Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 270 (2) (2011) 155–161
www.elsevier.com/locate/entcs

Programmable Hamiltonian for One-way Patterns

S. Salek[a] F. Seifan[a] E. Kashefi[b]

[a] *Faculty of Mathematics, Statistics and Computer Science, University of Tehran, Iran*

[b] *School of Informatics, University of Edinburgh, UK*

Abstract

We construct a family of time-independent Hamiltonians which are able to perform universally programmable quantum computation. The construction is obtained via direct translation of one-way computer assembly language code into a Hamiltonian evolution. We also present how to evolve adiabatically to this Hamiltonian. It is hoped that this approach contributes further into the study of the structural relationship between measurement-based and adiabatic models of quantum computing.

Keywords: Hamiltonian evolution, adiabatic evolution, universal quantum computer, quantum circuit model

1 Introduction

After Feynman's proposal for quantum computers different models have been designed to perform universal quantum computation, with the most widely used one being the quantum circuit model (QC) [3]. Recently, distinctly different models have emerged, namely adiabatic quantum computing (AQC) [2,4], and measurement-based quantum computing (MBQC) [5,6]. These new models suggest different architectures, and fault tolerant schemes, and provide specific approaches to new applications and algorithms, and specific means to compare classical and quantum computation.

The central question that this paper aims to make a progress upon is the study of the structural relationship between MBQC and AQC. Several methods for the adiabatic simulation of a given circuit have been already proposed [7,8]. These methods are essentially based on rewriting a circuit into rounds of computation with few gates in each round, which are then used to construct a corresponding local Hamiltonian. In another approach adiabatic evolution has been exploited to make the MBQC computation more robust [9]. Our construction differs from the previously known results in presenting a programmable Hamiltonian construction

doi:10.1016/j.entcs.2011.01.029

which leads to an Adiabatic evolution. A programmable model is one that includes 'program' and 'data' regions and the state of the program region determines which operations are to be applied to the data register. Hence, instead of hard wiring for every particular problem, we can have a programmable model and just initialise the state of the program region through software for different problems. Furthermore we achieve locality for our construction by the usage of geometric clock. Our approach can be also easily adapted to inherit the intrinsic parallel structure of the MBQC, however there would be a trade-off between parallelism and the required dimension of the physical system to implement the construction.

The Feynman Hamiltonian [10] was the first construction which considered two registers for the computer, one for the data and the other for what he called *cursor*:

$$H_f = \frac{1}{\sqrt{T+1}} \sum_{t=1}^{T} U_t \sigma_t^+ \sigma_{t-1}^- + U_t^\dagger \sigma_{t-1}^+ \sigma_t^- \tag{1}$$

where T is the number of unitary transformations to be applied, σ_k^+ and σ_k^- are raising and lowering operators on the kth cursor qubit out of $T+1$ ones. Later, Kitaev constructed a similar Hamiltonian, to encode quantum circuits into Hamiltonian interaction [11]. He considered a system with two registers, clock register and work register. Kitaev's Hamiltonian is a sum of three terms,

$$H_{\text{kitaev}} = H_{\text{prop}} + H_{\text{out}} + H_{\text{input}} \tag{2}$$

The ground state of the first term is the uniform superposition over the history state and checks the correct propagation of the computation, second term ensures that H_{kitaev} can have low eigenvalue only for the input states which provide "yes" answer on the output of corresponding circuit. Finally the last term checks the correct initialisation of the ancilla qubits. After Kitaev, other people constructed different Hamiltonians with different properties, like the one which acts on a line [12].

While these time-independent Hamiltonians are universal for quantum computation, one can also simulate any quantum circuit with a time-dependent Hamiltonian by adiabatic evolution [2]. In adiabatic quantum computation, we pick two Hamiltonians which act on our system; H_{init} and H_{final}, where the ground state of the first Hamiltonian is easy to construct while the ground state of the final Hamiltonian encodes the solution of the problem. We consider the time-dependent Hamiltonian as $H(s) = (1-s)H_{\text{init}} + sH_{\text{final}}$. The adiabatic theorem states that if for all s, $H(s)$ has a unique ground state, then for any fixed $\delta > 0$, if

$$T \geq \Omega\Big(\frac{|| H_{\text{final}} - H_{\text{init}} ||^{1+\delta}}{\epsilon^\delta \min_{s\in[0,1]}\{\Delta^{2+\delta}(H(s))\}}\Big) \tag{3}$$

then the final state of an adiabatic evolution according to H for time T is ϵ-close in l_2-norm to the ground state of H_{final}.[1] The equivalence of this model with quantum circuits was proved in [13,8].

On the other hand, a relatively different model called one-way quantum computer was introduced by Raussendorf and Briegel [5,6] where its algebraic struc-

[1] The matrix norm is the spectral norm defined as $|| H ||= \max_w || H_w || / || W ||$.

ture was formalised in [1] and a simple assembly language for one-way computation was provided. The basic commands of the language are: 1-qubit preparations N_i that prepares qubit i in state $|+\rangle_i$, 2-qubit entanglement operators $E_{ij} := \wedge Z_{ij}$ (controlled-Z), 1-qubit measurements M_i^α defined by orthogonal projections $|\pm_\alpha\rangle\langle\pm_\alpha|_i$, applied at qubit i, with the convention that $|+_\alpha\rangle\langle+_\alpha|_i$ corresponds to the outcome 0, while $|-_\alpha\rangle\langle-_\alpha|_i$ corresponds to 1, and 1-qubit Pauli corrections X_i, S_i, where i, j represent the qubits on which each of these operations apply, and α is a parameter in $[0, 2\pi)$. Qubits are measured at most once, therefore we may represent unambiguously the outcome of the measurement done at qubit j by s_j. Dependent corrections, used to control non-determinism, will be written $X_i^{s_j}$ and $Z_i^{s_j}$, with $X_i^0 = Z_i^0 = I$, $X_i^1 = X_i$, and $Z_i^1 = S_i$. A *measurement pattern*, or simply a pattern, is defined by the choice of V a finite set of qubits, two possibly overlapping subsets I and O determining the pattern inputs and outputs, and a finite sequence of commands acting on V.

One can count two of the main advantages of the one-way quantum computation as having an intrinsic parallel structure [14] and a rich graph theoretical toolkits for algorithm design [15,16,17]. Generally speaking, a model to have these properties beside the advantage of having a programmable and robust continuous-time evolution is the main goal of this paper. The first step is a translation of one-way pattern into Hamiltonian evolution, which is presented next.

2 Hamiltonian Construction

Here we wish to construct a Hamiltonian that is capable of doing a universal quantum computation which can be programmed by the one-way assembly language. Our Hamiltonian is similar to the construction in [18], but the underlying assembly language is different as it is based on the one-way patterns.

As said before, one-way quantum computation is performed by entangling operator, Pauli corrections and single qubit measurements with angle α. However since the measurements cannot be performed continuously, we have to rewrite one-way patterns where measurements are implemented coherently [19], to be able to construct a Hamiltonian and evolve it continuously. Next, we replace measurements and their dependent correction with a controlled gate of the state, correspondent to the angle of the measurement, tensored with the desired Pauli correction.

$$X_j^{s_i} M_i^\alpha \to \wedge X^\alpha := |-_\alpha\rangle\langle-_\alpha|_i \otimes X + |+_\alpha\rangle\langle+_\alpha|_i \otimes I$$

The nearest neighbour swap gate will be added to our commands list in order to construct the nearest neighbour Hamiltonian interaction. In another words we can enforce our commands to act only on the nearest neighbour using several swap gates. Therefor we have a new assembly language for the one-way patterns with commands sequence being

$$\{S_i, \wedge X_i^\alpha, E_i, I_i\} \tag{4}$$

where a command A_i acts on the pair of qubits $(i, i+1)$ and I_i stands for the identity

operator which will be used to help the pointer reach its correct place on time (see below).

In [18] the authors were interested in constructing a Hamiltonian that acts on one dimension and hence they had to consider extra data qubits and prepare them in $|0\rangle$ state to keep them out of the effect of the gates, which in their construction are normal controlled gate or swap or identity. However, in our construction we use a generalised control gate, so $|0\rangle$ state is not suitable for our case. Therefore we distinct the program and the data registers physically and prepare the data qubits in $|+\rangle$ state, just like the qubits in a conventional one-way pattern (we address later the arbitrary inputs case). Finally, to implement the notion of the geometric clock which links the discrete time steps of the one-way computation and the continuous evolution of the Hamiltonian, we add to our commands sequence the pointer symbol $\triangleright$ and the empty symbol $\bullet$, where their action is defined below [18].

- **Rule 1:** Symbols in the program register can move one step to the left, if there is an empty symbol in that position.
- **Rule 2:** When a command meets a pointer symbol, they are swapped in the program register and the two qubits beneath them are affected by the command.

To program our computer, first we need to initialise our program register in a certain style. The length of program register is $L = 2M = 2KN$ where K is the number of command sequences, M is the number of commands apart from $\triangleright$ and $\bullet$ and N is the number of data qubits. We have to initialise the left half of the program register with K pointers at positions kN for $k = 1, \cdots, K$ and empty symbols elsewhere. The right half holds the command sequences with an identity in front of each of them. A simple example has been given in the appendix.

Now we can construct a Hamiltonian for universal quantum computation with the same insight as started by Feynman, *i.e.* a Hamiltonian such that its ground state is the superposition over the history states:

$$H_{EMC} = -\sum_{j=1}^{L-1}(R + R^{\dagger})_{(j,j+1)} \tag{5}$$

where R corresponds to the rules defined above

$$R = \sum_{A\in\{S,\wedge X^{\alpha},E,I\}} |A\ \bullet\rangle\langle\bullet\ A|_{p_1,p_2} \otimes I_{d_1,d_2} + |A\ \triangleright\rangle\langle\triangleright\ A|_{p_1,p_2} \otimes A_{d_1,d_2} \tag{6}$$

where p stands for the program register and d for the data register of the respective program command. The final step is the design of an adiabatic algorithm to prepare the ground state of H_{EMC} which is the topic of the next section.

We finish this part with a comment on the parallelism. In our construction to achieve the same depth complexity of the one-way pattern, we could first dispose all the Z dependency of the measurements using the signal shifting scheme introduced in [1] and then parallelise the resulting controlled-gates using the techniques introduced in [14]. However this will effect the dimension of the program register. The study of structural links between locality and parallelism is outside the scope of this paper and demands a further investigation.

3 Adiabatic Evolution of H_{EMC}

In this part we will show how to obtain adiabatic quantum computation based on our architecture. We are interested in making an interpolation between the initial and the final configurations. However, our initial and final Hamiltonian must be different from the non-programmable models in a sense that our Hamiltonians must prepare the states of program and data registers in the same time, unlike the previous works in which one had to only prepare the input data state.

In what follows $|\phi_i\rangle$ is the state of the system at step i. For the initial Hamiltonian we penalise all configurations that are not desired as an initial state:

$$H_{\text{init}} := I - \sum_{i=1}^{K} \left|\begin{smallmatrix}\triangleright\\ \text{null}\end{smallmatrix}\right\rangle\left\langle\begin{smallmatrix}\triangleright\\ \text{null}\end{smallmatrix}\right|_{Ni} - \sum_{j=0}^{K-1}\sum_{i=jN+1}^{(j+1)N-1} \left|\begin{smallmatrix}\bullet\\ \text{null}\end{smallmatrix}\right\rangle\left\langle\begin{smallmatrix}\bullet\\ \text{null}\end{smallmatrix}\right|_i \tag{7}$$
$$- \sum_{i=KN+1}^{(K+1)N} \left|\begin{smallmatrix}A\\ +\end{smallmatrix}\right\rangle\left\langle\begin{smallmatrix}A\\ +\end{smallmatrix}\right|_i - \sum_{i=(K+1)N+1}^{L} \left|\begin{smallmatrix}A\\ \text{null}\end{smallmatrix}\right\rangle\left\langle\begin{smallmatrix}A\\ \text{null}\end{smallmatrix}\right|_i$$

where N, K and L are the same as defined in the last section, and all the qubits are prepared in $|+\rangle$. This ensures that the ground state is precisely $|\phi_0\rangle$.

To define the final Hamiltonian, as discussed in [13], we need to have a Hamiltonian whose ground state is a sum over history states. Instead of using directly H_{EMC} which might lead to some degenerate cases we add penalty terms to our H_{EMC} which prefers the transition elements.

$$H_{\text{final}} := \sum_{i=1}^{L-1} |\phi_i\rangle\langle\phi_i| - \frac{1}{2}\sum_{i=0}^{L} |\phi_t\rangle\langle\phi_{t-1}| + |\phi_{t-1}\rangle\langle\phi_t| + \frac{1}{2}|\phi_0\rangle\langle\phi_0| + \frac{1}{2}|\phi_L\rangle\langle\phi_L| \tag{8}$$

The proof in [13], that improved by [8], shows that the spectral gap of these Hamiltonians is inverse polynomial in the number of commands. Therefore, the interpolation between our initial and final Hamiltonian can be performed adiabatically.

4 Conclusion

In this paper we have constructed a local Hamiltonian which could be programmed through direct translation of one-way patterns based on the coherent implementation of the measurements. We have also showed that the ground state of this Hamiltonian could be obtained by adiabatic evolution from an easy to construct initial Hamiltonian. Our Hamiltonians are different from the previous adiabatic proofs in a sense that our construction has to prepare both data and program registers in our desired state to keep our construction both adiabatic and programmable. The locality of our Hamiltonian and the ability to perform adiabatic computation to our final state will give us more robustness against decoherence. Moreover the notion of assembly language from the one-way model gives us the ability of standardisation and having more parallel structures with graph theoretical toolkit. There are still many questions to be asked about the hybrid architecture. We will seek the ways to use real measurements besides keeping the system in its ground state, which is

seemingly incompatible however one special approach has been already investigated in [9].

Acknowledgement

Authors wish to thank Afsoon Ebrahimi for her useful discussions.

References

[1] V. Danos, E. Kashefi, and P. Panangaden. The measurement calculus. *Journal of ACM*, 54, 2007.

[2] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. quant-ph/0001106.

[3] D. Deutsch. Quantum computational networks. *Proc. Roy. Soc. Lond A*, 425, 1989.

[4] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 2001.

[5] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86, 2001.

[6] R. Raussendorf and H. J. Briegel. Computational model underlying the one-way quantum computer. *Quantum Information & Computation*, 2, 2002. quant-ph/0108067.

[7] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM JOURNAL OF COMPUTING*, 37, 2007.

[8] P. Deift, M. B. Ruskai, and W. Spitzer. Improved gap estimates for simulating quantum circuits by adiabatic evolution. *Quantum Information Processing*, 6, 2007.

[9] G. K. Brennen and A. Miyake. Measurement-based quantum computer in the gapped ground state of a two-body hamiltonian. *Physical Review Letter*, 101, 2008.

[10] R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16, 1986.

[11] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. Graduate Studies in Mathematics. American Mathematical Society, 2002.

[12] D. Aharonov, D. Gottesman, S. Irani, and J. Kempe. The power of quantum systems on a line. In *Proceedings of FOCS'07 – Symposium on Foundations of Computer Science*. LNCS, 2007.

[13] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. In *Proceedings of FOCS'04 – Symposium on Foundations of Computer Science*. LNCS, 2004.

[14] A. Broadbent and E. Kashefi. On parallelizing quantum circuits. *Theoretical Computer Science*, 2009.

[15] V. Danos and E. Kashefi. Determinism in the one-way model. *Physical Review A*, 2006.

[16] D. E. Browne, E. Kashefi, M. Mhalla, and S. Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9, 2007.

[17] N. de Beaudrap, V. Danos, E. Kashefi, and M. Roetteler. Quadratic form expansions for unitaries. In *Theory of Quantum Computation, Communication, and Cryptography Third Workshop, TQC 2008 Tokyo, Japan*, number 5106 in Lecture Notes in Computer Science, 2008.

[18] D. Nagaj and P. Wocjan. Hamiltonian quantum cellular automata in 1d. *Physical Review A*, 78, 2008.

[19] R. B. Griffiths and C. Niu. Semiclassical Fourier transform for quantum computation. *Physical Review Letters*, 76, 1996.

Appendix

The following sequence of the configurations shows how to initialise and execute a program that is obtained from the one-way pattern $X_2^{s_1} M_1^0 E_{12}$.