Adiabatic Quantum Computation on a 2D Grid

Amir Kamil
Computer Science Division, University of California, Berkeley
kamil@cs.berkeley.edu

May 20, 2007

When adiabatic quantum computation [2] was first proposed, it was unclear how powerful it was. Aharanov, van Dam, Kempe, Landau, Lloyd, and Regev eventually showed that it could simulate an arbitrary quantum circuit in polynomial time, and furthermore, that the simulation could be done on a two-dimensional grid of six-state particles [1]. In this report, we outline their procedure for adiabatically simulating quantum circuits on a grid.

1 Setup

We assume we are given an input quantum circuit on n qubits consisting of $O(poly(n))$ single-qubit and two-qubit gates.

We start by rewriting the circuit so that it consists of R rounds, each of which has the form in Figure 1. A round has a downward stage and an upward stage. In the *downward* stage, there are n gates, the first of which applies only to the first qubit, and the ith of which applies to qubits i and $i+1$. The *upward* stage has n identity gates, applied in succession from the last qubit to the first.

Each round can simulate at least one gate from the original circuit, so at most $O(poly(n))$ rounds are necessary. Since each round contains $2n$ gates, the rewritten circuit has $O(poly(n))$ total gates.

We now construct a two-dimensional grid of six-state quantum particles, with n rows and $R+1$ columns, as in each side of Figure 2.

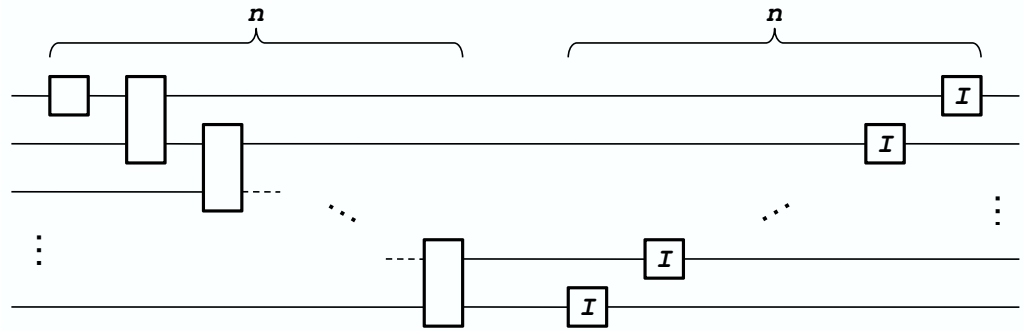

Figure 1: One round of the rewritten circuit.

2 The State of the System

The six states of a particle consist of the following:

- *unborn*: This is denoted by a blank ( ). A particle in this state has not been used yet.
- *first phase*: This is denoted by either an up arrow ($\uparrow$) or a down arrow ($\downarrow$), corresponding to logical states 0 and 1. A bidirectional arrow ($\updownarrow$) represents a particle in an arbitrary superposition of the two states.
- *second phase*: This is denoted by either a double up arrow ($\Uparrow$) or a double down arrow ($\Downarrow$), corresponding to logical states 0 and 1. A bidirectional double arrow ($\Updownarrow$) represents a particle in an arbitrary superposition of the two states.
- *dead*: This is denoted by a cross ($\times$) and corresponds to a particle that has been but is no longer in use.

At any point in time, only one particle in each row can be *active*, meaning either in the first or second phase, and the rest must be unborn or dead.

The state of the system consists of two pieces, the computational state and the clock. The *computational state* is the state of the n qubits in the rewritten circuit, and it is represented by the directionality of the n active particles. The *clock* denotes how many gates of the circuit have been applied. The positions and phases of the active particles represent the clock, as follows:

- In the downward stage of a round r, the clock must be $l = 2nr + k$, where $0 \leq k < n$. At this step, all particles in the r leftmost columns are dead, all particles in column $r+1$ are active, and the rest of the particles are unborn. The top k particles in the active column are in the second phase, while the bottom $n-k$ are in the first phase.
- In the upward stage of a round r, the clock must be $l = 2nr + n + k$, where $0 \leq k < n$. At this step, all particles in the r leftmost columns are dead. The top $n-k$ particles in column $r+1$ are in the second phase, while the rest are dead. The bottom k particles in column $r+2$ are in the first phase, and the rest are unborn. All particles in the remaining columns are unborn.

Both sides of Figure 2 are in the downward stage, and both sides of Figure 3 are in the upward stage.

The initial state of the system has the particles in the first column in the first phase, with their logical state corresponding to the input to the circuit, and the rest of the particles unborn. This corresponds to a clock of $l = 0$. In the final state of the system, all but the particles in the last column are dead, and the last column is in the first phase. The logical state of the last column is the output of the the circuit. The final state corresponds to a clock of $l = 2nR$.

3 Performing a Computational Step

In order to perform a computational step on the particle grid, both the computational state and the clock must be changed. This is done as follows:

- In the downward stage of a round r, where the clock is $l = 2nr + k$ for $0 \leq k < n$, the next gate U_{l+1} to be applied acts on the qubits k and $k+1$ from the top. (For the special case $k = 0$, it applies only on the first qubit.) The clock advances by transferring particle $k+1$ in column $r+1$ from the first phase to the second phase. Thus, to perform the next computational step, we must apply a two-local transformation that simultaneously applies U_{l+1} to the logical state of particles k and $k+1$ in column $r+1$ and moves particle $k+1$ into the second phase, as shown in Figure 2.

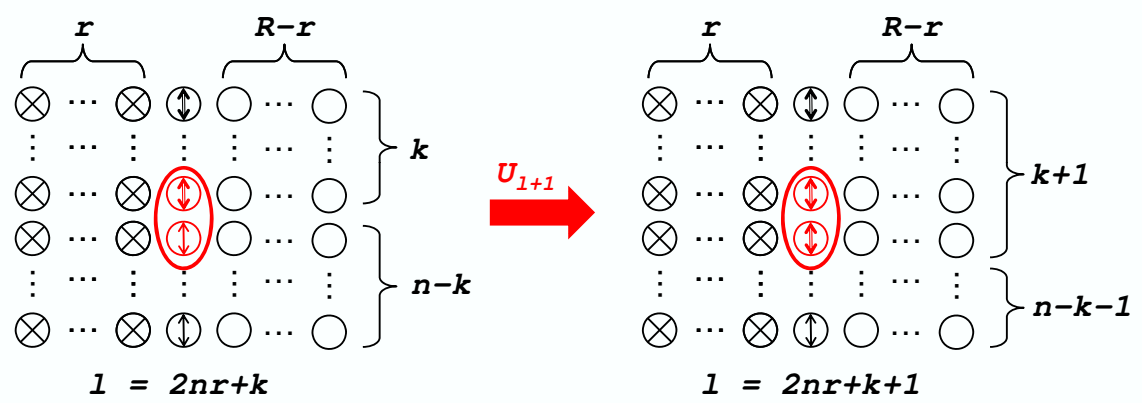

Figure 2: Application of a gate in the downward stage of a round.

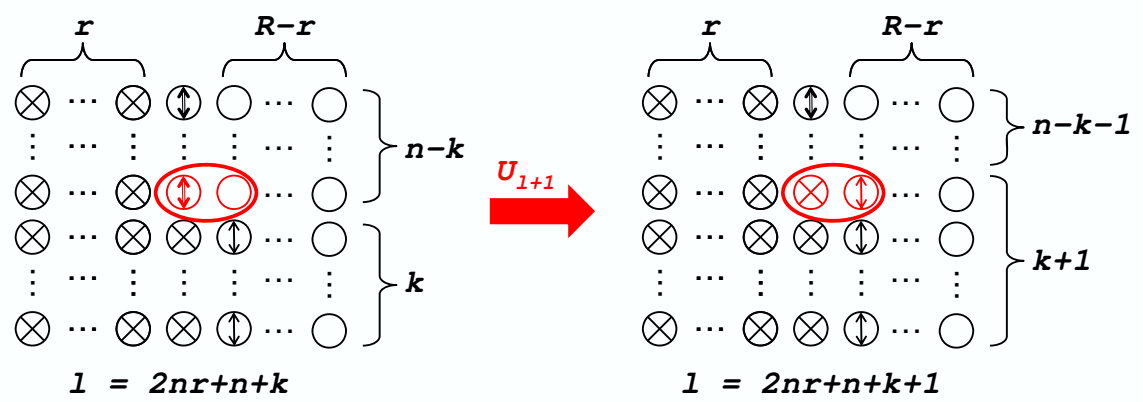

Figure 3: Application of a gate in the upward stage of a round.

- In the upward stage of a round r, where the clock is $l = 2nr + n + k$ for $0 \leq k < n$, the next gate U_{l+1} to be applied is the identity gate on qubit $k+1$ from the bottom. The clock advances by transferring particle $k+1$ in column $r+1$ from the second phase to the dead phase, and particle $k+1$ in column $r+2$ from the unborn phase to the first phase. Thus, to perform the next computational step, we must apply a two-local transformation that simultaneously updates the phases of particle $k+1$ in columns $r+1$ and $r+2$ while moving the logical state of the first particle to the second, as shown in Figure 3.

In both cases, updating the computational state and clock can be done by a single two-local operation.

4 The Adiabatic Simulation

Adiabatically simulating the above computation requires defining an initial and final Hamiltonian, where the initial Hamiltonian has a simple ground state and the output of the original circuit can be extracted from the ground state of the final Hamiltonian. This can be done by using a modified form of Kitaev's Hamiltonian for QMA-completeness [3] as the final Hamiltonian.

Without loss of generality, we assume that the input to the original quantum circuit is $|0^n\rangle$. Then we need an initial Hamiltonian that has as its sole ground state a grid configuration with $|0^n\rangle$ as its computational state and 0 as its clock.

We can do so by writing the initial Hamiltonian as the sum of three simpler Hamiltonians,

$$H_{init} = H_{clockinit} + H_{input} + J \cdot H_{clock}.$$

H_{clock} assigns an energy penalty to a grid state that does not correspond to a legal clock state. For example, it is illegal to have an unborn particle to the left of a particle that is not unborn, or an unborn particle above a dead particle. Any illegal clock state violates some two-local rule of this nature, and H_{clock} assigns an energy penalty for any such violation.

$H_{clockinit}$ and H_{input} check that the initial clock is 0 and that the computational input is $|0^n\rangle$, respectively. Both checks can be done locally on the individual particles in the first column, so these terms are one-local. Thus, H_{init} as a whole is two-local.

The final Hamiltonian is also the sum of simpler Hamiltonians,

$$H_{final} = \frac{1}{2}\sum_{l=1}^{2nR} H_l + H_{input} + J \cdot H_{clock}.$$

The term H_l checks that the propagation from step $l-1$ to l is correct by making sure that the logical states and the phases of the (at most) two particles involved in the step are correctly updated. This check is two-local, so the final Hamiltonian is also two-local.

The final Hamiltonian differs from Kitaev's Hamiltonian in the H_l terms and the multiplicative factor J on H_{clock}. In Kitaev's Hamiltonian, the term corresponding to H_l also checks to make sure that the lth operation occurs in the proper sequence, after $l-1$ but before $l+1$. However, this cannot be done by a two-local check, so it is omitted from H_l. In order to compensate, a much higher penalty J must be assigned to a state with an illegal clock.

The ground state of the final Hamiltonian is a uniform superposition over all computational steps,

$$|\eta\rangle = \frac{1}{\sqrt{2nR+1}}\sum_{l=0}^{2nR} |\gamma(l)\rangle,$$

where $\gamma(l)$ is the state of the grid after step l. The output of the original circuit can thus be obtained by measuring the clock of the system, and if it is $l = 2nR$, examining the computational state. Since $R \in O(poly(n))$, the entire procedure need only be repeated $O(poly(n))$ times to ensure success with high probability[1]. Since the procedure itself takes time in $O(poly(n))$, the total amount of time is in $O(poly(n))$.

References

[1] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 42–51, Washington, DC, USA, 2004. IEEE Computer Society.

[2] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution, 2000. arXiv:quant-ph/0001106v1.

[3] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*. Number 47 in Graduate Series in Mathematics. AMS, Providence, RI, 2002.

[1] It is actually more efficient to pad the original circuit with identity gates at the end, as this increases the number of clock steps at which the computational state is equal to the output of the circuit.