A new quantum ripple-carry addition circuit

Steven A. Cuccaro[*]   Thomas G. Draper[†]   Samuel A. Kutin[‡]

David Petrie Moulton[§]

February 1, 2008

Abstract

We present a new linear-depth ripple-carry quantum addition circuit. Previous addition circuits required linearly many ancillary qubits; our new adder uses only a single ancillary qubit. Also, our circuit has lower depth and fewer gates than previous ripple-carry adders.

1 Introduction

We present a new quantum circuit for addition. The circuit is based on the *ripple-carry* approach, in which we start with the low-order bits of the input and work our way up to the high-order bits. Since our computation must be reversible, we then work our way from the high-order bits back down to the low-order bits.

A ripple-carry adder has previously been proposed by Vedral, Barenco, and Ekert [4]. Their circuit takes two n-bit numbers as input, computes the sum in place, and outputs a single bit (the high bit of the sum). They also require $n - O(1)$ scratch qubits, or *ancillae*.

Our circuit is different in that it requires only one ancilla. Also, the depth and size of the circuit are smaller. The VBE adder is made up of $4n + O(1)$ CNOT (controlled-NOT) gates and $4n + O(1)$ Toffoli (doubly-controlled-NOT) gates, with little parallelism. Our circuit uses $2n + O(1)$ Toffoli gates, $5n + O(1)$ CNOT gates, and $2n + O(1)$ negations; the depth is $2n + O(1)$.

The key ingredient of the new adder is a circuit computing the majority of three bits in place. We present this circuit, and a simple version of the adder, in Section 2. We then give an optimized version in Section 3. See Figure 5 for a pseudocode version of the adder, and Figure 6 for a pictorial version.

[*]Center for Computing Sciences, 17100 Science Drive, Bowie, MD 20715. `cuccaro@super.org`

[†]Department of Mathematics, University of Maryland, College Park, MD 20742. `tgd@math.umd.edu`

[‡]Center for Communications Research, 805 Bunn Drive, Princeton, NJ 08540. `kutin@idaccr.org`

[§]Center for Communications Research, 805 Bunn Drive, Princeton, NJ 08540. `moulton@idaccr.org`

In Section 4, we discuss several variants of the adder: performing addition modulo 2^n, using an incoming carry bit, and computing only the high bit of the sum. This last variant can be modified to produce a comparator. The complexities of these variants are summarized in Table 1 on Page 6.

2   The basic idea

Our goal is to compute the sum of two n-bit numbers a and b. Write $a = a_{n-1} \cdots a_0$, with a_0 the lowest-order bit, and similarly write $b = b_{n-1} \cdots b_0$. We use A_i and B_i to denote the memory locations where a_i and b_i are initially stored.

We will add a and b in place; at the end, B_i will contain s_i, the ith bit of the sum. There is one additional output location, Z, for the high bit s_n.

We define the *carry string* for a and b recursively: Let $c_0 = 0$, and let $c_{i+1} = \text{MAJ}(a_i, b_i, c_i)$ for $i \geq 0$. Note that $\text{MAJ}(a_i, b_i, c_i) = a_i b_i \oplus a_i c_i \oplus b_i c_i$. We then have $s_i = a_i \oplus b_i \oplus c_i$ for all $i < n$, and $s_n = c_n$. In a classical ripple-carry adder, we compute each c_i in order, working our way from c_1 up to c_n. In a reversible ripple-carry adder, we must then erase the carry bits, working our way back down.

The first component of our adder, depicted in Figure 1, is a gate that computes the majority of three bits in place. We build our circuits out of negations, CNOTs, and Toffoli gates; time flows from left to right in our circuit diagrams. For the in-place majority, we apply first two CNOTs and then one Toffoli.

Figure 1: The in-place majority gate MAJ

The second component, depicted in Figure 2, is an "UnMajority and Add", or UMA, gate. We give two versions, each of which computes the same function on the qubits. The first is conceptually simpler, but the second admits greater parallelism.

(a) 2-CNOT version   (b) 3-CNOT version

Figure 2: Two implementations of the UMA gate

2

The effect of using these two gates together is shown in Figure 3. Suppose that we have just computed the carry bit c_i. We apply the MAJ gate, which writes c_{i+1} into A_i. We then continue our computation. After we are done using c_{i+1}, we apply the UMA gate, which restores a_i to A_i and c_i to A_{i-1} and writes s_i to B_i.

$$
\begin{array}{ccccc}
c_i - \boxed{\text{M}} & - c_i \oplus a_i - & \boxed{\text{U}} & - c_i \\
b_i - \boxed{\text{A}} & - b_i \oplus a_i - & \boxed{\text{M}} & - s_i \\
a_i - \boxed{\text{J}} & - c_{i+1} - & \boxed{\text{A}} & - a_i
\end{array}
$$

Figure 3: Combining the MAJ and UMA gates

It follows that we can string together MAJ and UMA gates to build a ripple-carry adder. Such an adder is depicted in Figure 4. We have one ancilla, labeled X, initialized to 0. We view X as containing the initial carry bit c_0. The output bit Z contains some value z when the circuit begins and $z \oplus s_n$ when the circuit concludes.
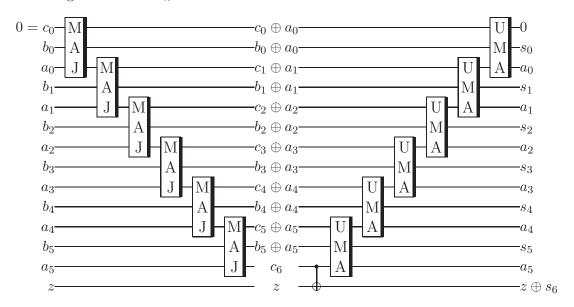

Figure 4: A simple ripple-carry adder for $n = 6$.

3 Improving the circuit

We can reduce the depth of the basic circuit of Figure 4 in several ways. It is necessary to use the 3-CNOT version of the UMA gate from Figure 2(b).

1. The first CNOTs of all the MAJ gates can be performed in a single time-slice at the beginning. Similarly, the final CNOTs of all the UMA gates can be performed in a single time-slice at the end.

$$\boxed{\begin{aligned}
&\textbf{input:} \qquad A_i = a_i \qquad B_i = b_i \qquad Z = z \qquad\qquad X = 0 \\
&\textbf{output:} \qquad A_i = a_i \qquad B_i = s_i \qquad Z = z \oplus s_n \qquad X = 0 \\
&\textbf{circuit:} \\
&\qquad \textbf{for } i = 1 \textbf{ to } n-1: \qquad B_i \oplus= A_i \\
&\qquad X \oplus= A_1 \\
&\qquad X \oplus= A_0 B_0 \quad ; \quad A_1 \oplus= A_2 \\
&\qquad A_1 \oplus= X B_1 \quad ; \quad A_2 \oplus= A_3 \\
&\qquad \textbf{for } i = 2 \textbf{ to } n-3: \\
&\qquad\qquad A_i \oplus= A_{i-1} B_i \quad ; \quad A_{i+1} \oplus= A_{i+2} \\
&\qquad A_{n-2} \oplus= A_{n-3} B_{n-2} \quad ; \quad Z \oplus= A_{n-1} \\
&\qquad Z \oplus= A_{n-2} B_{n-1} \quad ; \quad \textbf{for } i = 1 \textbf{ to } n-2: \quad \text{Negate } B_i \\
&\qquad B_1 \oplus= X \quad ; \quad \textbf{for } i = 2 \textbf{ to } n-1: B_i \oplus= A_{i-1} \\
&\qquad A_{n-2} \oplus= A_{n-3} B_{n-2} \\
&\qquad \textbf{for } i = n-3 \textbf{ down to } 2: \\
&\qquad\qquad A_i \oplus= A_{i-1} B_i \quad ; \quad A_{i+1} \oplus= A_{i+2} \quad ; \quad \text{Negate } B_{i+1} \\
&\qquad A_1 \oplus= X B_1 \quad ; \quad A_2 \oplus= A_3 \quad ; \quad \text{Negate } B_2 \\
&\qquad X \oplus= A_0 B_0 \quad ; \quad A_1 \oplus= A_2 \quad ; \quad \text{Negate } B_1 \\
&\qquad X \oplus= A_1 \\
&\qquad \textbf{for } i = 0 \textbf{ to } n-1: \qquad B_i \oplus= A_i
\end{aligned}}$$

Figure 5: The ripple-carry adder for $n \geq 4$. Each line of pseudocode corresponds to a single time-slice.
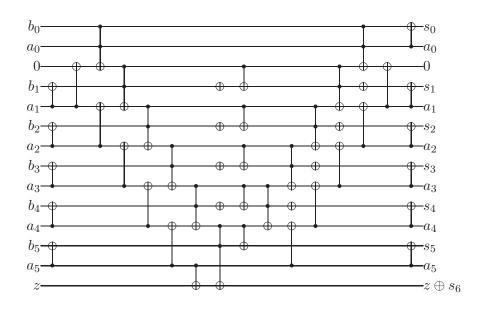

Figure 6: The ripple-carry adder for $n = 6$.

4

2. Consider the first half of the circuit: the MAJ ripple. The Toffoli at the end of the ith MAJ gate commutes with the second CNOT of the $(i+1)$th gate. If we swap these two gates for each i, then the depth decreases: the Toffoli of the ith MAJ gate can now be done in parallel with the second CNOT of the $(i+2)$th MAJ gate.

3. We can perform a similar transformation on the second half of the circuit. We swap the Toffoli of the $(i+1)$th UMA gate with the second CNOT of the ith UMA gate. Again, the depth decreases: the second CNOT of the ith UMA gate can be done in parallel with the Toffoli of the $(i+2)$th UMA gate.

4. We know $c_0 = 0$, so we do not need a MAJ gate to compute $c_1 = a_0 b_0$. Instead, we compute c_1 with a single Toffoli and store it in our ancilla. At the end of the circuit, we undo this same Toffoli, and then set B_0 to s_0 with a single CNOT.

5. It is inefficient to write c_n into A_{n-1}, copy it to the output, and then erase it. We can instead write directly to the output. We replace the central piece (two Toffolis, two CNOTs, and two negations) with one Toffoli and two CNOTs. One of the CNOTs can be done in parallel with other computation.

Our final ripple-carry circuit is described in Figure 5. The construction applies for any n, but the pseudocode in Figure 5 is valid only for $n \geq 4$. A sample circuit for $n = 6$ is depicted in Figure 6. Note that, in Figure 4, the ancilla contains c_0 and is the topmost wire; in Figure 6, the ancilla contains c_1 and is the third wire from the top.

Assuming $n \geq 2$, the circuit size is $2n - 1$ Toffoli gates, $5n - 3$ CNOTs, and $2n - 4$ negations. The depth is $2n + 4$: $2n - 1$ Toffoli time-slices and 5 CNOT time-slices.

4 Extensions

We now discuss various slightly-modified versions of the ripple-carry adder:

- modulo 2^n: We do not compute the high bit.

- incoming carry: We consider the ancilla c_0 to be an extra input bit.

- high bit only: We compute the high bit, but do not overwrite the b input. This circuit can be adapted to give a comparator.

In each case, the circuit is a simple modification of the circuit of Section 3. The only question is the exact depth and size of the circuit. The results are summarized in Table 1. For each circuit, we give the number of Toffoli gates, the number of CNOT gates, and the overall depth. In each case, the number of Toffoli time-slices is equal to the number of Toffoli gates; the remaining time-slices contain CNOTs. For the VBE adder, the circuit has $3n - 1$ Toffoli time-slices and $3n - 1$ CNOT time-slices.

5

		Number of Bits			Size		Depth
Function	IC?	In	Out	Anc.	Toffoli	CNOT	
$+$ in $\mathbb{Z}$	N	$2n$	1	1	$2n-1$	$5n-3$	$2n+4$
$+$ in $\mathbb{Z}$	Y	$2n+1$	1	0	$2n-1$	$5n+1$	$2n+6$
$+$ (mod 2^n)	N	$2n$	0	1	$2n-3$	$5n-7$	$2n+2$
$+$ (mod 2^n)	Y	$2n+1$	0	0	$2n-3$	$5n-3$	$2n+4$
Compare	N	$2n$	1	1	$2n-1$	$4n-3$	$2n+3$
Compare	Y	$2n+1$	1	0	$2n-1$	$4n+1$	$2n+5$
VBE adder [4]	N	$2n$	1	n	$4n-2$	$4n-2$	$6n-2$

Table 1: Circuit summary, for $n \geq 3$. The first column gives the function being computed. The second lists whether we take an incoming carry bit as input. We then list the number of input, output, and ancilla bits, the number of Toffoli and CNOT gates, and the overall depth. We do not include negations when counting size or depth.

4.1   Addition Modulo 2^n

Suppose that we wish to compute $a + b$ (mod 2^n); that is, we do not want to compute the high bit c_n. One approach is the following:

1. Add the low-order $n-1$ bits of a and b, using the circuit of Section 3. Use B_{n-1} as the output bit.

2. Set $B_{n-1} \oplus= A_{n-1}$.

After step 1, we have correctly computed s_0 through s_{n-2}, and we have written $b_{n-1} \oplus c_{n-1}$ into B_{n-1}. Then, in step 2, we complete the calculation of s_{n-1}. Note that step 2 occurs in parallel with the final time-slice of step 1.

For $n \geq 3$, this circuit contains $2n-3$ Toffolis, $5n-7$ CNOTs, and $2n-6$ negations. The depth is $2n+2$: $2n-3$ Toffoli time-slices and 5 CNOT time-slices.

4.2   Addition with Incoming Carry

Suppose we want to allow an incoming carry into our addition circuit. We have an additional input bit y, and we compute $a + b + y$.

We observe that the circuit of Section 2 already solves this problem; we use y in place of the ancilla c_0. We then correctly compute c_1, and the ripple continues.

We cannot use the fourth improvement from Section 3, since we can no longer assume the incoming bit is zero. The other improvements still apply.

We obtain a ripple-carry adder with incoming carry which consists of $2n - 1$ Toffolis, $5n + 1$ CNOTs, and $2n - 2$ negations. For $n \geq 2$, the circuit has depth $2n + 6$: $2n - 1$ Toffoli time-slices and 7 CNOT time-slices.

We can also apply the incoming-carry modification to the circuit of Section 4.1. For $n \geq 3$, we get a circuit with $2n - 3$ Toffolis, $5n - 3$ CNOTs, and $2n - 4$ negations. The depth is $2n + 4$: $2n - 3$ Toffoli time-slices and 7 CNOT time-slices.

4.3 High Bit Only

We now consider the problem of computing only the high bit of the sum $a + b$. The first half of the circuit is identical to the first half of our adder from Section 3: when we get to the middle point, we have written the high bit to Z. Now, we simply undo the first half of the circuit. We can view this as applying a series of MAJ gates, followed by a Toffoli and a series of MAJ^{-1} gates.

For $n \geq 2$, the resulting circuit contains $2n - 1$ Toffoli gates and $4n - 3$ CNOTs. The depth is $2n + 3$: $2n - 1$ Toffoli time-slices and 4 CNOT time-slices.

We can combine the high-bit circuit with the incoming-carry modification discussed in Section 4.2. We obtain a circuit with $2n - 1$ Toffolis and $4n + 1$ CNOTs. For $n \geq 2$, the depth is $2n + 5$: $2n - 1$ Toffoli time-slices and 6 CNOT time-slices.

It is worth noting that our ripple-carry adder can easily be turned into a subtractor. Whether we use one's-complement or two's-complement arithmetic, we have the identity

$$a - b = (a' + b)',$$

where $'$ denotes bitwise complementation. Hence, we can subtract by adding two time-slices: complement a at the start, and complement a and s at the end.

If we combine this subtraction idea with the high-bit computer of this section, we obtain a comparator: we compute the high bit of $a - b$, which is 1 if and only if $a < b$.

5 Conclusions

One interesting open problem is to construct an optimal addition circuit. In particular, if a reversible addition circuit uses just one ancilla, must it have linear depth? A logarithmic-depth adder has been constructed using $2n$ ancillae [3]; more generally, for any $k > 0$, we can construct a family of circuits using n/k ancillae with depth $O(k + \log n)$. Is there a logarithmic-depth addition circuit family using only a constant number of ancillae? If not, can we prove a lower bound on depth?

A version of our ripple-carry adder has been proposed that uses no ancillae [1]. That circuit requires that the output bit be initialized to zero. We do not know whether we can add in linear depth with no ancillae and without this restriction on the output bit.

It would be interesting to compare the ripple-carry adder of this paper to the transform adder [2]. Both circuits have linear depth. It is unclear which adder would be easier to

implement in practice; the answer depends on the relative costs of Toffoli gates and controlled rotations.

It is well-known that a Toffoli gate can be built from five controlled rotations. One might thus expect the controlled-unary depth of our ripple-carry adder to be $10n + O(1)$. In fact, the Toffolis can be overlapped; the depth is only $6n - 2$. An example with $n = 5$ is depicted in Figure 7.

We can also consider the cost of adding a classical quantity to a quantum quantity. We have some n-bit number in our quantum memory, and we wish to add a fixed n-bit number (known at compile time). Our ripple-carry adder does not become any simpler in this setting; we still need to use n quantum bits to store the classical addend. On the other hand, the transform adder benefits greatly: the classical information need not be stored in quantum memory, and the controlled rotations are replaced with fixed and known rotations. In this setting, the transform adder seems superior.

References

[1] Richard J. Dore and Samuel A. Kutin, *A logarithmic-depth quantum comparison circuit with one ancilla*, in preparation.

[2] Thomas G. Draper, *Addition on a quantum computer*, quant-ph/0008033.

[3] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore, *A logarithmic-depth quantum carry-lookahead adder*, EQIS, 2004, quant-ph/0406142.

[4] Vlatko Vedral, Adriano Barenco, and Artur Ekert, *Quantum networks for elementary arithmetic operations*, quant-ph/9511018.

Figure 7: 5-bit ripple carry adder written in terms of controlled rotations. The depth is 28. Here a circled i denotes a "square root of NOT"; i.e., a rotation by $\pi/2$. A circled $-i$ denotes the inverse operation.