Tight Bounds on Expected Time to Add Correctly and Add Mostly Correctly

Peter Gemmell* Mor Harchol†

April 16, 1993

Abstract

We consider the problem of adding two n-bit numbers which are chosen independently and uniformly at random where the adder is circuit of AND, OR, and NOT gates of fanin two.

The fastest currently known worst-case adder has running time $\log n + O(\sqrt{\log n})$ [Khrapchenko].

We first present a circuit which adds at least $1-\epsilon$ fraction of pairs of numbers correctly and has running time $\log\log(\frac{n}{\epsilon}) + O(\sqrt{\log\log(\frac{n}{\epsilon})})$.

We then prove that this running time is optimal.

Next we present a circuit which *always* produces the correct answer. We show this circuit adds two n-bit numbers from the uniform distribution in expected $\frac{1}{2}\log n + O(\sqrt{\log n})$ time, a speed up factor of two over the best possible running time of a worst-case adder.

We prove that this expected running time is optimal.

*Computer Science Division, UC Berkeley, CA 94720. Supported by NSF grant number CCR-9201092.

†Computer Science Division, UC Berkeley, CA 94720. Supported by National Physical Science Consortium (NPSC) Fellowship.

1 Introduction

Khrapchenko [Khrapchenko] exhibited a circuit of AND, OR, and NOT gates which could add any two n bit numbers in depth $\log(n)+O(\sqrt{\log(n)})$. This nearly matches the obvious $\log(n)$ lower bound for worst-case running time.

In this paper, we consider the issue of *expected* time to add two n bit numbers, where the numbers are chosen independently from the uniform distribution. Because most additions do not involve long propagations of carries, we can achieve considerable savings in time over worst-case adders.

We will design an adder circuit with optimal expected running time, $(\frac{1}{2}\log(n)+O(\sqrt{\log(n)}))$ which we call a *Fast Adder*. This Fast Adder circuit will consist of two parts. We call the first part a *Near Adder*. A Near Adder is a very fast adder, which is correct on all but an ϵ fraction of pairs of n bit numbers. In *section* 2, we construct the Near Adder, and in *section* 3, we prove that the Near Adder is optimal.

The second part of the Fast Adder circuit is a *Checker* which quickly deduces whether the Near Adder has done the addition correctly. The idea of a checker was introduced by Blum [Blum,Kannan]. If the checker determines the addition to be correct the Fast Adder outputs the answer determined by the Near Adder. If however the checker determines that the addition may have been incorrect, the addition is redone using the slow-but-sure conventional adder. It is important to note that the result of the Fast Adder is therefore guaranteed to be correct.

The Fast Adder circuit will be described in detail in *section* 4. The Near Adder computation can be overlapped with the checking computation. The total expected running time for the Fast Adder is then:

$$max(Time_{\text{Near Adder}}, Time_{Checker}) + Time_{\text{conventional adder}} \cdot \Pr[\text{Near Adder incorrect}]$$

In *section* 5 we prove that the expected running time of the Fast Adder is optimal, namely that no circuit for adding two n-bit integers chosen independently from the uniform distribution, has better expected running time. [1]

Our bound on computation time assumes unit gate delay for each gate, fanin of 2, and unbounded fanout.

2 Near Adder Circuit for Adding Most Numbers Correctly Quickly

In this section we show how to convert a conventional adder of two n-bit numbers into a much faster, but sometimes unreliable adder, which we call a Near Adder. The Near Adder circuit is fast, yet incorrect on a small (ϵ) fraction of the inputs. Near Adders take advantage of the property

[1] There are circuits which produce the correct sum in expected $\log\log(n)+O(\sqrt{\log\log(n)})$ time. However, these circuits do not also produce a guarantee that the output is correct for the particular input in expected sub $\frac{1}{2}\log(n)$ time. For those familiar with Blum's work on waffling [Blum], one could say that the expected waffling time of addition is at most $\log\log(n)+O(\sqrt{\log\log(n)})$. Here, a waffling circuit is one which continually outputs an answer. The waffling time is the time required to stabilize on the correct output.

that, for most inputs, each of the output bits depends only on a small number of adjacent input bits.

Theorem 1 *For all $\epsilon > 0$, there exists a Near Adder that has depth $\log\log(\frac{n}{\epsilon}) + O(\sqrt{\log\log(\frac{n}{\epsilon})})$ and that is correct on all but ϵ fraction of pairs of n bit inputs.*

Proof:

The structure of the Near Adder we propose is shown in figure 1. Given two n-bit numbers, the Near Adder divides them into $\frac{n}{d}$ blocks of size d-bits each. The Near Adder then uses the conventional adder to add consecutive 2d-bit blocks in parallel as shown in figure 1. When adding each of these 2d-bit blocks, the Near Adder assumes the carry-in to the $2d$-bit block is zero. The Near Adder returns the most significant d bits of each of these $2d$-bit summands (the unshaded parts) as the sum of the two n-bit numbers.

The running-time of the Near Adder is the time it takes for the conventional adder to add two $2d$-bit numbers. Using Khrapchenko's [Khrapchenko] circuit, this time is $\log(d) + O(\sqrt{\log(d)})$.

The probability of error in the Near Adder is the probability that for any of the n 2d-bit summands, the 2d-bit summand should have had a carry-in of 1 and the second d bits of the 2d-bit summand (the shaded part) consisted of all 1's. That is, the probability of error is the probability that for some 2d-bit summand the carry should have been 1 and it would have propagated through the lower order d-bits, making a change to the higher order d-bits which we keep as our answer. So,

$$\begin{aligned}\Pr[\text{ error in Near Adder}] &= (\text{no. 2d-bit summands}) \cdot (\Pr[\text{ carry-in 1}]) \cdot (\Pr[\text{ carry propagated via lower d bits}]) \\ &= (\frac{n}{d} - 2) \cdot (\frac{1}{2}) \cdot (\frac{1}{2^d})\end{aligned}$$

In order for to achieve the depth and error bounds described in theorem (1), we assign the block size to be $2d = 2\log(\frac{n}{\epsilon})$.

■

3 Lower Bound on Time To Add Most Numbers Correctly

In this section we determine a lower bound on the depth, d, of a circuit which adds 2 n-bit numbers with confidence $1 - \epsilon$. We will show that d must be $> \log\log(\frac{n}{\epsilon})$.

We divide the n-bit inputs and n bit output (ignoring a possible final carry) into $\frac{n}{2^d+1}$ blocks of size $2^d + 1$.

Denote by $block_1$ the leftmost block and denote by b_1 the most significant output bit of $block_1$. Because the circuit is restricted to having depth d, we know that there is at least one pair of input bits in $block_1$ such that neither of these input bits affects the value of the bit b_1. Let E_1 be the

event that these two input bits are either both 0 or both 1 and that the output bit b_1 is incorrect. Now throw away $block_1$ and all blocks such that b_1 depends on some inputs from those blocks. Note that we throw away at most $2^d + 1$ blocks in this step.

Denote by $block_2$ the leftmost remaining block and denote by b_2 the most significant output bit of $block_2$. We know that there is at least one pair of input bits in $block_2$ such that neither of these input bits affects the value of the bit b_2. Let E_2 be the event that these two input bits are either both 0 or both 1 and that the output bit b_2 is incorrect. Now throw away $block_2$ and all blocks such that b_2 depends on some inputs from those blocks.

We define events $E_1 \ldots E_{\frac{n}{(2^d+1)^2}}$ in this way.

Analyzing the probability of propagation-related error, we have:

$$\forall i : 1 \leq i \leq \frac{n}{2^d+1}, Pr[E_i] \geq \frac{1}{4}\frac{1}{2^{2^d}}$$

Furthermore, the events E_i are independent which yields:

$$\epsilon \geq 1 - \prod_{i=1}^{\frac{n}{(2^d+1)^2}} (1 - \frac{1}{2^{2^d+2}}) \approx \frac{n}{2^{2d}}\frac{1}{2^{2^d+2}}$$

$$\Rightarrow d \geq \log\log(\frac{n}{\epsilon})$$

4 Fast Adder Circuit For Adding Correctly in Fast Expected Time

In this section we construct a Fast Adder, which adds all numbers correctly, but has low expected running time. The Fast Adder circuit consists of two subcircuits which are run in parallel. The first subcircuit is the Near Adder described in *section* 2. The second subcircuit is a Checker which will determine if the output of the Near Adder is correct. If the Checker circuit determines the output of the Near Adder to be correct, the Fast Adder outputs the output of the Near Adder. If, on the other hand, the Checker circuit determines the output of the Near Adder to be incorrect, Khrapchenko's [Khrapchenko] worst-case adder will be run to determine the output of the Fast Adder. In *section* 4.1 below, we describe the Checker, and in *section* 4.2 we analyze the expected running time of the Fast Adder. Note that it is important to set parameters in the Fast Adder such that the running time of the Checker is low, and such that the probability of error in the Near Adder is very low.

4.1 The Design of a Checker for our Near Adder

We now show how to build a checker for Near Adder whose computation can be overlapped with the computation of the Near Adder. Our checker will alert us if the carry could have propagated

though the lower d bits of any of the 2d-bit blocks. For the sake of speed, our checker will not examine all the lower d bits of any block. Instead, it will examine a small number, c, of pairs of bits in each block. Also, our checker will not determine whether the carry-in should have been a 1 not a 0, but rather our checker will simply assume that the carry-in was a 1.

Our checker is illustrated in figure 2. It takes as its inputs the 2 n-bit operands. It then uses XOR gates (denoted by X), to check if two corresponding positions in the 2 numbers create a propagate. The checker must output $FAIL$ if the d bits making up the shaded part of some $2d$-bit block would propagate a carry. If the subblock of c-bits would propagate a carry, the corresponding AND gate is turned on and the checker in figure 2 outputs $FAIL$.

$$\Pr\left[\text{Checker detects a possible error}\right] \leq (\frac{n}{d} - 2) \cdot (\frac{1}{2^c})$$

The running time for the checker is constant to do all the XOR operations in parallel plus $\log c$ to do all the AND operations plus $\log(\frac{n}{d} - 2)$ to do the NOR operation.

$$\begin{aligned} \text{Running time for checker} &= 1 + \log c + \log(\frac{n}{d} - 2) \\ &\leq 1 + \log c + \log n - \log d \end{aligned}$$

To achieve a reasonably low probability of having to fail the Near Adder and a reasonably low circuit depth for the checker, we set $c = \log(n)$

Note that the computation of the checker in no way interferes with the computation of the Near Adder.

4.2 Combining the Near Adder and Checker into the Fast Adder

In this section we combine the Near Adder and the Checker described earlier to create a Fast Adder which outputs the correct output on all inputs.

The total expected running time for a probabilistic adder is :

$max(Time_{NearAdder}, Time_{Checker}) + Time_{convadder} \cdot Pr[$ checker determines Near Adder addition may be incorrect$]$

$$\|$$

$$max(\log(d) + O(\sqrt{\log(d)}), 1 + \log c + \log n - \log d) + (\log(n) + O(\sqrt{\log(n)})) \cdot (\frac{n}{d} - 2)(\frac{1}{2^c})$$

By setting the block size, $2d$, to be $2\sqrt{n}$ and the number of bits checked per block, c, to be $\log(n)$, we get the expected time to compute the sum of two n bit numbers to be $\frac{1}{2}\log(n) + O(\sqrt{\log(n)})$.

5 Lower Bound on Expected Time to Add Correctly

In this section we prove that any adder for 2 n-bit numbers, which are independently chosen from the uniform distribution must have expected running time $\geq \frac{1}{2}\log n$.

Let $T = E[\text{time to add with certainty}]$.

Divide the summands and output into blocks of size $2^T + 1$. Let b_i be the most significant bit in the i^{th} block.

Any circuit which adds with certainty must have a checker output bit.

Claim 2 *This checker output bit must be connected (either directly or indirectly) to at least one bit from each block.*

Proof: Since the adder circuit has depth T, each output bit can be connected to $\leq 2^T$ input bits per block. Therefore, there exists at least one bit in each block that isn't looked at by the checker output bit of the adder. Now assume we are given inputs in which all the bits in the block to the left of this bit that isn't looked at are propagates. Then we can set the bit not examined such that b_i is correct, or incorrect. In both cases the checker output bit will output correct, which is a contradiction.

Therefore, we have:

$$\begin{aligned} T &\geq \log(\text{number of bits examined}) \\ &= \log(\frac{n}{2^T}) \\ &= \log n - T \end{aligned}$$

Therefore, $T \geq \frac{1}{2}\log n$.

■

References

[Blum] M. Blum. Designing Programs to Check their Work. Submitted to the *CACM* for publication.

[Blum,Kannan] M. Blum, S. Kannan. Unbounded Programs that Check their Work. *21st Symposium on the Theory of Computation*, Seattle, 1989.

[Khrapchenko] V.M. Khrapchenko. Asymptotic Estimation of Addition Time of a Parallel Adder. *Systems Theory Research* vol. 19, pp. 107-125, 1967.

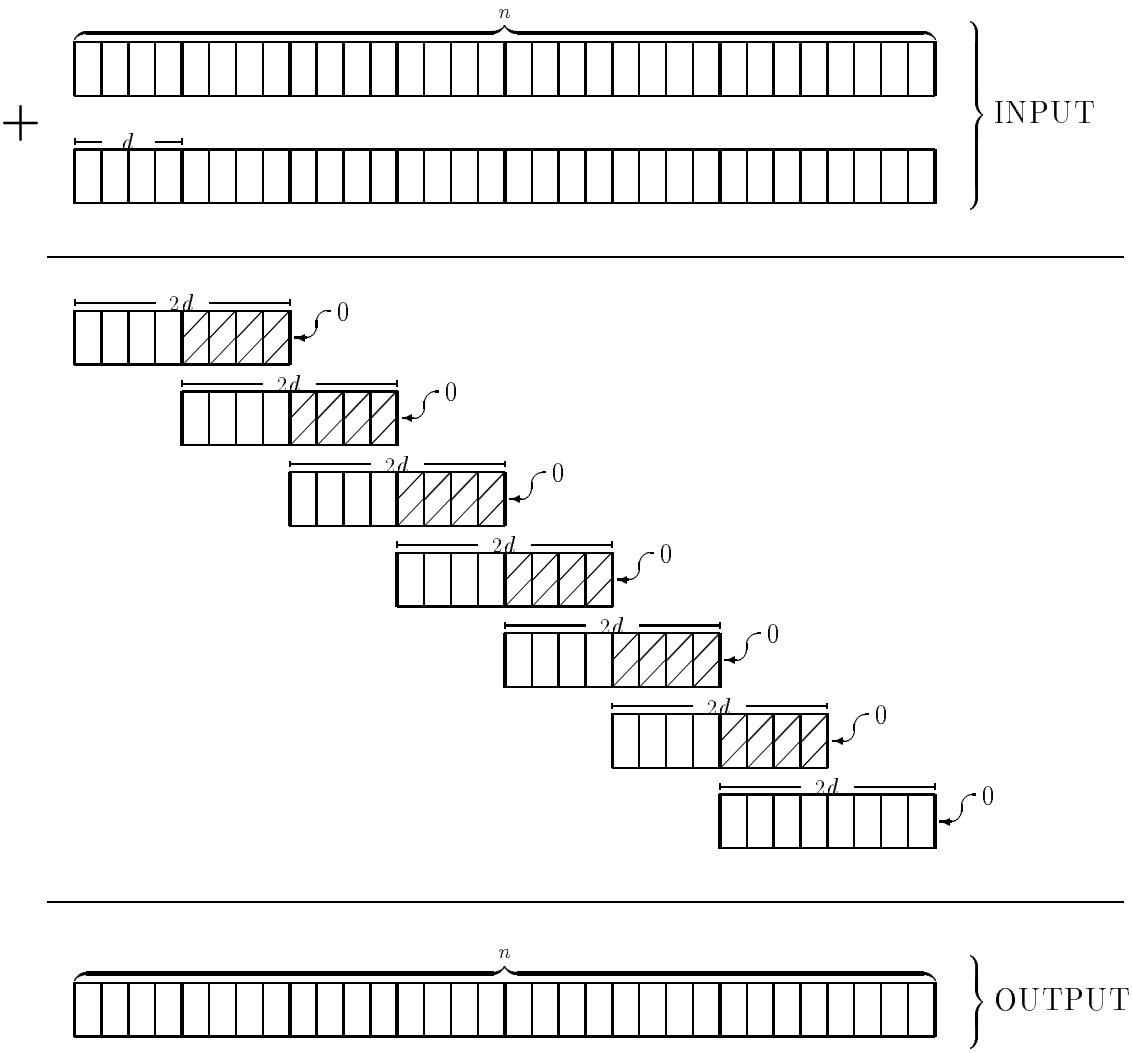

Figure 1: Near Adder

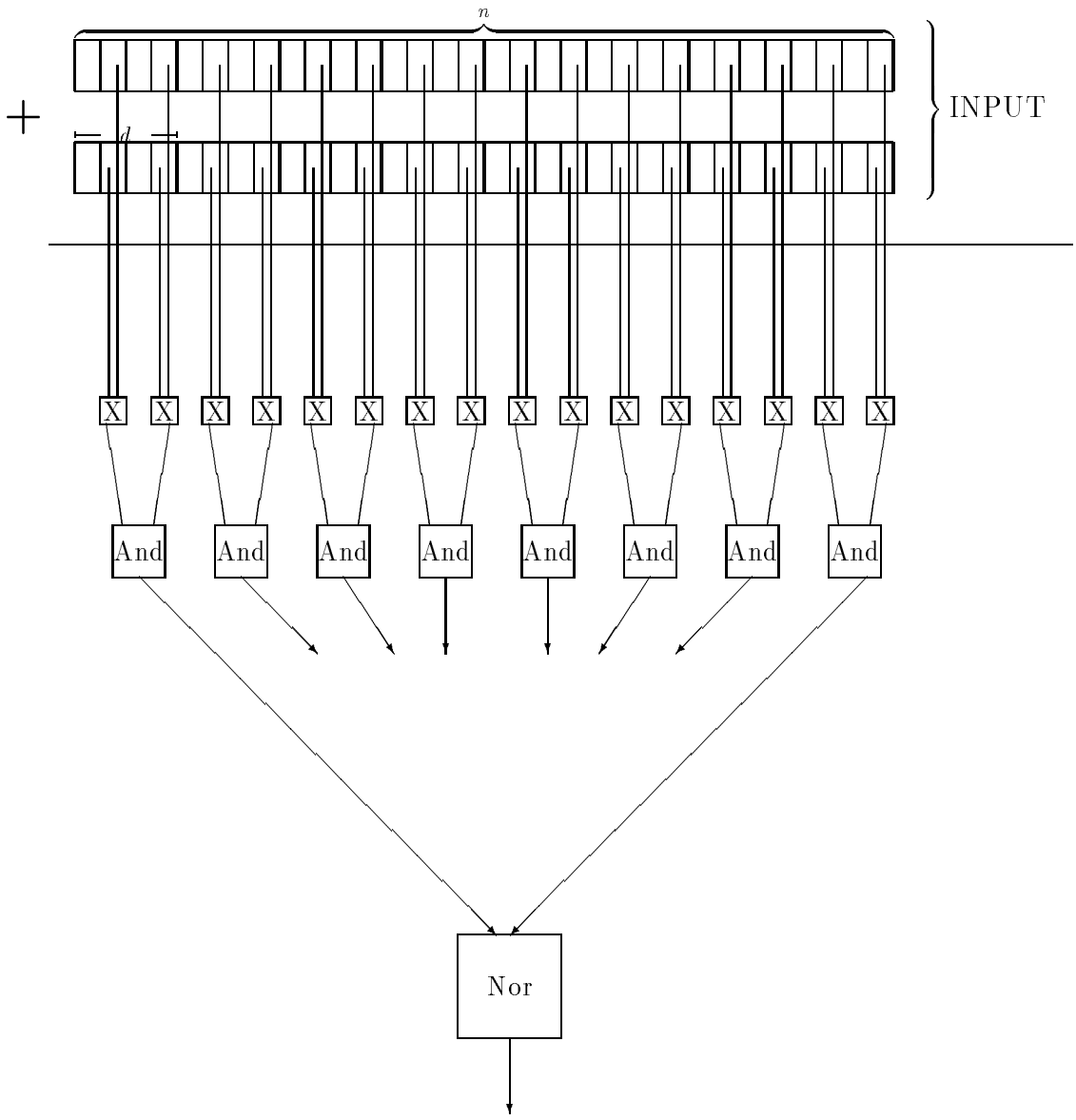

Figure 2: Checker for Near Adder