Quantum Circuits: From a Network to a One-Way Model - A Tutorial Version

Larisse D. Voufo, Gerardo Ortiz, Amr Sabry

***Abstract*—We present elements of quantum circuits translations from the (standard) network or circuit model to the one-way one. We present a translation scheme, give an account of currently existing tools to apply the scheme, and propose an extension of those tools into a complete translation calculus. We analyse the set of difficulties incurred from such work, and show an engendered opening to new sets of problems and ideas. Among others, this paper introduces the notions of graphical concatenation, graph state reduction (GSR) and graph state extension (GSE) passes, as well as an algorithm for running the (extended) measurement calculus with acceptable efficiency.**

CONTENTS

(Dated: November 21, 2007)

.
Larisse Voufo:
Institute for Scientific Interchange Foundation, 10133 Torino, Italy
Department of Computer Science, Indiana University, Bloomington, IN 47405
email: voufo@isi.it
.
Gerardo Ortiz:
Department of Physics, Indiana University, Bloomington, IN 47405
email: ortizg@indiana.edu
.
Amr Sabry
Department of Computer Science, Indiana University, Bloomington, IN 47405
email: sabry@cs.indiana.edu

Note 1: This is a more review-based version of the paper at http://www.cs.indiana.edu/~lvoufo/ow_add_preproc.pdf, which has been submitted to the pre-proceedings of the DCM-QPL'08 workshop (http://www.comlab.ox.ac.uk/people/bob.coecke/DCM_QPL_08.html). An abstract version of it (http://www.cs.indiana.edu/~lvoufo/ow_add_abstract.pdf) has already been accepted for a presentation at the same workshop.

Here, the audience is assumed to be an interested newcomer to the field of quantum computation, not so familiar with their computational models – especially the circuit model versus the one-way one. So, we gather sufficiently detailed information to set grounds for work towards an efficient, effective, and complete translation calculus.

I. INTRODUCTION

Quantum computation has been a remarkable source of interest since the introduction of its idea in 1982, by Paul

Benioff and Richard Feynman (separately), during the trend towards miniaturization in micro-circuitry. The idea was to use the properties of quantum particles or systems, such as superposition (quantum parallelism) and entanglement, to perform computation, improve computer efficiency, and henceforth more effectively keep up with Moore's Law[1]. It got even more interesting circa 1985, when it challenged presently used security protocols and provided algorithmic solutions. Those solutions, as many other emerging quantum algorithms at the time, showed impressive improvements over their equivalent classical algorithms (e.g. Shor's, Grover's, and Deutch's algorithms). For physical realizations reasons, and based on the demands of multiple experimental physical devices, several computational models have had to be established and include the standard network (or circuit), the one-way, the adiabatic, and the topological models.

More particularly, as interests and needs for experiments in one-way quantum computation rise, the needs for a systematic way to translate already existing circuits — especially those that derive from the standard model — into circuits obeying the one-way model, rise equivalently. Some work has already been done by Schlingemann [10] that systematically translates circuits from a one-way to a circuit model, but that does not address the question of optimization or efficiency of the translation. Other works implicitly provide methods of circuit optimizations in a particular model, through an account of possible circuits transformations and equivalence classes [9], [17]. However, to the best of our knowledge, there does not exist a straightforward and systematic way to transform circuits from one particular model to another, and surely not in a way that would produce the best translated circuit possible. This poses some cumbersome limits on the possibilities of implementing and testing simulations in and across given models.

In this paper, we address the problem of translating a circuit from a standard network to a one-way model. We give an account of elements that such a task would normally involve, reviewing current works and deriving more effective components such as a graphical concatenation for circuits in the one-way model. Furthermore, we approach the question of translations' efficiency and propose an extension of the measurement calculus [8] to include optimization passes. This gives rise to a new set of interesting problems and future works which we will also elaborate on. However, let us first look at some preliminary information on our two models of interest.

Throughout this paper, we will assume a basic familiarity with notions of quantum mechanics and linear algebra such as Hilbert spaces, their approach to quantum mechanics, and the Bloch Sphere. Following, Section I-A1 reviews some preliminary information on the fundamental differences between the circuit and the one-way models, as well as with the relation of one-way realizations to graphs. The more familiar user can thus skip it and move on with the paper.

[1]Which states that the number of transistors per processor that can be inexpensively placed on an integrated circuit is increasingly exponential, doubling approximately every two years (often every 18 months).

A. Circuit versus One-Way Models

1) Brief recalls from quantum mechanics: For the purpose of this work, it is necessary to recall some basic but important notions of quantum mechanics: entanglement and measurement. Considering Dirac's notation, a quantum state in a composite system is known to be *entangled* if it cannot be expressed in terms of a tensor product of elementary states of the subsystems that form the composite one. In other words, a measurement in one of those subsystems affects the measurement outcome in the remaining ones. In this case, it is said that the measurements are *correlated*. It is *maximally entangled* — e.g. EPR states such as $\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$ — if a partial trace of its density matrix representation, over any of the subsystems in question, returns a multiple of the identity matrix. In the case of the EPR states, the partial trace returns $\frac{1}{2}$ of the identity matrix. Also in this case, although the measurement of a first qubit will result in either one of the states $|0\rangle$ or $|1\rangle$ with probability $\frac{1}{2}$ (or probability amplitude $\frac{1}{\sqrt{2}}$), that measurement also determines with probability 1 whether that of the other qubit, afterwards, will return the state $|0\rangle$ or $|1\rangle$. For example, in the state $\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$, the measurements of both qubits are correlated so that that of the second qubit will always result in whatever the result of the first measurement was.

In parallel, performing a projective measurement in an observable O is equivalent to projecting it into the space spanned by the orthonormal eigenstates of O, which is viewed as an irreversible process. In fact, using Dirac's notation, that observable can be decomposed into a sum of orthonormal projections as $O=\sum_i \lambda_i P_i$, where $P_i=|\phi_i\rangle\langle\phi_i|$, with $|\phi_i\rangle$ representing each such eigenstate of O, and λ_i is the corresponding eigenvalue. Thus, the measurement of an arbitrary state $|\psi\rangle$ in O will collapse into one of the eigenstates $|\phi_i\rangle$ with probability amplitude $\langle\phi_i|\psi\rangle$ or classical probability $\langle\psi|P_i|\psi\rangle$, causing the normalized state of the system to become $\frac{P_i|\psi\rangle}{\langle\psi|P_i|\psi\rangle}$.

In a two-level quantum system (i.e. with 1-qubit states), the observable O can be expressed as a linear combination of the identity $I=\begin{bmatrix}1&0\\0&1\end{bmatrix}$ and Pauli matrices $\sigma_x=\begin{bmatrix}0&1\\1&0\end{bmatrix}$, $\sigma_y=\begin{bmatrix}0&-i\\i&0\end{bmatrix}$, and $\sigma_z=\begin{bmatrix}1&0\\0&-1\end{bmatrix}$ — which, from now on, we will refer to as X, Y, and Z for simplicity. When the parameter associated with the identity component is null, those associated with the X, Y, and Z components define the measurement direction, which is normalized in the Bloch sphere.

Let $\vec{r}$ be such normalized measurement direction corresponding to O so that $O=\vec{r}\cdot\vec{\sigma}$, where $\vec{\sigma}$ is the Pauli vector

$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$. It turns out that there is only two possible eigenvalues[2] for O: +1 and -1. We will refer to anything associated to those values as "*positive*" (for +1), and "*negative*" (for -1). Thus, O only has two eigenstates: a positive and a negative one. Let us call them $|\phi_+\rangle$ and $|\phi_-\rangle$ respectively. Hence, O can be expressed in terms of the positive and negative projections, $P_+ = |\phi_+\rangle\langle\phi_+|$ and $P_- = |\phi_-\rangle\langle\phi_-|$, as $O = (+1)P_+ + (-1)P_-$. Therefore, measuring $|\psi\rangle$ in O will "*collapse*" into $|\phi_+\rangle$ with probability amplitude $\langle\phi_+|\psi\rangle$, and into $|\phi_-\rangle$ with probability amplitude $\langle\phi_-|\psi\rangle$.

Additionally, the projections can be expressed in terms of $\vec{r}$ and $\vec{\sigma}$ as $P_\pm = \frac{1+(-1)^s\vec{r}\cdot\vec{\sigma}}{2}$, with s equal 0 for P_+ and 1 for P_-.

2) The Standard Network (or Circuit) Model: In quantum computation, the circuit model is the most general and widely used model of computation, especially given its similarity to the classical circuits' representation (in circuit design). It acts both as a framework for theoretical investigations and as a guide for experiments [2]. Furthermore, it consists of a succession of unitary operations on a register of qubits followed by a final measurement of those qubits.

Here, information is primarily encoded and measured in the Z observable, with the eigenstates $|0\rangle$ and $|1\rangle$ forming the computational basis. Measuring in any other observable is analogous to first changing the computational basis of the information into that containing the eigenstates of the observable in question, and then measuring in Z[3].

For example, for a measurement in X, this change of computational basis is performed through an application of the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ which turns the states $|0\rangle$ and $|1\rangle$ into $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ respectively[4]. In fact, applying a Hadamard on an arbitrary state $|a\rangle = \alpha_a|0\rangle + \beta_a|1\rangle$ (with $\alpha_a^2 + \beta_a^2 = 1$) produces the state

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_a \\ \beta_a \end{bmatrix} = (\frac{1}{\sqrt{2}}(\alpha_a + \beta_a)|0\rangle + \frac{1}{\sqrt{2}}(\alpha_a - \beta_a)|1\rangle) \tag{1}$$

So, measuring $|+\rangle$ ($|-\rangle$) in X will return $|0\rangle$ ($|1\rangle$) with probability 1, as $\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}) = \frac{1}{\sqrt{2}}$ and $\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}) = 0$.

Additionally, the entanglement of a pure composite state is introduced by controlled operations (or rotations) — which we will precede with the letter "C" for simplicity, from now on. Among others, a controlled X (NOT) or CX (CNOT) will take a 2-qubits state, containing a "*control*" and a "*target*" qubits, and flip the target if the control is $|1\rangle$. Alternatively, a CZ will add a phase to the target qubit if both qubits are $|1\rangle$. In fact, let the state $|a\rangle$ above represent the control, and let a second qubit $|b\rangle = \alpha_b|0\rangle + \beta_b|1\rangle$ be the target. A CZ from $|a\rangle$ to $|b\rangle$ will take in the pure state $|ab\rangle = (\alpha_a|0\rangle + \beta_a|1\rangle) \otimes (\alpha_b|0\rangle + \beta_b|1\rangle) = \alpha_a\alpha_b|00\rangle + \alpha_a\beta_b|01\rangle + \beta_a\alpha_b|10\rangle + \beta_a\beta_b|11\rangle$, and return the state

$$\begin{aligned} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_a\alpha_b \\ \alpha_a\beta_b \\ \beta_a\alpha_b \\ \beta_a\beta_b \end{bmatrix} & \\ = \alpha_a\alpha_b|00\rangle + \alpha_a\beta_b|01\rangle + \beta_a\alpha_b|10\rangle - \beta_a\beta_b|11\rangle & \\ = \alpha_a|0\rangle \otimes (\alpha_b|0\rangle + \beta_b|1\rangle) + \beta_a|1\rangle \otimes (\alpha_b|0\rangle - \beta_b|1\rangle) & \end{aligned} \tag{2}$$

which is entangled[5]. Also, notice that the CZ from $|b\rangle$ to $|a\rangle$ will return the same result. For simplicity, we will represent an application of an operator O on some states $a, b, c, \ldots$ with $O(a, b, c, \ldots)$. So, for the previous statement, we will write $CZ(|a\rangle, |b\rangle) = CZ(|b\rangle, |a\rangle)$.

Here, all operations are unitary, thus reversible, so any introduced entanglement, can also be removed by simply reversing operations. Moreover, any unitary can be decomposed into CNOT and rotation operations, which renders this model universal.

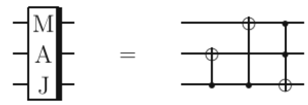


Figure 1. The in-place majority gate MAJ . [Picture gotten from [1]]

Experiments under this model get challenging when the coherence of quantum states has to be maintained over a long period of time. Among others, consider the circuit in Figure 1 which represents the realization of a majority (MAJ) gate in the circuit model [1]. The MAJ gate consists of, first, the preparation of a 3-qubits register. Then a CNOT is applied to the last two (2) qubits. Then, another CNOT is applied to the first and last qubits. Later, a CCNOT (or Toffoli) is applied to the resulting 3 qubits. Finally, the register is measured at once.

Here, it is easy to imagine that under this model, experiments consist of the implementation of sequences of highly controlled particle (qubit) interactions. This poses a serious challenge to the scalability of quantum systems to a large number of qubits, essential for practical applications such as Shor's algorithm [3]. As a result, circa 2000, Robert Raussendorf and Hans J. Briegel had the idea of formulating quantum computation in a different way, and "provide a new framework for experiment and new theoretical insights" [2], by introducing the one-way quantum computation.

3) The One-Way Model: Unlike the standard model of computation (I-A2), the one-way model consists entirely of

[2] In fact, when $\vec{r}$ is not normalized, the eigenvalues are $+\|r\|$ and $-\|r\|$.

[3] In reality, in this process, we are looking at the probability amplitudes of the state collapsing into the current computational eigenbases.

[4] Notice that these are the eigenstates of X.

[5] $CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$; $|ab\rangle = \begin{bmatrix} \alpha_a\alpha_b \\ \alpha_a\beta_b \\ \beta_a\alpha_b \\ \beta_a\beta_b \end{bmatrix}$

a series of one-qubit (projective) measurements (and single-qubit rotations) on a highly entangled state (cluster or graph state) [2], [3], [5], which drives the computation forward irreversibly. Here, measurements are performed, and results are encoded, in any arbitrary angle of the Bloch sphere. By definition, the entanglement between any two-qubits consists of CZ operations (although it can typically be done with any controlled operation) — which are generally created via a next-neighbor Ising-like interaction — and results are encoded in the X-, Y-, or Z-eigenbasis, dictating how to interpret readout measurements. While measurements in the Z-eigenbasis effectively remove qubits from a given graph state, those in the X- and Y- ones serve as "wires", propagating information forward. Other arbitrary ones realize arbitrary rotations, and usually depend on results of previous measurements [5].

For instance, simply consider the state (from Equation 2)

$$\alpha_a|0\rangle \otimes (\alpha_b|0\rangle + \beta_b|1\rangle) + \beta_a|1\rangle \otimes (\alpha_b|0\rangle - \beta_b|1\rangle)$$

resulting from the CZ on $|ab\rangle$ above. After measurement of the first qubit (corresponding to $|a\rangle$) in the Z observable, we obtain the state $\alpha_b|0\rangle + \beta_b|1\rangle = |b\rangle$ in the second qubit. This state is encoded in the Z-eigenbasis so that a previous outcome of $|0\rangle$ sets it directly to $|b\rangle$ and an outcome of $|1\rangle$ requires applying a Z to the result before obtaining $|b\rangle$ as well. In this process, we then notice that we have successfully removed the state $|a\rangle$ from the original state $|ab\rangle$. This concept can apply to any size of entangled state that we start the process from. We come back to this later on in Section V on translations optimizations.

Alternatively, if we had measured the first qubit in any other observable, especially either X or Y, the result in the second qubit would have been different. Among others, the information contained in qubit $|a\rangle$ would have been propagated through the second qubit in some other interesting way. As a matter of fact, consider starting with the product state of $|a\rangle$ and an extra ancilla $|+\rangle$:

$$|a+\rangle = (\alpha_a|0\rangle + \beta_a|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

After the entanglement $CZ(|a\rangle, |+\rangle)$ with $|a\rangle$ as the control qubit, we obtain the mixed state

$$\begin{aligned} \alpha_a|0\rangle \otimes Id(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) + \beta_a|1\rangle \otimes Z(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \\ = \alpha_a|0+\rangle + \beta_a|1-\rangle \end{aligned} \quad (3)$$

Now, let's measure what has become of $|a\rangle$ in the X-observable. Recall that this corresponds to applying, on the new state, the projection $P_{\pm} = \frac{1+(-1)^s X}{2}$ — a process which would cause an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to be viewed as $\langle +|\psi\rangle|0\rangle + \langle -|\psi\rangle|1\rangle$ after the change of basis (through a Hadamard's application) and before the Z-measurement. This said, the state in Equation 3 of the system will become

$$\begin{aligned} &\alpha_a(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)|+\rangle + \beta_a(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)|-\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle(\alpha_a|+\rangle + \beta_a|-\rangle) + \frac{1}{\sqrt{2}}|1\rangle(\alpha_a|+\rangle - \beta_a|-\rangle) \\ &= \frac{1}{\sqrt{2}}|0\rangle(\frac{1}{\sqrt{2}}(\alpha_a + \beta_a)|0\rangle + \frac{1}{\sqrt{2}}(\alpha_a - \beta_a)|1\rangle) \\ &\quad + \frac{1}{\sqrt{2}}|1\rangle(\frac{1}{\sqrt{2}}(\alpha_a - \beta_a)|0\rangle + \frac{1}{\sqrt{2}}(\alpha_a + \beta_a)|1\rangle) \end{aligned} \quad (4)$$

Measuring 4 into the Z-observable will project the first qubit (originally $|a\rangle$) into either $|0\rangle$ (positive projection) or $|1\rangle$ (negative projection) each with probability amplitude $\frac{1}{\sqrt{2}}$. If the projection is positive, then the resulting state of the second qubit (ancilla) will be

$$\frac{1}{\sqrt{2}}(\alpha_a + \beta_a)|0\rangle + \frac{1}{\sqrt{2}}(\alpha_a - \beta_a)|1\rangle) \quad (5)$$

If however the projection is negative, then the state will be

$$\frac{1}{\sqrt{2}}(\alpha_a - \beta_a)|0\rangle + \frac{1}{\sqrt{2}}(\alpha_a + \beta_a)|1\rangle) \quad (6)$$

instead. But, we see that after applying X on the later, we obtain the same result as in Equation 5, which is nothing but the result of applying a Hadamard operation on the original input $|a\rangle$ as in Equation 1.

We have therefore obtained a realization of the Hadamard operation based on the one-way model. We also realize that a realization of a CZ operation comes for free with the initial entanglement operations. Therefore, as per the circuit model, this model is universal since an arbitrary rotation and a two-qubits interaction (e.g. any controlled rotation) can be implemented under it. This constitutes a sufficient condition for the formation of universal sets (Cf. II-A and II-B bellow).

Moreover, the one-way model has been shown to be more effective against decoherence than the network model and, due to the generation of more commuting operators, to provide an even better computational complexity.

From the description above, it is intuitive to derive the following general procedure for the realization of circuits in a one-way fashion.

a) General procedure: This consists of three main steps. First, a product state of all the input qubits and extra (preferably) $|+\rangle$ ancillae, forming the set of qubits Q, must be prepared, and then entangled together accordingly. The third step of the procedure simply consists of performing a series of appropriate measurements, such that measuring the output qubits produces the expected results, after some classical post-processing operations. The later operations consist of Pauli-corrections that depend on the outcome of the previous measurements results, accounting for the randomness of those measurements.

To reduce ambiguities in the appellations "*input*" and "*output*", we refer to non-ancillae input qubits to one-way preparations as "*logical input*" qubits. We also refer to the qubits of graph states that carry the final information, after all measurements have been performed, as "*logical output*" qubits.

For example, for the CZ example above, we started from two logical inputs $|a\rangle$ and $|b\rangle$, which will become the logical outputs after the entanglement of two qubits. For the Hadamard, we start with one logical input $|a\rangle$ and an ancilla $|+\rangle$. After entanglement and measurement of the first qubit, the information in $|a\rangle$ is "wired" into the ancillae which now becomes the logical output, containing the result of $H(|a\rangle)$ encoded in the X-eigenstates.

b) To more complex gate realizations: Indeed, it is possible to combine a number of those realizations in order to produce that of more complex gates. Among others, by noticing that $I = H \cdot H$, we can imagine a process where the one-way realization of the Hadamard gate described above is repeated on the logical output of a previous similar realization. This way, the logical output of the first realization becomes the logical input of the second, and the logical output of the second is that of the entire identity operation. Similarly the logical input of the first Hadamard is that of the entire operation as well. Thus, the entire process starts with one logical input $|a\rangle$ and two ancillae $|+\rangle$. Then, $|a\rangle$ (qubit 1) is entangled with one of the ancillae (qubit 2), which in turn is entangled with the remaining ancilla (qubit 3), which will then become the logical output. Then, as expected, qubits 1 and 2 are measured in X. The information in $|a\rangle$ is thus propagated to qubit 3 so that, first, if the measurement of qubit 2 resulted in a negative projection, then we must apply X on qubit 3. Second, if the measurement of qubit 1 was likewise, then we must apply Z to what has become of qubit 3. In the end, qubit 3 is an exact replica of $|a\rangle$. For more clarity, we can re-express the relationship between the I and H operations as $I(1) = H(2) \cdot H(1)$, emphasizing the qubits numbering and each operation's logical input.

A similar reasoning can apply to the realization of a CNOT by noticing that $CX(1,2) = H(3) \cdot CZ(1,3) \cdot H(2)$. So, a product state of a control qubit $|a\rangle$ (qubit 1), a target qubit $|b\rangle$ (qubit 2), and two ancillae $|+\rangle$ (qubits 3 and 4) is prepared. Then qubits 1, 2, and 4 are entangled with 3, and qubits 2 and 3 are measured in X. In the end, the result of $CNOT(|a\rangle, |b\rangle)$ is encoded in qubits 1 and 4 such that we need to apply X on qubit 4 if the measurement outcome of qubit 3 was 1, and then Z on qubits 1 and 4 if that of qubit 2 was likewise.

How this concatenation scheme works, and how to efficiently derive even more complex gate realizations to meet each one of our particular needs, is exactly the point of this work. We analyse currently existing methods and derive a graphical concatenation method that relies on graph and cluster states representations.

4) Graph and Cluster states: The examples we have seen so far indicate the fact that, in one-way quantum computation, it is common to relate the (highly entangled) initial states to graphs. So, given the definition of a one-way model (I-A3),

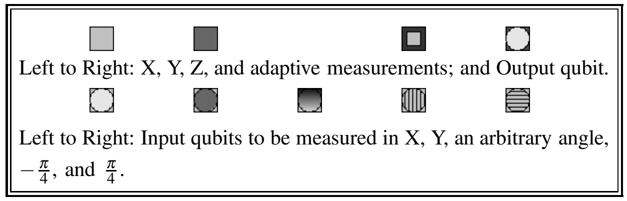


Figure 2. Graph state representation legend.

it is easy to associate such a state to a graph where each vertex corresponds to each input qubit, and the two-qubit interactions form the edges. Such a general state is called a *graph state*. When the associated graph constitutes an n-dimensional square grid, then the same state, which is now a special case of graph states, is called a *cluster state*. Here, it is important to mention that it is also common practice to refer to graph states as those graph states that are not cluster states. Nevertheless, the extra flexibility of graph states makes them more prone to having fewer qubits. However, cluster states are more physically realizable, especially in their quantum optics applications. In this paper, we will be analysing translations into both graph and cluster states.

a) Graph legend: To easily recognize how close a graph state is to a cluster one, we will borrow Raussendorf *et al.*'s 2-dimensional lattice grid design, where qubits with measurements in X and Y are, respectively, represented with light and dark gray colored squares. Adaptive measurements, i.e. those that strictly depend on previous one, are represented with a thick bordered square. Logical input and output qubits are represented with a circle inside a square [5]. We extend the design to include representations of logical input states that will be measured in observables different from X and Y. In relation to graph states, we add entanglement "wires" (simple lines) for cases where maintaining the rectangular grid representation is not possible. These "wires" will constitute the characteristic indication that the represented state is a graph state and not a cluster one. Finally, for step-by-step illustrations of concatenations, we also use arrows to represent which logical output qubit becomes which logical input one. Figure 2 presents the complete legend for qubits used throughout this paper.

B. A general translation scheme

We now turn to our original and main task of translating a given circuit from the circuit to the one-way model. From the earlier examples of the identity and CNOT gates (I-A3), we can already imagine a translation scheme that would be based on an analysis of quantum mechanical properties of given operators. In order to find the one-way realization of a given operation, if we can gather up a set of elementary realizations that we already know of, such as that of the Hadamard and CZ operations, and if we can decompose the operation in question into a sequence of applications of those elementary operations, then we can use a building block approach to derive the final realization, as we did with the examples. Now, looking at standard-model circuits such as the MAJ one in Figure 1, we notice that the realization already presents a clear sequential listing of elementary operations that could be composed into the operation represented by the circuit. So, the MAJ operation could be re-expressed as $MAJ(a,b,c) = CCNOT(a,b,c) \cdot CNOT(c,a) \cdot CNOT(c,b)$. Each one of these elementary operations can be grouped together or expanded as needed. For example, we could expand CNOT and Toffoli gates, decomposing them further into compositions of the CZ and H gates, or we could group the first two CNOT applications together in a single elementary operation.

In any event, the first task of the translation would be to determine the elementary subdivisions of a major circuit, and those of the subdivisions, and so on (in a tree-like recursive fashion), until we reach the smallest subdivisions, the one-way realizations of which we refer to as *universal sets*. Then, we will need to combine each sequence of subdivisions as necessary based on various combination methods that we call *concatenations*. This will produce a first realization that will need to be improved in some way, either by extension or reduction of the resulting graph state. We call this step *optimization*. The different classes of special cases for the final realization will describe particular needs that we refer to as *realization needs*, and which includes any variation from graph states to cluster states.

However, there is another way to translate circuits that relies solely on analysing the possible input and outputs of a given circuit, and that henceforth bypasses completely any reference to elementary decomposition (or the sequencing) imposed by the circuit model. This method is called the *phase map decomposition*. In fact, in the previous method, we could perform some of the optimization earlier in the scheme by replacing some "decomposition plus combination" steps with the phase-map decomposition. Indeed, if we apply this at the highest level of the recursive "tree" (obtained from the decomposition of the original circuit), then that would be the end of the story. However, if we use this at any intermediate step of the "tree", then we would still need to cleverly choose a method of concatenation for the higher levels. Thus, this method can in fact be considered to be a generator of additional universal sets that one can use in the first step of the translation scheme.

An important question this raises is then that of determining which alternative, or combination of alternatives, better meets the computational time, effectiveness, and classification needs of the final realization. This paper focuses on presenting those alternatives as well as new interesting problems that they suggest.

b) Related work: Raussendorf *et al.* have given a detailed account of one-way (or measurement based) quantum computation on cluster states that provides a first universal set for clusters as well as a concatenation method through by-product operators [5]. Alternatively, Danos *et al.* have defined a more robust and parsimonious universal set [7], as well as a more standardized method of concatenation called the measurement calculus [8]. Additionally, they have also defined the phase-map decomposition [6]. Several others, such as Hein *et al.* and D. Schlingemann, have studied graph state properties and their classifications into some equivalence classes that could be very helpful in the process of graph state optimization [10], [17].

This work gives a condensed account of those universal sets and concatenation methods. It analyses their respective limitations and proposes and alternative concatenation scheme that is based on an analysis of the graphical representations of given one-way realizations: A graphical concatenation. In the process, it presents an alternative and more descriptive way to define one-way patterns and introduces the concept of concatenation and standardization (C&S) algorithm, and extends the measurement calculus to include more standardization passes, as well as optimization passes derived from current studies. Further, it presents an example of practical interest with the translation of Cuccaro *et al.*'s quantum ripple carry addition circuit [1] and analyses the effects of the translation on the optimizations performed in the circuit model. As a result, this will open the possibilities of implementing a complete translation calculus, hence giving rise to a new thinking process and range of interesting problems.

One will find that, in this process of circuits translation, choosing the right set of unitaries to start from could constitute an essential factor for the efficiency of the final one-way realization with respect to one's particular realization needs. Along that, sides a correspondingly favorable approach of gates combinations (or concatenation) that the final circuit will normally be built out of. Thus, it goes without saying that different combinations of universal set definitions and concatenation methods will result in one-way realizations that are closer to fulfilling some particular realization needs than others.

c) Outline: This said, the purpose of this paper is four fold. First, we will present the various possible choices (or generations) of universal sets and methods of circuits combinations. In Section II, we will review currently existing universal sets (or sets of elementary unitaries) that are appropriate for clusters and graph states separately. Section III will study all methods of concatenations, going from the by-product operator (III-A), to the measurement calculus (III-B), to the graphical concatenation (III-D). In Section IV, we will completely by-pass references to the network model and generate additional or more effective elementary unitaries through the phase map decomposition. Second, in Section V, we will define some optimization passes and introduce the notions of Graph State Reduction (GSR) and Graph State Extension (GSE) transformations, after a thorough definition of some classes of realization needs. Third, we will present the example of practical interest in Section VI,

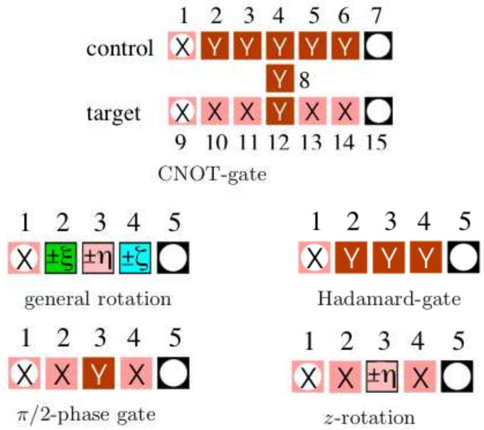
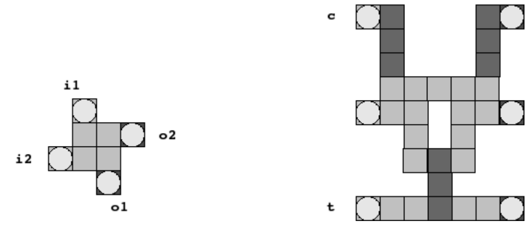


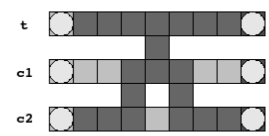
Figure 3. Cluster state realizations of elementary quantum gates on a one-way quantum computer. [Pictures from [5]]



(a) Phase and Swap

(b) Distant CNOT with 3 logical inputs qubits

Figure 4. Phase and Swap gate (a) and distant CNOT (b) built from it, according to [5].



(a) 2 CNOTs in the network model

(b) 2 CNOTs in the one-way model

Figure 5. Realization of the 2 CNOTs gates in (a) in a one-way cluster state (b), according to [5].

and analyse the effects of such systematic translations on algorithms efficiency in Section VII. We will review elements of algorithm analysis in both models of interest, and show that optimizations in the network model need not always be transmitted to its one-way equivalent. Fourth, Section VIII will introduce the possible complete translation calculus and present the set of engendered problems, as well as possible solutions.

II. Choosing a Universal Set

This section gathers currently defined universal sets with respect to some particular realization needs.

A. A Universal Set for Cluster States

Raussendorf *et al.*'s work provides a set of elementary gates realized in clusters (See Figure 3) [5]. Given the context of their work, it appears safe to concur that the combination of these realizations is more likely to result in a cluster state representation than in a graph state one. This is although it may at times result in a graph state, unless we apply a few transformation tricks. (The practical example and optimizations in the upcoming Sections V and VI will give us a good idea of what such tricks entails.) Better yet, translations are more likely to produce cluster states if we start from these elementary gates, rather than from some other set of unitary realizations in graph states. Notice that this set is indeed universal since any gate can be built from a combination of any of the elementary gates it contains.

In addition to these elementary gates, Raussendorf *et al.* also provide a number of optimized realizations of combined gates, which would be useful in the concatenation (plus optimization) step of the translation. For details on the optimized realizations beyond the scope of this paper, we refer the reader to their origin ([5]). Among those realizations are a Phase and Swap gate, a combination of two CNOTs, a distant CNOT (or CNOT between two qubits that are not direct neighbors of each other in the network model), and a CCPhase, which we represent in Figures 4, 5, and 6.

B. A Parsimonious and Robust Universal Set

Stepping out of cluster state realizations a bit, Danos *et al.* realize that all quantum universal sets in the one-way model can in fact be reduced to containing only two main operations: a CZ and a unitary J defined as

$$J(\alpha) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{i\alpha} \\ 1 & -e^{i\alpha} \end{pmatrix}$$

Here, $J(\alpha)$ generalizes all unitaries over $\mathbb{C}^2$ and can be viewed as an operator that arbitrarily changes bases into the x-y plane of the Bloch sphere, such that $J(0)$ is nothing but a Hadamard operation [7]. Again, we refer the user to their work ([7]) for more details and a list of definitions of common operators — such as the Pauli operators and rotations — in terms of those operations. One thing to keep in mind though is that $J(\alpha)$ and CZ have simple one-way realizations that use only two qubits, as illustrated in Figure 7. Thus, their combinations will usually result in realizations with relatively few auxiliary qubits.

In fact, any unitary U can be expressed in terms of those two operations as

$$U = e^{i\alpha} J(0) J(\beta) J(\gamma) J(\delta) \tag{7}$$

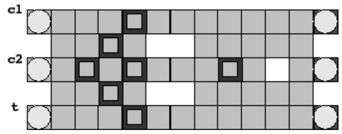


Figure 6. A Toffoli phase gate, according to [5].

(a) CZ gate (b) J unitary (c) Hadamard = J(0)

Figure 7. Robust and Parsimonious realizations of elementary quantum gates.

and a corresponding 14-qubits controlled unitary (CU) as:

$$CU_{12} = J_1^0 J_1^{\alpha+\frac{\beta+\pi+\delta}{2}} J_2^0 J_2^{\beta+\pi} \\ J_2^{-\frac{\gamma}{2}} J_2^{-\frac{\pi}{2}} J_2^0 \wedge Z_{12} J_2^{\frac{\pi}{2}} J_2^{\frac{\gamma}{2}} J_2^{\frac{-\pi-\delta-\beta}{2}} J_2^0 \wedge Z_{12} J_2^{\frac{-\beta-\delta-\pi}{2}} \quad (8)$$

where the angles that J acts upon have been superscripted, while the indexes of qubits acted on have been subscripted. CZ is also now represented as $\wedge Z$. This representation will be very useful in the understanding of the measurement calculus in Section III-B and related [7].

It turns out that graphs resulting from combinations of these elementary gates are more likely to represent graph states than cluster states. Indeed, the realization of CU produces the graph in Figure 9a. From it's definition, we can derive the remaining graphs of Figure 9 as the C$\frac{\pi}{2}$-Phase and C$(-\frac{\pi}{2})$-Phase gates realizations. A full description of the derivations is provided in Section III-B.

III. Concatenating subcircuits

Once one has identified a universal set for the construction of a larger circuit, the next step is to use it in a building block approach as introduced in the translation scheme we defined earlier. Here, we give an account of currently existing concatenation methods and derive a way to safely and systematically perform first hand concatenations, using subcircuits' graphical representations. Particularly important in the understanding of this section and further are the concepts of stabilizer formalism and Clifford groups in quantum computation.

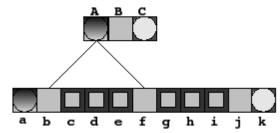
d) The Stabilizer formalism: The stabilizer formalism is a framework in which states are described and characterized in terms of operators under which they are invariant. Its main application is in quantum error-correcting codes, and compact description of multiparty entangled states [4]. We say that an operator K *stabilizes* a subspace S when, for all states $|\psi\rangle \in S$,

$$K|\psi\rangle = |\psi\rangle$$

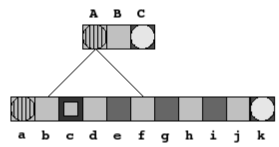
(i.e. $|\psi\rangle$ is an eigenstate of K with associated eigenvalue +1). Here, the set of all possible K's form a group whose generator is called the stabilizer generator.

(a) (b)

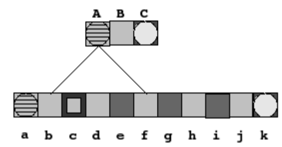
Figure 8. CNOT as defined in [3], [8](a), and Identify gate(b).



(a) General Controlled Unitary



(b) Controlled (Pi/2)-Phase



(c) Controlled (-Pi/2)-Phase

Figure 9. One-way realisations of a general controlled unitary (a) and special controlled phase gates derived from [7], [8].

This is directly related to the set of eigenvalue equations that uniquely defines a given graph state. Here, the stabilizer generator is called *a correlation operator and* has the form

$$K_a = X_a \bigotimes_{b \in nbgh(a)} Z_b$$

for each qubit a forming the state, and where $nbgh(a)$ represents the set of all neighbors of a in the associated graph. The set of all such K_a's, for a particular graph state, then forms the following set of eigenvalue equations which completely specifies[6] it:

$$K_a|\psi\rangle = (-1)^{k_a}|\psi\rangle \quad (9)$$

where all k_a's form a set a binary parameters (0 or 1) specifying the state[7]. Raussendorf *et al.*'s work provides detailed information on this ([5]).

e) Clifford groups: Clifford group operations refer to unitary operations that map Pauli group operators to the Pauli group under conjugation[2]. More formally, by definition, an operation C is in the Clifford group if it satisfies the condition

$$C\mathscr{P}_i C^\dagger = \mathscr{P}_j$$

where $\mathscr{P}_i$ and $\mathscr{P}_j$ are Pauli operators. The group is generated by the CZ, Hadamard, and $\frac{\pi}{2}$-Phase operations.

[6]Notice that this does not mean that the eigenvalue equations in question are the only ones satisfied by the state. In fact, new sets of those can be derived from composing several K_a's by addition, or (scalar) multiplication.

[7]Remember the parameter s in the definition of the two0level system's projection operators in Section I-A?

A. The by-product approach

1) The by-product operator: The original way of explaining the functioning of one-way realizations is the by-product approach, through a study of the given graph state ($|\psi\rangle$) 's correlation operators as in Equation 9. This is explained in greater detail in Raussendorf *et al.*'s work where the logical inputs are designed to be measured in the X-observable [5]. First, we lay out all the state's eigenvalue equations, and define the set $\mathscr{M}$ of one-qubit measurements needed, excluding measurements on logical inputs[8]. Then, we derive those equations that are compatible with $\mathscr{M}$, and henceforth deduce a new set of eigenvalue equations, where all the measurements in $\mathscr{M}$ have been applied:

$$\begin{cases} X_{in,i}\,(U\,X_{[i]}\,U^{\dagger})_{out}\,|\psi\rangle &= (-1)^{\lambda_{x,i}}\,|\psi\rangle \\ Z_{in,i}\,(U\,Z_{[i]}\,U^{\dagger})_{out}\,|\psi\rangle &= (-1)^{\lambda_{z,i}}\,|\psi\rangle \end{cases} \tag{10}$$

with $\lambda_{x,i}$ and $\lambda_{z,i}$ equal to 0 or 1 and $1 \leq i \leq n$, and where n is the number of qubits in $|\psi\rangle$ while the subscripts *in* (*out*) specify actions on sets of all logical input (output) qubits. This set of equations clearly defines the state in terms of actions on its logical inputs ($|\psi_{in}\rangle$) and outputs ($|\psi_{out}\rangle$) , and on the unitary function U that it realizes — such that $|\psi_{out}\rangle = U\,|\psi_{in}\rangle$. In addition, the set of outputs is defined such that each qubit corresponds to an input qubit whose information is propagated into it — as in the input and output states of a register in the network model — hence the extra i index, with $[i]$ specifying the logical output qubit that corresponds to the logical input qubit i.

In deducing Equation 10, recall from the examples in Section I-A3 that, for a graph state $|\phi\rangle$, measuring it's i-th qubit in an observable $O_i = \vec{r_i}\cdot\vec{\sigma}$ is equivalent to applying the projection operator $P_i = \frac{1+(-1)^{s_i}O_i}{2}$, where s_i represents the measurement outcome. In addition, if $P_i = \frac{1+(-1)^{s_i}O_i}{2}$ and O_i commute, then $P_i\,O_i = (-1)^{s_i}\,P_i$. Hence, given an eigenvalue equation

$$Z_1\,X_2\,Z_3\,|\phi\rangle \;=\; |\phi\rangle \tag{11}$$

where operations are subscripted with the qubit index they act on (and the correlation operator is assumed to be so that the corresponding k_a parameter has value 0), a measurement of qubit 2 transforms the equation into

$$Z_1\,(P_2\,X_2)\,Z_3\,|\phi\rangle \;=\; P_2\,|\phi\rangle \tag{12}$$

(by application of P_2 on both sides of the equation) and, since P_2 commutes with X_2, Equation 12 becomes

$$Z_1\,Z_3\,|\psi\rangle \;=\; (-1)^{s_2}\,|\psi\rangle \tag{13}$$

after replacement of $P_2\,|\phi\rangle$ with $|\psi\rangle$.

Now, once we have the eigenvalue equations in the form of Equation 10, we can simply derive the operator

$$U_p' \;=\; \bigotimes_{i=1}^{n}\left(X_{[i]}\right)^{\lambda_{z,i}}\left(Z_{[i]}\right)^{s_i+\lambda_{x,i}}$$

where i belongs to the set of all logical input qubits, such that

$$|\psi_{out}\rangle \;=\; U\,U_p'\,|\psi_{in}\rangle \tag{14}$$

The next and final step consists of reformulating Equation 14, by propagating the Pauli operations in U_p' forward, so that we now have

$$|\psi_{out}\rangle \;=\; U_p\,U\,|\psi_{in}\rangle \tag{15}$$

In the end, we have a realization of the unitary U that accounts for the randomness of previous measurements' results through extra Pauli corrections defined by U_p.

For simplicity, we will write all measurement-outcome-dependant operator $(O_i)^s$ as O_i^s.

a) Example: As illustration, consider again the CZ, Hadamard (H), and identity (Id) gates we looked at earlier in the example of Section I-A3. Again, the CZ will simply result in a form of Equation 15 with $U = CZ$, and $U_p = Id$, after entanglement and no measurement. For the Hadamard gate, we have the eigenvalue equations

$$\begin{cases} X_1\,Z_2\,|\phi\rangle &= |\phi\rangle \\ Z_1\,X_2\,|\phi\rangle &= |\phi\rangle \end{cases} \tag{16}$$

Then, there is no measurement other than that on the logical input qubit. However, Equation 16 can be rewritten as

$$\begin{cases} X_1\,H_2\,X_2\,H_2\,|\phi\rangle &= |\phi\rangle \\ Z_1\,H_2\,Z_2\,H_2\,|\phi\rangle &= |\phi\rangle \end{cases} \tag{17}$$

which is in the form of Equation 10. We then get a realization of the Hadamard unitary with U_p' operator $Z_2^{s_1}$. After propagation of the the by-product operator, through the relation $HZ = XH$, we get the unitary $X_2^{s_1}H$. Hence, a Hadamard transformation of the logical input will be obtained after applying a previous-measurement-dependant X correction on the logical output, as described in the earlier example.

For the Identity gate, we now have three eigenvalue equations:

$$\begin{cases} X_1\,Z_2 & |\phi\rangle \;=\; |\phi\rangle \\ Z_1\,X_2\,Z_3 & |\phi\rangle \;=\; |\phi\rangle \\ \quad Z_2\,X_3 & |\phi\rangle \;=\; |\phi\rangle \end{cases} \tag{18}$$

Here, qubit 2 will need to be measured in X. So, we modify the set of Equations 18 by applying the projection $P_2 = \frac{1+(-1)^{s_2}X_2}{2}$ on each side of the second equation. The result is then the same as Equation 13. Meanwhile, the first and last equations of the set 18 are combined to become

$$X_1\,X_3\,|\psi\rangle \;=\; |\psi\rangle \tag{19}$$

after applying P_2 on each side. Now, Equations 13 and 19 are in the form of Equation 10 with Identity as the unitary and the corresponding U_p' operator as $X_3^{s_2}Z_3^{s_1}$. After propagation, U_p' becomes $U_p = X_3^{s_2}Z_3^{s_1}$, which is still the same as U_p'. Again, this agrees with the earlier example where the previous-measurement-dependant conjugate transpose of U_p has to be applied to the final state of the logical outputs in

[8]Measurements on logical outputs are not accounted for as there is none by definition.

other to complete the meaning of the realization. Again, this reasoning can be applied to any other realization for proof of its effectiveness.

2) Concatenating the subcircuits: In this section, we are mainly concerned with combining multiple subcircuits — each realizing unitaries $U_{pi} \cdot U_i$ — into a larger circuit realizing unitary $U_p \cdot U$, where $U = \prod_i U_i$ for all subcircuit i included in the combination. We know that proving that the combination works for two subcircuits is sufficient for a general proof of the functioning of this building-block approach. In this sense, Raussendorf *et al.* demonstrate that combining two subcircuits realizing unitaries $U_{p1} \cdot U_1$ and $U_{p2} \cdot U_2$, with measurements through applications of P_1 and P_2 respectively, is totally valid as it results in a final unitary $U_p \cdot U$, where $U = U_2 \cdot U_1$ [9] [5]. First, recall that the realization of a unitary $U_p \cdot U$ is obtained from applying a series of projections onto an entangled initial state as $|\phi_{out}\rangle = P \cdot E \cdot |\phi_{in}\rangle$ with $|\phi_{in}\rangle$ representing the tensor product of the logical inputs and ancillae, E the entanglement, $P = \bigotimes_i \frac{1+(-1)^{s_i} O_i}{2}$ the applied projection operators during measurement (for all i in the set of all measured qubits), and $|\phi_{out}\rangle$ the tensor product of the logical outputs and the measured qubits. Now, the sequence of realizations of U_1 and U_2, in the combined circuit results in

$$|\phi_{out}\rangle = P_2 \cdot E_2 \cdot P_1 \cdot E_1 \cdot |\phi_{in}\rangle \tag{20}$$

However, for the concatenation to be valid, we would like $|\phi_{out}\rangle$ to be expressed as

$$|\phi_{out}\rangle = P \cdot E \cdot |\phi_{in}\rangle$$

Here, the first thing to consider is that the entanglement operations in the second subcircuit commutes with the measurements in the first one, as both sets of operations act on different sets of qubits. So, all qubits in the combined circuit can be safely entangled prior to the combined measurement step. This modifies Equation 20 as

$$|\phi_{out}\rangle = P_2 \cdot P_1 \cdot E_2 \cdot E_1 \cdot |\phi_{in}\rangle \ = \ P \cdot E \cdot |\phi_{in}\rangle \tag{21}$$

since $P_2 \cdot P_1 = \bigotimes_{i1} \frac{1+(-1)^{s_{i1}} O_{i1}}{2} \cdot \bigotimes_{i2} \frac{1+(-1)^{s_{i2}} O_{i2}}{2} = \bigotimes_i \frac{1+(-1)^{s_i} O_i}{2} = P$.

However, this will result in the realization of the unitary $U_{p2} \cdot U_2 \cdot U_{p1} \cdot U_1$, while we want it in the form $U_p \cdot U$. The trick here is then to again propagate the by-product operator U_{p1} forward across the unitary U_2. Raussendorf *et al.* provide a list of propagation relations to start with for the CNOT, arbitrary rotation, Hadamard and $\frac{\pi}{2}$-phase operations [5]. We will also present a more systematic propagation in the next section (III-B).

Meanwhile, we realize that while the propagation relations maintain the by-product operators in the Pauli group, a unitary operation U_i (that the by-product operators are propagated across) remains unchanged when it is in the Clifford group. In other words, if U_i was given a parameter α, then $U_i(\alpha)$ remains the same after propagation. However, if U_i is not in the Clifford group, for example if it is a general rotation, then $U_i(\alpha)$ will become $U_i(\alpha')$ after propagation, where α' is some function of α, so that the input to the final U can be modified accordingly. For example, a general rotation U_{rot} will be modified as per the following relations [5]:

$$\begin{cases} U_{rot}(\alpha,\beta,\zeta)X &= X\,U_{rot}(\alpha,-\beta,\zeta) \\ U_{rot}(\alpha,\beta,\zeta)Z &= Z\,U_{rot}(-\alpha,\beta,-\zeta) \end{cases} \tag{22}$$

a) Example: Once again, let's take the Identity and the CNOT realizations used in the earlier example of Section I-A3, and demonstrate how this building block approach will result in the same realization. In the network model, an Identity gate can be realized from a composition of two Hadamard gates, where the input of the second application is the output of the first one. Hence, we have[10]

$$X_3^{s_2} H_{[1]} \cdot X_3^{s_1} H_{[1]} \ = \ X_3^{s_2} Z_3^{s_1} H_{[1]} H_{[1]} \ = \ X_3^{s_2} Z_3^{s_1} Id_{[1]}$$

Similarly for the CNOT gate, we notice that it can be decomposed as $CNot(c,t) = H(t) \cdot CZ(c,t) \cdot H(t)$. Hence, given that $CZ_{c,t} \cdot X_t = X_t.Z_c \cdot CZ_{c,t}$, we have

$$\begin{aligned} X_4^{s_3} H_{[2]} \cdot CZ_{[1],[2]} \cdot X_4^{s_2} H_{[2]} &= X_4^{s_3} H_{[2]} X_4^{s_2} Z_1^{s_2} CZ_{[1],[2]} H_{[2]} \\ &= X_4^{s_3} Z_4^{s_2} Z_1^{s_2} H_{[2]} CZ_{[1],[2]} H_{[2]} \\ &= X_4^{s_3} Z_4^{s_2} Z_1^{s_2} CNot_{[1],[2]} \end{aligned}$$

We come back to this again later in Section III-D where we present a formula to blindly derive graphic representations of larger circuits based on those of their subdivisions, leaving the computation of the resulting by-product operators up to some future systematic work.

B. The Measurement Calculus approach

In the measurement calculus approach, it is important to first notice some limitations of the original by-product operator approach. Among others, the study of the correlation operators as described in the previous section (III-A) requires that measurements in the logical input qubits of a given graph state be only in the X observable. Moreover, the concatenation conserves each subcircuit's measurement pattern, or set of needed one-qubit measurements. As a result, not only is the final (combined) unitary not guaranteed to conserve the original parameters used in the subcircuits (e.g. Equation 22), the approach does not give us a chance to even slightly modify the final measurement pattern in the hopes of getting more viable realizations, without recomputing the final state's

[9] even if the actual parameters to each unitary could be different as a result of the by-product operators propagations.

[10] Notice that since qubit 2 (logical output of the first application of the Hadamard) is measured in the second application, its measurement outcome becomes that of the first qubit in the second application (subcircuit), while the by-product operator acting on it is now acting on it's corresponding output qubit.

characteristic eigenvalue equations and deriving the form of Equation 10 again. The situation gets even more complex when one simply wants to experiment on an arbitrary state with an arbitrary set of measurements, with or without a specific unitary in mind, and to see whatever unitary and corresponding by-product operators can be obtained that way.

As a result, Danos *et al.* introduce a more generalized and systematic way to combine one-way circuits [8]. In this section, we will be giving a condensed overview of their work on the subject. The authors first provide a more complete definition of the notion of *pattern* and classifies the types of combinations into *tensor product* and *composition*, with the restriction that the composition between two gates be performed so that the number of logical outputs of the first gate is the same as the number of logical inputs in the second one.

1) Pattern definition: Viewing a one-way realization of a gate as graph, Danos *et al.* define its *pattern* as consisting of a *pattern type* and a *command sequence* [8]. The *pattern type* is composed of the set V of all vertices of the associated graph, the set of its logical inputs I, and that of its logical outputs O. On the other hand, the *command sequence* (cmdS) consists of the sequence of operations that has to be performed on the original graph state, in order to realize a desired unitary. As one can already deduce from the defining scheme of one-way computation, those operations can be an entanglement E_{ij} between two qubits i and j, a one-qubit measurement M_i^{α} of qubit i in angle α, and a Pauli correction operator X_i^s, or Z_i^s, on qubit i based on some value s called *signal*. Here, the *signal* consists of a sum (modulo 2 in this case, since $X \cdot X = Z \cdot Z = Id$) of integers and previous measurement outcomes, while the representation of a measurement is extended to include those that depend on previous measurements with signals s and t as

$$.^t[M_i^{\alpha}]^s = M_i^{(-1)^s\alpha + t\pi}$$

In addition to the representation of the special measurements in angles 0 and $\frac{\pi}{2}$ as M_i^x and M_i^y respectively, the authors also introduce a "*Shift*" operation S_i^r on qubit i based on r which consists of riding a given qubit's measurement of dependencies induced by Z-actions so that $.^t[M_i^{\alpha}]^s = S_i^t[M_i^{\alpha}]^s$, and given that applying a Z on a qubit before measuring it, is similar to measuring it and then swapping the measurement outcomes. The way this S_i^r acts on subsequent commands in the command sequence is by shifting the measurement outcome s_i of i by r, such as we will see later in the resulting standardization passes.

a) Example: For the J unitary, Hadamard, and CZ realizations we saw in Section II-B (Figure 7), and with respect to the indexing we have been using so far in previous example, the respective pattern definitions will be

$$\mathscr{J}_{\alpha}(1) = (\{1,2\},\{1\},\{2\},X_2^{s_1}M_1^{-\alpha}E_{12}) \tag{23}$$

$$\mathscr{H}(1) = (\{1,2\},\{1\},\{2\},X_2^{s_1}M_1^{0}E_{12}) \tag{24}$$

$$\wedge\mathscr{Z}(1,2) = (\{1,2\},\{1,2\},\{1,2\},E_{12}) \tag{25}$$

It is important to point out that this scheme is defined so that the following three main conditions hold, for a given command: (1) It never acts on already measured qubits, (2) its signals never refer to qubits that have not been measured yet, and (3) no logical output is measured.

In some cases such as that of the concatenation of two circuits, and hence two patterns — which is what we are interested in, we would want to make sure that the command sequence of the final pattern is defined so that all entanglements happen first, followed by all measurements, and then all Pauli corrections, as we have seen in the previous section. Danos *et al.* refer to the last condition as "*(EMC) condition*". When a given pattern does not satisfy that condition, it is called *wild*, as opposed to *standard* when it does. Additionally, the process of transforming a wild pattern into a standard one, which is possible, is called *standardization*.

2) Patterns combination and standardization: This said, the concatenation of two given subcircuits, will consist of first designing their individual subpatterns $P_1 = (V_1, I_1, O_1, cmdS_1)$ and $P_2 = (V_2, I_2, O_2, cmdS_2)$, then systematically concatenating those patterns into $P = (V, I, O, cmdS)$, and then deriving the final circuit from the resulting pattern. In a first time, this definitely involves making sure that all qubits in the subpattern are uniquely named so that there is no ambiguity in the naming of qubits in the final pattern. However, assuming that all qubits are so uniquely named, we have to consider two different types of gate combinations, that we would normally encounter in the network model: tensor product and composition.

In a tensor product, V_1 and V_2 do not overlap. So we can blindly combine the two patterns by performing a direct "union" of each component as

$$P = (V_1 \cup V_2, I_1 \cup I_2, O_1 \cup O_2, cmdS) \tag{26}$$

where, if CS_1 and CS_2 are standard and hence in the forms $C_1M_1E_1$ and $C_2M_2E_2$, then $CS = C_2C_1M_2M_1E_2E_1 = CME$.

For a composition, it is a bit tricky given that I_2 and O_1 overlap. Hence, the final pattern will consist of $V = V_1 \cup V_2$, $I = I_1 \cup (I_2/O_1)$, $O = (O_1/I_2) \cup O_2$, and $cmdS = cmdS_2 \cdot cmdS_1$. We see that in the resulting command sequence, we have a problem similar to the one we had in the earlier section before the propagation of the by-product operation U_{p1} over the unitary U_2. This calls for an application of a standardization algorithm, which consists of running the passes[11] in Figure 10, where $A_{\vec{k}}$ is an arbitrary command acting on $\vec{k}$ which is different from i and j, and $s[x/s_i]$

[11] Danos *et al.* provide more detailed information on the obtainment of these passes [8].

$$
\begin{array}{rlcl}
1) & E_{ij} X_i^s & \longrightarrow & X_i^s Z_j^s E_{ij} \\
2) & E_{ij} Z_i^s & \longrightarrow & Z_i^s E_{ij} \\
3) & E_{ij} A_{\vec{k}} & \longrightarrow & A_{\vec{k}} E_{ij}, \text{ with } A \neq E \\
4) & .^t [M_i^{\alpha}]^s X_i^r & \longrightarrow & .^t [M_i^{\alpha}]^{s+r} \\
5) & .^t [M_i^{\alpha}]^s Z_i^r & \longrightarrow & .^{t+r} [M_i^{\alpha}]^s \\
6) & A_{\vec{k}} X_i^s & \longrightarrow & X_i^s A_{\vec{k}}, \text{ with } A \neq X \text{ and } A \neq Z \\
7) & A_{\vec{k}} Z_i^s & \longrightarrow & Z_i^s A_{\vec{k}}, \text{ with } A \neq X \text{ and } A \neq Z \\
8) & .^t [M_i^{\alpha}]^s & \longrightarrow & S_i^t [M_i^{\alpha}]^s \\
9) & X_j^s S_i^t & \longrightarrow & S_i^t X_j^{s[t+s_i/s_i]} \\
10) & Z_j^s S_i^t & \longrightarrow & S_i^t Z_j^{s[t+s_i/s_i]} \\
11) & .^t \left[M_j^{\alpha}\right]^s S_i^r & \longrightarrow & S_i^r \, .^{t[r+s_i/s_i]} \left[M_j^{\alpha}\right]^{s[r+s_i/s_i]} \\
12) & \perp S & \longrightarrow & \perp
\end{array}
$$

Figure 10. Standardization passes.

represents the substitution of every occurrence of s_i in s with x.

Here, we add the last pass (12), where $\perp$ specifies an empty command sequence, to indicate that when a "*Shift*" command reaches the end of a command sequence, it is simply dropped, as explained by Danos *et al* [8]. We introduce a simple algorithm for implementing and running those standardization passes with acceptable efficiently later on. We will see that this scheme generalizes the by-product concatenation method, opening the doors to realizations where logical inputs can be measured in any angle, measurement patterns can be modified systematically without changing the meaning of the final realization or unitary, and circuits can be combined so that inputs and outputs overlap whichever way we desire. In fact, to satisfy the requirement that composition between two gates be performed so that the number of logical outputs of the first gate is the same as the number of logical inputs in the second one, Danos *et al.* also define the trivial pattern

$$\mathscr{T}(1) = (\{1\},\{1\},\{1\},\perp) \tag{27}$$

to be "tensored" as needed with each component of the composition until the requirement is met [8]. Since that pattern does not change anything to the final outcome of the composition, we will often leave it implicit in our patterns combinations for simplicity. So, a pattern such as $\mathscr{H}(3) \cdot \wedge\mathscr{Z}(1,3) \cdot \mathscr{H}(2)$ is really short for $(\mathscr{T}(1) \otimes \mathscr{H}(3)) \cdot \wedge\mathscr{Z}(1,3) \cdot (\mathscr{T}(1) \otimes \mathscr{H}(2))$.

a) Example: Consider the Identity, CNOT, and *CU* from Figures 8b, 8a, and 9a. For the Identity, we have to concatenate two uniquely named Hadamard's patterns $\mathscr{H}(1) = (\{1,2\},\{1\},\{2\},X_2^{s_1}M_1^0E_{12})$ and $\mathscr{H}(2) = (\{2,3\},\{2\},\{3\},X_3^{s_2}M_2^0E_{23})$, where the later is applied last. The resulting command sequence will first be

$$X_3^{s_2}M_2^0E_{23}X_2^{s_1}M_1^0E_{12}$$

and then

$$X_3^{s_2}Z_3^{s_1}M_2^0M_1^0E_{23}E_{12}$$

after running the standardization passes 1, 3, 4, and 7, and noticing that $(-1)^s 0 = 0$. Therefore, the final pattern will be

$$\mathscr{I}(1) = (\{1,2,3\},\{1\},\{3\},X_3^{s_2}Z_3^{s_1}M_2^xM_1^xE_{23}E_{12}) \tag{28}$$

For the CNOT, we will start from the combined pattern $\mathscr{H}(3) \cdot \wedge\mathscr{Z}(1,3) \cdot \mathscr{H}(2)$ and get the command sequence

$$X_4^{s_3}M_3^0E_{34}E_{13}X_3^{s_2}M_2^0E_{23}$$

After standardization with the sequence of passes {1, 3, 1, 7, 3, 4, 7, 7}, we will get the pattern

$$\begin{aligned}\wedge\mathscr{X}(1,2) \;=\; & (\{1,2,3,4\},\{1,2\},\{1,4\}, \\ & X_4^{s_3}Z_4^{s_2}Z_1^{s_2}M_3^xM_2^xE_{34}E_{13}E_{23})\end{aligned} \tag{29}$$

Similarly, the *CU* described by Equation 8, in Section II-B, will result in a pattern with command sequence

$$\begin{aligned}CU_{Aa} \;=\; & X_C^{s_B}M_B^0E_{BC}X_B^{s_A}M_A^{-(\alpha+\frac{\beta+\pi+\delta}{2})}E_{AB}X_k^{s_j}M_j^0E_{jk} \\ & X_j^{s_i}M_i^{-\beta-\pi}E_{ij}X_i^{s_h}M_h^{\frac{\gamma}{2}}E_{hi}X_h^{s_g}M_g^{\frac{\pi}{2}}E_{gh}X_g^{s_f}M_f^0E_{fg} \\ & E_{Af}X_f^{s_e}M_e^{-\frac{\pi}{2}}E_{ef}X_e^{s_d}M_d^{-\frac{\gamma}{2}}E_{de}X_d^{s_c}M_c^{\frac{\pi+\delta+\beta}{2}}E_{cd} \\ & X_c^{s_b}M_b^0E_{bc}E_{Ab}X_b^{s_a}M_a^{\frac{\beta-\delta+\pi}{2}}E_{ab} \\ =\; & Z_k^{s_i+s_g+s_e+s_c+s_a}X_k^{s_j+s_h+s_f+s_d+s_b}X_C^{s_B}Z_C^{s_A+s_e+s_c} \\ & M_B^0M_A^{-(\alpha+\frac{\beta+\gamma+\delta}{2})}M_j^0\left[M_i^{-\beta-\pi}\right]^{s_h+s_f+s_d+s_b} \\ & \left[M_h^{\frac{\gamma}{2}}\right]^{s_g+s_e+s_c+s_a}\left[M_g^{\frac{\pi}{2}}\right]^{s_f+s_d+s_b}M_f^0\left[M_e^{-\frac{\pi}{2}}\right]^{s_d+s_b} \\ & \left[M_d^{-\frac{\gamma}{2}}\right]^{s_c+s_a}\left[M_c^{\frac{\pi+\delta+\beta}{2}}\right]^{s_b}M_b^0M_a^{\frac{\beta-\delta+\pi}{2}} \\ & E_{BC}E_{AB}E_{jk}E_{ij}E_{hi}E_{gh}E_{fg}E_{Af}E_{ef}E_{de}E_{cd}E_{bc}E_{Ab}E_{ab}\end{aligned} \tag{30}$$

where qubits A and a are the control and target logical inputs, and C and k are the control and target logical outputs. All the other qubits are simply extra ancillae.

3) A simple concatenation and standardization (C&S) algorithm: After running a few concatenation examples, we notice a few points that helps us derive a simple and efficient algorithm for executing any 2-standard-pattern concatenations followed by a standardization of the resulting command sequence. That algorithm is used in the module *Qcompile_ow* of [11] and consists of the following.

For a 2-patterns tensor product, we already know from the above that we can simply apply the direct union used in Equation 26. Now, for the 2-patterns composition, let E_x, M_x, and C_x represent the sequence of entanglement, measurement, and Pauli correction operations of two standard sub-patterns 1 and 2, where x indicates the sub-pattern acted upon. The resulting V, I, and O can still be found as trivially as described earlier. The most interesting part is that of the composition of command sequences. Thus, the rest of the algorithm will be as displayed in Algorithm 1.

In step 1, we apply the standardization passes 1 and 2, and then 3 implicitly, since we already know M_1 and E_2 commute. In step 2, we use passes 4 through 7. Then, step 3a consists of pass 8, while step 3b consists of running passes

Algorithm 1 A simple C&S Algorithm.

1) *Propagate E_2 backwards, across C_1* $\longrightarrow E = E_2.E_1$
2) *Propagate M_2 backwards, across C_1* $\longrightarrow M = M_2.M_1$ *and* $C = C_2.C_1$
3) *For each M*
   a) *Introduce Shift*
   b) *Propagate the Shift forward, across M and C, dropping it at the end of the command sequence.*

11, then 9 and 10, and finally 12. Indeed, this algorithm can be used on subsequent concatenations to produce the pattern for a final circuit, as in the building-block approach described so far. However, although it already succeeds at skipping some unnecessary passes such as 3 and 12, it is not the most efficient algorithm at completing the task yet. Building such is subject to further studies, that may include some compiler optimization techniques, or some more remarks on the behavior of concatenated patterns. We introduce a few of such remarks in the next two sections (III-C and III-D). Meanwhile, we can already see that, in the resulting patterns, measurements will be of the form $[M_i^\alpha]^s$, where the dependencies induced by Z-actions have been dropped into the introduced and propagated *Shifts*.

C. Extending The Measurement Calculus

Here, we extend the measurement calculus from the previous section, as defined by Danos *et al.* [8], and include more standardization passes. We also present a reduced and more descriptive way to define patterns, based on some notations derived from the Haskell functional programming language. One will notice that this simplifies the presentation of more complicated examples of circuits concatenation using the measurement calculus, such as the derivation of the one-way realization of a Toffoli (or CCX) gate. In order to better understand the next examples, it is important to point out two things.

a) More standardization passes: First, the standardization passes listed earlier can be extended by at least four extra passes by noticing the equalities in Figure 11. We will be implicitly using them in the subsequent examples. Furthermore, they could be useful in the extension of the simple C&S algorithm introduced earlier into a more efficient one.

b) A monadic pattern representation: Second, we will be using monadic functions, from the Haskell implementation of superoperators as arrows defined by Vizzotto *et al.* [12], to provide a reduced representation of patterns and their concatenations, which only pays attention to ancillae qubits as necessary and shows a direct relationship with the corresponding network model representation. Here, it suffices to view a representation $func\,(in1,in2,...)\quad @>>=$

1) ${}^t\left[M_i^{-\pi/2}\right]^s = {}^t\left[M_i^{\pi/2}\right]^{s+1} = {}^{t+s+1}\left[M_i^y\right] = S_i^{t+s+1}M_i^y$
2) ${}^t\left[M_i^{\pi/2}\right]^s = {}^{t+s}\left[M_i^y\right] = S_i^{t+s}M_i^y$
3) ${}^t\left[M_i^0\right]^s = {}^t\left[M_i^0\right] = S_i^tM_i^x$
4) ${}^t\left[M_i^{3\pi/4}\right]^s = {}^{t+1}\left[M_i^{\pi/4}\right]^{s+1} = S_i^{t+1}\left[M_i^{\pi/4}\right]^{s+1} = S_i^{t+1}\left[M_i^{-\pi/4}\right]^s$

Figure 11. Extending the standardization passes.

$\backslash(out1,out2,...)$ as the application of a given gate represented by the function $func$, on a sequence of some input qubits $(in1,in2,...)$, that will produce the sequence of outputs $(out1,out2,...)$. In addition, the arrow $\rightarrow$ represents the next step and order of the concatenation, and the function *vreturn* is used to specify the final output qubits of the pattern. So, building the identity pattern in Equation 28 from the composition of Hadamard ones (Equation 24) could be presented as

$$\begin{aligned} Id\;q1 \;=&\\ Had\;q1\;@>>=&\quad \backslash q2 \;\rightarrow\\ Had\;q2\;@>>=&\quad \backslash q3 \;\rightarrow\quad vreturn\;q3 \end{aligned}$$

where qn represents qubit n. Notice that, here, an input cannot be given to a function unless it has been introduced before, either by the very first function (in this case Id) or a backslash $\backslash$.

Now, let us construct the one way realization of our Toffoli gate. We realize that it can be built from a CCZ by applying a Hadamard before and after the target qubit, as in the following representation.

$$\begin{aligned} toffoli\,(A0,B0,Cx) \;=&\\ Had\;Cx\;@>>=&\quad \backslash C0 \;\rightarrow\\ CCZ\,(A0,B0,C0)\;@>>=&\quad \backslash(A0,B5,C3) \;\rightarrow\\ Had\;C3\;@>>=&\quad \backslash C4 \;\rightarrow\\ vreturn\,(A0,B5,C4)& \end{aligned} \tag{31}$$

Additionally, a CCZ can be built up from controlled rotations as

$$\begin{aligned} CCZ\,(A0,B0,C0) \;=&\\ C\frac{\pi}{2}Phase\,(B0,C0)\;@>>=&\quad \backslash(B1,C1) \;\rightarrow\\ CNot\,(A0,B1)\;@>>=&\quad \backslash(A0,B2) \;\rightarrow\\ C(-\frac{\pi}{2})Phase\,(B2,C1)\;@>>=&\quad \backslash(B3,C2) \;\rightarrow\\ CNot\,(A0,B3)\;@>>=&\quad \backslash(A0,B4) \;\rightarrow\\ C\frac{\pi}{2}Phase\,(B4,C2)\;@>>=&\quad \backslash(B5,C3) \;\rightarrow\\ vreturn\,(A0,B5,C3)& \end{aligned} \tag{32}$$

We have already derived a CNOT (or CX) in Equation 29. However, we now have to build the patterns for the C$\frac{\pi}{2}$Phase and C$(-\frac{\pi}{2})$Phase gates as in the following subsections

1) Building up $C\frac{\pi}{2}$Phase and $C(-\frac{\pi}{2})$Phase patterns from CU: We realize that we can use the definition of a controlled unitary in Equation 8 to build a one-way realization of the $C\frac{\pi}{2}$Phase gate. In a first time, this consists of noticing that $\frac{\pi}{2}Phase = J(0)J(\frac{\pi}{2})$ [7], which is equal to $U = e^{i\alpha}J(0)J(\beta)J(\gamma)J(\delta)$ with parameters $\beta = \frac{\pi}{2}$ and $\alpha = \gamma = \delta = 0$. Thus, by substitution of those parameters with their respective values in Equation 30, we get:

$$\begin{aligned} C\frac{\pi}{2}Phase_{Aa} \quad = \quad & Z_k^{s_i+s_g+s_e+s_c+s_a} X_k^{s_j+s_h+s_f+s_d+s_b} X_C^{s_B} Z_C^{s_A+s_e+s_c} \\ & M_B^0 M_A^{-\frac{\pi}{4}} M_j^0 \left[M_i^{-\frac{3\pi}{2}}\right]^{s_h+s_f+s_d+s_b} \\ & \left[M_h^0\right]^{s_g+s_e+s_c+s_a} \left[M_g^{\frac{\pi}{2}}\right]^{s_f+s_d+s_b} M_f^0 \\ & \left[M_e^{-\frac{\pi}{2}}\right]^{s_d+s_b} \left[M_d^0\right]^{s_c+s_a} \left[M_c^{\frac{3\pi}{4}}\right]^{s_b} M_b^0 M_a^{\frac{3\pi}{4}} \\ & E_{BC} E_{AB} E_{jk} E_{ij} E_{hi} E_{gh} E_{fg} E_{Af} \\ & E_{ef} E_{de} E_{cd} E_{bc} E_{Ab} E_{ab} \end{aligned}$$

After using the extra standardization equalities listed earlier, we now have:

$$\begin{aligned} C\frac{\pi}{2}Phase_{Aa} \quad = \quad & Z_k^{s_i+s_g+s_e+s_c+s_a} X_k^{s_j+s_h+s_f+s_d+s_b} X_C^{s_B} Z_C^{s_A+s_e+s_c} \\ & M_B^x M_A^{-\frac{\pi}{4}} M_j^x S_i^{s_h+s_f+s_d+s_b} M_i^y M_h^x S_g^{s_f+s_d+s_b} M_g^y \\ & M_f^x S_e^{s_d+s_b+1} M_e^y M_d^x S_c^1 \left[M_c^{-\frac{\pi}{4}}\right]^{s_b} M_b^x S_a^1 M_a^{-\frac{\pi}{4}} \\ & E_{BC} E_{AB} E_{jk} E_{ij} E_{hi} E_{gh} E_{fg} E_{Af} \\ & E_{ef} E_{de} E_{cd} E_{bc} E_{Ab} E_{ab} \end{aligned}$$

and after propagating the "$Shift$" operations and dropping them at the end, using the standardization passes in Figure 10, we have

$$\begin{aligned} C\frac{\pi}{2}Phase_{Aa} \quad = \quad & Z_k^{s_i+s_g+s_e+s_c+s_a+s_h+s_d+s_b+1} X_k^{s_j+s_h+s_f+s_d+s_b} \\ & X_C^{s_B} Z_C^{s_A+s_e+s_c+s_d+s_b} \\ & M_B^x M_A^{-\frac{\pi}{4}} M_j^x M_i^y M_h^x M_g^y M_f^x M_e^y M_d^x \left[M_c^{-\frac{\pi}{4}}\right]^{s_b} \\ & M_b^x M_a^{-\frac{\pi}{4}} E_{BC} E_{AB} E_{jk} E_{ij} E_{hi} E_{gh} E_{fg} \\ & E_{Af} E_{ef} E_{de} E_{cd} E_{bc} E_{Ab} E_{ab} \end{aligned} \tag{33}$$

as a command sequence for a one-way realization of the $C\frac{\pi}{2}$Phase gate. The pattern is illustrated by Figure 9b. The same process goes for the design of the pattern for the $C(-\frac{\pi}{2})$Phase gate and we obtain the following as a command sequence, with parameters $\beta = -\frac{\pi}{2}$ and $\alpha = \gamma = \delta = 0$. The pattern is illustrated by Figure 9c.

$$\begin{aligned} C(-\frac{\pi}{2})Phase_{Aa} \quad = \quad & Z_k^{s_i+s_g+s_e+s_c+s_a+s_h+s_d+s_b} X_k^{s_j+s_h+s_f+s_d+s_b} \\ & X_C^{s_B} Z_C^{s_A+s_e+s_c+s_d+s_b+1} \\ & M_B^x M_A^{\frac{\pi}{4}} M_j^x M_i^y M_h^x M_g^y M_f^x M_e^y M_d^x \left[M_c^{\frac{\pi}{4}}\right]^{s_b} \\ & M_b^x M_a^{\frac{\pi}{4}} E_{BC} E_{AB} E_{jk} E_{ij} E_{hi} E_{gh} E_{fg} \\ & E_{Af} E_{ef} E_{de} E_{cd} E_{bc} E_{Ab} E_{ab} \end{aligned} \tag{34}$$

2) Building up the CCZ and Toffoli patterns: At this point, it goes without saying that we are ready to use Equations 33, 34, and 29, as well as the representation 32, to build the command sequence for the CCZ gate's pattern. We get the result displayed in Figure 12.

In later sections, we will see that the realization in Figure 14 agrees with this result as well as that in Figure 16 (which presents a realization after the optimizations discussed in the later Section V have been performed, removing measurements in qubits $a0$, $B1$, $a2$ and $B3$). We then build the Toffoli pattern the same way, using the representation 31, and obtain the result in Figure 13, for which, again in later sections, Figure 17a will provide a graphic realization (after the optimization passes have been performed, removing in this case qubits $j4$ and $C3$ as well).

However, the biggest challenge is in deriving such patterns for even more complex circuits. While this, again, points out the need for more efficient C&S algorithms as discussed earlier, the next section deals with a first-time by-pass of the problem by providing a simpler mechanism for deriving patterns that could temporarily ignore the need for the computation of the by-product operators.

At this point, we have presented a more systematic and general way to concatenate one-way circuit realizations, using a building block approach to generate a translation of a circuit from the network into the one-way computational model. This gives more room for simulations and experiments in the matter, as one can easily navigate across different sets of elementary realizations (universal sets). However, this relies on having predefined realizations with known unitaries and by-product operators, and hence does not address the question of possibilities to perform arbitrary experiments (combinations of states and measurement patterns) in the hopes of finding out what unitary and corresponding by-product operator are henceforth performed. We will come back to this problem in the later Section VIII.

D. The Graphical Concatenation

We now proceed to present our graphical concatenation. We explore the standardization passes and the C&S algorithm further, and notice a few interesting points, during the concatenation of two circuits 1 and 2 where some outputs of circuit 1 (O_1) become inputs of circuit 2 (I_2), which results in a new scheme of concatenation. This time, the concatenation scheme is based on graphical representations of one-way realizations, as introduced in Section I-A4 and used throughout this paper.

1) A reduced graphical concatenation scheme: From the one-way scheme definition, we know that there would be some Pauli corrections C_1 on O_1, and that I_2 will be measured (or kept as output of circuit 2) according to a measurement pattern M_2. During a concatenation, for each intersecting qubit (from O_1 to I_2), we notice the following relationships between an I_2 qubit's measurement and the corresponding

$$
\begin{aligned}
CCZ_{\{A0,B0,C0\}} &= (\mathscr{T}_{(A0)} \otimes C\frac{\pi}{2}Phase_{\{B4,C2\}}) \,.\, (\wedge\mathscr{X}_{\{A0,B3\}} \otimes \mathscr{T}_{(C2)}) \,.\, (\mathscr{T}_{(A0)} \otimes C(-\frac{\pi}{2})Phase_{\{B2,C1\}}) \\
&\quad .\, (\wedge\mathscr{X}_{\{A0,B1\}} \otimes \mathscr{T}_{(C1)}) \,.\, (\mathscr{T}_{(A0)} \otimes C\frac{\pi}{2}Phase_{\{B0,C0\}}) \\
&= Z_{C3}^{s_{i4}+s_{g4}+s_{e4}+s_{c4}+s_{C2}+s_{h4}+s_{d4}+s_{b4}+1}\, X_{C3}^{s_{j4}+s_{h4}+s_{f4}+s_{d4}+s_{b4}}\, X_{B5}^{s_{a4}}\, Z_{B5}^{s_{B4}+s_{e4}+s_{c4}+s_{d4}+s_{b4}} \\
&\quad M^x_{a4} M^{-\frac{\pi}{4}}_{B4} M^x_{j4} M^y_{i4} M^x_{h4} M^y_{g4} M^x_{f4} M^y_{e4} M^x_{d4} \left[M^{-\frac{\pi}{4}}_{c4}\right]^{s_{b4}} M^x_{b4} M^{-\frac{\pi}{4}}_{C2} \\
&\quad E_{a4B5} E_{B4a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{B4f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4} E_{C2b4} E_{b4B4} \\
&\quad .\, X_{B4}^{s_{b3}} Z_{B4}^{s_{B3}} Z_{A0}^{s_{B3}} M^x_{b3} M^x_{B3} E_{A0b3} E_{B3b3} E_{b3B4} \,. \\
&\quad Z_{C2}^{s_{i2}+s_{g2}+s_{e2}+s_{c2}+s_{C1}+s_{h2}+s_{d2}+s_{b2}}\, X_{C2}^{s_{j2}+s_{h2}+s_{f2}+s_{d2}+s_{b2}}\, X_{B3}^{s_{a2}}\, Z_{B3}^{s_{B2}+s_{e2}+s_{c2}+s_{d2}+s_{b2}+1} \\
&\quad M^x_{a2} M^{\frac{\pi}{4}}_{B2} M^x_{j2} M^y_{i2} M^x_{h2} M^y_{g2} M^x_{f2} M^y_{e2} M^x_{d2} \left[M^{\frac{\pi}{4}}_{c2}\right]^{s_{b2}} M^x_{b2} M^{\frac{\pi}{4}}_{C1} \\
&\quad E_{a2B3} E_{B2a2} E_{j2C2} E_{i2j2} E_{h2i2} E_{g2h2} E_{f2g2} E_{B2f2} E_{e2f2} E_{d2e2} E_{c2d2} E_{b2c2} E_{C1b2} E_{B2b2} \\
&\quad .\, X_{B2}^{s_{b1}} Z_{B2}^{s_{B1}} Z_{A0}^{s_{B1}} M^x_{b1} M^x_{B1} E_{A0b1} E_{B1b1} E_{b1B2} \,. \\
&\quad Z_{C1}^{s_{i0}+s_{g0}+s_{e0}+s_{c0}+s_{C0}+s_{h0}+s_{d0}+s_{b0}+1}\, X_{C1}^{s_{j0}+s_{h0}+s_{f0}+s_{d0}+s_{b0}}\, X_{B1}^{s_{a0}}\, Z_{B1}^{s_{B0}+s_{e0}+s_{c0}+s_{d0}+s_{b0}} \\
&\quad M^x_{a0} M^{-\frac{\pi}{4}}_{B0} M^x_{j0} M^y_{i0} M^x_{h0} M^y_{g0} M^x_{f0} M^y_{e0} M^x_{d0} \left[M^{-\frac{\pi}{4}}_{c0}\right]^{s_{b0}} M^x_{b0} M^{-\frac{\pi}{4}}_{C0} \\
&\quad E_{a0B1} E_{B0a0} E_{j0C1} E_{i0j0} E_{h0i0} E_{g0h0} E_{f0g0} E_{B0f0} E_{e0f0} E_{d0e0} E_{c0d0} E_{b0c0} E_{C0b0} E_{b0B0} \\
&= Z_{C3}^{\xi_{C3}} X_{C3}^{\chi_{C3}} X_{B5}^{s_{a4}} Z_{B5}^{\xi_{B5}} Z_{A0}^{\xi_{A0}} \,. \\
&\quad M^x_{a4} \left[M^{-\frac{\pi}{4}}_{B4}\right]^{s_{b3}+s_{a2}+s_{b1}+s_{a0}} M^x_{j4} M^y_{i4} M^x_{h4} M^y_{g4} M^x_{f4} M^y_{e4} M^x_{d4} \,. \\
&\quad \left[M^{-\frac{\pi}{4}}_{c4}\right]^{\mu_{c4}} M^x_{b4} \left[M^{-\frac{\pi}{4}}_{C2}\right]^{\mu_{C2}} M^x_{b3} M^x_{B3} \,. \\
&\quad M^x_{a2} \left[M^{\frac{\pi}{4}}_{B2}\right]^{s_{b1}+s_{a0}} M^x_{j2} M^y_{i2} M^x_{h2} M^y_{g2} M^x_{f2} M^y_{e2} M^x_{d2} \,. \\
&\quad \left[M^{\frac{\pi}{4}}_{c2}\right]^{\mu_{c2}} M^x_{b2} \left[M^{\frac{\pi}{4}}_{C1}\right]^{\mu_{C1}} M^x_{b1} M^x_{B1} \,. \\
&\quad M^x_{a0} M^{-\frac{\pi}{4}}_{B0} M^x_{j0} M^y_{i0} M^x_{h0} M^y_{g0} M^x_{f0} M^y_{e0} M^x_{d0} \left[M^{-\frac{\pi}{4}}_{c0}\right]^{s_{b0}} M^x_{b0} M^{-\frac{\pi}{4}}_{C0} \,. \\
&\quad E_{a4B5} E_{B4a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{B4f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4} E_{C2b4} E_{b4B4} \,. \\
&\quad E_{A0b3} E_{B3b3} E_{b3B4} \,. \\
&\quad E_{a2B3} E_{B2a2} E_{j2C2} E_{i2j2} E_{h2i2} E_{g2h2} E_{f2g2} E_{B2f2} E_{e2f2} E_{d2e2} E_{c2d2} E_{b2c2} E_{C1b2} E_{B2b2} \,. \\
&\quad E_{A0b1} E_{B1b1} E_{b1B2} \,. \\
&\quad E_{a0B1} E_{B0a0} E_{j0C1} E_{i0j0} E_{h0i0} E_{g0h0} E_{f0g0} E_{B0f0} E_{e0f0} E_{d0e0} E_{c0d0} E_{b0c0} E_{C0b0} E_{b0B0}
\end{aligned}
$$

with

$$
\begin{cases}
\mu_{C1} = & s_{j0} + s_{h0} + s_{f0} + s_{d0} + s_{b0} \\
\mu_{c2} = & s_{b2} + s_{b1} + \mu_{C1} + s_{a0} \\
\mu_{C2} = & s_{j2} + s_{h2} + s_{f2} + s_{d2} + s_{b2} + \mu_{C1} \\
\mu_{c4} = & s_{b4} + s_{b3} + s_{j2} + s_{h2} + s_{f2} + s_{d2} + \mu_{c2} + s_{a2} \;=\; s_{b4} + s_{b3} + \mu_{C2} + s_{a2} + s_{b1} + s_{a0} \\
\xi_{A0} = & s_{B3} + s_{b3} + s_{B2} + s_{j2} + s_{h2} + s_{f2} + s_{e2} + s_{c2} + s_{a2} + 1 \\
\xi_{B5} = & s_{B4} + s_{e4} + s_{c4} + s_{d4} + s_{b4} + s_{B1} + s_{B0} + s_{e0} + s_{d0} + s_{c0} + s_{b0} + \xi_{A0} + 1 \\
\chi_{C3} = & s_{j4} + s_{h4} + s_{f4} + s_{d4} + s_{b4} + s_{j2} + s_{h2} + s_{f2} + s_{d2} + s_{j0} + s_{h0} + s_{f0} + s_{d0} + s_{b0} \\
\xi_{C3} = & s_{i4} + s_{g4} + s_{e4} + s_{c4} + s_{h4} + s_{d4} + s_{b4} + s_{b3} + s_{C2} + s_{j2} + s_{i2} + s_{g2} + s_{f2} + s_{e2} \\
& + s_{c2} + s_{a2} + s_{C1} + s_{C0} + s_{i0} + s_{h0} + s_{g0} + s_{e0} + s_{d0} + s_{c0} + s_{b0}
\end{cases}
$$

Figure 12. Working out the CCZ's pattern.

$$
\begin{aligned}
toffoli_{\{A0,B0,Cx\}} = {} & X_{C4}^{s_{C3}+\xi_{C3}+s_{Cx}} Z_{C4}^{\chi_{C3}+s_{Cx}} X_{B5}^{s_{a4}} Z_{B5}^{\xi_{B5}+s_{Cx}} Z_{A0}^{\xi_{A0}} . \\
& M_{C3}^{x} . \\
& M_{a4}^{x} \left[M_{B4}^{-\frac{\pi}{4}}\right]^{s_{b3}+s_{a2}+s_{b1}+s_{a0}} M_{j4}^{x} M_{i4}^{y} M_{h4}^{x} M_{g4}^{y} M_{f4}^{x} M_{e4}^{y} M_{d4}^{x} . \\
& \left[M_{c4}^{-\frac{\pi}{4}}\right]^{\mu_{c4}+s_{Cx}} M_{b4}^{x} \left[M_{C2}^{-\frac{\pi}{4}}\right]^{\mu_{C2}+s_{Cx}} M_{b3}^{x} M_{B3}^{x} . \\
& M_{a2}^{x} \left[M_{B2}^{\frac{\pi}{4}}\right]^{s_{b1}+s_{a0}} M_{j2}^{x} M_{i2}^{y} M_{h2}^{x} M_{g2}^{y} M_{f2}^{x} M_{e2}^{y} M_{d2}^{x} . \\
& \left[M_{c2}^{\frac{\pi}{4}}\right]^{\mu_{c2}+s_{Cx}} M_{b2}^{x} \left[M_{C1}^{\frac{\pi}{4}}\right]^{\mu_{C1}+s_{Cx}} M_{b1}^{x} M_{B1}^{x} . \\
& M_{a0}^{x} M_{B0}^{-\frac{\pi}{4}} M_{j0}^{x} M_{i0}^{y} M_{h0}^{x} M_{g0}^{y} M_{f0}^{x} M_{e0}^{y} M_{d0}^{x} \left[M_{c0}^{-\frac{\pi}{4}}\right]^{s_{b0}+s_{Cx}} M_{b0}^{x} \left[M_{C0}^{-\frac{\pi}{4}}\right]^{s_{Cx}} . \\
& M_{Cx}^{x} . \\
& E_{C3C4} . \\
& E_{a4B5} E_{B4a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{B4f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4} E_{C2b4} E_{b4B4} . \\
& E_{A0b3} E_{B3b3} E_{b3B4} . \\
& E_{a2B3} E_{B2a2} E_{j2C2} E_{i2j2} E_{h2i2} E_{g2h2} E_{f2g2} E_{B2f2} E_{e2f2} E_{d2e2} E_{c2d2} E_{b2c2} E_{C1b2} E_{B2b2} . \\
& E_{A0b1} E_{B1b1} E_{b1B2} . \\
& E_{a0B1} E_{B0a0} E_{j0C1} E_{i0j0} E_{h0i0} E_{g0h0} E_{f0g0} E_{B0f0} E_{e0f0} E_{d0e0} E_{c0d0} E_{b0c0} E_{C0b0} E_{b0B0} \\
& E_{CxC0} .
\end{aligned}
$$

Figure 13. Working out Toffoli's pattern.

correction on O_1, with respect to the resulting pattern's measurement.

In general:

- An X correction in O_1 induces X-action dependencies (or simply dependencies) in the corresponding measurement on I_2.
  $\left[M_i^{\alpha}\right]^{s_2} X_i^{s_1} \quad \longrightarrow \quad \left[M_i^{\alpha}\right]^{s_2+s_1}$
- A Z correction in O_1, however, induces Z-action dependencies which are later on transformed into *Shifts*.
  $\left[M_i^{\alpha}\right]^{s_2} Z_i^{s_1} \quad \longrightarrow \quad .^{s_1}\left[M_i^{\alpha}\right]^{s_2} \quad \longrightarrow \quad S_i^{s_1}\left[M_i^{\alpha}\right]^{s_2}$

Notice that both cases always leave the principal measurement angle (α) unchanged. However, in the more particular cases of X- and Y- measurements:

- If I_2 is measured in the X-observable and there is an X correction in O_1, then the measurement on I_2 remains unchanged.
  $\left[M_i^{x}\right] X_i^{s_1} \quad \longrightarrow \quad \left[M_i^{x}\right]^{s_1} = \left[M_i^{x}\right]$
- A measurement in X and a Z correction introduces dependencies induced by Z-actions which are transformed into *Shifts*.
  $\left[M_i^{x}\right] Z_i^{s_1} \quad \longrightarrow \quad .^{s_1}\left[M_i^{x}\right] \quad \longrightarrow \quad S_i^{s_1}\left[M_i^{x}\right]$
- Y-measurements introduce *Shifts* independently of the corrections.
  $\left[M_i^{y}\right] X_i^{s_1} \quad \longrightarrow \quad \left[M_i^{y}\right]^{s_1} = .^{s_1}\left[M_i^{y}\right] \quad \longrightarrow \quad S_i^{s_1}\left[M_i^{y}\right]$
  $\left[M_i^{y}\right] Z_i^{s_1} \quad \longrightarrow \quad .^{s_1}\left[M_i^{y}\right] \quad \longrightarrow \quad S_i^{s_1}\left[M_i^{y}\right]$

Moreover, we also notice that the propagation of an O_1's X-corrections over the second circuit's entanglements (E_2) introduces Z-actions on all the neighbors of the corresponding I_2, which is later on transformed into *Shifts* as well, according to the remarks we just made.

This said, we can conclude the following (during the two circuits concatenation), keeping in mind that introduced *Shifts* are dependant on corresponding O_2's corrections signals:

1) All X-correction on an intersecting qubit will always introduce *Shifts* on its neighbors, while the measurement on the qubit itself,
   a) will either simply remain unchanged if it is in the X-observable,
   b) or remain unchanged, accompanied with a *Shift*, if it is in the Y-observable,
   c) or become dependant on the correction's signals otherwise.
2) Meanwhile, All Z-correction on the intersecting qubit will always leave its measurement unchanged, accompanied with a *Shift*.

In this paper's graphical representation, we are only concerned with whether measurements are adaptive (i.e. dependant on previous measurements) or not, and with the particular measurements in X, Y, and $\pm\frac{\pi}{4}$ when they are not. Thus, a concatenation scheme could simply be reduced to determining whether the condition *1c* above is satisfied or not (if the corresponding measurement was not adaptive to start with), and then blindly converting the representation for input qubits into that of regular qubits while conserving the measurement angle, for each intersecting qubit. We use such a reasoning, in addition to some optimization passes, throughout the rest of our needed concatenations from the Toffoli one-way realization, up to the practical example of Section VI. In the meantime, Figure 14 illustrates a step-by-step process of how one could use the graphical representations of controlled rotations to build up that of the CCZ whose pattern was defined in the previous section on the measurement calculus approach to one-way circuits'

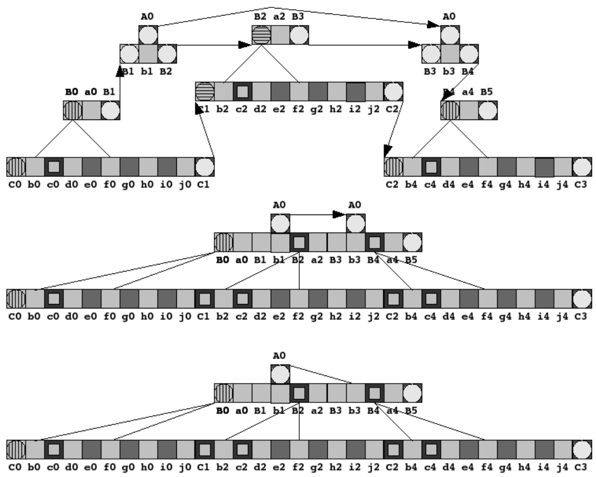


Figure 14. Building a CCZ gate from controlled rotations (graph) .

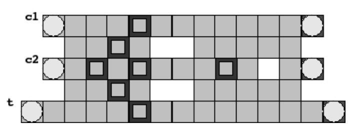


Figure 15. Extension of Toffoli Phase gate in Fig.6 into a regular Toffoli gate (cluster).

concatenation.

Additionally, the same process is used to construct the cluster realization of a Toffoli in Figure 15, by combining the cluster states universal set's CCZ and the parsimonious and robust universal set's Hadamards.

2) A general graphical concatenation scheme: In the reduced graphical scheme, we are not keeping track of the effect of the introduced *Shifts* in the final realization. Instead, we notice that the forward propagation of the *Shifts* never introduces non-pre-existing measurements or corrections, but instead modifies the signals, say s_2, of those pre-existing ones by adding on their own signals, say s_1, if s_2 contains the measurement outcome of the qubit that a given *Shift* is acting upon (cf. standardization passes 9, 10, and 11). So, whether the final circuit requires an adaptive (X) correction on the output qubits depends on whether the second circuit does so.

Furthermore, operations whose signals are modified by the *Shifts*' introductions can only get cancelled if the sum $s_2 + \sum s_i$ (for s_i representing the signals added by each modification on the signal s_2) modulo 2 is equal to 0. We notice that while measurement outcomes on independent s_i's may overlap — thus get cancelled every time they occur twice — since they all depend on measurement outcomes from the first circuit, the measurement outcomes in s_2 and $\sum s_i$, where all overlapping outcome has been removed, can never overlap since they both result from two different sets of measurements. Thus, the cancellation will only occur if $s_2 = \sum s_i = 1$, which is a rather rare case. So, we assume that the *Shifts*' propagation will never cancel signals — and hence neither measurements, nor corrections. Moreover, eventhough it modifies their signals, the final circuit will always conserve the type of measurements from both circuits and the type of corrections (X or Z) from the last circuit.

When the exact correction pattern in a circuit is unknown, it is safer to assume that there exists an X correction on each output one so that the corresponding input (in the later circuit) will be measured adaptively if it isn't yet, based on its original measurement angle.

Such is how we justify the functioning of the reduced graphical concatenation scheme we introduced earlier. We notice that it produces a final graphical representation in a more systematic and simpler way (without as much computation) than the C&S algorithm, which only consists of asking ourselves the following questions for each intersecting qubit:

1) *Is the corresponding measurement adaptive?*
2) *If not, is condition 1c met?*
3) *If not, what is the measurement's angle?*

In the end, we definitely do not have a complete pattern for the final circuit, as not only are the angles of adaptive measurements unknown, we have no indication of the exact final signals. Nevertheless, if we extend the graphical representation to include the specification of more particular measurement angles, the type of corrections (X or Z) that are performed on each output qubit, as well as the existing signals on each measurement and correction, we will see that we could easily introduce and propagate the *Shifts* accordingly, and then know the exact final pattern. Furthermore, by noticing that the *Shifts*' propagations affect only those operations (measurements and corrections) that have explicit signals (different from the number 1), we obtain an indication of a possibility to extend the C&S algorithm into a more efficient one that, among others, avoids non-adaptive measurements during the *Shifts*' propagation step.

IV. The Phase-Map Decomposition

At this point, we have studied a technique of translating circuits from the network to the one-way computational model that constitutes of breaking the network-model circuit into some elementary gates, finding their corresponding one-way realizations, and using a building block approach to concatenate those realizations into a final one-way realization for the original network circuit. We reviewed the original by-product approach, and its more systematic generalization called the measurement calculus. From the limitations of both, we extended the measurement calculus and derived a manual scheme that consists of analysing graphical representations, rather than pattern definitions of subsequent one-way realizations. We realize that the above methods are directly related to network circuit realizations and may require a lot more qubits than necessary. As a by-pass to this problem, Beaudrap *et al.* define a scheme where inputs and associated outputs of the network realization of a given circuit are studied in a way that derives a one-way realization without any reference to the network counterpart: The phase-map decomposition [6].

Let $\mathcal{H}_S$ denote the Hilbert space associated with the set S, and S^c denote the complement of S. According to the scheme,

"For every unitary $U : \mathcal{H}_I \longrightarrow \mathcal{H}_O$, there exists a phase map $\Phi : \mathcal{H}_V \longrightarrow \mathcal{H}_V$ such that:

$$U = R_O \cdot \Phi \cdot P_{I^c}$$

where $R_O : \mathcal{H}_V \longrightarrow \mathcal{H}_O$ is the restriction map to the output space, and $P_{I^c} : \mathcal{H}_I \longrightarrow \mathcal{H}_V$ is the preparation map: $|x\rangle \longmapsto |x\rangle \otimes |+...+\rangle_{I^c}$".

For example, the unitary J defined earlier in Section II-B can be decomposed as:

$$J(\alpha) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\alpha} & 0 \\ 0 & 0 & 0 & -e^{-i\alpha} \end{pmatrix} \cdot 2^{-1/2} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

where the middle map represents the phase map Φ. Since Φ is diagonal, the decomposition can be viewed as a diagonal decomposition and Φ is referred to as D [6].

Below follows Beaudrap *et al.*'s algorithm for finding that phase map, such that there exists a corresponding one-way pattern. First, we find a diagonal decomposition satisfying $U = R.D.P$, and since that does not always imply having an actual pattern definition, we find a matching one-way pattern (as defined in Section III-D) if it exists, or repeat the process with different parameters otherwise. Thus, given a unitary U:

1) Decompose U into a sum of orthonormal projections such that $U = \sum u_{pq} U_{pq}$.
   Notice the following conditions:
   a) $u_{pq} = \sum x_{pq}^{(i)}$, $i \leq 2^{n-|I|}$, and $x_{pq}^{(i)}$ are complex numbers.
   b) $2^{n/2}|x^{(i)}| = 1 \quad \Rightarrow \quad |x^{(i)}| = 2^{-n/2}$
2) Choose a number n of additional qubits
   Notice that $|u_{pq}| \leq 1$ implies that a safe choice for n (sufficient for a ϕ-map to exist) is such that $2^{n/2-|I|} \geq 1 \iff n \geq 2|I|$.
3) For each u_{pq},
   a) find the set $\{x_{pq}^{(i)}\}$, for $1 \leq i \leq 2^{n-|I|}$, such that the conditions (1a) and (1b) are satisfied.
4) Choose a permutation σ over $\{1,2,3, ..., 2^{n+|I|}\}$
5) Derive the set $D = \{d_{kk}\}$, for $1 \leq k \leq 2|V|$, and where $d_{kk} = \sqrt{2^n}\, x_{pq}^{(i)}$, such that the following conditions hold, and $U = R.D.P$.
   - binary representation of p agrees with that of k after the restriction R_o
   - $q \equiv k\, mod\, (2^{|I|})$
   - $i = \sigma\,(k/2^{|I|})$

   Notice that having a decomposition D does not imply having a pattern, hence the next step.
6) For all $\vec{x}$ in V-computational basis, can we find (G_E, $\{\alpha_j\}$) — an entanglement graph G_E over V, and a corresponding set of measurement angles $\{\alpha_j\}$ — such that $d_{xx} = e^{-i\sum_{O^c} \alpha_j x_j} e^{i\pi \sum_{j,k \in E} x_j x_k} = e^{i\lambda(x)}$ with $\{d_{kk}\}$ = $D = \Phi$?
   a) For $j \in \{1, 2, ..., |O^c|\}$ ($j \in O^c$)
      i) consider a $|V|$-bit string z_j, which only has 1 at position j.
      ii) set α_j such that $e^{-\alpha_j} = d_{z_j z_j}$.
   b) for all j and k, consider a $|V|$-bit string z_{jk}, which only has 1 at positions j and k. Is $d_{z_{jk}z_{jk}} = \pm e^{-i(\alpha_j + \alpha_k)}$?
      i) If Yes, then sign is -1, and E_{jk} is an edge in G. In other words, the qubits corresponding to j and k are entangled.
      ii) Otherwise, There is no matching graph for this combination, and we have to backtrack

Here, the backtracking step consists of repeating the following steps, while we can, and only move onto the next when we run out of possibilities: first step 5, then step 3, then step 2, and then expand further the computational space.

We then see that with this algorithm, it is very possible to specify a number of extra ancillae lower than the ones we have derived so far, and see if there could exist a corresponding one-way pattern. More particularly, let's consider the CCZ and Toffoli example we had earlier. Instead of building them up from the concatenation of the more elementary Hadamard and controlled rotations, we could experiment with finding the smallest possible phase map decomposition for the Toffoli. Alternatively, we could find such a decomposition for the CCZ gate and derive another realization for the Toffoli from there using a building block approach as earlier, only this time with a different pattern definition for CCZ.

This extends our experimental range even further and, unlike the building block approaches, allows us to experiment with arbitrary sets of qubits and measurement patterns, without any knowledge of the particular unitary operation realized or of the corresponding by-product operators. In addition, it gives us the capability to derive those unknown parameters later. We discuss this further in Section VIII.

V. Optimizing the results: Extending the Measurement Calculus Further

Let's say we now have a transformed circuit, independently of the chosen universal set and method of concatenation. We still need to improve our obtained circuit for various reasons that contribute to a common factor: The ease of physical realizability. In most cases, and as intuitively expected, one would like to make the resulting graph states as small as possible, i.e. with few qubits. However, there exists special cases for which it would be more advantageous to slightly increase the number of qubits necessary for the final realization. In this section, we introduce the notions of Graph State Reduction (GSR) and Graph State Extension (GSE) transformations, to classify currently defined optimizations and abstract them out into additional transformation passes. In order to maximize the effectiveness of our transformations — which would be by finding the smallest graph state that satisfies our realization needs, one will find it preferable to run the GSR transformations prior to the GSE ones. We will see that these transformations, again, extend the measurement calculus' passes defined in Section III-B further, and allow for arbitrary measurements that are not limited to the X-Y plane.

A. *Graph State Reduction transformations*

The likelihood of a quantum state to decohere (get entangled with the surrounding environment), in the course of a computation, is directly proportional to the size of the state itself. Essentially, it is proportional to the amount of multi-particles interactions imposed by the computation's algorithm, as well as the depth of that algorithm itself. Alternatively, at times, one is simply limited by the availability of physical systems that we could use to construct each needed qubit. Hence, when building up algorithms, it is be essential to keep the number of required qubits as small as possible. In fact, Raussendorf *et al.* [5] describes a scheme by which we could get around the physical device's size limitations and better control decoherence.

The scheme consists of breaking up the computation, or one-way circuit/pattern associated with it, into several smaller "chunks", so that the implementations of those chunks, consecutively, yields to the same final result. Here, each chunk is composed of a re-entanglement of the (sub-)clusters followed by the application of the corresponding subset of measurements. Thus, the scheme requires that the computation be broken up so that the input qubits of a subsequent chunk consists of the output qubits of the current one.

Following this scheme, we notice that commuting operations, which could be performed in a single (or reduced) time frame, would find themselves separated into different chunks. This will then increase the computation time, which brings us back to our need to keep the resulting circuit as small as possible. To this end, several techniques of optimizations have been the subject of numerous works. Among others, Hein *et al.* [17] and Schlingemann [9] have presented some graph state transformation rules and resulting equivalence classes that could be used in such effect. More importantly, these particular techniques all revolve around the removal of the Clifford part (see Section III) of a graph. According to the later, from a given measurement pattern, measurements in the Pauli observables (X, Y, and Z) can be removed without changing the meaning of the computation, especially since the randomness of quantum measurements results can be accounted for in the correction part of the pattern during classical post processing. The main explanation for this is the observation that performing all such measurements in a graph state produces an equivalent (reduced) graph state satisfying the eigenvalue Equations 9, with a different set of correlation operators, acting on the remaining qubits. (Raussendorf *et al.*'s work already gives a detailed and technical explanation for this [5]. So, we refer the user to it for additional information.)

In any event, and more particularly, two main such transformations are as follows:

- The removal of redundant qubits
  This concerns situations where one comes across an already formed cluster state with far more qubits than needed. Such situation needs not always result in a total reconstruction of the state. Instead, we can always remove the extra, i.e. redundant, qubits simply by measuring them in the Z-observable. As a result, and as explained in detail by Raussendorf *et al.* [5], the remaining qubits will have to be modified by a phase (i.e. we must apply a Z operation to each one of them), based on each measurement outcome and before any other operation. Thus, say Q and Q_N, respectively, are the sets of all qubits in an initial cluster and in the one remaining after a set of Z measurements. Type-lifting the notations from Section III-B, M^z_{Q/Q_N} and s_{Q/Q_N} will represent, respectively, the sets of all measurements in Z and of all resulting signals. Hence, the transformation pass will be:
  $$C_{Q_N} M_{Q_N} E_Q \quad \longrightarrow \quad C_{Q_N} M_{Q_N} Z_{Q_N}^{s_{Q/Q_N}} M^z_{Q/Q_N} E_Q$$
  which will result in the pattern $C_{Q_N} M_{Q_N} E_{Q_N}$; thus, $E_{Q_N} = Z_{Q_N}^{s_{Q/Q_N}} M^z_{Q/Q_N} E_Q$.
- The removal of unnecessary measurements
  In relating one-way patterns to graphs, Raussendorf *et al.* illustrate that whenever pairs of adjacent qubits with measurements in the X-observable are "surrounded above and below by either vacant lattices or Z-measurements, they can be removed from the pattern without changing the logical operation of the gate" [5]. Let's call this "*Property X*." In other words, for example, consider a chain of four qubits a, b, c, and d such that a is entangled (CZ) with b, b with a and c, c with b and d, and d with c. Measuring both b and c in the X-observable is as good as not having them at all and having a system consisted of only a and d, where the later are entangled with each other.
  Another way to look at this is to consider the identity protocol (or one-qubit teleportation) defined earlier. Let's define a *Hadamard transformation* as a version of one-qubit teleportation constituted of a single "wire", entangling two qubits, where information is transported from the first qubit to the second by measuring the

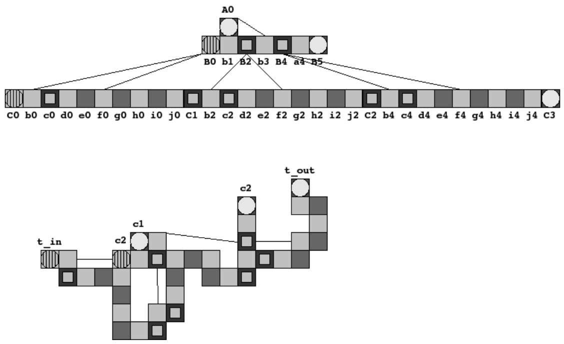
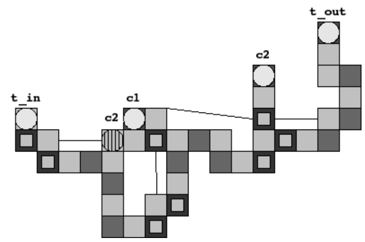
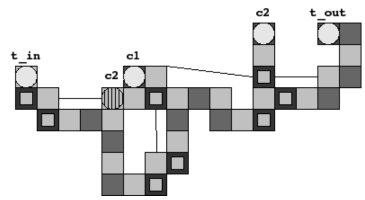


Figure 16. CCZ after removal of unnecessary measurements (graph) .



(a) Before



(b) After

Figure 17. Extension of CCZ to Toffoli gate, before and after the removal of unnecessary measurements (graph).

first one in X. In other words, it is simply the one-way realization of the Hadamard operation, as in Figure 7c. Thus, the identity protocol, which is then nothing but the Hadamard transformation of the Hadamard transformation of the input, successfully and conservatively carries the information from the input qubit to the output one, through an X-measurement of the input and middle qubits. So, interacting with a given qubit, directly, is as good as interacting with the same qubit a couple double-Hadamard transformations away.

The transformation pass for this will then be:

$$C_Q M_{Q_N} M^x_{Q/Q_N} E_Q \quad \longrightarrow \quad C_{Q_N} M_{Q_N} E_{Q_N}$$

where M^x_{Q/Q_N} represents the set of measurements in X such that the qubits satisfy "*Property X*", and hence such that the size of Q/Q_N is even. In addition, the resulting pattern is modified by removing all reference (e.g. entanglement edges and measurement outcomes) to the qubits that the removed measurements were acting upon.

For illustration, we use such reasoning in our process of deriving the graphical realizations of the CCZ and Toffoli gates in Figures 16 and 17.

B. Graph State Extension transformations

After reducing the graph as much as possible, it would be important, in some special cases, to extend it a bit in order to fit some necessary physical properties. This should especially not be troublesome if we can break the resulting circuit's pattern down effectively (as described previously in Section V-A). Nevertheless, concerning the physical properties needed, so far, we can only think of the transformations that we call "*clusterization*" and "*two-colorization*" as special cases.

a) Clusterization: The term "*clusterization*" refers to any transformation of general graph states into cluster states. The reason for this application is the physical realizability advantage of cluster states over that of graph states in several quantum computation devices, especially in optical lattices.

As stated earlier, the transformations defined by Hein *et al.* [17] and Schlingemann [9] already illustrate possibilities for this, but to the best of our knowledge, there is no completely defined calculus implemented for this yet. This gives an opening for future work. Indeed, such work would also require the definition of factors for identifying that a given graph state is in fact a cluster state. And perhaps one would think of such factors as incorporating the "distance" between entangled particles and the number of neighbors for each qubit.

b) two-colorization: Danos *et al.* state that "2-colorable entanglement graphs are interesting since purification protocols exist for their associated graph states, making them physically implementable in a way that is robust against decoherence. So it is good news that such robust implementations can be obtained in the one-way model for all unitary operators." An entanglement graph is 2-colorable when the distance from any input to any output qubit is even, and *two-colorization* refers to the transformation of a given general graph state into one that satisfies this particular property. It is already known that cluster states are a particular case of 2-colorable states. This indicates that we could optimize our results by limiting the range of our final realizations to cluster states. Nevertheless, as illustration, the controlled phase gates of Figure 9, built from the controlled unitary in Section III-B, is 2-colorable as expected [7], [8]. Therefore, so is the derived CCZ gate in Figure 16. However, due to the extra Hadamard transformations in the resulting Toffoli gate (Figure 17), the later is not 2-colorable.

In any event, the particular transformation passes in the GSE case can be generalized as converses of GSR transformations and as follows:

- The introduction of Z measurements.

At times, during a graphical concatenation step, we may find ourselves in a situation where the possibilities of maintaining a particular structure — such as a cluster state one — in the final realization, is jeopardized. This could result from situations where the logical input or output qubits, of a given realization, are not aligned evenly enough to ease any concatenation, with any other realization, that would still maintain the structure of interest. Among others, we would want all logical output qubits in a previous realization to "overlap" with their corresponding logical input qubits in the subsequent realization. However, this cannot always be guarantied, especially if there exists matching input and output qubits for which the "distances" (in the associated lattice representation) between the two qubits are even while that between other matching qubits are odd and vice-versa[12]. For a graph state structure, we might opt for overlapping one set of matching qubits — either those with even distances or odd ones — and then connecting the other ones with wires (as per the definition of our graphical representations). Alternatively, especially for a cluster state structure, instead of wires, we could insert an additional ancilla (between the two matching and non-overlapping qubits), only to remove it later through a measurement in the Z observable. Evidently, this technique will be using the removal of redundant qubits transformation as a feature for concatenation, or GSE optimization nonetheless. The resulting system will then behave just as if the original matching qubits had overlapped with each other in the first place. Thus, we introduce the following transformation pass:

$$C_{Q_N} M_{Q_N} E_{Q_N} \quad \longrightarrow \quad C_{Q_N} M_{Q_N} Z_{Q_N}^{s_{Q/Q_N}} M^z_{Q/Q_N} E_Q$$

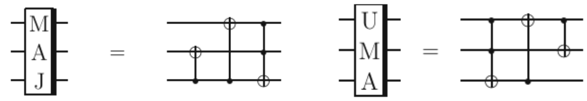
which will eventually produce a $C_Q M_Q E_Q$ pattern after propagation of the $Z_{Q_N}^{s_{Q/Q_N}}$ corrections, and introduction of the *Shift* commands, across the M_{Q_N} measurements and the C_{Q_N} corrections.

- The identity propagation.
  Similarly, we have the almost-converse of the removal of unnecessary measurements, which applies double Hadamard transformations whenever needed. The practical example in Section VI will give a clear illustration of usage of this pass. The transformation pass will then be:

$$C_{Q_N} M_{Q_N} E_{Q_N} \quad \longrightarrow \quad C_{Q_N} M_{Q_N} C_{Q_N}^{s_{Q/Q_N}} M^x_{Q/Q_N} E_Q$$

  where, again, the size of Q/Q_n is even.

It is important to clarify that these are not the only existing transformations. In addition to the Hadamard transformation itself which can be used very cleverly in combination with other techniques, Schlingemann studies the removal of the Clifford parts of a given graph state in depth, and presents additional transformations [9].



(a) (b)

Figure 18. The in-place majority gate MAJ (a) and "UnMajority and Add" gate UMA (b). [Picture from [1]]

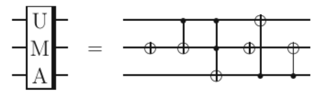


Figure 19. UMA_par: Version of the "UnMajority and Add" gate that admits greater parallelism. [Picture from [1]]

VI. Practical Example: A (new) One-Way Quantum Ripple Carry Addition Circuit

Now that we have gathered all the information needed for the application of the translation scheme we defined at the beginning of this work, it would be interesting to see it work in practice. To this end, we take a special look at Cuccaro *et al.*'s "new quantum ripple-carry addition circuit" [1].

A. The new quantum ripple-carry addition circuit

Cuccaro *et al.* define a new quantum ripple-carry addition circuit, in the network model, which, not only reduces the number of ancillae normally needed in previous ripple-carry addition circuits (cf. [13], [14], [15]) down to a single one from a linear bound, but also lowers the depth and the number of gates needed, producing a version with higher parallelism [1]. This definition presents several levels (or subdivisions) of circuits definitions up to the final optimized one. First, the authors define a "majority" (MAJ) and an "unmajority and add" (UMA) gates so that their composition produces a half adder (cf. Figures 18 and 20), and the MAJ gate itself produces the input carry for the next two qubits' addition (if any). Then, adding an extra ancilla and inserting a CNOT (on that ancilla) between the composition of the MAJ and UMA gates produces a full adder (cf. Figure 21a). The later is then extended to a multi-qubits addition circuit where subsequent applications of the MAJ and UMA gates, for each pair of qubits, are composed together in a manner depicted in Figure 22a.

In parallel, they redefine the UMA gate (cf. Figure 21b) to provide a higher parallelism version of the previous multi-qubit adder, which is illustrated in Figure 22b. To name gates involved in this version, we append a "*_par*" at the end of their corresponding non-optimized gates. For example, we

[12] Notice that, by "*distance*" between two qubits here, we mean the number of ancillae qubits that can be padded between the qubits, in a rectangular lattice representation, so that all qubits form a chain of entangled qubits.

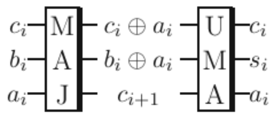
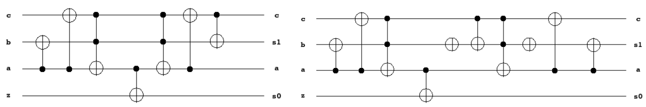
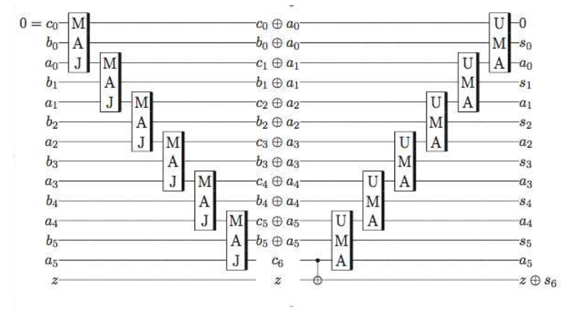


Figure 20. Basic addition circuit (Half adder). s_i is the modular sum of the inputs a_i and b_i, and $c_i$is the incoming carry bit. [Picture from [1]]

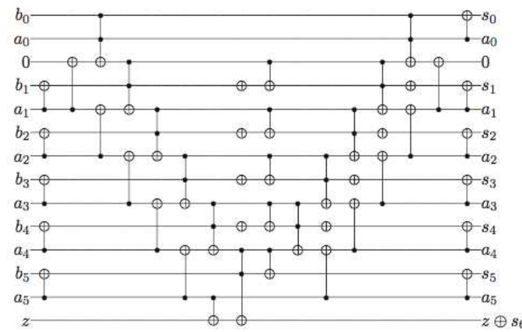


(a) (b)

Figure 21. Extension of Fig. 20 to full addition (a), and its High parallelism version (b).

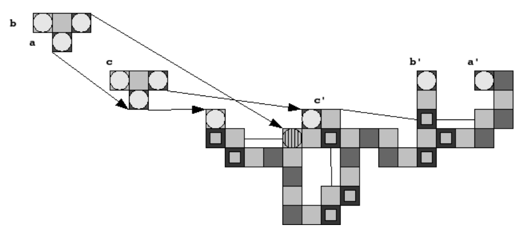


(a)

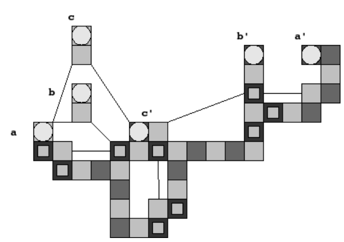


(b)

Figure 22. 6-qubit generalization of Fig. 21 (a), and its High parallelism version (b). [Pictures from [1]]



(a) Step 1



(b) Step 2: after removal of unnecessary measurements

Figure 23. Building the MAJ gate (graph).

call this version of the UMA gate the UMA_par gate. Figures present all the circuits briefly described herein, and that we will be translating.

B. To a One-way realization

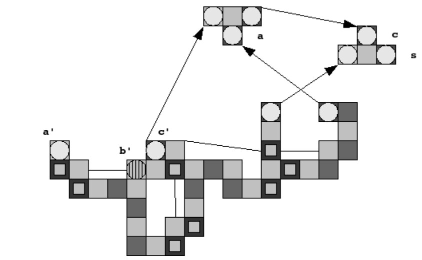
As described in the translation scheme, the first step is to define different levels of subdivisions of this circuit. We notice that we can start with the MAJ, UMA, and UMA_par gates, and combine them to produce the ADD1 and ADD1_par subcircuits, which can in turn be combined several times into the ADD6 and ADD6_par circuits. In the meantime, The "starting" gates themselves (MAJ, UMA, and UMA_par) can be decomposed into the elementary NOT, CNOT, and CCNOT gates.

Throughout previous sections, we collected and derived both graph-state and cluster-state realizations for the CNOT and CCNOT gates. A realization of the NOT gate can be obtained simply by adding an extra X correction to the output of the realization of an Id. As a matter of fact, if we look at X as $J(\pi)\cdot J(0)$ (as defined by Danos *et al.* [7]), we can build a pattern for it from the composition $\mathscr{J}_\pi(2)\cdot\mathscr{H}(1)$ which will result in the command sequence

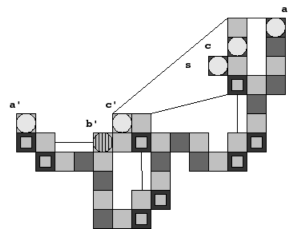
$$X_3^{s_2+1}Z_3^{s_1}M_2^xM_1^xE_{23}E_{12} \tag{35}$$

Now, using on the removal of unnecessary measurements optimization technique described in the previous section, we can build up the graph-state realizations for the MAJ, UMA, UMA_par, ADD1, and ADD1_par circuits, from the building block approach techniques discussed so far, and using our graphical concatenation. Those realizations are shown in Figures 23 through 27.

For the cluster-state realizations of the MAJ and UMA circuits in the same fashion, notice that the introduction of Hadamard transformations at the beginning and end of the third logical qubits of the cluster version of the CCZ gate — which produced the needed CCNOT one, causes the occurrence of "distance" mismatch between pairs of to-be-overlapped logical output and input qubits. It appears that we have a choice between only overlapping the third logical qubits, or the first and second ones, and having a graph state either way — since the non-overlapped qubits will have to be connected by explicit wires. However, since the existence of the introduction of Z-measurements optimization technique,

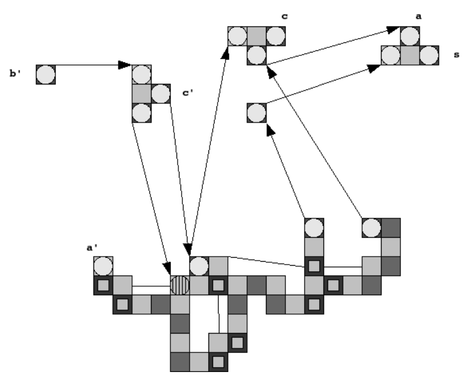


(a) Step 1

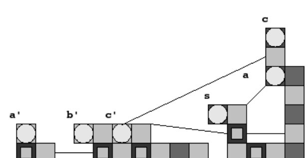


(b) Step 2: after removal of unnecessary measurements

Figure 24. Building the UMA gate (graph).



(a) Step 1

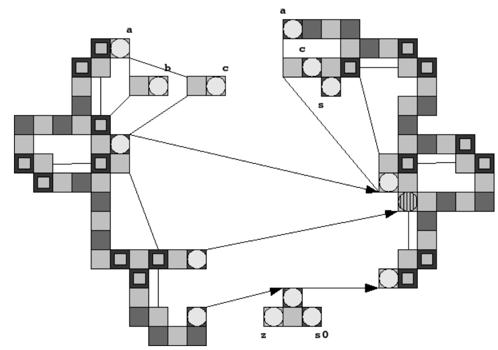


(b) Step 2: after removal of unnecessary measurements

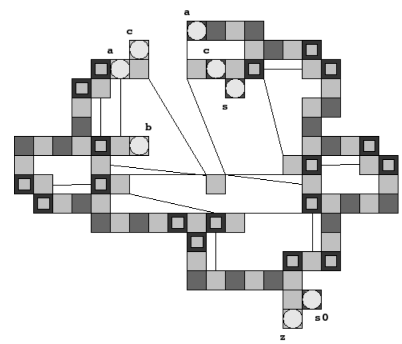
Figure 25. Building the UMA_par gate (graph).

we can bypass the need for those wires and derive translations for the MAJ, UMA, and UMA_par circuits as depicted in Figures 28, 29, and 30. Additionally, we can derive those for the ADD1 and ADD1_par circuits as in Figures 31 and 32.

Finally, through identity propagation transformations, we can extend the cluster-states one-qubit addition circuits to the



(a) Step 1



(b) Step 2: after removal of unnecessary measurements

Figure 26. Building the full addition circuit (graph).

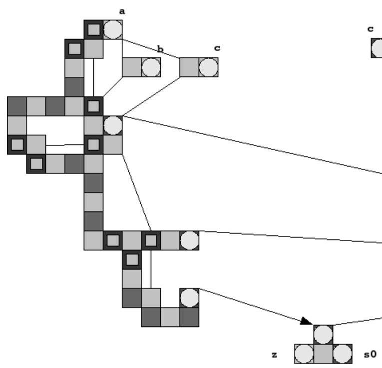
6-qubits ones as in Figures 33 and 34. Notice that the carry produced by each MAJ gate becomes input to the next MAJ gate, while that produced by the UMA gate becomes input to the previous UMA gate.

VII. Analysis: No Optimization Translation

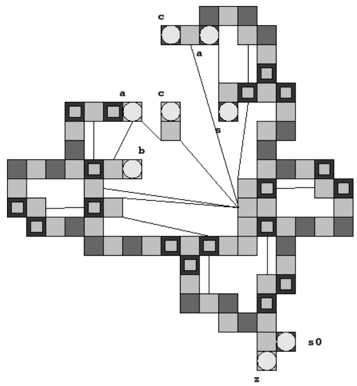
After any definition of a particular realization, it is common practice to analyze its efficiency. In doing so, several quantifiable parameters come into play, depending on the model in use, and are associated to particular values based on a careful study of the algorithm that defines the realization. The process is called *algorithm analysis* and, indeed, varies from one model of computation to another. However, as we will see soon, the corresponding parameters often have the common factors of *time* and *space*. In this section, we analyse the efficiency of our translations and translated circuits, and will notice that optimizations are not always conserved across different models of computations.

A. Algorithm analysis in the one-way model

Performance analyses are generally concerned with two main parameters: The computational time (or time spent performing the computation), and the computational space (or space needed to perform the computation). These parameters are expressed as functions of the input size, and their actual

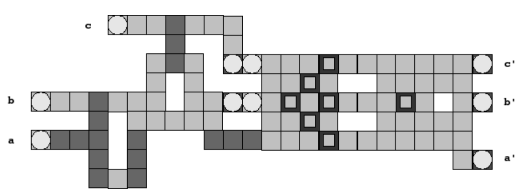


(a) Step 1

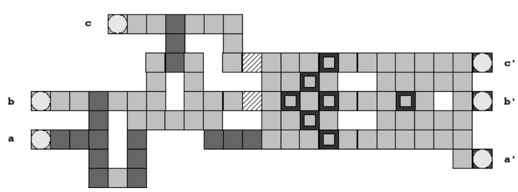


(b) Step 2: after removal of unnecessary measurements

Figure 27. Building the high parallelism version of the circuit in Fig. 26 (graph).



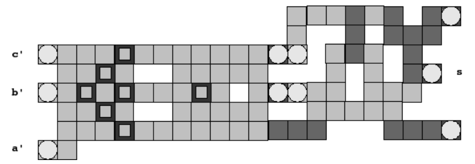
(a) Step 1



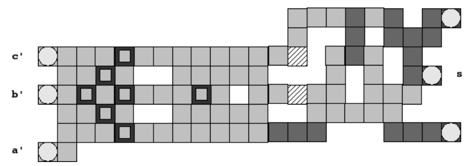
(b) Step 2: after introduction of Z measurement and some removal of unnecessary measurements

Figure 28. Building the MAJ gate (cluster).

definition varies from one set of realizations to another. In classical computation, time is often associated with the number of sequential instructions required (or some rational function of it in the case of parallel computation), while space is the amount of memory resources needed other than the

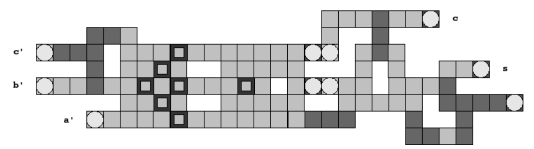


(a) Step 1

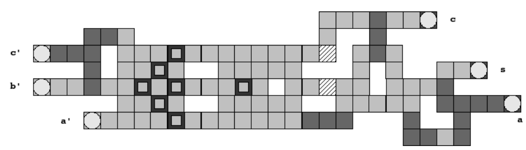


(b) Step 2: after introduction of Z measurement and some removal of unnecessary measurements

Figure 29. Building the UMA gate (cluster).



(a) Step 1



(b) Step 2: after introduction of Z measurement and some removal of unnecessary measurements

Figure 30. Building the UMA_par gate (cluster).

memory space occupied by the input bits.

Quantum computation takes those notions a bit further and defines space as the number of quantum registers (occupied by qubits) needed. Because of the reversibility aspect of the computation, algorithms are in-place — meaning that the space does not change during the computation — and each extra qubit that is added onto the logical input ones is called an *ancilla*. Thus, algorithm analyses often contain an account of the size of the whole space in use and the number of required ancillae. This is important as the preparation of states is still a major issue in quantum computation.

In parallel, time is now defined as *depth* of the computation. Again, because of the reversibility aspect, as described by Bennet, a computation is supposed to perform the corresponding algorithm until the end, copy out the results into other (classical) registers, and run the algorithm backwards to obtain the initial state [16]. This then gives a tree-like structure to the actual overall algorithm, with one outgoing way descending the tree while the other reversing one ascends it. Thus, since the analysis often only defines the outgoing way of the algorithm, we define the time corresponding to that algorithm as the depth of the actual overall one.

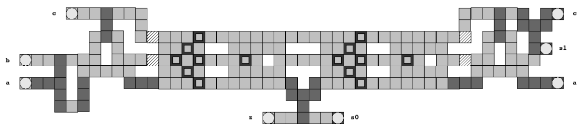


Figure 31. Building the full addition circuit (cluster).

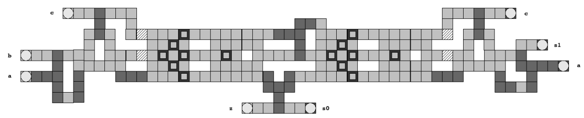


Figure 32. Building the high parallelism version of the circuit in Fig. 31 (cluster).

Often, as actual physical realizations of particular gates — such as the CNOT and Toffoli gates — may differ, a count of each individual type of gate is also included in the algorithm analysis.

In relation to our practical example from the previous section, Cuccaro *et al.* analyse the performance of the network-model addition circuits previous the previous section and find that, for input size $n \geq 2$, the circuit size of the higher parallelism version (Figure 22b) is "$2n-1$ Toffoli gates, $5n-3$ CNOTs, and $2n-4$ negations. The depth is $2n+4$" and the number of ancillae is 1. This is better than that of the non-optimized version (Figure 22a) which consists of n applications of the MAJ and the UMA gates, both with 1 Toffoli and 2 CNOTs each, plus an extra CNOT, resulting in a depth of $6n+1$, still with a single ancillae [1].

In the one-way model, the parameters definitions are extended further into *spacial*, *temporal*, and *operational* resources. While the *spacial resources* still refer to the number of qubits needed, *operational resources* refer to the number of evolutionary operations performed, consisting in this case of one-qubit measurements[13]. (Thus, the size of operational resources will always be less than or equal to that of spacial resources). Since measurements that are independent of each other can always be performed in parallel in a single round, *temporal resources* refer to the number of such measurement rounds required by an algorithm, and are called *logical depth*.

Given these parameters, and with respect to the subject of this paper, it is important that we pay some attention to how well optimizations are transmitted across computational models.

B. Circuits translations analysis

Given the scalability advantage of the one-way quantum computation model over the standard one, and the optimizations performed by the new quantum addition circuit in the previous section, we are left with the single question of whether circuits that are optimized in the one-way model will remain so, after translation. We already know that one-way computation do a great deal of space usage, and thus operations usage. The only question in that context is now that involving the factors discussed in the previous section (V). So, here we only focus on the change in temporal resource.

[13] which are applied no more than once on each qubit.

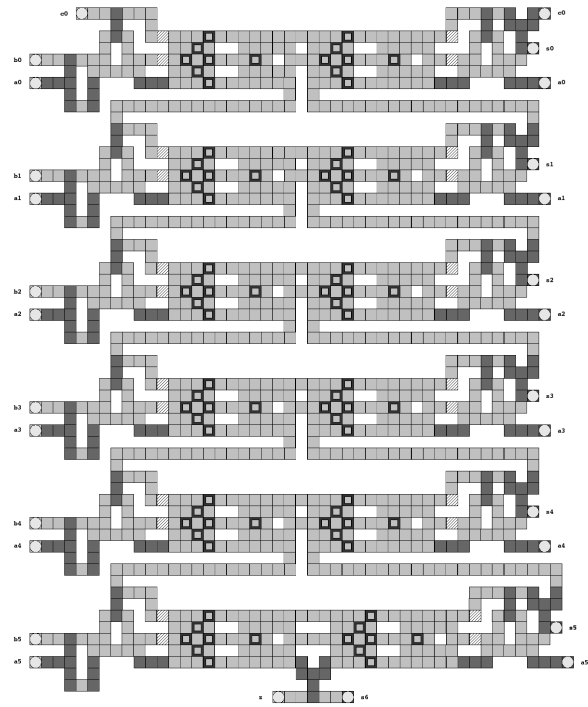


Figure 33. Building the 6-qubit generalization of Fig. 31 (cluster).

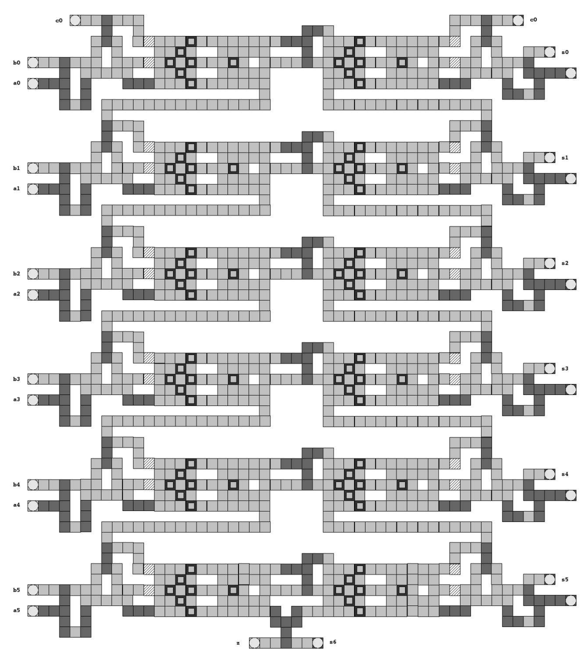


Figure 34. Building the high parallelism version of Fig. 33 (cluster).

In the final translations of the addition circuits from the practical example, the multi-qubit addition circuits are built off from a combination of multiple one-qubit addition circuit translations, and no dependency (or signal) is transferred from one one-qubit addition pattern to another. In other words, each connected one-qubit addition pattern (unit pattern) is independent of the other. Thus, each corresponding measurement subpattern, in the final pattern, can be performed simultaneously in as many rounds as the unit pattern requires. Since circuits can also be broken up into several chunks to meet physical realizability requirements as seen in Section V-A, we will say that the depth of the final algorithm T is such that: $T \geq T_{unit}$, where T_{unit} is the depth of the algorithm corresponding to the unit pattern.

To find a lower bound for T_{unit}, recall that the one-qubit addition circuits are built off from the MAJ, UMA, and UMA_par ones, which are mainly built from negation, CNOT, and Toffoli gates. The one-way negation and CNOT gates are in the Clifford group and do not have any adaptive measurements in their associated patterns. So they can all be computed in a single measurement round. However, in the case of realizations in graph states, the CCZ and Toffoli gates have, respectively, 7 and 8 adaptive measurements, and can thus be computed in at most 8 and 9 rounds (adding the first round composed of all Pauli measurements). However, a careful study of the dependencies in the measurements signals shows that all adaptive measurements (qubits c0, C1, c2,C2, c4, B2, B4 for CCZ, plus C0 for the Toffoli gate) can also be realized in a single round, which brings the lower bound of the depth down to 2. From that, and given the limitations of our simple graphical concatenations, we can imagine that, unless it lives the depth equal to that of the Toffoli, the construction of the MAJ and UMA_par circuits will add an extra round of measurements due to the extra dependency they each incur. Then, the ADD1 (ADD1_par) circuit will have depth equal to the sum of the depths of the MAJ and UMA (UMA_par) circuits — minus 1 for the ADD1_par circuit as there is no extra dependency incurred in concatenating the MAJ and UMA_par gates. Thus a lower bound for T_{unit} is 2, and an upper bound to that lower bound (in the case where the circuit is not divided into chucks) is 5. This is independent of the logical input size n, and thus shows a remarkable efficiency improvement over its network model's correspondent, where the depth is linear in n.

For the realization in cluster states, since we do not know the exact signals of measurements and correction commands, it is safe to provide, to T_{unit}, a lower bound of the number of all adaptive measurements, which is 8. Thus, the double of that number will be a safe upper bound for that lower bound. This is still constant compared to the linear bound in the network model's realizations. Moreover, the flexibility of one-way realization design suggested by the phase-map decomposition could result in an even more efficient one-way realization algorithm.

More importantly, whether we translate the original addition circuit, or its high parallelism version, the resulting one-way realizations for both have a low constant bounded depth, and the difference in extra ancillae is negligible. This suggests that circuit optimizations performed under the network model do not make much of a difference under the one-way one. Indeed, for as long as the optimizations consist of adding, removing, or modifying gates in the Clifford group, they can only be good at modifying the spacial and operational resources accordingly. The only way to affect the final depth is by performing appropriate changes involving non-Clifford operations, such as the Toffoli; but even so, high parallelism is already embedded into the scheme of one-way computation itself.

VIII. Future: A Complete Translation Calculus

Based on the information gathered from our translations, and the analysis of their results, we notice a few limitations on the currently available translation tools: (1) measurement angles, in the measurement calculus, are limited to the X-Y plane of the Bloch sphere, (2) algorithms for running the measurement calculus passes have not been designed, and nor has their efficiency been discussed, and (3) we still need to actually define complete sets of optimization passes. We have addressed those limitations and proposed extensions to the measurement calculus, as well as a new one-way realizations concatenation scheme: the graphical concatenation. We presented illustrations of the simplified version of the scheme, and described how it could be rendered general. We also proposed two particular groupings of optimization passes: The GSR and GSE algorithms. We found indications that realizations resulting from applications of the phase map decomposition algorithm could be optimal. Also, could the "no optimization translation" notices in Section VII become theorems (or rules)?

All these attempts at resolving the limitations of the currently existing schemes call for the establishment of a complete translation calculus. Indeed, this is very possible and will broaden our experimental range, while allowing for flexible navigations between different sub-categories of graph states (e.g. clusters, 2-colorable, etc...). In fact, it could even be extended to allow navigations and analysis across different models of computation. Some work in that direction is already in development [11], providing an extension of the Haskell implementation of superoperators as arrows defined by Vizzotto *et al.* [12]. The idea of implementing such a calculus, then running and analysing simulations raises a range of interesting questions that would be interesting to answer once the complete translation calculus has been defined and analysed:

- Which combination of methods is best?
  We have seen that different choices of universal sets, concatenations, and optimizations will be more appropriate for different situations. Indeed, the implementa-

tion of an efficient algorithm for the complete translation calculus will allow for a high flexibility across different such combinations during experimentations. This could result in a new class of theorems and simplified paradigms of translation, with respect to one-way realizations implementation needs.

- Are results from phase map decompositions optimal?
  In fact, one of the resulting new theorems could specify combinations that produce optimal one-way realizations for specific implementation needs. The very definition of a phase-map decomposition already suggests its production of an optimal realization in general, as one could specify a particular number of ancillae and work a way up until an acceptable solution is found. Independently of how efficient its specified algorithm is, the amount of backtracking involved in deriving the translated realization could be proven to be somewhat alarming. Determining the exact satisfactory point between the efficiency of the resulting realization and that of the applied algorithm is thus subject to future work.
- How well can one efficiently perform a systematic circuit's optimization?
  Another point to notice here is the fact that the suggested translation scheme, after obtaining a concatenated circuit, attempts to better it by performing GSR optimizations first before the GSE ones. The idea is that whether one would like a 2-colorable or a cluster graph at the end, s/he would prefer the smallest possible. That can only happen if we reduce the concatenated circuit first. However, there are cases where performing the GSR+GSE algorithms would result in a graph that is either the same as the original one, or not that much better. Hence, we open the question of how much efficiency, in average or in particular cases, is gained from performing those optimization steps, and of whether that efficiency is worth the overhead of applying the corresponding algorithm.

Meanwhile, in defining the calculus, some questions, related to possible future experiments, still need particular attention:

- Implementing the measurement calculus: A graph theoretical approach?
  When implementing the simple measurement calculus algorithm defined in Section III-B, one may find it necessary to add sanitizing passes that renders all measurement angles positive, or renders them belonging to a particular set such as $\{0, \frac{\pi}{2}, \frac{\pi}{4}\}$, or any other needed reduction. Additionally, during concatenation, one may want to make the task as flexible as possible by not requiring that the output size of the first circuit be the same as the input size of the second. In this case, we may have to reinforce the requirement that tensor products are prioritized, still efficiently, over compositions, without relying on the user too much. It appears that one way to accomplish this is to view patterns concatenations as graphs, where the patterns are nodes, and the compositions are directed arcs from the first to the second circuit. Here, before the composition of two patterns, all patterns (including the first one) at the beginning of all incoming arcs to the second pattern should be tensored together, as well as all outgoing patterns (including the second one) of the second pattern's incoming patterns. Later on, the two tensored patterns containing respectively the two original patterns will be composed instead of the two individual ones. This result will then be more accurate, and will involve less implementation's messiness than in any other way. Again, a clear idea of to an efficient implementation of the measurement calculus is subject to future work.
- Graphical concatenation: to and from pattern representations?
  Once one is given a concatenated (and optimized) pattern description, s/he may want to convert it to the graphical representation and vice-versa, for illustration and visualization purposes. Exactly what tools are needed to accomplish this is yet another open question.
- What about a trial-and-error approach?
  Here, we address the question of facing a general graphical representation, with no (or insufficient) Pauli correction specifications. Maybe we are simply trying out different sets of qubits and measurement patterns and would like to see what happens. Or maybe we are facing a one-way gate such as the cluster-state Toffoli in Figure 15, and would like to verify that it does in fact realize a Toffoli gate, and find out what the corresponding by-product operator U_p is. This opens another range of problems based on whether we know what unitary U is being performed or not, as well as the exact measurement angles and signals.
  It turns out that if we know what U is, then we could analyse the circuit's correlations as in Section III-A, and derive the U_p. However, that relies on Equation 10, which assumes that logical input qubits are measured in X. Nevertheless, an extended paradigm could perform Hadamard transforms on those inputs that are not measured in X first, before the analysis, and then derive U_p accordingly. An alternative is to find a flow f corresponding to the given pattern, as described by Beaudrap *et al.*, and then derive a corresponding complete pattern — that includes the complete definition of correction patterns this time — according to the following equation:

$$P_{f,G,\alpha} = \prod_{i \in O^c}^{\preceq} \left(X_{f(i)}^{s_i} \prod_{k \sim f(i), k \neq i} Z_k^{s_i} M_i^{\alpha_i} \right) E_G N_{I^c}$$

  where G is the associate graph — with inputs I and outputs O, α the set measurement angles, and $\preceq$ the partial order associated with the flow f [6]. Furthermore, if one does not know which unitary was performed to start with, we could derive the unitary embedding as well this way:

$$U_{G,I,O\alpha} \;=\; 2^{|O^c|/2}\left(\prod_{i\in O^c}\langle +_{\alpha_i}|_i\right) E_G N_{I^c}$$

A brute force approach, likely impracticable on large circuits, to finding the unitary embedding would be to analyse all branches of the pattern, which are defined by a permutation of the measurement outcome values (0 or 1) for each measured qubit, and derive a possible (U, U_p) couple [6].

ACKNOWLEDGEMENTS

We would like to thank all direct and indirect contributors to this work. We especially appreciated fruitful inputs from a number of people including Dirk Schlingemann and the rest of the Quantum Information Unit group at the Institute for Scientific Interchange.

REFERENCES

[1] Cuccaro, S., T. Draper, S. Kutin, D. Moulton. 2006. *A new quantum ripple-carry addition circuit*. arxiv:quant-ph/0410184 v1 22 Oct 2004.
[2] Browne, D., H. Briegel. 2006. *One-way Quantum Computation*. arxiv:quant-ph/0603226 v2 3 Oct 2006.
[3] Raussendorf, R., H. Briegel, 2000. *A One-Way Quantum Computer*. DOI: 10.1103/PhysRevLett.86.5188.
[4] Browne, D., R. Raussendorf, H. Briegel. 2000-01. *The One-Way Quantum Computer: Computation from Correlations*. [www.qunat.org/quoxic/talks/danbrowne.ppt]
[5] Raussendorf, R., D. Browne, H. Briegel. 2006. *Measurement-based quantum computation on cluster states*. arxiv:quant-ph/0301052 v2 14 Jan 2003.
[6] Beaudrap, N., Danos V., E. Kashefi. *Phase-map decomposition for unitaries.* arXiv:quant-ph/0603266 v1 29 Mar 2006.
[7] Danos, V., E. Kashefi, P. Panangaden. 2005. *Parsimonious and robust representation of unitaries in the one-way model.*
[8] Danos, V., E. Kashefi, P. Panangaden.*The measurement Calculus.* arXiv:quant-ph/0412135 v1 17 Dec 2004.
[9] Schlingemann, D. *Cluster states, algorithms and graphs*. arXiv:quant-ph/0305170 v2 4 Aug 2003.
[10] Schlingemann, D. *Logical network implementation for cluster states and graph codes*. arXiv:quant-ph/0202007 v1 1 Feb 2002.
[11] Voufo, L., 2007. Haskell Simulations of Quantum Computation. http://www.cs.indiana.edu/~lvoufo/qpl_doc/.
[12] Vizzotto, J. K., T. Altenkirch, A. Sabry. 2007. *Structuring quantum effects: superoperators as arrows*. arXiv:quant-ph/0501151 v1 25 Jan 2005.
[13] Vedral., V., A. Barenco, A. Ekert. *Quantum Networks for Elementary Arithmetic Operations*. arXiv:quant-ph/9511018 v1 16 Nov 1995.
[14] Draper, T. G., S. A. Kutin, E. M. Rains, K. M. Svore. 2006. *A Logarithmic-Depth Quantum Carry-Lookahead Adder*. arXiv:quant-ph/0406142 v1 20 Jun 2004.
[15] Drapper, T.G. 1998. *Addition on a Quantum Computer*. arXiv:quant-ph/0008033 v1 7 Aug 2000.
[16] Bennet, C. H. 1973. *Logical Reversibility of Computation*.
[17] Hein, M., J. Eisert, H. J. Briegel. 2006. *Multi-party entanglement in graph states*. arxiv:quant-ph/0307130 v7 9 Aug 2005