# AN $\Theta(\sqrt{n})$-DEPTH QUANTUM ADDER ON THE 2D NTC QUANTUM COMPUTER ARCHITECTURE ${ }^{a}$ 

BYUNG-SOO CHO\|b ${ }^{b}$<br>Center for Quantum Information Processing<br>Department of Electrical and Computer Engineering<br>University of Seoul<br>13 Siripdae-gil (90 jeonnong-dong), Dongdaemun-gu, Seoul, 130-743, Republic of Korea

RODNEY VAN METER ${ }^{c}$
Faculty of Environment and Information Studies Keio University
5322 Endo, Fujisawa, Kanagawa, 252-8520, Japan

Received (May 28, 2018)
Revised (revised date)


#### Abstract

In this work, we propose an adder for the 2D NTC architecture, designed to match the architectural constraints of many quantum computing technologies. The chosen architecture allows the layout of logical qubits in two dimensions and the concurrent execution of one- and two-qubit gates with nearest-neighbor interaction only. The proposed adder works in three phases. In the first phase, the first column generates the summation output and the other columns do the carry-lookahead operations. In the second phase, these intermediate values are propagated from column to column, preparing for computation of the final carry for each register position. In the last phase, each column, except the first one, generates the summation output using this column-level carry. The depth and the number of qubits of the proposed adder are $\Theta(\sqrt{n})$ and $O(n)$, respectively. The proposed adder executes faster than the adders designed for the 1D NTC architecture when the length of the input registers $n$ is larger than 58 .


Keywords: quantum arithmetic algorithms, quantum circuit, depth lower bound, adder, 2D NTC quantum computer architecture
Communicated by: to be filled by the Editorial

## 1 Introduction

Quantum computers have been proposed to exploit the exotic properties of quantum mechanics for information processing. Among many potential uses, two quantum algorithms have received the bulk of the attention. One is Shor's large number factoring algorithm [1] and the other is Grover's unstructured database search algorithm [2], though there has also been much progress recently on other algorithms [3, 4, 5]. Quantum algorithms are often shown to

[^0]be more efficient than classical ones by analyzing the number of queries to an oracle. However, for a more exact performance analysis, we need to analyze the quantum algorithms in terms of the detailed quantum circuits necessary to implement them. Among many circuits, as in classical computation, a core set of subroutines whose behavior will strongly impact the performance of the overall algorithm is arithmetic, hence we focus on the adder in this work.

Numerous quantum addition circuits have been proposed using abstract models of the computer itself. The basic elementary quantum arithmetic operations including addition have been proposed by Vedral et al. [6] and Beckman et al. [7], following seminal work on elementary reversible full- and half-adders by Fredkin and Toffoli [8, and Feynman [9]. Glassner proposed an one-qubit full adder [10]. Subsequently, Cheng and Tseng proposed an $n$-qubit full adder and subtractor based on the work of Glassner [11]. Reducing the space requirements for those earlier adders [6], Cuccaro et al. proposed a linear-depth ripple-carry adder with only a single ancillary qubit [12]. Meanwhile Draper proposed a transform adder based on the quantum Fourier transform [13]. Draper et al. proposed a fast quantum carrylookahead adder [14]. Takahashi and Kunihiro have shown that addition can actually be performed with no ancillae, at the expense of a deeper circuit [15].

Incorporating the behavior of these circuits, we can estimate the overall quantum speedup more accurately than simply addressing the issue at the query-level, and confirm again that the quantum speedup is very high. However, it is not possible to determine the exact performance gain unless the practical issues of architecture are considered; both the constant factors and the leading order of both the computational complexity and minimum execution time (or circuit depth) depend on the assumed underlying machine. Hence we have to consider many issues such as error correction, communication, gate, and qubit technologies [16]. For example, Maslov et. al [17] pointed out the importance of the problem of placing circuit variables on the underlying qubit layout. Unfortunately, it is impossible to consider all practical issues at the same time. To avoid this problem, we usually define a practical quantum computer architecture incorporating as many practical constraints as possible. For many quantum computer architectures, the 2D NTC architecture is a reasonable model capturing the key factors that impact performance. NTC allows $N$ earest-neighbor interactions, $T$ wo-qubit quantum gates, and Concurrent executions of gates [18]. An example of a potentially scalable architecture with the nearest-neighbor constraint is that of Kielpinski et. al [19. Barenco et al. [20] showed one way to decompose a given quantum circuit into two-qubit gates. Steane [21] investigated the necessity of concurrent execution for error correction and fault-tolerance; concurrency is also required at the application level for high performance. The 2D allows a single qubit to interact with four neighboring qubits. With more neighboring qubits than the 1D case, the 2D layout should show higher performance, thanks to reduced distance between many pairs of qubits and the potential for more concurrent movement of qubits. Likewise, a 3D layout should show higher performance than the 2D case, but the complexity of fabricating and controlling qubits in three dimensions likely makes it impractical. Therefore, we believe that the 2D layout is the most reasonable choice at the middle level of performance and control overhead. Thus, it would be interesting to understand the quantum speedup in this context.

Surprisingly, as far as we know, there is no quantum addition circuit designed specifically for the 2 D NTC architecture. Hence we have to design a quantum addition circuit for the 2 D

NTC architecture and estimate the performance gain. Based on this, our contributions are as follows.

## - Propose a quantum adder on the 2D NTC architecture.

First, we lay out the qubits in a $\sqrt{n} \times \sqrt{n}$ array where $n$ is the input size, in qubits. Based on this layout, we propose a three-phase quantum addition algorithm. In the first phase, the first column does a ripple-carry addition and the other columns do carry-lookahead operations. In the second phase, the column-level carry is propagated in ripple fashion between the columns. In the last phase, each column transports its column-level carry input into the cells to generate the final summation value.

## - Analyze the proposed adder.

We decompose the necessary quantum circuit blocks using only one- and two-qubit gates. Next, we add SWAP operations necessary to transport qubits in order to satisfy the NTC constraint. We found that the depth of the proposed adder is $150 \sqrt{n}-90$ in terms of one- and two-qubit gates. Asymptotically, the depth is $\Theta(\sqrt{n})$ meeting the depth lower bound we established in earlier work [22]. To execute many quantum gates in parallel, the proposed adder utilizes many working qubits as $2 n-\sqrt{n}$ qubits.
Since the 2D NTC layout generalizes the 1D NTC architecture, the adders designed for the 1D NTC architecture can also be implemented on the 2D NTC architecture without modification. After reevaluating the depth of the adders for the 1D NTC architecture, we find that our new 2D adder works faster when $n \geq 58$.

This paper is organized as follows. We explain the addition algorithm, and qubit and circuit layouts for the 2D NTC architecture in Section 2 The temporal and spatial resources are analyzed in Section 3 Finally, we conclude this work and point out some problems in Section 4

## 2 Adder on the 2D NTC Structure

In this section, we first explain how the qubits are laid out on the 2D structure firstly. Second, we explain an addition algorithm based on a slight modification of carry-lookahead addition. Third, we discuss how the addition algorithm is mapped with the circuit blocks. Finally, we show how the ancillae qubits can be initialized.

### 2.1 Qubit Layout

On the 2D NTC structure, we can lay out the qubits as shown in Figure 亿 In the figure, the two input registers are $A=a_{n} \cdot 2^{n-1}+a_{n-1} \cdot 2^{n-2}+\cdots+a_{1}$ and $B=b_{n} \cdot 2^{n-1}+b_{n-1} \cdot 2^{n-2}+$ $\cdots+b_{1}$. As shown in the figure, the two inputs $a_{i}$ and $b_{i}$ are interleaved where $1 \leq i \leq n$. The number of rows and columns are $2 \sqrt{n}$ and $\sqrt{n}$, respectively. Two inputs $a_{i}$ and $b_{i}$ are located at a ( $k$-th column, $j$-th row) cell where $k=\lceil i / \sqrt{n}\rceil$ and $j=i-(k-1) \sqrt{n}$. The figure shows only the input qubits for clarity. For simplicity, we assume without loss of generality that $\sqrt{n}$ is an integer.

### 2.2 Adapting Carry-Lookahead Addition to Limited Interaction Distance

To set the stage for the later arithmetic discussions, let us first explain the ripple for two $n$ qubit input registers, $a$ and $b$. Since the summation value for the $i$-th position $s_{i}$ is generated


Fig. 1. Layout of Input Qubits for a 2D NTC adder. Two inputs are $A=a_{n} \cdot 2^{n-1}+a_{n-1} \cdot 2^{n-2}+\cdots+a_{1}$ and $B=b_{n} \cdot 2^{n-1}+b_{n-1} \cdot 2^{n-2}+\cdots+b_{1}$. $i$-th qubit is located at $(k, j)$ position where $k=\lceil i / \sqrt{n}\rceil$ and $j=i-(k-1) \sqrt{n}$. Ancillae qubits are not shown for simplicity.
by $a_{i} \oplus b_{i} \oplus c_{i}$, where $a_{i}$ and $b_{i}$ are the $i$-th qubits in the input registers, and $c_{i}$ is the carry input from the summation of the $(i-1)$-th position, the time complexity of the addition depends on how fast the carry information can be transported between the bit positions.

The simplest circuit is the ripple carry adder, which propagates the carry information stepwise from position to position. The carry output for the $(i+1)$-th position, $c_{i+1}$, should be one if a majority of the bits $a_{i}, b_{i}$, and $c_{i}$ are one, and zero otherwise; it is generated by $a_{i} \cdot b_{i} \oplus a_{i} \cdot c_{i} \oplus b_{i} \cdot c_{i}$. Therefore, the final summation value $s_{n}$ is generated only after $n$ ripple carry time steps.

To reduce this time, a carry-lookahead method was devised. In this method, two additional values are defined as follows:

$$
\begin{align*}
g_{i} & =a_{i} \cdot b_{i} .  \tag{1}\\
p_{i} & =a_{i} \oplus b_{i} . \tag{2}
\end{align*}
$$

Implicitly, $g_{i}$ and $p_{i}$ determine whether this bit position generates a carry out independent of the carry in, or propagates its incoming carry to its output carry, respectively. Only one of these may be true, though both may be false (called carry kill, though kill is not necessary in the actual circuit). The carry output for ( $i+1$ )-th position is generated as $c_{i}=g_{i} \oplus p_{i} \cdot c_{i-1}$. Therefore, if $g_{i}$ is one, $c_{i}$ has no dependence on $c_{i-1}$, and hence disconnects the carry chain. However, if $g_{i}$ is zero, $c_{i}$ is dependent on $c_{i-1}$. In the worst case, the longest chain is from $c_{1}$ to $c_{n}$. To decompose this long chain into sub-units, two variables $G[i, j]$ and $P[i, j]$ are also defined as follows.

$$
\begin{align*}
G[i, j] & =g_{j} \oplus p_{j} \cdot G[i, j-1] .  \tag{3}\\
P[i, j] & =p_{j} \cdot P[i, j-1] . \tag{4}
\end{align*}
$$

$G[i, j]$ indicates whether an entire span of the addition, from qubit $i$ to qubit $j$, generates a carry. Similarly, $P[i, j]$ indicates whether the $[i, j]$ span propagates the carry from position $i$ all the way to position $j$. By calculating these values concurrently and progressively increasing the span of $G$ and $P$, the total time to create complete carry information for the entire register can be reduced to $O(\log n)$, provided that communication within the system is adequately fast.

Unfortunately, this carry-lookahead addition algorithm is defined assuming no limitation of interaction distance, and hence cannot be applied for the 2D NTC architecture without modification. In this work, we slightly modify the carry-lookahead, which consists of three phases as follows.

### 2.2.1 Phase 1: Ripple Carry Addition on the First Column, and Carry-Lookahead on the Other Columns

As shown in Figure 2, the first column does the typical ripple carry addition. From the first position to the last position, each position generates a summation value and a carry output as follows.

$$
\begin{align*}
s_{i} & =a_{i} \oplus b_{i} \oplus c_{i}  \tag{5}\\
c_{i+1} & =a_{i} \cdot b_{i} \oplus a_{i} \cdot c_{i} \oplus b_{i} \cdot c_{i} \tag{6}
\end{align*}
$$

where $c_{1}=0$. Since the carry output of the $i$-th position must be used as input for the next $(i+1)$-th position, there is an information dependency, hence this step takes about $O(\sqrt{n})$ time.

During this time, the other columns concurrently generate other necessary information for carry-lookahead operations. For example, the $k$-th column works as follows. First, each $(k, j)$ cell generates $g_{(k-1) \sqrt{n}+j}$ and $p_{(k-1) \sqrt{n}+j}$ concurrently,

$$
\begin{align*}
g_{(k-1) \sqrt{n}+j} & =a_{(k-1) \sqrt{n}+j} \cdot b_{(k-1) \sqrt{n}+j}  \tag{7}\\
p_{(k-1) \sqrt{n}+j} & =a_{(k-1) \sqrt{n}+j} \oplus b_{(k-1) \sqrt{n}+j} \tag{8}
\end{align*}
$$

where $1 \leq j \leq \sqrt{n}$. After that, each $(k, j)$ cell generates $G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]$ and $P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]$ sequentially,

$$
\begin{align*}
G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]= & g_{(k-1) \sqrt{n}+j} \oplus  \tag{9}\\
& p_{(k-1) \sqrt{n}+j} \cdot G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j-1] \\
P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]= & p_{(k-1) \sqrt{n}+j} \cdot P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j-1(10)
\end{align*}
$$

where $G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+1]=g_{(k-1) \sqrt{n}+1}$ and $P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+1]=$ $p_{(k-1) \sqrt{n}+1}$. The same process is applied for the other columns.

After this phase, the first column generates its final summation output and also the carry output $c_{\sqrt{n}+1}$. The other columns generate the column-level carry-lookahead information $G[(k-1) \sqrt{n}+1, k \sqrt{n}]$ and $P[(k-1) \sqrt{n}+1, k \sqrt{n}]$.

### 2.2.2 Phase 2: Inter-Column Carry Propagation

The final carry output of the first column, $c_{\sqrt{n}+1}$, is given as an initial input value for the column-level carry generation logic as shown in Figure 3. Each column, except the first,

6 An $\Theta(\sqrt{n})$-depth Quantum Adder on a 2D NTC Quantum Computer Architecture


Fig. 2. First phase.
During this phase, the first column executes a ripple-carry adder. The other $k$-th column generates $g_{(k-1) \sqrt{n}+j}$ and $p_{(k-1) \sqrt{n}+j}$ concurrently, and then $G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]$ and $P[(k-$ 1) $\sqrt{n}+1,(k-1) \sqrt{n}+j$ ] sequentially.
generates its column-level carry output as follows.

$$
\begin{equation*}
\text { Column_carry }=c_{k \sqrt{n}+1}=G[(k-1) \sqrt{n}+1, k \sqrt{n}] \oplus c_{(k-1) \sqrt{n}+1} \cdot P[(k-1) \sqrt{n}+1, k \sqrt{n}] . \tag{11}
\end{equation*}
$$

### 2.2.3 Phase 3: Carry Generation and Summation

After the first phase, each $(k, j)$ cell has the carry-lookahead information $G[(k-1) \sqrt{n}+1,(k-$ 1) $\sqrt{n}+j]$ and $P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]$. After the second phase, each column has the incoming column-level carry $c_{(k-1) \sqrt{n}+1}$. By propagating incoming column-level carry as shown in Figure 4 each $(k, j)$ cell can calculate its final carry input as

$$
\begin{align*}
c_{i}=c_{(k-1) \sqrt{n}+j}= & G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]  \tag{12}\\
& \oplus c_{(k-1) \sqrt{n}+1} \cdot P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j] .
\end{align*}
$$

After that, each cell can generate the final summation value as

$$
\begin{equation*}
s_{i}=s_{(k-1) \sqrt{n}+j}=a_{(k-1) \sqrt{n}+j} \oplus b_{(k-1) \sqrt{n}+j} \oplus c_{(k-1) \sqrt{n}+j} . \tag{13}
\end{equation*}
$$

### 2.3 Circuit Layout

In the first phase, the first column and the other columns use different circuit blocks. The circuit blocks for the first column are shown in Figure 5(a). To do the ripple carry addition, a half-adder (HA) for the first position and $\sqrt{n}-1$ full-adders (FA) are used. The circuit blocks for the other columns are shown in Figure 5(b) As explained in the previous part, it generates first $g_{(k-1) \sqrt{n}+j}$ and $p_{(k-1) \sqrt{n}+j}$ concurrently by using the $\mathbf{g}, \mathbf{p}$ circuit blocks and


Fig. 3. Second phase.
The purpose of this phase is to generate column-level carry output for each column sequentially.
then $G[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]$ and $P[(k-1) \sqrt{n}+1,(k-1) \sqrt{n}+j]$ sequentially by using the $\mathbf{G}, \mathbf{P}$ circuit blocks.

The block-level circuit for the second phase is shown in Figure 6. The circuit block Colcarry has three inputs: $G$ and $P$ from the corresponding column and Column_carry from the lower column.

Figure 7 shows the circuit blocks for the third phase. In the figure, $\mathbf{c}$ and $\mathbf{c 1}$ represent the blocks for generating carry output for $i$-th position. Note for the first row, $p$ and $g$ are the same as $P$ and $G$, and hence the circuit block is slightly different. SUM, SUM1, and SUM2 are for generating the final summation value for $j$-th position.

### 2.4 Clearing Ancillae Qubits

As shown in Table 2, three types of ancillae qubits are used, $c_{i}, P[i, j]$, and Column_carry ${ }_{k}$. To clean these ancillae, we have used the strategy proposed in Reference [14]. The key idea of this approach is based on the observation that in two's complement arithmetic

$$
\begin{align*}
-x & \equiv \bar{x}+1 \quad\left(\bmod 2^{n}\right),  \tag{14}\\
\bar{x}+x & \equiv-1 \quad\left(\bmod 2^{n}\right),  \tag{15}\\
-x-1 & \equiv \bar{x} \quad\left(\bmod 2^{n}\right), \tag{16}
\end{align*}
$$

where $\bar{x}$ is the bit-wise inversion of $x$. Let us consider an addition of $A$ and $B, A D D(A, B, 0)=$ $(A, S, C)$, where $S$ and $C$ are the bitwise sum and carry vectors, respectively. Let us consider another addition of $A$ and $\bar{S}, A D D(A, \bar{S}, 0)=(A, \bar{B}, D)$, where $\bar{B}$ and $D$ are sum and carry

> First, transport Column_carry Second, generate $c_{i}$ for each position Third, generate $s_{i}$ for each position


Fig. 4. Third phase.
Using the incoming carry for each column, all carry and sum are generated sequentially.
vectors, respectively. Note the bitwise sum is $\bar{B}$ because

$$
\begin{equation*}
A+\bar{S}=A-(A+B+1)=-B-1=\bar{B} . \tag{17}
\end{equation*}
$$

It is worth noting that $C$ must be equal to $D$ because of

$$
\begin{align*}
A \oplus \bar{S} \oplus D & =\bar{B}  \tag{18}\\
A \oplus A \oplus B \oplus C \oplus 1 \oplus D & =\bar{B}  \tag{19}\\
B \oplus C \oplus 1 \oplus D & =\bar{B},  \tag{20}\\
\bar{B} \oplus C \oplus D & =\bar{B}  \tag{21}\\
C \oplus D & =0  \tag{22}\\
C & =D . \tag{23}
\end{align*}
$$

Now we follow the circuit as shown in Figure 8. Conceptually any addition circuit can be divided into two parts, $C A R R Y$ generation $\left(C_{i}\right)$ and $S U M$ generation $\left(S_{i}\right)$. As shown in the figure, we apply $C A R R Y$ as follows.

$$
\begin{equation*}
C A R R Y(A, B, 0) \Longrightarrow(A, A \oplus B, C) \tag{24}
\end{equation*}
$$

As the second step, we apply $S U M$ as follows.

$$
\begin{equation*}
S U M(A, A \oplus B, C) \Longrightarrow(A, A \oplus B \oplus C, C)=(A, S, C) \tag{25}
\end{equation*}
$$

Apply two operations

$$
\begin{equation*}
N O T_{2}(A, S, C) \Longrightarrow(A, \bar{S}, C) \tag{26}
\end{equation*}
$$



Fig. 5. Circuit flow for the first phase.
Note FA and HA are the full adder and the half adder, respectively. $s$ and $c$ are initially $|0\rangle$, and $s_{i}$ and $c_{i}$ are summation and carry for each position, respectively.

$$
\begin{equation*}
C N O T_{1,2}(A, \bar{S}, C) \Longrightarrow(A, A \oplus \bar{S}, C) \tag{27}
\end{equation*}
$$

Meanwhile,

$$
\begin{equation*}
C A R R Y(A, \bar{S}, 0) \Longrightarrow(A, A \oplus \bar{S}, D) \tag{28}
\end{equation*}
$$

Since the two carry vectors $C$ and $D$ for $A+B$ and $A+\bar{S}$ are the same, the above line changes to

$$
\begin{equation*}
C A R R Y(A, \bar{S}, 0) \Longrightarrow(A, A \oplus \bar{S}, C) \tag{29}
\end{equation*}
$$

Therefore, running the inverse operation,

$$
\begin{equation*}
C A R R Y^{-1}(A, A \oplus \bar{S}, C) \Longrightarrow(A, \bar{S}, 0) \tag{30}
\end{equation*}
$$

Finally, apply $\mathrm{NOT}_{2}$ as follows.

$$
\begin{equation*}
\operatorname{NOT}_{2}(A, \bar{S}, C) \Longrightarrow(A, S, 0) \tag{31}
\end{equation*}
$$

to generate the final sum and clean ancillae.

## 3 Analysis

### 3.1 Depth Analysis

To analyze the depth of the proposed adder, we have to decompose the circuit blocks into elementary gates, which can be decomposed into unit delay gates. In this work, we assume one-qubit, CNOT, and Control- $\sqrt{N O T}$ gates have unit delay. The elementary gates we have chosen for constructing our circuits are SWAP, CCNOT, CNOT, Control- $\sqrt{N O T}$, and one-qubit gates. In this paper, we use the three-CNOT construction for SWAP gate. Figure 9 shows the conventional form of CCNOT (left) and its decomposition into one-qubit and two-qubit gates (right).

### 3.1.1 Circuit Decomposition with NTC Constraints

Now we decompose the circuit blocks for three phases with the chosen elementary gates and necessary SWAP operations to satisfy the NTC constraints. The blocks are shown in Figures [10 to 14. The circuit of HALF ADDER is shown in Figure 10(a). Figure 10(b) 11] shows a


Fig. 6. Circuit flow for the second phase.
Col-carry block generates a column-level carry output, which is used for the actual incoming carry value for the next (right) column.
decomposition of FULL ADDER into elementary gates. In this figure, there is no limitation on the distance between operands for a gate. To satisfy the NTC constraint, we redesign it as shown in Figure 10(c) by adding several SWAP gates to move the qubits to neighboring positions. This approach is also applied for the following circuit blocks. The circuits for $\mathbf{g}$ and $\mathbf{p}$, and the generalized $\mathbf{G}$ and $\mathbf{P}$ are shown in Figure 11. The circuit of Column_carry is shown in Figure 12, For generating $\mid$ Col_Carry $\left._{k+1}\right\rangle$, a single CCNOT is enough. However, to propagate it to the next column and to propagate $\left|C_{o l} C_{-} C a r r y_{k}\right\rangle$ to the rows, a SWAP is necessary. For implementing the last SWAP gate in the neighbor interaction only case, several SWAPs are necessary as shown in Figure 12(b). The initial circuit for Carry is shown in Figure 13(a). Since the Col_carry has to be moved to the upper row, several SWAPs are necessary as shown in Figure 13(b). After this circuit, the Col_carry is transported to the top position, and the others are to the lower row. Since the carry for the first row is different from other rows, Figures $13(\mathrm{c})$ and $13(\mathrm{~d})$ show its circuits. The circuits for SUM are shown in Figures 14(a) and 14(b) For the second and the first row, we have to use slightly different circuits as shown in Figures 14(c) and 14(d) and Figure 14(e) respectively.

### 3.1.2 Total Depth

Based on the revised circuits with satisfying the NTC constraint, we can summarize the depth of each elementary gate and circuit block as shown in Table 1 .

The proposed adder works in three sequential phases, and hence the overall depth is the sum of the depths for each phase. The depth for each phase is the "long pole", or the longest delay among the parallel execution paths. In the first column, one HA and $(\sqrt{n}-1)$ FA operations are executed sequentially. Since HA needs 10 unit-gate steps and FA needs 26 unit-gate steps, $26 \sqrt{n}-16$ unit-gate steps are needed. On the other hand, the other columns


Fig. 7. Circuit flow for the third phase. c represents the block for generating carry output for $i$-th position. SUM is for generating the final summation value for $i$-th position.
need one $\mathbf{g}, \mathbf{p}+(\sqrt{n}-1) \mathbf{G}, \mathbf{P}$, which is $36 \sqrt{n}-26$. The overall depth for the first phase is the longer of the two column types, hence $36 \sqrt{n}-26$. The second phase consists of $(\sqrt{n}-1)$ Column_carry operations, requiring a total of $18 \sqrt{n}-18$ time steps. The third phase consists of $(\sqrt{n}-1)$ Carry + Carry1 and SUM1 operations for the longest path. Hence, the depth is $21 \sqrt{n}+1$ unit-gate steps. By summing depths of each phase, the total depth is $75 \sqrt{n}-43$.

The above depth is only for generating the summation output without clearing the ancillae. For clearing ancillae, we apply more circuits as shown in Figure 8. Based on this figure, we can decompose the above three phases into the carry generation flow and the sum generation flow. The first and the second phases are for the carry generation flow. The third phase has to be divided into the carry generation flow and the sum generation flow. The above depth is apportioned as $75 \sqrt{n}-50$ for carry generation flow and 7 for sum generation flow. As shown in Figure 8 we need to apply NOT and CNOT gates and then inverse of the carry generation flow again with the final NOT gate. Hence, the overall depth is $75 \sqrt{n}-50+7+$ $1+1+75 \sqrt{n}-50+1=150 \sqrt{n}-90$.

### 3.2 Required Space

The number of qubits for the adder is shown in Table 2. As shown in the first column, some qubits work for multiple purposes. Note the additional number of qubits is $2 n-\sqrt{n}$, which is less than twice the minimum $2 n$ qubits [12, 15].


Fig. 8. Clearing Ancillae Qubits. By applying the inverse of the carry generation flow, the ancillae qubits can be cleaned.


Fig. 9. Circuit for CCNOT

### 3.3 Comparison to Other Adders

When only interactions between neighboring qubits are allowed, the depth of arithmetic circuits increases. For the 2D case, the depth lower bound was proven to be $\Omega(\sqrt{n})[22]$. Therefore, the depth of the proposed adder is asymptotically optimal.

Beyond the asymptotic behavior, it seems more interesting and important to compare with other adders in the practical cases. Specifically, it is necessary to compare adders designed for the 1D NTC architecture since they can be implemented on the 2D NTC architecture without modification, using a simple serpentine qubit layout. The overall analysis and the comparison between the adders are shown in Table 3. The first column distinguishes the architecture and the second column lists the adder type. For the 1D NTC architecture, we choose three typical adders. Vedral et al. proposed a plain ripple-carry adder [6], named VBE in the table. VBE-Improved is the Van Meter and Itoh update to this adder [18. Cuccaro et al. proposed a ripple carry adder with only one ancillae qubit [12], named $C D K M$. For the 2D NTC architecture, the present adder is shown. For the architecture with arbitrary distance interaction, several adders are evaluated. Draper proposed a quantum Fourier transform adder [13], named QFT-based. By exploiting the classical fast addition algorithm, Draper et al. also proposed a carry-lookahead adder [14, named CLA-based. Kawata et al. also proposed an adder based on the combination of ripple carry adder and carry-lookahead adder [24], named $R C A+C L A$-based. For comparison, the depth and the size of each adder is shown in the third column. In this work, the depth is measured by in units of one- and two-qubit gates for the 1 D and 2D NTC architectures. On the other hand, the depth for the $\mathbf{A C}$ architecture is based on one-, two-, and CCNOT gates. The size is for the number of qubits for input, output, and ancillae. In the fourth column, the input size is shown when the selected adder works faster than the present adder. In the fifth column, we calculate $K Q$, the product of qubits and depth where $K$ and $Q$ are the numbers of logical qubits and computational steps, respectively [23]. KQ is used to estimate the strength of error correction required.

From this table we can point out three key results. First, when the size of input is larger than 58 , the present adder works faster than 1D NTC adders. Second, the present adder


Fig. 10. Circuit for HALF ADDER (a); Circuits for FULL ADDER with arbitrary interaction (b) and with only nearest-neighbor interaction (c)

(a)


Fig. 11. Circuits for $\mathbf{g}, \mathbf{p}$ (a); Circuits for $\mathbf{G}$ and $\mathbf{P}$ with arbitrary interaction (b) and with only nearest-neighbor interaction (c)
needs about two times number of qubits than 1D NTC adders. Lastly, the present adder has a smaller KQ factor when the input size is larger than 278.

## 4 Conclusion and Open Problems

In this work, we proposed a quantum adder for the 2D NTC architecture for the first time. Van Meter and Oskin indicated that an adder would be in $O(\sqrt{n})$ time complexity on a 2 D architecture, but no circuit has been provided [25]. The proposed adder has the depth complexity $\Theta(\sqrt{n})$ with $O(n)$ qubits. We found that the proposed adder works faster than a 1 dipple-carry adder when the length of the input registers is larger than 58 , and requires about two times the number of additional qubits.

Although this adder is, to the best of our knowledge, the first one specifically designed for a 2 D architecture, we suspect it will not be the last; we anticipate that several improvements are possible. First, the number of additional gates is very large. Most of the gates for the proposed adder are used for transporting qubits to neighboring positions so that gates can be executed. By arranging qubits in a better way, we may be able to reduce the necessary propagattion


Fig. 12. Circuits for Column_carry with arbitrary interaction (a) and with only nearest-neighbor interaction (b)


Fig. 13. Circuits for Carry with arbitrary interaction (a) and with only nearest-neighbor interaction (b); Circuits for Carry1 with arbitrary interaction (c) and with only nearest-neighbor interaction (d)
operations. Second, the phase for cleaning the ancillae qubits roughly doubles the total number of quantum operations. In the present adder, the ancillae qubits are reinitialized by applying the inverse circuit, doubling the overall depth. Perhaps there is some way to reduce this drawback by exploiting some overlap of the clearing phase with the computation phase. Third, the number of ancillae is also very large. The proposed design attempts to achieve the highest parallel execution at the expense of requiring more ancillae, but this tradeoff may prove to be less than optimal for two reasons. First, qubits themselves are expensive resources, and in many applications could be allocated to other work if not used directly in the adder; second, inserting the ancillae into our layout increases the distance between qubits, forcing the addition of more SWAPs and slowing down the circuit.

## Acknowledgements

This research is supported in part by the Japan Society for the Promotion of Science (JSPS) through its Funding Program for World-Leading Innovative R\&D on Science and Technology (FIRST Program), and in part by the National Research Foundation of Korea Grant funded by the Korean Government(Ministry of Education, Science and Technology).[NRF-2010-359D00012]

## References

1. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a


Fig. 14. Circuits for SUM with arbitrary interaction (a) and with only nearest-neighbor interaction (b); Circuits for SUM1 with arbitrary interaction (c) and with only nearest-neighbor interaction (d); Circuit for SUM2 (e)

Table 1. Depth analysis of each gate and circuit

| Name | Composition of the longest path | \# of unit-gate steps |
| :--- | :--- | ---: |
| SWAP | 3 CNOTs | 3 |
| CCNOT | 1 SWAP + 6 unit gates | 9 |
| HALF ADDER | 1 CCNOT + 1 CNOT | 10 |
| FULL ADDER | 2 CCNOTs + 2 CNOTs + 2 SWAPs | 26 |
| g and p | 1 CCNOT + 1 CNOT | 10 |
| G and P | 2 CCNOTs + 6 SWAPs | 36 |
| Column_carry | 1 CCNOT + 3 SWAPs | 18 |
| Carry | 1 CCNOT + 4 SWAPs | 21 |
| Carry1 | 1 CCNOT + 2 SWAPs | 15 |
| SUM | 1 CNOT + 4 SWAPs | 13 |
| SUM1 | 1 CNOT + 2 SWAPs | 7 |
| SUM2 | 1 CNOT | 1 |

quantum computer. SIAM Journal on Computing, 26(5):1484-1509, 1997.
2. Lov K. Grover. A fast quantum mechanical algorithm for database search. In STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 212-219, New York, NY, USA, 1996. ACM.
3. Michele Mosca. Quantum algorithms. http://arxiv.org/abs/0808.0369, 2008.
4. Dave Bacon and Wim van Dam. Recent progress in quantum algorithms. Communications of the ACM, 53(2):84-93, 2010.
5. Katherine L. Brown, William J. Munro, and Vivien M. Kendon. Using quantum computers for quantum simulation. http://arxiv.org/abs/1004.5528, 2010.
6. Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. Phys. Rev. A, 54(1):147-153, Jul 1996.
7. David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. Phys. Rev. A, 54(2):1034-1063, Aug 1996.
8. Edward Fredkin and Tommaso Toffoli. Conservative logic. International Journal of Theoretical Physics, 21:219-253, 1982.
9. Richard P. Feynman. Feynman Lectures on Computation. Addison Wesley, 1996.
10. Andrew S. Glassner. Quantum computing, part 2. IEEE Computer Graphics and Applications, 21(5):86-95, 2001.
11. Kai-Wen Cheng and Chien-Cheng Tseng. Quantum full adder and subtractor. Electronics Letters, $38(22): 1343-1344$, Oct 2002.
12. Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. http://arxiv.org/abs/quant-ph/0410184, 2004.
13. Thomas G. Draper. Addition on a quantum computer. http://arxiv.org/abs/quant-ph/0008033, 2000.

16 An $\Theta(\sqrt{n})$-depth Quantum Adder on a 2D NTC Quantum Computer Architecture
Table 2. Number of Qubits

| Name | Number of qubits | Explanation |
| :--- | :--- | :--- |
| $a_{i}$ | $n$ | Input $A$ |
| $b_{i} \rightarrow p_{i} \rightarrow s_{i}$ | $n$ | Input $B$, Carry propagate for $i$-th position, and Summation $S$ |
| $\|0\rangle \rightarrow g_{i} \rightarrow G[i, j] \rightarrow c_{i}$ | $n$ | Carry generation for $i$-th position, Carry generation between $i$ and $j$, and <br> carry for $i$-th position |
| $\|0\rangle \rightarrow P[i, j]$ | $n-2 \sqrt{n}+1$ | Carry propagation between $i$ and $j$ |
| Column_carry | Inter column carry. The last Column_carry is for the final carry output. |  |
| Total | $\sqrt{n}$ | $(2 n+1)+(2 n-\sqrt{n})$ |

Table 3. Comparison with Other Designs

| Architecture | Name of Adder | (Depth, Number of Qubits) | When is the present adder faster than the corresponding adder? | KQ [23] |
| :---: | :---: | :---: | :---: | :---: |
| 1D NTC | VBE [6] | (76n-30, $3 n+1)$ | $n \geq 4$ | $228 n^{2}-O(n)$ |
|  | VBE-Improved 18] | $(20 n-15,3 n+1)$ | $n \geq 49$ | $60 n^{2}-O(n)$ |
|  | CDKM 12] | (18n+14, $2 n+2)$ | $n \geq 58$ | $36 n^{2}+O(n)$ |
| 2D NTC | Present Adder | $(150 \sqrt{n}-90,4 n-\sqrt{n}+1)$ |  | $600 n \sqrt{n}-O(n)$ |
| AC | QFT-based 13] | (3 $\log n, 2 n+1)$ | N/A | $6 n \log n+O(\log n)$ |
|  | CLA-based 14] | ( $2 \log n+2,4 n-\log n)$ | N/A | $8 n \log n+O(n)$ |
|  | RCA+CLA-based [24] | $(10 \log n+6 n / \log n, n+4 n / \log n)$ | N/A | $10 n \log n+O\left(n^{2} / \log n\right)$ |

14. Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. A logarithmic-depth quantum carry-lookahead adder. http://arxiv.org/abs/quant-ph/0406142, 2004.
15. Yasuhiro Takahashi and Noboru Kunihiro. A linear-size quantum circuit for addition with no ancillary qubits. Quantum Information and Computation, 5(6):440-448, 2005.
16. Tzvetan S. Metodi and Frederic T. Chong. Quantum Computing for Computer Architects. Synthesis Lectures on Computer Architecture. Morgan \& Claypool Publishers, 2006.
17. Dmitri Maslov, Sean M. Falconer, and Michele Mosca. Quantum circuit placement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(4):752-763, April 2008.
18. Rodney Van Meter and Kohei M. Itoh. Fast quantum modular exponentiation. Phys. Rev. A, 71(5):052320, May 2005.
19. D. Kielpinski, C. Monroe, and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. Nature, 417(6890):709-711, June 2002.
20. Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. Phys. Rev. A, 52(5):3457-3467, Nov 1995.
21. Andrew M. Steane. Space, time, parallelism and noise requirements for reliable quantum computing. Fortschritte der Physik, 46(4-5):443-457, 1998.
22. Byung-Soo Choi and Rodney Van Meter. Effects of interaction distance on quantum addition circuits. http://arxiv.org/abs/0809.4317, 2008.
23. Andrew M. Steane. Overhead and noise threshold of fault-tolerant quantum error correction. Phys. Rev. A, 68(4):042322, Oct 2003.
24. Yoshinori Kawata, Satoshi Yayu, and Shuichi Ueno. An efficient quantum addition circuit : Extended abstract. IEICE Technical Report. Circuits and systems, 107(527):95-96, March 2008.
25. Rodney Van Meter and Mark Oskin. Architectural implications of quantum computing technologies. ACM Journal on Emerging Technologies in Computing Systems, 2(1):31-63, 2006.

[^0]:    ${ }^{a}$ A two-page short abstract was presented at AQIS 2010. This version includes all details of design and analysis of the proposed adder.
    ${ }^{b}$ Corresponding Author, bschoi3@gmail.com
    ${ }^{c}$ rdv@sfc.wide.ad.jp

