Delay Optimization of Linear Depth Boolean Circuits with Prescribed Input 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. We consider boolean circuits C over the basis $\Omega = \{\vee, \wedge\}$ with inputs x_1, x_2,..., x_n for which arrival times $t_1, t_2, ..., t_n \in \mathbf{N}_0$ are given. For $1 \leq i \leq n$ we define the delay of x_i in C as the sum of t_i and the number of gates on a longest directed path in C starting at x_i. The delay of C is defined as the maximum delay of an input.

Given a function of the form

$$f(x_1, x_2, ..., x_n) = g_{n-1}(g_{n-2}(...g_3(g_2(g_1(x_1, x_2), x_3), x_4)..., x_{n-1}), x_n)$$

where $g_j \in \Omega$ for $1 \leq j \leq n-1$ and arrival times for $x_1, x_2, ..., x_n$, we describe a cubic-time algorithm that determines a circuit for f over Ω that is of linear size and whose delay is at most 1.44 times the optimum delay plus some small constant.

Keywords. circuit; straight-line program; depth; delay; computer arithmetic; VLSI design

1 Motivation

The motivation for the present work is a problem in VLSI design. At one of the final stages in the design process of a chip, the tool that performs the so-called static timing analysis [2, 3, 4] detects paths of 'negative slack'. These are paths on which the propagation of the signal is too slow to guarantee the correct functioning of the chip. The analysis tool reports these paths, which usually consist of a sequence of gates $g_1, g_2, ...g_m$ that perform some elementary logical operation on their inputs (see Figure 1).

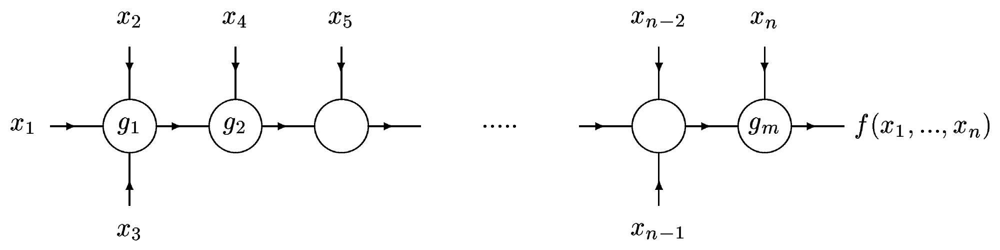

Figure 1

The output of the final gate g_m is a boolean function $f(x_1, ..., x_n)$ of the inputs. If we are given an arrival time, say $\mathrm{t}(x_i)$, for each input x_i and a delay, say $\mathrm{d}(g_j)$, for each gate g_j, then static timing analysis will determine the arrival time of the output of gate g_m, i.e. the time at which the evaluation of f terminates, as the maximum, over all paths from an input x_i to the output of g_m, of the sum of $\mathrm{t}(x_i)$ and all gate delays along the path. If for example for the path in Figure 1, $m = 3$, g_1 is a 3-and, g_2 is a 2-or and g_3 is a 2-nand (for undefined terminology we refer to [8] or [11]), then $f(x_1, x_2, x_3, x_4, x_5) = \neg(((x_1 \wedge x_2 \wedge x_3) \vee x_4) \wedge x_5)$ and the evaluation of f terminates at

$$\begin{aligned}\max \quad & \{\mathrm{t}(x_1) + \mathrm{d}(g_1) + \mathrm{d}(g_2) + \mathrm{d}(g_3), \mathrm{t}(x_2) + \mathrm{d}(g_1) + \mathrm{d}(g_2) + \mathrm{d}(g_3), \\ & \mathrm{t}(x_3) + \mathrm{d}(g_1) + \mathrm{d}(g_2) + \mathrm{d}(g_3), \mathrm{t}(x_4) + \mathrm{d}(g_2) + \mathrm{d}(g_3), \mathrm{t}(x_5) + \mathrm{d}(g_3)\}.\end{aligned}$$

In order to guarantee that the chip works correctly, we have to find a faster representation of f. This leads us to the algorithmical problem which we state more precisely in the next section.

2 Problem

We consider boolean circuits [8, 11] over the basis $\Omega = \{\vee, \wedge\}$ whose elements have fan-in 2 for functions $f : \{0,1\}^n \to \{0,1\}$ of the form

$$f(x_1, x_2, ..., x_n) \quad = \quad g_{n-1}(g_{n-2}(...g_3(g_2(g_1(x_1, x_2), x_3), x_4)..., x_{n-1}), x_n) \tag{1}$$

where $g_j \in \Omega$ for $1 \leq j \leq n-1$. Clearly, (1) immediately leads to a circuit as in Figure 1.

If we are given a non-negative integer arrival time $t_i \in \mathbf{N}_0 = \{0, 1, 2, ...\}$ for input x_i for $1 \leq i \leq n$, then we define the *delay* $\mathrm{delay}(x_i)$ of x_i in some circuit C as the sum of t_i and the number of gates on a longest directed path in C starting at x_i. The *delay* $\mathrm{delay}(C)$ of C is defined as the maximum delay of an input in C. Given a function f and arrival times as above, we denote the minimum delay of a circuit for f by $\mathrm{delay}(f)$. For some first and fundamental results on this notion of delay we refer the reader to [7].

There is a simple lower bound on the achievable delay extending a classical observation of Winograd [12].

Lemma 1 *If $f : \{0,1\}^n \to \{0,1\}$ is computable over Ω and dependent on each of its inputs $x_1, x_2, ..., x_n$, which have arrival times $t_1, t_2, ..., t_n \in \mathbf{N}_0$, then*

$$\mathrm{delay}(f) \quad \geq \quad \left\lceil \log_2 \left(\sum_{i=1}^{n} 2^{t_i} \right) \right\rceil . \tag{2}$$

Proof: For $0 \leq i \leq \mathrm{delay}(f)$ let $n_i = |\{1 \leq j \leq n \mid t_j = \mathrm{delay}(f) - i\}|$. Clearly, $\mathrm{delay}(f) \geq \max\{t_i \mid 1 \leq i \leq n\}$ and thus $n_0 + n_1 + ... + n_{\mathrm{delay}(f)} = n$.

Let C be a circuit for f over Ω of delay $\mathrm{delay}(f)$. Let T_0 be a breadth-first search tree of C rooted at the output gate of C. By repeatedly attaching new leaves to the leaves of T_0, we obtain a rooted binary tree that has at least n_i leaves at distance i from the root for each $0 \leq i \leq \text{delay}(f)$. The existence of such a tree is equivalent to the following sequence of inequalities.

$$\begin{aligned}
0 &\leq 1 - n_0 \\
0 &\leq 2\,(1 - n_0) - n_1 \\
0 &\leq 2\,(2\,(1 - n_0) - n_1) - n_2 \\
\ldots & \\
0 &\leq 2^{\text{delay}(f)} - \sum_{j=0}^{\text{delay}(f)} 2^{(\text{delay}(f)-j)} n_j = 2^{\text{delay}(f)} - \sum_{i=1}^{n} 2^{t_i}.
\end{aligned}$$

Obviously, each inequality is implied by the next inequality and from the final inequality we obtain (2). □

Note that if $f(x_1, x_2, ..., x_n) = \bigvee\limits_{i=1}^{n} x_i$ or $f(x_1, x_2, ..., x_n) = \bigwedge\limits_{i=1}^{n} x_i$, then a tree as considered in the above proof immediately leads to a circuit for f of minimum delay and can obviously be constructed in polynomial time (see [7]).

Our main result is a cubic-time dynamic programming algorithm that produces a circuit for functions f as in (1) whose delay is at most about 1.44 times the value of the lower bound (2). We describe this algorithm first for the function $f_0 : \{0, 1\}^{2n} \to \{0, 1\}$ with

$$f_0(x_1, y_1, x_2, y_2, ..., x_n, y_n) \quad = \quad ((...(((x_1 \wedge y_1) \vee x_2) \wedge y_2) \vee ...) \vee x_n) \wedge y_n. \tag{3}$$

The function f_0 is known in computer arithmetic [9, 10]. It can be used to perform the carry-bit calculation for the addition of two n-bit binary numbers. As part of their circuits for addition Brent [1] and Khrapchenko [5] both described circuits for f_0 of depth $\log_2(n) + O\left(\sqrt{\log_2(n)}\right)$ (cf. also [6]). Nevertheless, their original constructions and analysis hardly generalize to the case of arrival times and would certainly not lead to polynomial time algorithms. In this context we mention the very recent work [13] where a heuristic approach for the construction of an adder is presented taking arrival times into account.

In Section 3 we first describe the algorithm for functions as in (3). In Section 4, we analyse the delay of the circuits constructed in Section 3. In Section 5, we describe the algorithm for functions as in (1) and state the main result. Finally, in Section 6 we make some concluding remarks.

3 Algorithm for f_0 as in (3)

For $1 \leq l \leq n - 1$ the function f_0 satisfies the following identity.

$$\begin{aligned}
& f_0(x_1, y_1, x_2, y_2, ..., x_n, y_n) \\
= \; & ((...(((x_1 \wedge y_1) \vee x_2) \wedge y_2) \vee ...) \vee x_n) \wedge y_n
\end{aligned}$$

$$\begin{aligned}
&= ((\dots((((x_1 \wedge y_1 \wedge y_2) \vee (x_2 \wedge y_2)) \vee x_3) \wedge y_3)\dots) \vee x_n) \wedge y_n \\
&= \dots \\
&= \bigvee_{i=1}^{n} \left(x_i \wedge \bigwedge_{j=i}^{n} y_j \right) \\
&= \left(\left(\bigvee_{i=1}^{l} \left(x_i \wedge \bigwedge_{j=i}^{l} y_j \right) \right) \wedge \left(\bigwedge_{j=l+1}^{n} y_j \right) \right) \vee \left(\bigvee_{i=l+1}^{n} \left(x_i \wedge \bigwedge_{j=i}^{n} y_j \right) \right) \\
&= \left(f_0(x_1, y_1, \dots, x_l, y_l) \wedge \left(\bigwedge_{j=l+1}^{n} y_j \right) \right) \vee f_0(x_{l+1}, y_{l+1}, \dots, x_n, y_n). \qquad (4)
\end{aligned}$$

Note that we commit a small *abus de langage* using 'f_0' to denote formally different functions. We now describe the algorithm for f_0.

Algorithm I
Input: Integers $n \in \mathbf{N} = \{1, 2, \dots\}$ and $t_1, s_1, t_2, s_2, \dots, t_n, s_n \in \mathbf{N}_0$.
Output: A circuit $C_0(t_1, s_1, t_2, s_2, \dots, t_n, s_n)$ over Ω with inputs $x_1, y_1, \dots, x_n, y_n$ that has the two outputs $f_0(x_1, y_1, x_2, y_2, \dots, x_n, y_n)$ and $\bigwedge_{j=1}^{n} y_j$.

In what follows, we use t_i as the arrival time for x_i and s_i as the arrival time for y_i for $1 \leq i \leq n$. Furthermore, we denote the subcircuit of $C_0(t_1, \dots, s_n)$ that computes $f_0(x_1, \dots, y_n)$ by $C_{0,f_0}(t_1, \dots, s_n)$ and the subcircuit of $C_0(t_1, \dots, s_n)$ that computes $\bigwedge_{j=1}^{n} y_j$ by $C_{0,\wedge}(t_1, \dots, s_n)$.

Step 1
If $n = 1$, then let the circuit $C_0(t_1, s_1)$ be as in Figure 2.

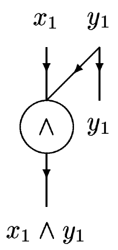

Figure 2 $C_0(t_1, s_1)$

Step 2
If $n \geq 2$, recursively construct $C_0(t_1, \dots, s_n)$ using $C_0(t_1, \dots, s_l)$ and $C_0(t_{l+1}, \dots, s_n)$ for some $1 \leq l \leq n-1$ such that

$$\max\{\text{delay}(C_{0,f_0}(t_1, \dots, s_l)) + 1, \text{delay}(C_{0,f_0}(t_{l+1}, \dots, s_n))\}$$

is minimized.

The output of $C_{0,f_0}(t_1, \dots, s_n)$ is calculated exactly as in (4) with one $\wedge$-gate and one $\vee$-gate using the output of $C_{0,f_0}(t_1, \dots, s_l)$, the output of $C_{0,f_0}(t_{l+1}, \dots, s_n)$ and the output of $C_{0,\wedge}(t_{l+1}, \dots, s_n)$.

Furthermore, the output of $C_{0,\wedge}(t_1, ..., s_n)$ is calculated with one $\wedge$-gate using the output of $C_{0,\wedge}(t_1, ..., s_l)$ and the output of $C_{0,\wedge}(t_{l+1}, ..., s_n)$. See Figure 3 for an illustration.

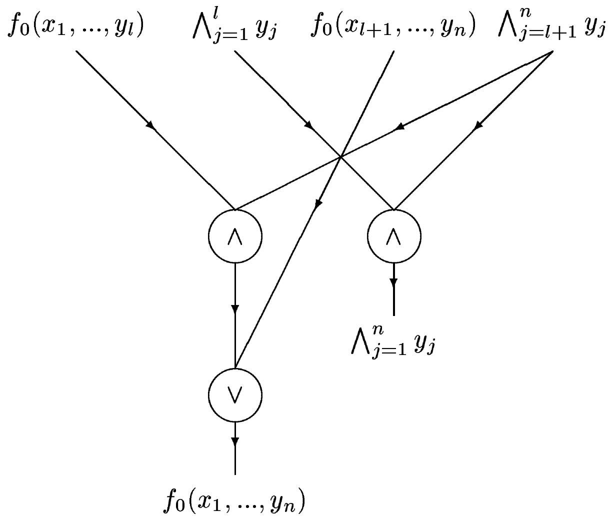

Figure 3 $C(x_1, ..., y_n)$

We collect some observations in the following lemma.

Lemma 2

(i) Algorithm I works correctly.

(ii) The number of $\vee$- or $\wedge$-gates in $C_0(t_1, ..., s_n)$ is $4n - 3$.

(iii) In $C_0(t_1, ..., s_n)$ all inputs have fan-out at most 3 and all $\wedge$- or $\vee$-gates have fan-out at most two.

(iv) $\text{delay}(C_{0,f_0}(t_1, s_1)) = \max\{t_1, s_1\} + 1$.

(v) $\text{delay}(C_{0,\wedge}(t_1, ..., s_n)) \leq \text{delay}(C_{0,f_0}(t_1, ..., s_n)) - 1$.

(vi) $\text{delay}(C_{0,f_0}(t_1, ..., s_n))$ *equals*

$$\min_{1 \leq l \leq n-1} \max\{\text{delay}(C_{0,f_0}(t_1, ..., s_l)) + 2, \text{delay}(C_{0,f_0}(t_{l+1}, ..., s_n)) + 1\}.$$

(vii) Algorithm I can be implemented to run in cubic time.

Proof: (i) follows from (4). (ii), (iii) and (iv) are obvious. (v) follows easily by induction and immediately implies (vi). (vii) follows, since Algorithm I only needs to calculate the delays of the $\binom{n}{2}$ circuits $C_{0,f_0}(t_i, s_i, ..., t_j, s_j)$ for $1 \leq i < j \leq n$ using the recursion given by (iv) and (vi). This can clearly be done in cubic time. □

In order to analyse the quality of the construction we study the recursion in Lemma 2 (iv) and (vi) in the next section.

4 Growth

For $n \geq 2$ and non-negative integers a, b, a_1, b_1, ..., a_n, $b_n \in \mathbf{N}_0$ let $\mathcal{D}_0$ be defined recursively by

$$\begin{aligned} \mathcal{D}_0(a,b) &= \max\{a,b\}+1 & (5)\\ \mathcal{D}_0(a_1,b_1,...,a_n,b_n) &= \min_{1\leq l\leq n-1} \max\{\mathcal{D}_0(a_1,b_1,...,a_l,b_l)+2, \mathcal{D}_0(a_{l+1},b_{l+1},...,a_n,b_n)+1\}. \end{aligned}$$

Clearly, this corresponds to the recursion in Lemma 2. If we define $\mathcal{D}_1$ similarly by

$$\begin{aligned} \mathcal{D}_1(a) &= a & (6)\\ \mathcal{D}_1(a_1,...,a_n) &= \min_{1\leq l\leq n-1} \max\{\mathcal{D}_1(a_1,...,a_l)+2, \mathcal{D}_1(a_{l+1},...,a_n)+1\}, \end{aligned}$$

then the following properties are immediate. In order to simplify our notation we write (A,B) to denote the vector $(a_1,a_2,...,a_{n_A},b_1,b_2,...,b_{n_B})$ where $A=(a_1,a_2,...,a_{n_A})$ and $B=(b_1,b_2,...,b_{n_B})$.

Lemma 3 *Let $a,a_1,a_2,...,a_n,a'_1,a'_2,...,a'_n,b_1,b_2,...,b_n \in \mathbf{N}_0$ be such that $a_i \leq a'_i$ for $1 \leq i \leq n$. Let $A \in \mathbf{N}_0^{n_A}$ and $B \in \mathbf{N}_0^{n_B}$ with $n_A+n_B \geq 1$. Then*

(i) $\mathcal{D}_0(a_1,b_1,...,a_n,b_n) = \mathcal{D}_1(\max\{a_1,b_1\}+1, \max\{a_2,b_2\}+1,...,\max\{a_n,b_n\}+1)$,

(ii) $\mathcal{D}_1(a_1+a,a_2+a,...,a_n+a) = \mathcal{D}_1(a_1,a_2,...,a_n)+a$,

(iii) $\mathcal{D}_1(a_1,a_2,...,a_n) \leq \mathcal{D}_1(a'_1,a'_2,...,a'_n)$ and

(iv) $\mathcal{D}_1(A,B) \leq \mathcal{D}_1(A,a,B)$.

Before we proceed to the analysis, we give a combinatorial interpretation for $\mathcal{D}_1$. Let n non-negative integers $a_1,a_2,...,a_n \in \mathbf{N}_0$ be given. We consider a rooted binary tree T with root r and exactly n leaves $u_1,u_2,...,u_n$. For every non-leaf v in T one of the two edges from v to its children is assigned a length of 1 whereas the other edge is assigned a length of 2.

The vertices $u_1,u_2,...,u_n$ are ordered in such a way that for $1 \leq i < j \leq n$ the last edge on the path from u_i to the closest common ancestor u' of u_i and u_j has length 2 and, consequently, the last edge on the path from u_j to u' has length 1. If D denotes the maximum over all $1 \leq i \leq n$ of the sum of a_i and the total length of the path from u_i to r, then $\mathcal{D}_1(a_1,a_2,...,a_n)$ equals the minimum value of D over all such binary trees. See Figure 4 for some examples of optimal trees where all edges of length 2 are pointing left.

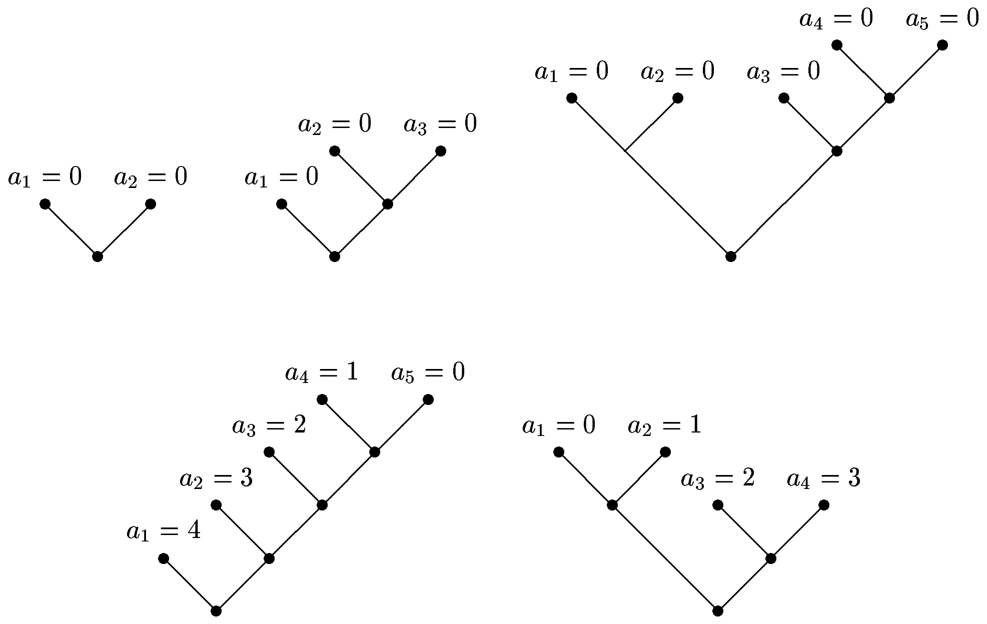

Figure 4

Let F_k denote the k-th Fibonacci number, i.e. $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$. For $k \in \mathbf{N}$ let $Z(k)$ denote the vector of k zeros.

Lemma 4 *Let $k \in \mathbf{N}_0$ and $l, n, m \in \mathbf{N}$. Let $A \in \mathbf{N}_0^n$ and $B \in \mathbf{N}_0^m$.*

(i) $\max\{i \in \mathbf{N} \mid \mathcal{D}_1(Z(i) \leq k\} = F_{k+1}$.

(ii) $\mathcal{D}_1(A, l) \leq \mathcal{D}_1(A, Z(F_{l+1}))$.

(iii) $\mathcal{D}_1(l, B) \leq \mathcal{D}_1(Z(F_{l+2}), B)$.

(iv) $\mathcal{D}_1(A, l, B) \leq \mathcal{D}_1(A, Z(F_{l+3} - 1), B)$.

Proof: (i) Let $\max(k) = \max\{i \in \mathbf{N} \mid \mathcal{D}_1(Z(i)) \leq k\}$. It is easy to verify that $\max(0) = 1$ and $\max(1) = 1$.

By (6), for $l \geq 2$ we have $\mathcal{D}_1(Z(l)) = \max\{\mathcal{D}_1(Z(l_1)) + 2, \mathcal{D}_1(Z(l_2)) + 1\}$ for some $l_1, l_2 \in \mathbf{N}$ with $l_1 + l_2 = l$. This immediately implies the recursion $\max(k) = \max(k-2) + \max(k-1)$ for $k \geq 2$ and thus we obtain $\max(k) = F_{k+1}$, which completes the proof of (i).

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

First, we assume that $\mathcal{D}_1(A, Z(F_{l+1})) = \max\{\mathcal{D}_1(A_1) + 2, \mathcal{D}_1(A_2, Z(F_{l+1})) + 1\}$ for some non-trivial A_1 and some A_2 with $(A_1, A_2) = A$.

If either A_2 is non-trivial or $l \geq 2$, then (6) and (i) or the choice of (A, l) imply the contradiction

$$\begin{aligned}\mathcal{D}_1(A, l) &\leq \max\{\mathcal{D}_1(A_1) + 2, \mathcal{D}_1(A_2, l) + 1\}\\ &\leq \max\{\mathcal{D}_1(A_1) + 2, \mathcal{D}_1(A_2, Z(F_{l+1})) + 1\}\\ &= \mathcal{D}_1(A, Z(F_{l+1})).\end{aligned}$$

If A_2 is trivial ($A_1 = A$) and $l = 1$, then $\mathcal{D}_1(A_2, l) + 1 = \mathcal{D}_1(1) + 1 = 2 \leq \mathcal{D}_1(A_1) + 2$ and we obtain a similar contradiction.

Therefore, there is some $1 \leq r \leq F_{l+1} - 1$ such that

$$\mathcal{D}_1(A, Z(F_{l+1})) = \max\{\mathcal{D}_1(A, Z(F_{l+1} - r)) + 2, \mathcal{D}_1(Z(r)) + 1\}. \tag{7}$$

By (6), we have $\mathcal{D}_1(A, l) \leq \max\{\mathcal{D}_1(A) + 2, l + 1\}$.

If $\mathcal{D}_1(A) + 2 \geq l + 1$, then (7) implies the contradiction

$$\mathcal{D}_1(A, l) \leq \mathcal{D}_1(A) + 2 \leq \mathcal{D}_1(A, Z(F_{l+1} - r)) + 2 \leq \mathcal{D}_1(A, Z(F_{l+1})).$$

Hence $l + 1 > \mathcal{D}_1(A) + 2$ and $\mathcal{D}_1(A, l) \leq l + 1$.

If $r \geq F_l + 1$, then (i) implies the contradiction

$$\mathcal{D}_1(A, l) \leq l + 1 \leq \mathcal{D}_1(Z(F_l + 1)) + 1 \leq \mathcal{D}_1(Z(r)) + 1 \leq \mathcal{D}_1(A, Z(F_{l+1})).$$

Therefore, $r \leq F_l$ which implies $F_{l+1} - r \geq F_{l-1}$. Again by (i), we obtain the contradiction

$$\mathcal{D}_1(A, l) \leq l + 1 \leq \mathcal{D}_1(Z(F_{l-1} + 1)) + 2 \leq \mathcal{D}_1(A, Z(F_{l-1})) + 2 \leq \mathcal{D}_1(A, Z(F_{l+1})).$$

This final contradiction completes the proof of (ii).

(iii) This proof is very similar to the proof of (ii) and we just include it for the sake of completeness. For contradiction, we assume that (l, B) is a counterexample of minimum length $1 + m$.

As before, this implies that there is some $1 \leq r \leq F_{l+2} - 1$ such that

$$\mathcal{D}_1(Z(F_{l+2}), B) = \max\{\mathcal{D}_1(Z(r)) + 2, \mathcal{D}_1(Z(F_{l+2} - r), B) + 1\}. \tag{8}$$

By (6), we have $\mathcal{D}_1(l, B) \leq \max\{l + 2, \mathcal{D}_1(B) + 1\}$.

If $\mathcal{D}_1(B) + 1 \geq l + 2$, then (8) implies the contradiction

$$\mathcal{D}_1(l, B) \leq \mathcal{D}_1(B) + 1 \leq \mathcal{D}_1(Z(F_{l+2} - r), B) + 1 \leq \mathcal{D}_1(Z(F_{l+2}), B).$$

Hence $l + 2 > \mathcal{D}_1(B) + 1$ and $\mathcal{D}_1(l, B) \leq l + 2$.

If $r \geq F_l + 1$, then (i) implies the contradiction

$$\mathcal{D}_1(l, B) \leq l + 2 \leq \mathcal{D}_1(Z(F_l + 1)) + 2 \leq \mathcal{D}_1(Z(r)) + 2 \leq \mathcal{D}_1(Z(F_{l+2}), B).$$

Therefore, $r \leq F_l$ which implies $F_{l+2} - r \geq F_{l+1}$. Again by part (i), we obtain the contradiction

$$\mathcal{D}_1(l, B) \leq l + 2 \leq \mathcal{D}_1(Z(F_{l+1} + 1)) + 1 \leq \mathcal{D}_1(Z(F_{l+1}), B) + 1 \leq \mathcal{D}_1(Z(F_{l+2}), B).$$

This final contradiction completes the proof of (iii).

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

As before, this implies that there is some $1 \leq r \leq F_{l+3} - 2$ such that

$$\mathcal{D}_1(A, Z(F_{l+3} - 1), B) \quad = \quad \max\{\mathcal{D}_1(A, Z(r)) + 2, \mathcal{D}_1(Z(F_{l+3} - 1 - r), B) + 1\}. \tag{9}$$

If $r \geq F_{l+1}$, then (6), (9) and (ii) imply the contradiction

$$\begin{aligned}
\mathcal{D}_1(A, l, B) &\leq \max\{\mathcal{D}_1(A, l) + 2, \mathcal{D}_1(B) + 1\} \\
&\leq \max\{\mathcal{D}_1(A, Z(F_{l+1})) + 2, \mathcal{D}_1(B) + 1\} \\
&\leq \max\{\mathcal{D}_1(A, Z(r)) + 2, \mathcal{D}_1(Z(F_{l+3} - 1 - r), B) + 1\} \\
&= \mathcal{D}_1(A, Z(F_{l+3} - 1), B).
\end{aligned}$$

Therefore, $r \leq F_{l+1} - 1$ which implies that $F_{l+3} - 1 - r \geq F_{l+2}$ and (6), (9) and (iii) imply the contradiction

$$\begin{aligned}
\mathcal{D}_1(A, l, B) &\leq \max\{\mathcal{D}_1(A) + 2, \mathcal{D}_1(l, B) + 1\} \\
&\leq \max\{\mathcal{D}_1(A) + 2, \mathcal{D}_1(Z(F_{l+2}), B) + 1\} \\
&\leq \max\{\mathcal{D}_1(A, Z(r)) + 2, \mathcal{D}_1(Z(F_{l+3} - 1 - r), B) + 1\} \\
&= \mathcal{D}_1(A, Z(F_{l+3} - 1), B).
\end{aligned}$$

This final contradiction completes the proof of (iv). □

Theorem 1 *If* $a_1, a_2, ..., a_n \in \mathbf{N}_0$*, then*

$$\begin{aligned}
\mathcal{D}_1(a_1, a_2, ..., a_n) &\leq \mathcal{D}_1\left(Z\left(\sum_{i=1}^{n}(F_{a_i+3} - 1)\right)\right) \\
&< \log_{\frac{\sqrt{5}+1}{2}}\left(\sum_{i=1}^{n} 2^{a_i}\right) + 2 \\
&\approx 1.44 \log_2\left(\sum_{i=1}^{n} 2^{a_i}\right) + 2.
\end{aligned}$$

Proof: The first inequality follows immediately from Lemma 3 and Lemma 4 (iv).

By Lemma 4 (i), $\mathcal{D}_1(Z(l)) = k$ implies that $l > F_k \geq \left(\frac{\sqrt{5}+1}{2}\right)^{k-2}$ for $k \in \mathbf{N}$ and $l \in \mathbf{N}$. Therefore, $\mathcal{D}_1(Z(l)) < \log_{\frac{\sqrt{5}+1}{2}}(l) + 2$. Since $F_{i+3} - 1 \leq 2^i$ for $i \in \mathbf{N}_0$, the remaining inequalities follow. □

Corollary 1 *If* $a_1, b_1, a_2, b_2, ..., a_n, b_n \in \mathbf{N}_0$*, then*

$$\begin{aligned}
\mathcal{D}_0(a_1, b_1, a_2, b_2, ..., a_n, b_n) &< \log_{\frac{\sqrt{5}+1}{2}}\left(\sum_{i=1}^{n}\left(2^{a_i} + 2^{b_i}\right)\right) + 3 \\
&\approx 1.44 \log_2\left(\sum_{i=1}^{n}\left(2^{a_i} + 2^{b_i}\right)\right) + 3.
\end{aligned}$$

Proof: By Lemma 3 and Theorem 1, we obtain

$$\begin{aligned}
& \mathcal{D}_0(a_1, b_1, a_2, b_2, ..., a_n, b_n) \\
= \; & \mathcal{D}_1(\max\{a_1, b_1\} + 1, \max\{a_2, b_2\} + 1, ..., \max\{a_n, b_n\} + 1) \\
= \; & \mathcal{D}_1(\max\{a_1, b_1\}, \max\{a_2, b_2\}, ..., \max\{a_n, b_n\}) + 1 \\
< \; & \log_{\frac{\sqrt{5}+1}{2}} \left(\sum_{i=1}^{n} 2^{\max\{a_i, b_i\}} \right) + 3 \\
< \; & \log_{\frac{\sqrt{5}+1}{2}} \left(\sum_{i=1}^{n} \left(2^{a_i} + 2^{b_i} \right) \right) + 3
\end{aligned}$$

and the proof is complete. □

5 Algorithm for f as in (1)

We now describe the algorithm for functions f as in (1).

Algorithm II
Input: A function f with inputs $x_1, x_2, ..., x_n$ as in (1) specified by gates $g_1, g_2, ..., g_{n-1} \in \Omega$ and an arrival time $t(x_i)$ for x_i for $1 \leq i \leq n$.
Output: A circuit C_f for f over Ω.

Step 1
Set $t_1 \leftarrow t(x_1)$ and $s_1 \leftarrow 0$.

For $1 \leq i \leq n-1$ set $t_{i+1} \leftarrow t(x_{i+1})$ and $s_{i+1} \leftarrow 0$, if $g_i = \vee$.

For $1 \leq i \leq n-1$ set $t_{i+1} \leftarrow 0$ and $s_{i+1} \leftarrow t(x_{i+1})$, if $g_i = \wedge$.

Step 2
Use Algorithm I to construct the circuit $C_{0,f_0}(t_1, s_1, t_2, s_2, ..., t_n, s_n)$ on the inputs x_1', x_1'', x_2', x_2'',..., x_n', x_n'' with arrival times t_i for x_i' and s_i for x_i'' for $1 \leq i \leq n$.

Step 3
Set $x_1' \leftarrow x_1$ and $x_1'' \leftarrow 1$.

For $1 \leq i \leq n-1$ set $x_{i+1}' \leftarrow x_{i+1}$ and $x_{i+1}'' \leftarrow 1$, if $g_i = \vee$.

For $1 \leq i \leq n-1$ set $x_{i+1}' \leftarrow 0$ and $x_{i+1}'' \leftarrow x_{i+1}$, if $g_i = \wedge$.

Step 4
The circuit C_f arises from the circuit constructed so far by eliminating all constant inputs using the relations $x \vee 0 = x \wedge 1 = x$, $x \vee 1 = 1$ and $x \wedge 0 = 0$.

Lemma 5 *Algorithm II works correctly and can be implemented to run in cubic time.*

Proof: Using the identities $x \vee y = (x \vee y) \wedge 1$ and $x \wedge y = (x \vee 0) \wedge y$, it is straightforward to check that C_f computes f (cf. Figure 5). Hence Algorithm II works correctly. Its time complexity follows from the time complexity of Algorithm I and the fact that considering each of the less than $8n-3$ $\vee$- or $\wedge$-gates of $C_{0,f_0}(t_1, s_1, t_2, s_2, ..., t_n, s_n)$ once in non-increasing distance from the output gate, Step 4 can be done in linear time. □

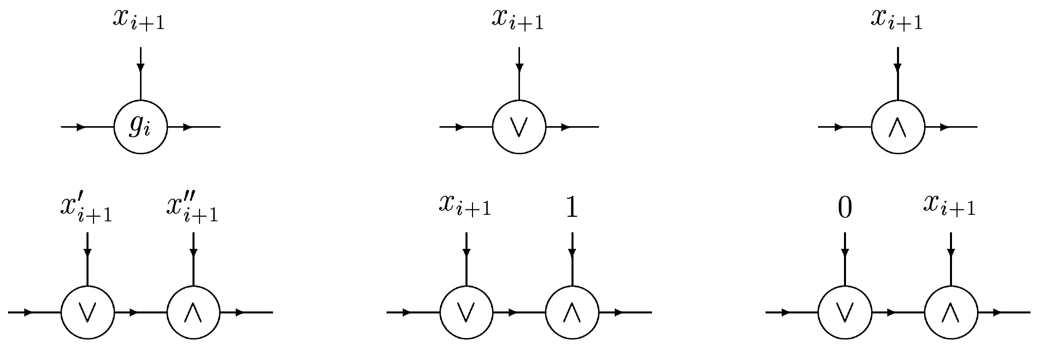

Figure 5

Theorem 2

(i) If C_{0,f_0} *denotes the circuit generated by Algorithm I for* f_0 *as in (3) given arrival times for the inputs, then* $\text{delay}(C_{0,f_0}) \leq 1.44\,\text{delay}(f_0) + 3$.

(ii) If C_f *denotes the circuit generated by Algorithm II for* f *as in (1) given arrival times for the inputs, then* $\text{delay}(C_f) \leq 1.44\,\text{delay}(f) + 4.44$.

Proof: (i) This follows immediately from Lemma 1 and Corollary 1.

(ii) Using the same notation as above, we have

$$\sum_{i=1}^{n} \left(2^{t_i} + 2^{s_i}\right) \leq 2 \sum_{i=1}^{n} 2^{t(x_i)}.$$

By Lemma 1 and Corollary 1, we obtain

$$\begin{aligned}
\text{delay}(C_f) &\leq \text{delay}(C_{0,f_0}(t_1, s_1, t_2, s_2, ..., t_n, s_n)) \\
&\leq 1.44 \log_2 \left(\sum_{i=1}^{n} \left(2^{t_i} + 2^{s_i}\right) \right) + 3 \\
&\leq 1.44 \log_2 \left(2 \sum_{i=1}^{n} 2^{t(x_i)} \right) + 3 \\
&\leq 1.44 \log_2 \left(\sum_{i=1}^{n} 2^{t(x_i)} \right) + 4.44 \\
&\leq 1.44\,\text{delay}(f) + 4.44
\end{aligned}$$

and the proof is complete. □

6 Conclusion

We have described a simple cubic-time algorithm for the construction of circuits for functions as in (1) whose delay is at most 1.44 times the lower bound plus some small constant.

As we mentioned, the functions as in (3) are closely related to addition. As a consequence, we can construct circuits over the basis $\{\vee, \wedge, \neg\}$ for the addition of two binary n-digit numbers whose delay is at most 1.44 times the optimal delay plus some small constant. Unfortunately, the number of gates of these circuits is quadratic in n. In [7] we describe circuits for the same task whose delay is essentially at most twice the lower bound and whose size is $O(n \log_2(\log_2(n)))$.

In view of the motivation explained in the first section, it is obvious that many technical details not contained in the mathematical abstraction can actually be incorporated in the algorithm. This motivation is also the reason for controlling the number of gates and the maximum fan-out.

References

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

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

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

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

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

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

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

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

[9] E.E. Swartzlander (ed.), Computer arithmetic. Vol. I, IEEE Computer Society Press, 1990, 378p.

[10] E.E. Swartzlander (ed.), Computer arithmetic. Vol. II, IEEE Computer Society Press, 1990, 396p.

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

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

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