The Delay of Circuits whose Inputs have Specified Arrival Times

Dieter Rautenbach, Christian Szegedy and **Jürgen Werber**
Forschungsinstitut für Diskrete Mathematik, Lennéstr. 2, D-53113 Bonn, Germany
{rauten,szegedy,werber}@or.uni-bonn.de

Abstract. Let C be a circuit representing a straight-line program on n inputs $x_1, x_2, ..., x_n$. If for $1 \leq i \leq n$ an *arrival time* $t_i \in \mathbb{N}_0$ for x_i is given, we define the *delay* of x_i in C as the sum of t_i and the maximum number of gates on a directed path in C starting in x_i. The *delay* of C is defined as the maximum delay of one of its inputs.

The notion of delay is a natural generalization of the notion of depth. It is of practical interest because it corresponds exactly to the *static timing analysis* used throughout the industry for the analysis of the timing behaviour of a chip. We prove a lower bound on the delay and construct circuits of close-to-optimal delay for several classes of functions. We describe circuits solving the prefix problem on n inputs that are of essentially optimal delay and of size $O(n \log(\log n))$. Finally, we relate delay to formula size.

Keywords. circuit; straight-line program; depth; size; prefix problem; computer arithmetic; VLSI design; static timing analysis

1 Introduction

We consider *circuits* representing *straight-line programs* and refer to [22] or [27] for basic definitions. Every such circuit is a *directed acyclic graph* whose vertices have been identified either with *inputs*, *outputs* or *computation steps*. The vertices identified with computation steps are called *gates*. The functions evaluated by gates belong to a finite set Ω of functions, which is called the *basis*. Two classical measures associated with a circuit C are its *size* $\text{size}(C)$, which is the number of its gates, and its *depth* $\text{depth}(C)$, which is the maximum number of gates on a directed path in C.

One of the main motivations to study circuits is VLSI design where the main optimization issues are area consumption, power consumption and speed. Whereas size is an appropriate measure for area and power consumption, the relation between depth and speed is more problematic, since input signals may arrive at different times. The approach used in the industry to analyze the timing behaviour of a chip is the so-called *static timing analysis* [5, 6, 9], which computes estimates for the arrival times for all relevant signals on a chip. This motivates the following definition.

Definition 1 *Given a circuit C with inputs $x_1, x_2, ..., x_n$ and given an integer arrival time $t_i \in \mathbb{N}_0 = \mathbb{N} \cup \{0\} = \{0, 1, 2, ...\}$ for x_i for $1 \leq i \leq n$, the delay of input x_i for $1 \leq i \leq n$ in C is defined as the sum of t_i and the maximum number of gates on a directed path in C starting at x_i. The delay $\text{delay}(C)$ of C is defined as the maximum delay of an input.*

This definition perfectly corresponds to the worst-case static timing analysis necessary to guarantee correct functioning of a chip for all inputs: If we replace in Definition 1 the maximum number of gates with the maximum accumulated gate delay, then delay(C) equals the arrival time as calculated by static timing analysis. Therefore, this notion of delay is more appropriate for practical application than average case notions [8].

Clearly, if C is a circuit for some boolean function f depending on the inputs x_1, x_2,..., x_n with arrival times t_1, t_2,...,$t_n \in \mathbb{N}_0$, then

$$\max\{\text{depth}(C), \max\{t_1, t_2, ..., t_n\}\} \leq \text{delay}(C) \leq \text{depth}(C) + \max\{t_1, t_2, ..., t_n\},$$

which implies that the delay of a minimum depth circuit for f is at most twice the optimum delay. To some extent this remark justifies the use of circuits of small depth which are mostly known for a long time (cf. e.g. [1, 11, 17, 23, 26]) to realize fundamental functions such as addition or multiplication on a chip. Nevertheless, arrival time differences are typically large compared to individual gate delays, and thus speed and reliability of a chip can considerably be improved by taking arrival times into account. For some few first attempts to do so we refer the reader to [4, 10, 15, 25] and [29].

Whereas circuits of minimum depth often display a very regular structure, circuits of minimum delay may look quite irregular even for simple functions. Therefore, apart from having a purely practical motivation, the notion of delay leads to interesting theoretical problems.

In the present paper we prove some fundamental results on the delay. In Section 2, we prove a lower bound and construct circuits of close-to-optimal delay for some classes of functions. In Section 3, we describe circuits solving the prefix problem on n inputs that are of essentially optimal delay and of size $O(n \log(\log n))$. Finally, in Section 4, we relate formula size and delay.

2 A lower bound and simple cases

First we extend a lower bound on the depth due to Winograd [28].

Proposition 1 *If C is a circuit of fan-in at most r for some boolean function f depending on the inputs $x_1, x_2, ..., x_n$ with arrival times $t_1, t_2, ..., t_n \in \mathbb{N}_0$, then*

$$\text{delay}(C) \geq \left\lceil \log_r \left(\sum_{i=1}^{n} r^{t_i} \right) \right\rceil . \tag{1}$$

Proof: The existence of a circuit C of fan-in at most r and delay T for f implies the existence of a rooted r-ary tree with n leaves of depths $(T - t_1), (T - t_2), ..., (T - t_n) \in \mathbb{N}_0$. By Kraft's inequality, such a tree exists if and only if $\sum_{i=1}^{n} r^{-(T-t_i)} \leq 1$ or, equivalently, $T \geq \log_r \left(\sum_{i=1}^{n} r^{t_i} \right)$, and the proof is complete. □

Since a tree as considered in the proof of Proposition 1 can clearly be constructed in polynomial time, the following result is immediate.

Proposition 2 *Let $\circ : D^2 \to D$ be an associative and commutative operation on two inputs defined on some domain D.*

Then a circuit over the basis $\{\circ\}$ for the function $x_1 \circ x_2 \circ ... \circ x_n$ on inputs $x_1, x_2, ..., x_n$ with arrival times $t_1, t_2, ..., t_n \in \mathbb{N}_0$ whose delay matches the lower bound (1) for $r = 2$ exists and can be constructed in polynomial time.

It is a simple exercise — leading to an alternative proof of Proposition 2 — to show that circuits of minimum delay for functions as in Proposition 2 can also be obtained by a greedy algorithm that iteratively replaces two inputs, say x_i and x_j, of smallest arrival times t_i and t_j with a new input of arrival time $\max\{t_i, t_j\} + 1$. It is obvious that Proposition 2 and the greedy procedure generalize to $r > 2$.

The next theorem shows that the functions considered in Proposition 2 are essentially the only ones for which we can always achieve a delay as in (1) for each arrival time assignment.

Theorem 1 *Let f be a boolean function depending on the inputs $x_1, x_2, ..., x_n$.*

For all assignments of arrival times $t_i \in \mathbb{N}_0$ to x_i for $1 \leq i \leq n$ there exists a circuit for f of fan-in at most 2 with a delay that matches the lower bound (1) for $r = 2$ if and only if either

$$f(x_1, x_2, ..., x_n) = y_1 \wedge y_2 \wedge ... \wedge y_n$$

or

$$f(x_1, x_2, ..., x_n) = y_1 \vee y_2 \vee ... \vee y_n$$

or

$$f(x_1, x_2, ..., x_n) = y_1 \oplus y_2 \oplus ... \oplus y_n,$$

where y_i equals either x_i or $\neg x_i$ for $1 \leq i \leq n$.[1]

Proof: The '*if*-part of the statement follows easily from Proposition 2 and we proceed to the proof of the '*only if*-part.

Let f have the described property. By assigning arrival times $1, 1, 2, 3, ..., (n-2), (n-1)$ to the inputs, it follows that for every permutation $\pi \in S_n$ there is a representation of f of the form

$$f(x_1, x_2, ..., x_n) \;=\; g_1(x_{\pi(1)}, g_2(x_{\pi(2)}, ...g_{n-2}(x_{\pi(n-2)}, g_{n-1}(x_{\pi(n-1)}, x_{\pi(n)}))...)) \qquad (2)$$

such that g_i depends on both of its inputs for $1 \leq i \leq n-1$.

By considering all 10 different boolean functions depending on two inputs and using the relations $\neg(x \wedge y) = (\neg x) \vee (\neg y)$, $\neg(x \vee y) = (\neg x) \wedge (\neg y)$ and $\neg(x \oplus y) = (\neg x) \oplus y = x \oplus (\neg y)$, it is possible to transform (2) to

$$f(x_1, x_2, ..., x_n) \;=\; y^\pi_{\pi(1)} \circ^\pi_1 (y^\pi_{\pi(2)} \circ^\pi_2 (...(y^\pi_{\pi(n-2)} \circ^\pi_{n-2} (y^\pi_{\pi(n-1)} \circ^\pi_{n-1} y^\pi_{\pi(n)}))...)) \qquad (3)$$

[1] As usual $\neg x$ denotes the negation of x; $\wedge$, $\vee$ and $\oplus$ denote `and`, `or` and `exclusive or`, respectively.

where y_i^π equals x_i or $\neg x_i$ for $1 \leq i \leq n$ and $\circ_i^\pi \in \{\wedge, \vee, \oplus\}$ for $1 \leq i \leq n-1$.

For contradiction, we assume that $\circ_i^{\mathrm{id}} \neq \circ_{i+1}^{\mathrm{id}}$ for some $1 \leq i \leq n-2$. First let $n = 3$.

If $(\circ_1^{\mathrm{id}}, \circ_2^{\mathrm{id}}) \in \{(\vee, \oplus), (\wedge, \oplus), (\wedge, \vee), (\vee, \wedge)\}$, then considering the cardinality of the set $f^{-1}(\{1\})$ gives $(\circ_1^\pi, \circ_2^\pi) = (\circ_1^{\mathrm{id}}, \circ_2^{\mathrm{id}})$ in (3) for all permutations π. For each of the four different possibilities this easily implies a contradiction to (3). We leave the details to the reader and give just one example: If $(\circ_1^\pi, \circ_2^\pi) = (\vee, \oplus)$, then x_1 can force $f(x_1, x_2, x_3) = 1$, which is not true for x_2 or x_3. This contradicts (3) for permutations π with $\pi(1) \in \{2, 3\}$.

Hence we may assume that $(\circ_1^{\mathrm{id}}, \circ_2^{\mathrm{id}}) \in \{(\oplus, \vee), (\oplus, \wedge)\}$. In both cases, changing the value of x_1 changes the value of $f(x_1, x_2, x_3)$, which is not true for x_2 or x_3. Again, this easily implies a contradiction to (3).

Now let $n \geq 4$. By substituting appropriate constants to all inputs except x_i, x_{i+1} and x_{i+2} we reduce f in (3) for $\pi = \mathrm{id}$ to $y_i^{\mathrm{id}} \circ_i^{\mathrm{id}} (y_{i+1}^{\mathrm{id}} \circ_{i+1}^{\mathrm{id}} y_{i+2}^{\mathrm{id}})$. Clearly, similar arguments as above imply a contradiction and the proof is complete. □

3 The prefix problem

In this section we consider the so-called *prefix problem*.

Prefix problem
Input: An associative operation $\circ : D^2 \to D$ and inputs $x_1, x_2, ..., x_n \in D$.
Output: $x_1 \circ x_2 \circ ... \circ x_i$ for all $1 \leq i \leq n$.

The prefix problem lies at the core of many fundamental problems. The hard part in designing a fast adder, for example, is the calculation of the *carry bits*, which is equivalent to a prefix problem. It is an easy exercise to construct circuits for the prefix problem of depth $\log_2(n) + o(\log(n))$ and size $O(n \log(n))$ or of depth $2\log_2(n) + o(\log(n))$ and size $O(n)$. With a little more effort Ladner and Fischer [14] construct such circuits with depth $\lceil \log_2(n) \rceil + k$ and size $2n\left(1 + \frac{1}{2^k}\right)$ for each $0 \leq k \leq \lceil \log_2(n) \rceil$. None of these constructions can accommodate arrival times.

Our main result in this section is the recursive construction of circuits $P(t_1, t_2, ..., t_n)$ over the basis $\{\circ\}$ that solve the prefix problem on n inputs $x_1, x_2, ..., x_n$ with arrival times $t_1, t_2, ..., t_n \in \mathbb{N}_0$ which are of close-to-optimal delay and of size $O(n \log(\log(n)))$. Constructions similar to those described by Liu et al. in [15] yield circuits for the prefix problem with close-to-optimal delay but quadratic size.

Construction of $P(t_1, t_2, ..., t_n)$

For $n = 1$ the circuit $P(t_1)$ consists just of the input vertex x_1 having fan-out 1.

For $n \geq 2$ we apply the following steps.

Step 1

Partition the set $\{1, 2, ..., n\}$ into $l := \left\lceil \sqrt{n} \right\rceil$ sets

$V_1 = \{1, 2, ..., n_1\}$,

$$\begin{aligned} V_2 &= \{(n_1+1),(n_1+2),...,(n_1+n_2)\},..., \\ V_l &= \{(n_1+n_2+...+n_{l-1}+1),(n_1+n_2+...+n_{l-1}+2),...,(n_1+n_2+...+n_l)\} \end{aligned}$$

such that $n_1 \geq n_2 \geq ... \geq n_l$ and $n_1 - n_l \leq 1$.

Step 2
For $1 \leq i \leq l$ we use the following dynamic programming approach to construct a circuit C_i over the basis $\{\circ\}$ calculating $y_i := \circ_{j \in V_i} x_j$: In what follows C_{j_1,j_2} will denote a circuit calculating

$$\circ_{j=j_1}^{j_2} x_{(n_1+n_2+...+n_{i-1}+j)}$$

for $1 \leq j_1 \leq j_2 \leq n_i$. If $j_1 = j_2$, then C_{j_1,j_2} consists just of the corresponding input vertex.

If $j_1 < j_2$, we recursively construct $C(j_1, j_2)$ using one $\circ$-gate joining the outputs of two circuits $C(j_1, l)$ and $C(l, j_2)$ such that

$$\max\{\text{delay}(C(j_1,l)), \text{delay}(C(l,j_2))\}$$

is minimized.

Let $C_i = C(1, n_i)$. Clearly, C_i use $(n_i - 1)$ $\circ$-gates. It will follow from Lemma 1 below that the computation of y_i by C_i terminates at time $t(y_i)$ with

$$t(y_i) \leq \log_2 \left(\sum_{j \in V_i} 2^{t_j} \right) + 2.$$

Step 3
For $1 \leq i \leq l$ we recursively construct

$$P(t_{(n_1+n_2+...+n_{i-1}+1)}, ..., t_{(n_1+n_2+...+n_i-1)})$$

and use these circuits to calculate all $(n_i - 1)$ prefixes on the inputs x_j for $j \in V_i \setminus \{n_1 + n_2 + ... + n_i\}$.

Step 4
We construct $P(t(y_1), t(y_2), ..., t(y_{l-1}))$ to calculate all $(l-1)$ prefixes on the inputs y_j for $1 \leq j \leq l-1$ calculated by the circuits constructed in Step 2.

Step 5
For $2 \leq i \leq l$ and $j \in V_i \setminus \{n_1 + n_2 + ... + n_i\}$ we join the output $(y_1 \circ y_2 \circ ... \circ y_{i-1})$ of the circuit constructed in Step 4 with the output $\circ_{\substack{k \in V_i \\ k \leq j}} x_k$ of the circuit constructed in Step 3 using one $\circ$-gate which calculates

$$\circ_{k=1}^{j} x_k = (y_1 \circ y_2 \circ ... \circ y_{i-1}) \circ \left(\circ_{\substack{k \in V_i \\ k \leq j}} x_k \right).$$

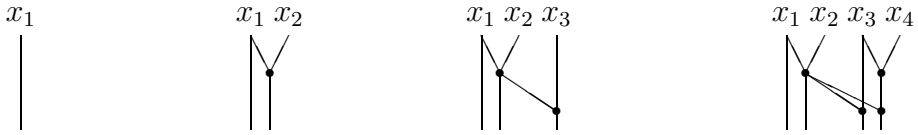

Figure 1: $P(t_1, t_2, ..., t_n)$ for $n \leq 4$

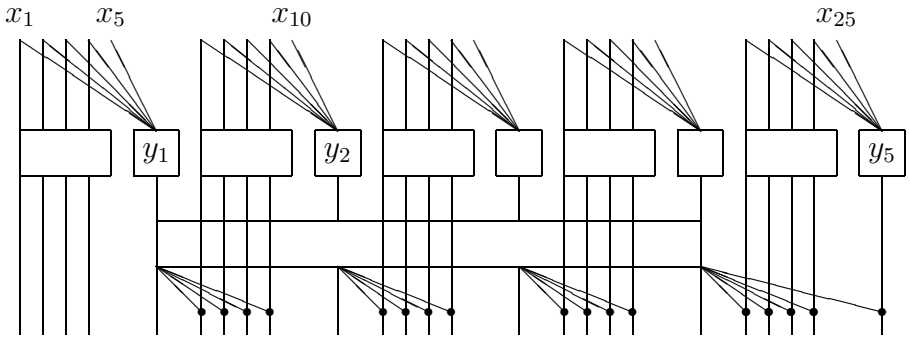

Figure 2: $P(t_1, t_2, ..., t_{25})$.

Step 6
Finally, we join the output $(y_1 \circ y_2 \circ ... \circ y_{l-1})$ of the circuit constructed in Step 4 with the output y_l of the circuit constructed in Step 2 using one $\circ$-gate which calculates $\circ_{i=1}^{n} x_i$.

In Figures 1 and 2 we illustrate $P(t_1, t_2, ..., t_n)$ for $n \leq 4$ and $P(t_1, t_2, ..., t_{25})$. The next lemma proves the claim made in Step 2.

Lemma 1 *For $a, a_1, a_2, ..., a_n \in \mathbb{N}_0$ let $\mathcal{D} : \cup_{i \in \mathbb{N}} \mathbb{N}_0^i \to \mathbb{N}_0$ be defined recursively by $\mathcal{D}(a) = a$ and*

$$\mathcal{D}(a_1, a_2, ..., a_n) \quad = \quad \min_{1 \leq l \leq n-1} \max\{\mathcal{D}(a_1, a_2, ..., a_l), \mathcal{D}(a_{l+1}, a_{l+2}, ..., a_n)\} + 1. \qquad (4)$$

Then

$$\mathcal{D}(a_1, a_2, ..., a_n) \leq \left\lceil \log_2 \left(\sum_{i=1}^{n} 2^{a_i} \right) \right\rceil + 1.$$

Proof: We start with a series of claims.

Let $n_A, n_B \in \mathbb{N}$, $l \in \mathbb{N}_0$, $A, A' \in \mathbb{N}_0^{n_A}$ with $A \leq A'$ (componentwise) and $B \in \mathbb{N}_0^{n_B}$. In order to simplify our notation we denote the vector $(a_1, a_2, ..., a_{n_A}, b_1, b_2, ..., b_{n_B})$ by (A, B) where $A = (a_1, a_2, ..., a_{n_A})$ and $B = (b_1, b_2, ..., b_{n_B})$. Furthermore, for $l \geq 1$ let $Z(l)$ denote the vector of l zeros.

Claim 1 $\mathcal{D}(A) \leq \mathcal{D}(A')$, $\mathcal{D}(A) \leq \mathcal{D}(A, 0)$, $\mathcal{D}(B) \leq \mathcal{D}(0, B)$, $\mathcal{D}(A, B) \leq \mathcal{D}(A, 0, B)$.

Proof of Claim 1: All these monotonicity properties follow immediately from (4) by induction. $\square$

Claim 2 $\mathcal{D}(Z(l)) = \lceil \log_2(l) \rceil$ for $l \geq 1$.

Proof of Claim 2: Again by induction, we obtain $\mathcal{D}(Z(2^i)) = i$ for $i \geq 0$, which immediately implies the desired result. □

Claim 3 $\mathcal{D}(A, l) \leq \mathcal{D}(A, Z(2^l))$ and $\mathcal{D}(l, B) \leq \mathcal{D}(Z(2^l), B)$.

Proof of Claim 3: We only prove the first inequality. The second follows by symmetry.

For contradiction, we assume that (A, l) is a counterexample of minimum length $n_A + 1$.

If $\mathcal{D}(A, Z(2^l)) = \max\{\mathcal{D}(A_1), \mathcal{D}(A_2, Z(2^l))\} + 1$ for some non-trivial A_1 and some A_2 with $(A_1, A_2) = A$, then (4) and the choice of (A, l) imply the contradiction

$$\begin{aligned} \mathcal{D}(A, l) &\leq \max\{\mathcal{D}(A_1), \mathcal{D}(A_2, l)\} + 1 \\ &\leq \max\{\mathcal{D}(A_1), \mathcal{D}(A_2, Z(2^l))\} + 1 \\ &= \mathcal{D}(A, Z(2^l)). \end{aligned}$$

(Note that if $A_1 = A$, then $\mathcal{D}(A_2, l) = l = \lceil \log_2(2^l) \rceil = \mathcal{D}(A_2, Z(2^l))$ by Claim 2.)

Therefore, there is some $1 \leq r \leq 2^l - 1$ such that

$$\mathcal{D}(A, Z(2^l)) = \max\{\mathcal{D}(A, Z(r)), \mathcal{D}(Z(2^l - r))\} + 1.$$

By (4), we have

$$\mathcal{D}(A, l) \leq \max\{\mathcal{D}(A), \mathcal{D}(l)\} + 1 = \max\{\mathcal{D}(A), l\} + 1.$$

If $\mathcal{D}(A) \geq l$, then

$$\mathcal{D}(A, l) \leq \mathcal{D}(A) + 1 \leq \mathcal{D}(A, Z(r)) + 1 \leq \mathcal{D}(A, Z(2^l)),$$

which is a contradiction. Hence $\mathcal{D}(A) < l$ and we obtain the contradiction

$$\mathcal{D}(A, l) \leq l + 1 = \lceil \log_2(2^l + 1) \rceil = \mathcal{D}(0, Z(2^l)) \leq \mathcal{D}(A, Z(2^l))$$

and the proof of the claim is complete. □

Claim 4 $\mathcal{D}(A, l, B) \leq \mathcal{D}(A, Z(2^{l+1}), B)$.

Proof of Claim 4: This can be proved similarly to Claim 3 and we leave the proof to the reader. □

Altogether we obtain

$$\begin{aligned} \mathcal{D}(a_1, a_2, ..., a_n) &\leq \mathcal{D}\left(Z\left(\sum_{i=1}^{n} 2^{(a_i+1)}\right)\right) \leq \left\lceil \log_2\left(\sum_{i=1}^{n} 2^{(a_i+1)}\right)\right\rceil \\ &= \left\lceil \log_2\left(\sum_{i=1}^{n} 2^{a_i}\right)\right\rceil + 1 \end{aligned}$$

and the proof is complete. □

From the above construction it is obvious that some gates have fan-out up to $O(\sqrt{n})$. Similarly, the constructions of Ladner and Fischer [14] lead to large fan-outs. For many practical applications though, a fan-out of l at a gate should actually contribute $\Theta(\log(l))$ to the delay of that gate.

In the present situation we model this by using the basis $\{\circ, \mathrm{id}\}$, where $\mathrm{id} : D \to D$ is the identity function, and the following fan-out conditions.

(i) Input vertices and $\circ$-gates have fan-out at most 1.

(ii) id-gates have fan-out at most 2.

Next, we construct circuits $P'(t_1, t_2, ..., t_n)$ over the basis $\{\circ, \mathrm{id}\}$ that satisfy Conditions (i) and (ii) and solve the prefix problem on $n \geq 2$ inputs $x_1, x_2, ..., x_n$ with arrival time $t_i \in \mathbb{N}_0$ for x_i for $1 \leq i \leq n$.

Construction of $P'(t_1, t_2, ..., t_n)$

Starting from $P(t_1, t_2, ..., t_n)$ we apply the following steps.

Step 1
Add one id-gate at input vertices of fan-out 2. (*Note that all other input vertices already have fan-out* 1.)

Step 2
For $2 \leq i \leq l-1$ add $(n_i - 1)$ id-gates at the $\circ$-gate calculating $y_1 \circ y_2 \circ ... \circ y_{i-1}$ in such a way that they contribute a delay of $\lceil \log_2(n_i) \rceil$. (*Note that this is clearly possible using balanced binary trees.*)

Step 3
Add n_l id-gates at the $\circ$-gate calculating $y_1 \circ y_2 \circ ... \circ y_{l-1}$ in such a way that they contribute a delay of $\lceil \log_2(n_l + 1) \rceil$.

Step 4
Recurvisely apply the above changes to the subfunctions of the form $P(t'_1, t'_2, ..., t'_{l'})$ used in $P(t_1, t_2, ..., t_n)$.

For $w \geq n \geq 1$ let $\mathrm{size}(n)$ and $\mathrm{delay}(w, n)$ denote the maximum size and the maximum delay of a circuit $P(t_1, t_2, ..., t_n)$ such that $t_i \in \mathbb{N}_0$ for $1 \leq i \leq n$ and $w = \sum_{i=1}^{n} 2^{t_i}$. Define $\mathrm{size}'(n)$ and $\mathrm{delay}'(w, n)$ for $P'(t_1, t_2, ..., t_n)$ similarly. We have the following recursions.

Lemma 2 *For* $w \geq n \geq 3$

$$\begin{aligned}
\mathrm{size}(n) &\leq \left(\lceil \sqrt{n} \rceil + 1\right) \mathrm{size}\left(\lceil \sqrt{n} \rceil - 1\right) + 2\left(n - \lceil \sqrt{n} \rceil\right), \\
\mathrm{size}'(n) &\leq \left(\lceil \sqrt{n} \rceil + 1\right) \mathrm{size}'\left(\lceil \sqrt{n} \rceil - 1\right) + 4\left(n - \lceil \sqrt{n} \rceil\right), \\
\mathrm{delay}(w, n) &\leq \mathrm{delay}\left(4w, \lceil \sqrt{n} \rceil - 1\right) + 1 \text{ and} \\
\mathrm{delay}'(w, n) &\leq \mathrm{delay}'\left(4w, \lceil \sqrt{n} \rceil - 1\right) + \log_2\left(\sqrt{n}\right) + 5.
\end{aligned}$$

Proof: Let $t_1, t_2, ..., t_n \in \mathbb{N}_0$ be such that $w = \sum_{i=1}^{n} 2^{t_i}$. We use the same notation as during the construction of $P(t_1, t_2, ..., t_n)$ and $P'(t_1, t_2, ..., t_n)$. Since $n \geq 3$, we have $2 \leq n_1 \leq l$.

The circuit $P(t_1, t_2, ..., t_n)$ contains $(l+1)$ subcircuits of the form $P(t'_1, t'_2, ..., t'_{l'})$ on at most $(l-1)$ inputs each. To evaluate y_i for $1 \leq i \leq l$, a number of $(n_1 - 1) + (n_2 - 1) + ... + (n_l - 1) = (n - l)$ $\circ$-gates are used. Finally, to compute the remaining outputs $(n_2 - 1) + (n_3 - 1) + ... + (n_l - 1) + 1 \leq (n - l)$ more $\circ$-gates are used. This implies the recursion for size(n).

Since the construction of $P'(t_1, t_2, ..., t_n)$ from $P(t_1, t_2, ..., t_n)$ recursively adds

$$(n - l) + (n_2 - 1) + (n_3 - 1) + ... + (n_{l-1} - 1) + n_l \leq 2\left(n - \left\lceil \sqrt{n} \right\rceil\right)$$

id-gates, the recursion for size$'(n)$ follows.

Now we proceed to delay(w, n) and delay$'(w, n)$. We have

$$\sum_{i=1}^{l} 2^{t(y_i)} \leq \sum_{i=1}^{l} 2^{\left(\log_2\left(\sum_{j \in V_i} 2^{t_j}\right)+2\right)} = 4 \sum_{i=1}^{l} \sum_{j \in V_i} 2^{t_j} = 4 \sum_{i=1}^{n} 2^{t_i} = 4w.$$

As delay(w, n) is obviously non-decreasing in w, the recursion for delay(w, n) follows.

Since the construction of $P'(t_1, t_2, ..., t_n)$ from $P(t_1, t_2, ..., t_n)$ recursively increases the delay by

$$\begin{aligned} 1 + \max\left\{\left\lceil \log_2(n_2) \right\rceil, ..., \left\lceil \log_2(n_{l-1}) \right\rceil, \left\lceil \log_2(n_l + 1) \right\rceil\right\} &\leq 1 + 1 + \log_2\left(\sqrt{n} + 2\right) \\ &\leq 4 + \log_2\left(\sqrt{n}\right), \end{aligned}$$

the recursion for delay$'(n)$ follows. $\square$

In the next lemma we solve the above recursions.

Lemma 3
(i) Let $s : \mathbb{N} \to \mathbb{N}$, $\alpha, \beta \geq 0$ and $n_0 \in \mathbb{N}$ be such that for $n \geq n_0$

$$s(n) \leq \left(\sqrt{n} + \alpha\right) s\left(\left\lceil \sqrt{n} \right\rceil - 1\right) + \beta n.$$

Then there is some $\gamma \geq 0$ such that for all $n \in \mathbb{N}$

$$s(n) \leq \gamma n \log_2(\log_2(n)) + s(1). \tag{5}$$

(ii) Let $d : \mathbb{N}^2 \to \mathbb{N}$, $\alpha, \beta \geq 0$ and $n_0 \in \mathbb{N}$ be such that for $w \in \mathbb{N}$ and $n \leq n_0 - 1$ the term $d(w, n) - \log_2(w)$ is bounded and for $w \in \mathbb{N}$ and $n \geq n_0$

$$d(w, n) \leq d\left(\alpha w, \left\lceil \sqrt{n} \right\rceil - 1\right) + \beta.$$

Then there is some $\gamma \geq 0$ such that for all $w, n \in \mathbb{N}$

$$d(w, n) \leq \log_2(w) + (\log_2(\alpha) + \beta) \log_2(\log_2(n)) + \gamma. \tag{6}$$

(iii) Let $d : \mathbb{N}^2 \to \mathbb{N}$, $\alpha, \beta \geq 0$ *and* $n_0 \in \mathbb{N}$ *be such that for* $w \in \mathbb{N}$ *and* $n \leq n_0 - 1$ *the term* $d(w,n) - \log_2(w)$ *is bounded and for* $w \in \mathbb{N}$ *and* $n \geq n_0$

$$d(w,n) \leq d\left(\alpha w, \left\lceil \sqrt{n} \right\rceil - 1\right) + \log_2\left(\sqrt{n}\right) + \beta.$$

Then there is some $\gamma \geq 0$ *such that for all* $w, n \in \mathbb{N}$

$$d(w,n) \leq \log_2(w) + \log_2(n) + (\log_2(\alpha) + \beta)\log_2(\log_2(n)) + \gamma. \tag{7}$$

Proof: We just prove (i) and leave the analogous proofs of (ii) and (iii) to the reader.

We will prove (5) by induction. Let $\gamma' > \beta$. Clearly, there is some $n_1 \geq n_0$ such that for $n \geq n_1$ and $\gamma \geq \gamma'$

$$\alpha\gamma\sqrt{n}\log_2(\log_2(n)) + s(1)\left(\sqrt{n} + \alpha\right) + \beta n - \gamma\sqrt{n}\left(\sqrt{n} + \alpha\right) \leq s(1).$$

Let $\gamma \geq \gamma'$ be such that (5) holds for $n \leq n_1 - 1$. For $n \geq n_1$ we obtain, by induction, that

$$\begin{aligned} s(n) &\leq \left(\sqrt{n} + \alpha\right) s\left(\left\lceil \sqrt{n} \right\rceil - 1\right) + \beta n \\ &\leq \left(\sqrt{n} + \alpha\right)\left(\gamma\sqrt{n}\log_2\left(\log_2\left(\sqrt{n}\right)\right) + s(1)\right) + \beta n \\ &= \gamma n \log_2(\log_2(n)) + \alpha\gamma\sqrt{n}\log_2(\log_2(n)) + s(1)\left(\sqrt{n} + \alpha\right) + \beta n - \gamma\sqrt{n}\left(\sqrt{n} + \alpha\right) \\ &\leq \gamma n \log_2(\log_2(n)) + s(1) \end{aligned}$$

and the proof of (5) is complete. □

Combining Lemmata 2 and 3 with the obvious fact that $\log_2\left(\sum_{i=1}^{n} 2^{t_i}\right) \geq \max\{t_i \mid 1 \leq i \leq n\}$ we obtain the main result of this section.

Theorem 2 *The prefix problem on inputs* $x_1, x_2, ..., x_n$ *with arrival times* $t_1, t_2, ..., t_n \in \mathbb{N}_0$ *can be solved by*
(i) a circuit over the basis $\{\circ\}$ *with size* $O\left(n \log\left(\log\left(n\right)\right)\right)$ *and delay*

$$\log_2\left(\sum_{i=1}^{n} 2^{t_i}\right) + 3\log_2\left(\log_2\left(n\right)\right) + O(1);$$

(ii) a circuit over the basis $\{\circ, \mathrm{id}\}$ *satisfying the fan-out conditions (i) and (ii) with size* $O\left(n \log\left(\log\left(n\right)\right)\right)$ *and delay*

$$\log_2\left(\sum_{i=1}^{n} 2^{t_i}\right) + \log_2\left(n\right) + 7\log_2\left(\log_2\left(n\right)\right) + O(1).$$

Furthermore, both kinds of circuits can be constructed in polynomial time.

Clearly, Theorem 2 is most interesting for considerable arrival time differences. In this case $\log_2(n)$ may be arbitrarily small compared to $\log_2(\sum_{i=1}^{n} 2^{t_i})$. Hence, applying well-known methods for fan-out reduction (e.g. [7]) to $P(t_1, t_2, ..., t_n)$ leads to weaker results than (ii) in Theorem 2.

As we mentioned at the beginning of this section, circuits for the prefix problem can be used to construct adders. Given arrival times, say $t_1, t_2, ..., t_n \in \mathbb{N}_0$ and $t'_1, t'_2, ..., t'_n \in \mathbb{N}_0$, for the bits of two n-bit binary numbers, say x and y, and using a well-known construction (cf. e.g. [14, 27]) we obtain a circuit over the basis $\{\vee, \wedge, \neg\}$ of fan-in 2 for $\vee$- or $\wedge$-gates and fan-in 1 for $\neg$-gates calculating the sum of x and y with size $O(n \log(\log(n)))$ and delay

$$2\log_2\left(\sum_{i=1}^{n}\left(2^{t_i} + 2^{t'_i}\right)\right) + 6\log_2(\log_2(n)) + O(1).$$

In view of Proposition 1, the bounds on the delay given in Theorem 2 are close-to-optimal and the bounds on the size are optimal up to a factor of $O(\log(\log(n)))$. The best known adders are of depth $\log_2(n) + O\left(\sqrt{\log(n)}\right)$ and size $O(n\log(n))$ [1] or size $O(n)$ [11], respectively. The adder developed in [29], which takes arrival times into account, has size $O(n\log(n))$, but no delay bound has been proved.

4 Formula size and delay

In this section we extend a well-known type of result relating formula size and depth. The first such result was proved by Spira [24], whose original idea underwent numerous variations [2, 3, 12, 13, 16, 18, 19]. Most of these can be generalized from depth to delay similarly to the next theorem.

The following proof relies on restructuring a given formula using of the so-called *select function* $\mathrm{sel}(x, y, z) = (x \wedge y) \vee ((\neg x) \wedge z)$. Since most standard cell libraries in VLSI design contain a primitive gate for this function, the proof can easily be turned into a practical strategy to speed up a late signal on a chip by applying the restructuring step to some part of its fan-in cone.

Theorem 3 *Let $r \geq 2$ and let Ω be a set of boolean functions on at most r inputs. Let $\alpha \in \mathbb{N}$ be the minimum depth of a circuit over Ω for the function $\mathrm{sel}(x, y, z) = (x \wedge y) \vee ((\neg x) \wedge z)$. For every function in Ω on $l \geq 2$ inputs, let Ω contain all functions on $(l-1)$ inputs that arise by setting one of the inputs to 0 or 1.*

Then there is some constant $\beta = \beta(\Omega)$ with the following property: Let C be a read-once formula, i.e. a circuit for a boolean function $f : \{0,1\}^n \to \{0,1\}$ over Ω such that the fan-out of input vertices and gates is at most 1. For $1 \leq i \leq n$ let $t_i \in \mathbb{N}_0$ be the arrival time of the i-th input of f.

Then there is a circuit $\tilde{C}$ for f over Ω such that

$$\mathrm{delay}(\tilde{C}) \quad \leq \quad \frac{\alpha}{\log_r(r+1) - 1} \log_r\left(\sum_{i=1}^{n} r^{t_i}\right) + \beta. \tag{8}$$

Proof: We prove the result by induction over n. Let $w = \sum_{i=1}^{n} r^{t_i}$ and note that $\frac{\alpha}{\log_r(r+1)-1} = \frac{\alpha}{\log_r\left(\frac{r+1}{r}\right)} > \alpha \geq 1$. If $n = 1$, then

$$\frac{\alpha}{\log_r(r+1)-1} \log_r \left(\sum_{i=1}^{n} r^{t_i} \right) + \beta = \frac{\alpha}{\log_r(r+1)-1} t_1 + \beta \geq t_1 + \beta.$$

Therefore, there is some $\beta \in \mathbb{N}$ independent of C and f such that (8) holds for $n = 1$. We may assume that $\beta \geq \alpha$.

Now let $n \geq 2$. The directed graph underlying C is a rooted tree T whose leaves are the inputs $x_1, x_2, ..., x_n$. For every vertex u of T let $w(u)$ denote the sum of r^{t_i} where the sum extends over all i such that x_i lies in the subtree of T rooted at u.

Let the vertex u be chosen such that (i) $w(u) > \frac{w}{r+1}$, (ii) $w(u)$ is minimum subject to (i) and u has maximum distance from the root subject to (i) and (ii). It is easy to see that $w(u) < w$. Let C_u denote the subcircuit of C corresponding to the subtree of T rooted at u. For $i \in \{0, 1\}$ let C_i denote the circuit that arises from C by replacing the output of u by the constant i.

Since C_u, C_1 and C_0 are circuits for functions defined on at most $(n-1)$ inputs, we can apply the induction hypothesis to them. This implies the existence of circuits $\tilde{C}_u$, $\tilde{C}_1$ and $\tilde{C}_0$ over Ω for the same functions whose delay is bounded as in (8).

Note that if g denotes the function computed at the vertex u, then clearly $f = \mathrm{sel}(g, f\ |_{g=1}, f\ |_{g=0})$. Therefore, using $\tilde{C}_u$, $\tilde{C}_1$, $\tilde{C}_0$ and the circuit for sel over Ω, we can construct a circuit $\tilde{C}$ for f over Ω such that

$$\mathrm{delay}(\tilde{C}) \leq \alpha + \frac{\alpha}{\log_r(r+1)-1} \max\left\{\log_r(w(u)), \log_r\left(w - w(u)\right)\right\} + \beta. \tag{9}$$

By the choice of u, we have $w - w(u) \leq \frac{wr}{r+1}$. If $w(u) \leq \frac{wr}{r+1}$, then (9) implies

$$\mathrm{delay}(\tilde{C}) \leq \alpha + \frac{\alpha}{\log_r(r+1)-1} \log_r \left(\frac{wr}{r+1} \right) + \beta = \frac{\alpha}{\log_r(r+1)-1} \log_r(w) + \beta.$$

Hence, we may assume that $w(u) > \frac{wr}{r+1}$. In this case u must be a leaf of T and C_u has delay $\log_r(w(u))$. Therefore, we can strengthen (9) as follows.

$$\mathrm{delay}(\tilde{C}) \leq \alpha + \max\left\{ \log_r(w(u)), \frac{\alpha}{\log_r(r+1)-1} \log_r\left(w - w(u)\right) + \beta \right\}. \tag{10}$$

If the right-hand term yields the maximum in (10), then we can proceed as before. Hence, we may assume that the left-hand term yields the maximum in (10) and trivially we obtain

$$\mathrm{delay}(\tilde{C}) \ \leq \ \alpha + \log_r(w(u)) \leq \frac{\alpha}{\log_r(r+1)-1} \log_r(w) + \beta,$$

which completes the proof. □

5 Conclusions

Motivated by the use of circuits as a mathematical model in VLSI design we proposed the notion of delay. It naturally extends the notion of depth using information provided for example by static timing analysis. Several engineering publications and industrial trends show that chip designers are becoming aware of the need for such a notion [4, 10, 15, 25, 29].

We proved several fundamental results about delay and described algorithms leading to circuits of small delay. The general strategies used in these algorithms can clearly be applied to a variety of problems that are both of theoretical and practical interest.

The definition of delay grew naturally out of a close ongoing cooperation between our own institute and the IBM company that has been lasting for more than 15 years. The theoretical results presented here are presently being implemented as part of our BONN tools, which are electronic design automation tools developed at our institute for industrial use.

References

[1] R. Brent, On the addition of binary numbers, *IEEE Trans. Comput.* **19** (1970), 758-759.

[2] R. Brent, D. Kuck and K. Maruyama, The parallel evaluation of arithmetic expression without division, *IEEE Trans. Comput.* **22** (1973), 532-534.

[3] R. Brent, The parallel evaluation of general arithmetic expressions, *J. Assoc. Comput. Mach.* **21** (1974), 201-206.

[4] M.D. Ercegovac and T. Lang, Fast Multiplication Without Carry-Propagate Addition, *IEEE Trans. Comput.* **39** (1990), 1385-1390.

[5] R.B. Hitchcock, Timing Verification and the Timing Analysis Program, in *Proc. 19th IEEE Design Automation Conference*, 1982, 594-604.

[6] R.B. Hitchcock, G.L. Smith and D.D. Cheng, Timing Analysis of Computer Hardware, *IBM J. Res. Develop.* **26** (1982), 100-105.

[7] H.J. Hoover, M.M. Klawe and N.J. Pippenger, Bounding fan-out in logical networks, *J. Assoc. Comput. Mach.* **31** (1984), 13-18.

[8] A. Jakoby, R. Reischuk and C. Schindelhauer, Circuit Complexity: From the Worst Case to the Average Case, *Proc. 26. Symposium on the Theory of Computer Science (STOC'94)* (1994), 58-67.

[9] N.P. Jouppi, Timing analysis for nMOS VLSI, in *Proc. 20th IEEE Design Automation Conference*, 1983, 411-418.

[10] R.K. Kolagotla, H.R. Srinivas and G.F. Burns, VLSI implementation of a 200-Mhz 16×16 Left-to-Right Carry-free Multiplier in 0.35μm CMOS Technology for Next-Generation DSPs, in *Proc. IEEE 1997 Custom Integrated Circuits Conf.*, 469-472.

[11] V.M. Khrapchenko, Asymptotic estimation of addition time of parallel adder, *Syst. Th. Res.* **19** (1970), 105-122.

[12] V.M. Khrapchenko, On the relation between the complexity and depth of formulas (Russian), *Metody Diskretn. Anal.* **32** (1978), 76-94 (1978).

[13] V.M. Khrapchenko, On a relation between complexity and depth of formulas in a basis containing the median (Russian), *Metody Diskretn. Anal.* **37** (1981), 77-84.

[14] R.E. Ladner and M.J. Fischer, Parallel prefix computation, *J. Assoc. Comput. Mach.* **27** (1980), 831-838.

[15] J. Liu, S. Zhou, H. Zhou and C.-K. Cheng, An algorithmic approach for generic parallel adders, in *Proc. International Conference on Computer Aided Design (ICCAD) '03* (2003), 734-740.

[16] D.E. Muller and F.P. Preparata, Restructuring of arithmetic expressions for parallel evaluation, *J. Assoc. Comput. Mach.* **23** (1976), 534-543.

[17] Y. Ofman, On the algorithmic complexity of discrete functions (Russian, English), *Sov. Phys. Dokl.* **7** (1963), 589-591.

[18] F.P. Preparata and D.E. Muller, Efficient parallel evaluation of Boolean expressions, *IEEE Trans. Comput.* **25** (1976), 548-549.

[19] F.P. Preparata, D.E. Muller and A.B. Barak, Reduction of depth of Boolean networks with a fan-in constraint, *IEEE Trans. Comput.* **26** (1977), 474-479.

[20] D. Rautenbach, C. Szegedy and J. Werber, Fast Circuits for Functions whose Inputs have Specified Arrival Times, manuscript (2003).

[21] D. Rautenbach, C. Szegedy and J. Werber, Delay Optimization of Linear Depth Boolean Circuits with Prescribed Input Arrival Times, manuscript (2003).

[22] J.E. Savage, Models of computation: exploring the power of computing, Reading, MA: Addison Wesley Longman, 1998.

[23] A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen, *Computing* **7** (1971), 281-292.

[24] P.M. Spira, On time-hardware complexity tradeoffs for Boolean functions, in *Proc. 4th Hawaii Symp. Syst. Sc.*, 1971, 525-527.

[25] P.F. Stelling and V.G. Oklobdzija, Optimal designs for multipliers and multiply-accumulators, in *Proceedings of the 15th IMACS World Congress Scientific Computation, Modeling, and Applied Mathematics*, August 1997, 739-744.

[26] C.S. Wallace, A suggestion for a fast multiplier, *IEEE Trans. Electron. Comput.* **13** (1964), 14-17.

[27] I. Wegener, The complexity of Boolean functions, Wiley-Teubner Series in Computer Science. Stuttgart: B. G. Teubner; Chichester etc.: John Wiley & Sons., 1987.

[28] S. Winograd, On the time required to perform addition, *J. Assoc. Comput. Mach.* **12** (1965), 277-285.

[29] W.-C. Yeh, C.-W. Jen, Generalized earliest-first fast addition algorithm, *IEEE Trans. Computers* **52** (2003), 1233-1242.