On the Effect of Quantum Interaction Distance on Quantum Addition Circuits

BYUNG-SOO CHOI
Ewha Womans University
and
RODNEY VAN METER
Keio University

We investigate the theoretical limits of the effect of the quantum interaction distance on the speed of exact quantum addition circuits. For this study, we exploit graph embedding for quantum circuit analysis. We study a logical mapping of qubits and gates of any \( \Omega(\log n) \)-depth quantum adder circuit for two \( n \)-qubit registers onto the NTC architecture, which limits interaction distance to the nearest neighbors only and supports only one- and two-qubit logical gates. Unfortunately, on the \( k \)-dimensional NTC architecture, we prove that the depth of the quantum adder is no longer \( \Omega(\log n) \) that is possible on an ideal machine, but \( \Omega(\sqrt[3]{n}) \). This result, the first application of graph embedding to quantum circuits and devices, provides a new tool for compiler development, emphasizes the impact of quantum computer architecture, and acts as a cautionary note when evaluating the time performance of quantum algorithms.

1. INTRODUCTION

Quantum computers can show very high speedup compared to classical computers on certain problems. One example is Shor's large number factoring algorithm [Shor 1997], which can factor a large number within polynomial complexity. Grover's database search algorithm [Grover 1996] can help to find the desired item from an unstructured database search space of \( n \) elements in \( O(\sqrt{n}) \) computational steps. Many other quantum algorithms have been proposed recently, and the design of quantum algorithms is an active research area [Mosca 2008; Bacon and van Dam]...
In general, the calculation of the speedup of a quantum algorithm over a classical one is based on an ideal quantum computer model, similar to the Random Access Machine model [Knuth 1998] for classical computing. Hence, we can say that the quantum speedup in the literature is the “best case” performance improvement. However, if we consider the real physical constraints on the practical quantum model, the quantum speedup may decrease, especially when considering the circuit depth (time performance). Related to this, some upper bounds of the quantum arithmetic circuits from the ideal model to the specific and practical models have been investigated [Cheung et al. 2007]. On the other hand, very little work has been done to analyze the lower bounds. In this work, we focus on the lower bounds when practical constraints are accounted in the quantum circuit.

While we are interested in the general problem of hardware/software co-design for quantum computers, we focus here on the problem of addition for two \( n \)-qubit numbers. Addition is a well-defined problem, making direct comparison of competing solutions straightforward. It is also a fundamental building block for important applications, such as Shor’s factoring algorithm. For the ideal quantum computer model, such as an arbitrary concurrent (AC) architecture, several types of circuit with a depth of \( O(\log n) \) have been designed [Draper et al. 2004; Van Meter and Itoh 2005].

In this study, we establish a quantum depth lower bound for adders when the quantum interaction distance is only one, i.e., the nearest-neighbor, two-qubit, and concurrent execution (NTC) architecture is used. Arithmetic circuits are large and complex unitary transforms, usually decomposed into circuits of one-, two- and three-qubit quantum gates. If the target and source qubits of e.g. a controlled-NOT gate are not neighbors, the target or source qubit must be transported to a neighboring position by using SWAP operations, or using a chain of gates, as shown in Sec. 2.2. Therefore, on a practical quantum computer, a number of SWAP operations is necessary to emulate the behavior of a quantum adder running on an ideal machine, increasing the circuit depth. In our study, we consider the \( k \)-dimensional (\( kD \)) NTC architecture. Then our question can be rephrased as determining whether or not an \( \Omega(\log n) \)-depth quantum adder exists for these models.

To investigate the theoretical limit of the ideal depth lower bound on these quantum architectures, we exploit some graph theoretical approaches such as graph embedding, for the first time. We show that any \( \Omega(\log n) \)-depth quantum Boolean circuit on the AC architecture must be modeled as a set of log-depth binary trees (LBTs), where each log-depth binary tree produces one qubit of the sum. Then the question can be rephrased again to ask how much additional depth is required for embedding a log-depth binary tree into a \( kD \) graph having edges between neighboring nodes for the corresponding \( kD \) NTC architecture. In the graph embedding for quantum circuits, the additional depth caused by the necessary SWAP operations is measured by the dilation value. Based on the analysis of dilation, for embedding a log-depth binary tree into the target graph, we find that the theoretical depth lower bound is \( \Omega(\sqrt{n}) \) for the \( kD \) NTC structure. Therefore, there is no \( \Omega(\log n) \)-depth quantum adder on any \( kD \) NTC structure by simple logical mapping because of a
practical limitation, the interaction distance.

This work is organized as follows. Section 2 describes several quantum architectures, quantum Boolean circuits, exact and approximate quantum adders, log-depth binary tree, and graph embedding. Section 3 studies the depth lower bounds of the quantum addition circuits on the target quantum architectures. Section 4 describes how a typical $\Omega(\log n)$-depth adder, the carry lookahead adder, can be mapped to a set of log-depth binary trees. Section 5 discuss some related work and the differences from ours. Section 6 concludes this manuscript with several research questions.

2. BACKGROUND

2.1 Quantum Computer Architectures

Some systems, such as those using “flying qubits” held on photons and measurement-based quantum computing [Raussendorf et al. 2003], allow an approximation of arbitrary-distance interaction. At the other extreme, there is a NTC architecture allowing the nearest neighbor interaction only with one- or two-qubit gates executing concurrently. Since most quantum computer proposals are based on variations of this model, we focus on the NTC model. Depending on the layout of qubits, there are three architectures as follows:

—1D NTC Model: The 1D model, called Linear Nearest Neighbor (LNN), consists of qubits located in a single line. In this model, only two neighboring qubits can interact. Some trapped-ion systems [Häffner et al. 2005] and liquid nuclear magnetic resonance (NMR) [Laforest et al. 2007] technologies are experimental systems based on this model. The original Kane model [Kane 1998] is also based on this model. The effects of the 1D NTC model on performance have been investigated for the quantum Fourier transform [Takahashi et al. 2007; Van Meter 2004] and Shor’s algorithm [Kutin 2007].

—2D NTC Model: The 2D NTC model is a lattice structure where the links are located on a two-dimensional Manhattan grid. In this model, a qubit can interact with four neighboring qubits unless, of course, it is on an edge. Therefore, it can help to reduce the communication cost over the 1D NTC model. Several proposed quantum technologies will correspond to this model, such as the array of trapped ions [Häffner et al. 2005] and Josephson junctions [Helmer et al. 2007; Doucet et al. 2004].

—3D NTC Model: The 3D NTC model is simply a set of 2D lattices stacked in the third dimension. As expected, since a qubit can interact with six neighboring qubits, it has more flexibility than the 2D NTC model. Although it has some advantages over the 2D NTC model, it suffers from the difficulty of controlling 3D qubits using the global classical control system, as well as difficult fabrication. However, some approaches have been proposed based on this model [Pérez-Delgado et al. 2006].

2.2 Long-Distance Quantum Gates

In systems that do not directly support long-distance interactions, we must construct circuits of building blocks using only nearest-neighbor operations. Nearest-neighbor operations can be used in three ways to effect gates between two qubits that are initially stored some distance apart:
—swap one or more of the qubits we wish to interact along a path in the graph
that will bring the qubits together;
—execute logical gates in a chain along a path so that the end result is the desired
gate; or
—use the graph links to create long-distance entanglement (Bell pairs) that can be
used to execute long-distance gates (“telegate”) or to teleport data qubits.

We focus primarily on the first method, but let us briefly examine the other
two. It is well known that a carefully-chosen chain of neighboring gates can act
equivalently to a long-distance gate. For example, on a line of qubits $A, B, C$ with
the notation $\text{CNOT}($control,target$)$ and gates ordered left to right,

\[
\text{CNOT}(A, C) = \\
\text{CNOT}(A, B) \text{CNOT}(B, C) \text{CNOT}(A, B) \text{CNOT}(B, C).
\]

This approach results in identical asymptotic circuit depth and complexity as the
swapping approach, as the gates must be cascaded in an identical fashion. Constant
factors can vary, however, as a result of the usage pattern of the variables and the
gate execution time; in general, the principle of locality [Hennessy and Patterson
2006] suggests moving the variable will be more effective than using the gate chain
method.

A long-distance Bell pair can be created using pairwise entangling gates along
a path in the connectivity graph, measuring the middle qubits, and propagat-
ing a Pauli frame correction to the end points, as is done in quantum repeaters
and measurement-based quantum computation [Dür et al. 1999; Raussendorf et al.
2003]. The quantum operations in this approach can be executed in only two time
steps; however, the classical information will be limited by the speed of signal prop-
gagation in the system. This limitation assumes that non-Clifford group operations
are executed at each end of the movement. For our purpose, this restriction holds,
as addition circuits require non-Clifford group operations. Equally important, this
approach consumes significant spatial resources: the intermediate qubits along the
path cannot hold important data values, as they are measured and discarded.

Browne, Kashefi and Perdrix have recently shown that one-way quantum com-
putation (measurement-based quantum computation) is equivalent in power to
unbounded-fanout circuits [Browne et al. 2009]. Our results are argued using both
the fanout and the computational aspects of the problem.

Thus, the results presented here are restricted: they are not yet shown to apply
to measurement-based quantum computation, they assume that classical signal
propagation is restricted to the same connectivity as the quantum operations, and
the operations of interest before and after data movement must be non-Clifford
group operations.

2.3 Quantum Boolean Circuit

A classical Boolean circuit is a circuit for $n$ inputs with one output. Since the
number of outputs is one and the value of the output is zero or one, sometimes the
classical Boolean circuit can be called a binary decision circuit. As with classical
Boolean circuits, in a quantum Boolean circuit, our goal is to compute a single
output qubit that is a function of the \( n \) input qubits. The final output is stored in the output qubit, and any ancillae may be cleaned by undoing the computation.

2.4 Quantum Addition

In-place addition on a quantum computer performs the transform \( |a, b\rangle \rightarrow |a, a + b\rangle \), where \( |a\rangle \) and \( |b\rangle \) are \( n \)-qubit registers holding binary numbers. If we consider each summation output as a single output qubit, the quantum addition circuit consists of a set of quantum Boolean circuits, one for each output qubit.

Numerous quantum addition algorithms have been proposed, and even implemented at small scales, based on classical addition algorithms. Ripple-carry algorithms include those proposed by Vedral \textit{et al.} \cite{Vedral1996}, Beckman \textit{et al.} \cite{Beckman1996}, and Cuccaro \textit{et al.} \cite{Cuccaro2004}. The depth of ripple-carry adders is linear in the length of the numbers being added, and they typically do not require long-distance interactions. Logarithmic-depth adders, including the carry-lookahead and conditional-sum adders, have been designed using longer-distance operations \cite{Draper2004, VanMeter2005}, assuming the AC abstraction architecture; one of these has been adapted to measurement-based quantum computation \cite{Trisetyarso2009}.

Note that the above adders are exact, rather than approximate, integer adders. That is, we expect that \( |0111...11\rangle + |0000...01\rangle \) will yield the result \( |1000...00\rangle \). However, we can consider non-exact adders. Draper proposed an \( O(\log n) \)-depth adder based on the quantum Fourier transform \cite{Draper2000}. This adder is quite different from the above adders since it is based on a genuine quantum approach, rather than classical techniques. In order to achieve full \( n \)-qubit precision, the depth of Draper’s adder is \( O(n) \).

2.5 Log-depth Binary Tree

A log-depth binary tree is defined as a class of binary tree \cite{Weisstein2010} that has one root, one or two child nodes from each non-leaf node, and all other leaf nodes. In a tree, the depth can be defined as the number of nodes in the longest path from the root to any leaf node. For a log-depth tree, the highest depth must be \( O(\log n) \), when the number of leaves is \( n \). Figure 1 is an example of a log-depth binary tree and its application. Since many digital algorithms are based on binary decisions with one- or two-input gates, the log-depth binary tree is a very useful model. Likewise, many acyclic circuits with one- or two-input gates can be modeled as log-depth binary trees. Therefore, we use the log-depth binary tree for the analysis of arithmetic quantum circuits on the NTC architecture.

2.6 Graph Embedding

Graph embedding is a widely used tool for analyzing the performance of different structures. For example, to analyze a specific network topology for a different architecture, we use graph embedding techniques (see e.g. Diestel \cite{Diestel2005}). A guest graph \( G \) is embedded on a host graph \( H \) when the nodes in \( G \) are mapped to the nodes in \( H \), and the edges in \( G \) are mapped to paths in \( H \). Figure 2 shows an example of embedding a log-depth binary tree (leftmost) into a line graph (rightmost). Each node in \( G \) is mapped to a node in \( H \). The two edges \((1,5)\) and \((4,6)\) in \( G \) cannot be directly mapped to any edge in \( H \), but can be mapped...
Fig. 1. An example of a log-depth binary tree and its application.
(left) The root node 1 has two children, 2(left child) and 3(right child). Likewise, non-leaf nodes 2 and 3 each have two children. The smallest depth of the log-depth binary tree is $\lceil \log n \rceil$, where $n$ is the number of nodes.
(right) A mapping of a 6-bit AND circuit $a_1 \land a_2 \land a_3 \land a_4 \land a_5 \land a_6$ with two-input AND gate is shown. A circuit with two-input gates can be modeled as a log-depth binary tree where the initial input is mapped to the leaf nodes, all two-input gates to the non-leaf nodes, and the final output to the root-node.

to paths, as shown by the dotted lines. Graph embedding has many interesting properties [Keh and Lin 1997]:

— dilation: The dilation is defined as the maximum distance between adjacent nodes in $H$ after embedding. In general, the dilation lower bound is calculated as [Unger 2008]

$$\frac{\text{diameter of the host graph}}{\text{diameter of the guest graph}}.$$  (2)

In this equation, the diameter is defined as the longest path of a graph. Then the lower bound of dilation occurs when there is a best map with the smallest increase of the distance between nodes in the guest graph. To achieve this, we can map the longest path (diameter) of the guest graph to the longest path of the host graph. Then the following mapping of other nodes needs the same or higher distance than the guest graph. Therefore, the lowest ratio of the diameters for the guest and host graphs is the lower bound of the dilation value. For example, as shown in Figure 2, the dilation is two since the edge (1,5) in $G$ must be embedded into a path (1,2)&(2,5) in $H$. Therefore to emulate the interaction between 1 and 5 in $G$, two interactions are required between 1 and 2, and 2 and 5 in $H$.

— expansion: The expansion is defined as the ratio of the number of nodes in $H$ over the number of nodes in $G$.

— load: The load is defined as the maximum number of nodes in $G$ which must be embedded into a node in $H$.

— congestion: The congestion is defined as the maximum number of edges in $G$ which must be embedded into an edge in $H$.

ACM Transactions on Computational Logic, Vol. 2, No. 3, 09 2001.
Fig. 2. Embedding a log-depth binary tree (leftmost) into a line graph (rightmost).
Dotted lines represent the graph dilation. The edges (1,5) and (4,6) in the log-depth binary tree cannot be directly mapped to any edges in the line graph. They are mapped to paths (1,2)&(2,5) and (4,3)&(3,6), respectively.

In our study we consider only the dilation. In circuit complexity [Vollmer 1999], this means that a certain path of the guest graph must increase by a factor of the dilation value. If such a path is the longest path in the guest graph, the circuit depth of the host graph must increase by a factor of the dilation value as well.

3. DEPTH LOWER BOUNDS

First, we need to understand the depth lower bound of quantum Boolean circuits for $n$ inputs when only one- and two-qubit gates are allowed with no limitation of interaction distance.

**Fact 1.** The depth lower bound of an exact quantum Boolean circuit for $n$ inputs is $\Omega(\log n)$ when only one- or two-qubit gates are allowed, without limitation of interaction distance.

**Proof.** We consider the general structure of any $\Omega(\log n)$-depth quantum Boolean circuit. Any $\Omega(\log n)$-depth quantum Boolean circuit can be executed in the following manner. For generating the final output qubit, a two-qubit gate (which we will place at the root of the binary tree) must be applied to two temporary input qubits which are generated at the previous level. These temporary two input qubits are generated from other two-qubit gates with each set of temporary two input qubits which are generated at the previous level. This backtracking must continue until the temporary input qubits are the same as the actual input qubits. We need to calculate how many levels are needed. Since each two-qubit gate needs two inputs, the number of temporary inputs doubles. Hence, if the level of backtracking is $k$, then the number of inputs is $2^k$. Therefore, the minimum level of backtracking...
ing or levels must satisfy $2^k = n$, and hence $k = \log n$. In this manner, we can make any $\Omega(\log n)$-depth quantum Boolean circuit with one- and two-qubit gates, as explained by Cleve and Watrous [Cleve and Watrous 2000, P.532]. □

Next, we need to investigate the graph structure of $\Omega(\log n)$-depth quantum Boolean circuit, where $\Omega(\log n)$ indicates that the circuit is asymptotically bounded above by some constant multiple of $\log n$.

**Theorem 1.** Any $\Omega(\log n)$-depth quantum Boolean circuit can be represented by a log-depth binary tree when only one- or two-qubit gates are allowable with no limitation of interaction distance.

**Proof.** The one- and two-qubit gates in a quantum Boolean circuit can be mapped to non-leaf nodes and the root node in the log-depth binary tree. The root node contains the final output. The actual inputs can be mapped to leaf nodes, respectively. The two inputs for each two-qubit gate can be mapped to the left and the right child nodes for the corresponding parent node. In this manner, we can map any $\Omega(\log n)$-depth quantum Boolean circuit into a log-depth binary tree. Note that the edges in the graph represent the information flow from the child node to the parent node. The time for communication of information between the child and the parent node is ignored in this analysis. □

As an example, a mapping of a quantum Boolean circuit for an 8-qubit PARITY function into a log-depth binary tree is shown in Figure 3. In the first level, four CNOT operations – $\text{CNOT}_{1,0}$, $\text{CNOT}_{1,1}$, $\text{CNOT}_{1,2}$, and $\text{CNOT}_{1,3}$ – are applied to the corresponding qubits. The outputs are stored in $Q_1$, $Q_3$, $Q_5$, and $Q_7$, respectively. In the second level, two CNOT operations – $\text{CNOT}_{2,0}$ and $\text{CNOT}_{2,1}$ – are applied for each corresponding qubits. The results are stored in $Q_3$ and $Q_7$. In the last level, one CNOT operation $\text{CNOT}_{3,0}$ is applied, and the result is stored in $Q_7$. Now we can map this circuit into a log-depth binary tree, as shown in the right part of Figure 3. In the figure, input qubits are mapped to the leaf nodes. The CNOT operations in the circuit are mapped to the non-leaf nodes in the log-depth binary tree. The final output is stored in the root node.

**Theorem 2.** A quantum Boolean circuit for summation output $s_i$ can be mapped to a log-depth binary tree when only one- and two-qubit gates are used without limitation of interaction distance.

**Proof.** A summation output $s_i$ can be generated by an exact quantum Boolean circuit for $n$ inputs since the input carry for $s_i$ position depends on all $a_i$ and $b_i$ where $i \in \{0, \cdots, i - 1\}$. Therefore, the depth lower bound of the quantum Boolean circuit for $s_i$ is $\Omega(\log n)$ by Fact 1. Since a quantum Boolean circuit with $\Omega(\log n)$-depth can be mapped to a log-depth binary tree as shown by Theorem 1, a quantum Boolean circuit for summation output $s_i$ can be mapped to a log-depth binary tree. □

Up to this point, we have discussed a quantum Boolean circuit for $s_i$ and its log-depth binary tree structure. To reduce the overall addition time, each summation output $s_i$ must be generated as fast as possible. For this purpose, the quantum Boolean circuits for all output qubits $s_i$ must be executed in parallel. However, since
On the Effect of Quantum Interaction Distance on Quantum Addition Circuits

Fig. 3. Mapping an $\Omega(\log n)$-depth quantum Boolean circuit (left) for an 8-bit PARITY function into a log-depth binary tree (right). The inputs of the left are mapped to the leaf nodes in the right. Two-qubit gates in the left are mapped to non-leaf nodes in the right. Final output is generated on the root node.

![Diagram](image)

Fig. 4. An input qubit $|a_0\rangle$ is fanout four times in the quantum equivalent of a FANOUT circuit. The depth of the circuit is $\lceil \log n \rceil$, where the number of fanout is $n$.

Each summation output $s_i$ needs to use the inputs $a_j$ and $b_j$, where $j \in \{0, \cdots, i\}$, copies of the inputs $a_j$ and $b_j$ must be prepared for each quantum Boolean circuit for $s_i$, where $k \in \{0, \cdots, j\}$. Therefore, each input $a_j$ and $b_j$ must be fanout for each quantum Boolean circuit for $s_i$.

**FACT 2.** A fanout circuit for a single qubit to $n$ target qubits can be mapped to a log-depth binary tree.

**PROOF.** For example, an input qubit $|a_0\rangle$ can be fanout four times as shown in Figure 4. Since each input qubit $a_i$ and $b_i$ must be fanout at most $n - i$ times, the lower bound of depth of these fanout circuit is $\Omega(\log n)$. □
Now, we want to know the depth lower bound of any quantum addition circuit when only one- and two-qubit gates are allowed with no limitation of interaction distance.

**Theorem 3.** On a quantum computer architecture of limited gate width $g$, meaning one- to $g$-qubit gates are allowed, no quantum adder can be asymptotically faster than one composed of a set of quantum Boolean circuits, where each circuit can be mapped to a log-depth binary tree, respectively.

**Proof.** This theorem is founded on the information flow in an addition circuit. First, we discuss the case of the gate width $g = 2$, meaning only one- and two-qubit gates are allowed. To minimize the depth of the circuit, we need to maximize the parallelism in the circuit. Hence, we can consider the overall circuit to consist of $n + 1$ separate quantum Boolean circuits, one to calculate each output qubit (including the final carry out). We call this phase of the circuit the computation part.

The gate width limits the use of each input qubit. Because the $i$-th output qubit depends on all of the input qubits $|a_j\rangle$ and $|b_j\rangle$ for all $j \leq i$, each quantum Boolean circuit for each output must have its own copy of the input qubits before execution, in order to run concurrently. Therefore, we need to construct another circuit for fanout the input qubits, making one copy for each tree. We call this phase the fanout part of the addition circuit.

After the fanout of the input qubits, the $n$ summation and one output carry quantum Boolean circuits are executed in parallel.

Similar arguments follow for any fixed gate width $g > 2$.

Now we consider a graph structure for quantum addition circuit. As we already discussed, the addition consists of two parts: fanout part and computation part. By Theorem 2 and Fact 2, a quantum addition circuit can be mapped to a set of log-depth binary trees.

We observe that this construction is efficient in time, but not in space; $O(n^2)$ physical qubits are required. In practice, both the carry-lookahead and conditional-sum adders (the two known types of $O(\log n)$-depth quantum adders) do not require the full fanout of data, but reuse the input qubits and partial results more efficiently.

However, the proof above shows that no circuit can do better than this construction in the circuit depth.

We have now explained how to map a quantum addition circuit to a set of log-depth binary trees. Next, we show how to embed such log-depth binary tree into a $kD$ mesh structure, which is the graph structure for the $kD$ NTC architecture.

**Fact 3.** A log-depth binary tree can be mapped to a $kD$ mesh with dilation $\Omega(\sqrt[2k]{n})$, hence the depth lower bound of the embedded graph is $\Omega(\sqrt[2k]{n})$.

**Proof.** To understand the effect of graph embedding, we need to calculate the dilation of embedding a guest graph into a host graph. The dilation of a graph mapping is the ratio of the diameters. Formally, the dilation for graph mapping a guest graph to a host graph is calculated by Equation (2) in Section 2.6.

---

1Note this is not quantum cloning, but quantum fanout by using CNOT gates.

ACM Transactions on Computational Logic, Vol. 2, No. 3, 09 2001.
In our study, the guest graph is a log-depth binary tree whose diameter is \( \Omega(\log n) \) since the max distance is between the two leaves where the path passes through the root node. On the other hand, the host graph is a \( kD \) mesh graph whose diameter is \( \Omega(\sqrt{k \log n}) \). Therefore, the dilation of graph embedding from the log-depth binary tree into a \( kD \) mesh graph is \( \Omega(\sqrt{k \log n}) \) as shown by Heckmann et al. [Heckmann et al. 1991]. Finally, the depth of the embedded graph is \( \Omega(\sqrt{k \log n}) \) since the depth of the guest graph increases by the dilation factor.

**Theorem 4.** The depth lower bound of the exact quantum addition circuit on the \( kD \) NTC structure is \( \Omega(\sqrt{k \log n}) \).

**Proof.** By Theorem 3, a depth-optimal quantum adder can be mapped to a set of log-depth binary trees. By Fact 3, a log-depth binary tree can be embedded to a \( kD \) mesh with depth \( \Omega(\sqrt{k \log n}) \). Therefore, a depth-optimal exact quantum addition circuit in the AC architecture can be mapped into the \( kD \) NTC architecture with a depth of \( \Omega(\sqrt{k \log n}) \).

Therefore, there is no \( \Omega(\log n) \)-depth quantum adder on the \( kD \) NTC quantum computer model.

### 4. CASE STUDY: CARRY LOOKAHEAD ADDER

We show how a carry-lookahead adder (CLA) can be mapped to a set of log-depth binary trees as follows. Let us consider the computation part first. Conceptually, the computation part of the CLA works in two steps: 1) find all \( i \)-th carry value concurrently and then 2) generate \( i \)-th summation value concurrently, as shown in Figure 5.

A carry-lookahead adder consists of three networks: generate for \( g_i \), propagate for \( p_i \), and carry-lookahead for \( c_i \). The final results \( s_i = a_i \oplus b_i \oplus c_i \) will be calculated for all \( i \) in parallel.

In the first step, \( g_i = a_i \land b_i \) and \( p_i = a_i \lor b_i \) are generated at the same time for all \( i \). Since \( g_i \) and \( p_i \) depends on \( a_i \) and \( b_i \), there is no information dependency and hence each \( g_i \) and \( p_i \) can be generated at the same time. On the other hand, the carry \( c_{i+1} \) is generated by using carry lookahead logic [Ercegovac and Lang 2003] as follows.

\[
c_{i+1} = g_i + p_i \land c_i
\]

\[
= g_i + p_i \land (g_{i-1} + p_{i-1} \land c_{i-1})
\]

\[
= g_i + p_i \land (g_{i-1} + p_{i-1} \land (g_{i-2} + p_{i-2} \land c_{i-2}))
\]

\[\vdots\]

\[
= g_i + p_i \land (g_{i-1} + p_{i-1} \land (g_{i-2} + p_{i-2} \land (\cdots (g_0 + p_0 \land c_0) \cdots )))
\]

\[
= g_i + p_i \land g_{i-1} + p_i \land p_{i-1} \land g_{i-2} + p_{i-2} \land \cdots + p_i \land p_{i-1} \land p_{i-2} \land \cdots \land p_0 \land c_0.
\]

As the above equation explains, the \( c_i \) depends on \( g_j \) and \( p_j \) where \( j \in \{0, \ldots, i\} \).
Fig. 5. Two steps of addition for a Ω(log n)-depth adder for n qubits.
In the first step, carry values for each position are generated concurrently. This step is log n depth. In the second step, each summation output is generated concurrently by using the corresponding carry value.

Although the final summation \( s_i = a_i \oplus b_i \oplus c_i \) depends on \( a_i, b_i, \) and \( c_i \), the depth is bounded by the circuit for \( c_i \). Therefore, we need to consider the circuit for \( c_i \). From Equation (3), we know that the carry-lookahead logic consists of the summation of products. Therefore, in the first step, each product term must be generated, and then all products must be summed. As a result, we need to map each product into a log-depth binary tree, and the last summation part into another log-depth binary tree.

Let us first consider the product terms. Although there are many products, it is sufficient to consider the worst case \( p_i \wedge p_{i-1} \wedge p_{i-2} \wedge \cdots \wedge p_0 \wedge c_0 \) since other products can be mapped in the same way. This product is generated as the AND function of \( i - 1 \) \( p_i \) values and \( c_1 \). An AND function for \( i \) inputs can be implemented by using a log-depth binary tree with some additional qubits as shown in Figure 6(a).

Since a two-qubit AND gate cannot be implemented directly, we use CCNOT and SWAP gates for it as shown in Figure 6(b). Note this construction needs one ancilla since the two-input AND gate cannot be designed as a unitary gate without using an extra qubit, which increases the overhead. However, this overhead is linear.
On the Effect of Quantum Interaction Distance on Quantum Addition Circuits

Fig. 6. Two-qubit gate implementation of an AND log-depth binary tree. In this example, eight values are ANDed together, with the seven AND gates executed in three time steps.

in this case since the maximum overhead is $\frac{n}{2}$. Although the SWAP operator is not technically necessary, we introduce it in order to have a consistent representation storing the output on one of the input qubits.

Because the NTC architecture allows only two- or one-qubit gates, we must further decompose the CCNOT gate; one such decomposition, using eight two-qubit gates, is shown in Figure 6(c) [Barenco et al. 1995]. Therefore, each AND gate in the log-depth binary tree can be implemented by eight two-qubit gates, which increases the coefficient part of the circuit complexity. Note that the ancilla can be initialized again after completing the whole addition by uncomputing in the usual fashion [Bennet 1973]. In this way, we can generate each log-depth binary tree for each product.

Now let us consider the final summation of products. The summation in the Boolean function requires an OR function, and the structure of OR is almost same as the AND function. Hence, we can generate another log-depth binary tree for this summation circuit in the same way.

Thus, we can find each log-depth binary tree for generating the $i$-th carry value. Then, the final output can be generated by using this value with other $a_i$ and $b_i$ values. In this manner, we finally can find a set of log-depth binary trees for each output.

As we discussed in the previous section, the CLA needs another circuit for fanout of inputs for parallel computation for $c_i$. The necessary log-depth binary tree can be built using the fanout circuit shown in Figure 4.
Therefore, for fanout of inputs and for computation of outputs, we can find a set of log-depth binary trees.

5. RELATED WORK

Various researchers have investigated specific circuit implementations for the linear nearest neighbor architecture. However, the theoretical bounds have not been investigated in depth.

Möttönen and Vartiainen studied the decomposition of a uniformly controlled gate [Möttönen and Vartiainen 2006]. They investigated the decomposition of a uniformly controlled gate into one-qubit gates and CNOT gates. They also investigated the effect of interaction distance on the control and target qubits on the nearest neighbor architecture. They shown that the number of gates for one-qubit and CNOTs does not dramatically increase. Similarly, Shende et al. [Shende et al. 2006] also studied the synthesis of quantum-logic circuits. They proposed quantum multiplexor circuits which are elementary circuits for synthesizing a given \( n \)-qubit circuit. They investigated the overhead when the architecture is limited to a linear nearest neighbor architecture, and showed that the LNN architecture increases the depth by a constant factor of nine times over the generic case. Note that the limitation of interaction distance causes some overhead of the total gate complexity in these works because they focused on the general case. In our case, the focus is on the special case of addition.

Maslov investigated circuits for the quantum Fourier transform and the stabilizer code in the LNN architecture [Maslov 2007] as did Takahashi [Takahashi et al. 2007]. Maslov showed that these circuits can be mapped to LNN architecture with linear depth because of interaction distance. Maslov et al. [Maslov et al. 2008] investigated the technical mapping of logical qubits to the physical qubits. Since the mapping of logical qubits to the physical qubits affects the quantum gate time, very similar to interaction distance, they shown that the overall computation time heavily depends on the qubit mapping, a problem they called the quantum circuit placement in their paper.

6. CONCLUSION AND OPEN PROBLEMS

We have investigated the effect of the allowed quantum interaction distance on the performance of arithmetic circuits. Since the proposed quantum addition circuits such as the carry-lookahead adder have no limitation on the allowed quantum interaction distance, the depth lower bound shown in some previous papers is near to the ideal limit of \( O(\log n) \). However, as we have shown in this work, when the quantum interaction distance is one, the quantum addition circuit must use a number of SWAP operations. Unfortunately, some of the SWAP operations will be in the longest path in the circuit, and hence will increase the depth lower bound.

While this restriction has been recognized in practical terms in some other papers [Van Meter and Itoh 2005; Kutin 2007; Fowler et al. 2004], it has not had a formal basis. In this study, we investigated a logical mapping of adders on the AC architecture into the ones on the \( kD \) NTC architecture, showing \( \Omega(\sqrt[3]{n}) \) depth because of a practical limitation, the interaction distance by exploiting graph embedding, for the first time. Therefore, we can conclude that when the interaction distance
is limited to one, there is no $\Omega(\log n)$ depth exact quantum addition circuit on any $kD$ NTC structure by using simple logical mapping.

We should note that these results apply to the logical structure of the systems; the physical structure may differ due to the impact of quantum error correction on the physical arrangement of qubits. Also, our method can be applied to analyze reversible classical circuits as well as quantum circuits.

Although the exact quantum integer adder circuit is an important circuit, it is also desirable to analyze other quantum arithmetic circuits in the same fashion. For example, it would be interesting to investigate multipliers, modulo adders, and multipliers over $\mathbb{Z}_p$ or $GF(2^n)$ as well as other application circuits.

The only tool from graph theory that we have used in this study is the dilation property of graph embedding. However, graph embedding has many other interesting properties which may affect the layout of final quantum arithmetic circuit on a specific graph structure. For example, the congestion, expansion, and load are also important [Keh and Lin 1997], and their effects on quantum arithmetic circuits for the 1D, 2D, and 3D structures should be studied. We may investigate the results of Bein et. al. [Bein et al. 2000] in the view of embedding quantum arithmetic circuits in $kD$ NTC structures.

REFERENCES

Bacon, D. and van Dam, W. 2010. Recent progress in quantum algorithms. Commun. ACM 53, 2, 84–93.

Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A., and Weinfurter, H. 1995. Elementary gates for quantum computation. Phys. Rev. A 52, 5 (Nov), 3457–3467.

Beckman, D., Chari, A. N., Devabhaktuni, S., and Preskill, J. 1996. Efficient networks for quantum factoring. Phys. Rev. A 54, 2 (Aug), 1034–1063.

Bein, W. T., Larmore, L. L., Shields, C. J., and Sudborough, I. H. 2000. Three-dimensional embedding of binary trees. In Parallel Architectures, Algorithms and Networks, 2000. I-SPAN 2000. Proceedings. International Symposium on. 140–147.

Bennet, C. H. 1973. Logical reversibility of computation. IBM Journal of Research and Development 17, 525–532.

Brown, D., Kashefi, E., and Perdrix, S. 2009. Computational depth complexity of measurement-based quantum computation. http://arxiv.org/abs/0909.4673

Cheung, D., Maslov, D., and Severini, S. 2007. Translation techniques between quantum circuit architectures. Poster Presentation, Workshop on Quantum Information Processing.

Cleve, R. and Watrous, J. 2000. Fast parallel circuits for the quantum fourier transform. In Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on. 526–536.

Cuccaro, S. A., Draper, T. G., Kutin, S. A., and Moulton, D. P. 2004. A new quantum ripple-carry addition circuit. http://arxiv.org/abs/quant-ph/0410184

Diestel, R. 2005. Graph Theory, 3 ed. Springer-Verlag.

Doucet, B., Ioffe, L. B., and Vidal, J. 2004. Discrete non-abelian gauge theories in josephson-junction arrays and quantum computation. Phys. Rev. B 69, 21 (Jun), 214501.

Draper, T. G. 2000. Addition on a quantum computer. http://arxiv.org/abs/quant-ph/0008033

Draper, T. G., Kutin, S. A., Rains, E. M., and Svore, K. M. 2004. A logarithmic-depth quantum carry-lookahead adder. http://arxiv.org/abs/quant-ph/0406142

Dür, W., Briegel, H.-J., Cirac, J. I., and Zoller, P. 1999. Quantum repeaters based on entanglement purification. Phys. Rev. A 59, 1 (Jan), 169–181.

Ercegovac, M. D. and Lang, T. 2003. Digital Arithmetic, 1 ed. Morgan Kaufmann Publishers.

Fowler, A. G., Devitt, S. J., and Hollenberg, L. C. 2004. Implementation of shor’s algorithm on a linear nearest neighbor qubit array. Quantum Information and Computation 4, 4, 237–251.
Byung-Soo Choi and Rodney Van Meter

Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. In STOC ’96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. ACM, New York, NY, USA, 212–219.

Häffner, H., Hänsel, W., Roos, C. F., Benhelm, J., Chek-al Kar, D., Chwalla, M., Köhler, T., Rapol, U., Riebe, M., Schmidt, P. O., Böcher, C., Gühne, O., Dür, W., and Blatt, R. 2005. Scalable multi-particle entanglement of trapped ions. Nature 438, 643–646.

Heckmann, R., Klasing, R., Monien, B., and Unger, W. 1991. Optimal embedding of complete binary trees into lines and grids. In Proc. 17th Int. Workshop on Graph-Theoretic Concepts in Comp. Sci., LNCS 570. Springer Verlag, 25–35.

Helmer, F., Mariantoni, M., Fowler, A. G., von Delft, J., Solano, E., and Marquardt, F. 2007. Two-dimensional cavity grid for scalable quantum computation with superconducting circuits. http://arxiv.org/abs/0706.3625

Hennessy, J. L. and Patterson, D. A. 2006. Computer Architecture: A Quantitative Approach, 4th ed. Morgan Kaufman.

Kane, B. E. 1998. A silicon-based nuclear spin quantum computer. Nature 393, 133–137.

Kutin, S. A. 2007. Shor’s algorithm on a nearest-neighbor machine. Poster Presentation, Proc. of Asian Conference on Quantum Information Science.

Lafont, M., Simon, D., Boileau, J.-C., Baugh, J., Ditty, M. J., and Laflamme, R. 2007. Using error correction to determine the noise model. Physical Review A 75, 1, 012331.

Maslov, D. 2007. Linear depth stabilizer and quantum fourier transformation circuits with no auxiliary qubits in finite-neighbor quantum architectures. Physical Review A 76, 5, 052310.

Mosca, M. 2008. Quantum algorithms. http://arxiv.org/abs/0808.0369

Möttönen, M. and Vartiainen, J. J. 2006. Decompositions of general quantum gates. http://arxiv.org/abs/0504100

Pérez-Delgado, C. A., Mosca, M., Cappellaro, P., and Cory, D. G. 2006. Single spin measurement using cellular automata techniques. Physical Review Letters 97, 10, 100501.

Raussendorf, R., Browne, D. E., and Briegel, H. J. 2003. Measurement-based quantum computation on cluster states. Phys. Rev. A 68, 2 (Aug), 022312.

Shende, V. V., Bullock, S. S., and Markov, I. L. 2006. Synthesis of quantum-logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25, 6 (June), 1000–1010.

Shor, P. W. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26, 5, 1484-1509.

Takahashi, Y. 2009. Quantum arithmetic circuits: A survey. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E92-A, 5, 1276–1283.

Takahashi, Y., Kunihoro, N., and Ohta, K. 2007. The quantum fourier transform on a linear nearest neighbor architecture. Quantum Information and Computation 7, 4, 383–391.

Trisetyarso, A. and Van Meter, R. 2009. Circuit design for a measurement-based quantum carry-lookahead adder. http://arxiv.org/abs/0903.0748

Unger, W. 2008. private communication. private communication.

Van Meter, R. 2004. Communications topology and distribution of the quantum fourier transform. In Proc. of Quantum Information Technology Symposium (QIT10). 2.

Van Meter, R. and Itoh, K. M. 2005. Fast quantum modular exponentiation. Phys. Rev. A 71, 5 (May), 052320.

Vedral, V., Barenco, A., and Ekert, A. 1996. Quantum networks for elementary arithmetic operations. Phys. Rev. A 54, 1 (Jul), 147–153.

ACM Transactions on Computational Logic, Vol. 2, No. 3, 09 2001.
VOLLMER, H. 1999. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

WEISSTEIN, E. W. 2010. *Binary tree*. http://mathworld.wolfram.com/BinaryTree.html

Received February ???; November ???; accepted January ???