rotation operations are created using a standard Jaynes-Cummings interaction, these are often easier to perform than our controlled displacement operation. Van Loock et al. [3] demonstrated that it was possible to use these controlled rotations, combined with uncontrolled displacements to deterministically generate a controlled displacement. A suitable

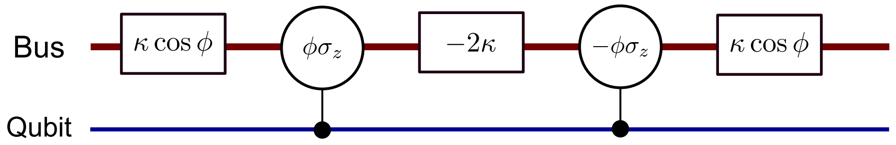


Figure 2.9: This figure shows how to build a controlled displacement operator from uncontrolled displacement (shown as rectangular gates), and controlled rotation operations (shown as circular gates).

circuit is shown in figure 2.9. This circuit consists of the operations

$$U = D(\kappa \cos\phi) e^{-i\phi\sigma_z a^\dagger a} D(-2\kappa) e^{i\phi\sigma_z a^\dagger a} D(\kappa\cos\phi)\,. \tag{2.26}$$

Van Loock et al. [3] introduce the identity

$$e^{-i\phi\sigma_z a^\dagger a} D(-2\kappa) e^{i\phi\sigma_z a^\dagger a} = D(-2\kappa e^{-i\phi\sigma_z}) \tag{2.27}$$

Therefore we can see that figure 2.9 gives a controlled displacement given by

$$U = D(2i\kappa \sin\phi\sigma_z)\,. \tag{2.28}$$

This gives an alternative technique which might be more physically feasible for performing our controlled displacements, particularly in a system based on quantum optics.

2.3.5 Combining several operations

A significant advantage of our qubus system is that we can change the order in which we interact qubits with the bus to reduce the total number of operations required. This allows us to generate large amounts of bipartite entanglement in relatively few operations. The qubus allows us to entangle several qubits to one quadrature without generating any entanglement between them. We can therefore entangle two sets of qubits in $2N$ operations, where N is the number of qubits in both sets. In fact we find that generating large amounts of entanglement using our qubus is relatively simple, with the challenge being how to restrict this entanglement so that it is useful.

The concept of saving operations by interacting multiple qubits with one quadrature of the bus was first discussed by Louis et al. [25], who showed how to get optimal improvements for star graphs and for chains of qubits. We will outline the two schemes which they present here. While the star graph scheme is not applicable to later work, the results on chains of qubits are useful both in their original context of generating cluster state, and for simulating the BCS Hamiltonian.

Louis et al. [25] note that it is possible to generate a star shaped graph using a sequence of displacement operators such that

$$\prod_{n=2}^{N} D(-\beta\sigma_{z,n}) D(-i\beta\sigma_{z1}) \prod_{n=2}^{N} D(\beta\sigma_{z,n}) D(i\beta\sigma_{z1}) = \exp\left[-2i\beta^2\sigma_{z1}\left\{\sum_{n=2}^{N}\sigma_{z,n}\right\}\right] \tag{2.29}$$

provided $\beta = \sqrt{\pi/8}$. We will consider the fundamental unit of our interactions to be the controlled displacement operator, and so when counting operations a single controlled displacement will be considered to be a single operation. Experimentally, it may be possible that we choose to implement this controlled displacement through controlled rotations, as discussed in section 2.3.4. However in this work, we will assume that we can implement the controlled displacement directly. It would be relatively trivial to convert our results to take into consideration the number of rotation operations required. By looking at equation (2.29) we can see that $2N$ operations will be required to make a star graph of N qubits. This star graph consists of $N-1$ interaction gates between our qubits. If we were to use a naïve method to generate this graph, using 4 operations per gate of entanglement, a total of $4(N-1)$ operations would be required. We therefore see that we get roughly a factor of 2 saving for generating star shaped graphs.

One of the important things to note, is that if we were considering generating a scheme such as a cluster state, we would want to limit how many other qubits, a single qubit is entangled to. The scheme presented above generates a large number of extra links, and results in a too highly entangled system for many practical purposes. Louis et al. [25] discuss how to use this system to generate a chain of interactions. In particular, they consider an interaction sequence given by

$$\begin{aligned}D(-\beta\sigma_{z,N})D(-i\beta\sigma_{z,N-1})\ldots D(\beta\sigma_{z4})D(-\beta\sigma_{z2})D(i\beta\sigma_{z3})D(-i\beta\sigma_{z1})D(\beta\sigma_{z2})D(i\beta\sigma_{z1})\\ = \exp\left[2i\beta^2\sum_{n=1}^{N-1}(-1)^n\sigma_n\sigma_{n+1}\right]\end{aligned} \tag{2.30}$$

where once again $\beta = \sqrt{\pi/8}$. Similar to the above scheme, this requires only $2N$ operations compared to the $4(N-1)$ required for a direct naïve implementation. Therefore we get a factor of 2 improvement when generating a chain of qubits. If we consider a grid-like structure, such as the square lattice graph often used for a cluster state, it is easy to see that only $4N$ operations will be needed to generate all the necessary interactions using horizontal and vertical chains. Such a scheme would therefore be useful for cluster state generation. The details of this, and further developments of the work are discussed in section 8.2.3 and chapters 9 and 10.

We make one addition to the work of Louis et al. [25] here, and note that it is possible to remove the $(-1)^n$ in equation 2.30 by changing the sign of operations. In particular we consider

$$\begin{aligned}D(\beta\sigma_{z,N})D(i\beta\sigma_{z,4})\ldots D(\beta\sigma_{z,4})D(\beta\sigma_{z,2})D(-i\beta\sigma_{z3})D(-i\beta\sigma_{z1})D(-\beta\sigma_{z,2})D(i\beta\sigma_{z,1})\\ = \exp\left[2i\beta^2\sum_{n=1}^{N-1}\sigma_n\sigma_{n+1}\right]\end{aligned} \tag{2.31}$$

so that our sequence results in a positive interaction between the qubits. When we consider adaptations of the Louis scheme in chapters 4 and 9, it is this scheme that we will refer to. It is possible to get similar savings when performing a sequence of CNOT gates, provided we are considering a sequence of operations with multiple target qubits, and a single control qubit. In cases where we need to switch the target and control qubit, then no reductions would be possible.

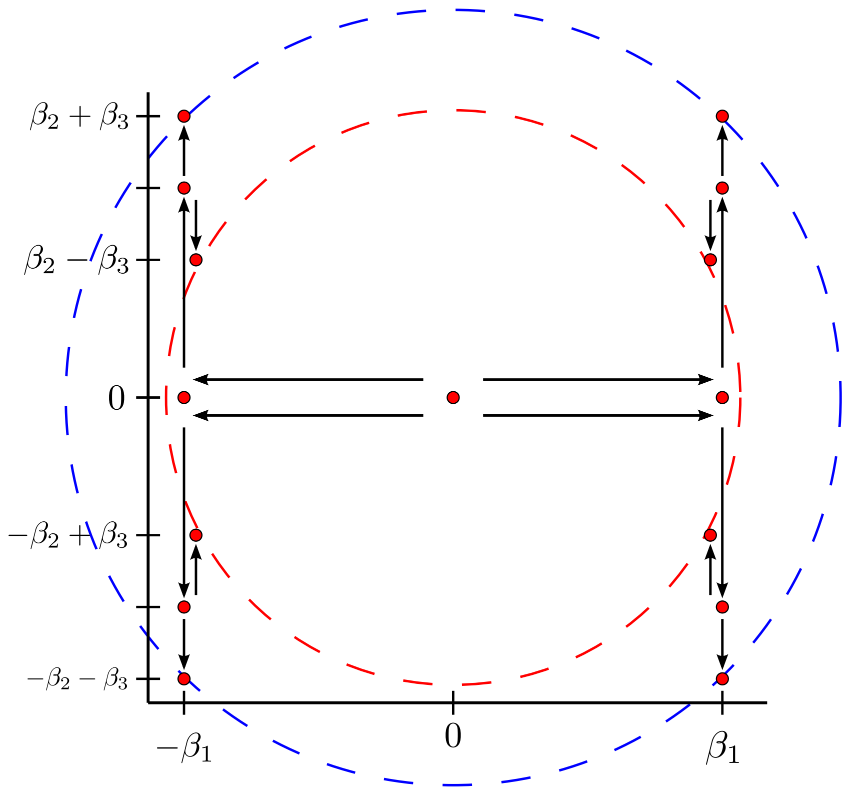


Figure 2.10: This figure shows the path through phase space traced out by equation (2.32). A photon number resolving measurement will distinguish the states aligned with the red circle, from those aligned with the blue circle, and therefore is not suitable in this case.

Finally, it is worth considering what happens to our measurement-based scheme described in section 2.3.3. In this scheme we reduced the number of operations required by replacing our disentangling operations with a measurement. We noted that a photon-resolving measurement would not allow us to distinguish the states, so we could measure the bus while generating entanglement between the qubits. We now consider a more complicated arrangement, such as a 3 qubit version of equation (2.31) which incorporates measurement. We consider a sequence of displacement operators given by

$$D(-i\beta\sigma_{z2})D(-i\beta\sigma_{z3})D(-\beta\sigma_{z1}) \tag{2.32}$$

where we measure after the final operation. From figure 2.10 we see that in this case a photon number resolving measurement would be able to distinguish some of our states, therefore this form of measurement is no longer suitable. In particular we note that we have two sets of states which would give a different result for a photon number measurement. We can clearly see that a measurement of either the position or momentum quadrature of the bus would also allow us to distinguish some of the states, therefore this would not be suitable. As we perform more operations on our bus, choosing a suitable measurement to disentangle the bus from our qubits without revealing any information and thus destroying the entanglement becomes more difficult. One possibility would be to form a budget measurement, which revealed whether we had no photons, or some photons. However, this might be error prone in the case of large photon number. In section 4.4 we discuss measurement-based schemes for reducing the number of operations required. In some cases where only two operations are left on the bus before we disentangle, it would be possible to use a photon number resolving measurement. However, in many cases a budget measurement would be required. In either case, the need for perfect measurement means that our measurement-based schemes are highly theoretical.

The savings we obtain by rearranging operations, such as in the Louis case, occur when we consider a net operation which contains multiple repeats of an interaction on a particular qubit. The degree of saving available depends upon the number of repeated terms, and how often these terms are repeated. While it is technically possible to get these savings in a measurement-based scheme, the addition of extra operations removes some of the symmetry from our system, and choosing a suitable measurement becomes a lot more difficult. Therefore, for any practical implementation, a measurement-free scheme would be the best solution.

2.3.6 Performing local unitaries using the ancilla

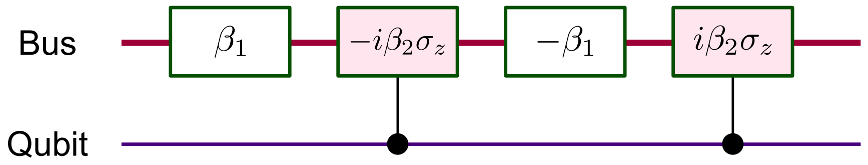


Figure 2.11: This figure shows how to use the qubus to generate an arbitrary rotation of a qubit around the Z-axis of the Bloch sphere. We do not need to be able to perform local unitaries on our qubit, but we do need to be able to perform general displacements on the qubus.

So far, all of our work has concentrated on using the qubus to generate entanglement between two qubits. However for universality we also need to be able to perform single qubit operations. Previously we assumed that our qubits could be chosen such that they have individual addressability, however we will now discuss what happens if this is not the case, and we need to use our qubus to generate the required local unitaries. We can perform any local unitary of the form $\exp(i2\beta_1\beta_2\sigma_z)$ using the circuit given in figure 2.11.

If we wanted to convert this gate into a local unitary based on the Pauli X, or Pauli Y operations then we would need to use local unitaries on either side of our operation set. To convert from Pauli Z to Pauli Y these local unitaries would be given by

$$U_{L1} = \exp(i\pi\sigma_x/4) \tag{2.33}$$
$$U_{L2} = \exp(-i\pi\sigma_x/4) \tag{2.34}$$

While to convert from the Pauli Z to the Pauli X, the same unitaries would be used except with the σ_x replaced by a σ_y. We therefore see a significant problem, in order to get a local unitary that is a component of a Pauli operator, other than the Pauli Z, we need to be able to perform local unitaries. This implies that to perform a general local unitary, we need to be able to perform general local unitaries - an unwelcome tautology. However, a standard result [33] shows that any local unitary can be written in the form

$$U_L = e^{i\alpha}R_z(\gamma)R_y(\delta)R_z(\epsilon) \tag{2.35}$$

where

$$R_z(\theta) = \exp(-i\theta\sigma_z/2) \tag{2.36}$$
$$R_y(\theta) = \exp(-i\theta\sigma_y/2) \tag{2.37}$$

and α, γ, δ and ϵ are real constants. Therefore, provided we can perform the local unitaries in equations (2.33) and (2.34) we can perform any local unitary on our qubits.

A disadvantage of this technique is that our local unitaries are expensive, with each operation requiring 12 bus operations, 4 for each rotation. We would also need 2 local operations, bringing the total number of operations required up to 14. The global phase factor does not affect the result of the final measurement, we therefore do not discuss how to implement it here. It should however be possible to add it in through global operations if necessary. To implement our local unitary, we consider performing a sequence of operations such as

$$R_z(\gamma) = D(\beta_1)D(i\beta_2\sigma_z)D(-\beta_1)D(-i\beta_2\sigma_{z3}) \tag{2.38}$$
$$R_y(\delta) = \exp(i\pi\sigma_x/4)D(\beta_3)D(i\beta_4\sigma_z)D(-\beta_3)D(-i\beta_4\sigma_z)\exp(-i\pi\sigma_x/4) \tag{2.39}$$
$$R_z(\epsilon) = D(\beta_5)D(i\beta_6\sigma_z)D(-\beta_5)D(-i\beta_6\sigma_z) \tag{2.40}$$

where $\beta_1\beta_2 = \gamma/2$, $\beta_3\beta_4 = \delta/2$ and $\beta_5\beta_6 = \epsilon/2$.

In order to be able to perform local operations using our qubus quantum computer as a mediating system, we need to be able to perform direct local operations. This seems to make our qubus superfluous for this purpose. However, we note that only one local unitary $U_{L1} = \exp(i\pi\sigma_x/4)$ and its Hermitian transpose are required to generate any local unitary, provided we can perform general displacements on the bus. In the physical systems which we would use to create the qubus, it is hoped we could perform these local unitaries directly. However if this is not possible we can see that with limited additional resources, we can create these by using our bus.

2.4 Experimental schemes for implementing the qubus

We now want to look at what physical systems it might be feasible to use, to create our qubus quantum computer. Spiller at al. [2] suggest using either a super-conducting, or ion trap system for the qubits as these should allow the necessary interactions. Van Loock et al. [3] propose using an optical system for the bus, and a matter system for the qubits. Controlled rotation operations are used to build up controlled displacement operations. This is something we discussed in section 2.3.4. The advantage of this is that the controlled rotations can be built from the readily available Jaynes-Cummings type interactions, which should be possible to implement in any quantum optical system. This suggests that it should be experimentally possible to use a light field as our bus.

Quantum dots are a hopeful new architecture for quantum computing, and are another architecture where it is potentially possible to implement a qubus style scheme. Yamamoto et al. [34] show a potential scheme for implementation, where the quantum dots are in a 2D lattice, which is embedded in a micro-cavity. An optical pulse, slightly de-tuned from a single quantum dots excitation energy is used to generate entanglement by shining on two qubits at once. The disadvantage of this gate is the two qubits have to be next to each other. They also consider making a gate using a single operation, so this is very different from our qubus scheme, since we could not rearrange the operations to get savings in the resources required, as we do throughout this work. Although Yamamoto et al. [34] propose other bus based schemes, these are reliant on measurement, and do not allow the majority of the schemes we have suggested to be implemented. This is because the bus consists of a single mode, rather than the two modes of the qubus, so we are limited to the gates discussed in section 2.2.

A similar scheme to the one proposed by Yamamoto et al. but using flux qubits and a resonator was proposed by Rodrigues et al. [35]. Once again this scheme uses measurement to generate probabilistic entanglement. Huo et al. [36] have shown a method for entanglement which can be turned on and off using a bus that is a super-conducting transmission line resonator. Unfortunately they still seem to be limited to a single bus mode, so can not perform the general gates which are available on the qubus. While these schemes are still limited in what they can perform compared to our theoretical qubus, they are still useful examples of possible ways of coupling continuous variable fields with matter qubits. Experimentally such schemes are still in their early days. However Sillanpää et al. [37] show how to couple two phase qubits to a resonant cavity, and Majer et al. [38] experimentally couple two qubits using a mediating bus of microwave photons. This work has been developed further to demonstrate quantum algorithms [39], and three qubit entangled states [40].

Manufacturing a qubus quantum computer, which can perform the necessary controlled gates to produce our deterministic C-Phase and CNOT gates, is an experimental challenge that is still being worked on. This discussion is far from a comprehensive review on the subject however we demonstrate that there are numerous schemes in existence that show coupling qubits to a continuous variable field is experimentally feasible. As an experimentally realisable qubus quantum computer is still a long way off, we will concentrate on an architecture independent implementation. This can show us efficient methods of generating the required operations. However to carry out a full error analysis, a model architecture is required. It would be possible to develop this work by providing a full summary of a practical architecture, including an error model, then investigating the best way to perform gates given this.

2.5 Conclusions

In this section we have introduced the qubus quantum computer, and showed how to use it to perform gates that will be useful for quantum computing. In section 2.2 we introduced some of the first proposed gates, which are possible to perform on the qubus system [2]. This included a probabilistic technique for creating a parity gate, and a deterministic but imperfect technique for creating a standard maximally entangling gate, such as the C-Phase. Although it was possible to adapt the parity gate to make it perfect and deterministic, it is not a particularly useful gate for the quantum circuit model, while the C-Phase gate was error-prone even in the ideal case. Therefore, we considered an alternative technique for creating gates that involved using controlled displacements.

In section 2.3 we introduced the idea of controlled displacement operations, and showed that they could be used to generate a probabilistic parity gate, in a similar fashion to our controlled rotation operations [2]. After this we introduced two deterministic and perfect in the ideal case ways to create a standard C-Phase gate. One of these techniques was measurement-free, while the second involved measurement. We also discussed a way proposed by Van Loock et al. [3] to make our controlled displacement operations from controlled rotations. In section 2.3.5, we explored the techniques used by Louis et al. [25] to reduce the number of operations required. While this was trivial in our measurement-free case, in the measurement-based case it became difficult to choose a suitable measurement of the bus which would not erase some of the entanglement. This suggests that the measurement-free case is the more physically feasible of the two. We also briefly introduced a technique for creating local operations using our qubus. Throughout this work we have assumed that we can directly perform an arbitrary rotation on a qubit. However if this was not possible, then it would be useful to use our entangling operations to generate the necessary local unitaries. This is similar to the work by Anders et al. [41] where a single operation with an ancilla qubit is used to create both entanglement, and single qubit unitaries.

Finally in section 2.4 we briefly discuss some of the physical systems which could possibly be used to build a qubus quantum computer. While these systems are still in the very early stages of development, we see that it is possible to couple two qubits to a single field within super-conducting flux qubits. The main disadvantage is that none of the current experiments use the two quadratures of the field in the way we do, therefore they can only generate entanglement in a probabilistic fashion. At the same time, these results do suggest that it should be feasible to use a continuous variable as a mediating bus, therefore the qubus should be physically possible.

While a large amount of the theory for creating gates for the qubus has been ironed out, and in the ideal case, a gate can be created in a error-free deterministic fashion, the experimental work still has a long way to go to catch up. Until a particular experimental system is chosen it will be difficult to create a complete error model for the qubus system. Our work on the qubus could be developed further by choosing this realistic error model, and seeing how it affects the results from later chapters. However, for now we will conclude that the qubus architecture of quantum computing has the potential to be a useful ancilla style system.

Chapter 3

Quantum simulation

3.1 Introduction

In this chapter we will introduce the idea of using a quantum computer to simulate a quantum system. Simulating a quantum system on a classical computer requires resources which grow exponentially with the size of the system being simulated. One of the largest completely general simulations of a quantum system that has been carried out on a classical computer consisted of only 36 qubits [6]. It is possible to simulate larger quantum systems if we take into account the symmetries within the system, however we are still often limited to simulating quantum systems of hundreds of qubits. This suggests that an alternative technique is needed. In 1982 Feynman proposed using one quantum system to simulate another [4]. This would allow the state of the system to be stored efficiently within the computer. However even this simple requirement, is not possible in the classical case. While Feynman's work paved the way for quantum simulation, it did not show that implementing the necessary operations on our quantum computer would be efficient. Lloyd built upon this work, and demonstrated that it should also be possible to implement the necessary time evolution efficiently in a large number of physical cases [5].

This early work on quantum computing has attracted a large amount of interest from numerous sources, for a recent review see [10]. The field is of particular interest because the break even point, where quantum computers outperform classical computers, is far lower for quantum simulation than for other algorithms such as Shor's algorithm [1]. This suggests quantum simulation will be one of the earliest practical uses for a quantum computer. Obviously, one difficulty with quantum simulation is that we are considering a problem where the answer can not be verified efficiently. Therefore we need to consider data extraction procedures which give the correct answer with high probability. Suitable data extraction procedures exist, however this suggests that error correction might be even more important for simulation than for other algorithms.

In order to conduct a complete simulation, three separate processes are required. The first is creating a suitable initial state. This is discussed in section 3.2, and is QMA-complete in the general case. QMA stands for quantum Merlin Arthur, and represents a problem which requires exponential resources to solve on a quantum computer, but whose solution can be verified in a polynomial time with high success probability. The 'complete' refers to the fact that any other QMA problem can be mapped onto this one, therefore solving it efficiently would mean that any QMA problem could be solved efficiently on a quantum computer. This shows very clearly that arbitrary state generation is a difficult problem. At the same time, in section 3.2 we do show some techniques which can be used for creating useful physical states, and discuss some of the limits on what can be created. Creating a suitable initial state is possibly the most difficult aspect of quantum simulation due to the limitations of state preparation techniques. However, by choosing other aspects of the algorithm carefully, it should be possible to get around this difficulty.

In section 3.3 we introduce some time evolution procedures that can be used to conduct the necessary simulation. Despite being the most important, this is perhaps the simplest of our three phases. In this section we discuss in detail Lloyd's method [5] of implementing the necessary time evolution, and also describe some alternative methods including direct Hamiltonian simulation and the pseudo-spectral method. The results of this section show that implementing the time evolution of a physical Hamiltonian is efficient in a large proportion of cases.

The final stage of a quantum simulation is data extraction. We discuss this in section 3.4 where we discuss the workings of two data extraction procedures - the phase estimation algorithm [23], and a technique for extracting correlation functions [42]. As a result we show that it is possible to extract useful data from our system in an efficient fashion. Although once again, we encounter problems when trying to increase the precision of our result.

In section 4.6 we explore some of the difficulties with quantum simulation, that do not occur in classical simulation. In particular, we note that this state, and the data extraction processes often require an exponential increase in time to get an exponential increase in precision. While this scaling is still linear, this is exponentially worse than in a classical digital computer where a linear increase in time is required to get an exponential increase in precision. We also briefly discuss the fact that quantum simulation requires a lower error scaling than classical computation. This is one of the most significant, and often under-explored disadvantages of quantum computing, and could significantly impact on other algorithms, as well as quantum simulation.

Finally, in section 3.6 and 3.7 we consider a simulation of a particular system - the BCS Hamiltonian. We discuss previous results which have been obtained from an NMR simulation [12–17] before discussing how to adapt this technique to be suitable for a more general architecture. In section 3.7 we see how the techniques we discussed in earlier sections come together to form a complete simulation of a Hamiltonian. We then conclude in section 3.8.

3.2 Initialisation

Unlike a lot of quantum algorithms, in quantum simulation the initial state is an unknown state which we are trying to characterise. In fact in many cases the purpose of the simulation is to work out data which would have already been found if we had easy access to the initial state. For example, when considering the simulation of the BCS Hamiltonian to extract the energy gap, Wu et al. [12] prepare the system in an initial state which consists of a superposition of the ground and first excited state of the Hamiltonian. If we knew what this state was, then we would not need to run the simulation as the necessary data would already be at hand. Creating a general unknown state, or even just an unknown ground state, is QMA-complete in the most general case. This provides us with the undesirable fact that preparing our state is a non-trivial problem. Indeed even if our state is known it is not always possible to prepare it efficiently. Luckily this is not as problematic as it first sounds. It is likely that we will want to limit our simulations to Hamiltonians which describe physically realistic systems, and this limits the level of generality necessary. Also, while state preparation is QMA-complete in the general case, there are a large selection of states which it is possible to create efficiently using several techniques. Since state preparation is a field of its own, we will only discuss the very basic results here. However even these demonstrate that there are many cases where a suitable state can be prepared for simulation.

3.2.1 Adiabatic evolution

One of the most commonly desired (unknown) starting states is the ground state of a Hamiltonian, or an approximation thereof. Once the ground state has been prepared it is possible to use this to find information about the eigenvalues and eigenvectors of said state [9], and other information about the low lying spectrum such as the energy gap [12]. Unfortunately such a task is QMA-complete to find in the general case, however it is often possible to prepare such a state using an adiabatic evolution [43].

In an adiabatic evolution, a well known and easy to find ground state of a Hamiltonian H_1 is prepared, a slow evolution procedure is then applied to switch from H_1 to H_2. This evolution can be represented by the equation

$$H_{ad} = (1 - s/S)H_1 + (s/S)H_2 \,, \tag{3.1}$$

We consider starting our evolution in the ground state of H_1. The adiabatic evolution Hamiltonian is applied repeatedly from $s = 0$ to $s = S$, and the overall result is the ground state of H_2. If the evolution procedure is applied suitably slowly, then the system remains in the ground state throughout, and the result is the ground state of H_2. However, if the evolution is applied too fast, the system becomes excited, and the resultant state contains components of the first, and sometimes other, excited states. The efficiency of the preparation depends upon how slowly the sequence needs to be implemented to ensure that the system remains in a ground state. In particular, we note that provided the energy gap does not become exponentially small, an adiabatic evolution can be implemented in polynomial time [44]. This suggests that it should be possible to use this state generation technique in a large proportion of cases where we want a ground state.

While this is a hopeful result, a lot of quantum simulation focuses on the simulation of phase-transitions, therefore it seems likely that exponentially small energy gaps might present a problem. This is an issue which has been explored by Dziarmaga and Rams [45] who show how to avoid a disruption to the adiabatic evolution in many interesting cases where there is an inhomogeneous phase transition. An inhomogeneous phase transition is one where the order parameter varies across the system. Hence the phase change occurs at different times in different parts of the system. Dziarmaga and Rams [45] show that provided the boundary between the two phases of the system moves slower than the local transitions within the relevant regions it is possible to perform the adiabatic evolution in polynomial time for a finite-sized system.

An interesting result from this work which is worth mentioning, is that it is possible to combine the state initialisation and phase estimation algorithm when it comes to finding eigenvalues and eigenvectors, see [46]. This could be used to find data about how the initial input conditions effect the ground state energy, and thus cycle through various models to find one which fits physical parameters. It should be possible to adapt such a technique to be relevant to our BCS Hamiltonian simulation in chapter 6. We could use this to characterise how the strength of the coupling between modes within the BCS Hamiltonian affects the energy gap.

3.2.2 Other methods of state preparation

While my work on the BCS Hamiltonian has concentrated on using an adiabatic method of state preparation, this is not the only technique for generating a quantum state that has been suggested. Indeed in many cases it is possible to simply prepare the desired state using direct state preparation. How useful this will be for quantum simulation will depend upon how many algorithms propose starting with a known initial state. However, in the simplest case we can see how it can be used with the adiabatic evolution above, since the ground state of H_1 would almost certainly be prepared directly.

Soklakov and Schack [47, 48] provide a method for efficiently generating a state, provided that the description of that state is suitably efficient. This has been optimised by Plesch and Brukner who work on reducing the number of CNOT gates required [49]. Unsurprisingly direct state preparation is QMA-complete in the general case, however Poulin and Wocjan [50] demonstrate how to reduce errors and get quadratic speed ups when preparing ground states. Once again we see that we can generate some of the desired states but we are strictly limited in what can be generated efficiently.

Finally, it is worth noting the amount of research which is going into preparing thermal equilibrium states. It would be useful to prepare these as an initial state to study temperature dependent properties of matter. However, once again we encounter the problem that these are unlikely to be efficient to prepare in the general case. Despite this issue a lot of research has looked at what it is possible to prepare, and how efficiently. One obvious technique is that of preparing the quantum state at the correct temperature by using a thermal bath. The thermal bath is a surrounding system used to keep the state being prepared at the correct temperature. Terhal and DiVincenzo [51] consider such a system and show that it is possible using a relatively small bath system. However they do not give bounds of the specific running time, and suggest that such a technique will not be efficient in the general case. Poulin and Wocjan [50] provide upper bounds on the running time of such thermalisation algorithms, and confirm that these might not be applicable in the general case. An alternative technique for preparing thermal states is using the quantum Metropolis algorithm [52] which uses samples from the energy eigenstates to produce a suitable thermal initial state. This should be efficient even when there are degenerate subspaces.

3.2.3 Conclusions

Generating a suitable initial state can be one of the most difficult parts of quantum simulation. For many problems, once the initial state has been prepared most of the computational difficulties have already been solved. Despite these problems, and the fact that preparing a ground state is QMA-complete in the general case, it should be possible to do this using adiabatic evolution provided the energy gap is not exponentially small. Recent results also suggest that it will be possible to get around problems with adiabatic evolution, in cases where there is an inhomogeneous phase-transition. This suggests that provided we are particular about which Hamiltonians interest us, there is a reasonable set for which we can obtain results about the ground state, and even the low lying spectrum.

Alternative techniques for state preparation include direct preparation, and preparing thermal states using thermalization or the quantum Metropolis algorithm. This suggests that our initial state is not simply limited to ground state preparation, and that a selection of other states could be prepared.

While quantum state preparation remains a challenging problem, the set of states required as an initial state for our quantum computer should be a simplified sub-set of these problems. Algorithms which require the preparation of a generic initial state are in danger of pushing the hard part of the problem to the initialisation, rather than removing the exponential scaling. Therefore, while the need for an initial state does provide a significant restriction on what it is possible to simulate, it should still be possible to conduct many useful quantum simulations.

3.3 Time evolution techniques

While Feynman [4] discussed the possibility of using a quantum computer for simulating quantum systems, most of his reasoning was based on storing the state, since the memory required to do this grew exponentially with the size of the system on a classical computer, but scaled efficiently on a quantum computer. However, in order to conduct a simulation it is also important to be able to implement the desired unitaries in an efficient fashion. Lloyd [5] was the first person to flesh out Feynman's ideas, and look at the practicality of implementing the time evolution. This technique of splitting the Hamiltonian into a sum of l-local Hamiltonians has attracted a lot of interest. In particular, it is the technique that has been used for simulating the BCS Hamiltonian [12–17] in work that is discussed in more detail in section 3.7. We will also briefly discuss two other techniques for simulation. Firstly, in section 3.3.2 we will discuss direct Hamiltonian simulation where a known Hamiltonian of the system is used to form the desired Hamiltonian. Secondly in section 3.3.3 we will look at the spectral method which involves applying diagonal operators in two bases, and using a Fourier transform to switch between them. This technique is of particular interest since in section 6.2 we show how to get speed ups in the quantum Fourier transform by using a qubus computer.

3.3.1 Lloyd's method

Lloyd shows that by turning on and off a sequence of Hamiltonians, it should be possible to simulate any quantum system [5]. By decomposing the unitary operator into a sequence of standard quantum gates, Vartiainen et al [53] provide a method for doing this within the gate model of quantum computing. However, to specify an arbitrary unitary operator requires an exponential number of parameters, and therefore would require an exponential amount of time to simulate, in both the quantum and classical case. Therefore, an arbitrary quantum simulation is not necessarily efficient using Lloyd's technique.

Luckily, physically realisable quantum systems are only a limited subset of the general case. In particular any system that is consistent with special and general relativity evolves according to local interactions. As a result they can be written in the form

$$H = \sum_{j=1}^{\ell} H_j \tag{3.2}$$

where each of the ℓ local Hamiltonians H_j act on a limited space containing at most k of the total of N variables. By 'local' we only mean that k remains fixed as N increases, we do not require that the k variables are physically local.

In the same way that classical simulation of the time evolution of dynamical systems is often performed, the total simulation time t is divided up into k small discrete steps. Each step is approximated using a Trotter-Suzuki [54, 55] formula,

$$e^{iHt} = (e^{iH_1t/k} \dots {}^{iH_\ell t/k})^k + \sum_{i>j} [H_i, H_j]t^2/2k + \text{higher order terms.} \tag{3.3}$$

The higher order term, of order p is bounded by $k||Ht/k||^p_{\text{sup}}/p!$, where $||A||_{sup}$ is the supremum, or maximum expectation value, of the operator A over the states of interest. The total error is less than $||k(e^{iHt/k}-1-iHt/k)||_{\text{sup}}$ if just the first term in equation (3.3) is used to approximate e^{iHt}. By taking k to be sufficiently large the error can be made as small as required. For a given error ϵ, from the second term in equation (3.3) we have $\epsilon \propto t^2/k$. A first order Trotter-Suzuki simulation thus requires $k \propto t^2/\epsilon$.

Now we can check that the simulation is efficient in the number of operations required. Simulating $e^{iH_jt/k}$ requires m_j^2 operations where $m_j \leq m$ is the dimension of the variables involved in H_j. In equation (3.3), each local operator H_j is simulated k times. Therefore, the total number of operations required for simulating e^{iHt} is bounded by $k\ell m^2$, where m is the largest m_j. Using $k \propto t^2/\epsilon$, the number of operations is proportional to $t^2\ell m^2/\epsilon$. The only dependence on N is in ℓ, and we already determined that ℓ is polynomial in N, so the number of operations is indeed efficient by the criterion of polynomial scaling in the problem size.

Lloyd's straightforward, but very general, method laid the groundwork for subsequent quantum simulation development, by providing conditions under which it will be possible in theory to carry out efficient quantum simulation, and describing an explicit method for doing this.

3.3.2 Direct Hamiltonian simulation

A major difficult with the method proposed by Lloyd [5] is the error due to the Trotter approximation, and the fact that an exponential increase in precision requires an exponential increase in the time required to implement the simulation. One way around this would be a direct simulation where we use local unitaries, and the natural Hamiltonian of the system we are considering to generate the Hamiltonian of interest. For more complex systems we can combine this direct simulation with the approach of Lloyd to create the desired Hamiltonian.

The technique of using a known Hamiltonian to directly simulate another Hamiltonian, is well know experimentally from NMR computing, where a fixed Hamiltonian is used to generate the necessary operations. This mapping of one Hamiltonian to another has been proved universal for quantum computing by Dodd et al. [56, 57]. In particular this work shows the universality of 2-body entangling Hamiltonians acting on N qubits for simulating other 2-body entangling Hamiltonians acting on N qubits, when combined with local unitaries. While this work is significant in the field of quantum simulation, particularly because many physical systems can be described by two body interactions, this still has strong overlaps with the gate model and the focus of the papers is on universal computation rather than direct simulation. However, the same group has expanded this research to look at qubit entangling Hamiltonians which couple more than 2 bodies [58], qudit entangling Hamiltonians acting on 2-body [59], and qudit entangling Hamiltonians acting on multi-body systems [60]. This work showed the universality of all multi-body qudit entangling Hamiltonians for simulating a system of the same size, with the exception of a qubit entangling Hamiltonian acting on an odd number of systems [60]. This exception did not prove too detrimental, as with correct mapping it was possible to use an N-qubit entangling Hamiltonians acting on an odd number of systems to simulate an $N-1$ qubit Hamiltonian. Apart from the one exception, this work proved the universality of Hamiltonians for quantum simulation, however it is far removed from experimental implementations as it does not describe how to determine a control sequence, let alone implement one. Unfortunately, this highlights one of the practical difficulties of Hamiltonian simulation, and Wocjan et al. [61–63] have shown that the optimal control sequence is hard to obtain in a general case. This is obviously detrimental when considering a general simulation, however it is worth noting that one of the reasons we are interested in these direct simulations is because we have an architecture that is described by a natural Hamiltonian, and we want to simulate a Hamiltonian close to this.

By moving away from the idea of a universal quantum simulator, and hence a universal quantum computer we can begin to obtain interesting simulation results, which do not make any claims about the universality of the resources being used. While still theoretical, these results have a more experimental leaning looking at the practicalities of potentially feasible physical architectures for simulation of very specific Hamiltonians. In particular, work on ion traps and optical lattices has looked at the possibility of simulating models such as the Bose-Hubbard models [64–66], and the Ising and Heisenberg interactions [67–70], while work on NMR has concentrated on behaviours within a spin chain acting according to the Heisenberg interaction [71–74]. The more advanced nature of research on NMR quantum computers has shown how to use this system to simulate many body systems such as superconductivity [12, 13, 75], however these more advanced simulations require the Trotter procedure discussed by Lloyd [5] and therefore are not a full direct simulation.

In summary, the necessary control sequence for turning one general entangling Hamiltonian into another is hard to compute in the general case. However, when we begin to consider turning one simple physical Hamiltonian into another this becomes more realistic. Therefore, there are numerous papers which explore how to use a physical Hamiltonian of a particular architecture to simulate similar Hamiltonians. This suggests that the earliest quantum simulations will be using a physical system to simulate the behaviour of the same, or very similar physical system, in a different regime. While this is a limited result, it would be important in showing the feasibility of using a quantum computer for simulation purposes.

3.3.3 The pseudo-spectral method

A common classical simulation method is the pseudo-spectral method, this involves using the fast Fourier transform to switch between the position and momentum representation of a desired interaction, allowing the necessary interaction to be built up by applying diagonal operators in the correct representation. Despite the fact that the fast Fourier transform requires a significant computational cost, this technique has proved popular due to the speed and accuracy of the simulation. The quantum pseduo-spectral method uses the same concept, taking advantage of the exponential speed up of the quantum Fourier transform compared to the fast Fourier transform. In particular, particles moving in external potentials often have Hamiltonians with terms that are diagonal in the position basis, and terms that are diagonal in the momentum basis. Evaluating these terms in their diagonal bases allows us to significantly simplify the computation. Wiesner [76] and Zalka [77, 78] gave the first detailed descriptions of this approach for particles moving in one spatial dimension, and showed that it can easily be generalized to a many particle Schrödinger equation in three dimensions.

Benenti and Strini [79] describe this technique in considerable detail, and also provide a quantitative analysis of the number of operations required for small simulations. They estimate that for present day capabilities of six to ten qubits, the number of operations required for a useful simulation is in the tens of thousands, which is many more than can currently be performed coherently. Nonetheless, the efficiency savings over the Lloyd method will make this the preferred option whenever the terms in the Hamiltonian can be diagonalised separately.

We mention this technique here briefly as it would be an ideal simulation to consider on our qubus quantum computer, since we show a significant speed up in performing the quantum Fourier transform compared to a naïve case, see chapter 6. The qubus also allows us to perform the basic diagonal gates such as the C-Phase gate, which should make it possible to implement the necessary diagonal operations. It is worth noting, that this technique would also be applicable to a large range of classical systems which are difficult to simulate in the classical case such as the Navier-Stoke's equation.

3.3.4 Conclusions

The possibilities of using a quantum computer for quantum simulation was first considered thoroughly by Lloyd [5] who demonstrated that it was possible to perform the necessary unitaries efficiently provided the desired Hamiltonian could be expressed as a sum of l-local Hamiltonians, where l does not increase with the size of the system. The major disadvantage of the Lloyd proposal is the need to recombine the individual Hamiltonians using the Trotter approximation. This leads to a poor scaling of precision, where an exponential increase in operations is required for an exponential increase in precision. This is discussed in more detail in section 3.5. One possible away around this problem is to use a direct simulation where the Hamiltonian of the physical system being used to build the computer is combined with local unitaries to implement the necessary Hamiltonian. In the cases where the mapping between the two Hamiltonians is relatively simple this technique works well, however in the majority of cases we still require the Trotter approximation.

There are numerous alternative techniques for implementing our time evolution using a quantum computer. We have discussed the pseudo-spectral method which involves applying diagonal operators in the position and momentum basis, and using a quantum Fourier transform to switch between the two. This gives us an exponential speed up over the equivalent classical simulation and can be useful in many cases, particularly because it should be faster than Lloyd's method. Work expanding Lloyd's method has looked at how to simulate noise on a quantum system, in particular how to use the noise of your quantum simulator to simulate noise on the system being simulated [5, 80]. Alternative techniques such as lattice gas automata [81–84] and simulations of quantum chaos [85–88] have been described in previous work, but are beyond the scope of this work.

3.4 Data extraction

When we talk about performing a simulation of a quantum system on a classical computer we are normally considering a strong simulation [89, 90]. This means that the whole probability distribution can be extracted using a single run of the computer. This is obviously not the same when we conduct our simulation on a quantum computer, where the amount of data we can get from a single run is restricted to a single measurement outcome. To make a fair comparison between a quantum simulation and a classical simulation, we should compare our quantum results with the results of a weak simulation conducted on our classical system. However weak classical simulations are still in the early days of research. Therefore we can not yet make a full comparison.

While we are extremely limited in what it should be possible to extract using a single run of our quantum computer, algorithms exist which result in the output of our computer being an interesting state with a high probability. In the best cases our computer should give the correct output with a single run of the computer. One of the best known extraction algorithms, which finds uses in other quantum algorithms such as Shor's factoring algorithm [1], is phase estimation. The phase estimation algorithm which is discussed in section 3.4.1, allows us to find eigenvalues, and details about the low lying spectrum of a Hamiltonian. In section 3.4.2 we introduce a technique for extracting correlation functions, this method defines the desired function as U and can be used for any correlation function which can be computed efficiently, and which is between operators that can be described as the sum of unitary operators.

3.4.1 The phase estimation algorithm

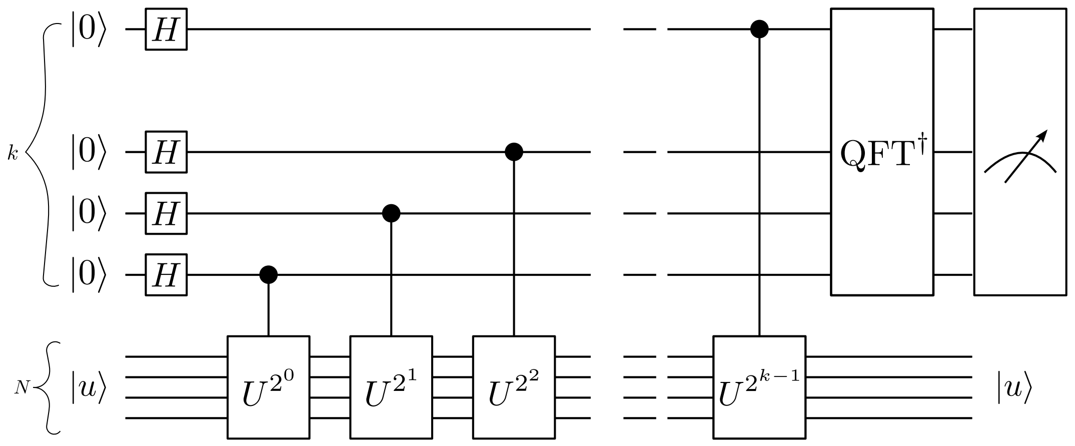


Figure 3.1: This circuit shows how to perform the phase estimation algorithm to find the eigenvalue of unitary U, where $|u\rangle$ is an eigenvector of U. We note that the register containing $|u\rangle$ remains in the same state, while our ancilla register gives us the necessary phase.

The phase estimation algorithm is one of the standard methods of data extraction for eigenvalues and eigenvectors [9], as well as in Shor's algorithm [1], and quantum computing in general. For simplicity we begin by discussing in detail a technique to use the phase-estimation procedure to extract the eigenvalue $\exp(i\Phi)$ of an eigenstate $|u\rangle$, of the unitary U. The circuit required to do this is illustrated in figure 3.1. Our system of qubits consists of two registers. The first contains N qubits and will be used to store the eigenvector of U, while the second consists of k 'readout' qubits which will contain our estimation of the phase Φ, once the calculation is complete. By performing multiple instances of U on our register qubits dependent upon a readout qubit, we are encoding information about the time evolution of U into this readout system. Once this information has been stored in our readout system, we can simply perform a quantum Fourier transform on these qubits to extract the necessary phase.

To illustrate the workings of the phase estimation algorithm, we will consider our first register to be in a single eigenstate of U, therefore the implementation of U will leave the state of the register unchanged. The readout qubits are prepared in the state $|+\rangle$. Implementing CU^{2^j}, a controlled form of U^{2^j}, on a readout qubit results in

$$CU^{2^j}\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)|u\rangle = \frac{1}{\sqrt{(2)}}\left(|0\rangle + e^{i2^j\Phi}|1\rangle\right)|u\rangle\,. \tag{3.4}$$

We now consider applying these gates from $j = 0$ to $j = k-1$ with the net result

$$\frac{1}{\sqrt{2^k}}\left(|0\rangle + e^{i2^{k-1}\Phi}|1\rangle\right)\ldots\left(|0\rangle + e^{i2\Phi}|1\rangle\right)\left(|0\rangle + e^{i\Phi}|1\rangle\right)|u\rangle = \frac{1}{\sqrt{2^k}}\sum_{y=0}^{2^k-1} e^{i\Phi y}|y\rangle|u\rangle \tag{3.5}$$

Tracing out the register containing the eigenstate leaves us with our k readout qubits in the state $\sum_{y=0}^{2^k-1} e^{i\Phi y}|y\rangle$. Applying the inverse quantum Fourier transform encodes the phase into the register of readout qubits. We can see this since the action of the quantum Fourier transform [33] on N qubits is given by

$$|a\rangle \rightarrow \frac{1}{\sqrt{N}}\sum_{b=0}^{N-1} e^{2\pi iab/N}|b\rangle\,. \tag{3.6}$$

Assuming that our register starts in an exact eigenstate, the final result of the phase estimation procedure is $|\tilde{\Phi}\rangle|u\rangle$, where $|\tilde{\Phi}\rangle$ is a k qubit approximation of $|\Phi\rangle$. If the first register starts in an approximation of the eigenstate, then the phase estimation algorithm results in the second register being transformed into $|\Phi\rangle$ with a probability that increases with the accuracy of the eigenstate and with k [33].

It is possible to extract data from our phase estimation algorithm efficiently. If our register of N qubits is initially in a single exact eigenstate, then the phase estimation algorithm encodes the phase into our ancilla qubits with certainty. Therefore only one measurement in the computational basis is required to extract the necessary data. By determining whether every single qubit is in $|0\rangle$ or $|1\rangle$ a integer, R, between 0 and $2^k - 1$ can be extracted. We then find that $\Phi = 2R\pi/2^k$, and the eigenvalue is $\exp(i\Phi)$. If our register of N qubits is initially in a superposition of numerous eigenstates, then the phase estimation algorithm will give each eigenvalue with a probability that depends on the fraction of the corresponding eigenstate in our initial superposition [33]. One thing to note is that, as the number of qubits in our readout register increases the probability of ending up in the eigenvalue that makes the largest contribution to our eigenstate also increases. As a result of this, for a larger number of readout qubits we need to further relax the adiabatic condition in our initialisation procedure. Even with a large number of qubits, it is easy to see that it should be possible to extract both the eigenvalues for our ground and first excited state using only a few runs of the quantum computer, therefore we could find the energy gap of such a system.

3.4.2 Correlation functions

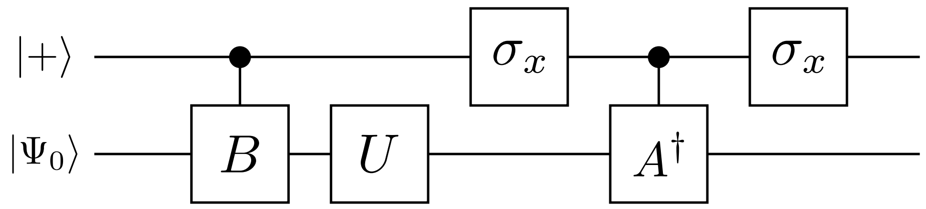


Figure 3.2: This figure shows a circuit that is suitable for calculating a correlation function U between systems A and B.

Another commonly used data extraction technique is the one proposed by Somma et al. [42] for finding correlation functions. These correlation functions take the form

$$C_{AB}(t) = \langle U^\dagger A U B \rangle \tag{3.7}$$

where A and B are expressible as a sum of unitary operators, and U is described by the form of correlation function being considered. In a standard case, where U describes a dynamical correlation function then it is the time evolution of the system. If we want to consider spatial correlation functions then we take $U = e^{-ipx}$ where p and x are operators in the spatial domain.

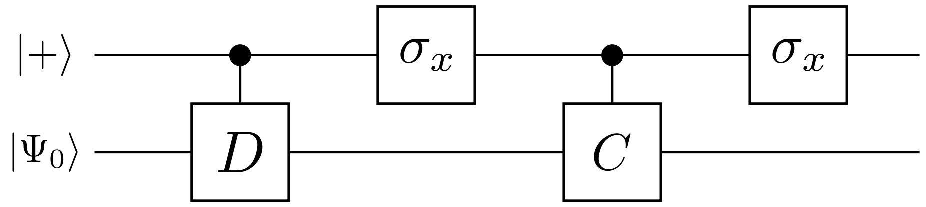


Figure 3.3: This figure shows a technique for calculating a simple correlation function of the form $\langle C^\dagger D \rangle$ where $C^\dagger$ and D are unitary operators.

Somma et al. [42] show two data extraction procedures, one which only uses a single qubit, and one which requires multiple qubits. We will briefly outline the process that Somma et al. [42] describe for a one ancilla data extraction procedure, this assumes that A and B are single unitary operators. This process is illustrated in figure 3.2. To extract a simple correlation function of the form $\langle C^\dagger D \rangle$ a circuit such as the one in figure 3.3 is required. The ancilla system is prepared in the state $|+\rangle$, and the main system in the initial state $|\Psi_0\rangle$. We then apply our unitary operators in a controlled fashion. In particular we apply D conditionally based on the ancilla qubit being in the state $|1\rangle$, and C based on the ancilla qubit being in the state $|0\rangle$. By measuring the expectation value of the ancilla qubit in the $|0\rangle$, $|1\rangle$ basis we can find the necessary correlation function. Even for the multi-qubit data extraction procedure, only a single qubit needs to be measured to obtain the required function [42].

We now want to use this circuit to extract our more complex correlation function from equation (3.7), we therefore take $C^\dagger = U^\dagger A$ and $D = UB$. To a calculate our correlation function, we apply the two operators $C = A^\dagger U$ and $D = UB$. We are now applying the operator U conditional on the ancilla being in state $|0\rangle$, then on the ancilla being in state $|1\rangle$, this is equivalent to just one single application of the operator U. Therefore, we see the result is the circuit in figure 3.2.

Somma et al. [42] provide a technique for adapting this to find correlation functions when A and B are sums of unitary operators. The disadvantage of this technique is the need to find the expectation value of the resultant ancilla system. However, even in the multi-qubit case it is only necessary to find the expectation value of a single ancilla qubit so this process should be efficient.

3.4.3 Conclusions

We have introduced two methods of data extraction, which allow useful information to be extracted efficiently from a quantum computer. The first technique we outlined, the phase estimation algorithm, is one of the most commonly used data extraction techniques in quantum computing. It is useful for a wide variety of purposes including finding eigenvalues, and energy gaps. It is this technique we consider as part of our work on quantum simulation. The second technique allows us to find information about the correlation functions between two operators, this is particularly interesting because even for a multi-qubit system, we only have to extract data from a single qubit. While phase estimation only requires one run of the computer in an ideal case, information about correlation functions will require multiple runs.

To increase the accuracy of the phase estimation technique we add an extra readout qubit, this gives us an exponential increase in precision, at the cost of an exponential increase in time resources. We can increase the accuracy of the correlation function technique by increasing the number of runs of the computer. In this case we find a linear increase in time resources will give us a linear increase in precision. Either way, we find this is exponentially worse than a classical computer, where a linear increase in resources is required to give an exponential increase in precision. This is one of the fundamental problems with quantum computing, and is discussed in more detail in section 3.5.

While we have only looked at two methods of data extraction, we have demonstrated that it is possible to use our quantum computer to obtain useful results efficiently. Obviously quantum state tomography is also a relevant technique for data extraction [91] and other direct characterisation techniques can be considered [92] although these are not always efficient. Either way, we see that it is possible to use our quantum computer to conduct a simulation, then extract useful data.

3.5 The limits of quantum simulation

Unfortunately, while efficient, Lloyd's method [5] highlights two of the most significant problems with quantum computing. The first is the poor scaling of precision with time, and the second is the scaling of errors. Both of these problems give results which are exponentially worse than a classical computer, but which are similar to classical analogue quantum computing, this has led to some work looking at using a quantum analogue computer [93] since they have a similar error scaling.

We will begin by considering performing our quantum simulation in an error-free ideal case. If we consider using Lloyd's method, we note that the accuracy of our result will be limited by the use of the Trotter-Suzuki approximation. While it would be possible to increase the accuracy, by increasing k or using higher order approximations, in either case we find that an exponential increase in time is required to get an exponential increase in precision. This is exponentially worse than when we use a digital computer, where a linear increase in time is required for an exponential increase in precision. While it might be possible to get around this using our direct simulation techniques, we actually find similar problems with data extraction.

In section 3.4 we introduced two data extraction protocols, the phase estimation algorithm, and another which gave correlation functions. In both cases we found that an exponential increase in operations was required to give an exponential increase in precision. In the case of the phase estimation algorithm, each additional readout qubit doubled the resources required, but also doubled the necessary precision. When we consider the procedure to extract correlation functions, then multiple runs of the computer are required to increase the precision, with a linear increase in the number of runs giving a linear increase in precision. In some cases it might be possible to parallelize operations, so our spatial, rather than temporal resources increase, however we are still seeing a performance exponentially worse than a classical computer.

When we begin to consider errors, then we see the other problem with quantum simulation. In particular Lloyd noted that to keep the total error below ϵ then each operation must have an error less than $\epsilon/(k\ell m^2)$. From our definition of ℓ we know that, $\ell = \text{poly}(N)$, therefore our maximum error scales inversely with a polynomial of N. However, in classical digital computation to keep the error below ϵ the resources required scale as $\log_2(1/\epsilon)$. If we contrast this to quantum computing, where the resources required scale as $t^2\ell m^2/\epsilon$, we can see our quantum computer requires a small error scaling per operation to get the same overall error.

The consequences of this were first discussed by Brown et al [13], who point out that all the work on error correction for quantum computation assumes a logarithmic scaling of errors with the size of the computation, and they experimentally verify that the errors do indeed scale inversely for an NMR implementation of quantum simulation. To correct these errors thus requires exponentially more operations for quantum simulation than for a typical (binary encoded) quantum computation of similar size and precision. This is potentially a major issue, once quantum simulations reach large enough sizes to solve useful problems. The time efficiency of the computation for any quantum simulation method will be worsened due to the error correction overheads. This problem is mitigated somewhat because we may not actually need such high precision for quantum simulation as we do for calculations involving integers, for example. However, Clark et al [94] conducted a resource analysis for a quantum simulation to find the ground state energy of the transverse Ising model performed on a circuit model quantum computer. They found that, even with modest precision, error correction requirements result in unfeasibly long simulations for systems that would be possible to simulate if error correction was not necessary. One of the major sources of inefficiency is the use of Trotterization, which limits the accuracy both through the approximation, and through the large number of steps τ requiring many control operations, each with its associated imperfections.

On the other hand, this analysis does not include potential savings that can be made when implementing the Lloyd method, such as by using parallel processing to compute simultaneously the terms that commute. Nonetheless, the unfavorable scaling of the error correction requirements with system size in quantum simulation remains an under-appreciated issue for all implementation methods.

3.6 About the BCS Hamiltonian

Now that we have introduced the idea of quantum simulation, we will explore in detail one particular quantum simulation, which we will work on later. By concentrating on a single Hamiltonian we get to see the inner workings of an algorithm, and also see adaptation that work on different forms of quantum computer.

In the next two sections, we will concentrate on simulating the BCS Hamiltonian. This Hamiltonian has attracted some interest in the quantum simulation community, with a lot of work focusing on simulating this Hamiltonian on an NMR computer [12–17]. For now we will introduce previous work done on the BCS Hamiltonian on an NMR computer, we will also consider how we can adapt this work to be suitable for implementation on other architectures of quantum computing. We concentrate on this Hamiltonian because of the large amount of work that has been previously done on it. We also consider it because it is a model of superconductivity, this is a subject which is still poorly understood, so a simulation of the BCS Hamiltonian should help with this. It should also be possible to map the BCS Hamiltonian onto other interesting physical systems.

Later in chapters 4, 6, 5, and 7, we will look at simulating the BCS Hamiltonian on a qubus computer, showing we get significant savings in the efficiency compared to the NMR computer. As part of this work we demonstrate how to use the qubus to do several important primitives such as phase estimation, and the quantum Fourier transform, generating long range interactions, and making operations controlled on an ancilla qubit.

The BCS Hamiltonian is a model of superconductivity proposed by Bardeen, Cooper and Schrieffer in 1957 [11]. The basis of the theory is the idea that electrons within a Debye energy of the Fermi Surface form bound pairs within a metal. These bound pairs of electrons behave in some respects as bosons, with pairing occurring between electron states with opposite momentum and spin to form a spin singlet state. In general this requires a many body description. To formulate the Hamiltonian, it is necessary to introduce a limit on the number of electrons through the chemical potential, μ, which constrains the average number of particles to be $\bar{N}$. If we assume that the electrons can be considered to be a Landau Fermi liquid, then the creation operator can be treated as the creator of quasiparticles rather than electrons. These quasiparticles can be considered to be an electron and a polarization-correlation cloud which is responsible for the motion of all the remaining electrons. This requires effective mass instead of the mass of an electron. Finally, to form the Hamiltonian, we restrict the wavefunction to the region which contributes to superconductivity, the attractive part that leads to formation of pairs with opposite momenta and spins. The BCS Hamiltonian is therefore given by

$$H_{\text{pair}} = \sum_{m=1}^{N} \frac{\epsilon_m}{2}(n_m^F + n_{-m}^F) + \sum_{m,l=1}^{N} V_{ml}^{+} C_m^{\dagger} C_{-m}^{\dagger} C_{-l} C_l \tag{3.8}$$

where $C_m^{\dagger}$ and C_m are the Fermionic creation, and annihilation operators, $n_{\pm m}^F = C_{\pm m}^{\dagger} C_{\pm m}$ are the number operators, N is an effective state number that represents the number of modes to be simulated, V_{ml}^{+} is the interaction potential, and ϵ_m is the on-site energy of a pair in mode m. Pairs of fermions have quantum numbers m and $-m$ where pairs have equal energies but opposite momentum and spin, $m = (\mathbf{p}, \uparrow)$ and $-m = (-\mathbf{p}, \downarrow)$. The conditions on m and l ensure energy conservation.

We are going to concentrate on extracting the energy gap from the BCS Hamiltonian. This quantity is interesting because it is a non-perturbative function, this means that the Hamiltonian needs to be exactly diagonalised to obtain an accurate value for the energy gap. The non-pertubative nature of the Hamiltonian also means that to get an accurate value for the energy gap, it is important to consider information from all interactions near the Fermi surface including long range interactions [95]. To get a solution for the energy gap Bardeen et al. [11] took $V_{ml}^{+} = -V$ for states near the Fermi surface, and 0 elsewhere. This approximation works for most metallic superconductors, but there are some cases where it is inaccurate. In these cases it would be useful to perform a simulation with a generalised V_{ml}^{+}. While there have been attempts to do this classically, many classical approximation methods either do not take into account these long range interactions, or require them to obey very specific rules [96–98]. Those which apply to the general case, such as the Richardson solution [99], can only be solved numerically in a limited number of cases. Very recent results obtained by Ho et al. [100] obtain the energy gap of the BCS Hamiltonian without the need for the BCS approximation. However these results are not as accurate for eigenvectors in the weak interaction regime. This means that in order to get an accurate result direct diagonalisation is often still necessary. However, it is resource intensive on a classical computer to diagonalise the exact Hamiltonians for more than a few tens of qubits. Therefore it is still difficult to characterise systems for cases where V^+_{ml} can not be held constant.

Previous work on an NMR quantum computer [12] has shown that it is possible to conduct a simulation of the BCS Hamiltonian efficiently, then to use this simulation to extract information about the energy gap. While Wu et al. [12] concentrate on a nearest neighbour case, they also demonstrate a case with general V^+_{ml}. We adapt this algorithm to be suitable for a qubus quantum computer, in particular this involves using the phase estimation algorithm for data extraction, rather than simply measuring in the computational basis. This significantly reduces the number of runs of the computer required when comparing a qubus system to an NMR system, even when the ensemble nature of NMR which allows the probability of a qubit being in a particular state to be extracted with a single run is taken into account.

This work on simulating the BCS Hamiltonian should be particularly applicable to small metallic grains, since it has yet to be confirmed that the BCS ansatz holds for these system. However the number of states within the Debye frequency cut off from the Fermi energy is in the region of one hundred [12, 95]. This suggests that it is a reasonable size system to simulate on an early quantum computer. The fact that results inaccessible to a classical computer are available for small systems, is one of the major benefits of quantum simulation, which should make it practical to implement before many of the more complex algorithms.

In the next section we introduce previous work looking at simulating the BCS Hamiltonian on a quantum computer. As part of this we outline the algorithm used, and discuss what adaptions need to be made to make this algorithm relevant to a qubus quantum computer

3.7 Previous quantum simulations of the BCS Hamiltonian

Previous work on simulating the BCS Hamiltonian on a quantum computer has concentrated on using an NMR computer. The two most significant pieces of work in this area have looked at finding the energy gap between the ground and first excited state of the Hamiltonian, and were conducted by Wu et al. [12] and Brown et al. [13]. Wu et al. [12] outline an algorithm for simulating the BCS Hamiltonian on an NMR quantum computer, and then extracting the low lying spectrum from this simulation. Brown et al. [13] build upon this work by analysing the errors due to the Trotter approximation, and the Fourier transform, as well as performing a simulation of the BCS Hamiltonian on a small NMR quantum computer and comparing the results with theory.

In order to find the energy gap of the BCS Hamiltonian, the computer is prepared in a superpositon of the ground and first excited state of the Hamiltonian. This is done using a quasi-adiabatic evolution from the ground state of simpler Hamiltonian. Wu et al. [12] then apply the time evolution procedure to this superposition of states, and thus generate a phase difference between the ground and first excited state. The advantages of the NMR quantum computer then come into their own, since by measuring a single spin the ensemble average can be extracted [13] from the system. For almost any other architecture of quantum computing, a large number of runs of the quantum computer would be required to extract the same data, and an alternative data extraction procedure that does not require the determination of probabilities is needed. This is a standard problem and a solution which works in this case is the phase estimation algorithm [23].

The first stage of simulating the BCS Hamiltonian on a quantum computer involves mapping the BCS Hamiltonian onto the qubit Hamiltonian. This mapping is worked out by Wu et al [12] and results in the BCS Hamiltonian taking the form:

$$H_{\text{BCS}} = \sum_{m=1}^{N} \frac{\varepsilon_m}{2} \sigma_{zm} + \sum_{m<l}^{N} \frac{V_{ml}}{2} \left(\sigma_{xm}\sigma_{xl} + r\sigma_{ym}\sigma_{yl} \right) \tag{3.9}$$

where σ_{xk}/ σ_{yk} are the Pauli X/Y on the k^{th} qubit, and r is a parameter determined by the mapping. Since $V_{ml}^{-} = 0$ we take $V_{ml}^{+} = V_{ml}$, and $\varepsilon_m = \epsilon_m + V_{mm}^{+}$. We note that, we can consider this to be the sum of two Hamiltonians, one consisting entirely of single qubit local unitaries,

$$H_0 = \sum_{m=1}^{N} \frac{\varepsilon_m}{2} \sigma_{zm} \tag{3.10}$$

and the other containing the two qubit terms

$$H_1 = H_{xx} + H_{yy} = \sum_{m<l}^{N} \frac{V_{ml}}{2} \left(\sigma_{xm}\sigma_{xl} + r\sigma_{ym}\sigma_{yl} \right) . \tag{3.11}$$

In this section we will outline the details of simulating the BCS Hamiltonian on an NMR quantum computer. We will then discuss the phase-estimation algorithm, and how to use this as an alternative to extract data from our system.

3.7.1 State initialisation

Extracting the energy gap of the BCS Hamiltonian, relies on first preparing our system in a superposition of the ground and first excited state of this Hamiltonian. This uses a quasi-adiabatic evolution from H_0 to H_{BCS}, this technique of initialisation is discussed in section 3.2.1. Wu et al. [12] choose a suitable ground state for H_0, with n qubits in the state $|1\rangle$ and $N-n$ qubits in the state $|0\rangle$. The ground state, and hence the energy gap vary depending upon n. The adiabatic evolution is implemented with the result

$$|\Psi(0)\rangle_0 = |g_n\rangle + \theta|e_n\rangle \tag{3.12}$$

where $|g_n\rangle$ is the ground state, and $|e_n\rangle$ is the first excited state given n and $\theta \ll 1$.

Wu et al. [12] now consider performing an additional rotation, so that this initial state also includes contributions from states with $n \pm 1$. This is done because the energy gap is dependent upon n, and Wu et al. are interested in probing states with different n. Brown et al. [13] do not consider this stage in their implementation, concentrating instead on finding the energy gap for a specific value of n. The resultant initial state after this rotation is given by

$$|\Psi(0)\rangle = e^{-i\phi\sigma_{y,\alpha}}|\Psi(0)\rangle_0 \approx |g_{n,n\pm1}\rangle + \theta'|e_{n,n\pm1}\rangle \tag{3.13}$$

where $\phi \ll 1$, $\theta' \ll 1$ and α is the index of the single measured qubit.

For simplicity, we will consider contributions from only a single value of n in our description of the algorithm. This will allow us to reproduce the results of Brown et al. [13]. An alternative method of preparing the initial state was proposed by Wang et al. [17]. This removes the requirement for the quasi-adiabatic evolution. Wang et al. show how to calculate an estimate of the low lying spectrum by estimating the lowest and highest eigenvalues. In our simulation, we will continue to use the adiabatic evolution as this does not require us to worry about the classical resources being used.

3.7.2 Performing the BCS Hamiltonian on the initial state

Now that we have an initial state we need to perform the BCS Hamiltonian, so that we develop a phase difference between the ground and first excited state. This involves implementing the BCS Hamiltonian on the initial state for various time intervals. Here we briefly outline how we go about implementing H_{BCS}. In section 3.7.3 we explain how to use the phase difference between the two states to find the energy gap.

Once it has been mapped onto spin operators the BCS Hamiltonian is given by

$$H_{BCS} = \sum_{m=1}^{N} \frac{\varepsilon_m}{2}\sigma_{zm} + \sum_{m<l}^{N} \frac{V_{ml}}{2}\left(\sigma_{xm}\sigma_{xl} + r\sigma_{ym}\sigma_{yl}\right) . \tag{3.14}$$

For implementing $U_{BCS} = \exp(it_0 H_{BCS})$ we consider implementing three separate unitaries, given by

$$U_0 = \exp\left(it_0 \sum_{m=1}^{N} \frac{\varepsilon_m}{2}\sigma_{zm}\right) \tag{3.15a}$$

$$U_{xx} = \exp\left(it_0 \sum_{m<l}^{N} \frac{V_{ml}}{2}\sigma_{xm}\sigma_{xl}\right) \tag{3.15b}$$

$$U_{yy} = \exp\left(it_0 \sum_{m<l}^{N} \frac{V_{ml}}{2} r\sigma_{ym}\sigma_{yl}\right) . \tag{3.15c}$$

The unitaries in equation (3.15) are non-commuting, therefore they need to be recombined using the Trotter-Suzuki approximation. When we are increasing the contribution of H_{BCS} to prepare the ground state in section 3.7.1, we use the first order approximation. For the main implementation of the BCS Hamiltonian Brown et al. [13] use the second order Trotter approximation which is given by

$$U_{BCS} \approx [U_0(t_0/2k)U_{xx}(t_0/2k)U_{yy}(t_0/k)U_{xx}(t_0/2k)U_0(t_0/2k)]^k \tag{3.16}$$

where t_0 is the total time of the simulation and k is the number of time intervals. This reduces the error to $\mathrm{O}(t_0^3/k^2)$, as opposed to $\mathrm{O}(t_0^2/k)$ in the first order case. For both levels of approximation, we see the problem discussed by Brown et al. [13] and discussed in detail in section 3.5 that an exponential increase in the number of gates is required to get an exponential increase in precision.

To implement the BCS Hamiltonian on an NMR computer, recoupling is used [12]. The Hamiltonian which describes an NMR system is given by

$$H_{NMR} = \sum_{l=1}^{N} \frac{\omega_l}{2}\sigma_{zl} + \sum_{l=1}^{N_1} J_l\sigma_{z,l}\sigma_{z,l+1} \tag{3.17}$$

where ω_l is the on-site energy and J is the coupling between qubits. By allowing the free evolution of this Hamiltonian, and supplementing this with pulses that flip the qubits as appropriate it is possible to build the desired interaction. Wu et al. [12] give an un-optimised technique for implementing the nearest-neighbour BCS Hamiltonian. They find that for each time step in the Trotter approximation $(28N^4 - 47N^3 - 4N^2 + 32N)/3$ operations are required.

It should be possible to get a significant reduction in the number of operations needed for numerous other architectures of quantum computing, as these will allow us to generate the Hamiltonian of interest rather than being restricted to a single Hamiltonian.

3.7.3 Data extraction using an NMR computer

Implementing the BCS Hamiltonian causes the ground and first excited state to evolve at different rates. We find that

$$|\Psi(t)\rangle_0 = U_{BCS}|\Psi(0)\rangle_0 = e^{-i\frac{E_g t}{\hbar}}|g_n\rangle + e^{-i\frac{E_e t}{\hbar}}|e_n\rangle = e^{-i\frac{E_e t}{\hbar}}\left(e^{i\frac{(E_e-E_g)t}{\hbar}}|g_n\rangle + |e_n\rangle\right) \quad (3.18)$$

Therefore the net result is a phase difference, which varies with time. By extracting data about the state of a qubit at various time intervals, it is possible to work out the period of the phase difference, and thus find the energy gap. In particular we note that when

$$e^{i\frac{(E_e-E_g)}{\hbar}t_1} = e^{i\frac{(E_e-E_g)}{\hbar}t_2} \quad (3.19)$$

then

$$e^{i\frac{(E_e-E_g)}{\hbar}t_2} = e^{i2\pi m}e^{i\frac{(E_e-E_g)}{\hbar}t_1} \quad (3.20)$$

where $m \in \mathbb{Z}$. We can therefore say that,

$$2\pi + \frac{(E_e - E_g)}{\hbar}t_1 = \frac{(E_e - E_g)}{\hbar}t_2 \Rightarrow \frac{(E_e - E_g)}{\hbar} = \frac{2\pi}{(t_2 - t_1)}. \quad (3.21)$$

Therefore by finding the period $(t_2 - t_1)$ we are able to find the energy gap.

On an NMR quantum computer, the easiest way to find the period and therefore the energy gap, is to perform the BCS Hamiltonian for various time intervals then measure a single spin to extract the ensemble average [13]. The result of this is a quantity which oscillates with a period 2Δ, where 2Δ is the energy gap. A classical Fourier transform is performed on this data, and the result is a graph of frequency with a peak at 0 and another at 2Δ [12].

3.7.4 The phase estimation algorithm

The method of data extraction discussed in section 3.7.3 is not applicable to architectures other than NMR, because numerous runs of any other quantum computer would be required to extract the ensemble average. In most cases this will not be efficient with the number of qubits in the system. As an alternative Brown et al. [13] propose using the phase-estimation algorithm, a form of data extraction which was previous discussed in section 3.4.1.

We want to consider using the phase estimation algorithm to extract the phase difference between two eigenvectors, so that we can determine the energy gap. In this case, the workings of the phase estimation algorithm can be seen as being essentially equivalent to the technique in section 3.7.3. In summary, information about the time evolution of the system is encoded into our readout qubits, then a quantum Fourier transform is used to extract the data.

We begin our algorithm by using the initialisation procedure in section 3.7.1 to prepare our first register in an superposition of the ground and first excited state of the BCS Hamiltonian. Our readout qubits start in the state $|0\rangle$, and are then prepared in $|+\rangle$ using Hadamard operations. We now use the phase estimation algorithm pictured in figure 3.1, to encode our eigenvalues into the register of readout qubits. This is done by implementing our unitary form of the BCS Hamiltonian in a controlled fashion dependent upon the state of the readout qubits. By performing $U^{2^{k-1}}$ where k is the number of the readout qubit. During this process we are encoding data about the time evolution of the BCS Hamiltonian for various time steps into our readout qubits. We then perform the inverse quantum Fourier transform on our readout qubits. This encodes the phase into these qubits in a readable format, and is equivalent to the technique in section 3.7.3, except for the fact that in this case we have additional data. Instead of getting two peaks, one at 0 and one at 2Δ we have two peaks, one at each eigenvalue. Once these have been found it is trivial to find the energy gap.

3.7.5 Conclusions

In conclusion, we have introduced the BCS Hamiltonian and adapted the techniques discussed by Wu et al. [12], to find an algorithm for simulating it on a general quantum computer. We note that the adiabatic state preparation process remains the same for all forms of quantum computing, while the time evolution and data extraction stages are different for NMR. In particular, the ensemble nature of NMR means we can extract the ensemble average with a single run of the computer. For any other architecture of quantum computing, a number of runs that scales exponentially with the size of the system would be required. As a result we switch from the data extraction procedure suggested by Wu et al. [12], to the phase estimation algorithm. This switch has been previously suggested by Brown et al. [13].

In this section, we have outlined how to perform all three stages of our quantum simulation efficiently. However, we note that the phase-estimation algorithm requires us to perform our unitaries in a controlled fashion. We do not discuss how to do this here, but do in chapter 5. In conclusion, in this section we describe how to combine the techniques given in the previous sections to implement a useful simulation, and extract some interesting data (the energy gap).

3.8 Conclusions

In this chapter we introduced the idea of using a quantum computer to simulate a quantum system. In the first three sections, we concentrated on the three main process involved with quantum simulation, demonstrating that it was possible to perform them all efficiently on a quantum computer in certain specific cases. Possibly the hardest part of simulation is generating a suitable initial state, this is QMA-complete in the general case. In section 3.2 we briefly outlined a few techniques that can be used to create some useful initial state, this includes certain ground states, and some thermal states. While it is unlikely that we will be able to create a general state efficiently, the techniques provided should allow us to create useful states for starting a simulation.

The second process in quantum simulation is the time evolution of the system, we discussed this in detail in section 3.3. We introduced three methods of implementing the time evolution, the first was Lloyd's method [5], which involved splitting our Hamiltonian using the Trotter approximation, the second was a direct simulation, while the third was the pseudo-spectral method. We found that Lloyd's method resulted in significant problems with precision, we discussed this in section 3.5. In this section we noted that in general an exponential increase in time was required, to get an exponential increase in precision. A similar problem can be seen in section 3.4, where we looked at data extraction procedures. In this section we described two techniques for data extraction, and showed that in both cases an exponential increase in operations was required for an exponential increase in precision. While precision scales linearly with time, standard classical simulations have results where the precision scales logarithmically with time. Therefore our quantum case is exponentially worse than our classical case. This poor scaling of precision is partly due to the need to use the Trotter approximation, and partly due to data extraction procedures.

Finally we have outlined procedures for simulating the BCS Hamiltonian on a quantum computer. In section 3.6 we introduced the BCS Hamiltonian, and explain some of the motivations for simulating it, while in section 3.7 we discussed in detail the algorithm for this one simulation. This puts our previous work in a practical context.

In conclusion, using a quantum computer to simulate a quantum system will be one of the first practical uses of quantum computing. Unfortunately, we find that there are significant problems with the scaling of precision in this context. Particularly we note that introducing error correcting protocols significantly increases the resources required to conduct a simulation. However this is also true of other quantum algorithms. By describing a simulation of the BCS Hamiltonian, we show that the three stages of simulation can be combined to extract useful data in an efficient fashion from our quantum computer.

Chapter 4

Two qubit operations with the qubus

4.1 Introduction

In the next four chapters we will look at simulating the BCS Hamiltonian on the qubus quantum computer. The bulk of our work is split into three chapters to discuss the three main results we have obtained while researching this subject. The fourth chapter pulls together the results of the other three, to give results for a complete simulation of the BCS Hamiltonian. Our three chapters can be broadly divided into a chapter exploring the initialisation procedure, a chapter exploring the time evolution procedure, and a chapter discussing data extraction. However, these subjects have significant overlaps therefore the divide is not quite that elegant.

In this first chapter we will discuss how to produce our two qubit unitaries on a qubus quantum computer. In particular, we show how to get significant reductions in the number of operations required, compared to implementing these two qubit unitaries in a naïve case. We show that it is possible to lower bound the number of operations required, and in our results we demonstrate that we come close to meeting these lower bounds, being only a constant factor out. Once we have performed these operations we can use them as part of the adiabatic initialisation process discussed in section 4.5. Therefore we conclude this chapter by working out the number of operations required to make our desired initial state.

Unfortunately, to perform our simulation we need to implement these unitaries in a controlled fashion. This leads to the second chapter of work on this subject, chapter 5. Here, we look at how to make an N-qubit unitary controlled on an ancilla qubit. This process of performing our operations in a controlled fashion is the main technique we use for evolving our state with respect to time. We do not provide a summary of the total number of operations required in this chapter, leaving that for the next chapter where we introduce our phase-esimation algorithm.

In chapter 6 we look at using the qubus quantum computer to implement the phase estimation algorithm. We begin by taking our results from chapter 5, and using this to work out the number of operations required to perform our controlled unitaries, dependent upon the number of ancilla qubits which are used. The second part of this chapter looks at performing the quantum Fourier transform using our qubus system, and compares the number of operations required to a naïve method.

Finally in chapter 7 we put the results from the previous three chapters together, to work out the total number of operations required to perform a simulation of the BCS Hamiltonian on the qubus quantum computer. We compare this to results that were obtained on an NMR quantum computer and show that our qubus quantum computer performs better in the majority of cases.

4.2 Converting our 2-qubit unitaries

We can not directly implement the terms U_{xx} and U_{yy} on our qubus quantum computer, however we can implement the term

$$U_{zz} = \exp\left(-i\sum_{m<l}^{N}\frac{V_{ml}}{2}\sigma_{zm}\sigma_{zl}\right) \tag{4.1}$$

It is possible to transform U_{zz} to U_{xx} and U_{yy} using local unitaries. These transformation sequences are given by

$$U_{xx} = \exp\left(i\sum_{m'=1}^{N}\frac{\pi}{4}\sigma_{ym'}\right)U_{zz}\exp\left(-i\sum_{m'=1}^{N}\frac{\pi}{4}\sigma_{ym'}\right) \tag{4.2}$$

and

$$U_{yy} = \exp\left(i\sum_{m'=1}^{N}\frac{\pi}{4}\sigma_{xm'}\right)U_{zz}\exp\left(-i\sum_{m'=1}^{N}\frac{\pi}{4}\sigma_{xm'}\right) . \tag{4.3}$$

We therefore consider how to implement our U_{zz} on the qubus quantum computer, to generate the necessary entanglement.

4.3 Placing a minimum bound on the number of operations needed

We begin by placing a lower bound on the number of operations required to generate the sum in equation (4.1). We take into account the number of local unitaries required later in the chapter.

4.3.1 Long range interactions with arbitrary V_{ml}

We start by considering the case where V_{ml} can be set arbitrarily between qubits, this is the most general case, and therefore the most difficult to simulate classically. We would expect this system to be efficient to simulate on a quantum computer since it is a sum of 2 qubit terms. It is therefore one of the Hamiltonians that Lloyd proved possible to simulate efficiently on a quantum computer in 1996 [5]. At the same time it is useful to work out a specific algorithm for this case. Wu et al. [12] demonstrate that the general case can be computed efficiently on a quantum computer, however they do not explicitly calculate the number of operations required in their paper.

We are going to consider placing a lower bound on the number of operations required, given a measurement-free ancilla system. This does not include the NMR computer used in previous work on simulating the BCS Hamiltonian [12, 13]. However it should be applicable to a large range of physical systems, we will concentrate in particular on the qubus system, as this is the architecture we are familiar with. In fact it should be possible to generalise these results to be applicable to a wide range of ancilla systems. One property of this type of system is that our operations occur in pairs. Each pair consists of one entangling and one disentangling operation. These pairs of operations can each generate at most one of the values of V_{ml} in equation (4.1). The simplest lower bound can therefore be given by $2\times$ number of V_{ml}.

The number of V_{ml} in equation (4.1) can be worked out by taking into account that $m < l$ and $m = 1, 2, \ldots, N-1$. When we have $m = 1$, we find that l can take $N-1$ possible values, i.e. $l = 2, 3, \ldots, N$; similarly when $m = 2$, there are $N-2$ possible values for l, i.e. $l = 3, 4, \ldots, N$. This pattern continues until $m = N-1$ where we have only one possible value for l, such that $l = N$. We therefore express the number of V_{ml}, Nb_v as

$$\mathrm{Nb}_v = \sum_{1}^{N-1} k = \frac{1}{2}N(N-1) = \frac{1}{2}(N^2 - N)\,. \tag{4.4}$$

As we have previously established, each of these V_{ml} will require one pair of operations to create, therefore an absolute minimum on the number of operations required is given by $N^2 - N$. It would be highly surprising if we could meet this bound, since in generating the bound we have completely ignored the need to use both quadratures of the bus in order to create entanglement between the qubits.

4.3.2 Short range interactions with arbitrary V_{ml}

If we reduce the number of qubits which interact with each other, we also reduce the number of V_{ml}. One scenario considered by Wu et al. [12] is where U_{zz} describes only nearest neighbour interactions. In this example $l = m + 1$, where $m = 1, 2, \ldots, N-1$ and $l = 2, 3, \ldots, N$

respectively. For every assigned value of m there is only one possible value for l, therefore, there are $N-1$ possible values of V_{ml}. In total we can therefore easily lower bound the number of operations required for our measurement free ancilla scheme to be $2N-2$.

This result can be extended to interactions of range p, where $p=1$ is the nearest-neighbour case. We provide a very simple lower bound for this scenario by setting our value for m such that $m=1,2,\ldots,N-1$. Then due to the fixed range nature of the interaction l takes values from $m+1$ to $p+1$. This gives us a lower bound of $p(N-1)=pN-p$. However, we can make this more precise by remembering $l \leq N$. From this we can see that when $m=N-p+1$ we no longer have p possible qubit values for l. The number of possible values for l now decreases from p to 1 in intervals of 1. We can therefore calculate our lower bound on the number of V_{ml}

$$\text{Nb}_v = (N-p)p + \sum_{1}^{p}(p-k) = Np - \frac{1}{2}p^2 - \frac{1}{2}p \tag{4.5}$$

From this we see that we require at least $Np-(p^2+p)/2$ pairs of operations. Therefore we get a lower bound of $2Np-p^2-p$ operations.

4.3.3 Long range interactions with constant V_{ml}

The above result applies to the case where we can set each value of V_{ml} completely independently, however this is not the only interesting case. To compare to classical simulation, we want to consider the easiest, and most common case, where we set V_{ml} to be constant. When we were considering the number of V_{ml} above, we were actually considering how the qubits interacted with each other, finding the total number of terms in equation (4.1). However, the case of $V_{ml}=-V$ has the same number of terms as for a completely general V_{ml}. Yet as we will demonstrate, this can be implemented using fewer operations. Similarly, in this case, we have only one value for V_{ml}, yet it is obviously impossible to perform using just 2 operations. Therefore while the number of values of V_{ml} will definitely provide us with a lower bound, we want to derive a more realistic bound, one which we have a chance of saturating.

As we established in chapter 2, to generate entanglement on our qubus system we need to entangle one set of qubits with the position quadrature of the bus, and the second set with the momentum quadrature. This will generate entanglement between the two sets of qubits. We will now derive a crude lower bound on the number of operations required by considering which qubits have to connect to each quadrature of the bus. Due to the symmetries in the bus, provided we sandwich the operations, it does not matter which quadrature of the bus is used first. We will consider a simple picture where the term σ_{zm} from equation (4.1) always refers to qubits connecting to the position quadrature, and σ_{zl} refers to qubits connecting to the momentum quadrature of the bus. This means that $N-1$ qubits need to entangle with the position quadrature of the bus, and $N-1$ qubits need to entangle with the momentum quadrature of the bus. Each entangling operation will require a disentangling operation. We can therefore see that the total number of operations required, Nmb is given by

$$\text{Nmb} = 4 \times (N-1) = 4N-4\,. \tag{4.6}$$

4.3.4 Other cases of interest and conclusions

The above technique is limited to cases where we are considering long range interactions as part of our BCS Hamiltonian. A bound on the number of operations required for fixed range interactions with $V_{ml} = -V$, or any other more complex dependency remains an open problem. However since we know that this bound lies between $4N-4$ and $2N$ operations, we can see that we will not get significant savings.

We have now derived two techniques for lower bounding the number of operations required. The first involves counting the number of different V_{ml} in equation (4.1), and the second involves considering which qubits need to entangle to each quadrature of the bus. Since these are both accurate, in each case we will take the highest value for the lower bound. This is a valid assumption to make since if we require at least N^2-N then we require more than $4N-4$, and, as yet, we have not proved that we can saturate either of these lower bounds.

To complete our discussion, we need to consider cases where V_{ml} is not set completely arbitrarily, but at the same time is not as limited as $V_{ml} = -V$. In particular we note that if the values of V_{ml} are highly dependent upon each other, then we require far fewer operations than the standard bound of N^2-N. Therefore our above statement of lower bounds applies to an all or nothing case in terms of generality. However, there are far too many possible dependencies to cover ever single case in this section.

We will therefore content ourselves with noting that when we can characterise our V_{ml} so that they can be generated when each qubit only connects to each quadrature of the bus once, the same lower bound as for $V_{ml} = -V$ is valid. We have defined our displacement operators so that we are able to 'set' the values for V_{ml} by changing the strength of the interaction with the bus. If we connect each qubit with each quadrature of the bus only once, then we can express our V_{ml} as $K_m \times K_l$ where the Ks represent real numbers. In the case of our completely general interactions K_m for a fixed m takes several different values, as we do not limit how often we connect a qubit to the bus. However, if we limit each qubit to connect with each quadrature of the bus only once, then we find K_m remains fixed for a given value of m. This gives us two sets of constants to form our V_{ml}, with each set containing $N-1$ terms. Using these sets of constants it is possible to create some interesting patterns for V_{ml} for example interactions which decay exponentially as the distance between modes increases. An example for the 3 qubit case is $K_{m=1} = 1$, $K_{m=2} = 2$, $K_{m=3} = 4$, $K_{l=2} = 1$, $K_{l=3} = 1/2$, and $K_{l=4} = 1/4$. These values give $V_{12} = 1$, $V_{13} = 0.5$, $V_{14} = 0.25$, $V_{23} = 1$, $V_{24} = 0.5$ and $V_{34} = 1$.

In this section we have established lower bounds on the number of operations required to implement U_{zz} in equation (4.1) for 3 specific situations. The first is the case where each V_{ml} is set independently and we have long range interactions in our Hamiltonian; we then restrict the range of our Hamiltonian but keep V_{ml} general and derive lower bounds for this scenario. Finally we consider a case where V_{ml} is held constant, or where we have a strong dependancy between our values of V_{ml}, such that they can be calculated from a set of constants attached to our qubits. This section leaves many open problems; in particular we do not consider fixed range interactions with a constant V_{ml} nor do we consider other dependencies between the values of V_{ml}. We have, however, covered the most significant scenarios; including the most general case. In the next section we demonstrate that we can get extremely close to meeting all the lower bounds provided, requiring at most 2 additional operations. Since this includes the most general case, we can therefore demonstrate a maximum number of operations required to implement any U_{zz}.

4.4 Implementing U_{zz} on our qubus quantum computer

In section 4.3 we placed a lower bound on the number of operations required to perform equation (4.1) in three cases; a fully generalised case, the case of limited range interactions, and a case where V_{ml} can be expressed as a set of fixed constants K_m and K_l. We now want to consider a physical implementation of equation (4.1) on our qubus quantum computer.

We will start by considering a naïve implementation where we generate each term in our sum using a separate bus, or a bus which completely disconnects from our qubits between operations. In this case each term from U_{zz} is implemented using four operations. As we established in section 4.3, the number of terms in equation (4.1) is given by $(N^2 - N)/2$ in the most general case. This represents every qubit interacting with every other qubit. As we require four operations per term, the total number of operations required will be given by $2(N^2 - N)$. Since we are generating each term in the sum separately, each value of V_{ml} can be set arbitrarily, so this result is for a completely general case. We can therefore conclude that we will never require greater than $2(N^2 - N)$ operations when generating U_{zz} on the qubus in our measurement-free case. However, the number of operations required is significantly larger than our lower bounds, and since we know it is possible to achieve significant savings over a naïve method using the qubus, we will now consider improved methods that allow us to get closer to the lower bounds.

We can also consider a naïve method of implementing U_{zz} if we use measurement to clear the bus, rather than disentangling the qubits. This fits well with the qubus scheme, since we can create our terms from U_{zz} with the entangling operations, and use the measurement simply to disentangle the qubus from the qubits, ignoring the result of the measurement. To get such a scheme to work we would simply need to double one of our values of β in each set of operations. In this case, we would require only three operations (including measurement) for each term in the sum. We would therefore need $3(N^2 - N)/2$ operations.

4.4.1 Long range interactions with arbitrary V_{ml}

We will start by considering the most general case, where every qubit interacts with every other qubit, with an interaction strength governed by a set of V_{ml} which can be set arbitrarily. As we want to set our V_{ml} completely independently of each other, we are limited in how efficiently we can perform our operations.

Each term in U_{zz} consists of an interaction between two qubits, qubit m and qubit l. If we connect qubit m to the momentum quadrature of the bus, we can then connect all the possible options for qubit l to the position quadrature of the bus. This generates the set of interactions between qubits m and $m+1, \ldots, l$, and requires only $(N+1-m)$ operations to perform. The strength of the interaction, V_{ml} can be set arbitrarily between pairs by choosing the β in the displacement operators, $D(-i\beta\sigma_{zl})$. To interact all qubits numbered $> m$ with qubit m we need to use a sequence of displacement operators given by

$$D(i\beta_N\sigma_{z,N})D(i\beta_{N-1}\sigma_{z,N-1})\ldots D(i\beta_{m+2}\sigma_{z,m+2})D(i\beta_{m+1}\sigma_{z,m+1})D(\beta_m\sigma_{z,m})\,. \tag{4.7}$$

This expression leaves the bus highly entangled with the qubit system. The simplest way to continue would be to perform these operations in reverse, and hence disconnect our bus entirely from the system. To interact qubit m with all qubits numbered $> m$, then disconnect the bus, we need a set of displacement operators given by

$$\begin{aligned}&D(-i\beta_N\sigma_{z,N})D(-i\beta_{N-1}\sigma_{z,N-1})\ldots D(-i\beta_{m+2}\sigma_{z,m+2})D(-i\beta_{m+1}\sigma_{z,m+1})D(-\beta_m\sigma_{z,m})\\&D(i\beta_N\sigma_{z,N})D(i\beta_{N-1}\sigma_{z,N-1})\ldots D(i\beta_{m+2}\sigma_{z,m+2})D(i\beta_{m+1}\sigma_{z,m+1})D(\beta_m\sigma_{z,m})\,.\end{aligned} \tag{4.8}$$

It is worth noting that if we had an ancilla that required measurement, and no local corrections, we could roughly halve the number of operations we require since we would need only one additional operation instead of an additional $(N+1-m)$.

In the case of our measurement-free ancilla we clearly need $2(N+1-m)$ operations, for each qubit m we connect to the momentum quadrature of the bus. It is clear from equation (4.1) that m can take $N-1$ possible values ranging from 1 to $N-1$. We can therefore express the total number of operations required, Nmb, as

$$\text{Nmb} = 2\sum_{m=1}^{N-1}(N+1-m) = 2(N+1)(N-1) - N(N-1) = N^2 + N - 2\,. \tag{4.9}$$

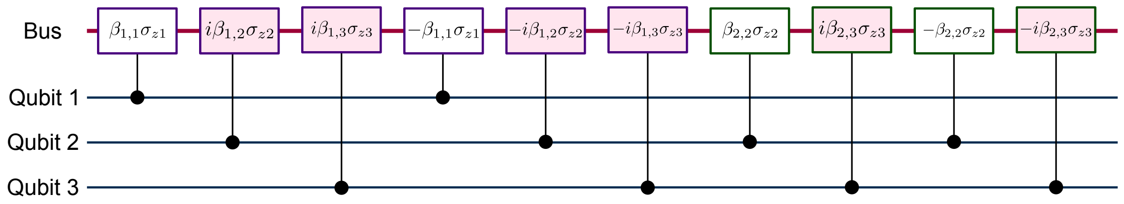

Bus
Qubit 1
Qubit 2
Qubit 3
$\beta_{1,1}\sigma_{z1}$
$i\beta_{1,2}\sigma_{z2}$
$i\beta_{1,3}\sigma_{z3}$
$-\beta_{1,1}\sigma_{z1}$
$-i\beta_{1,2}\sigma_{z2}$
$-i\beta_{1,3}\sigma_{z3}$
$\beta_{2,2}\sigma_{z2}$
$i\beta_{2,3}\sigma_{z3}$
$-\beta_{2,2}\sigma_{z2}$
$-i\beta_{2,3}\sigma_{z3}$

Figure 4.1: A circuit to generate pairs of Pauli Z operations between three qubits. The operations shown are displacement gates on the bus performed dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus, and unshaded boxes represent operations on the momentum quadrature of the bus.

Figure 4.1 shows a qubus circuit to generate the necessary operations between three qubits. We write our displacement operators as $D(-i\beta_{n,l}\sigma_{zl})$, where $\beta_{n,l}$ is the value of β when qubit l connects to the bus for the n^{th} time. We will use this notation every time we are considering a set of operations where each qubit connects to the bus more than once.

If we used measurement then we would need to measure $N-1$ times, once for each cycle where we clear the bus. The total number of operations required would therefore be given by $\frac{1}{2}(N^2+3N-4)$.

This technique gives us a significant saving in the number of operations required over the naïve method, however, our lower bound is N^2-N operations, so for large N we are still $2N$ operations above this bound. This suggests that it might be possible to get further savings on the number of operations required. The most obvious way to do this is to leave some qubits connected to the bus at the end of each step. However, it is important that we do not leave too many qubits on the bus, or our values for V_{ml} will become dependent upon each other.

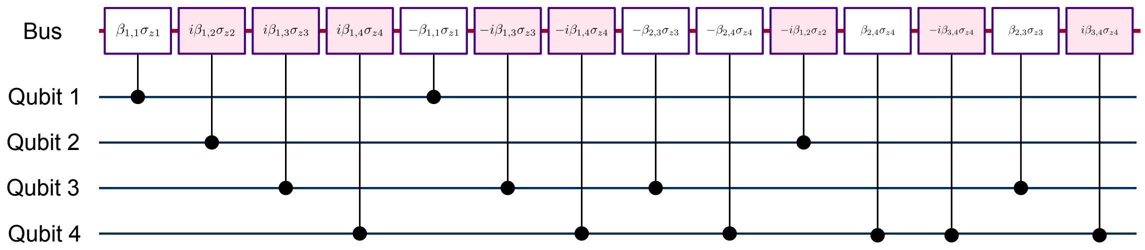

Bus
Qubit 1
Qubit 2
Qubit 3
Qubit 4
$\beta_{1,1}\sigma_{z1}$
$i\beta_{1,2}\sigma_{z2}$
$i\beta_{1,3}\sigma_{z3}$
$i\beta_{1,4}\sigma_{z4}$
$-\beta_{1,1}\sigma_{z1}$
$-i\beta_{1,3}\sigma_{z3}$
$-i\beta_{1,4}\sigma_{z4}$
$-\beta_{2,3}\sigma_{z3}$
$-\beta_{2,4}\sigma_{z4}$
$-i\beta_{1,2}\sigma_{z2}$
$\beta_{2,4}\sigma_{z4}$
$-i\beta_{3,4}\sigma_{z4}$
$\beta_{2,3}\sigma_{z3}$
$i\beta_{3,4}\sigma_{z4}$

Figure 4.2: A circuit showing a way to implement our U_{zz} while getting further reductions. In this case we leave one qubit connected to the bus at the end of each cycle. The operations shown are displacement gates on the bus performed dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus and unshaded boxes represent operations on the momentum quadrature of the bus.

We will therefore consider the simplest possible scenario, leaving one qubit connected to the bus at the end of each cycle. Our scheme of implementation now has a sense of direction. While in our previous example, we could start with any value of m, generating the sets of operations in any order we chose, it now makes sense to start generating our operations from $m = 1$, working up to $m = N - 1$.

The first step of our implementation involves connecting qubit 1 to the momentum quadrature of the bus, then the other $N-1$ qubits to the position quadrature of the bus. We then disentangle qubit 1 from the bus, before disentangling qubits $N - 1$ through to 3. This involves a sequence of displacement operators given by

$$\begin{aligned} &D(-i\beta_N\sigma_{z,(N)})D(-i\beta_{N-1}\sigma_{z,(N-1)})\ldots D(-i\beta_3\sigma_{z,3})D(-\beta_1\sigma_{z1}) \\ &D(i\beta_N\sigma_{z,(N)})D(i\beta_{N-1}\sigma_{z,(N-1)})\ldots D(i\beta_3\sigma_{z,3})D(i\beta_2\sigma_{z,2})D(\beta_1\sigma_{z1}) \end{aligned} \tag{4.10}$$

This is very similar to our previous scheme, the only difference being that we allow qubit 2 to remain on the bus. We can now entangle qubits 3 through to $N - 1$ with the momentum quadrature, remembering that qubit 2 is still entangled with the position quadrature. This sequence, therefore, generates all the necessary interactions between qubit 2 and higher numbered qubits. We continue this process, removing all the qubits but qubit $m + 1$ from the bus, then entangling qubits $m + 2 \ldots N$ to the opposite quadrature of the bus to qubit $m + 1$. The final step of this process involves removing all the remaining qubits from the bus, so that our bus completely disentangles from the system.

Each set of operations will now require one pair of operations less than before. This means each set will now require $2(N - m)$ operations to perform, we will also need an additional two operations for the first set, since in this case there will not be a qubit that has been held on the bus to entangle these qubits with. The total number of operations required will therefore be given by

$$\text{Nmb} = 2\sum_{m=1}^{N-1}(N-m) + 2 = 2N(N-1) - N(N-1) + 2 = N^2 - N + 2\,. \tag{4.11}$$

This is only two operations away from our lower bound.

We now need to prove that this technique allows us to set V_{ml} arbitrarily. To do this we consider how our various values of β depend upon each other, and demonstrate that it is possible to chose values for our β such that we have the ability to set any V_{ml}. If we are given an arbitrary value for $\beta_{1,1}$, we can choose our $\beta_{l,1}$ where $l = 2, 3, \ldots N$ as appropriate, such that we have any V_1, l that we wish; this fixes our value for $\beta_{1,2}$. We now remove all the qubits from the bus except qubit 2. Therefore $\beta_{1,2}$ is the only constant that is reused in the next step. While for the first stage $V_{1,l}$ = $\beta_{1,1} \times \beta_{1,l}$ in the second stage $V_{2,l} = \beta_{1,2} \times \beta_{2,l}$, however since we have complete control of $\beta_{2,l}$ we can generate any $V_{2,l}$. The same logic applies for every value of m up to $m = N-1$. In this case we have complete control over how we set $\beta_{N,N}$ and this constant is never reused. Therefore we have demonstrated that a scheme which leaves one qubit entangled to the bus at the end of each step, allows us to set V_{ml} arbitrarily given physical constraints.

Unlike our previous scheme, it would be hard to map this onto a scheme where our ancilla required measurement. The additional savings we obtain over the N^2+N-2 shown above are entirely due to our ability to leave qubits on the bus. In fact we only save $N-1$ operations through measurement, requiring a total of N^2-1 operations.

We have shown that it is possible to simulate the general case of the BCS Hamiltonian in N^2-N+2 operations. This is roughly a factor of 2 improvement over a naiv̈e implementation. Even in the simplest case with no approximations, the qubus allows us to achieve significant improvements over a basic method. Interestingly, we do not quite saturate the lower bound, requiring one additional pair of operations. There is no obvious way around these additional operations, and they occur because we have one more value of β than the number required to set our V_{ml} arbitrarily.

Our results also show almost a factor of 3 improvement over the naïve scheme if we use measurement to disconnect our qubits from the bus, rather than disconnecting the qubits individually. In this case, the first scheme proposed, gives the most significant improvements requiring only $\frac{1}{2}(N^2+3N-4)$ operations. It seems unlikely that we could obtain any significant reductions from this.

4.4.2 Short range interactions with arbitrary V_{ml}

We now want to consider generating U_{zz} in the case of short range interactions. For simplicity we will assume that our values of V_{ml} still remain arbitrary. We begin by considering nearest-neighbour interactions, before we move on and consider interactions of range p.

In section 4.3 we lower bounded the number of operations needed to perform the nearest neighbour case by considering the number of independent V_{ml} required. This gave us a lower bound of $2N-2$ operations. We can easily see that this bound is impossible to meet by noting that we have N qubits which need to entangle to the bus, and that each qubit needs to interact with the bus twice. Therefore it would be impossible to generate the necessary U_{zz} using fewer than $2N$ operations. Luckily the technique previously published by Louis et al. [25], which we worked through independently, is suitable for this case, and allows us to generate U_{zz} in the nearest-neighbour case. A circuit that allows us to generate nearest-neighbour interactions is shown in figure 4.3.

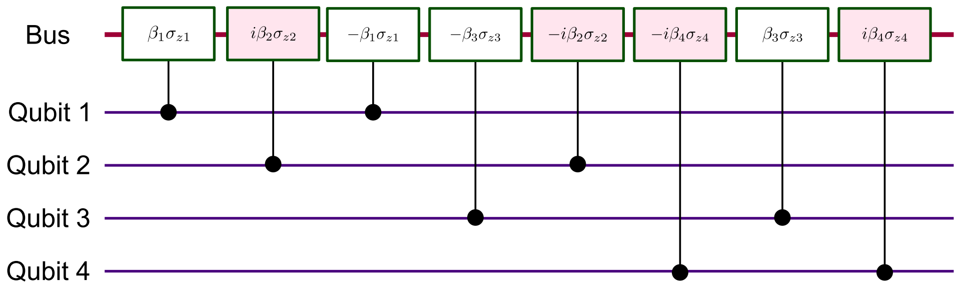


Figure 4.3: A circuit that generates U_{zz} in the case of nearest-neighbour interactions. The operations shown are displacement gates on the bus performed dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus, and unshaded boxes represent operations on the momentum quadrature of the bus.

In this case our U_{zz} is given by

$$U_{zz} = \exp\left(-i\sum_{m=1}^{N-1}\frac{V_{m,m+1}}{2}\sigma_{z,m}\sigma_{z,(m+1)}\right) \tag{4.12}$$

We begin by connecting qubit 1 to the momentum quadrature of the bus, then qubit 2 to the position quadrature. This generates the necessary entanglement between these two qubits. We now disconnect qubit 1, and instead attach qubit 3 to the momentum quadrature, generating entanglement between it and qubit 2. It is possible to continue this process. In each stage we start with qubit m connected to one quadrature of the bus, and qubit $m+1$ to the other. We remove qubit m from the bus, then entangle qubit $m+2$ to the same quadrature. Finally we remove qubit $N-1$ and qubit N from the bus, leaving the bus completely disentangled from our qubit system. The set of displacement operators for this, is given by

$$D(\beta_N\sigma_{z,N})D(-i\beta_{N-1}\sigma_{z,N-1})D(-\beta_N\sigma_{z,N})D(-\beta_{N-2}\sigma_{z,N-1})D(i\beta_{N-1}\sigma_{z,N-2})\ldots \tag{4.13}$$

Interestingly, it would only be possible to save a single operation if we were to introduce measurement. Our measurement free scheme for ancilla computation requires us to leave qubits on the bus until the end point. Therefore the only saving we can gain through measurement is to replace the two operations required to empty the bus with a single measurement. If we consider performing each pair individually using measurement, then we would require three operations per pair; two to entangle the qubits to the bus and one to remove them. This would result in $3N-3$ operations. In the nearest neighbour case, measurement does not provide an advantage.

Our measurement-free scheme for nearest neighbour interactions, and our results for long range from section 4.4.1 are both specific examples of a system of p-range interactions. To obtain general results we consider a similar scheme, using $N-1$ cycles, where we leave one qubit connected to the bus at the end of each cycle. For the first step, $p+1$ qubits need to interact with the bus twice, this represents qubit 1 and the p qubits connected to it. The next set of steps need p qubits to interact with the bus twice, since we assume the first qubit in each step is already connected to the bus. We require $2p$ qubits operation on the bus for every single step except for the very first one and the p final steps. We therefore need to perform $2p$ operations a total of $N-p-1$ times. Finally for the last set of steps there are no longer p qubits left in the chain connected to our active qubit, this means the number of bus operations required reduces from $2(p-1)$ to 0 in intervals of 2. Therefore the total number of operations required is given by

$$\mathrm{Nmb}(p) = 2(p+1) + 2p(N-p-1) + 2\sum_{k=1}^{p}(p-k) = 2pN - p^2 - p + 2\,. \tag{4.14}$$

This is two operations above our lower bound, as previously mentioned this is likely to be due to the need to generate entanglement.

For completeness it is worth considering how measurement affects this result. We can consider two scenarios, the first is using the above technique then using measurement to clear the bus. This is inefficient and would only save us at most p operations. The other option is to have $N-1$ cycles where we connect all the qubits of interest to the bus, then use measurement to clear the bus. This would require a total of $pN + 2N - \frac{1}{2}p^2 - \frac{1}{2}p - 2$ operations, and agrees with our results for the nearest neighbour case, and the case of general long range V_{ml}. From inspection, we can see that for large N, a measurement-free implementation is more efficient than a measurement-based scheme for the nearest-neighbour case. However in the case of long range general interactions the reverse is true. We therefore want to find the cross over point, where using measurements makes our results more efficient. If we assume N is large then this occurs when

$$pN - 2N = 0 \rightarrow p = 2\,. \tag{4.15}$$

We see that in the majority of cases measurement reduces the total number of operations required. However, even without using measurement, we can achieve significant reductions on the number of operations needed to implement limited range interactions compared to a naïve case.

4.4.3 Interactions where V_{ml} can be written as $K_m \times K_l$

If we consider our V_{ml} to be made of constants $K_m \times K_l$, where there is a single value of K_m/K_l for every m/l, then we do not need as many operations as in the cases described above. We will now consider a highly entangled scheme, where a large number of qubits are connected to the bus at once, and remain on the bus until the entire sequence of gates has been generated.

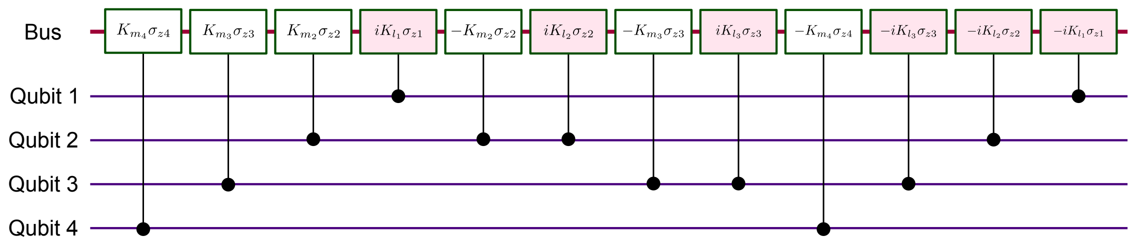


Figure 4.4: This figures shows a circuit which generates all the necessary values of U_{zz}, when our V_{ml} are governed by strict rules that restrict how many constants are required. The operations shown are displacement gates on the bus performed, dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus, and unshaded boxes represent operations on the momentum quadrature of the bus.

We start by connecting qubits 2 through to N to the momentum quadrature of the bus, before entangling qubit 1 to the position quadrature of the bus. This requires N operations, and entangles qubit 1 with all the other qubits. We now consider a routine where in every step we remove a qubit from the momentum quadrature of the bus and connect it to the position quadrature. We start this process with qubit 2 and continue until qubit $N-1$. In this case, flipping the quadrature of the bus a qubit is entangled to generates the necessary interactions between that qubit and the qubits numbered higher than it. This process requires $2(N-2)$ operations (since we do not need to flip the first or final qubit). At the end of this cycle we end up with 1 qubit on the momentum quadrature of the bus and $N-1$ qubits connected to the position quadrature. In our measurement-free scheme, we then use N operations to clear the bus. In our measurement-based scheme we use one operation to clear the bus. The sequence of displacement operators required for the measurement-free scheme is given by

$$\begin{aligned}&D(-iK_{m=1}\sigma_{z1})D(-iK_{m=2}\sigma_{z2})\ldots D(-iK_{m=N-1}\sigma_{z,N-1})D(-K_{l=N}\sigma_{z,N})\\&D(iK_{m=N-1}\sigma_{z,N-1})D(-K_{l=N-1}\sigma_{z,N-1})D(iK_{m=N-2}\sigma_{z,N-2})D(-K_{l=N-2}\sigma_{z,N-2})\\&\ldots D(iK_{m=2}\sigma_{z2})D(-K_{l=2}\sigma_{z,2})D(iK_{m=1}\sigma_{z1})D(K_{l=2}\sigma_{z2})\ldots D(K_{l=N-1}\sigma_{z,N-1})\end{aligned} \tag{4.16}$$

Figure 4.4 shows a circuit to implement a scheme such as the one described above for a 4 qubit case.

For the measurement-free scheme a total of $4N-4$ operations are required. In the measurement based scheme $3N-3$ operations are needed. Therefore, once again using measurement provides a significant saving. The measurement-free scheme meets the lower bound we derived in section 4.3.

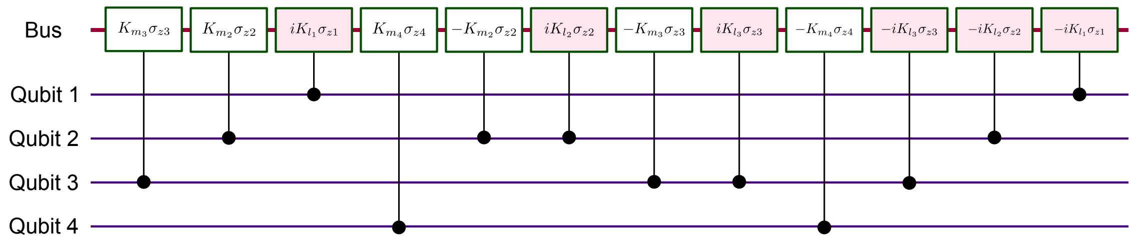


Figure 4.5: A circuit to generate all the necessary values of U_{zz} in the case of limited range interactions, in this case when $p = 2$. Our V_{ml} are governed by strict rules that restrict how many constants are required. The operations shown are displacement gates on the bus performed dependent on the state of the control qubits. Shaded boxes represent operations on the position quadrature of the bus, and unshaded boxes represent operations on the momentum quadrature of the bus.

If we have limited range interactions, it is also possible to perform all the necessary operations using just $4N-4$ operations in the measurement-free case and $3N-2$ in the case of measurement. The extra operation is used to disentangle a single qubit and cancel our unwanted terms. We use a similar procedure to the one above, moving qubits from one quadrature of the bus to another. However extra operations are added between the move. These create some unwanted interaction, but provided all the operations are performed in the correct order these additional interactions cancel out. The example below is for when $N = 4$ and $p = 2$

$$\begin{aligned}&D(-iK_{l=4}\sigma_{z4})D(-iK_{l=3}\sigma_{z3})D(-iK_{l=2}\sigma_{z2})D(-iK_{l=1}\sigma_{z1})\\&D(-K_{l=4}\sigma_{z4})D(iK_{m=3}\sigma_{z3})D(-K_{l=3}\sigma_{z3})D(iK_{m=2}\sigma_{z2})\\&D(-K_{l=2}\sigma_{z2})D(K_{l=4}\sigma_{z4})D(iK_{m=1}\sigma_{z1})D(K_{l=2}\sigma_{z2})D(K_{l=3}\sigma_{z3})\end{aligned} \tag{4.17}$$

A circuit illustrating these displacement operators is shown in figure 4.5.

As before, we can check when this gives advantages over our other techniques for limited range interactions. In both the measurement-free and measurement-based case we find that this technique gives savings when $p \geq 2$. It seems unlikely that further savings would be possible, because for $p > 1$ we have $N-1$ qubits which need to interact with each quadrature of the bus. In the measurement-free scheme this bounds us at $4N-4$ operations. A crude bound for the measurement based scheme can be found by considering the fact that we need to entangle $N-1$ qubits to the momentum quadrature of the bus, $N-2$ qubits need to switch quadrature, 1 qubit needs to connect to the position quadrature, but not to the momentum quadrature and we need 1 measurement. This gives us $3N-3$ as our bound. We therefore find we are close to meeting the bound in the measurement based case, and meet this bound when we do not have limited range interactions.

4.5 Preparing our qubus in the desired initial state

To prepare our initial state we follow a similar procedure to Wu et al. [12] which was outlined in section 3.7.1. We consider our qubus to begin in the state where all the qubits are in state $|0\rangle$. To initialise our system into a state with n qubits in state $|1\rangle$ we simply perform a Pauli X operation on the appropriate qubits. These flips are chosen such that we end up in a ground state of U_0.

We now want to perform a slow evolution on this ground state so that we end up in the ground state of the BCS Hamiltonian. To do this we use the adiabatic procedure in section 3.2.1. This involves performing a direct implementation of the BCS Hamiltonian for small time intervals. Following the work of Wu et al. [12], we will use the first order Trotter approximation to recombine the terms, this gives

$$U_{BCS} \approx \left[\exp(-iH_0 t/S)\exp(-iH_{xx}t/S)\exp(-iH_{yy}t/S)\right]^S . \tag{4.18}$$

In order to work out the total number of operations required for our initialisation procedure, we need to consider the total number of operations required per time interval, and the total number of time intervals.

We start by reducing the number of operations per time interval, by simplifying our adiabatic procedure from

$$H_{ad}(s) = (1 - s/S)(H_0) + (s/S)(H_0 + H_{xx} + H_{yy}) \tag{4.19}$$

to

$$H_{ad}(s) = H_0 + (s/S)(H_{xx} + H_{yy}) . \tag{4.20}$$

This is identical but no longer requires us to implement U_0 twice for each time step. An advantage of our qubus scheme is that U_0 is trivial to implement since we just perform a single local unitary on each qubit. Therefore U_0 requires N operations to implement.

In section 4.4 we established the number of operations, Nmb, required to implement U_{zz} in three different scenarios. Also in section 4.2 showed that it was possible to convert U_{zz} to U_{xx} or U_{yy} using just two local unitaries on each qubit. We can therefore see that our two qubit terms can be implemented using $2(\text{Nmb} + 2N)$ operations. Therefore the total number of operations required for each time step of the initialisation procedure, $I(N)$, is

$$I(N) = 2\text{Nmb} + 5N . \tag{4.21}$$

The total number of time steps for the initialisation procedure is S, where the adiabatic condition requires that $S \gg \pi/(t_{ad}\Delta)$. In the case of the BCS Hamiltonian the short time approximation is valid when $\tau \ll 1/d$ where d is the level spacing, and τ the length of a single time interval. This gives $S \gg \pi d/\Delta$. Brown et al [13] include another factor which represents the precision of the desired energy gap, and is given by δ. This results in the term $S = \pi d/\delta\Delta$. Taking $d/\Delta = 0.1$ from Wu et al [12] we get $S = 0.1\pi/\delta$. If we assume $\delta = 1/100$ as given by Brown et al. [13], the total number of time steps required is given by $S = 10\pi$. The total number of operations needed for the initialisation procedure is therefore given by

$$I_{\text{tot}} = 20\pi\left(\text{Nmb} + \frac{5}{2}N\right). \tag{4.22}$$

We can now substitute in values for Nmb that we found in section 4.4. This will allow us to work out the total number of operations required for the initialisation procedure dependent upon which technique we use to generate our 2 qubit interactions.

In the case of our two qubit interactions being long range and with general V_{ml} we find that

$$I_{\text{tot}} = 20\pi(N^2 + \frac{3}{2}N + 2). \tag{4.23}$$

If we limit the range of these interactions to p then we can get a more general term

$$I_{\text{tot}} = 20\pi\left(2pN + \frac{5}{2}N - p^2 - p + 2\right). \tag{4.24}$$

Finally in the case where our V_{ml} can be expressed as a set of constants $K_m \times K_l$ where there is only one value of K for each value of m and l, we find that our initialisation procedure would require a number of operations given by

$$I_{\text{tot}} = 20\pi\left(\frac{13}{2}N - 4\right). \tag{4.25}$$

4.6 Conclusions and comments on errors

We have looked at performing U_{zz} from equation 4.1 with long, and limited range interactions with completely general V_{ml}, and where $V_{ml} = K_m \times K_l$. In these cases we have considered using measurement to clear the bus, and a standard measurement-free scheme. We find that in all cases except the nearest-neighbour case measurement provides us with savings in the total number of operations required.

Our results are summarised in table 4.1. Some interesting results worth noting are that, in the measurement-based case we require $3N - 3$ for long range interaction, but $3N - 2$ for short range interactions. The short range interactions require an extra operation since we need to create, then destroy, unwanted interactions. The extra operation ensures that all unwanted terms are removed. This is not important in the long range case where these unwanted interactions are

	Lower bound	Total number of operations required:	
	w/o measurement	w/o measurement	with measurement
General V_{ml}	N^2+N	N^2+N-2	$\frac{1}{2}(N^2+3N-4)$
Limited range interaction	$2pN-p^2-p$	$2pN-p^2-p+2$	$pN+2N-\frac{1}{2}p^2-\frac{1}{2}p-2$
Nearest neighbour	$2N-2$	$2N$	$3N-4$
Long $V_{ml}=K_m\times K_l$	$4N-4$	$4N-4$	$3N-3$
Short $V_{ml}=K_m\times K_l$		$4N-4$	$3N-2$

Table 4.1: This table shows the number of operations required to implement our two qubit unitaries in the measurement-based, and the measurement-free cases. It also shows a lower bound for the measurement-free case.

not created. We also find that in the measurement-free case we do not meet the lower bounds on the number of operations required in the majority of cases. This is not surprising since when we consider the case of nearest-neighbour interactions our lower bound is $2N-2$, which is not enough interactions to connect then remove every qubit from the bus. The additional 2 operations bring us to a total of $2N$ operations, which is enough to interact each qubit with the bus, then remove it exactly once. In section 4.5 we used these results to find the total number of operations required for the initialisation procedure of our algorithm. We demonstrated that this can be performed using only 10π repetitions of our result.

It would be possible to expand this work by considering savings for other specific sets of V_{ml}. However as we have demonstrated the general case only requires N^2+N-2 operations, and no case can be done with less than $4N-4$ operations; therefore there is only a limited capacity for savings. An important development for this work would be to look at how to make these results fault tolerant. In particular we note that all our measurement-free techniques above keep qubits entangled to the bus throughout the entire sequence of operations. This means that a single error would propagate throughout the system. While it would be possible to disconnect the bus from the qubits completely with only 2 additional operations in the case of completely general V_{ml}, in the case where $V_{ml}=K_m\times K_l$ this could require significantly more operations. The redundancy in operations appears to be what allows fault tolerance.

Bill Munro provided an error model that took into account de-phasing due to an error on the bus, and de-phasing due to the time required to generate operations. The de-phasing on the bus is due to photon loss from the bus during operations, while the de-phasing due to the time required, is standard decoherence. The total probability of de-phasing is therefore given by

$$P_d = \frac{1}{2}(1-\exp[-N_b\gamma\tau - 4C\eta\beta^2]) \tag{4.26}$$

where N_b is the number of bus operations, γ is the de-phasing for one qubit, C is the number of gates constructed with one use of the bus, η is the loss parameter for the bus, and $\beta=\chi t$

as defined previously. We now compare three cases. The first is our scheme for when $V_{ml} = K_m \times K_l$. The second is our scheme for generating general V_{ml} while completely disentangling the bus between each set of operations. The third is the naïve case.

In the first scheme the number of gates generated before disentangling the qubus is given by $C = (N^2 - N)/2$. This provides all the necessary gates for our U_{zz}. To generate these gates requires a number of bus operations given by $N_b = 4N - 4$. We compare this to the scheme we outlined in section 4.4.1 where we disconnect our bus between every cycle of qubits. The total number of bus operations needed to create our U_{zz} in this case is given by $N_b = N^2 + N - 2$. The number of gates constructed with one use of the bus varies between $N - 1$ and 2. We therefore take $C = \frac{1}{2}(N + 1)$. We find that our first scenario gives less de-phasing provided $\eta\beta \lesssim \gamma\tau/2$. We can also compare our scheme where $V_{ml} = K_m \times K_l$ to a naïve one where we generate each gate individually, in this case $C = 1$ and $N_b = 2(N^2 - N)$. Our improved technique gives less de-phasing provided $\eta\beta \lesssim \gamma\tau$.

If we now take, $\gamma\tau = 5 \times 10^{-4}$, $\eta = 10^{-4}$ and have an error threshold of $P_d = 10^{-2}$ then we can work out how many gates each technique can create. We find that the naïve technique, and the technique where we disconnect the bus between steps, can create gates between 5 qubits, which is the equivalent of 10 gates. In the case of $V_{ml} = K_m \times K_l$ we can create gates between 8 qubits, which is equivalent of 28 gates. The technique where we disconnect our bus between steps also gives a minimal improvement over the naïve method, with the difference being between 5.9 and 5.0 qubits.

A more comprehensive analysis of errors will allow us to work out which technique is beneficial in a practical setting, and would allow us to adapt these techniques for experimental implementation. However for the correct experimental parameters we can see that our reduced scheme gives less de-phasing than our naïve method, since the reduction in decoherence due to the shorter time required to conduct our simulation, is more significant than error accumulation on the bus.

Chapter 5

Making our N-qubit unitaries controlled

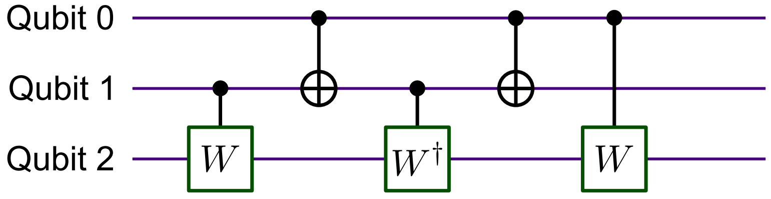


Figure 5.1: A standard technique for converting a two-qubit controlled unitary to a three-qubit controlled unitary. In this case $W^2 = U$ where U is the unitary we want to make doubly controlled.

To use our operations as part of the phase estimation algorithm we need to make them controlled on an ancilla qubit. This is a different ancilla system to the mediating ancilla we use for our bus, and we will refer to the qubit(s) in this system as control qubit(s). A standard gate for making a local unitary, U, controlled on one qubit, doubly controlled is shown in figure 5.1. This circuit takes a unitary $U = W^2$ which acts on qubit 2 dependent upon qubit 1 and makes it unitary acting on qubit 2 dependent upon qubit 1 and qubit 0. We first note that the standard decomposition given to make a controlled unitary doubly controlled [24] shown in figure 5.1, does not work in the cases we are interested in. For simplicity we will consider the unitary we are interested in making controlled to be

$$U_{zz} = \exp\left(i \sum_{m<l}^{N} \frac{V_{ml}}{2} \sigma_{zm}\sigma_{zl}\right) . \tag{5.1}$$

The standard decomposition works in any case where our 2-qubit unitary is in the form of a controlled 1-qubit unitary. Although our U_{zz} is equivalent under local unitaries to a matrix of this form, it is not itself in this form. While it would be possible to transform our sequence to be a set of unitaries on qubit l controlled on qubit m, once we had made this sequence controlled on qubit 0 it would be impossible to change the operations on qubit m and l back to their original form when qubit 0 was in $|1\rangle$ and the identity when qubit 0 was in $|0\rangle$.

We would therefore need to implement this sequence directly, however it is easy to see that this would not work. The sequence involves performing our 2-qubit unitary between qubits 0 and 2, however does not involve performing the Hermitian conjugate of the unitary between qubits 0 and 2. This leaves some unwanted operations on qubit 0, and therefore is not the correct control procedure to make our 2-qubit unitary controlled.

In fact it is impossible to turn a generic N-qubit unitary given by an oracle into a controlled unitary [101]. It is worth noting that this result is unsurprising since the sequences for making even a local unitary controlled, relies on being able to perform W and $W^\dagger$ where normally $U = W^2$. While it is obviously possible to split our U_{zz} sequence into CNOT gates and local unitaries, this is likely to lose a large proportion of the savings which we have obtained in previous sections. It also means we have to reconsider what operations we perform on the qubus, and therefore move to a less 'natural' implementation of our local unitaries. We therefore want to consider a more general method of making our operations controlled, something that can act on a large proportion of the terms in U_{zz}, without control gates between every term. While Barenco et al. [24] provide numerous techniques for making unitaries controlled, these mainly focus on single qubit unitaries. We will therefore outline some simple techniques to implement the N-qubit controlled unitaries of interest. While we provide no arguments for the optimality of these techniques, our most efficient technique requires just over double the number of operations needed to perform our unitary without control. These techniques can therefore be seen to be suitably efficient.

In section 5.1 we will discuss two methods to turn a unitary of the form in equation (5.1) into a controlled unitary, CU_{zz}, of the same form. We will then show how to implement these on the qubus. The first method requires considerably fewer operations to perform on the qubus, and in particular requires fewer CNOT gates. However it relies on our ability to split our U_{zz} into several terms. The second method we present requires only CNOT gates, and the ability to perform W and $W^\dagger$ where $W^4 = U_{zz}$, at the cost of a significant increase in the number of gates required. If we make the unrealistic assumption that all the terms in U_{zz} can be implemented independently as an oracle, then the second method requires a factor of N more gates than the first. In fact for the qubus scheme we find the second method requires only a factor of 2 additional gates. Our second method therefore presents no significant advantages in this case, but we include it for completeness.

In section 5.2 we will look at how to implement terms of the form

$$U_{xx} = \exp\left(i\sum_{m<l}^{N}\frac{V_{ml}}{2}\sigma_{xm}\sigma_{xl}\right) \tag{5.2}$$

and

$$U_{yy} = \exp\left(i\sum_{m<l}^{N}\frac{V_{ml}}{2}\sigma_{ym}\sigma_{yl}\right) . \tag{5.3}$$

In particular we prove that the method we presented in section 5.1 is equally valid for these cases. We therefore show that making our operations controlled on an ancilla qubit does not present a significant disadvantage in our algorithm, and that it should be feasible to use the phase estimation algorithm.

For completeness in section 5.3 we consider how to make our short range interactions controlled on an ancilla qubit, and work out the number of operations required to do this on our qubus. In section 5.4 we present a technique to make certain single qubit local unitaries controlled without the requirement to perform $W^\dagger$, and show in which cases we can use this for our larger U_{zz}. Finally in section 5.5 we show how many operations are required to make our single qubit local unitaries given by

$$U_0 = \exp\left(it\sum_{m=1}^{N}\frac{\varepsilon_m}{2}\sigma_{zm}\right) \tag{5.4}$$

controlled.

5.1 Making U_{zz} controlled on an ancilla qubit

For simplicity, when considering our controlled unitary we can consider the four quadrants of our matrix separately. We will label the qubits in our local unitary as being qubits 1 through to N and our control qubit as being qubit 0. Quadrant 1 represents operations which occur when the control qubit is in the state $|0\rangle$. We want this to remain as identity. Quadrants 2 and 3 represent the control qubit undergoing a change in state from $|0\rangle$ to $|1\rangle$ or vice versa, we therefore want every term in these two quadrants to be 0. Finally, quadrant 4 represents our control qubit being in state $|1\rangle$, and is where our local unitary will be implemented, and we want this to be U_{zz}.

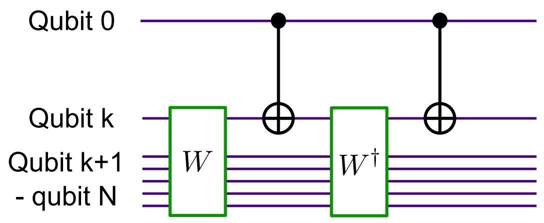


Figure 5.2: A circuit to turn an $N+1-k$ qubit unitary into a controlled unitary. Our unitary is in the form of pairs of Pauli Z gates, where one always acts on qubit k

5.1.1 Method one

We consider splitting our U_{zz} into several terms where m is fixed and $l > m$. We then make each of these terms controlled independently. Since all the terms commute, we can simply perform our CU_{zz} by multiplying them together.

Figure 5.2 shows the architecture independent circuit necessary to turn U_{kl} into a controlled unitary, where k represents a fixed value of m. This sequence is identical to the sequence needed to make a 1-qubit local unitary on qubit k controlled on qubit 0 [24]. However it is worth showing that it will work in our case.

We can easily see that in the first quadrant we perform $W^\dagger W$, which is the identity since W is a unitary matrix. The CNOT gate, W and $W^\dagger$ are all 0 in the second and third quadrant, therefore these are zero in our resultant matrix. We therefore want to consider what happens in the fourth quadrant on its own.

We will consider our operator W to be of the form

$$W = \begin{pmatrix} Q_1 & 0 & 0 & \dots & 0 \\ 0 & Q_2 & 0 & \dots & 0 \\ 0 & 0 & Q_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & Q_{2p} \end{pmatrix} \tag{5.5}$$

where Q are complex numbers. In this case W represents the operation to be performed in the first and fourth quadrant of our matrix, the N qubit unitary. We will consider this matrix to be arranged such that qubit k is the lead qubit. If we consider only the fourth quadrant, then the

CNOT gates are equivalent of a Pauli X on qubit k. Performing our circuit therefore leads to

$$\sigma_{x,k}W^{\dagger}\sigma_{x,k}W = \begin{pmatrix} Q^*_{p+1}Q_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & Q^*_{p+2}Q_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & Q^*_{2p}Q_p & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & Q^*_1 Q_{p+1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & Q^*_p Q_{2p} \end{pmatrix} \tag{5.6}$$

In this case the Q are described by equation (5.1) where k is a fixed value of m. This means for $a \leq p$ we can easily see that $Q_a = Q^*_{p+a}$ therefore $\sigma_{x,k}W^{\dagger}\sigma_{x,k}W = W^2$. For $p < a \leq 2p$, then $Q_a = Q^*_{2p-a}$ so once again $\sigma_{x,k}W^{\dagger}\sigma_{x,k}W = W^2$.

We can see that in each quadrant of our matrix the circuit in figure 5.2 performs the desired result. This provides an easy way to make our U_{zz} operation controlled. In fact it is unsurprising that this technique works when we consider the case of the single qubit unitary. Barenco et al. [24], show that the technique we have outlined above is a valid way to make a single qubit unitary controlled, provided it can be written in the form

$$W = \begin{pmatrix} e^{i\alpha}\cos\theta/2 & \sin\theta/2 \\ -\sin\theta/2 & e^{-i\alpha}\cos\theta/2 \end{pmatrix}. \tag{5.7}$$

While the operation we are considering is based on N qubits, it can easily be written in the form

$$W = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{-i\alpha} \end{pmatrix} \tag{5.8}$$

where $e^{i\alpha}$ acts on qubits $k+1$ through to N when qubit k is $|0\rangle$ and $e^{-i\alpha}$ acts on qubits $k+1$ through to N when qubit k is in $|1\rangle$.

We now consider how many operations we need to implement U_{zz} in equation (5.1) in a controlled fashion on our qubus architecture. In section 2.3.2 we established that it was possible to perform a CNOT gate using 8 operations. Meanwhile in section 4.4.1 we provided a technique that generates all the necessary operations in U_{zz} between qubit m and qubits $m+1$ through to N. In order to be able to implement our control sequence, we will consider the series of operations where the bus is completely disentangled from the qubits during each stage. We found that each state of implementation required $2(N+1-m)$ operations. We now want to perform an equivalent sequence of operations twice, once for W and once for $W^{\dagger}$. While we will need to adjust the parameters we use so that our new parameters $P_{ml} = \sqrt{V_{ml}}$. This should be trivial as our circuit allows us to set this set of constants arbitrarily. In each cycle we will also need to perform 2 CNOT gates, requiring a total of 16 operations. We can therefore express the total number of operations, CNmb, required to perform our U_{zz} in a controlled fashion as,

$$C\text{Nmb} = 4\sum_{m=1}^{N-1}(N+5-m) = 2(N^2+9N-10) \tag{5.9}$$

Equation 5.9 shows we require just over double the number of operations needed to perform our U_{zz} with no control. This suggests that the number of operations needed to make our unitaries controlled will not cause any significant problems with the efficiency of our algorithm.

5.1.2 Method two

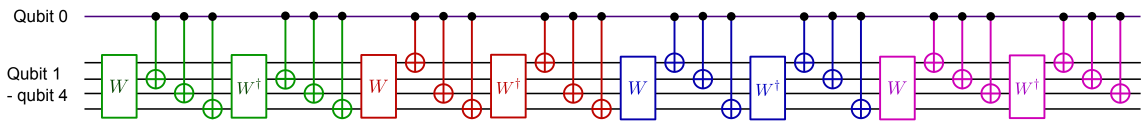


Figure 5.3: A circuit to make a unitary of the form U_{zz} controlled on an ancilla qubit. This circuit is for a U_{zz} acting on 4 qubits but it can be scaled up efficiently. We take $U_{zz} = W^4$.

In section 4.4.3 we showed a technique for generating U_{zz} in some specific cases while getting a saving of O(N). This suggests that it might be beneficial to perform our entire unitary in one go, rather than splitting it into sequences which we perform with a specific lead qubit. We therefore consider an alternative sequence, such as the one shown in figure 5.3. It is worth noting that in the case of this sequence $U_{zz} = W^4$. As before, it is trivial to see that this sequence works in the first, second and third quadrant of our matrix. We therefore only need to consider what is happening in the fourth quadrant.

In the fourth quadrant our CNOT gates behave as Pauli X gates on the desired qubit. This allows us to express what is happening in a simpler fashion. Our operations can be split into sets of a group of Pauli X gates, W and $W^\dagger$. Each of these sets of operations generate the terms in U_{zz} which contain the qubit not being acted upon by the Pauli X gates. For instance the first set illustrated in green in figure 5.3, and given by $\sigma_{x4}\sigma_{x3}\sigma_{x2}W^\dagger\sigma_{x4}\sigma_{x3}\sigma_{x2}W$ generates the interactions between qubit 1 and all other qubits. Any terms that do not involve qubit 1 become zero. Our second set of gates shown in red does the same with qubit 2. Our third set shown in blue gives the required gates for qubit 3. Our fourth set in purple performs terms on qubit 4. As a result we end up with $W^4 = U_{zz}$. Once again it should be trivial for us to adjust the V_{ml} in W such that $W^4 = U_{zz}$ and we obtain the desired interaction strength between our qubits.

To prove that our gate sequence does indeed give the correct results, we will look at what happens when we take W of the form shown in equation (5.5). We will consider qubit k to be in the position of qubit 1, where qubit k is the qubit not being acted upon by a Pauli X. This is just swap operations away from our general case but allows us to illustrate that our sequence works relatively easily. After the application of a single set of gates from figure 5.3 we end up with a system in the form

$$\sigma_{x,2-N}W^{\dagger}\sigma_{x,2-N}W = \begin{pmatrix} Q_p^*Q_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & Q_{p-1}^*Q_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & Q_1^*Q_p & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & Q_{2p}^*Q_{p+1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & Q_{p+1}^*Q_{2p} \end{pmatrix}. \tag{5.10}$$

For simplicity we have chosen to use the notation $\sigma_{x,2-N}$ to represent a Pauli X operation on qubits 2 through to N. Up to $a = p$ we end up with pairs of operations of the form $Q_a^*Q_{p+1-a}$ and for $2p \geq a > p$ the pairs take the form $Q_a^*Q_{3p+1-a}$. Either way, we can see that the first term in our pair was originally in a position where all the qubits except k has been flipped. Any terms in U_{zz} that involve our lead qubit, (the one not being flipped), give $W^2 = F$, while all other terms give $W^2 = \mathbb{1}$. Each term in our U_{zz} will go through this process twice, once for each qubit within the entanglement. Once we have left each qubit un-flipped once our sequence therefore performs the desired operations and we generate U_{zz}. While figure 5.3 features four qubits, this sequence is valid for any number of qubits with $W^4 = U_{zz}$. Even though we are performing W and $W^{\dagger}$, N times each, the resultant operation is $W^4 = U_{zz}$. This is due to the fact that in a large proportion of cases, performing our W and $W^{\dagger}$ just gives the identity. It is worth emphasising that we do not have the scaling $W^N = U_{zz}$, which would require our P_{ml} to become exponentially small.

We now want to consider what the minimum number of operations needed to implement a circuit such as the one shown in figure 5.3 on a qubus architecture. In section 4.4.3 we established that it was possible to perform our U_{zz} using just $4N - 4$ operations. While our W and $W^{\dagger}$ require different values of V_{ml}, it should be possible in perform them using the same total number of operations. We need to perform each of these terms N times, therefore this needs a total of $8N^2 - 8N$ operations to perform. However, we also need a large number of CNOT gates. Our circuit has $2N$ sequences of CNOT gates, each consisting of $N-1$ CNOT operations. A set of $N - 1$ CNOT operations all controlled on the same qubit can be generated using $2N$

bus operations, and $3N - 2$ local unitaries. The CNOT gates therefore require $10N^2 - 4N$ operations in total. Our circuit will require $18N^2 - 12N$ to perform. This is almost a factor of 9 increase over our previous method.

If we do not limit our values of V_{ml}, then each time we perform W or $W^\dagger$ we would require $N^2 - N + 2$ operations. In this case we would require $2N^3 + 8N^2$ operations to implement our controlled U_{zz}. This is an order N increase over the number required in section 5.1.1.

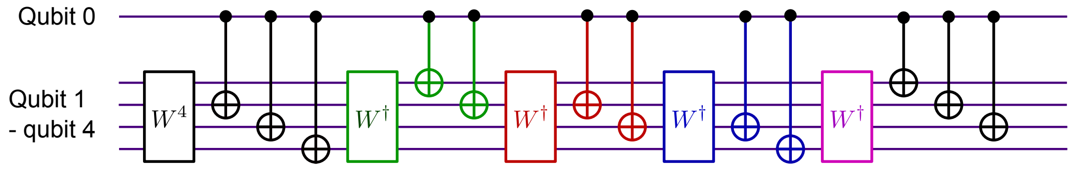


Figure 5.4: A reduced circuit to make a unitary of the form U_{zz} controlled on an ancilla qubit. This circuit is for a U_{zz} acting on 4 qubits but it can be scaled up efficiently.

The large number of repeated operations in figure 5.3 suggests that it should be possible to perform an equivalent circuit more efficiently. We therefore consider the circuit given in figure 5.4 if we use $4N - 4$ operations to perform W. This circuit relies on the fact that $\sigma_x\sigma_x$ is identity, and is directly equivalent to our previous circuit. Once again we can calculate how many operations are required to implement this circuit in the minimum case. For simplicity we will assume that W^N can be implemented in the same number of operations as W, since we are considering only commuting operations and we have not yet fixed our values of V_{ml}. This circuit will therefore require $4N^2 + 23N - 21$ operations, still considerably more than the technique in section 5.1.1.

5.2 Making U_{xx} and U_{yy} controlled on an ancilla qubit

We now want to see if we can use the same transformation sequence to make our U_{xx} and U_{yy} controlled. To turn our U_{zz} into U_{xx} or U_{yy} we need to apply local unitaries on either side of the N-qubit gate. We therefore want to argue that our previous control sequence is still valid.

We can still consider taking a W of the form of a sum of pairs of Pauli Z operations. We then use transforms to modify this to a form of pairs of Pauli X or Pauli Y operations. In section 5.1 we established two control sequences that would work in these cases. It is easy to see that by simply applying a transform before our gate sequence, and the Hermitian conjugate of our transform at the end of the gate sequence, such as in figure 5.5 then we will obtain our U_{xx} and U_{yy}. In this case we consider using $T = \exp(i\pi\sigma_x/4)$ to obtain U_{yy} and $T = \exp(i\pi\sigma_y/4)$ to obtain U_{xx}, where the Pauli operators act on every single qubit in our unitary excluding the control qubit.

We can therefore see that we can easily perform U_{xx} and U_{yy} with just $2N$ gate above the

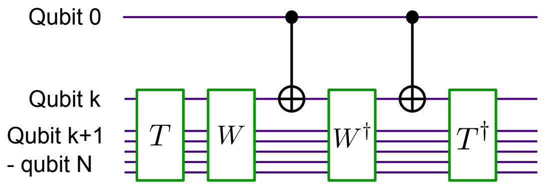


Figure 5.5: A circuit to turn an $N+1-k$ qubit unitary into a controlled unitary where our unitary is in the form of pairs of Pauli X or Pauli Y gates, and where one operation in the pair always acts on qubit k. We set $T = \exp(-i\pi\sigma_y/4)$ to generate a sequence of Pauli X operations and $T = \exp(-i\pi\sigma_x/4)$ to generate a sequence of Pauli Y operations.

number required to implement U_{zz}. This means we can implement each U_{xx} and U_{yy} using $2(N^2+10N-10)$ operations. If we consider a method such as the one in 5.1.2 then we would require a minimum of $4N^2+23N-21$ operations to implement our U_{xx} and U_{yy}. If we were considering an alternative architecture where we could only generate operations in the form of pairs of Pauli X or Pauli Y operations, then we would need to use transforms to turn these operations into Pauli Z operations, perform our control sequence then transform back.

We now want to consider a naïve qubus method where we performed each C-Phase gate individually, and made it controlled independently. This technique requires 24 operations to make each controlled gate. These operations can be split into 8 to perform the gate twice and 16 to perform two CNOTs. We find that, in total our U_{zz} would need $12N^2-12N$ operations, therefore our U_{xx} and U_{yy} would need $12N^2-10N$ operations.

We have shown a technique that allows us to make our U_{xx} and U_{yy} controlled. This requires a little over double the number of operations required to implement U_{xx} and U_{yy} in the general case without any control. Unfortunately our technique can not be adapted so that we get the O(N) savings possible in our specific case in section 4.4.3. However we can still use these savings in the initialisation stage and we get significant savings over a naïve qubus method. We therefore see that the need for control in the phase estimation algorithm is not detrimental to us determining results.

5.3 Performing limited range interactions

In section 4.4.2 we showed that it was possible to implement a Hamiltonian with only short range interactions in fewer operations. In particular we considered interactions of range p. We now want to apply these results to making our operations controlled. We consider using the same control procedure as in section 5.1.1. The only difference is that, since our cut off is earlier, fewer operations will be required in total.

Since we are disconnecting the bus at the end of each lead qubit we need to perform $N-p$

cycles of $4(p+5)$ operations. Each cycle consists of $2(p+1)$ operations to perform W, and $2(p+1)$ operations to perform $W^\dagger$, and 8 operations per CNOT gate. We also want to consider the shorter cycles where we have fewer than p qubits left to connect to the bus. These will require a total of $2p^2+18p-20$ operations including CNOT gates. Therefore to make a N qubit controlled unitary of the form U_{zz} would require $4Np-2p^2-2p+20N-20$ operations. This is a significant increase on the number required to implement our unitary without control. In the nearest neighbour case this is $24N-24$ operations, almost 12 times what we required just to generate U_{zz}.

We find that it is proportionally more costly to make our limited range interactions controlled because our technique in section 5.1.1 relies on splitting our interactions to those performed on a single qubit. While we could use the technique shown in section 5.1.2 this would not present any advantage as we need to perform our U_{zz} sequence $N+1$ times as well as a large sequence of CNOT gates. This means that for our nearest neighbour case we would require $2N^2+25N-17$ operations. While we find that limited range interactions are proportionally more costly to make controlled, it is still cheaper in terms of the total number of operations required to make these controlled, than a U_{zz} with long range interactions. The savings we found in section 4.4.2 are therefore still relevant once we take into account our control procedure.

5.4 Controlled operations without the Hermitian transpose

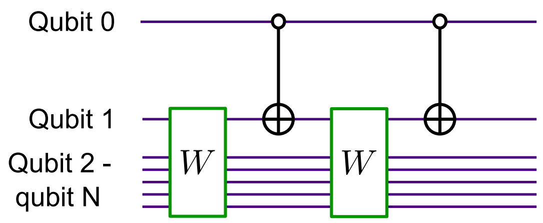


Figure 5.6: This circuit show an alternative control procedure to make a unitary in the form of equation 5.13 controlled on an ancilla qubit. The CNOT gates illustrated flips the target qubit if the control is in $|0\rangle$.

If we have a unitary of the form

$$W = \begin{pmatrix} e^{\alpha} & 0 \\ 0 & e^{-\alpha} \end{pmatrix} \tag{5.11}$$

or

$$W = \begin{pmatrix} Q & 0 \\ 0 & Q^\dagger \end{pmatrix} \tag{5.12}$$

then we can use the simple sequence shown in figure 5.6 to make W^2 controlled on an ancilla qubit. This technique will work independently of how many qubits the operations act upon, provided we can express it this form. In particular if Q and $Q^\dagger$ act on qubits 1 to N, then provided we can arrange our matrix so that Q acts on qubits 2 to N when qubit 1 is in $|0\rangle$ and that $Q^\dagger$ acts to qubits 2 to N when qubit 1 is in $|1\rangle$, this control sequence will work. This is a multi-qubit generalisation of the technique discussed by DiVincenzo in his proof that two qubit gates are universal for quantum computing [102].

We can prove this fairly simply by considering a matrix of the form

$$W = \begin{pmatrix} Q_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & Q_p & 0 & \dots & 0 \\ 0 & \dots & 0 & Q_1^\dagger & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & \dots & Q_p^\dagger \end{pmatrix}. \tag{5.13}$$

We perform the gate sequence in figure 5.6 on this unitary. It is obvious that in our fourth quadrant we are performing WW therefore end up with W^2. Our second and third quadrants give us all zeros. We therefore want to consider what happens in our first quadrant. Here we find

$$\sigma_{x,1} W \sigma_{x,1} W = \begin{pmatrix} Q_1^\dagger Q_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & Q_p^\dagger Q_p & 0 & \dots & 0 \\ 0 & \dots & 0 & Q_1 Q_1^\dagger & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & \dots & Q_p Q_p^\dagger \end{pmatrix}. \tag{5.14}$$

This is the identity. We can therefore see that this control sequence turns our unitary W into a controlled unitary given by W^2.

This technique has the significant advantage that we do not need to be able to perform $W^\dagger$, and it therefore comes close to a technique for making a unitary controlled if it is given by an oracle. It requires the same number of operations as our technique mentioned in section 5.1.1, however we can not adapt it to work in the same way as our second method outlined in section 5.1.2. This means that it is still very limited in the operations it can perform, and it does not function as a general oracle technique, even for operations which are pairs of Pauli operators.

5.5 Making our local unitaries controlled

We also need to make our one-qubit local unitaries in U_0 controlled. This is easy to do using the technique given by Barenco et al [24] and illustrated for a multi-qubit case in figure 5.2. In this case for each unitary we need to perform two CNOT gates, W and $W^\dagger$ where $W^2 = U_0$. We now want to consider the total number of operations required by the qubus to implement our sequence of controlled one-qubit on the qubus. It is possible to implement W and $W^\dagger$ using N operations each.

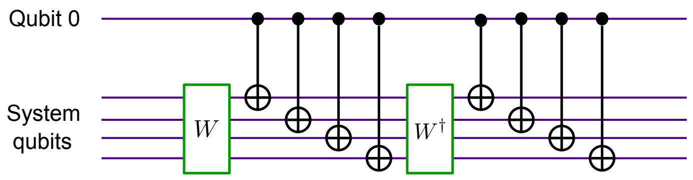


Figure 5.7: This circuit makes a series of single qubit local unitaries controlled. While W and $W^\dagger$ are shown to be acting on all the system qubits we assume their actions are local on each qubit.

The main cost of our implementation is going to be the cost of performing $2N$ CNOT gates. A naïve method of implementation would require 8 operations per CNOT gate, 4 which involve an interaction with the bus, and 4 of which are local unitaries. This means we would need $16N$ operations to implement our CNOT gates. However as all our operations act on individual qubits we can change the sequence of operations so that we perform our CNOT gates in two sets, each one consisting of N gates all acting on different qubits. This results in a circuit such as the one illustrated in figure 5.7. We are now considering performing a large run of CNOT gates in a row. If we consider entangling our bus to our control qubit, then our system qubits, before disentangling our control qubit, we can perform the operations with the qubus in $2N+2$ operations per set of CNOT gates. This means that we can reduce the number of operations required for all our CNOT gates to $10N+6$. Our total sequence would therefore need $12N+6$ operations.

As we are considering only single qubit unitaries we can reduce this even further by using classical computing to work out net local unitaries that give us a combination of $W^\dagger$, and the local unitaries needed to make our CNOT gates. This could save a further N operations meaning that we would need only $10N + 6$ operations.

While making our terms in U_0 controlled on an ancilla qubit presents no significant difficulties and uses basic circuits, it is still expensive requiring 12 times the number of operations needed to implement our single-qubit unitaries without control.

5.6 Conclusions

In this chapter we have explored how to make our N qubit unitary operations controlled upon an ancilla. While we present a technique for doing this that allows us to use all the savings we found in section 4.4 we find that this is costly as it relies on performing our local unitary $N + 1$ times. We therefore consider an alternative technique where we split our operations into those performed on a particular lead qubit as in section 4.4.1. This technique requires roughly a factor of 2 increase in the operations required compared to performing them in an uncontrolled fashion, needing $2(N^2 + 10N - 10)$. We find that if we consider limited range interactions, as the range of the interaction gets proportionally smaller, more gates are required to make our sequence controlled. In the case of nearest neighbour interactions we need roughly 12 times the number of gates that we require for performing our uncontrolled unitary. Despite this, limited range interactions still save us operations and we find that the total needed is given by $4Np - 2p^2 - 2p + 22N - 20$ where p is the range of the interaction.

In section 5.4 we provided an alternative control sequence that requires the same number of gates as our previous one but does not require us to be able to perform $W^\dagger$. This comes close to a technique to make a gate given by an oracle controlled but only works in a very limited number of cases. Finally in section 5.5 we calculated the number of operations which would be required to implement our U_0 controlled on an ancilla qubit. This was a factor of 12 increase over simply performing U_0 and therefore relatively expensive.

This section demonstrates the feasibility of implementing our BCS Hamiltonian controlled on an ancilla qubit, and therefore demonstrates the feasibility of implementing the phase-estimation algorithm on our qubus architecture. In the next chapter we will consider how to get data out of this, and the total number of operations required to get results from our algorithm.

Chapter 6

The phase estimation algorithm

Now that we have established how to make our operations controlled, we need to consider how many operations it will take for us to perform the phase estimation algorithm. We will split this into two sub-sections. The first, section 6.1 considers how many operations will be required to implement our controlled unitaries. The second, section 6.2 considers the number of operations required to implement the quantum Fourier transform.

6.1 Performing our controlled unitaries

The phase estimation algorithm is outlined in section 3.4.1. We will consider U to be an implementation of the BCS Hamiltonian where we use one time interval from our Trotter approximation. Higher order implementations of U will require us to repeat the single time interval, so for example, U^2 will require two repeats of U, and thus double the number of operations. If we could implement the unitary corresponding to our Hamiltonian exactly, it would be possible to perform U^2 with the same number of operations as U. However here we keep our time per step small in order to prevent large errors within our implementation.

We continue the procedure of applying U multiple times to perform the necessary terms up to U^{k-1}, which will require $k-1$ implementations of our BCS Hamiltonian. We can now see why we choose to use k to represent the number of ancilla qubits, since this loosely corresponds to the total number of time intervals required by Wu et al. [12]. However, our phase estimation algorithm will give us an estimate of Φ without regular sampling. Therefore the number of control qubits, and hence the number of time intervals required makes a difference only to the precision of the output.

We can use the adiabatic condition to bound the length of each time interval, τ. Wu et al. [12] calculate that for the small time approximation to hold $\tau \ll 1/d$, where $\epsilon_l = \epsilon_0 + ld$. When we assume $d/\Delta = 0.1$ this gives $\tau \ll 10/\Delta$. Multiplying by the desired precision

$\delta = \Delta/100$ given by Brown et al. [13] gives $\tau = 10/100 = 0.1$. This is a maximum bound and it would be possible to use a smaller time interval than this to increase the accuracy of our results.

We now want to consider how many operations will be required to implement our BCS Hamiltonian in a controlled fashion, as part of the phase estimation procedure. The number of operations for U can be found by putting our results from section 5 into our second order Trotter approximation which for a single small time interval is given by

$$U_{BCS} \approx \left[U_0(\tau/2)U_{xx}(\tau/2)U_{yy}(\tau)U_{xx}(\tau/2)U_0(\tau/2)\right]. \tag{6.1}$$

In particular we note that in chapter 5 we found that U_{yy} and U_{xx} required $2(N^2 + 10N - 10)$ operations to implement in a controlled fashion, while U_0 required $12N + 6$. Provided τ is small, so that the Trotter approximation is sufficiently accurate, we require the same number of operations to implement $U_0(\tau/2)$ as $U_0(\tau)$. The total number of operations, N_{BCS} required per implementation of U_{BCS} is given by

$$N_{\text{BCS}} = 6N^2 + 84N - 48 \tag{6.2}$$

In the case of limited range interactions we find that,

$$N_{\text{BCS}} = 12Np - 6p^2 - 6p + 90N - 48\,. \tag{6.3}$$

The number of times we implement U_{BCS} depends on the number of control qubits we use. If we consider our k control qubit system then we find that we need to repeat our unitary a total of $2^k - 1$ times. For every additional ancilla we get an exponential increase in precision, therefore an exponential increase in precision costs us an exponential number of gates. The level of precision available can be expressed as a function of 2π such that the smallest phase difference we can detect is given by $2\pi/2^k$. If we have an equivalent precision to Brown et al. [13]. We want the smallest phase difference we can detect to be given by δ. This gives us $2^k = 2\pi/\delta$, taking $\delta = 1/100$, we find $k = 10$. Therefore the total number of operations required for our controlled unitaries is

$$N_{\text{cont}} = (2^{10} - 1)(N_{\text{BCS}}) = 6138N^2 + 85932N - 49104 \tag{6.4}$$

in the general case, and

$$N_{\text{cont}} = 12276Np - 6138p^2 - 6138p + 9207N - 49104 \tag{6.5}$$

in the limited range case. It is also worth considering how to express this in terms of precision.

With k ancilla qubits we $(2^k - 1)$ times the number of operations that we do with a single ancilla. Therefore we find that in the case of a Hamiltonian with general interactions,

$$N_{\text{cont}} = \frac{2\pi}{\delta}(6N^2 + 84N - 48)\,. \tag{6.6}$$

While in the case of limited range interactions

$$N_{\text{cont}} = \frac{2\pi}{\delta}(12Np - 6p^2 - 6p + 90N - 48)\,. \tag{6.7}$$

Now we have worked out the number of operations required to implement the controlled sequence from the phase estimation algorithm, we want to go on and consider the inverse quantum Fourier transform.

6.2 The quantum Fourier transform

An essential part of the phase-estimation algorithm is the inverse quantum Fourier transform (QFT) [23]. We will outline here how to perform the quantum Fourier transform in the completely general case. The QFT is of general interest in quantum computing since it is responsible for the exponential speed up in a large proportion of quantum algorithm including Shor's algorithm [23]. We will also briefly mention how this changes if we measure immediately after performing the QFT.

6.2.1 Implementing the quantum Fourier transform

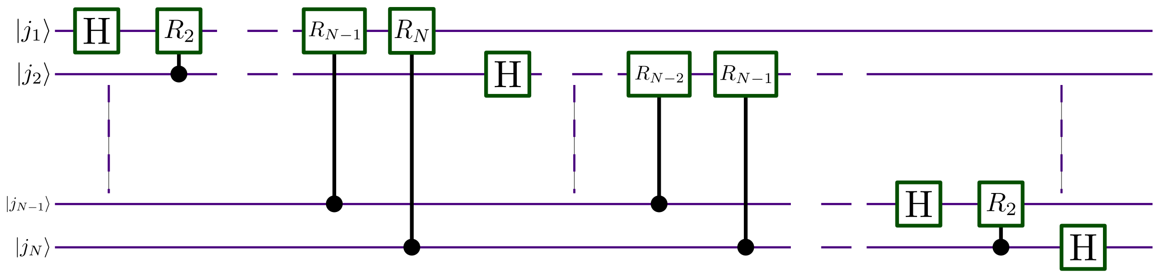


Figure 6.1: A circuit diagram for the quantum fourier transform on N qubits, where R_k are rotations on qubit k given in equation (6.8), and H is the Hadamard operation.

A circuit for the quantum Fourier transform (QFT) is shown in figure 6.1. This figure excludes the necessary swap operations, which reverse the order of the register qubits. We will show later that performing these operations is the dominant term in performing the quantum

Fourier transform on a qubus quantum computer. These operations should therefore be avoided where possible. The gates R_k are rotations on the qubits which take the from

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} . \tag{6.8}$$

The rotations R_k are performed conditionally based upon the state of another qubit. If the rotations are being formed on qubit m then R_k is performed conditionally on qubit $m+k-1$. If we consider only the two-qubit gates, then we can easily bound ourselves to $4N-4$ operations to implement the QFT on an N qubit system. The QFT differs from our implementation of U_{zz} for the BCS Hamiltonian, because we also need to consider local unitaries, in particular Hadamard operations and corrections.

Due to the limitations of the qubus architecture, we are unable to create the desired local unitaries without performing local corrections. For each desired CR_k given by

$$CR_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix} \tag{6.9}$$

we instead perform

$$CR_k' = \begin{pmatrix} e^{\pi i/2^{k+1}} & 0 & 0 & 0 \\ 0 & e^{-\pi i/2^{k+1}} & 0 & 0 \\ 0 & 0 & e^{-\pi i/2^{k+1}} & 0 \\ 0 & 0 & 0 & e^{\pi i/2^{k+1}} \end{pmatrix} . \tag{6.10}$$

To achieve the desired gate we need to perform local corrections on each qubit, which are given by

$$C_k = \begin{pmatrix} e^{-\pi i/2^{k+2}} & 0 \\ 0 & e^{3\pi i/2^{k+2}} \end{pmatrix} . \tag{6.11}$$

These local corrections commute with the operations in equation (6.10) and thus can be performed at the end of a sequence of rotations. However they do not commute with the Hadamard operations, so any set of local corrections need to be performed in the correct order compared to this operation.

A circuit diagrams for implementing the QFT on a qubus quantum computer is shown in figure 6.2, this was worked out jointly by me and Suvabrata De. From this figure it is easy to see that our two qubit gate sequence is almost identical to the one in section 4.4.3, where we flip qubits from one quadrature of the bus to the other. We note that when a qubit is the target

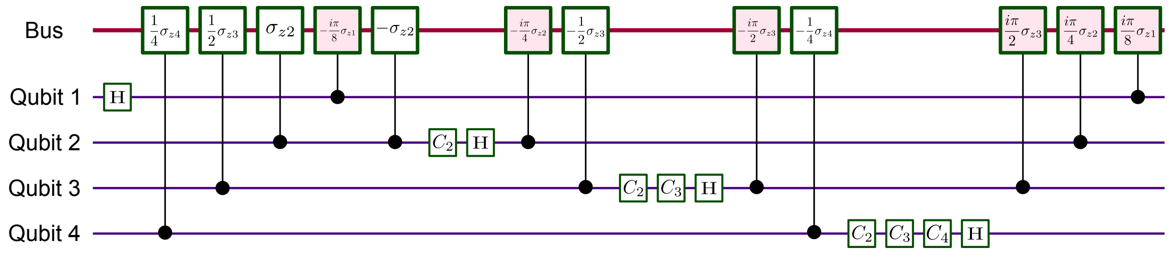

Bus
Qubit 1
Qubit 2
Qubit 3
Qubit 4
$\frac{1}{4}\sigma_{z4}$
$\frac{1}{2}\sigma_{z3}$
σ_{z2}
$-\frac{i\pi}{8}\sigma_{z1}$
$-\sigma_{z2}$
$-\frac{i\pi}{4}\sigma_{z2}$
$-\frac{1}{2}\sigma_{z3}$
$-\frac{i\pi}{2}\sigma_{z3}$
$-\frac{1}{4}\sigma_{z4}$
$\frac{i\pi}{2}\sigma_{z3}$
$\frac{i\pi}{4}\sigma_{z2}$
$\frac{i\pi}{8}\sigma_{z1}$
H
C_2
C_3
C_4

Figure 6.2: A diagram showing how to perform a 4 qubit Fourier Transform on the qubus quantum computer. The boxes show displacements performed on the continuous variable field controlled by the qubits or local operations. Shaded boxes represent an operation acting on the position quadrature of the bus, while unshaded boxes represent operations on the momentum quadrature.

qubit it is connected to the position quadrature of the bus, and when it is a controlled qubit it is connected to the momentum quadrature of the bus. The key difference, is that when we remove a qubit from the momentum quadrature of the bus, we apply local unitaries before connecting it to the position quadrature. These local operations take the form of a set of corrections and a Hadamard operation. As our local corrections are diagonal operations, it is possible to combine them efficiently on a classical computer, then to perform a single correction for each set.

We can now work out the total number of operations required to perform the QFT (excluding the swap operations) in the case of N qubits. The two qubit operations can be performed in $4N-4$ interactions with the bus. We require N Hadamard operations, and two local corrections on each qubit; one performed before the Hadamard, and one after. In actual fact, qubit 1 does not require any corrections to be performed before the Hadamard, and qubit N does not require any corrections after the Hadamard. Therefore a total of $7N-6$ operations are required in the general case.

It is possible to perform an approximate version of the QFT by implementing only the largest rotations [103]. However we can see that in the qubus case this will not give us any significant reductions in the number of operations required, since each qubit (except qubit 1 and N) still acts as a target qubit and a control qubit for at least one interaction. Therefore we would still need $4N-4$ two qubit operations and 2 local corrections on each qubit. An interesting consequence of this, is that it would be impossible to use a nearest-neighbour only reduction like the one in section 4.4.2 since this would not allow us to apply the Hadamards at the necessary point. Therefore even the nearest-neighbour case would require $4N-4$ operations to generate the necessary two qubit gates.

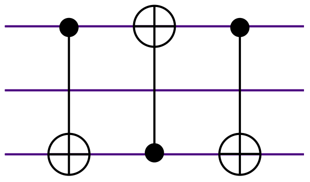


Figure 6.3: A SWAP operation on 2 qubits, made up of 3 CNOT gates

6.2.2 The need for SWAP gates

We now want to consider how to perform the necessary SWAP gates. It is possible to decompose a SWAP gate into a sequence of 3 CNOT gates [33] with a circuit like the one shown in figure 6.3. Each CNOT gate requires 8 operations to perform, 4 entangling operations and 8 local corrections. This means each SWAP gate would require 24 operations in total to perform.

It would be impossible to reduce this further because the CNOT gate requires that we perform $\exp(\pm i\pi\sigma_y/4)$ either side of the bus operations on the qubit where we require an interaction of the form σ_x. When we perform our 3 CNOTs we switch which qubit has an interaction of the form σ_x, therefore we need to disconnect the bus between each set of operations. As we want to reverse the order of the qubits we require $\lfloor N/2 \rfloor$ SWAP gates. Therefore the total number of operations required for the SWAP gates is $24\lfloor N/2 \rfloor$.

A full quantum Fourier transform therefore requires

$$\text{Nmb operations for QFT} = 24\lfloor N/2 \rfloor + 7N - 6\,. \tag{6.12}$$

This shows quite clearly that the SWAP gates are the dominant term within the QFT.

6.2.3 The quantum Fourier transform with measurement

It is possible to reduce the number of operations needed if we measure in the z-basis straight after performing the QFT. This allows us to remove one set of corrections and also the SWAP gates. Therefore if we consider measuring straight after performing the QFT, then the number of operations required is reduced to

$$\text{Nmb operations for QFT if then measure in z-basis} = 6N - 5\,. \tag{6.13}$$

In fact, when we measure immediately after performing the quantum Fourier transform, we do not need to use the complete QFT, and can instead use the semi-classical quantum Fourier transform [104], or even perform the entire operation classically [105]. We have not outlined either of these two methods here since neither require any 2-qubit gates, and so they are beyond the scope of this work. It is worth noting, however, that both of these techniques could provide significant savings in the number of operations required for algorithms such as phase estimation.

We consider the case where we measure straight after performing the QFT in particular, as it is the one will use as part of our phase estimation algorithm. To maintain notation we have used N in the context of the QFT to represent the number of qubits our quantum Fourier transform acted upon. In the case of simulating the BCS Hamiltonian, our QFT acts on our control qubits and therefore the number of operations required is given by $6k - 4$. In section 6.1 we established that for the desired level of precision we wanted to consider a system of 10 control qubits. Therefore we find that our Fourier transform sequence requires 56 operations. If we were considering performing the phase estimation procedure and not measuring, then 184 operations would be required.

6.3 Conclusions

In this section we have established the number of operations required for both sections of our phase estimation procedure. By considering the required precision we can work out the required number of ancilla qubits, provided $\delta = 1/100$. We found that a total of $6138N^2 + 85932N - 49104$ were required to implement the necessary set of controlled unitaries in the general case, and $N_{\text{cont}} = 12276Np - 6138p^2 - 6138p + 92070N - 49104$ in the case of limited range interactions. The quantum Fourier transform required significantly fewer resources, only needing 56 operations. The total number of operations required for the data extraction procedure is therefore given by $6138N^2 + 685932N - 49048$ in the general case and $12276Np - 6138p^2 - 6138p + 92070N - 49048$ in the case of limited range interactions. This is a significant number of operations and shows that our need to perform the phase estimation algorithm is expensive in terms of the number of operations required. For large N we still show a saving of $\mathrm{O}(N^3)$ over an NMR implementation. However we want to look in detail at how significant this saving is by considering the more realistic small cases.

Chapter 7

A complete simulation of the BCS Hamiltonian

7.1 Comparing to previous work

We now want to compare the number of operations required to implement the BCS Hamiltonian on a qubus quantum computer to those required on an NMR quantum computer. One significant advantage of the qubus quantum computer is the theoretical ability to scale up our simulation to a larger number of qubits, since even if we ignore problems with decoherence time, it would be impossible to run large simulations on an NMR quantum computer. We show that as well as allowing us to run larger simulations, the qubus gives a saving in the number of operations required provided $N > 6$.

To make our comparison accurate we will consider the nearest-neighbour case. In chapter 6 we found that the phase estimation algorithm required a total of $12276Np - 6138p^2 - 6138p + 92070N - 49048$ operations to extract the phase from a system with short range interactions. Setting $p = 1$ we see that in the nearest neighbour case this equates to $104346N - 61326$. We also want to consider the initialisation procedure discussed in section 4.5, which prepares our first register in the superposition of eigenstates. In the short range case, this required a total of $20\pi(2pN + 5N/2 - p^2 - p + 2)$, or $100\pi N$ operations in the nearest neighbour case. The total number of operations required is therefore given by

$$T_{\text{qubus}} = 104660N - 61326 \tag{7.1}$$

Wu et al. [12] claim to require significantly more than $3N^4$ operations, however this excludes the initialisation procedure. Since they use the first order Trotter approximation throughout we will use the same number of time steps for the initialisation procedure as the longest run of the computer. When using the precision bounds given by Brown et al. [13] this becomes

$$T_{\text{NMR}} = 600N^4 \tag{7.2}$$

If we compare the two results, then it is easy to see that our qubus system requires fewer operations when $N \geq 6$. However up to this point the NMR system is more efficient. In the case of $N = 10$ our qubus system requires $985,724 \approx 1 \times 10^6$ operations, while the NMR system requires 6×10^6 operations. Therefore we can already see a significant difference. Using a similar number of operations on our qubus system, it would be possible to generate operations for a BCS Hamiltonian of 56 qubits.

We also want to consider the case of full range interactions, in this case

$$T_{\text{qubus}} = 6201N^2 + 85932N - 49104\,. \tag{7.3}$$

Wu et al. [12] do not give the number of operations required in the general case but state that it is O(N^5). We will therefore make the approximation that we require a total of N^5 operations for each run meaning,

$$T_{\text{NMR}} = 66N^5\,. \tag{7.4}$$

In this case we find that the qubus architecture gives us advantages over the NMR system provided $N > 7$. This is an approximate result since we do not know the exact number required by the NMR computer, so this could be out by a few qubits. If on the NMR quantum computer we considered each run requiring $9N^5$ operations (equivalent to the $9N^4$ for the nearest neighbour case), then our qubus system would give advantages for all $N > 4$. We find that in the general case our qubus system requires 1.45×10^6 operations to generate operations between 10 qubits. Using the number of operations required by the NMR system to create nearest-neighbour operations between 10 qubits, we could create our long range interactions between 25 qubits.

We can also consider a more general comparison by not specifying precision, in the general case the number of operations required will be given by

$$T_{\text{qubus}} \approx \frac{0.1\pi}{\delta}(122N^2 + 1683N - 956) \tag{7.5}$$

and in the limited range case it will be

$$T_{\text{qubus}} \approx \frac{0.1\pi}{\delta}(244Np - 122p^2 - 122p + 1805N - 956)\,. \tag{7.6}$$

As before, we can now compare our results to those found by Wu et al. [12]. We incorporate the precision such that

$$T_{\text{NMR}} = \frac{6}{\delta}N^4 \tag{7.7}$$

in the nearest neighbour case to leading order. For our qubus system in the nearest-neighbour case we have

$$T_{\text{qubus}} = \frac{\pi}{\delta}(204.9N - 120)\,. \tag{7.8}$$

We can therefore see that provided $N \geq 5$ our qubus system requires fewer operations than an equivalent NMR simulation regardless of the level of precision required. We note that this result seems to contradict our previous result. This is because we had to round up the number of ancilla/control qubits thus increasing our number of operations. Using a factor for our precision of $2\pi/\delta$ rather than 2^k gives values for 2^k with non-integer k. This means we may require more operations than the number given by our precision factor in certain cases. In the specific example above, our qubus architecture gives a higher precision result than the NMR case, in particular we notice we are considering $\delta = 0.0061$, as opposed to $\delta = 0.01$.

7.2 Conclusions

In this chapter we have discussed simulating the BCS Hamiltonian on the qubus quantum computer, and we have compared our results to a simulation on an NMR system. In particular we have provided speeds ups compared to a naïve method for all three stages of our quantum algorithm.

For the initialisation stage, we have demonstrated how to implement our 2-qubit unitaries, U_{xx} and U_{yy} in a near optimal fashion getting up to O(N) saving over a naïve method of implementation. This allows us significant savings in the number of operations required to prepare our fist register in a superposition of eigenstates of the BCS Hamiltonian.

The time evolution stage has been combined with the phase-estimation algorithm, and is represented by the sequence of controlled unitaries within this algorithm. To implement the time evolution, we considered how to make our unitaries U_{xx} and U_{yy} controlled upon an ancilla qubit. We presented two possible methods of making our N-qubit unitaries controlled, and a third method which could produce $U = W^2$ from W without the requirement to perform $W^\dagger$. Our first method of making our unitaries controlled relied on being able to reduce our operations into sets, where all the two-qubit operations in each set acted on a particular qubit and on an unknown other qubit. Our second control method required more operations, but did not need us to be able to split our unitaries into sets, only be able to perform W and $W^\dagger$ where $U = W^4$. While our control methods used some of the previous savings found for implementing U_{xx} and U_{yy}, we found that a lot of savings were lost in the control procedure. Despite this, we managed to obtain almost a factor of 5 saving over a naïve method.

Finally we looked at extracting data from our phase estimation algorithm using the quantum Fourier transform. We showed a technique for implementing the QFT that only required $24\lfloor N/2 \rfloor + 7N - 6$ an $\mathrm{O}(N)$ improvement over a naïve method. This result is valid for any input to the QFT and does not require measurement straight after implementation. The QFT is widely used in a lot of quantum algorithms, including Shor's algorithm [1]. The results we show in this section, are therefore likely to be useful for a wide class of problems. This demonstrate that the savings we show on the qubus are very general, and not just for limited scenarios.

Putting all our results together allowed us to compare our qubus system to an NMR system and show that we need $\mathrm{O}(N^3)$ fewer operations. We found that in the nearest-neighbour case discussed by Wu et al. [12] the qubus system was more efficient in the number of operations required provided $N \geq 6$. To compare with the general case was more difficult as Wu et al. [12] did not mention the number of operations needed, however we estimated that the qubus will give improvements provided $N \geq 7$. In particular we noted for the number of operations Wu et al. [12] require to implement a 10 qubit nearest-neighbour system the qubus would allow us to simulate a 56 qubit nearest-neighbour BCS Hamiltonian, or a 25 qubit BCS Hamiltonian with long range interactions.

We can therefore see that the qubus system provides significant advantages over the NMR for simulating the BCS Hamiltonian, requiring far fewer operations to implement any system larger than around 6 qubits. The main disadvantage is the need for control qubit ancillas to use as part of our data extraction procedure. The number of these control qubits required depends on the precision and not the size of the system, therefore as N becomes large these additional qubits become insignificant. In particular our qubus gives us significant advantages in generating the long range interactions present in the BCS Hamiltonian.

Chapter 8

Cluster State Computation

8.1 Introduction

Similar to classical computing, while the circuit model is one of the most commonly considered forms of quantum computing, there are alternative models. In quantum computing, the three most famous examples are continuous variable quantum computing (the quantum equivalent of the analogue computer) [106], adiabatic quantum computing [107], and one-way quantum computing [18]. Adiabatic and one-way quantum computation are of particular interest because they have no classical counter part. We consider using adiabatic quantum computing as part of our algorithm for simulating the BCS Hamiltonian in section 3.7.1. However in this chapter we will concentrate on one-way or measurement-based quantum computation.

A one-way quantum computer consists of a particular graph state, where the nodes are qubits, and the edges are entanglement between qubits, taking the form of either C-Phase or CNOT gates. Operations are performed on the computer by measuring qubits, then performing local corrections based on the results of these measurements. While it would be possible to design a graph for the necessary calculation based on standard mappings between quantum gates and a one-way computing 'circuit', often a cluster state is considered instead. A cluster state is a square lattice graph with interactions between nearest-neighbour qubits, (although not diagonal neighbours), which is universal for one-way computation. This means that once the cluster state has been generated, no further entangling operations are required, and that any calculation that can be performed on a universal quantum computer can be implemented using just local operations and measurements.

In section 8.2.1 we consider the initial proposal by Raussendorf et al. [18] to create a cluster using a single interaction which entangles all the qubits at once. Unsurprisingly our square cluster state is not the only universal scheme for one-way quantum computation. A lot of work has been done looking at the quantity of entanglement required to make a universal resource for one-way computation [108]. Most of this work is beyond the scope of this discussion, however we briefly mention some of this work in section 8.2.1, in the context of one-shot and cooling schemes to create a cluster state. While the square cluster state is not a ground state of a physical system, it is hoped that some of these other universal resources will be.

While one-shot or cooling schemes for generating cluster states, are perhaps the simplest, they are far from the most common. Due to the difficulties in implementing these schemes, other work has looked at using optical cluster states, and generating the necessary C-Phase gates in a probabilistic fashion. This work is discussed in section 8.2.2, and looks at two different areas; building deterministic gates from probabilistic ones, and using probabilistic gate chains to grow a cluster.

The one-way quantum computer is of particular interest, because it allows the generation of entanglement to be pushed off-line, after which point only local corrections and measurements are required. In particular, we note that the need for a large number of C-Phase gates is something we can generate tidily on our qubus quantum computer. We consider previous work on this subject in section 8.2.3. This section considers deterministic methods for building a cluster state, then discusses the method proposed by Louis et al. [25] for building clusters using the qubus. In original work, Clare Horsman bounds the number of operations required using Louis' method. We then show an alternative technique of generation, which meets this bound.

While we concentrate mainly on building the cluster state needed for one-way computation, section 8.3 highlights some of the key results in how one-way computation can be used for universal quantum computation. In particular, we summarise the most basic mapping of a two qubit gate to a cluster to demonstrate the size of cluster needed to perform such operations. This work also shows how relatively expensive cluster state computation is. Finally in section 8.4 we contrast this by highlighting some of the advantages of using cluster states.

8.2 Previous results for cluster state generation

8.2.1 Constant shot and ground states schemes

In their initial paper on measurement-based quantum computation, Raussendorf and Briegel [18] briefly describe a one shot scheme for generating a cluster state. They propose preparing a system of qubits in the state, $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, then applying the Ising interaction for a suitable length of time to generate the necessary entanglement. A possible physical implementation for this would be atoms stored in micro-traps or optical lattices.

However, while the one shot Ising scheme is elegant in terms of the small number of operations needed, the Ising interaction is difficult to implement experimentally. As a result

Borhani and Loss look at using the Heisenberg interaction [109]. While this scheme increased the number of operations required, it also made the scheme more physically realistic. In particular, by interacting pairs, they showed how to build a cluster using $2d$ operations, where d was the dimension of the cluster. This meant that a square cluster could be built using only 4 time steps. It is easiest to see how this scheme works in the 1D case, considering a chain of n qubits. In this case the first set of operations entangle qubits 1 and 2, 3 and 4, 5 and 6 etc. The second set of operations then entangle qubits 2 and 3, 4 and 5, 6 and 7 etc. Expanding into 2 dimensions, the first two set of operations can be used to create m horizontal chains, while the second two sets could combine these chains into a grid of width n. In fact, we can use our qubus scheme to do something directly equivalent provided we have access to enough buses. The disadvantage of this scheme is the need to perform several operations in parallel. However provided that this is physical possible, it is likely to be the most efficient practical way to generate the necessary entanglement, particularly since it addresses the issues of how to avoid creating unnecessary entanglement when the interaction is switched on. Borhani and Loss propose using quantum dots as a physical implementation. However their scheme would be valid in any system where it was possible to create the Heisenberg interaction in a suitably controlled fashion.

Difficulties with practical implementation means that only a limited amount of work has looked at schemes for generating clusters which require a constant number of operations. When considering doing multiple operations at once, it is easy to lower bound the total number of operations required. It is therefore, easy to see that if a qubit can only interact with one other qubit in a single operation, the scheme presented by Borhani and Loss [109] is optimal. In other cases we would require at least one operation so we could never improve upon the scheme originally presented by Raussendorf and Briegel [18]. However, very few suggestions exist on how to practically implement these schemes. This has led to work which looks at cluster states as ground states of a physical system. Unfortunately this has been shown to be impossible for a physical system consisting of two-body interactions [110, 111]. In his review paper, Nielsen [110] uses a more liberal definition of cluster states than the one in this work, using it to describe a graph used for one-way computation. However, despite this, work has looked at what alternative entangled states are universal for one-way computation. While two-body Hamiltonians are physically realistic, it is still potentially possible to create a cluster state as a ground state of a system which is not naturally occurring but which can be manufactured artificially [111].

Characterising the amount of entanglement in a system is a field of research in its own right. Similarly, the large amount of work on defining what resources would be suitable for one-way computation is worthy of a review article, and is beyond the scope of this work. Here we will summarise only the most useful results for practical methods for creating a universal resource in an efficient way. Recently, Wei et al. [108] showed that the ground state of AKLT (Affleck-Lieb-

Kennedy-Tasaki) model on 2D honeycomb lattice is a universal resource for measurement-based quantum computation. An ALKT is the ground state of a Hamiltonian which has nearest-neighbour two body interactions, shares all the symmetries of the lattice it is based on, and which is rotational invariant in the spin space.This work is particularly useful because it demonstrates what could be a simplified technique for generating the cluster. This result, was a culmination of a lot of work, that was trying to find more physical states that were equivalent to a cluster state [112–115].

8.2.2 Probabilistic schemes

While some of the schemes for generating cluster states have been one shot schemes, the majority of work has concentrated on how to use the probabilistic gates from linear optics to generate cluster states. The initial work in this area concentrated on how to turn a probabilistic gate into a deterministic one, and was focused on optical quantum computing in general, not just cluster state quantum computing. Knill, Laflamme and Milburn (KLM) [20] demonstrated the feasibility of using a simple set of linear optical elements combined with measurement to do universal quantum computing. In particular using teleportation they demonstrate how to turn a probabilistic entangling gate into a "asymptotically unit" probability gate that could be used to build a cluster. This work was built upon by numerous groups including Yoran et al. [116] who demonstrate a scheme that can a create a gate with unit probability rather than asymptotically unit probability, and Nielsen [117] who combines cluster state computation with the KLM scheme [20] to get a significant reduction in the resources required.

The difficulty with a lot of these schemes is that they require a large number of resources to create a single gate. In cluster state computation the entangled resources is created entirely off-line, therefore it should be possible to use these probabilistic gates in their probabilistic form to grow the cluster. The work in this area focuses on the idea of a heralded C-Phase operation occurring between two qubits. With some probability this operation generates the required C-Phase gate, and with some probability it is destructive, removing at least one C-Phase gate from the current chain. This scheme was proposed by Browne et al. [118] and Duan et al. [19] in 2005, as it reduced the resources required, compared to a scheme where probabilistic gates were combined to make a deterministic operation. Browne et al. [118] provide a basic scheme for entangling photons into a cluster, while Duan et al. [19] prove that probabilistic generation does indeed allow the cluster to grow. This work has been built upon by Gross et al. [119] who look at different models for generating clusters from probabilistic gates to work out which is more efficient. In particular they look at a greedy strategy where large chains are fused together, and a modest strategy where small chains are fused together, and show that for small clusters a modest strategy is almost optimal. They also place minimum bounds on the amount of resources that need to be consumed to generate the necessary gates.

The work by Browne et al. [118], and Gross et al. [119] inspired numerous other groups to look at using probabilistic methods for growing cluster states from base elements [120–127]. Gilbert et al. [120] provide speed ups by growing graph-states equivalent to those needed to build a cluster state, and thus simplify the operations needed. Building on the work of Gross et al. [119], Kieling et al. [121] noted that for high success probability it made minimal difference which strategy was used to generate large clusters. Rohde et al. [122] found that if the success probability is small, strategies where you combine chains of equal length perform better than strategies where the chains are different lengths. They also found that for large clusters the generation technique was strategy independent provided you combined chains of equal length. Other work looked at trying to simplify the system experimentally. In particular, Wilde et al. [123] and Gong et al. [124] proposed techniques for building cluster states without the need for photon-number detectors, therefore simplifying the procedure. Many of the above schemes had problems due to error accumulation and the need for re-routing. Campbell et al. [125] consider how certain monitored errors can be corrected for, thereby reducing the number of events which need to be considered failures. Kieling et al. [126] demonstrate a technique to get around the need for re-routing therefore avoiding the conditional dynamics which can make a cluster difficult to create, and Matsuzaki et al. [127] consider a different technique of generation which has a lower error accumulation than previous techniques.

Of particular interest is the work conducted by Louis et al. [25], which looked at generating cluster states on the qubus using probabilistic, then deterministic gates. The work using deterministic gates is summarised in section 8.2.3, however the probabilistic work is still interesting. They showed that by using additional qubits in the bus scheme it was possible to increase the success probability of an entangling gate. An increase from a two qubit entangling gate, to a three qubit entangling gate increased the success probability from $1/2$ to $3/4$. Louis et al. [25] also consider the fusing elements, and the best way to build up the cluster given the qubus scheme. This allows them to demonstrate that their scheme can build a cluster state efficiently.

When combined, all this work shows that using probabilistic gates is a realistic strategy for cluster state computation. However probabilistic schemes can still be resource intensive and difficult to implement in practice. We therefore want to look for a technique for generating cluster states where we have a deterministic entangling gate which does not require a large number of operations to create. Ancilla-based quantum computation seems to offer a solution to this problem.

8.2.3 Deterministic schemes and previous qubus results

While most of the available results have concentrated on using a probabilistic gate, recent work has noted that by using ancillas it becomes possible to generate gates deterministically. Lin et al. [128] consider a specific deterministic entangler, and how to use this to grow cluster states. In particular, given the limited nature of their entangler, they consider what operations can be used to build up an entire cluster. A similar deterministic entangler has been proposed by Devitt et al. [129], which can be used to generate stabiliser states. Ionicioiu et al. [130] build upon this work, and demonstrate how to use a photonic model scheme to generate a cluster state, using an atom in a cavity instead of a parity gate as the photonic module. Both of these schemes would require routing to generate the entanglement. The process of routing might be difficult, however this routing can be passive, and this should be easier to implement. Combined, these papers demonstrate that by using an ancilla it is possible to generate deterministic gates without a large overhead in the number of operations required. This would provide a significant improvement in the feasibility of generating large cluster states. A similar scheme, which used polarising beam splitters as a 'module' to generate entanglement was discussed by Zhang et al. [74]. This is particularly interesting because it is one of few deterministic schemes which do not directly require an ancilla. Unlike the previous schemes where an atom was used as the ancilla, and photons as the qubit, Zhang et al. considered matter based qubits such as electrons or quantum dots. This scheme required a large number of resources either in terms of the number of beam splitters used, or the time required to pass qubits through the beam splitters individually.

One example of work using fixed qubits but a moving bus, was discussed by Zheng [131], in the context of ion traps. This work also showed that, by using a 3 level system to create our C-Phase gates between a quasi two level system, it was possible to perform deterministic operations in an ideal case. However including realistic error values, pushed the fidelity of the gates down to 0.65 or 0.85 dependent upon the exact scheme used. Previous work on deterministic generation, highlights that with current technology, a compromise has to be made to achieve a deterministic gate, whether this is by using an ancilla system, or by starting with a system with more levels than those which will be used computationally.

An alternative to these methods was proposed by Wang et al. [132], who consider using a standard scheme to generate a cluster in optical cavities, then to transfer this cluster to photonic qubits. This is still an ancilla scheme but it uses the ancilla system in a dramatically different way to the other schemes mentioned. It has significant advantages in that the cluster not currently being used can remain in the atomic cluster. It is also an interesting contrast to the qubus scheme, where a photonic field is used to generate gates in a matter system.

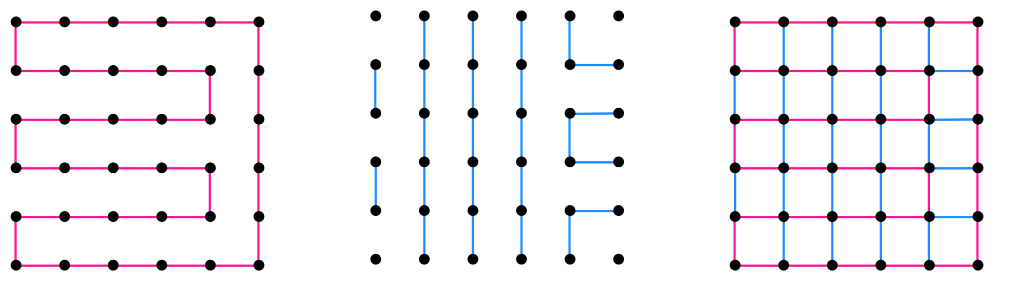
We concentrate our work on an ancilla scheme where the ancilla system is routed, rather than using a module and routing our cluster state. This will allow us to use static qubits to generate our cluster. We will also consider a measurement-free scheme, where we use entangling then disentangling operations. This is often costly in the number of operations required, however measurement is often slow and relatively resource intensive. As in section 4.4 we can consider a naïve technique for generating our cluster state that involves implementing each C-Phase gate individually, using four operations with the bus. We will consider a cluster containing N qubits, with a width of n qubits, and a length of m qubits. It is easy to see from counting arguments that in such a scheme there is a total of $n(m-1)+m(n-1)$ C-Phase gates. In our naïve method we require four operations with the bus to generate each C-Phase gate; therefore we require $8nm-4(n+m)$ operations to generate our entire cluster.

Louis et al. [25] have previously looked at generating cluster states on the qubus using a deterministic method. They demonstrated that by generating the cluster state in the form of rows and columns, it is possible to get significant savings over a naïve qubus method. Most significantly, they demonstrate that it is possible to perform C-Phase gates between a line of n qubits using just $2n$ bus operations. For our $n \times m$ cluster, each qubit is part of one row and one column; if we consider generating C-Phase gates between these rows and columns in lines using the method proposed by Louis et al. [25] then a total of $4N$ bus operations is required.

In fact we can get a small improvement over this technique even if we limit ourselves to generating only lines of C-Phase gates. We consider a path across our qubits to be a line of C-Phase gates with turns were necessary. These turns do not require any additional operations as the qubits in a path could be rearranged as a line without breaking any gates. Clare Horsman argued that if we generate a path that visits all the qubits, then we can can generate C-Phase gates between the qubit of interest, and, at most, two other qubits. However, all the qubits, except the four corner qubits are connected by a C-Phase gate to either three or four other qubits. It should be possible to generate our entire cluster using two paths of gates, one path which visits every single qubit, and the second line which visits $N-4$ qubits. A lower bound on the number of operations required for generating a cluster state using our single line of C-Phase gates can therefore be given by $4N-8$.

By choosing the two paths carefully we demonstrated that it is possible to saturate this lower bound. We illustrate this in figure 8.1, where the black dots represent qubits; the pink path in figure 8.1(a) is a line of C-Phase gates of length N and the blue path in figure 8.1(b) is a line of C-Phase gates of length $N-4$. If two qubits are connected by either a pink or blue line, then they are connected by a C-Phase gate. Figure 8.1(c) illustrates the fact that when combined, the pink and blue paths do indeed create all the necessary C-Phase gates to generate a complete cluster.

The method proposed by Louis et al. [25] is almost a factor of two improvement over the naïve method. With the difficulty in creating quantum gates and maintaining quantum coherence, this represents a significant improvement. One major problem with the scheme proposed by Louis et



(a) A path visiting all N qubits (b) A path visting $(N-4)$ qubits (c) A combination of the paths

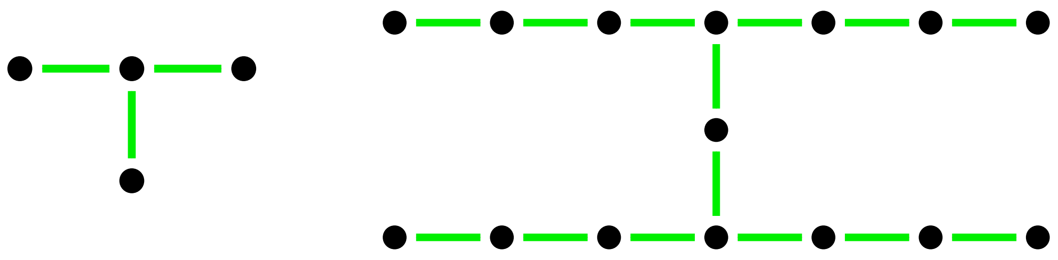
Figure 8.1: The pink and blue paths are lines of C-Phase gates. From these diagrams we can see the path in figure (a) combined with the path in figure (b) generate C-Phase gates between all the qubits. The total combined length of these two paths is $2N-4$, and they require a total of $4N-8$ bus operations to create.

al. [25], and our extensions to the scheme, is that the cluster state is not generated dynamically. In the naïve scheme we can generate the top of the cluster first, and begin to perform operations on this while we create the rest of the cluster. This is not possible in the scheme based on lines of C-Phase gates and could therefore lead to a problem with the cluster undergoing decoherence before we have the opportunity to use it.

8.3 Mapping a circuit to a cluster state

In this section we will briefly discuss results that show that it is possible to efficiently map a quantum circuit to a cluster state. We will also discuss some basic results on characterising entanglement within cluster states, which show they are fundamentally different from W and GHZ states. This has led to work looking at the power of these states in the context of measurement-based quantum computation.

Mapping a quantum circuit to a cluster state, so that it can be used to perform one-way quantum computation, was an integral part of the proof that measurement-based quantum computation was universal. Raussendorf and Briegel discuss this in their seminal paper on the work [18]. This work is of interest to us because it allows us to clarify the size of the cluster required to perform useful calculations. While the mappings provided by Raussendorf and Briegel are not optimally efficient, we see that four qubits arranged in 2 rows, and 3 columns of our cluster are required for a single CNOT gate. This gate relies on having information about the initial state of the qubits between which you want to perform the CNOT. The necessary entanglement structure for this gate is shown in figure 8.2(a). An arbitrary rotation requires a chain of 5



(a) If the inputs are known (b) If the inputs are unknown but on adjacent qubits

Figure 8.2: The minimum amount of cluster required to build a CNOT gate

qubits. It would of course be possible to consider alternative ways to make this structure, but this provides us with a reasonable limit. A 6-qubit cluster state was recently created by Lu et al. [133] using a standard probabilistic fusion method with post selection. The potential uses of this state, other than as a cluster state were explored by Paul et al. [134].

While this initial work into performing operations showed a suitable mapping which was efficient, it only discussed the issues of universality and efficiency briefly. Along with Browne, Raussendorf and Briegel extended their work in a later paper [135]. In this paper they introduce a larger form of the CNOT gate which is more general, and can be used as part of a circuit. The necessary cluster for this gate is shown in figure 8.2(b). They also demonstrate numerous circuits for performing useful quantum algorithms such as the quantum Fourier transform, as well as introducing building blocks such a CNOT gate between distant qubits. Combined, the two papers show the feasibility of using cluster states for quantum computing. However the cluster state is still resource intensive, particularly in terms of entanglement. Therefore work still needs to be done on how to arrange the circuit so that it uses a small amount of the cluster as possible. An example of such work looks at how to build a quantum adder on a cluster state while reducing the temporal resources required [136]. As with circuit based quantum computing, finding the most efficient way to perform a desired algorithm is incredibly important, since entanglement is an expensive resource with a short shelf life.

The level of entanglement within a cluster state is actually incredibly important. Briegel and Raussendorf [137] characterise the nature of the entanglement within a cluster state. They note that while it has many similarities with a GHZ state, the entanglement is actually more robust. While it is possible to destroy all the entanglement within a GHZ state with a single measurement, the same is not true of the spin-chain (1D cluster state) or the cluster state. Indeed while for $N = 3$ the GHZ state and spin-chain are equivalent, this is not true for larger states. It has been demonstrated that too much entanglement is detrimental for cluster states and quantum computing in general [138]. Meanwhile GHZ states are universal for classical measurement-based computation when combined with classical feed forward and quantum measurement [139]. Therefore we see that the strength and degree of entanglement makes a significant difference in the power that can be achieved with the entangled resource.

8.4 The advantages of cluster states

One of the principal advantages of cluster states is that once the initial resource has been generated we can reduce the number of time intervals required to perform our quantum gates. Cluster states are particularly useful in this context because all Clifford group operations can be performed in a single time step [135]. The Clifford group is a particularly useful set of operations which include CNOT, σ_x, σ_y, σ_z and the Hadamard. They are a classically simulatable set, however when accompanied by a single non-Clifford rotation, they are universal for quantum computing [33]. Unfortunately, being able to perform all our gates in a single step is no longer possible when the cluster is split into smaller sections to allow fault tolerance. However, in terms of circuit depth, cluster state models have been shown to be equivalent to the standard quantum circuit model with the addition of the fanout gate [21]. While the theoretical idea of adding the fanout gate to the circuit model does not provide an increase in computational power, it has been shown to reduce the circuit depth from logarithmic to constant, for circuits such as the Quantum Fourier Transform [22]. This reduction in gate complexity, shows one of the most significant advantages of measurement-based quantum computation. The ease of parallelising operations on the cluster state compared to in the circuit model has led to work where the one-way model is an intermediary calculation step that aids in parallelising operations in the circuit model [140].

Another significant advantage of measurement-based quantum computing, is that a lot of work has been done on using certain forms of cluster states in a fault tolerant way. This represents one of the principal fields of research currently being conducted on cluster states. Most fault tolerant protocols concentrate on 3D clusters and involve performing topological codes [141]. These topological codes were initially proposed in the context of anyonic computing [142], and have been adapted into more physical models [143]. Other work has begun looking at techniques for making the resource fault tolerant from the ground up, taking into account both generation of the fundamental resource and the implementation of operations [144]. In particular it is important to have error correction models to detect faults within the cluster itself. Surface codes are used to detect errors within the entanglement generated as part of our cluster, and allow that to be corrected for. Briegel et al. [145] provide a useful review of cluster state computation, which includes a large discussion on fault tolerance. However, we note that most work on fault tolerance, and error correction assumes that any errors within the cluster are uncorrelated. In section 10.5, we discuss both the impact this has on our work, and the possible developments to avoid these issues.

One of the most significant disadvantages of measurement-based computation is that it consumes a large amount of entanglement to perform the necessary gates. Even a single qubit rotation requires entanglement between 5 qubits, while a two qubit gate, such as a CNOT gate requires entanglement between 15 qubits. The entangled gates being consumed are maximally entangling gates, equivalent under local unitaries to CNOT gates. Therefore, to perform the equivalent gates in the circuit model, in an ideal case, would require 1 entangling operation per two qubit gate, and none for a typical one qubit rotation. In reality though, performing both entangling and general single qubit unitaries can be a hard task. This has led to the proposal of schemes where either entangling operations, local operations, or both are performed by an ancilla system [41, 146–149]. Our qubus system is one of these architectures [2].

Finally it is worth mentioning that for measurement-based quantum computation, all the entanglement is generated off-line. This might not actually seem like a significant advantage, particularly since as we mentioned above, the cluster state uses more entanglement than an equivalent circuit based scheme. However, it does allow us to use probabilistic methods to generate our cluster, and therefore increases the feasibility of being able to generate such a large resource. Another significant advantage is that the first layer of error correction can be performed before we use our cluster to do calculations. This would allow us to discard a highly error prone cluster, and thus ensure the accuracy of our results. A potential disadvantage of this scheme is that it is possible that the cluster will undergo dephasing before we perform operations on it, and thus cause errors. This highlights one of the principal problems with this form of quantum computing. If we could store the entire cluster for large periods of time, then it would be possible to perform all the Clifford group operations in a single time step, wherever they are in our cluster. Obviously if we consider splitting our cluster into sections which are operated on at a separate time then this would be impossible. We therefore see that we have to trade off the speed so that we can perform operations in a fault tolerant way.

8.5 Conclusions

In this chapter we have introduced the idea of cluster state computation. We have outlined several theoretical schemes to efficiently build cluster states. The most efficient of these build the entire cluster in a constant number of operations, independent of the cluster size [18, 109]. We have also discussed work which shows that it is possible to build a large cluster with heralded probabilistic generation. Building the necessary gates in a deterministic way is often impractical, therefore many of the practical schemes proposed to do this require some form of compromise. This is often either adding in an ancilla system, or building our cluster from 3 state systems, then only using 2 of these states for operations on our cluster.

We introduced previous work done on the qubus for generating cluster states [25]. In this paper Louis et al. looked at using both probabilistic, and deterministic methods for generating cluster states. We outlined their deterministic scheme, and showed some basic improvements, reducing the number of operations required from $4nm$ to $4(nm-2)$. In later chapters 9 and 10 we discuss alternative techniques for generating cluster states using the qubus.

In section 8.3 we briefly mentioned work on mapping a quantum circuit to a cluster state, and how this had been adapted to reduce the total number of gates required. We also highlighted the main advantages of cluster state computation in section 8.4. The two most significant advantages are that cluster states are relatively easy to make fault tolerant, and that all the entanglement is generated off-line. This however does come at a cost, and we show that a cluster state uses more entanglement than the circuit model to perform the same operations.

Cluster states are a useful, and different architecture for quantum computing. The relative simplicity in the entangling gates required, leads some people to think they are the future of quantum computing, however others disagree due to the large amount of entanglement required. Either way they are an interesting model of computation, that is likely to have a significant impact on quantum computing.

Chapter 9

Generating cluster states layer by layer

9.1 Introduction

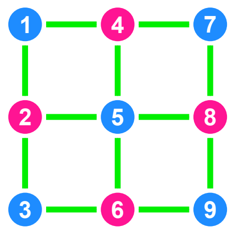
In section 8.2.3 we discussed some previous work on generating cluster states that used a line of qubits to generate the cluster. We will now consider expanding this work to look at building our cluster from layers larger than a single line. By using these layers we will keep the generation of our cluster dynamic while speeding up the generation process.

In section 9.2 we discuss the limitations of the qubus system for generating cluster states. In particular we consider the limits on the size of the largest layer of cluster we can create, while interacting our bus only once with the relevant qubits. This allows us to explore possible patterns for combining our layers, something we discuss in section 9.3. In this section we look at reducing the total number of layers needed, so that we can decrease the total number of operations required as part of our transitions. We show two methods to do this which minimise the number of operations given the techniques we are considering.

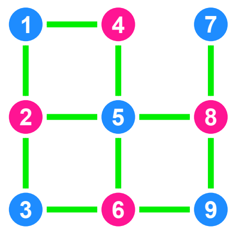
In section 9.4 we lower bound the number of operations needed to build our cluster. We compare this result with results from the previous sections, and therefore calculate how expensive it is to generate the cluster in a layered fashion. By adapting our layered method so that we stitch our layers together during creation, it is possible to save additional operations, and thus to get closer to our lower bound. We discuss this in section 9.5. We conclude this chapter in section 9.6.

9.2 The limitations of the bus

To begin placing a limit on the number of operations required to generate our cluster state, we will consider what is the widest path of C-Phase gates our bus can generate is, if it is only allowed to interact with each qubit once, and can not turn corners. We will call this path of



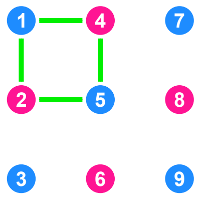
(a) A 9 qubit box.



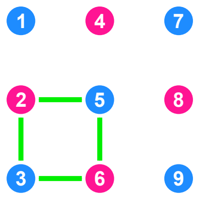
(b) The largest section that can be created interacting each qubit with the bus once

Figure 9.1: The box of 9 qubits in figure (a) can not be created using just one pair of interactions with the bus per qubit; however the section in subfigure (b) can. The green lines represent C-Phase gates between the qubits

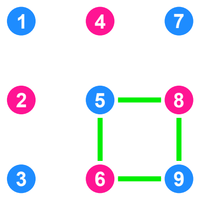
C-Phase gates with no turns, a layer. In order to generate entanglement between neighbouring qubits, it is necessary that nearest neighbour qubits interact with opposite quadratures of the bus. We will illustrate this by colouring our qubits pink and blue; where pink qubits always connect to the momentum quadrature of the bus, and blue qubits always connect to the position quadrature of the bus. We will consider building our line of connected qubits along the x-direction.



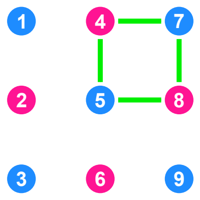
(a) Corner 1



(b) Corner 2



(c) Corner 3



(d) Corner 4

Figure 9.2: A 9 qubit box can be split into four sections which need to be created separately. The green lines represent C-Phase gates between the qubits.

The largest sealed shape we can create has width 2. We can prove this by considering a box of 9 qubits as shown in figure 9.1(a). This box consists of four sections which are shown in figure 9.2. For each section, the corner qubit has to interact with one qubit with which the corner qubit of the adjacent section does not interact. For example in figure 9.2(a) qubit 1 interacts with qubit 2 and 4, but not with 6 and 8, while in figure 9.2(b) qubit 3 interacts with qubit 2 and 6, but not qubit 4 and 8. This means that we can not create two corners at the same time, since we need both to add qubits to the bus, and remove qubits from the bus between them. We therefore consider, creating our sections sequentially. Since, one section always has a qubit in common with the next section, it makes sense to create them by rotating around our 9 qubit box. However, this leads to a problem since our final section will be adjacent to our first section, therefore they will have a qubit in common. This means to create an entirely sealed section like the one shown in figure 9.1(a) we would need to interact one qubit with the bus twice. We are therefore limited to the section shown in figure 9.1(b).

This suggests that it might be possible to create a section of the form shown in figure 9.3(a), however this is not possible. At the end of generating the piece shown in figure 9.1(b) we have left qubits 7 and 9 connected to the position quadrature of the bus, and qubit 8 connected to the momentum quadrature. As a result we have a choice between creating the section with qubit 9 as a corner, or the section with qubit 7 as a corner. While it would be possible to create a section such as the one shown in figure 9.3(b) this does not contain any more connections than figure 9.3(c) so it is not beneficial.

We can extend this and look at how wide an unsealed section can become. It is easy to see that the section can not be wider than 4 qubits, because if it was, it would require a sealed path of greater than 2, something we just showed was impossible. We therefore want to demonstrate that it is possible to perform the 4 qubit path. We will demonstrate this diagrammatically, as this is easier to visualise than a list of displacement operators. For completeness a list of displacement operators for each set of diagrams is provided in Appendix A.

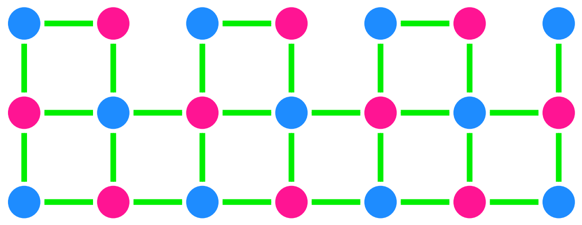
9.2.1 About the diagrams in this chapter

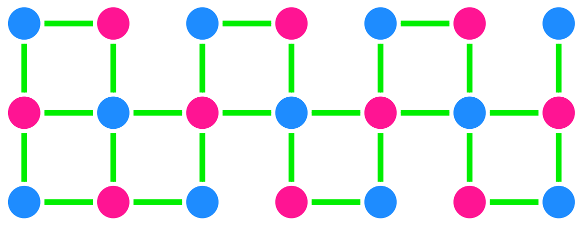
It is worthwhile establishing some common notation for our diagrams, that will be used henceforth. To reduce the total number of diagrams we will consider displaying several operations in each diagram shown. Each set of operations will consist of one sequence of operations where qubits disconnect from the bus, and one sequence of operations where qubits entangle to the bus. These operations often need to be performed in a specific order, our diagrams do not allow this order to be reconstructed simply by inspection, however they do clearly illustrate that such a sequence can be found. We will colour our qubits in diagrams where they are entangled to a particular quadrature of the bus, pink qubits are connected to the momentum quadrature, and blue qubits to the position quadrature. A qubit that is not entangled to either quadrature of the bus will be represented in black.

We now need to consider how operations work together to make a gate; if we consider just two displacement operators then we have,

$$D\left(\sqrt{\frac{\pi}{8}}\sigma_{z1}\right)D\left(-\sqrt{\frac{\pi}{8}}i\sigma_{z2}\right) = \exp\left(\frac{i\pi}{8}\sigma_{z1}\sigma_{z2}\right)D\left(\sqrt{\frac{\pi}{8}}(\sigma_{z1} - i\sigma_{z2})\right) . \tag{9.1}$$

This only gives us half of our desired interaction, $\exp(i\pi\sigma_{z1}\sigma_{z2}/4)$, and also results in a net


(a) The ability to create figure (9.1(b)) means we wonder if we can create a shape such as this.


(b) However the largest shape we can create it this one.

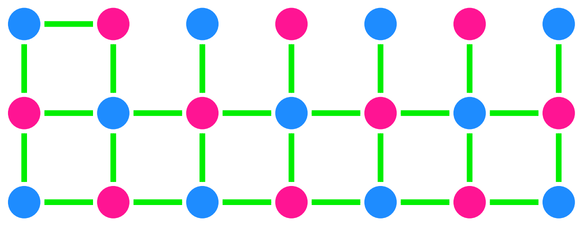

(c) Which does not have any significant advantages over this section.

Figure 9.3: It is not possible to create a section as large as the one in figure (a). Both figure (b) and (c) are possible. These both consist of the same number of C-Phase gates.

displacement operation on the bus. We generate the second half of our interaction by removing the qubits from the bus such that

$$D\left(\sqrt{\frac{\pi}{8}}\sigma_{z1}\right)D\left(-\sqrt{\frac{\pi}{8}}i\sigma_{z2}\right)D\left(-\sqrt{\frac{\pi}{8}}\sigma_{z1}\right)D\left(\sqrt{\frac{\pi}{8}}i\sigma_{z2}\right)=\exp\left(\frac{i\pi}{4}\sigma_{z1}\sigma_{z2}\right). \quad (9.2)$$

Therefore in our diagrams we want to show this interaction being created in two parts. A single line will represent the interaction $\exp(i\pi\sigma_{za}\sigma_{zb}/8)$ between qubits a and b, with two lines being required for a maximally entangling interaction. Since all our interactions commute these two lines do not need to be generated by sequential operations. An orange line will represent the fact that the interaction is being generated by displacement operators in the set illustrated on the diagram. A dark blue line will represent an interaction that has been generated previously.

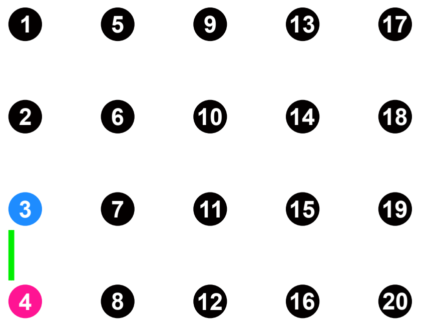
9.2.2 An open path of width four and unbounded length

In 9.4 we illustrate how to generate an open path of width 4. This figure is split into several subfigures, each of which represent a set of displacement operators being performed on the bus. It is possible to see that such a section can be generated by interacting each qubit with the bus a single time, by observing that in figure 9.4 each qubit only connects to the bus once (represented by the qubit being illustrated in pink or blue). A corresponding list of displacement operators to this section is shown in Appendix A.1.

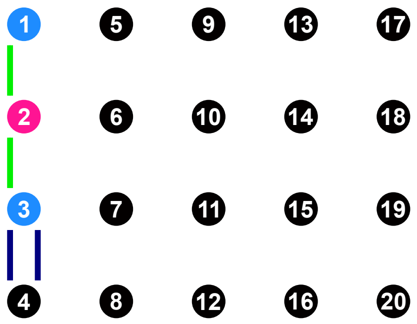
Since we can generate the open path of width 4 it is easy to see we can generate an open path of width 3 or a closed path of width 2, as these are just the path of width 4 with certain operations removed. We can now proceed, and see how many operations will be required to generate our cluster in a layered fashion.

9.3 A simple method for combining layers

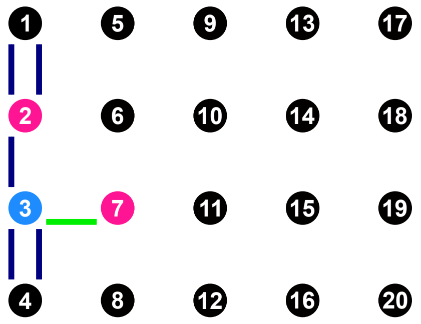
The simplest possible method for combining layers when generating our cluster involves removing our bus entirely from the qubits between each layer. In section 9.5 we show that it is possible to achieve further savings by keeping qubits active between the generation of two layers, however for now using this crude technique of connecting our layers allows us to consider what technique is most efficient for creating our large cluster. To compare our various techniques we will consider only the ‘additional’ interactions required. These are the number of operations above the minimum possible $2nm$ that are required if no qubit has to interact with the bus more than once. We can find these by considering the overlap between layers.



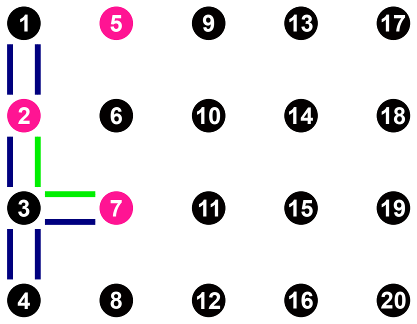
(a) Stage1 of the generation process, this is the activation stage. We begin by entangling qubit 4 to the momentum quadrature of the bus, then entangle qubit 3 to the position quadrature of the bus.



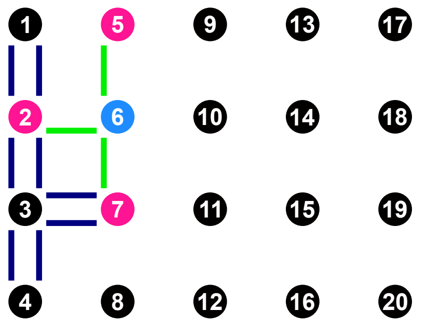
(b) We now remove qubit 4 from the bus, before connecting qubit 1 to the position quadrature, and qubit 2 to the momentum quadrature of the bus.



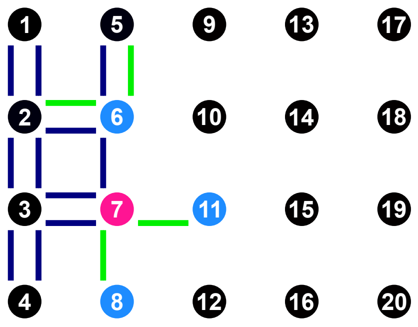
(c) Qubit 1 is disconnected from the bus, and then qubit 7 is attached to the momentum quadrature of the bus.



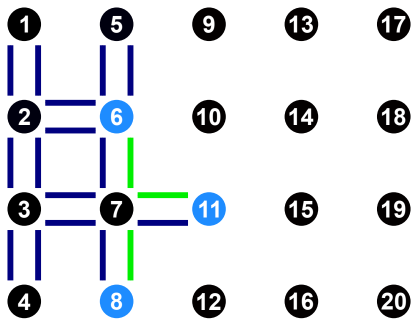
(d) Qubit 3 is removed from the bus, then qubit 5 is attached to the momentum quadrature of the bus. This unit is the first of a set of four repeated units that allow us to grow the cluster to the necessary length.



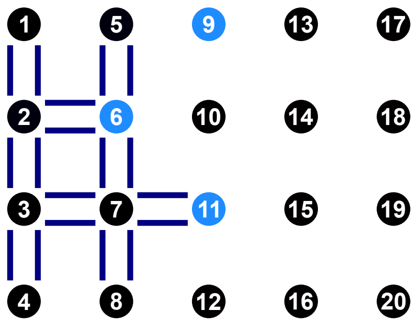
(e) In the second of our repeating units qubit 6 is attached to the position quadrature of the bus.



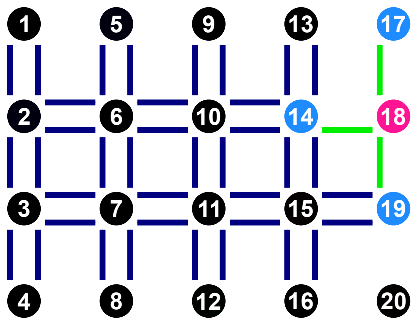
(f) We now remove qubits 2 and 5 from the momentum quadrature before entangling qubits 8 and 11 with the position quadrature of the bus.



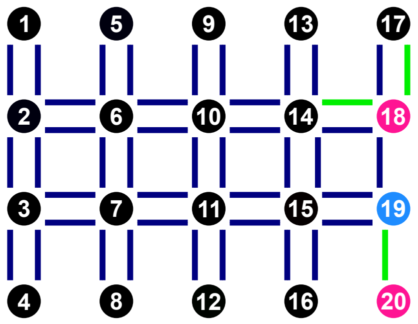
(g) We now disconnect qubit 7 from the bus.



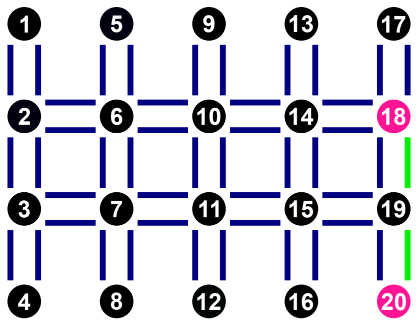
(h) We remove qubit 8 and entangle qubit 9 to the position quadrature, to continue growing. We repeat our steps from 9.4(e) onwards.



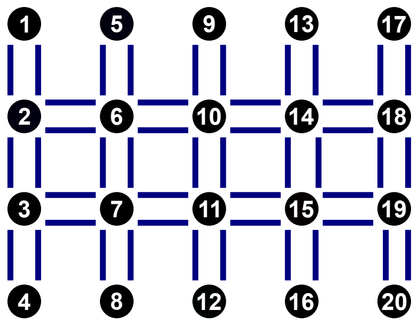
(i) We now want to consider how to disconnect our qubus from the qubits.



(j) We remove qubits 14 and 17 from the bus, then connect qubit 20 to the momentum quadrature.

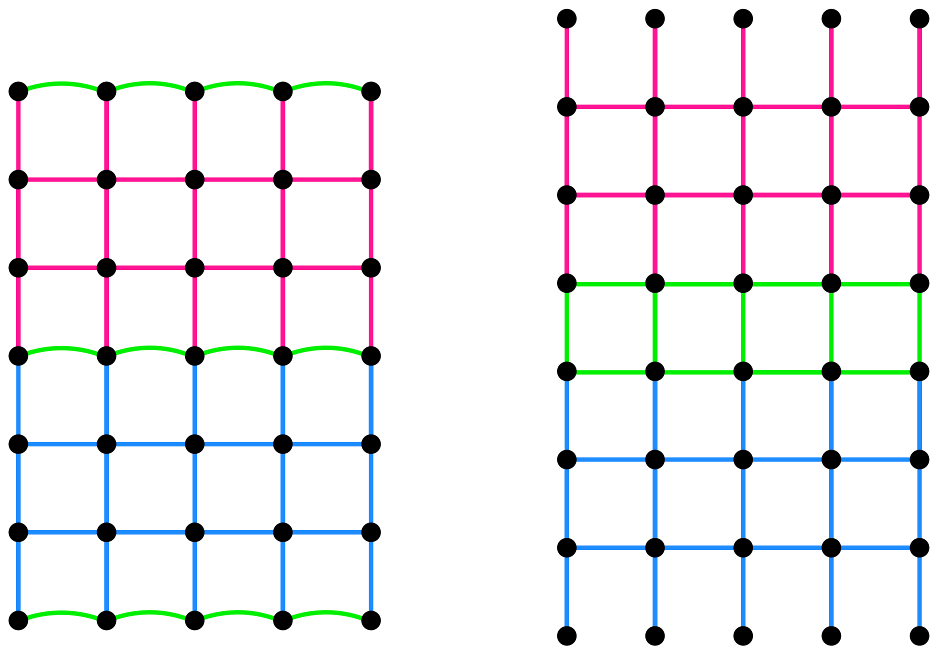


(k) Qubit 19 is removed from the bus and all necessary interactions have been generated.



(l) Finally we disconnect qubits 18 and 20 from the bus.

Figure 9.4: This figure shows how to generate an open path of width 4, while only connecting each qubit to the bus once. This can easily be seen by the fact each qubit only becomes coloured once. A colour key for this table is discussed in section 9.2.1. A list of the corresponding displacement operators is given in Appendix A.1.



(a) A cluster made up of several layers of width 4 open at either end. The layers are shown in pink and blue, and are stitched together using the green 'thread'.

(b) A cluster made up of layers of width 4 interspersed with layers of width 2. We can seal the ends either by stitching them closed or by using a layer of width 2 and increasing the size of the cluster.

Figure 9.5: This figure shows two techniques to generate a cluster using a path of width 4. Subfigure 9.5(a) involves stitching the qubits together using a 'thread' that is a path of C-Phase gates between single qubits. The method in 9.5(b) involves alternating between a path of width 4 and a path of width 2.

9.3.1 Using a path of width 4 for the largest layer

The first technique we consider is using a path of width 4 to create each layer, then stitching the layers together using a string of qubits. This is illustrated in figure 9.5(a). To work out the total number of operations required, we need to consider how many qubits interact with the bus more than once. Every qubit on the join of two sections has to interact with the bus three times; once as part of each layer, and once as part of the stitching. Each of these qubits therefore requires 4 operations with the bus, above the standard 2 required for all qubits. Since each layer has width 4, the number of qubits that are between two layers is given by $n(m-4)/3$. An additional $4n$ operations are required to seal the top and bottom of the cluster. We can therefore see that the total number of operations, above the standard $2nm$ required for generating a cluster is,

$$\mathrm{Ex}_{4,1} = \frac{4nm}{3} - \frac{4n}{3} \tag{9.3}$$

An alternative method involves using a path of width 4 for every other layer, with sealed boxes of width 2 providing the intermediate layers. This is illustrated in figure 9.5(b). In this case we need to repeat two rows of qubits (requiring an additional $2n$ operations per row) every $(m-4)/4$ rows. We also need to close the top and bottom using an additional $4n$ operations. Therefore in this case the total number of operations above our standard $2nm$ is given by

$$\mathrm{Ex}_{4,2} = nm\,. \tag{9.4}$$

To be exact, our two methods require m to take two different forms. The first assumes that m takes the form $3r+1$ where r is an integer, while the second assumes m takes the form $4q$ where q is an integer. To make a comparison between the two techniques, we therefore need to assume that m is large, since in this case the difference in the height of the cluster generated in our two techniques is a very small percentage of the total height. It is clear to see that if we make this assumption then the second technique is the most efficient.

We now consider making a small adaption to this second technique. By having our cluster start and finish with a layer of width 2, we can increase the width of our cluster by 2 for an additional $4n$ operations. To illustrate how this changes the efficiency of the generation process, we replace m with m' where $m' = m+2$. The additional operations required to generate the extra layer are incorporated into our $2nm'$ therefore we only require

$$\mathrm{Ex}'_{2,4} = nm' - 2n \tag{9.5}$$

above our standard number. Using boxes instead of simply stitching the ends is slightly more efficient.

9.3.2 Using a path of width 3 for the largest layer

An alternative technique involves using a path of width 3 for the largest layer, since this will only require stitching on one end. The layer of width 3 consists of a sealed layer of width 2 with free edges as illustrated in figure 9.4. However, in this case the free edges are only on one side of our layer of width 2. We will refer to these free edges as tails. If our entire cluster is made of layers of width 3, we see that $n(m-3)/2$ qubits interact with the bus as part of two separate sections. Each qubit that is part of 2 sections will require 2 operations above the standard number needed to make a cluster. An additional $2n$ operations are also required to seal the cluster, therefore a total of

$$\mathrm{Ex}_3 = nm - n \tag{9.6}$$

extra operations are required to generate the entire cluster. Once again we want to consider what happens if we use a layer of width 2 to close our cluster rather than just stitching it closed with a line of qubits. In this case we need to consider replacing m with $m' = m + 1$. The additional operations needed to add this additional line of qubits are entirely contained within the $2nm'$ minimum for a cluster. Therefore, the additional operations above the standard minimum is given by

$$\mathrm{Ex}'_3 = nm' - 2n\,. \tag{9.7}$$

We can see that we have two equally efficient techniques for generating the cluster. The first involves alternating between sealed layers of width 2, and open layers of width 4, while the second involves using open layers of width 3 and a single sealed layer of width 2 at the start or end of the cluster. One difference between these two techniques is the restrictions placed on m. While the technique involving open layers of width 4 requires $m = 4q$, the technique using open layers of width 3 requires $m = 2w$. In the next section we will established a lower bound on the number of operations required to generate our cluster. Then in section 9.5, we will make a full comparison between the two techniques by considering how many qubits can be left connected to the bus at the end of generating each layer, therefore establishing how close to our lower bound a layered method of generation allows us to reach.

9.4 A lower bound on the number of operations required

In our work on errors within the bus, Clare Horsman argues that we can consider a path of width 2 to be a fundamental unit of our cluster generation [150]. She derives a bound on the number of operations required by considering a path of width 2 visiting every qubit in the cluster. This generates $(3nm-4)/2$ C-Phase gates. Whereas before we were considering layers, we are now considering a path that allow turns. We concentrate on the path of width 2, as turns in this path are free, while turns in larger paths can be costly. In the next section we will consider how we would generate these paths by considering how we stitch layers together. For now we consider this simplified path so that we can get a lower bound.

Our entire cluster consists of $2nm - n - m$ C-Phase gates. Therefore the path of width 2 leaves $nm/2 - n - m + 2$ gates to be created separately. In an ideal scenario, all of these gates could be created using the tail of trailing qubits that can be added to our path of width 2, and therefore only 2 additional operations will be required for each additional C-Phase gate. While this could add extra turn costs, at present we do not want to prove that we can saturate our given lower bound, so we will ignore these additional operations. We can therefore bound the number of operations required to build the entire cluster as

$$\mathrm{Cl}_{\text{min}} = 3nm - 2n - 2m + 4 \tag{9.8}$$

with the extra above the standard $2nm$ given by

$$\mathrm{Ex}_{\text{min}} = nm - 2n - 2m + 4\,. \tag{9.9}$$

We can see that this is considerably fewer than the additional number of operations required in the previous section. This presents a lower bound on the number of operations required in the case of our width 2 path. Since we cannot create a wider sealed path without interacting qubits with the bus several times, this seems a feasible lower bound. Using a less physical system than the qubus it might be possible to create wider paths without interacting qubits twice. However this would involve defining the behaviour of the system and constructing it artificially, which is beyond the scope of this work.

9.5 Saving operations when connecting layers

We have established the total number of operations required to generate a cluster in a layered fashion is given by

$$\mathrm{Cl}_{\text{layer}} = 3nm - 2n\,. \tag{9.10}$$

In section 9.4 we placed a lower bound on the number of operations required which was given by

$$\mathrm{Cl}_{\text{min}} = 3nm - 2n - 2m + 4\,. \tag{9.11}$$

In both optimal cases discussed in section 9.3 we switched layers a total of $(n-2)/2$ times. Therefore to meet the minimum bound we would need to make a saving of 4 operations per transition. In this section we show that making a saving of 2 operations is trivial, and can be done for every transition. However, making a saving of 4 operations is less trivial, and can be done at most every other layer. In this section we introduce the idea of a hanging edge, which is a single tail connected to the bottom of one layer. This hanging edge assists in the creation of a later layer, and as we show provides a significant advantage.

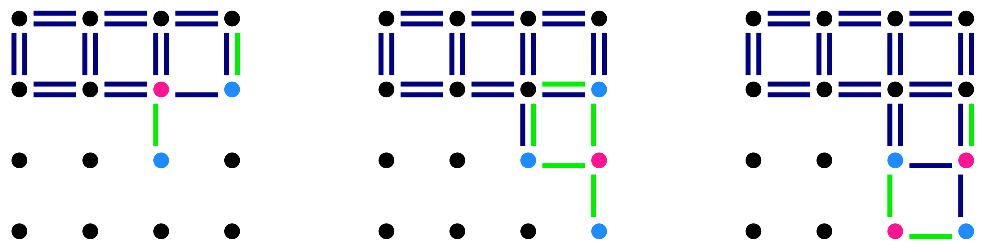
9.5.1 Saving two operations per transition

We can easily see why saving two operations is trivial if we consider how we go about generating each layer. In figure 9.4 we see that we start generating our layer on one of the corner qubits. It should always be possible to choose this corner to be the top left or right corner, therefore we should never have to deactivate the bottom corner qubit from the layer before. It is easy to see that, it will always be possible to generate our cluster in

$$\text{Cl}_{\text{layer}} = 3nm - 2n - m + 2 \tag{9.12}$$

operations. In the next few sections we will look at when it is possible to save 4 operations in our transition, so that we can work out how close to the lower bound our connected layer method allows us to reach.

9.5.2 Transferring between a layer of width 2 and a layer of width 3



(a) The end of the layer of width 2.

(b) We continue the path of width 2 around the corner.

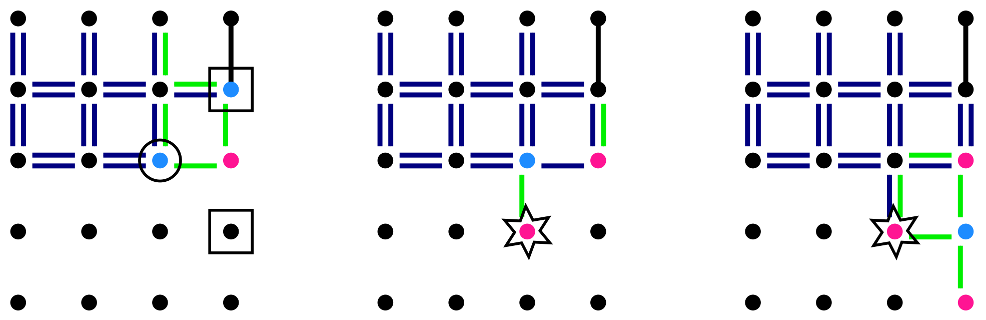
(c) We can now create our layer of width 3.

Figure 9.6: We connect a layer of width 2, to a layer of width 3. Two qubits which would have to interact with the bus twice if we did not stitch our layers together, have to interact with the bus only once in this case.

Figure 9.6 shows that it is possible to connect our layer of width 2, to a layer of width 3 while saving 4 operations compared to generating each layer separately. The key thing we note from the figure is that connecting our two layers is equivalent to a path of width 2 with a turn, and with tails on the top of the second layer. This makes it unsurprising that we can create this join while saving 4 operations. The operations saved come from the bottom right hand corner of our layer of width 2.

9.5.3 Transferring between two layers of width 3

In figure 9.7 we can see that if we have the hanging edge shown in black already created, then we can make a saving of four operations when transferring between two layers of width 3. Without this hanging edge, only two operations could be saved. We can see this by looking at figure 9.7(a), where we would need to remove and then reconnect the circled qubit from the bus so that we could create the operation shown in black. The hanging edge provides us with a significant



(a) The tail means we do not have to deactivate the circled qubit.

(b) Therefore 2 qubits remain active in the transition.

(c) Thus we can save a total of 4 operations.

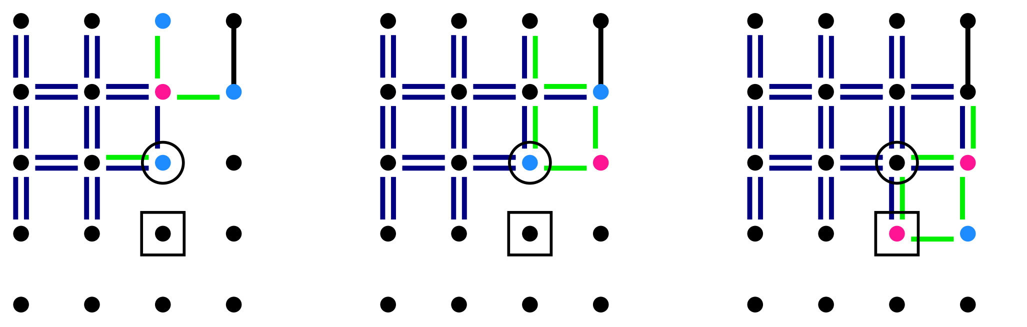
Figure 9.7: We connect a layer of width 3, to another layer of width 3, where the link in black has already been created. This saves us 4 operations.

saving in operations since it removes the needs for the two boxed qubits to both connect to separate qubits. Both of these boxed qubits connect to a qubit that the circled qubit does, as well as their corner qubits. With the limitations of ordering this means we can not create the extra links without disconnecting the circled qubit, or an alternative nearby qubit.

By considering the previous two diagrams we can see that it would not be possible to create a hanging edge in either of these two cases, unless we used extra operations to disconnect then reconnect a qubit. This is because in figure 9.6(b) and 9.7(c) the bottom corner of our layer of width 3 is created without removing the starred qubit from the bus. If we wanted to add in a hanging egde, we would at some point need to have this starred qubit disconnected from the bus, but the bottom corner qubit connected to the bus. It is easy to see that this is impossible without having one of the qubits in question (or a nearby qubit) interacting with the bus twice. As we are unable to create this hanging edge in these cases, it is quite clear that we would also be unable to create our path of width 4, since this is essentially our path of width 3 with an additional set of tails.

9.5.4 Transferring between a layer of width 4 and a layer of width 2

We also want to consider the transitions when we have layers that alternate between being of width 2 and width 4. Figure 9.8 demonstrates that it is possible to save 4 operations, if we are transitioning from a layer of width 4 to a layer of width 2, provided that we have the hanging edge shown in black already generated. As before it is impossible to save these 4 operations if we do not have this hanging edge generated, since we would need to reactivate the circled qubit. Once again we can see from this diagram that it would be impossible to generate an


(a) A layer of width 4.

(b) The tail delays activation of circled qubit.

(c) We can now create the path of width 2.

Figure 9.8: When connecting a path of width 4 to width 2, the previously created operation shown in black means we can delay interacting the circled qubit with the bus. This means it only needs to interact with the bus once, not twice.

additional hanging edge. This is because the boxed qubit, and the qubit in the bottom right hand corner both need to connect to the same qubit, as well as separate qubits. As we do not want to remove the boxed qubit from the bus, but have to connect it, either before or within the same sequence of operations as the corner qubit, we would need to remove one of the nearby qubits from the bus to create all the necessary operations. It would be difficult to change the order of our operations, because we are transferring directly from the previous section.

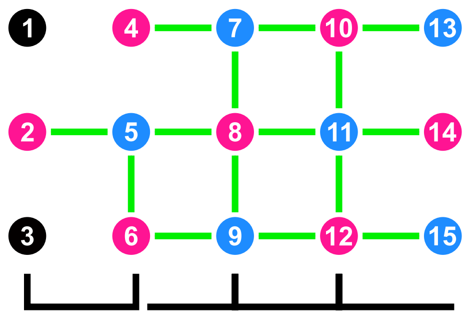
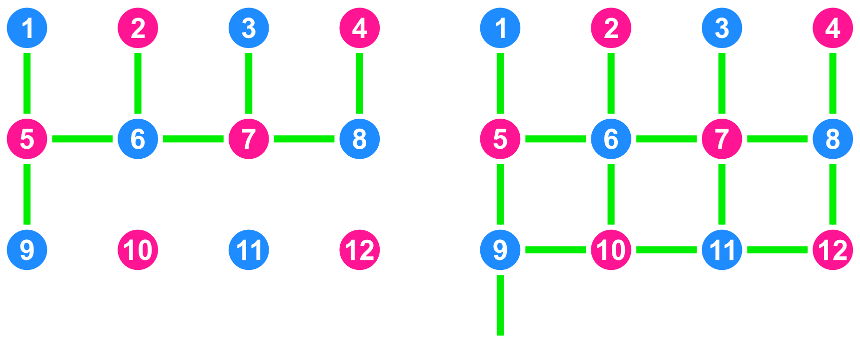


Figure 9.9: This figure shows why it is impossible to connect a layer of width 2 to a layer of width 4 with a saving of 4 operations, since at some point we would need to disconnect and then reconnect qubit 8.

If we look at a transition between a layer of width 2, and a layer of width 4, then a pre-created hanging edge will not provide any saving in the number of operations required. This is because we need to create a section such as the one shown in figure 9.9. In this case our layer of width 2 contains qubits 1-6 and is marked in black, to fit the diagram in the text better, the layers run in the vertical direction. The first layer runs down the page, while the second layer runs up. When we create the second layer we are connecting from the previous layer, so we have to start at the bottom left hand corner. This means it is impossible to connect qubit 12 to qubit 15 without first disconnecting qubit 8.

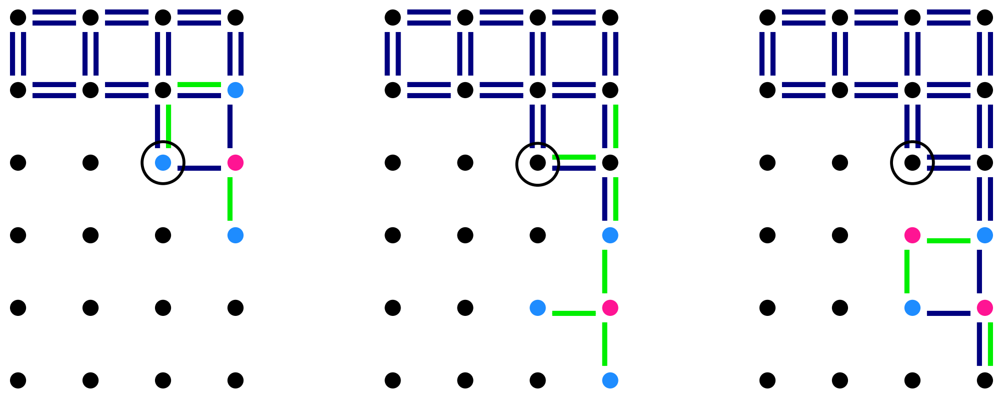
9.5.5 Creating hanging edges



(a) A layer of width 2 with a tail. (b) A layer of width 3 with a tail.

Figure 9.10: This figure shows a layer of width 2 with a tail, and a layer of width 3 with a tail. These can be created easily if we assume the layer is the first layer of our cluster.

Finally we want to prove that adding our hanging edges does not require any extra operations. Above we showed several cases where it was not possible to create a hanging edge. We will now consider the cases where this is possible. It is easy to see that if we consider building a layer of width 2 from scratch, a hanging edge can be added with only two extra operations (those needed to interact with the qubit to which the hanging edge is connected). We can see this in figure 9.10(a) where the hanging edge connects to qubit 5. Figure 9.10(b) shows a similar result for when we are building our layer of width 3 from scratch. We find that adding the hanging edge is possible, provided we want to add the hanging edge to the sealed side of our path of width 3. As we can easily create a layer of width 4 without any reactivation, we can add this hanging edge using just 2 extra interactions. In both these cases the two extra operations required are saved in the next layer involving these qubits. These savings come from the fact that previously qubit 5 or 9 would have had to interact with the bus for a second time, but the hanging edge removes this requirement. If we consider creating either of these layers from a previous layer, then the hanging edges can be added, provided we keep only one qubit from the previous layer entangled with the bus during our transition. In section 9.5.3 we demonstrated that it was impossible to create these hanging edges if two qubits remained entangled to the bus between layers. If only one qubit remains entangled to the bus, we are simply beginning our layer from the top right/left hand corner and the hanging edge can be added trivially.


(a) The layer of width 2 turns the corner.

(b) We disconnect the circled qubit.

(c) We then create the tail.

Figure 9.11: This figure shows that it is possible to generate a hanging edge when moving between a layer of width 2 and width 4.

For completeness we show that we can generate a hanging edge when we are moving from a layer of width 2, to a layer of width 4. We demonstrate that this is possible in figure 9.11 where we see that by removing the circled qubit at the correct time, we can create the tail without requiring more than 2 additional operations. As it would be impossible to create this layer of width 4 without these additional reactivations, creating this hanging edge does not require additional operations. While we demonstrate how to create a hanging edge when switching between layers of width 2, and layers of width 4, this does not provide us with any advantages. This is because the layer which would 'use' this tail would be a layer of width 2, and we have previously shown that a pre-created hanging edge does not give us any benefits when transferring from a layer of width 2 to a layer of width 4

9.6 Conclusions

In this chapter we looked a layered technique for building our cluster states. We first showed that the largest layer we can create, while only attaching each qubit in the layer to the bus once, is an open path of width 4. We then demonstrate two techniques to build our cluster using only

$$\mathrm{Cl}_{\mathrm{layer}} = 3nm - 2n\,. \tag{9.13}$$

operations. These techniques used layers of width 2 and 4, or 2 and 3. Comparing this to our lower bound of,

$$\mathrm{Cl}_{\mathrm{min}} = 3nm - 2n - 2m + 4 \tag{9.14}$$

we saw that our layered method of generation required considerably more operations than the lower bound. As a result we looked at what savings we can achieve by keeping our qubits attached to the bus as we transitioned between layers. We found that even in the simplest case we could save 2 operations per transition, and if we carefully considered the path we took, we could increase our savings further. We present arguments to show when it is possible to save 4 operations in a transition, and when it is possible to save only 2. Our results are summarised in table 9.1, which we can use to work out the total saving possible for our two layered techniques of generation.

Size of layers	Hanging edge available?	Saving in transfer	Generate hanging edge?
$2 \to 3$	Yes and No	4	No
$3 \to 3$	No	2	Yes
$3 \to 3$	Yes	4	No
$2 \to 4$	Yes and No	2	Yes
$4 \to 2$	No	2	Yes
$4 \to 2$	Yes	4	No

Table 9.1: This table shows the savings that can be obtained when generating the transition between layers of the cluster. If we generate a hanging edge then it will be used as part of the transition after next.

A key point is that a saving of 4 operations prevents us from generating a hanging edge, and thus achieving a saving in the transition after next. Therefore our results are simplest for the case where our cluster is made of layers of width 3. Here we will see a pattern of two transitions where we save 4 operations, two transitions where we save 2 operations, two transitions where we saving 4 operations etc. For a large cluster the number of transitions is given by $(m-2)/2$, therefore our total saving is given by $3(m-2)/2$. This gives

$$\mathrm{Cl}_3 = 3nm - 2n - \frac{3}{2}m + 3\,. \tag{9.15}$$

We can see that this is $(m-2)/2$ operations above our lower bound.

The savings we find in the case where we use our layers of width 4 are less significant. In this case we save 2 operations for 3 out of every 4 transitions, only saving 4 operations for a quarter of our transitions. The total saving is therefore given by $5(m-2)/4$, therefore

$$\mathrm{Cl}_{2,4} = 3nm - 2n - \frac{5}{4}m + \frac{5}{2}\,. \tag{9.16}$$

This is $3(m-2)/4$ operations above our lower bound - significantly more operations than needed to create our cluster using layers of width 3.

Neither of the layered techniques we have discussed allow us to meet the lower bound. We might naïvely ask what would happen if we used entirely layers of width 2, as these would allows us to save 4 operations per transition. However other than for small cases this would significantly increase the number of transitions required, and therefore increase the total number of operations needed. We do not provide any proof that this method of generating our cluster layer by layer can never meet the lower bound, however given the results above, it seems unlikely that a suitable switching technique could be found. In the next chapter we look at some alternative techniques for cluster state generation. These include a spiral technique, which allows us to meet the lower bound in the number of operations required, and a column technique which reduces the time taken to generate the cluster by creating some of the C-Phase gates in parallel. It is worth noting however, that these layered results are useful as a dynamic method of generating cluster states using a single bus.

Chapter 10

Further improvements in cluster state generation

10.1 Introduction

In the previous chapter, we introduced the idea of using the qubus to produce a cluster state in a layered fashion. However, there are numerous problems involved with this technique. In particular we do not meet the lower bound on the number of operations required, and the time taken to generate a single layer of our cluster is relatively long. In this chapter, we look at some of these problems in detail and discuss potential solutions.

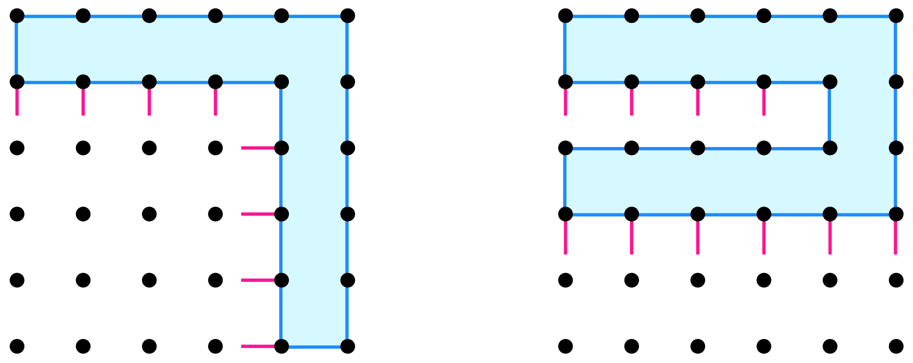
In section 10.2, we look at a technique for generating our cluster that meets the minimum bound we found in the previous chapter. In this section, we show that right-angle turns in our cluster always allow us to save the required 4 operations per corner. While this technique is more efficient, it has significant practical disadvantages, as we lose the dynamic property that we had previously. Therefore, this technique for generation is more a theoretical, rather than a practical solution to our difficulties.

A potential solution to the the problem of the large amount of time required to generate a single layer is discussed in section 10.3, where we consider splitting our cluster into columns rather than rows. These columns are generated in parallel, thus a single layer of the cluster is generated far faster. The disadvantage of such a technique is the need for a large number of separate buses to generate each column of operations. Since all these buses would be operating at the same time we would also need a highly controlled system.

Finally, in section 10.4 we consider some of the problems with fault tolerance in our cluster. In particular, we trade off the probability of an error due to decoherence, with the probability of an error on the bus to find the optimal size section to use to build our cluster. While this is a very limited discussion of the errors it gives an idea about some of the issues that we face by using our qubus to generate cluster states.

10.2 A more efficient way to generate cluster states

In section 9.5, we found that while it was occasionally possible to leave 2 qubits active at the end of each layer, we can often only leave one qubit active between layers. Our open path of width 3 can be considered to be a closed path of width 2 with tails of trailing qubits. Therefore we would hope to be able to leave 2 qubits active between 'layers' so that we never have to break our closed path. In this section we will demonstrate that this is possible if we consider using 90 degree turns between layers rather than 180 degree turns. This results in a spiral path around our cluster rather than simply combining layers of the same length.

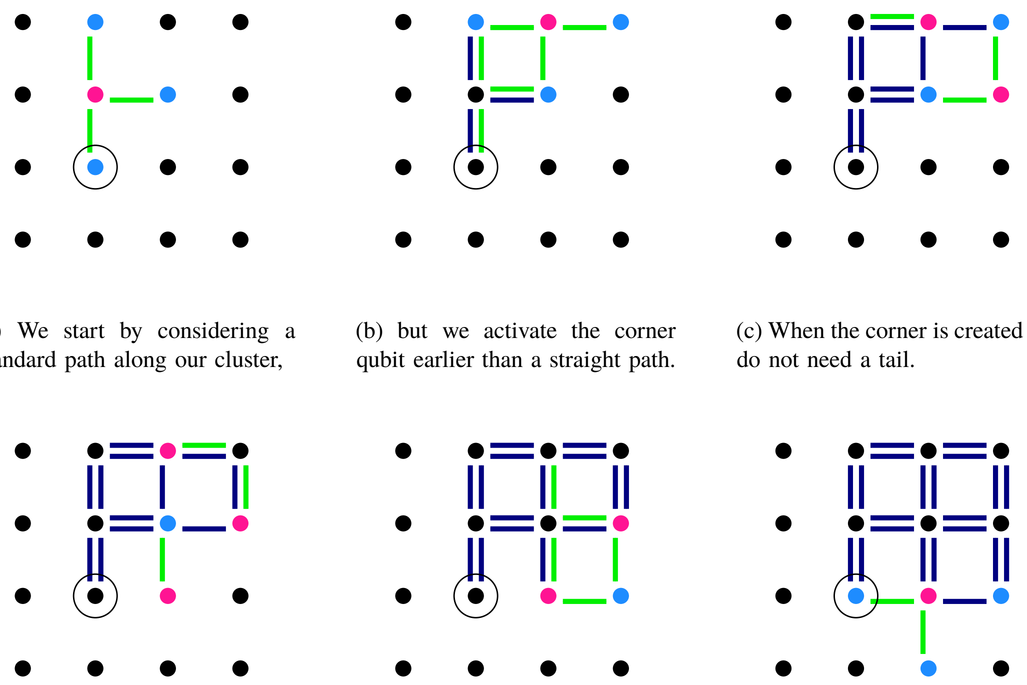


(a) A 90 degree turn (b) A 180 degree turn

Figure 10.1: This figure shows a 90 degree and a 180 turn in our path. The closed path of width 2 is shown in blue while the trailing edges are shown in pink. In the 90 degree turn the trailing edges remain on the same side of the path, while in the 180 degree turn they flip sides.

.

A spiral path containing turns of only 90 degrees should meet the lower bound we derived above. We can begin to see this if we look at what happens in the necessary turn as shown in figure 10.1. For the 90 degree turn, we can clearly see our path of width 2 (illustrated in blue) with the trailing edges shown in pink. If we imagine stretching out the turn, we can see that we have an unbroken path of width 2, with the necessary trailing edges. However if we consider our 180 degree turn, then stretching out the turn would mean that the trailing edges switched sides on our path. This suggests that it might be feasible to create the spiral path without the need for the additional operations that we needed in our layered generation technique. However, this does not provide a proof.


(a) We start by considering a standard path along our cluster,

(b) but we activate the corner qubit earlier than a straight path.

(c) When the corner is created we do not need a tail.

(d) We can remove the corner qubit,

(e) and continue our path as normal.

(f) One qubit needs to reactivated but this was expected.

Figure 10.2: Here we show it is possible to create a 90 degree corner without having to break our path of width 2. Only the circled qubit needs to interact with the bus twice. This agrees with our counting arguments.

In appendix A.2 we show a list of the operations required to generate a 10×10 spiral. Simply counting the number of displacement operators shows that only 264 operations are required. However, this is not easy to see, diagrammatically illustrating the entire spiral would take up a large amount of space and would be unnecessary since we are essentially considering the repeat of a 90 degree turn. Therefore in figure 10.2 we show that a 90 degree turn can be created while maintaining our path of width 2.

Given this we can do a simple calculation to work out the total number of operations required. For simplicity we will consider our spiral path across the cluster to start by covering the shortest length. We will choose this length to be n, while the longest length will be m. The path itself needs $2nm$ operations to generate, so we need to consider how many tails are needed for the activation. Figure 10.3 shows a 6×10 cluster with the path, and tails marked. We can see the first section of our path requires $n - 2$ tails, the second section requires $m - 4$, the third $n - 4$

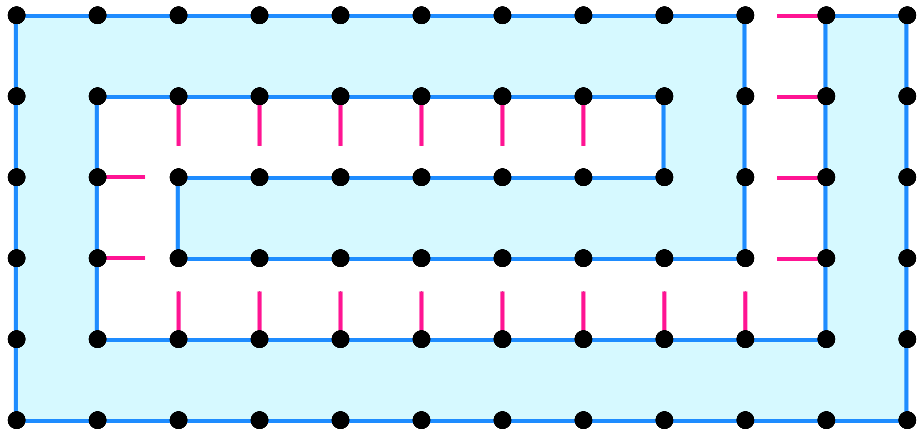


Figure 10.3: This figure has been rotated so that it fits on the page. We consider the cluster to have length, $n = 6$, and width $m = 12$. The blue path represents the closed path of width 2, while the pink lines represent the additional tails. We can see that the number of tails required for each section of the path decreases by 2 at every turn.

etc. We can use this pattern to work out the total number of tails, N_{tail} required for a n times m cluster. This is given by

$$N_{\text{tail}} = \sum_{a=1}^{\frac{1}{2}n} (n - 2a) + \sum_{a=2}^{\frac{1}{2}n} (m - 2a)\,. \tag{10.1}$$

In the alternative case where n' is the largest dimension (and also the first covered by our path), then we end up with

$$N'_{\text{tail}} = \sum_{a=1}^{\frac{1}{2}m'-1} (n' - 2a) + \sum_{a=2}^{\frac{1}{2}m'-1} (m' - 2a)\,. \tag{10.2}$$

These two expressions are equivalent and we can see that

$$N_{\text{tail}} = N'_{\text{tail}} = \frac{1}{2}nm - n - m + 2\,. \tag{10.3}$$

We have shown that we can turn a corner in a path activating only one qubit twice (see figure 10.2), therefore we can create all the tails in figure 10.3 with only two operations per tail. This means that we require $nm - 2n - 2m + 4$ operations above the standard $2nm$ to create a cluster, giving

$$\text{Cl}_s = 3nm - 2n - 2m + 4\,. \tag{10.4}$$

We therefore see, that our spiral path meets the minimum bound in the number of operations required to create a cluster.

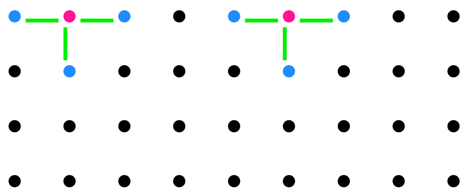
In this section, we showed that a spiral path meets the lower bound on the number of operations required, which we derived in section 9.4. However there are practical problems with the spiral path, in particular that we are no longer generating our cluster dynamically. This means that we have to generate the entire cluster before we are able to perform operations on it. Therefore, we can see that although this technique provides efficiency savings, our layered technique would make a more realistically useable cluster. However, even with our layered technique of generation we still have significant problems with the time taken to generate the cluster, and fault tolerance. We discuss these two problems in more detail in sections 10.3 and 10.4. This allows us to consider a more practical technique for building clusters.

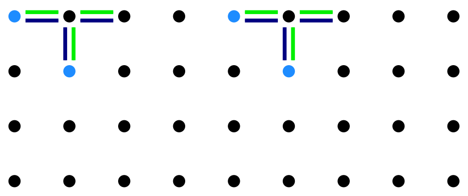
10.3 Generating our cluster state using columns

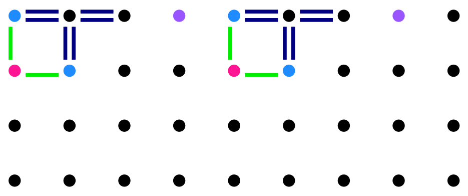
As Viv Kendon pointed out, one of the significant problems with the layered system is that if n is large, a large amount of time is required to generate each layer [150]. We will therefore experience a problem with decoherence before the layer is useable. A method to circumvent this involves generating our cluster in columns rather than in layers. Here we will assume that we have access to multiple buses, and that a separate bus is used to generate each column of operations. By creating our columns of operations simultaneously it is possible to create the entirety of each row in relatively few time steps.

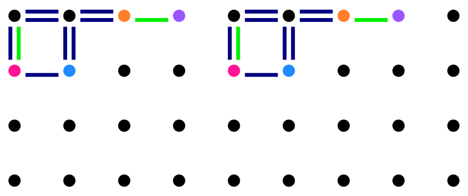
There are many possible ways to divide up the columns, however we will make the simplifying assumption that the best way to do this is to use the path of width 3 which we used previously. This system presents numerous advantages over other techniques. First and foremost, since every column segment is the same width we do not have to plan the optimal technique for weaving together our segments in great detail. If we varied the width of the segments then we would have to delay operations in our shorter segments to ensure that they are moving at the same speed as our longer segments, since otherwise we would end up creating some of our line far too early or interacting some qubits with two buses at once. We also have the advantage that each bus only interacts with the qubits in its set once. The scheme therefore is relatively simple. Finally in section 10.2, we argued that the path of width 3 (equivalent to a path of width 2 with trailing edges) was likely to be optimally efficient for generating our cluster. This suggests that such a split is also optimal.

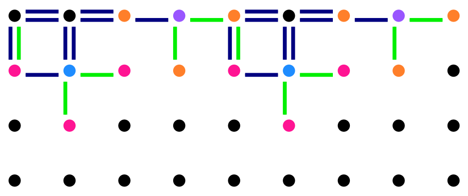
We now want to derive the total time taken to generate a cluster of width m, given the time taken to generate a suitable chunk of our cluster to perform gate operations. In section 8.3, we introduced the idea of performing operations on cluster states and showed that roughly 3 rows of the cluster need to be generated to perform a single two qubit gate. Figure 10.4 shows our cluster being grown column by column. One thing we note is that to avoid a qubit being connected to two buses at the same time we need to delay the start of every other column by

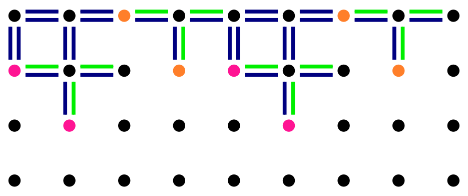

(a) The initial 4 operations required to start the first two columns.

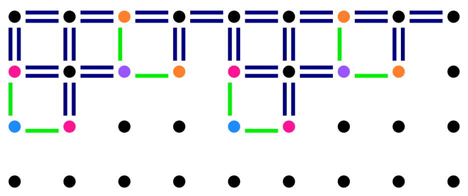

(b) Only two operations are shown in this diagram.


(c) Only one operation shown, we now begin the other columns.


(d) Again only one operation is shown.


(e) Two operations are illustrated.


(f) A further two operations.


(g) Only a single operations is shown.

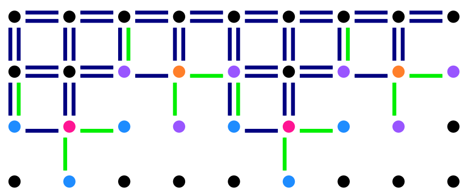

(h) Three operations are shown in this diagram.

Figure 10.4: In this diagram we show how to grow the cluster using columns across the cluster. There is a 6 operation delay between adjacent columns. For clarity every other column uses orange to represent qubits connected to the position quadrature, and purple for qubits connected to the momentum quadrature of the bus. The other columns use the same notation as before.

6 operations. For each subfigure we have used the caption to explain how many displacement operators are shown, figure 10.4(c) shows the start of the second column after 6 operations. We use the standard notation discussed in section 9.2.1 to represent the interactions of our qubits with the bus. The exception is that every other column uses orange and purple, instead of pink and blue to represent qubits which are entangled to the bus. Orange qubits are entangled to the position quadrature, and purple qubits to the momentum quadrature. This saves confusion about which qubits become entangled to which bus. A list of the displacement that correspond to this growth are shown in appendix A.3.

Other than the first row, each row of operations requires 17 operations to create, and there is a delay of 6 operations between each row with the second row starting after only 3 operations. Therefore, to create m rows of our cluster requires a number of time steps given by

$$S = 6m + 8 \tag{10.5}$$

If we are considered generating 6 lines of our cluster we would need 44 time steps, compared to the $16n - 6$ required previously. We therefore see, that building our cluster state using columns is more efficient provided $n > 3.2$.

If we have access to multiple buses then we can get further significant savings on the number of operations required to generate a single row of our cluster. This allows us to generate a useable section of the cluster far faster than before. While it would be equally possible to do something equivalent by generating our layers in parallel, this would lead to problems since a large proportion of the cluster would be ‘waiting’ for operations to be performed on it for longer than the decoherence time of the qubits. It would also generate rows of the cluster less efficiently. Using multiple buses to generate columns of our cluster increases the practicality of using an ancilla system for generating a cluster. However we have not yet discussed one of the primary difficulties in using an qubus to generate a cluster, since we have failed to look at how errors accumulate on the bus, and how this affects what it is practical to perform. We consider this in the next section.

10.4 Problems with fault tolerance in cluster state generation

Our techniques for generating a cluster state involve using a mediating ancilla system which entangles with a large proportion of the cluster before it can be cleared. While we are not considering entangling all the qubits in our cluster to the bus at the same time, we still have to consider error propagation within this system. In particular, we note that if an error occurs on the bus at any point, this error will be propagated to all interactions which occur after the error. Therefore, a single bus error could be detrimental to a large proportion of our cluster.

One solution to this would be to generate each of our C-Phase gates individually. However if we are only considering using a single bus, this would require a considerably longer time to generate our cluster. We, therefore, want to look at the trade off between the error propagation on the bus and decoherence errors that occur due to the cluster waiting to be used.

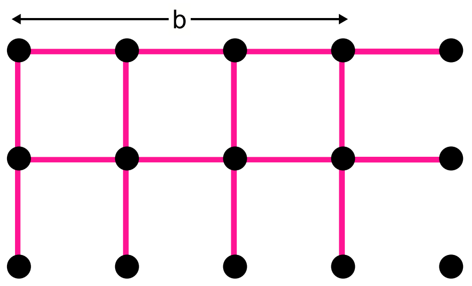


Figure 10.5: To build our cluster state without excessive error accumulation we need to use smaller sections than an entire layer. We therefore consider using bricks such as the one shown above

We will begin by considering the simplifying assumption that we have access to only a single bus system, and that this is the only way to generate entanglement between our qubits. In section 9.3 we looked at generating clusters by using layers to increase the efficiency of generating the rows of our cluster. However, as suggested by Viv Kendon, and Clare Horsman, if we are going to consider generating smaller sections of our cluster rather than an entire layer, we need to begin looking at bricks. A single brick is shown in figure 10.5, and has width 3 and length b. A single brick can be created by using $6b + 4$ operations with the bus.

To explore the optimal size for a brick, we once again call on the error model proposed by Bill Munro in our paper on robust cluster state generation [150]. This was previously mentioned in section 4.6. This gave us a total probability of de-phasing given by

$$P_d = \frac{1}{2}(1 - \exp[-N_b \gamma \tau - 4C\eta\beta^2]) \tag{10.6}$$

where N_b is the number of bus operations, γ is the de-phasing for one qubit, C is the number of gates constructed with one use of the bus, η is the loss parameter for the bus, and $\beta = \chi t$ as defined previously. We will consider the number of operations required to create only a single brick. In the case where we create a single brick without ever disentangling the bus, we require a total of $6b + 4$ operations to create $4b$ C-Phase gates. This gives

$$P_d = \frac{1}{2}(1 - \exp[-(6b + 4)\gamma\tau - 16b\eta\beta^2])\,. \tag{10.7}$$

In the case where only one C-Phase gate can be made at a time, the total number of bus operations required is given by $16b$ therefore

$$P_d = \frac{1}{2}(1 - \exp[-16b\gamma\tau - 4\eta\beta^2])\,. \tag{10.8}$$

Comparing the two results, we find that the brick based scheme produces less de-phasing provided $\eta\beta^2 \lesssim 5\gamma\tau/8$. Bill Munro provides us with some typical values for the qubus scheme, in this case $\gamma\tau = 5 \times 10^{-4}$ and $\eta = 10^{-4}$. If we consider an error threshold of $P_d = 10^{-2}$ we can obtain values for b. We find that in the scheme where we generate our brick $b = 5$. While in the scheme where we generate each C-Phase gate individually $b = 2$. Our brick method allows us to create 20 C-Phase gates, while creating operations separately only allows us to create 8. Therefore, we can see that the brick method can create more C-Phase gates with the same limit in de-phasing.

For completeness, we want to consider how many additional operations are required to generate our cluster state in a brick by brick fashion compared to generating it layer by layer. An $n \times m$ cluster contains $nm/2b$ bricks, however fewer operations are required since we do not need to consider adding the tails onto the edge bricks. This allows us to save $2(m + n)$ operations in total. We therefore need a total number of operations given by

$$\mathrm{Cl}_b = nm(6b + 4)/2b - 2(m + n) = (3 + 2/b)nm - 2(m + n)\,. \tag{10.9}$$

This is $2nm/b$ more than required for our optimal method, and $(4nm/b - m - 6)/2$, above our layered method.

Generating a large section of our cluster fast, becomes easier with the more buses that are available. For instance if we had $2n$ buses, it would be possible to generate six layers of our cluster in only 24 time steps. This would quickly give us a large operational area, with very few difficulties with error accumulation or decoherence. The problem with such a technique would be the level of control required, since $2n$ separate buses would all have to interact with the qubits at the same time. Therefore we need to trade off complexity of implementation with the need to minimise errors. One possible solution to this is to use the column growth technique we discussed in section 10.3 combined with splitting our columns into bricks. Once again we could use bricks with $b = 5$. However, this time multiple bricks would be grown simultaneously. This would allow us to generate the cluster fast using only $(n - 1)/2$ buses. If we were to use these buses to create individual operations then we would require $16m + 8(m - 1)/(n - 1) - 8$ time steps to generate m rows. If we take $8(m - 1)/(n - 1)$ to be small then using the columns is faster when $m > 1.7$

To compare our methods fully we will now consider generating a complete $n \times m$ cluster in 3 ways. The first is a brick method, the second builds each C-Phase gate individually in separate time steps, and the third uses $(n-1)/2$ buses to build each C-Phase gate individually but in parallel. We will replace N in our error model shown in equation (10.6) with S which is the total number of time steps required. In the case where we use our bricks, we require m/b sets of bricks, with $m/b-1$ requiring $6(b-2)+20$ operations and 1 requiring $6(b-2)+16$ operations. Simplifying this means that $S = m(6b+8)/b$ time steps are required, $4b$ C-Phase gates are created by a single bus before it is disentangled from the system, therefore

$$P_d = \frac{1}{2}(1 - \exp[-(6m + 8m/b)\gamma\tau - 16b\eta\beta^2])\,. \tag{10.10}$$

When we build each C-Phase gate individually and sequentially then $8nm - 4(n+m)$ time steps are required, with only 1 C-Phase gate being created by each bus before it is disentangled. Therefore

$$P_d = \frac{1}{2}(1 - \exp[-(8nm - 4n - 4m)\gamma\tau - 4\eta\beta^2])\,. \tag{10.11}$$

Finally when we create our C-Phase gates individually but using $(n-1)/2$ buses, $16m-8$ time steps are required, with only 1 C-Phase gate per bus before it is disentangled. Therefore

$$P_d = \frac{1}{2}(1 - \exp[-(16m - 8)\gamma\tau - 4\eta\beta^2])\,. \tag{10.12}$$

From inspection of equations (10.11) and (10.12), it is easy to see that our C-Phase gates being created in parallel always give less de-phasing than gates being generated entirely sequentially. We therefore want to compare our column generation with our sequential generation. For our column technique to cause less de-phasing we require

$$(6m + 8m/b)\gamma\tau + 16b\eta\beta^2 < (16m - 8)\gamma\tau + 4\eta\beta^2 \tag{10.13}$$

If we use our values from $\gamma\tau$ and η from before, then we can say that

$$40m/b + 2\pi b + 80 - 0.5\pi < 50m\,. \tag{10.14}$$

Since $m \geq b \geq 1$ we can place bounds on when this inequality is met. If $b = 1$ then we find our column generation technique gives less de-phasing when $m > 8.5$. If $b = m$ then our column generation technique gives less de-phasing when $m > 2.7$. Therefore provided that $b > 2.7$, the column generation always gives less de-phasing than generating our C-Phase gates individually, since the minimum value of m required to obtain improvements is less than b. We have therefore shown that our column generation technique is the most error resistant method of generating our cluster with $(n-1)/2$ buses.

The results above considered the error across the whole cluster, something which may be misleading as we would be using the cluster before it had been entirely generated. However we want to consider this result because considering only a single brick ignores the advantages of the multiple buses.

In this section, we have explored methods to reduce error accumulation on our cluster. While this does not make the process of building the cluster fault tolerant, it explores ways to make our generation more robust. This is important for a practical scheme. We have shown that provided $\eta\beta^2 \lesssim 5\gamma\tau/8$, our layered method results in fewer errors per brick than by generating the entire cluster using single C-Phase gates. By putting in suitable values for an experimental system, we have also shown that our column technique has less de-phasing than using the multiple buses to create single C-Phase gates. While our generation scheme needs to be worked on further in order to provide complete fault tolerance, we have demonstrated that a layered and a column technique of generating our cluster can be made robust.

10.5 Conclusions

In this chapter we introduced two alternative techniques for generating our cluster states. The first technique consisted of using a spiral path of C-Phase gates to generate our cluster, because right-angle turns provide more significant savings than the 180 degree turns. In particular, this is because during right-angle turns we do not break our path of C-Phase gates, while during 180 degree turns we do. We demonstrate that by using the spiral path it is possible to reduce the total number of operations required to

$$\mathrm{Cl}_s = 3nm - 2n - 2m + 4\,. \tag{10.15}$$

which is identical to our lower bound. This suggests that our spiral technique of generation is the most efficient available. However this ignores several issues. In particular the spiral technique of generation is not dynamic. This means the entire cluster has to be generated before we can perform operations on it. Given this cost, it is often best to use the more expensive layered technique, as in this case we could use the earlier parts of the cluster sooner.

Our second alternative technique of generation, looks at generating our cluster using columns, rather than layers. While this does not provide any saving in the number of operations required, it provides significant savings in the number of time steps required. Since the columns are being generated in parallel it is possible to reduce the total number of times steps to

$$S = 6m + 8\,. \tag{10.16}$$

In the case of large n provided $m > 3.2$, the column technique provides a saving in the number of time steps required. The disadvantage of the column technique is that a relatively large number of buses are required, and this requires a relatively high degree of control. However, in section 10.4 we show that our column technique is the fastest method of generating the cluster given the number of buses available.

Finally in section 10.4 we discussed some of the problems with fault tolerance, that arise from our technique of generating cluster states. A major difficulty with the technique we present is error accumulation. A single error on the bus will propagate throughout any parts of the cluster which interact with this bus after the error. Therefore we consider occasionally disconnecting our bus entirely from the cluster, and either refreshing it, or using a new bus. This splits our columns or layers into smaller sections, which we refer to as bricks. In section 10.4 we balance the errors due to error accumulation on the bus, with errors due to decoherence that occur due to the increased length of time our cluster takes to create if we generate each operation separately. We find that provided $\eta\beta^2 \lesssim 5\gamma\tau/8$, the scheme using bricks has less de-phasing than creating each C-Phase gate separately. By using a given set of error values ($\gamma\tau = 5 \times 10^{-4}$, $\eta = 10^{-4}$ and $P_d = 10^{-2}$), we find that we can create 20 C-Phase gates by using a brick based scheme, but only 8 by using a scheme where we create the operations individually.

If we compare our column based scheme to a scheme where we build our C-Phase gates individually using multiple buses, then we find that if we use the same error values as before, the column based schemes gives advantages provided $m > 8.5$ if $b = 1$, and $m > 2.7$ if $b = m$. While this gives a range of values, we can easily see that provided $b > 2.7$ our columns technique of generation is always advantageous, when it comes to error accumulation.

One problem with our results is that error correction procedures in cluster states, particularly those associated with the topological code, are designed to deal with individual errors rather than correlated errors. Any error on our bus has a high chance of causing a correlated error which would then be difficult to correct for. This is not unique to our system, and would also apply to the one shot and other reduced schemes for creating cluster states discussed in section 8.2.1. An extension to this work could look at possible ways to circumnavigate these difficulties, either by reducing the correlated errors or by choosing plaquettes in such a way that a correlated error on a particular area could be detected.

Our work on cluster states has discussed three main techniques of generation and looked at error accumulation on all three. Other than problems with correlated errors, we have found that provided we have access to multiple buses, a column technique of generation is both most efficient, and leads to the smallest error accumulation. It would be possible to expand this work by looking at ways to make our generation technique fault tolerant, and to use standard error correction techniques to detect and correct errors within our bricks.

Chapter 11

Conclusions

In this thesis we looked at using the qubus quantum computer for two main purposes, quantum simulation, and generating cluster states. The main focus of our work was on how to increase the efficiency of both processes. We also bounded the number of operations required, and demonstrated implementation techniques, that were within a constant number of operations to this bound. The qubus quantum computer uses a continuous variable field to generate interactions between the qubits. In particular, qubits that are entangled to position quadrature of the bus, entangle to qubits which are entangled to the momentum quadrature of the bus. By attaching multiple qubits to the bus at one time, we can reduce the number of times certain qubits have to entangle with the bus in order to generate the necessary interactions between qubits. This forms the basis of our work.

11.1 Simulating the BCS Hamiltonian

The first part of our work concentrates on quantum simulation. After introducing the basics of quantum simulation, and the BCS Hamiltonian in chapter 3, we consider how to simulate the BCS Hamiltonian on our qubus quantum computer. As with a standard quantum simulation, the BCS Hamiltonian has to be split into several non-commuting unitaries, which are performed individually, then recombined using the Trotter approximation [12]. The necessary two qubit unitaries, are local equivalent to the term

$$U_{zz} = \exp\left(-i \sum_{m<l}^{N} \frac{V_{ml}}{2} \sigma_{zm}\sigma_{zl}\right) , \tag{11.1}$$

In chapter 4 we consider various techniques for using our qubus to implement this term. We consider 3 specific scenarios. The first is the completely general case. Here we demonstrated that it is possible to implement the necessary terms in $N^2 - N + 2$ operations. This is only 2 operations above our derived lower bound of $N^2 - N$. The next scenario is the case of limited range interactions. We now introduce a new variable p which represents the range of the interactions being considered. We found that we could lower bound the number of operations required as $2Np - p^2 - p$, and we found a technique that could generate these operations using only $2Np - p^2 - p + 2$ operations. Therefore, once again we were two operations above our lower bound. Finally we considered a case where the strength of interactions between the modes of our BCS Hamiltonian are highly dependent upon each other. This would include the normal model proposed by Bardeen et al. [11] where V_{ml} is held constant, and would also include interactions which decayed exponentially with the distance between modes. In this scenario, we lower bounded the number of operations required as $4N - 4$, and showed how to meet this lower bound.

We can compare these results to a naïve qubus implementation, and with previous results obtained on an NMR computer. In the case of a naïve qubus implementation, we need $2(N^2 - N)$ operations to implement the necessary two qubit unitary. We therefore see that in all cases we get a factor of 2 improvement, and in specific cases this increases to $\mathrm{O}(N)$. To compare with results for an NMR computer, we need to consider recombining all our local unitaries. In this case we will consider using the first order Trotter approximation, as this is the level of accuracy used by Wu et al. [12]. For the nearest neighbour case, we find the total number of operations required for a single time interval is given by $9N$, while Wu et al. require $(28N^4 - 47N^3 - 4N^2 + 32N)/3$. Therefore, we see $\mathrm{O}(N^3)$ improvement over an NMR quantum computer. In the general case, the Wu et al. require $\mathrm{O}(N^5)$ operations while we need $2N^2 + 3N + 4$, so once again we see an $\mathrm{O}(N^3)$ improvement. A further improvement can be found if we consider a case where the interactions between modes were highly dependent on each other, in which case we would only need $13N - 8$ operations, to achieve an $\mathrm{O}(N^4)$ improvement over NMR. We can therefore, easily see that the qubus quantum computer is far better than NMR for generating long range interactions.

Unfortunately, to use the phase estimation algorithm, we need to implement these operations in a controlled fashion dependent upon the state of an ancilla qubit. We show that this is possible in chapter 5, where we adapt a procedure for making a one qubit unitary controlled, in order to make our N qubit unitary controlled. We therefore show that, while making our operations controlled is expensive, it can be done using only a factor of 2 increase in gates in the general case. The techniques we develop are valid only for the general case, and the limited range case. As the range of the interaction decreases, making the operations controlled becomes relatively more expensive. In the nearest-neighbour case we need a factor of 12 increase. This is because the necessary CNOT gates represent a larger proportion of our operations in the limited range case, than in the general case. Although the same number of CNOT gates is required, the number of operations required to generate our interactions decreases, as the range of the interaction decreases. While we show a method of making the operations controlled, in the case of highly dependent connections between modes, this is more expensive than the general case, so is not useful. From this chapter we can conclusively see that it is possible to make our N qubit operations controlled, without requiring too many additional operations. In particular, we demonstrate that we can still use some of the methods shown in the previous chapter to reduce the number of operations required, and to make our operations controlled. Therefore, the savings made in generating long range interactions can be computationally useful.

In chapter 6 we considered the phase-estimation algorithm, and demonstrated how we would perform this on a qubus quantum. This work concentrated on performing the quantum Fourier transform (QFT). We demonstrated that it was possible to perform the QFT on a qubus quantum computer efficiently. In particular we showed how to reduce the number of operations required from $3N^2-3N+24\lfloor N/2\rfloor$ in the naïve case, to $24\lfloor N/2\rfloor+7N-6$. This is O$(N)$ improvement. We also found that the most expensive stage of our QFT in the adapted case is the swap gates. If we considered a scenario where we measured straight after the QFT then the total number of operations required would be $6N-5$. We therefore, show significant improvements when implementing the QFT on the qubus quantum computer.

Finally, in chapter 7 we take the results from our previous chapters to work out the total number of operations required to conduct a quantum simulation. Given the bounds on precision previously considered by Wu et al. [12], and Brown et al. [13], we find that the total number of operations required for the longest run of the computer, in the nearest neighbour case is given by,

$$T_{\text{qubus}} = \frac{\pi}{\delta}(204.9N - 120)\,. \tag{11.2}$$

while the NMR computer required

$$T_{\text{NMR}} = \frac{6}{\delta}N^4 \tag{11.3}$$

Therefore, we can see once again that our qubus system gives an O(N^3) improvement over NMR. Provided we take $N \geq 6$, our qubus quantum computer requires fewer operations. Making a comparison in the general case is more difficult, as Wu et al. [12] do not give the exact number of operations required. However we see an O(N^3) improvement, with the qubus system almost certainly being more efficient provided $N \geq 7$. The qubus quantum computer should also require significantly fewer runs than the NMR computer. We can therefore see that, the qubus quantum computer has significant advantages over NMR for quantum simulation. The fact that the advantages are present for some of the fundamental primitives of quantum computing, suggest that these savings should also be applicable to other systems.

11.2 Generating cluster states

We also considered using the qubus quantum computer for a second purpose, that of generating cluster states. After introducing some previous work conducted on cluster state generation in chapter 8, we moved on to consider how to use our qubus system to build a cluster state. Previous work had shown how to do this using just $4nm$ operations. This was already a factor of 2 saving, over a naïve method, which required $8nm - 4(n+m)$. In chapter 9 we considered a layered technique for generating our cluster, which allowed us to reduce the number of operations required to

$$\mathrm{Cl}_3 = 3nm - 2n - \frac{3}{2}m + 3\,. \tag{11.4}$$

The savings came from using our bus quadratures carefully, to ensure that we created as large sections as possible in one go. We chose to consider a layered technique, because this allowed us to generate the cluster in a dynamic fashion. In this chapter, we also defined a lower bound on the number of operations required, this was given by

$$\mathrm{Cl}_{\mathrm{min}} = 3nm - 2n - 2m + 4\,. \tag{11.5}$$

We can therefore see that our layered technique does not meet this lower bound.

In chapter 10 we considered some alternative techniques for creating cluster states. The first of these was a spiral technique, which involved generating our cluster state using a path like structure, with only 90 degree turns in the path. We demonstrated that it was possible to meet the lower bound in the number of operations required to generate a cluster, using such a system. While this was optimal, it did not generate the cluster in an easily workable order. Both the spiral, and the layered technique of generation were fairly slow. We therefore considered using columns to generate our cluster. This required $N/2 - 1$ buses but generated the cluster in a number of time steps given by

$$S = 6m + 8\,. \tag{11.6}$$

Building our cluster state from columns, was more efficient than by layers provided $m > 3.2$. This seems to give us an optimal way to build cluster states, since we can create it in a deterministic fashion, very quickly. However, it ignores issues with errors.

The final section of work in chapter 10 looked at how errors on the bus trade off with decoherence errors. In particular, we looked at splitting our layers, or columns into smaller building bricks, at the cost of extra operations. We found the brick method of building, always gave less de-phasing provided, $\eta\beta^2 \lesssim 5\gamma\tau/8$. With some given error parameters ($\eta = 10^{-4}$ and $\gamma\tau = 5 \times 10^{-4}$), provided by Bill Munro, we showed that for an error of less than $P_d = 10^{-2}$, our brick scheme could generate 20 C-Phase gates, while generating operations separately could only generate 8. We also showed that given these values, our column based brick generation technique gives less de-phasing than generating operations individually. This applies even if we consider making our entire cluster before we start performing operations. One difficulty with this work, is that our qubus technique will lead to correlated errors. This is something that current error correcting protocols do not take into account.

11.3 Further work

One of the major limits of this work, is that we are not considering the physical system which we can use to implement the qubus. This means that there is a limit to the accuracy of characterising the errors. It would be useful to build upon this work by considering a physical architecture, such as super-conducting qubits, and using this to build a more realistic error model. This would pave the way for finding a physical implementation of the algorithms presented. We hope to look at reducing the dimensions of the ancilla, from our current continuous variable, to a finite system. In particular, we want to find the smallest finite system, that would allow us to generate similar results. This would allow us to explore the usefulness of the continuous nature of the qubus.

We hope to build on the work on cluster states by considering whether it would be possible to adapt current error-correction techniques, to be more suitable for this purpose. In particular, in the brick based scheme, while we are dealing with correlated errors, we do know the bounds of where these errors could be. It should therefore be possible to implement several stages of error correction, such that we detect which bricks have errors in them, and how far through the brick these errors have propagated.

A final development on this work, would be to look at other algorithms in which it would be possible to achieve savings in the number of operations required using the qubus. As we have demonstrated phase-estimation, and the quantum Fourier transform, it should be possible to use our results as part of Shor's algorithm [1]. It would also be useful to consider using the qubus to perform other primitives of quantum computing.

Appendix A

Displacement operators needed to build a cluster state

A.1 An open path of width four and unbounded length

The displacement operators listed below correspond to generating an open path of width 4 and length 5. These operations can be extended so that the length of the path is unbounded. This is discussed in more detail in section 9.2.2. In particular these displacement operators correspond to figure 9.4.

$D(\beta\sigma_{z4})$	$D(-i\beta\sigma_{z3})$	$D(-\beta\sigma_{z4})$	$D(-i\beta\sigma_{z1})$	$D(-\beta\sigma_{z2})$	$D(i\beta\sigma_{z1})$	$D(-\beta\sigma_{z7})$
$D(i\beta\sigma_{z3})$	$D(-\beta\sigma_{z5})$	$D(i\beta\sigma_{z6})$	$D(\beta\sigma_{z2})$	$D(\beta\sigma_{z5})$	$D(i\beta\sigma_{z8})$	$D(i\beta\sigma_{z11})$
$D(\beta\sigma_{z7})$	$D(-i\beta\sigma_{z8})$	$D(i\beta\sigma_{z9})$	$D(\beta\sigma_{z10})$	$D(-i\beta\sigma_{z9})$	$D(-i\beta\sigma_{z6})$	$D(\beta\sigma_{z12})$
$D(\beta\sigma_{z15})$	$D(-i\beta\sigma_{z11})$	$D(-\beta\sigma_{z12})$	$D(\beta\sigma_{z13})$	$D(-i\beta\sigma_{z14})$	$D(-\beta\sigma_{z10})$	$D(-\beta\sigma_{z13})$
$D(-i\beta\sigma_{z16})$	$D(-i\beta\sigma_{z19})$	$D(-\beta\sigma_{z15})$	$D(i\beta\sigma_{z16})$	$D(-i\beta\sigma_{z17})$	$D(-\beta\sigma_{z18})$	$D(i\beta\sigma_{z14})$
$D(i\beta\sigma_{z17})$	$D(-\beta\sigma_{z20})$	$D(i\beta\sigma_{z19})$	$D(\beta\sigma_{z18})$	$D(\beta\sigma_{z20})$		

A.2 Generating a cluster using a spiral path

We consider generating a 10×10 cluster, using a path of width two with hanging edges. This can be built using the list of operations below. Only 264 operations are required in total. The qubits in the cluster are numbered first along, then down the cluster.

$D(\beta\sigma_{z11})$	$D(-i\beta\sigma_{z1})$	$D(-i\beta\sigma_{z21})$	$D(-\beta\sigma_{z12})$	$D(-\beta\sigma_{z11})$	$D(i\beta\sigma_{z21})$	$D(-\beta\sigma_{z2})$
$D(i\beta\sigma_{z1})$	$D(-\beta\sigma_{z22})$	$D(-\beta\sigma_{z13})$	$D(i\beta\sigma_{z12})$	$D(\beta\sigma_{z22})$	$D(i\beta\sigma_{z3})$	$D(\beta\sigma_{z2})$
$D(i\beta\sigma_{z23})$	$D(i\beta\sigma_{z14})$	$D(\beta\sigma_{z13})$	$D(-\beta\sigma_{z23})$	$D(\beta\sigma_{z4})$	$D(-\beta\sigma_{z3})$	$D(\beta\sigma_{z24})$
$D(\beta\sigma_{z15})$	$D(-i\beta\sigma_{z14})$	$D(-\beta\sigma_{z24})$	$D(-i\beta\sigma_{5})$	$D(-\beta\sigma_{z4})$	$D(-i\beta\sigma_{z25})$	$D(-i\beta\sigma_{z16})$
$D(-\beta\sigma_{z15})$	$D(i\beta\sigma_{z25})$	$D(-\beta\sigma_{z6})$	$D(i\beta\sigma_{z5})$	$D(-\beta\sigma_{z26})$	$D(-\beta\sigma_{z17})$	$D(i\beta\sigma_{z16})$
$D(\beta\sigma_{z26})$	$D(i\beta\sigma_{z7})$	$D(\beta\sigma_{z6})$	$D(i\beta\sigma_{z27})$	$D(i\beta\sigma_{z18})$	$D(\beta\sigma_{z17})$	$D(-i\beta\sigma_{z27})$
$D(\beta\sigma_{z8})$	$D(-i\beta\sigma_{z7})$	$D(\beta\sigma_{z28})$	$D(\beta\sigma_{z19})$	$D(-i\beta\sigma_{z18})$	$D(-\beta\sigma_{z28})$	$D(\beta\sigma_{z10})$
$D(-i\beta_{z9})$	$D(-\beta\sigma_{z8})$	$D(-i\beta\sigma_{z20})$	$D(-\beta\sigma_{z10})$	$D(-\beta\sigma_{z29})$	$D(-\beta\sigma_{z19})$	$D(-i\beta\sigma_{z9})$
$D(-\beta\sigma_{z30})$	$D(i\beta\sigma_{z20})$	$D(-\beta\sigma_{z28})$	$D(-\beta\sigma_{z39})$	$D(i\beta\sigma_{z29})$	$D(\beta\sigma_{z28})$	$D(i\beta\sigma_{z40})$
$D(\beta\sigma_{z36})$	$D(i\beta\sigma_{z38})$	$D(i\beta\sigma_{z49})$	$D(\beta\sigma_{z39})$	$D(-i\beta\sigma_{z38})$	$D(\beta\sigma_{z50})$	$D(-i\beta\sigma_{z40})$
$D(\beta\sigma_{z48})$	$D(\beta\sigma_{z59})$	$D(-i\beta\sigma_{z49})$	$D(-\beta\sigma_{z48})$	$D(-i\beta\sigma_{z60})$	$D(-\beta\sigma_{z50})$	$D(-i\beta\sigma_{z58})$
$D(-i\beta\sigma_{z69})$	$D(-\beta\sigma_{z59})$	$D(i\beta\sigma_{z58})$	$D(-\beta\sigma_{z70})$	$D(i\beta\sigma_{z60})$	$D(-\beta\sigma_{z68})$	$D(-\beta\sigma_{z79})$
$D(i\beta\sigma_{z69})$	$D(\beta\sigma_{z68})$	$D(i\beta\sigma_{z80})$	$D(\beta\sigma_{z70})$	$D(i\beta\sigma_{z78})$	$D(i\beta\sigma_{z89})$	$D(\beta\sigma_{z79})$
$D(-i\beta\sigma_{z78})$	$D(i\beta\sigma_{z100})$	$D(\beta\sigma_{z90})$	$D(-i\beta\sigma_{z80})$	$D(\beta\sigma_{z99})$	$D(-i\beta\sigma_{z100})$	$D(\beta\sigma_{z88})$
$D(-i\beta\sigma_{z89})$	$D(-\beta\sigma_{z90})$	$D(-i\beta\sigma_{z98})$	$D(-\beta\sigma_{z99})$	$D(-i\beta\sigma_{z78})$	$D(-i\beta\sigma_{z87})$	$D(-\beta\sigma_{z88})$
$D(i\beta\sigma_{z78})$	$D(-\beta\sigma_{z97})$	$D(i\beta\sigma_{z98})$	$D(-\beta\sigma_{z77})$	$D(-\beta\sigma_{z86})$	$D(i\beta\sigma_{z87})$	$D(\beta\sigma_{z77})$
$D(i\beta\sigma_{z96})$	$D(\beta\sigma_{z97})$	$D(i\beta\sigma_{z76})$	$D(i\beta\sigma_{z85})$	$D(\beta\sigma_{z86})$	$D(-i\beta\sigma_{z76})$	$D(\beta\sigma_{z95})$
$D(-i\beta\sigma_{z96})$	$D(\beta\sigma_{z75})$	$D(\beta\sigma_{z84})$	$D(-i\beta\sigma_{z85})$	$D(-\beta\sigma_{z75})$	$D(-i\beta\sigma_{z94})$	$D(-\beta\sigma_{z95})$
$D(-i\beta\sigma_{z74})$	$D(-i\beta\sigma_{z83})$	$D(-\beta\sigma_{z84})$	$D(i\beta\sigma_{z74})$	$D(-\beta\sigma_{z93})$	$D(i\beta\sigma_{z94})$	$D(-\beta\sigma_{z73})$
$D(-\beta\sigma_{z82})$	$D(i\beta\sigma_{z83})$	$D(\beta\sigma_{z73})$	$D(-\beta\sigma_{z91})$	$D(i\beta\sigma_{z92})$	$D(\beta\sigma_{z93})$	$D(i\beta\sigma_{z81})$
$D(\beta\sigma_{z91})$	$D(i\beta\sigma_{z72})$	$D(\beta\sigma_{z82})$	$D(-i\beta\sigma_{z92})$	$D(\beta\sigma_{z71})$	$D(-i\beta\sigma_{z81})$	$D(\beta\sigma_{z73})$
$D(\beta\sigma_{z62})$	$D(-i\beta\sigma_{z72})$	$D(-\beta\sigma_{z73})$	$D(-i\beta\sigma_{z61})$	$D(-\beta\sigma_{z71})$	$D(\beta\sigma_{z63})$	$D(-i\beta\sigma_{z52})$
$D(-\beta\sigma_{z62})$	$D(i\beta\sigma_{z63})$	$D(-\beta\sigma_{z51})$	$D(i\beta\sigma_{z61})$	$D(-\beta\sigma_{z53})$	$D(-\beta\sigma_{z42})$	$D(i\beta\sigma_{z52})$
$D(\beta\sigma_{z53})$	$D(i\beta\sigma_{z41})$	$D(\beta\sigma_{z51})$	$D(i\beta\sigma_{z43})$	$D(i\beta\sigma_{z32})$	$D(\beta\sigma_{z42})$	$D(-i\beta\sigma_{z43})$
$D(i\beta\sigma_{z21})$	$D(\beta\sigma_{z31})$	$D(-i\beta\sigma_{z41})$	$D(\beta\sigma_{z22})$	$D(-i\beta\sigma_{z21})$	$D(\beta\sigma_{z33})$	$D(-i\beta\sigma_{z32})$
$D(-\beta\sigma_{z31})$	$D(-i\beta\sigma_{z23})$	$D(-\beta\sigma_{z22})$	$D(-i\beta\sigma_{z43})$	$D(-i\beta\sigma_{z34})$	$D(-\beta\sigma_{z33})$	$D(i\beta\sigma_{z43})$
$D(-\beta\sigma_{z24})$	$D(-i\beta\sigma_{z23})$	$D(-\beta\sigma_{z44})$	$D(-\beta\sigma_{z35})$	$D(i\beta\sigma_{z34})$	$D(\beta\sigma_{z44})$	$D(i\beta\sigma_{z25})$
$D(\beta\sigma_{z24})$	$D(i\beta\sigma_{z45})$	$D(i\beta\sigma_{z36})$	$D(\beta\sigma_{z35})$	$D(-i\beta\sigma_{z45})$	$D(\beta\sigma_{z26})$	$D(-i\beta\sigma_{z25})$
$D(\beta\sigma_{z46})$	$D(\beta\sigma_{z37})$	$D(-i\beta\sigma_{z36})$	$D(-\beta\sigma_{z46})$	$D(\beta\sigma_{z28})$	$D(-i\beta\sigma_{z27})$	$D(-\beta\sigma_{z26})$
$D(-i\beta\sigma_{z38})$	$D(-\beta\sigma_{z28})$	$D(-i\beta\sigma_{z47})$	$D(-\beta\sigma_{z37})$	$D(i\beta\sigma_{z27})$	$D(-\beta\sigma_{z48})$	$D(i\beta\sigma_{z38})$

$D(-\beta\sigma_{z46})$	$D(-\beta\sigma_{z57})$	$D(i\beta\sigma_{z47})$	$D(\beta\sigma_{z46})$	$D(i\beta\sigma_{z58})$	$D(\beta\sigma_{z48})$	$D(i\beta\sigma_{z56})$
$D(i\beta\sigma_{z67})$	$D(\beta\sigma_{z57})$	$D(-i\beta\sigma_{z56})$	$D(i\beta\sigma_{z78})$	$D(\beta\sigma_{z68})$	$D(-i\beta\sigma_{z58})$	$D(\beta\sigma_{z77})$
$D(-i\beta\sigma_{z78})$	$D(\beta\sigma_{z66})$	$D(-i\beta\sigma_{z67})$	$D(-\beta\sigma_{z68})$	$D(-i\beta\sigma_{z76})$	$D(-\beta\sigma_{z77})$	$D(-i\beta\sigma_{z56})$
$D(-i\beta\sigma_{z65})$	$D(-\beta\sigma_{z66})$	$D(i\beta\sigma_{z56})$	$D(-\beta\sigma_{z75})$	$D(i\beta\sigma_{z76})$	$D(-\beta\sigma_{z55})$	$D(-\beta\sigma_{z64})$
$D(i\beta\sigma_{z65})$	$D(\beta\sigma_{z55})$	$D(-\beta\sigma_{z73})$	$D(i\beta\sigma_{z74})$	$D(\beta\sigma_{z75})$	$D(i\beta\sigma_{z63})$	$D(\beta\sigma_{z73})$
$D(i\beta\sigma_{z54})$	$D(\beta\sigma_{z64})$	$D(-i\beta\sigma_{z74})$	$D(i\beta\sigma_{z43})$	$D(\beta\sigma_{z53})$	$D(-i\beta\sigma_{z63})$	$D(\beta\sigma_{z44})$
$D(-i\beta\sigma_{z43})$	$D(\beta\sigma_{z55})$	$D(-i\beta\sigma_{z54})$	$D(-\beta\sigma_{z53})$	$D(\beta\sigma_{z46})$	$D(-i\beta\sigma_{z45})$	$D(-\beta\sigma_{z44})$
$D(-i\beta\sigma_{z56})$	$D(-\beta\sigma_{z55})$	$D(-\beta\sigma_{z46})$	$D(i\beta\sigma_{z45})$	$D(i\beta\sigma_{z56})$		

A.3 Growing a cluster in columns

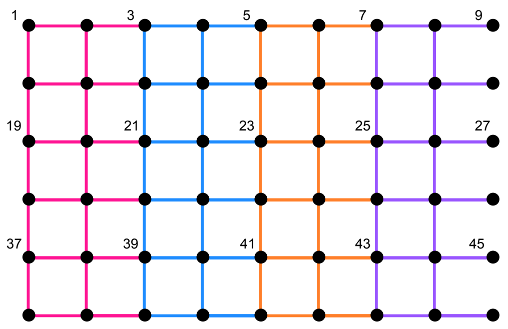


Figure A.1: A figure showing the necessary columns for growing a cluster.

The displacement operations listed below correspond to generating a 9×6 cluster in columns. The benefits of this technique are discussed in section 10.3. The pink and orange operations represent the columns created first. In these columns a bold qubit is the first bus interaction with a particular row. The blue and purple operations represent the delayed columns. The blank space at the beginning of the line is the six operation delay. In this case bold operations are removing the bus completely from a row. The qubits are numbered, first along the cluster, then down as shown in figure A.1. From inspection of these operations we can see the data that goes into creating equation (10.5).

$\boldsymbol{D(\beta\sigma_{z2})}$	$D(-i\beta\sigma_{z1})$	$D(-i\beta\sigma_{z3})$	$\boldsymbol{D(-i\beta\sigma_{z11})}$	$D(-\beta\sigma_{z2})$	$D(i\beta\sigma_{z3})$	$D(-\beta\sigma_{z10})$
$D(i\beta\sigma_{z1})$	$D(-\beta\sigma_{z12})$	$\boldsymbol{D(-\beta\sigma_{z20})}$	$D(i\beta\sigma_{z11})$	$D(\beta\sigma_{z12})$	$D(i\beta\sigma_{z19})$	$D(\beta\sigma_{z10})$
$D(i\beta\sigma_{z21})$	$\boldsymbol{D(i\beta\sigma_{z29})}$	$D(\beta\sigma_{z20})$	$D(-i\beta\sigma_{z21})$	$D(\beta\sigma_{z28})$	$D(-i\beta\sigma_{z19})$	$D(\beta\sigma_{z30})$
$\boldsymbol{D(\beta\sigma_{z38})}$	$D(-i\beta\sigma_{z29})$	$D(-\beta\sigma_{z30})$	$D(-i\beta\sigma_{z37})$	$D(-\beta\sigma_{z28})$	$D(-i\beta\sigma_{z39})$	$\boldsymbol{D(-i\beta\sigma_{z47})}$
$D(-\beta\sigma_{z38})$	$D(i\beta\sigma_{z39})$	$D(-\beta\sigma_{z46})$	$D(i\beta\sigma_{z37})$	$D(-\beta\sigma_{z48})$	$D(i\beta\sigma_{z47})$	$D(\beta\sigma_{z46})$
$D(\beta\sigma_{z48})$						

						$D(\beta\sigma_{z4})$
$D(-i\beta\sigma_{z3})$	$D(-i\beta\sigma_{z5})$	$D(-i\beta\sigma_{z13})$	$D(-\beta\sigma_{z4})$	$D(i\beta\sigma_{z5})$	$D(-\beta\sigma_{z12})$	$\boldsymbol{D(i\beta\sigma_{z3})}$
$D(-\beta\sigma_{z14})$	$D(-\beta\sigma_{z22})$	$D(i\beta\sigma_{z13})$	$D(i\beta\sigma_{z14})$	$D(i\beta\sigma_{z21})$	$\boldsymbol{D(\beta\sigma_{z12})}$	$D(i\beta\sigma_{z23})$
$D(i\beta\sigma_{z31})$	$D(\beta\sigma_{z22})$	$D(-i\beta\sigma_{z23})$	$D(\beta\sigma_{z30})$	$\boldsymbol{D(-i\beta\sigma_{z21})}$	$D(\beta\sigma_{z32})$	$D(\beta\sigma_{z40})$
$D(-i\beta\sigma_{z31})$	$D(-\beta\sigma_{z32})$	$D(-i\beta\sigma_{z39})$	$\boldsymbol{D(-\beta\sigma_{z30})}$	$D(-i\beta\sigma_{z41})$	$D(-i\beta\sigma_{z49})$	$D(-\beta\sigma_{z40})$
$D(i\beta\sigma_{z41})$	$D(-\beta\sigma_{z48})$	$\boldsymbol{D(i\beta\sigma_{z39})}$	$D(-\beta\sigma_{z50})$	$D(i\beta\sigma_{z49})$	$D(\beta\sigma_{z48})$	$D(\beta\sigma_{z50})$

$\boldsymbol{D(\beta\sigma_{z6})}$	$D(-i\beta\sigma_{z5})$	$D(-i\beta\sigma_{z7})$	$\boldsymbol{D(-i\beta\sigma_{z15})}$	$D(-\beta\sigma_{z6})$	$D(i\beta\sigma_{z7})$	$D(-\beta\sigma_{z4})$
$D(i\beta\sigma_{z5})$	$D(-\beta\sigma_{z16})$	$\boldsymbol{D(-\beta\sigma_{z24})}$	$D(i\beta\sigma_{z15})$	$D(\beta\sigma_{z16})$	$D(i\beta\sigma_{z23})$	$D(\beta\sigma_{z14})$
$D(i\beta\sigma_{z25})$	$\boldsymbol{D(i\beta\sigma_{z33})}$	$D(\beta\sigma_{z24})$	$D(-i\beta\sigma_{z25})$	$D(\beta\sigma_{z32})$	$D(-i\beta\sigma_{z23})$	$D(\beta\sigma_{z34})$
$\boldsymbol{D(\beta\sigma_{z42})}$	$D(-i\beta\sigma_{z33})$	$D(-\beta\sigma_{z34})$	$D(-i\beta\sigma_{z41})$	$D(-\beta\sigma_{z32})$	$D(-i\beta\sigma_{z43})$	$D(-i\beta\sigma_{z51})$
$D(-\beta\sigma_{z42})$	$D(i\beta\sigma_{z43})$	$D(-\beta\sigma_{z50})$	$D(i\beta\sigma_{z41})$	$D(-\beta\sigma_{z52})$	$D(i\beta\sigma_{z51})$	$D(\beta\sigma_{z50})$
$D(\beta\sigma_{z52})$						

						$D(\beta\sigma_{z8})$
$D(-i\beta\sigma_{z7})$	$D(-i\beta\sigma_{z9})$	$D(-i\beta\sigma_{z17})$	$D(-\beta\sigma_{z8})$	$D(i\beta\sigma_{z9})$	$D(-\beta\sigma_{z16})$	$\boldsymbol{D(i\beta\sigma_{z7})}$
$D(-\beta\sigma_{z18})$	$D(-\beta\sigma_{z26})$	$D(i\beta\sigma_{z17})$	$D(\beta\sigma_{z18})$	$D(i\beta\sigma_{z25})$	$\boldsymbol{D(\beta\sigma_{z16})}$	$D(i\beta\sigma_{z27})$
$D(i\beta\sigma_{z35})$	$D(\beta\sigma_{z26})$	$D(-i\beta\sigma_{z27})$	$D(\beta\sigma_{z34})$	$\boldsymbol{D(-i\beta\sigma_{z25})}$	$D(\beta\sigma_{z36})$	$D(\beta\sigma_{z44})$
$D(-i\beta\sigma_{z35})$	$D(-\beta\sigma_{z36})$	$D(-i\beta\sigma_{z43})$	$\boldsymbol{D(-\beta\sigma_{z34})}$	$D(-i\beta\sigma_{z45})$	$D(-i\beta\sigma_{z53})$	$D(-\beta\sigma_{z44})$
$D(i\beta\sigma_{z45})$	$D(-\beta\sigma_{z52})$	$\boldsymbol{D(i\beta\sigma_{z43})}$	$D(-\beta\sigma_{z54})$	$D(i\beta\sigma_{z53})$	$D(\beta\sigma_{z52})$	$D(\beta\sigma_{z54})$

Bibliography

[1] P. W. Shor. 'Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer'. *SIAM Journal on Computing*, 1997. **26** (5) 1484.

[2] T. P. Spiller, K. Nemoto, S. L. Braunstein, W. J. Munro, P. van Loock, and G. J. Milburn. 'Quantum computation by communication'. *New Journal of Physics*, 2006. **8** (2) 30.

[3] P. van Loock, W. J. Munro, K. Nemoto, T. P. Spiller, T. D. Ladd, S. L. Braunstein, and G. J. Milburn. 'Hybrid quantum computation in quantum optics'. *Phys. Rev. A*, 2008. **78** (2) 022303.

[4] R. P. Feynman. 'Simulating Physics wih Computers'. *Internat. J. Theoret. Phys.*, 1982. **21** (6/7) 467.

[5] S. Lloyd. 'Universal Quantum Simulators'. *Science*, 1996. **273** (5278) 1073.

[6] K. D. Raedt, K. Michielsen, H. D. Raedt, B. Trieu, G. Arnold, M. Richter, T. Lippert, H. Watanabe, and N. Ito. 'Massive Parallel Quantum Computer Simulator'. *Comp. Phys. Comm.*, 2007. **176** (2) 121.

[7] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon. 'Simulated Quantum Computation of Molecular Energies'. *Science*, 2005. **309** (5741) 1704.

[8] Smirnov, S. Savel'ev, L. G. Mourokh, and F. Nori. 'Modelling chemical reactions using semiconductor quantum dots'. *EPL*, 2007. **80** (6) 67008.

[9] D. S. Abrams and S. Lloyd. 'Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors'. *Phys. Rev. Lett.*, 1999. **83** (24) 5162.

[10] K. L. Brown, W. J. Munro, and V. M. Kendon. 'Using Quantum Computers for Quantum Simulation'. *Entropy*, 2010. **12** (11) 2268.

[11] J. Bardeen, L. N. Cooper, and J. R. Schrieffer. 'Theory of Superconductivity'. *Phys. Rev.*, 1957. **108** (5) 1175.

[12] L.-A. Wu, M. S. Byrd, and D. A. Lidar. 'Polynomial-Time Simulation of Pairing Models on a Quantum Computer'. *Phys. Rev. Lett.*, 2002. **89** (5) 057904.

[13] K. R. Brown, R. J. Clark, and I. L. Chuang. ‘Limitations of Quantum Simulation Examined by a Pairing Hamiltonian Using Nuclear Magnetic Resonance’. *Phys. Rev. Lett.*, 2006. **97** (5) 050504.

[14] X.-D. Yang, A. M. Wang, F. Xu, and J. Du. ‘Experimental Simulation of a Pairing Hamiltonian on an NRM Quantum Computer’. arXiv:quant-ph/0410143v2, 2004.

[15] F. Xu, A. M. Wang, X. Yang, H. You, and X. Ma. ‘Exact Numerical Solution of the BCS Pairing Problem’. arXiv:quant-ph/0407100v3, 2004.

[16] X. Yang, A. M. Wang, F. Xu, and J. Du. ‘Experimental simulation of a pairing Hamiltonian on an NMR quantum computer’. *Chem. Phys. Lett.*, 2006. **422** 20.

[17] A. M. Wang and X. Yang. ‘Quantum simulation of pairing models on an NMR quantum computer’. *Phys. Lett. A*, 2006. **352** 304.

[18] R. Raussendorf and H. J. Briegel. ‘A One-Way Quantum Computer’. *Phys. Rev. Lett.*, 2001. **86** (22) 5188.

[19] L.-M. Duan and R. Raussendorf. ‘Efficient Quantum Computation with Probabilistic Quantum Gates’. *Phys. Rev. Lett.*, 2005. **95** (8) 080503.

[20] E. Knill, R. Laflamme, and G. J. Milburn. ‘A scheme for efficient quantum computation with linear optics’. *Nature*, 2001. **409** (6816) 46.

[21] D. Browne, E. Kashefi, and S. Perdrix. ‘Computational depth complexity of measurement-based quantum computation’. In ‘Proceedings of the 5th conference on Theory of quantum computation, communication, and cryptography’, TQC’10 (Springer-Verlag, Berlin, Heidelberg), 2011 pages 35–46.

[22] R. Hoyer, P. Spalek. ‘Quantum Fan-out is powerful’. *Theory Of Computing*, 2005. **1** (5) 81.

[23] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. ‘Quantum Algorithms Revisited’. *Proc. Roy. Soc. London A*, 1998. **454** (1969) 339.

[24] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. ‘Elementary gates for quantum computation’. *Phys. Rev. A*, 1995. **52** (5) 3457.

[25] S. G. R. Louis, K. Nemoto, W. J. Munro, and T. P. Spiller. ‘The efficiencies of generating cluster states with weak non-linearities’. *New Journal of Physics*, 2007. **9** (6) 193.

[26] E. Jané, G. Vidal, W. Dür, P. Zoller, and J. I. Cirac. ‘Simuation of quantum dynamics with quantum optical systems’. *Quantum information and computation*, 2003. **2003** 15.

[27] K. Nemoto and W. J. Munro. ‘Nearly Deterministic Linear Optical Controlled-NOT Gate’. *Phys. Rev. Lett.*, 2004. **93** (25) 250502.

[28] W. J. Munro, K. Nemoto, T. P. Spiller, S. D. Barrett, P. Kok, and R. G. Beausoleil. ‘Efficient optical quantum information processing’. *Journal of Optics B*, 2005. **7** (7) S135.

[29] T. Sleator and H. Weinfurter. ‘Realizable Universal Quantum Logic Gates’. *Phys. Rev. Lett.*, 1995. **74** (20) 4087.

[30] H. F. Baker. ‘Alternants and Continuous Groups’. *Proceedings of the London Mathematical Society*, 1905. **s2-3** (1) 2447.

[31] J. E. Campbell. ‘On a Law of Combination of Operators (Second Paper)’. *Proceedings of the London Mathematical Society*, 1897. **s1-29** (1) 14.

[32] F. Hausdorff. ‘Die symbolische Exponentialformel in der Gruppentheorie’. *Leipz. Ber.*, 1906. **58** 19.

[33] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information* (Camb), 2000.

[34] Y. Yamamoto, T. D. Ladd, D. Press, S. Clark, K. Sanaka, C. Santori, D. Fattal, K. M. Fu, S. Höfling, S. Reitzenstein, and A. Forchel. ‘Optically controlled semiconductor spin qubits for quantum information processing’. *Physica Scripta*, 2009. **2009** (T137) 014010.

[35] D. A. Rodrigues, C. E. A. Jarvis, B. L. Gyorffy, T. P. Spiller, and J. F. Annett. ‘Entanglement of superconducting charge qubits by homodyne measurement’. *Journal of Physics: Condensed Matter*, 2008. **20** (7) 075211.

[36] W. Huo and G. Long. ‘Generating quantum entanglement in scalable superconducting charge qubits’. *Internat. J. Quantum Information*, 2007. **5** (6) 829.

[37] M. A. Sillanpää, J. I. Park, and R. W. Simmonds. ‘Coherent quantum state storage and transfer between two phase qubits via a resonant cavity’. *Nature*, 2007. **449** (7161) 438.

[38] J. Majer, J. M. Chow, J. M. Gambetta, J. Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, A. A. Houck, A. Wallraff, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. ‘Coupling superconducting qubits via a cavity bus’. *Nature*, 2007. **449** (7161) 443.

[39] L. DiCarlo, J. M. Chow, J. M. Gambetta, L. S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. ‘Demonstration of two-qubit algorithms with a superconducting quantum processor’. *Nature*, 2009. **460** (7252) 240.

[40] L. DiCarlo, M. D. Reed, L. Sun, B. R. Johnson, J. M. Chow, J. M. Gambetta, L. Frunzio, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf. ‘Preparation and measurement of three-qubit entanglement in a superconducting circuit’. *Nature*, 2010. **467** (7315) 574.

[41] J. Anders, D. K. L. Oi, E. Kashefi, D. E. Browne, and E. Andersson. ‘Ancilla-driven universal quantum computation’. *Phys. Rev. A*, 2010. **82** (2) 020301.

[42] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme. 'Simulating physical phenomena by quantum networks'. *Phys. Rev. A*, 2002. **65** (4) 042323.

[43] G. Ortiz, J. E. Gubernatis, and R. Laflamme. 'Quantum algorithms for fermionic simulations'. *Phys. Rev. A.*, 2001. **64** 022319.

[44] D. Aharonov and A. Ta-Shma. 'Adiabatic Quantum State Generation and Statistical Zero Knowledge'. In 'Proceedings of the 35th Annual ACM Symposium on Theory of Computing, San Diego, California, USA', (ACM Press, New York, USA), 2003 pages 20–29.

[45] J. Dziarmaga and M. M. Rams. 'Adiabatic Dynamics of an Inhomogeneous Quantum Phase Transition: The Case of $z > 1$ Dynamical Exponent'. *New Journal of Physics*, 2010. **12** 103002.

[46] S. Oh. 'Quantum computational method of finding the ground-state energy and expectation values'. *Phys. Rev. A*, 2008. **77** (1) 012326.

[47] A. N. Soklakov and R. Schack. 'Efficient State Preparation for a Register of Quantum Bits'. *Phys. Rev. A*, 2006. **73** (1) 012307.

[48] A. N. Soklakov and R. Schack. 'State Preparation Based on Grover's Algorithm in the Presence of Global Information About the State'. *Optics and Spectroscopy*, 2005. **99** (2) 211.

[49] M. Plesch and C. Brukner. 'Quantum-state preparation with universal gate decompositions'. *Phys. Rev. A*, 2011. **83** (3) 032302.

[50] D. Poulin and P. Wocjan. 'Sampling From the Thermal Quantum Gibbs State and Evaluating Partition Functions With a Quantum Computer'. *Phys. Rev. Lett.*, 2009. **103** 220502.

[51] B. M. Terhal and D. P. DiVincenzo. 'Problem of Equilibration and the Computation of Correlation Functions on a Quantum Computer'. *Phys. Rev. A*, 2000. **61** 22301.

[52] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. 'Quantum Metropolis Sampling'. arXiv:0911.3635, 2009.

[53] J. J. Vartiainen, M. Mottonen, and M. M. Salomaa. 'Efficient decomposition of quantum gates'. *Phys. Rev. Lett.*, 2004. **92** 177902.

[54] H. F. Trotter. 'On the Product of Semi-Groups of Operators'. *Proc. Am. Math. Phys.*, 1959. **10** 545.

[55] M. Suzuki. 'Improved Trotter-like Formula'. *Phys. Lett. A*, 1993. **180** (3) 232 .

[56] J. L. Dodd, M. A. Nielsen, M. J. Bremner, and R. T. Thew. 'Universal quantum computation and simulation using any entangling Hamiltonian and local unitaries'. *Phys. Rev. A*, 2002. **65** 040301.

[57] M. J. Bremner, C. M. Dawson, J. L. Dodd, A. Gilchrist, A. W. Harrow, D. Mortimer, M. A. Nielsen, and T. J. Osborne. 'Practical scheme for quantum computation with any two-qubit entangling gate'. *Phys. Rev. Lett.*, 2002. **89** 247902.

[58] M. J. Bremner, J. L. Dodd, M. A. Nielsen, and D. Bacon. 'Fungible dynamics: There are only two types of entangling multiple-qubit interactions'. *Phys. Rev. A*, 2004. **69** 012313.

[59] M. A. Nielsen, M. J. Bremner, J. L. Dodd, A. M. Childs, and C. M. Dawson. 'Universal simulation of Hamiltonian dynamics for quantum systems with finite-dimensional state spaces'. *Phys. Rev. A*, 2002. **66** 022317.

[60] M. J. Bremner, D. Bacon, and M. A. Nielsen. 'Simulating Hamiltonian dynamics using many-qudit Hamiltonians and local unitary control'. *Phys. Rev. A*, 2005. **71** 052312.

[61] P. Wocjan, D. Janzing, and T. Beth. 'Simulating arbitrary pair-interactions by a given Hamiltonian: graph-theoretical bounds on the time-complexity'. *Quantum Information & Quantum Computation*, 2002. **2** (2) 117.

[62] P. Wocjan, M. Rötteler, D. Janzing, and T. Beth. 'Simulating Hamiltonians in quantum networks: Efficient schemes and complexity bounds'. *Phys. Rev. A*, 2002. **65** (4) 042309.

[63] P. Wocjan, M. Roetteler, D. Janzing, and T. Beth. 'Universal simulation of Hamiltonians using a finite set of control operations'. *Quantum Information & Quantum Computation*, 2002. **2** 133.

[64] X. L. Deng, D. Porras, and J. I. Cirac. 'Effective spin quantum phases in systems of trapped ions'. *Phys. Rev. A*, 2005. **72** (6) 063407.

[65] A. F. Ho, M. A. Cazalilla, and T. Giamarchi. 'Quantum simulation of the Hubbard model: The attractive route'. *Phys. Rev. A*, 2009. **79** (3) 033620.

[66] S. Trotzky, L. Pollet, F. Gerbier, U. Schnorrberger, I. Bloch, N. V. Prokofév, B. Svistunov, and M. Troyer. 'Suppression of the critical temperature for superfluidity near the Mott transition'. *Nature Physics*, 2010. **6** (12) 998.

[67] D. Porras and J. I. Cirac. 'Effective Quantum Spin Systems with Trapped Ions'. *Phys. Rev. Lett.*, 2004. **92** (20) 207901.

[68] J. Cho, D. G. Angelakis, and S. Bose. 'Simulation of high-spin Heisenberg models in coupled cavities'. *Phys. Rev. A*, 2008. **78** (6) 062338.

[69] A. Kay and D. G. Angelakis. 'Reproducing spin lattice models in strongly coupled atom-cavity systems'. *Europhysics Letters*, 2008. **84** (2) 20001.

[70] Z.-X. Chen, Z.-W. Zhou, X. Zhou, X.-F. Zhou, and G.-C. Guo. 'Quantum simulation of Heisenberg spin chains with next-nearest-neighbor interactions in coupled cavities'. *Phys. Rev. A*, 2010. **81** (2) 022303.

[71] X. Peng, J. Du, and D. Suter. 'Quantum phase transition of ground-state entanglement in a Heisenberg spin chain simulated in an NMR quantum computer'. *Phys. Rev. A*, 2005. **71** (1) 012307.

[72] X. Peng, J. Zhang, J. Du, and D. Suter. 'Quantum Simulation of a System with Competing Two- and Three-Body Interactions'. *Phys. Rev. Lett.*, 2009. **103** (14) 140501.

[73] A. K. Khitrin and B. M. Fung. ‘NMR simulation of an eight-state quantum system’. *Phys. Rev. A*, 2001. **64** (3) 032306.

[74] J. Zhang, G. L. Long, W. Zhang, Z. Deng, W. Liu, and Z. Lu. ‘Simulation of Heisenberg XY interactions and realization of a perfect state transfer in spin chains using liquid nuclear magnetic resonance’. *Phys. Rev. A*, 2005. **72** (1) 012331.

[75] C. Negrevergne, R. Somma, G. Ortiz, E. Knill, and R. Laflamme. ‘Liquid-state NMR simulations of quantum many-body problems’. *Phys. Rev. A*, 2005. **71** (3) 032344.

[76] S. Wiesner. ‘Simulations of Many-Body Quantum Systems by a Quantum Computer’. arXiv:quant-ph/9603028v1, 1996.

[77] C. Zalka. ‘Simulating quantum systems on a quantum computer’. *Proc. R. Soc. Lond. A*, 1998. **454** 313.

[78] C. Zalka. ‘Efficient Simulation of Quantum Systems by Quantum Computers’. *Forschr. Phys.*, 1998. **46** 877.

[79] G. Benenti and G. Strini. ‘Quantum simulation of the single-particle Schrodinger equation’. arXiv:0709.1704v2, 2007.

[80] D. Bacon, A. M. Childs, I. L. Chuang, J. Kempe, D. W. Leung, and X. Zhou. ‘Universal simulation of Markovian quantum dynamics’. *Phys. Rev. A*, 2001. **64** (6) 062302.

[81] S. Succi and R. Benzi. ‘Lattice Boltzmann Equation for Quantum Mechanics’. *Physica D*, 1993. **69** 327.

[82] D. A. Meyer. ‘From Quantum Cellular Automata to Quantum Lattice Gases’. *J. Stat. Phys.*, 1996. **85** 551.

[83] B. M. Boghosian and W. Taylor. ‘Quantum lattice-gas model for the many-particle Schr&ouml;dinger equation in d dimensions’. *Phys. Rev. E*, 1998. **57** (1) 54.

[84] B. M. Boghosian and W. Taylor. ‘Simulating quantum mechanics on a quantum computer’. *Physica D*, 1998. **120** (1-2) 30.

[85] R. Schack. ‘Using a quantum computer to investigate quantum chaos’. *Phys. Rev. A*, 1998. **57** (3) 1634.

[86] T. A. Brun and R. Schack. ‘Realizing the quantum baker’s map on a NMR quantum computer’. *Phys. Rev. A*, 1999. **59** (4) 2649.

[87] B. Lévi and B. Georgeot. ‘Quantum computation of a complex system: The kicked Harper model’. *Phys. Rev. E*, 2004. **70** 056218.

[88] B. Georgeot. ‘Quantum computing of Poincaré recurrences and periodic orbits’. *Phys. Rev. A*, 2004. **69** (3) 032301.

[89] M. Van den Nest. ‘Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond’. arXiv:0811.0898, 2008.

[90] M. Van den Nest. 'Simulating Quantum Computers with Probabilistic Methods'. arXiv:0911.1624, 2009.

[91] J. L. O'Brien, G. J. Pryde, A. Gilchrist, D. F. V. James, N. K. Langford, T. C. Ralph, and A. G. White. 'Quantum process tomography of a controlled-not gate'. *Phys. Rev. Lett.*, 2004. **93** 080502.

[92] M. Mohseni and D. A. Lidar. 'Direct characterization of quantum dynamics'. *Phys. Rev. Lett.*, 2006. **97** (17) 170501.

[93] V. M. Kendon, K. Nemoto, and W. J. Munro. 'Quantum analogue computing'. *Phil. Trans. Roy. Soc. A*, 2010. **368** 3609.

[94] R. J. Clark, T. Lin, K. R. Brown, and I. L. Chuang. 'A two-dimensional lattice ion trap for quantum simulation'. *Journal of Applied Physics*, 2009. **105** (1) 013114.

[95] P. W. Anderson. 'Random-phase approximation in the theory of superconductivity'. *Phys. Rev.*, 1958. **112** (6) 1900.

[96] A. Mastellone, G. Falci, and R. Fazio. 'Small superconducting grain in the canonical ensemble'. *Phys. Rev. Lett.*, 1998. **80** (20) 4542.

[97] J. Dukelsky and G. Sierra. 'Density matrix renormalization group study of ultrasmall superconducting grains'. *Phys. Rev. Lett.*, 1999. **83** (1) 172.

[98] A. Volya, B. A. Brown, and V. Zelevinsky. 'Exact solution of the nuclear pairing problem'. *Phys. Lett. B*, 2001. **509** (1/2) 37.

[99] R. W. Richardson and N. Sherman. 'Exact eigenstates of the pairing-force Hamiltonian'. *Nuclear Physics*, 1964. **52** 221.

[100] S. Y. Ho, D. J. Rowe, and S. De Baerdemacker. 'Eigenstate estimation for the Bardeen-Cooper-Schrieffer (BCS) Hamiltonian'. arXiv:1011.4304, 2010.

[101] A. Soeda and M. Murao. 'On the feasibility of adding a control to an oracle'. Unpublished, 2011.

[102] D. P. DiVincenzo. 'Two-bit gates are universal for quantum computation'. *Phys. Rev. A*, 1995. **51** (2) 1015.

[103] A. Barenco, A. Ekert, K. A. Suominen, and P. Törmä. 'Approximate quantum Fourier transform and decoherence'. *Phys. Rev. A*, 1996. **54** (1) 139.

[104] R. B. Griffiths and C. S. Niu. 'Semiclassical Fourier transform for quantum computation'. *Phys. Rev. Lett.*, 1996. **76** (17) 3228.

[105] D. Browne. 'Efficient classical simulation of the semi-classical quantum Fourier transform'. *New Journal of Physics*, 2007. **9** 146.

[106] S. Lloyd and S. L. Braunstein. 'Quantum Computation over Continuous Variables'. *Phys. Rev. Lett.*, 1999. **82** (8) 1784.

[107] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. ‘Adiabatic quantum computation is equivalent to standard quantum computation’. In ‘Proceedings. 45th Annual IEEE Symposium on Foundations of Computer Science’, 2004 pages 42 – 51.

[108] T. C. Wei, I. Affleck, and R. Raussendorf. ‘Affleck-Kennedy-Lieb-Tasaki State on a Honeycomb Lattice is a Universal Quantum Computational Resource’. *Phys. Rev. Lett.*, 2011. **106** (7) 070501.

[109] M. Borhani and D. Loss. ‘Cluster states from Heisenberg interactions’. *Phys. Rev. A*, 2005. **71** (3) 034308.

[110] M. Nielsen. ‘Cluster-state quantum computation’. *Reports on Mathematical Physics*, 2006. **57** (1) 147.

[111] M. Van den Nest, K. Luttmer, W. Dür, and H. J. Briegel. ‘Graph states as ground states of many-body spin 1/2 Hamiltonians’. *Phys. Rev. A*, 2008. **77** (1) 012301.

[112] D. Gross and J. Eisert. ‘Novel schemes for Measurement-Based quantum computation’. *Phys. Rev. Lett.*, 2007. **98** (22) 220503.

[113] X. Chen, B. Zeng, Z. C. Gu, B. Yoshida, and I. L. Chuang. ‘Gapped Two-Body Hamiltonian whose unique ground state is universal for One-Way quantum computation’. *Phys. Rev. Lett.*, 2009. **102** (22) 220501.

[114] A. C. Doherty and S. D. Bartlett. ‘Identifying phases of quantum many-body systems that are universal for quantum computation’. *Phys. Rev. Lett.*, 2009. **103** (2) 020506.

[115] J. Cai, A. Miyake, W. Dür, and H. J. Briegel. ‘Universal quantum computer from a quantum magnet’. *Phys. Rev. A*, 2010. **82** (5) 052309.

[116] N. Yoran and B. Reznik. ‘Deterministic linear optics quantum computation with single photon qubits’. *Phys. Rev. Lett.*, 2003. **91** (3) 037903.

[117] M. A. Nielsen. ‘Optical quantum computation using cluster states’. *Phys. Rev. Lett.*, 2004. **93** (4) 040503.

[118] D. E. Browne and T. Rudolph. ‘Resource-Efficient Linear Optical Quantum Computation’. *Phys. Rev. Lett.*, 2005. **95** (1) 010501.

[119] D. Gross, K. Kieling, and J. Eisert. ‘Potential and limits to cluster-state quantum computing using probabilistic gates’. *Phys. Rev. A*, 2006. **74** (4) 042343.

[120] G. Gilbert, M. Hamrick, and Y. S. Weinstein. ‘Efficient construction of photonic quantum-computational clusters’. *Phys. Rev. A*, 2006. **73** (6) 064303.

[121] K. Kieling, D. Gross, and J. Eisert. ‘Cluster state preparation using gates operating at arbitrary success probabilities’. *New Journal of Physics*, 2007. **9** (6) 200.

[122] P. P. Rohde and S. D. Barrett. ‘Strategies for the preparation of large cluster states using non-deterministic gates’. *New Journal of Physics*, 2007. **9** (6) 198.

[123] M. Wilde, S. Federico, J. Dowling, and H. Lee. ‘Alternative scheme for optical cluster-state generation without number-resolving photon detectors’. *International Journal of Quantum Information*, 2007. **5** (4) 617.

[124] Y. X. Gong, X. B. Zou, T. C. Ralph, S. N. Zhu, and G. C. Guo. ‘Linear optical quantum computation with imperfect entangled photon-pair sources and inefficient non–photon-number-resolving detectors’. *Phys. Rev. A*, 2010. **81** (5) 052303.

[125] E. T. Campbell, J. Fitzsimons, S. C. Benjamin, and P. Kok. ‘Adaptive strategies for graph-state growth in the presence of monitored errors’. *Phys. Rev. A*, 2007. **75** (4) 042303.

[126] K. Kieling, T. Rudolph, and J. Eisert. ‘Percolation, renormalization, and quantum computing with nondeterministic gates’. *Phys. Rev. Lett.*, 2007. **99** (13) 130501.

[127] Y. Matsuzaki, S. C. Benjamin, and J. Fitzsimons. ‘Probabilistic growth of large entangled states with low error accumulation’. *Phys. Rev. Lett.*, 2010. **104** (5) 050501.

[128] Q. Lin and B. He. ‘Efficient generation of universal two-dimensional cluster states with hybrid systems’. *Phys. Rev. A*, 2010. **82** (2) 022331.

[129] S. J. Devitt, A. D. Greentree, R. Ionicioiu, J. L. O’Brien, W. J. Munro, and L. C. L. Hollenberg. ‘Photonic module: An on-demand resource for photonic entanglement’. *Phys. Rev. A*, 2007. **76** (5) 052312.

[130] W. J. Ionicioiu, R. Munro. ‘Constructing 2D and 3D cluster states with photonic modules’. *International Journal of Quantum Information*, 2010. **8** (1-2) 149.

[131] S. B. Zheng. ‘Generation of cluster states in ion-trap systems’. *Phys. Rev. A*, 2006. **73** (6) 065802.

[132] H. Wang, C. P. Yang, and F. Nori. ‘Robust and scalable optical one-way quantum computation’. *Phys. Rev. A*, 2010. **81** (5) 052332.

[133] C.-Y. Lu, X.-Q. Zhou, O. Guhne, W.-B. Gao, J. Zhang, Z.-S. Yuan, A. Goebel, T. Yang, and J.-W. Pan. ‘Experimental entanglement of six photons in graph states’. *Nature Physics*, 2007. **3** (2) 91.

[134] N. Paul, J. Menon, S. Karumanchi, S. Muralidharan, and P. Panigrahi. ‘Quantum tasks using six qubit cluster states’. *Quantum Information Processing*, 2010. pages 1–14.

[135] R. Raussendorf, D. E. Browne, and H. J. Briegel. ‘Measurement-based quantum computation on cluster states’. *Phys. Rev. A*, 2003. **68** (2) 022312.

[136] A. Trisetyarso and V. M. R. ‘Circuit design for a measurement-based quantum carry-lookahead adder’. *Internat. J. Quantum Information*, 2010. **8** (5) 843.

[137] H. J. Briegel and R. Raussendorf. 'Persistent entanglement in arrays of interacting particles'. *Phys. Rev. Lett.*, 2001. **86** (5) 910.

[138] D. Gross, S. T. Flammia, and J. Eisert. 'Most quantum states are too entangled to be useful as computational resources'. *Phys. Rev. Lett.*, 2009. **102** (19) 190501.

[139] J. Anders and D. E. Browne. 'Computational Power of Correlations'. *Phys. Rev. Lett.*, 2009. **102** (5) 050502. *Preprint:* arXiv:0805.1002.

[140] A. Broadbent and E. Kashefi. 'Parallelizing quantum circuits'. *Theoretical Computer Science*, 2009. **410** (26) 2489.

[141] R. Raussendorf, J. Harrington, and K. Goyal. 'Topological fault-tolerance in cluster state quantum computation'. *New Journal of Physics*, 2007. **9** (6) 199.

[142] A. Kitaev. 'Fault-tolerant quantum computation by anyons'. *Annals of Physics*, 2003. **303** (1) 2.

[143] S. Bravyi and A. Kitaev. 'Universal quantum computation with ideal Clifford gates and noisy ancillas'. *Phys. Rev. A*, 2005. **71** (2) 022316.

[144] S. J. Devitt, K. Nemoto, and W. J. Munro. 'The idiots guide to quantum error correction', 2009. ArXiv:0905.2794.

[145] H. J. Briegel, D. E. Browne, W. Dur, R. Raussendorf, and M. Van den Nest. 'Measurement-based quantum computation'. *Nature Physics*, 2009. pages 19–26.

[146] J. I. Cirac, P. Zoller, H. J. Kimble, and H. Mabuchi. 'Quantum state transfer and entanglement distribution among distant nodes in a quantum network'. *Phys. Rev. Lett.*, 1997. **78** (16) 3221.

[147] S. Mancini and S. Bose. 'Engineering an interaction and entanglement between distant atoms'. *Phys. Rev. A*, 2004. **70** (2) 022307.

[148] L. M. Duan, B. Wang, and H. J. Kimble. 'Robust quantum gates on neutral atoms with cavity-assisted photon scattering'. *Phys. Rev. A*, 2005. **72** (3) 032333.

[149] S. D. Barrett and P. Kok. 'Efficient high-fidelity quantum computation using matter qubits and linear optics'. *Phys. Rev. A*, 2005. **71** (6) 060310.

[150] C. Horsman, K. L. Brown, W. J. Munro, and V. M. Kendon. 'Reduce, reuse, recycle for robust cluster-state generation'. *Phys. Rev. A*, 2011. **83** (4) 042327.