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

Peter Gemmell[*] Mor Harchol[†]

October 7, 1993

Abstract

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

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

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. Also supported by NSF grant number CCR-9201092

1 Introduction

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 unit gate delay for each gate, fan-in of 2, and unbounded fan-out.

The fastest currently known worst-case adder is due to Khrapchenko and has running time $\log n + O(\sqrt{\log n})$ [Khr]. This nearly matches the obvious $\log(n)$ lower bound for worst-case running time.

In Section 2, we present a circuit, which we call a *Near Adder*, which adds at least $1-\epsilon$ fraction of pairs of n-bit numbers correctly and has running time $\log\log(\frac{n}{\epsilon}) + O(\sqrt{\log\log(\frac{n}{\epsilon})})$. Because most additions do not involve long propagations of carries, we can achieve considerable savings in time over worst-case adders.

In Section 3, we prove that our Near Adder's running time is optimal.

In Section 4, we describe a model of for circuits, which always produces *correct* outputs, in which the circuit may have different running times for different inputs, and where the circuit must produce a signaling bit which indicates that it has finished. We then present a *Fast Adder*, which is a circuit corresponding to the above model, for adding two n-bit numbers. We show the Fast Adder 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. The Fast Adder circuit combines a version of the Near Adder with a Checker which quickly deduces whether the Near Adder has done the addition correctly.[1] If the Checker determines that the addition may have been incorrect, the addition is redone using the slow-but-sure conventional adder.

In Section 5, we prove that the expected running time of the Fast Adder circuit is optimal, namely that no circuit producing the sum of two n-bit integers chosen independently from the uniform distribution and a bit signaling that the answer is correct has better expected running time.

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

Definition 1 *Throughout this paper when adding $a_1a_2\ldots a_n$ to $b_1b_2\ldots b_n$, when we refer to a* **propagate pair** *we mean a pair of bits (a_i, b_i) such that either $a_i = 1$ and $b_i = 0$, or $a_i = 0$ and $b_i = 1$.*

[1] Note that our Checker is not a checker in the [Blum], [Blum,Kannan] sense, but really operates more like a mask for certain "problem inputs". (See Section 4.1).

Theorem 2 *For all $\epsilon > 0$, there exists a Near Adder that has depth* $\log\log(\frac{n}{\epsilon}) + O(\sqrt{\log\log(\frac{n}{\epsilon})}) + 1$ *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 [Khr] circuit, this time is $\log(d) + 1 + O(\sqrt{\log(d)})$.

The Near Adder produces an incorrect output if for any of the d-bit input blocks (excluding the first and last blocks), the block consists exclusively of propagate pairs, and the carry-in to the block is a 1. The problem here is that the carry-in is propagated through at least d bits, so that it goes past the shaded part of the $2d$-bit summand and into the output of the Near Adder.

$$\begin{aligned}
& \text{Pr[Error in Near Adder]} \\
\leq\ & (\text{no. d-bit blocks}) \cdot (\text{Pr[carry-in 1]}) \cdot (\text{Pr[All pairs in block are propagate pairs]}) \\
=\ & (\frac{n}{d}) \cdot (\frac{1}{2}) \cdot (\frac{1}{2^d})
\end{aligned}$$

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

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 $1 - \epsilon$.

Theorem 3 *For any circuit which adds two n-bit numbers with confidence $1 - \epsilon$, the depth d must be at least* $\lg\lg(\frac{n}{2\epsilon}) - 1$.

Proof:

Let d be the depth of any circuit which adds two n-bit numbers with confidence $1 - \epsilon$. Assume the 2 n-bit inputs are independently and uniformly distributed.

We divide the n-bit numbers into $\frac{n}{2^d+1}$ blocks, each of size $2^d + 1$. We also divide the output bits into blocks of size $2^d + 1$. Denote by $block_1$ the rightmost (least significant) input block and denote by b_1 the most significant output bit of this block. Denote by $block_i$ the 2^{d+1} input bit pairs associated with the ith rightmost (least significant) block and denote by b_i the most significant output bit of the ith block. Let cb_i equal the correct value of the most significant output bit of the ith block.

Let E_i be the event that b_i is not equal to cb_i.

The proof has three main parts:

1. We construct a set S, $|S| \geq \frac{n}{(2^d+1)^2}$, such that $\forall i, j \in S, j < i$, output bit b_j is not connected to any input bit from $block_i$.

   To do this, we start by putting 1, the index for the least significant block, in S. Then we throw out $block_1$ and all the blocks on which b_1 depends. Next, we put in S the index of the least significant remaining block. Then we throw out this block and all the blocks on which the most significant output bit of that block depends. We repeat this process until there are no blocks remaining. Because we throw out at most $2^d + 1$ blocks for every block we place in S, we have $|S| \geq \frac{n}{(2^d+1)^2}$.

2. We observe that $block_i$ is independent of event E_j, $\forall j < i$, where $i, j \in S$.

   This follows from the following argument: Let $j < i$ and $i, j \in S$. Let all input bits be set arbitrarily. Now look at b_j and cb_j for this setting. Either we have $b_j = cb_j$, or we have $b_j \neq cb_j$. Now altering the bits in $block_i$ can't affect cb_j by definition of addition. Altering the bits in $block_i$ can't affect b_j because, by definition of S, b_j is not connected to $block_i$.

3. We show that $\forall i \in S$, $Pr[E_i | \bigcap_{j<i; j \in S} \overline{E}_j] \geq \frac{1}{2^{2^d+1}}$.

   Because the circuit is restricted to having depth d and because the block size is $2^d + 1$, we know that there is at least one pair of input bits in $block_i$ such that neither of these input bits affects the value of the output bit b_i. Let p_i denote the leftmost such pair. If p_i is the kth pair of bits in $block_i$ (looking from left to right), then there will be an error if the first $k - 1$ pairs (looking from left to right) are all propagates and either the kth pair is $(1, 1)$ or $(0, 0)$. (If $b_i = 1$, the pair $(0, 0)$ implies $cb_i = 1$; if $b_i = 0$, the pair $(1, 1)$ implies $cb_i = 0$). So $Pr[E_i] \geq \frac{1}{2^{k+1}} \geq \frac{1}{2^{2^d+1}}$. Since by (2) above, the setting of $block_i$ is independent of event $E_j, \forall j < i$, we have $Pr[E_i | \bigcap_{j<i; j \in S} \overline{E}_j] = Pr[E_i] \geq \frac{1}{2^{2^d+1}}$.

The probability of union of the events E_i lower bounds the total error ϵ and this yields a lower bound on the depth:

$$\begin{aligned}
\epsilon &\geq \text{Pr[Error in Output]} \\
&\geq Pr[\bigcup_{i \in S} E_i] \\
&= 1 - Pr[\bigcap_{i \in S} \overline{E}_i] \\
&\geq 1 - (1 - \frac{1}{2^{2^d+1}})^{\frac{n}{(2^d+1)^2}}
\end{aligned}$$

$$\begin{aligned}
&= 1-(1-\frac{1}{2^{2^d+1}})^{2^{2^d+1}\cdot\frac{n}{(2^d+1)^2}\cdot\frac{1}{2^{2^d+1}}} \\
&\geq 1-e^{-\frac{n}{(2^d+1)^2}\cdot\frac{1}{2^{d}+1}} \text{ (by } (1-\tfrac{1}{k})^k \leq e^{-1}) \\
&\geq 1-e^{-\frac{n}{2^{2^{d+1}}}} \text{(true for } d \geq 3)
\end{aligned}$$

So,

$$\begin{aligned}
\epsilon - 1 &\geq -e^{-\frac{n}{2^{2^{d+1}}}} \\
1-\epsilon &\leq e^{-\frac{n}{2^{2^{d+1}}}} \\
\ln(\frac{1}{1-\epsilon}) &\geq \frac{n}{2^{2^{d+1}}} \\
\frac{n}{\ln(\frac{1}{1-\epsilon})} &\leq 2^{2^{d+1}} \\
d &\geq \lg\lg(\frac{n}{\ln(\frac{1}{1-\epsilon})})-1 \\
d &\geq \lg\lg(\frac{n}{2\epsilon})-1 \text{ (true for } \epsilon \leq \tfrac{1}{2})
\end{aligned}$$

■

4 Fast Adder Circuit For Adding Correctly in Fast Expected Time

We consider a model of circuits with variable-running times. We will call such circuits VRTC (Variable Running Time Correct) circuits because they may have different running times on different inputs, but are guaranteed to produce the correct answer. The model for these circuits is as follows:

Definition 4 *An* VRTC *circuit C for a function f takes f's inputs as its inputs and has the outputs for f as well as one extra bit called a* signal *bit. The signal bit is set to* 0 *initially; at some time after the inputs are introduced, it must be set to* 1. *When the signal bit is set to* 1, *the output bits must be correct.*

We assume that the circuit C may incorporate a clock of some kind (perhaps a chain of gates each of unit delay) so that it may select bits from different subcircuits to output at different times.

We will derive upper and lower bounds on the expected running time of VRTC circuits for adding two n-bit numbers (by running time we mean time until the signal bit is set to 1). In this section we give an upper bound, by describing a VRTC circuit which we call the Fast Adder, which has expected running time $\frac{1}{2}\log(n)+O(\sqrt{\log n})$. In Section 5 we prove a lower bound of $\frac{1}{2}\log(n)-1$ on the expected running time of any VRTC addition circuit.

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, the Fast Adder uses the third subcircuit, Khrapchenko's [Khr] worst-case adder, to determine the output of the Fast Adder. In subsection 4.1 below, we describe the Checker, and in subsection 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

Our Checker is very different from the class of checkers described in [Blum,Kannan], [Blum]. It merely checks if the input is of a particular nice form, and it's output does not depend on the Near Adder's result at all. The Checker will always output FAIL if the input is of a form that will cause the Near Adder to add incorrectly. However, the Checker may output FAIL even if the computation of the Near Adder is correct.

Recall that the input to the Near Adder is divided into $\frac{n}{d}$ d-bit blocks. The Checker is based on the following observation: *If any input block consists of all propagate pairs and the carry-in to that block is a* 1*, then the Near Adder output will be incorrect* (This is actually true for any block except the first and last). Therefore, ideally, the checker should check if any input block consists only of propagate pairs. For the sake of speed, our Checker will only examine c (arbitrarily chosen) pairs in each block. If for any block all c pairs examined were propagate pairs, the Checker will output FAIL. Also, the Checker ignores the carry-in altogether.

Our Checker is illustrated in Figure 2. It takes as its inputs the two n-bit operands. It then uses XOR gates (denoted by X), to check if pairs of bits are propagate pair. An AND gate is then used to check if all c pairs within a block are propagate pairs. The Checker outputs FAIL if for any block all c pairs examined were propagate pairs.

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

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 $\log(\frac{n}{d})$ to do the NOR operation.

$$\text{Running time for checker} = 1 + \log c + \log n - \log d$$

Note that the computation of the Checker in no way interferes with the computation of the Near Adder, and therefore the computation of the Checker may be overlapped with the computation of the Near Adder.

4.2 Combining the Near Adder and Checker into the Fast Adder

The Fast Adder Circuit consists of first running the Checker and Near Adder circuits in parallel. If the Checker outputs PASS, then the Near Adder computation must be correct, so the signal bit goes on. If the Checker outputs FAIL, then the conventional adder is run on the input, and the signal bit doesn't go on until the conventional adder computation is complete.

Theorem 5 *The expected running time of the Fast Adder Circuit is* $\frac{1}{2}\log(n) + O(\sqrt{\log(n)})$.

Proof:

The total expected running time for the Fast Adder is :

$$max(Time_{NearAdder}, Time_{Checker}) + Time_{convadder} \cdot \; Pr[\text{ Checker outputs FAIL}]$$

$$||$$

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

By setting the block size, d, to be $\sqrt{n}$ 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:

Theorem 6 *Any fan-in 2 VRTC circuit adder for* 2 *n-bit numbers, which are independently chosen from the uniform distribution must have expected running time* $\geq \frac{1}{2}\log n - 1$.

Proof:

Given any VRTC addition circuit, let T denote its expected running time. Suppose that $T < \frac{1}{2}\log n - 1$. Then there exists a setting of the input bits which causes the signal bit of the adder to go on after $< \frac{1}{2}\log n$ time.

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. Observe that there are more than $\sqrt{n}$ blocks.

Since at most $2^T < \sqrt{n}$ bits can influence the signal bit of the adder by time T, there is at least one block, $block_i$, such that no bit in that block influences the signal bit by time T.

Also, since at most 2^T bits can influence b_i (the most significant output bit in $block_i$), there is at least one pair, p_i, of input bits in $block_i$ which doesn't influence b_i.

Now we know from the assumption that there exists a setting of the input bits which causes the signal bit to go on after T time. Let's change just $block_i$ of this setting such that all pairs of bits within $block_i$ are propagates, except for p_i. Note, altering $block_i$ doesn't change the signal bit. Even without setting p_i, the signal bit is still on, and b_i is set to some value. By setting p_i we can switch the value of b_i. That is, there is a setting of p_i which causes b_i to be wrong. In this case, the signal bit is still on, but b_i is incorrect.

Therefore, we must have that $T \geq \frac{1}{2}\log(n) - 1$. ■

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.

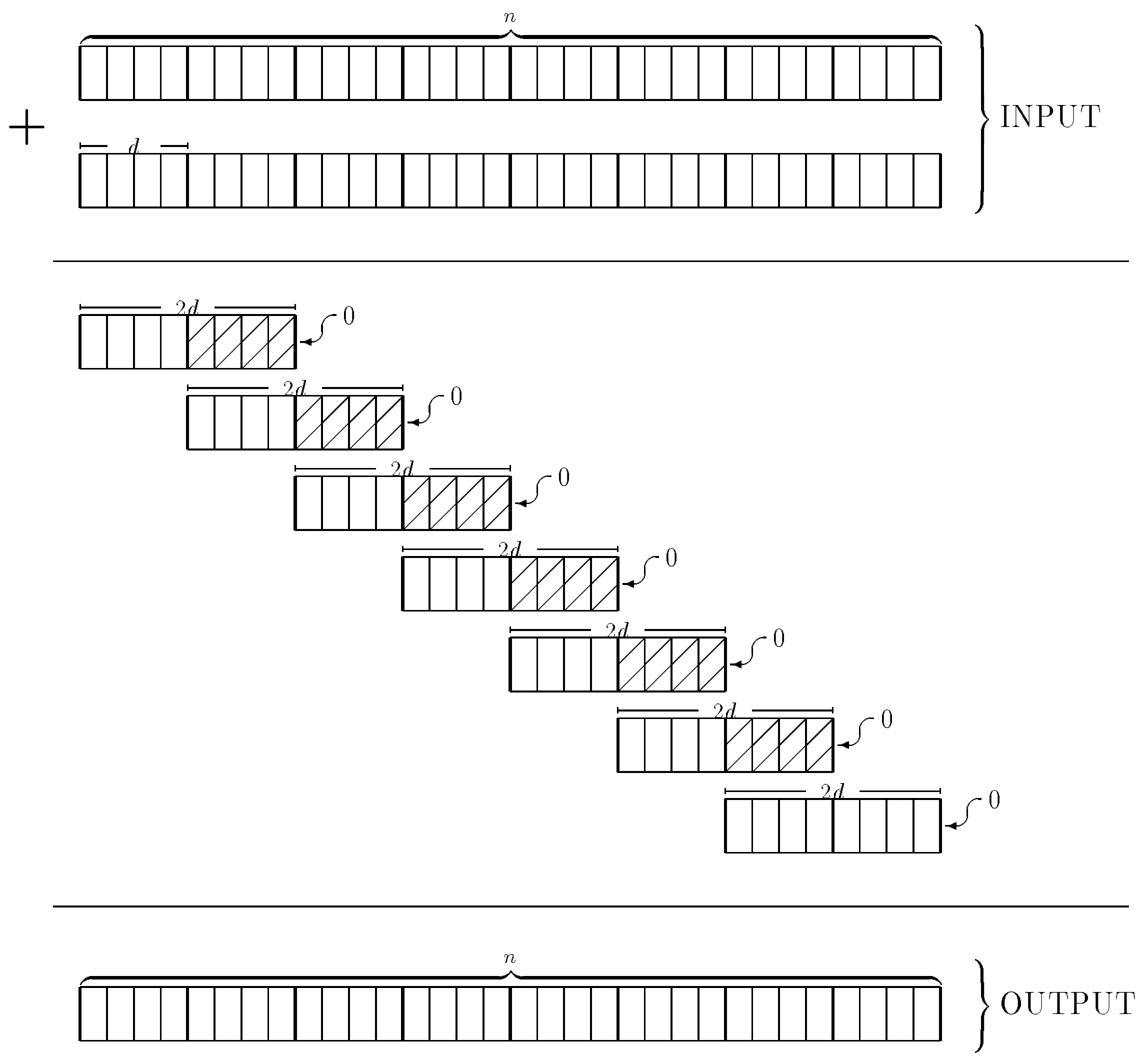

Figure 1: Near Adder

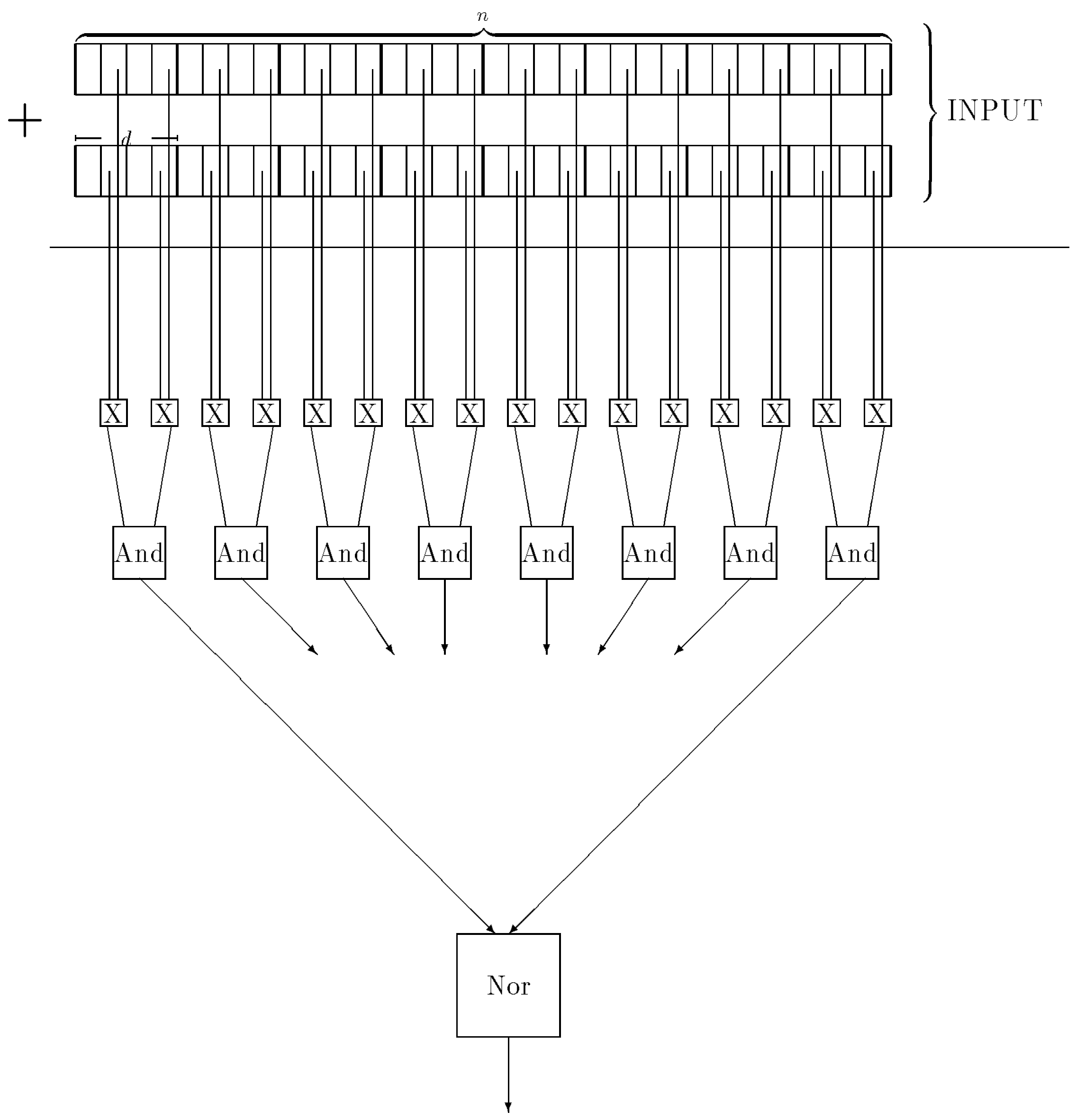

Figure 2: Checker for Near Adder