Optimal Carry Save Networks

Michael S. Paterson * *Nicholas Pippenger* † *Uri Zwick* ‡

Abstract

A general theory is developed for constructing the asymptotically shallowest networks and the asymptotically smallest networks (with respect to formula size) for the carry save addition of n numbers using any given basic carry save adder as a building block.

Using these optimal carry save addition networks the shallowest known multiplication circuits and the shortest formulae for the majority function (and many other symmetric Boolean functions) are obtained.

In this paper, simple basic carry save adders are described, using which multiplication circuits of depth $3.71 \log n$ (the result of which is given as the sum of two numbers) and majority formulae of size $O(n^{3.21})$ are constructed. Using more complicated basic carry save adders, not described here, these results could be further improved. Our best bounds are currently $3.57 \log n$ for depth and $O(n^{3.13})$ for formula size.

1. Introduction

The question 'How fast can we multiply?' is one of the fundamental questions in theoretical computer science. Ofman-Karatsuba [12] and Schönhage-Strassen [30] (see also [1],[9],[18],[36]) tried to answer it by minimising the

*Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England. This author was partially supported by a Senior Fellowship from the SERC and by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM).

†Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada V6T 1W5. This author was partially supported by an NSERC operating grant and an ASI fellowship award.

‡Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England. This author was partially supported by a Joseph and Jeanne Nissim postdoctoral grant from Tel Aviv University and by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM). His present affiliation is the Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.

number of bit operations required, or equivalently the circuit size. A different approach was pursued by Avizienis [2], Dadda [7],[8] Ofman [20], Wallace [35] and others. They investigated the depth, rather than the size of multiplication circuits.

The main result proved by the above authors in the early 1960's was that, using a process called *Carry Save Addition*, n numbers (of linear length) could be added in depth $O(\log n)$. As a consequence, depth $O(\log n)$ circuits for multiplication and polynomial size formulae for all the symmetric Boolean functions are obtained.

They all used a component called a *'3 $\to$ 2' Carry Save Adder ($CSA_{3\to 2}$)* which reduces the sum of three numbers (of arbitrary length) to the sum of only two in a (small) constant depth. It is easy to see that using $\log_{3/2} n + O(1)$ levels of such $CSA_{3\to 2}$'s it is possible to reduce the sum of n numbers to the sum of only two. The resulting two numbers could be added (if required) using a *Carry Look Ahead Adder* (see [4],[13]) with additional depth $(1+o(1))\log m$ (where m is the length of the numbers).

In this paper we look more carefully at the construction of CSA-networks for carry save addition. A moment's reflection shows that any $CSA_{3\to 2}$-network for the carry save addition of n numbers will have at least $\log_{3/2} n$ levels of $CSA_{3\to 2}$'s. Thus, if a $CSA_{3\to 2}$ is regarded as a black box to which the three inputs should be supplied simultaneously and which then returns, after a fixed delay, the two outputs simultaneously, then this naive construction is optimal. It turns out however that the best $CSA_{3\to 2}$'s do not produce their two outputs simultaneously, nor do they require the three inputs to be supplied at the same time. The task of constructing networks with minimal total delay using such $CSA_{3\to 2}$'s becomes much more interesting.

In general we assume that we are given a $CSA_{k\to \ell}$ whose delay characteristics are described by a *delay matrix* M. The entry m_{ij} of the matrix gives the relative delay of the i-th output with respect to the j-th input. In particular, if the k inputs to a $CSA_{k\to \ell}$ are ready at times $x_1, \ldots, x_k$, we assume that the i-th output is ready at time $y_i = \max_{1\le j\le k}\{m_{ij} + x_j\}$. This corresponds to taking the $\{\max, +\}$ inner product between M and x. We show how to extract from any delay matrix M the minimal constant $\delta = \delta(M)$ such that $CSA_{k\to \ell}$-networks for the carry save addition of n numbers with a total delay of $(\delta + o(1))\log n$ can be constructed using $CSA_{k\to \ell}$'s with delay matrix M. We exhibit explicit constructions achieving this optimal behaviour.

For a given implementation of a $CSA_{k\to \ell}$ using Boolean circuitry, if we define m_{ij} to be the length of the longest path from any bit of the j-th input number to any bit of the i-th output number then the above result translates immediately to a result about depth.

Several basic designs of CSA's are described in the next section. Using these designs optimally we get U_2-circuits (circuits over the unate dyadic basis $U_2 = B_2 - \{\oplus, \equiv\}$) of depth $5.42 \log n$ and B_2-circuits (circuits over the basis B_2 of all dyadic Boolean functions) of depth $3.71 \log n$ for the carry save addition of n numbers[1]. Using more complicated CSA's described in [23],[24] these results could be improved to $4.95 \log n$ and $3.57 \log n$ respectively. As a consequence, we derive circuits of depth $5.95 \log n$ and $4.57 \log n$ for the addition of n numbers (of linear length) and for the multiplication of two n bit numbers. This improves a previous result of Khrapchenko [17] and the naive estimates of Ofman and Wallace.

Multiple addition circuits (of n numbers of n bits each) are necessarily of size $\Omega(n^2)$. Our circuits are composed of $O(n)$ CSA's each of size $O(n)$ so they have this optimal size.

The Schönhage-Strassen multiplication algorithm [30] (see also [1],[18],[36]) uses the Discrete Fourier Transform (DFT) to obtain multiplication circuits of size $O(n \log n \log\log n)$. Since the computation of a DFT essentially involves multiple additions, carry save adders could be used to implement DFT's, and therefore the whole Schönhage-Strassen algorithm, in logarithmic depth (cf. [19],[36]). The implied constant factors however are much larger than those obtained here.

Another special case of multiple addition is bit counting. A counter for n bits could be obtained by carry save adding the n input bits, treating each as a number, and then adding the two output numbers. Note that the length of the two output numbers here is $O(\log n)$ so the additional depth required to add them up in this case is only $O(\log\log n)$. As a consequence we get depth $4.95 \log n$ U_2-circuits and depth $3.57 \log n$ B_2-circuits for counting. Many symmetric Boolean functions, such as majority and MOD_k for any fixed k, can be computed in depth $o(\log n)$ once the bit count is done, so the same bounds are valid for them as well. Shallower U_2-circuits for MOD_k with $k = 3, 5, 7$ were obtained by Chin [6].

An analogous theory is developed for formula size. We assume that the formula size characteristics of a $CSA_{k\to\ell}$ are described by an *occurrence matrix* N. The entry n_{ij} gives the number of appearances of the j-th input number in the formula for the i-th output number. If the k inputs to a $CSA_{k\to\ell}$ have formula sizes $x_1, \ldots, x_k$ then the i-th output number will have formula size $y_i = \sum_{j=1}^{k} n_{ij} x_j$. Note that this corresponds to the usual $\{+, \times\}$ inner

[1] All the logarithms in this paper are taken to base 2, unless otherwise stated.

product between N and x.

Again we show how to extract from an occurrence matrix N the minimal $\epsilon = \epsilon(N)$ such that $CSA_{k\to\ell}$-networks for the carry save addition of n numbers with formula size $n^{(\epsilon+o(1))}$ can be constructed using $CSA_{k\to\ell}$'s with occurrence matrix N, and again we describe explicit constructions with optimal behaviour.

Using the CSA designs of Section 2 optimally, we get U_2-formulae of size $O(n^{4.70})$ and B_2-formulae of size $O(n^{3.21})$ for each output bit in the carry save addition of n numbers and for many symmetric Boolean functions as before. Again, using more complicated CSA's, to be described in [24], these bounds could be improved to $O(n^{4.57})$ and $O(n^{3.13})$ respectively. These constructions improve previous results of Khrapchenko [16], Pippenger [26], Paterson [21] and Peterson [25].

Relevant results on formula size are the construction of concise B_2-formulae for MOD_k with $k = 3, 5, 7$ by Van Leijenhorst [34], the construction of monotone formulae for the majority function by Valiant [33] (see also [3]), and for general threshold functions by Boppana [3] and Friedman [11].

Depth and the logarithm of formula size are closely related. It is known for example that $\log L_{B_2}(f) \le D_{B_2}(f) \le 2.47 \log L_{B_2}(f)$ (in fact even that $D_{U_2}(f) \le 2.47 \log L_{B_2}(f)$) and that $\log L_{U_2}(f) \le D_{U_2}(f) \le 1.81 \log L_{U_2}(f)$ (see [5],[28],[31]). These relations are insufficient for the derivation of optimal constants however, and we have to optimise separately for depth and for formula size. The known connections between B_2 and U_2, namely $D_{U_2}(f) \le 2D_{B_2}(f)$ and $L_{U_2}(f) \le O\left((L_{B_2}(f))^{\log_3 10}\right)$ (see [27]), are also too crude to be of any help to us.

The theories developed for depth and formula size are analogous. However some differences result from the fact that the usual $\{+, \times\}$ inner product is used in the formula size case while the less usual $\{\max, +\}$ inner product is used for depth. In particular, while the parameters that should be optimised in the formula size case are continuous, some of them are discrete in the delay case. This changes the nature of the optimisation problems involved.

A summary of the 'numerical' results obtained in this work together with the previously known results is given in Table 1.1. The right columns give the dimensions of the CSA's used. Results marked by a single star ($\star$) are obtained using building blocks described in this paper, those with a double star ($\star\star$) using building blocks described in [23], and those with ($\star\star\star$) using building blocks that will appear in [24]. In three out of the four cases we improve the previously known results even using the same CSA's used by

U_2-depth				B_2-depth			
$5.12 \log n$	Khr[17]	(1978)	$7 \to 3$	$3.71 \log n$	$\star$	(1990)	$3 \to 2$
$5.07 \log n$	$\star$	(1990)	$7 \to 3$	$3.57 \log n$	$\star\star$	(1990)	$6 \to 3$
$4.95 \log n$	$\star\star$	(1990)	$11 \to 4$				
U_2-formula size				**B_2-formula size**			
n^5	Pip[26]	(1974)	$3 \to 2$	$n^{3.54}$	Pip[26]	(1974)	$3 \to 2$
$n^{4.62}$	Khr[16]	(1972)	$7 \to 3$	$n^{3.47}$	Pat[21]	(1978)	$3 \to 2$
$n^{4.60}$	$\star$	(1990)	$7 \to 3$	$n^{3.32}$	Pet[25]	(1978)	$3 \to 2$
$n^{4.57}$	$\star\star\star$	(1990)	$11 \to 4$	$n^{3.21}$	$\star$	(1990)	$3 \to 2$
				$n^{3.16}$	$\star\star\star$	(1990)	$7 \to 3$
				$n^{3.13}$	$\star\star\star$	(1990)	$6 \to 3$

Table 1.1. Results for multiple carry save addition.

the previous authors. In all the four cases we get further improvements by designing improved *CSA*'s. The improvements we get are quite marginal in some of the cases. Our results, however, could only be improved by either improving our basic carry save adders or by designing circuits not based on carry save adders at all.

A preliminary report (with some inaccuracies) on the results described here appeared in [22]. Some results mentioned there are not covered here. One such is the construction of shallow circuits and short formulae for all symmetric Boolean functions. These are obtained by constructing circuits and formulae for counting, in which the less significant output bits are produced much before (or with smaller formula size than) the more significant ones.

2. Carry Save Adders

A *k-bit full adder* (FA_k) receives k input bits and outputs $\lceil \log(k+1) \rceil$ bits representing, in binary notation, their sum. Usually k is of the form $2^\ell - 1$.

Arrays of *FA*'s could be used to construct *CSA*'s. A construction of a $CSA_{3\to 2}$ using FA_3's, for example, is illustrated in Fig 2.1, where three input numbers a, b and c are carry save added to give the output pair u, v. The depth of the

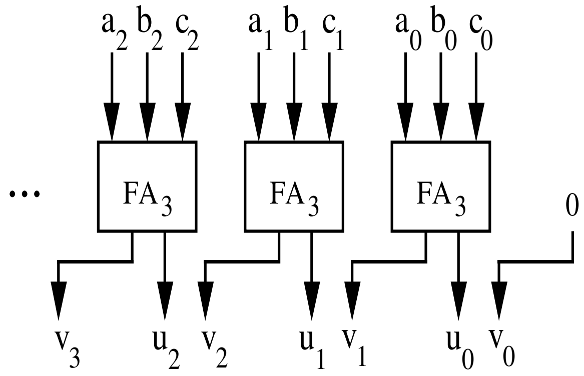

Figure 2.1. Constructing a $CSA_{3\to 2}$ using FA_3's.

CSA obtained is equal to the depth of the *FA* used and is independent of the length of the numbers to be carry save added.

A B_2-implementation of an FA_3 is given in Fig. 2.2(a). The *delay matrix* of this FA_3 which describes the relative delay of each output with respect to each input, is easily seen to be $\begin{pmatrix} 1 & 2 & 2 \\ 2 & 3 & 3 \end{pmatrix}$. The delay characteristics of the $CSA_{3\to 2}$ obtained are also described by this delay matrix. It can be checked that no other B_2-implementation of an FA_3 has a better delay matrix.

Notice that x_1 may be supplied to this FA_3 one unit of time after x_2 and x_3 are supplied, and that y_0 is obtained one unit of time before y_1. Thus, the FA_3 can be represented schematically by the 'gadget' appearing in Fig. 2.2(c).

The two formulae obtained by expanding the circuit of Fig 2.2(a) are given in Fig 2.2(b). The formula for y_1 has size 5. The variable x_1 appears only once in it while each of the variables x_2, x_3 appears twice. The *occurrence vector* of this formula is therefore $(1,2,2)$. The occurrence vector of the formula for y_0 is $(1,1,1)$. Combining these vectors we get that the *occurrence matrix* of the implementation is $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \end{pmatrix}$.

An alternative B_2-implementation of an FA_3 is given in Fig. 2.3. This implementation has a worse delay matrix $\begin{pmatrix} 1 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$ but a better occurrence matrix $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 3 \end{pmatrix}$. It can be checked that no other B_2-implementation has a better occurrence matrix.

Both the implementations of Fig. 2.2 and Fig. 2.3 are also minimal with respect to circuit size (for relevant results concerning circuit size see [29],[32] and [37]). This, however, will not happen in general. Since we are not concerned here with the size of the circuits (which will always be $O(n^2)$) we

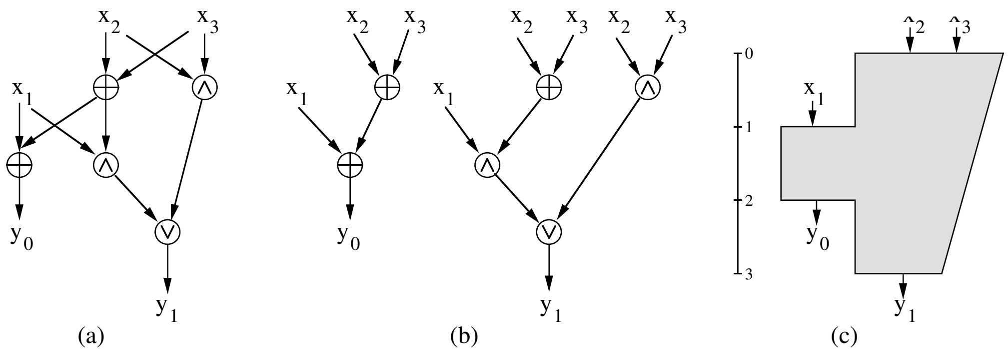

Figure 2.2. An optimal depth implementation of a B_2-FA_3.

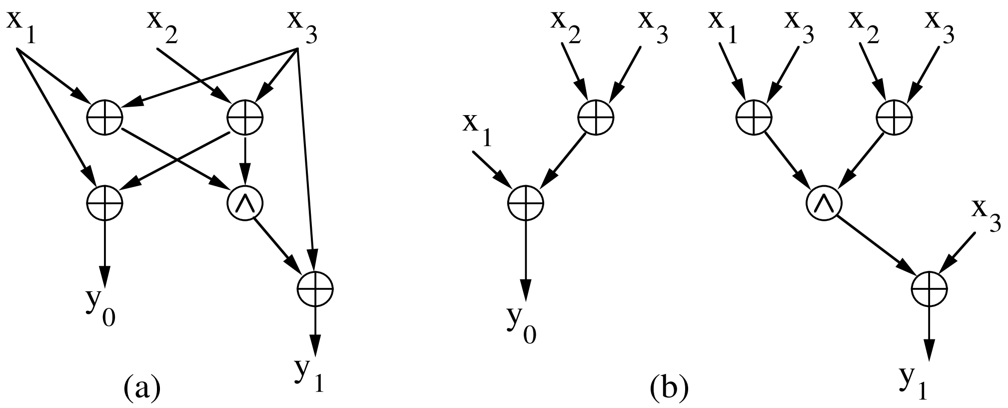

Figure 2.3. An optimal formula size implementation of a B_2-FA_3.

can think of an implementation of an *FA* as a set of formulae, one for each output bit. This also stresses the fact that the structure of each one of these formulae could be optimised separately.

A U_2-implementation of an FA_3 is given in Fig. 2.4. It has delay matrix $\begin{pmatrix}2&4&4\\2&3&3\end{pmatrix}$ and occurrence matrix $\begin{pmatrix}2&4&4\\1&2&2\end{pmatrix}$. It can be checked that both these are optimal. Note that this time it is not clear how to schedule the inputs to this unit. If x_1 is supplied two units of time after x_2, x_3, then delays are introduced in the circuitry for y_1; if x_1 is supplied one unit of time after x_2, x_3, then delays will be introduced in the y_0 circuitry. These alternatives give rise to the two gadgets shown on the right in Fig. 2.5. This is a simple example of *non-modularity*.

Using the implementation of Fig. 2.2 we get depth $3.71 \log n$ B_2-circuits for carry save addition, and using that of Fig. 2.3 we can get $O(n^{3.21})$ B_2-formulae for each output bit of carry save addition. With the implementation of Fig. 2.4 we get depth $5.42 \log n$ U_2-circuits and $O(n^{4.70})$ formulae for carry save ad-

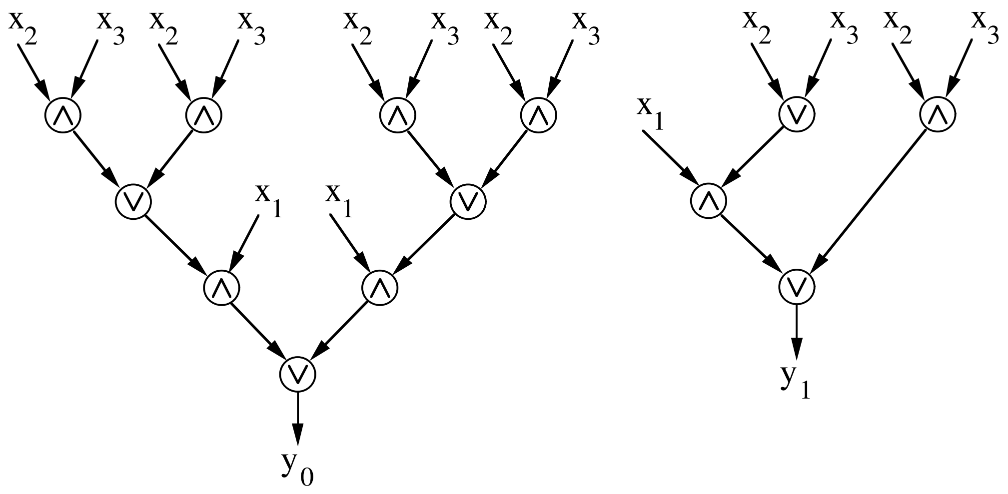

Figure 2.4. An optimal U_2-implementation of an FA_3

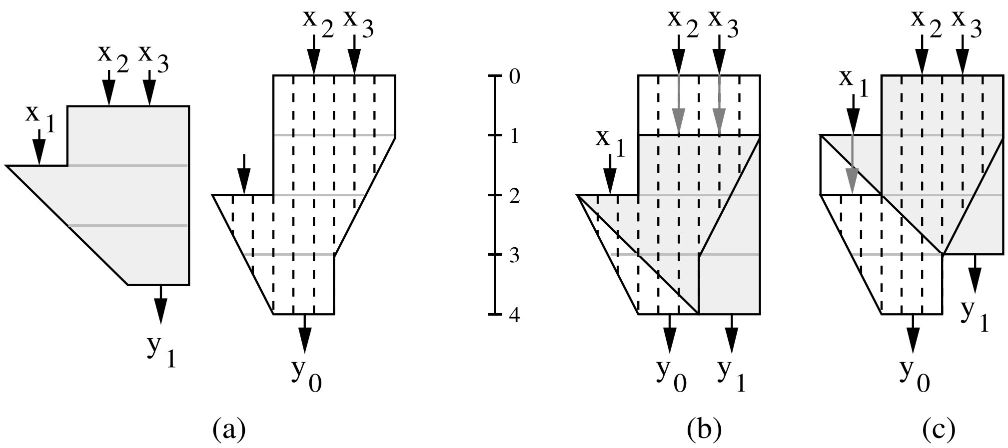

Figure 2.5. The time characteristics of the U_2-FA_3.

dition. These are the best results possible using $CSA_{3\to 2}$'s based on FA_3's. The better results stated in the previous section are obtained using more complicated building blocks.

The construction of the building blocks that are used to get our best results is technically involved. Since we want to concentrate on the general theory we will not describe their construction here. These constructions will appear in [23],[24]. We will just point out here that *CSA*'s could be built using any *bit adder*, not necessarily using full adders. Our currently best U_2 results, for example, are obtained using a bit adder that adds 7 bits with weight 1 together with 4 bits of weight 2. It is also not necessary for the result to be represented in a non-redundant form.

3. *CSA*-networks

A *CSA* is regarded henceforth as a black box with k inputs and ℓ outputs ($\ell < k$) with the property that the sum of its outputs is always equal to the sum of its inputs. The delay and formula size characteristics of a *CSA* are described by its $\ell \times k$ delay and occurrence matrices.

We assume that all the entries in the delay matrix are positive and that all the entries in the occurrence matrix are at least 1. This corresponds to the assumption that every output depends on every input and that no computation is instantaneous.

A more general setting could allow the outputs to depend on subsets of the inputs. In such cases, $-\infty$ entries will appear in the delay matrices and 0 entries will appear in the occurrence matrices. Almost all the results of this paper could be extended to cover this more general situation. The complete proofs of these results are longer and will not be presented here. Some of them appear in [23],[24]. In the sequel, whenever a delay matrix is considered, it is therefore assumed that all its entries are positive and whenever an occurrence matrix is considered it is assumed that all its entries are at least one.

A *CSA*-network is an acyclic network composed of *CSA* units of a fixed type. An inductive argument shows that any *CSA*-network has the property that the sum of its outputs is equal to the sum of its inputs.

Given the delay and occurrence matrices, M and N respectively, of the *CSA* unit used, we can assign a delay and (formula) size to each 'wire' in the network. The inputs to a network are assigned delay 0 and size 1. If the k inputs to a *CSA* have delays $x_1, \ldots, x_k$ then the i-th output of this *CSA* will have delay $y_i = \max_{1 \le j \le k}\{m_{ij} + x_j\}$. We express this by $y = M \diamond x$ where $\diamond$ denotes the $\{\max, +\}$ inner product. If the k inputs to a *CSA* have sizes $x_1, \ldots, x_k$ then the i-th output of this *CSA* will have size $y_i = \sum_{j=1}^{k} n_{ij} x_j$. We abbreviate this by writing $y = Nx$ where this time the usual $\{+, \times\}$ inner product is used. The delay (respectively, size) of a network is the maximum of the delays (respectively, sizes) of its outputs.

Given a fixed $CSA_{k \to \ell}$ with delay matrix M and occurrence matrix N, our task is to use these units to construct networks with n inputs and ℓ outputs and minimal delay or formula size. We denote by $D_M(n)$ the minimal delay of such an $n \to \ell$ network and by $F_N(n)$ the minimal (formula) size of such a network. Strictly speaking, $n \to \ell$ networks exist only if $(k - \ell)$ divides $(n - \ell)$, and then use exactly $(n - \ell)/(k - \ell)$ *CSA*'s. This condition is relaxed however if we allow constant zero inputs. Such *dummy* inputs also simplify the presentation of the constructions described in Sections 8 and 9.

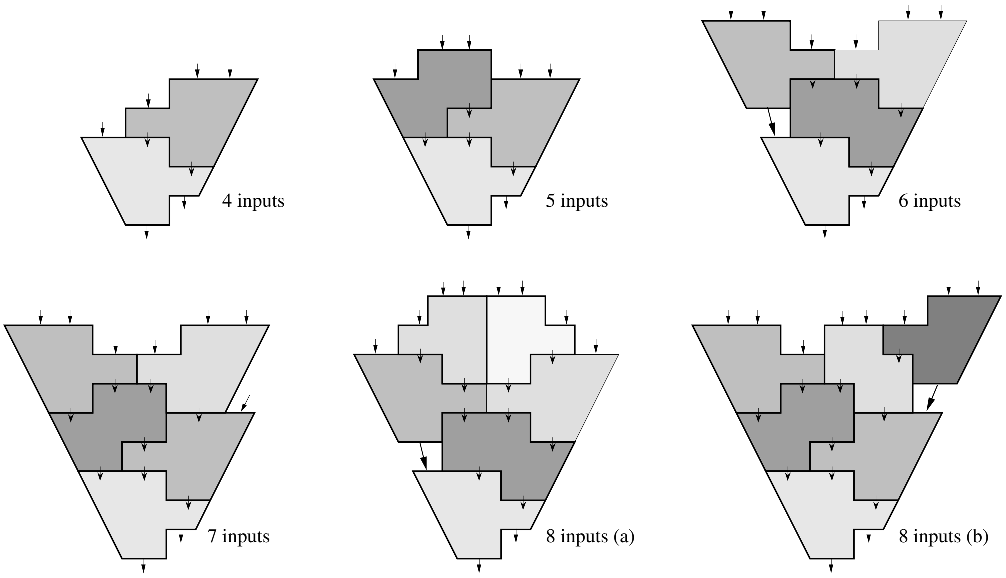

Figure 3.1. Optimal $n \to 2$ networks for $n = 4, 5, 6, 7, 8$.

The optimal networks for some small values of n, constructible using the $CSA_{3\to 2}$ of Fig. 2.2, are shown in Fig. 3.1. For every fixed CSA unit there is a polynomial time algorithm which constructs for every n the set of $n \to \ell$ networks with minimal delay.

We are interested in the asymptotic behaviour of the functions $D_M(n)$ and $F_N(n)$. The next theorem states that $D_M(n)$ behaves logarithmically and $F_N(n)$ polynomially.

Theorem 3.1 *For every delay matrix M and occurrence matrix N there exist constants $\delta(M)$ and $\epsilon(N)$ such that*

(i) $D_M(n) = (\delta(M) + o(1)) \log n$;

(ii) $F_N(n) = n^{\epsilon(N)+o(1)}$.

Proof : By collapsing first the columns then the rows of an $n \times m$ array of inputs, we see that

$$D_M(n \cdot m) \leq D_M(n) + D_M(m) + D_M(\ell^2).$$

It follows that the function $D_M(n) + D_M(\ell^2)$ behaves sub-additively (as its argument n multiplies), and thus the limit $\delta(M) = \lim_{n\to\infty} D_M(n)/\log n$ exists. By a similar argument, the limit $\epsilon(M) = \lim_{n\to\infty} \log F_N(n)/\log n$ exists. These two constants satisfy the required conditions. $\square$

Our goal in the next sections will be to determine $\delta(M)$ and $\epsilon(N)$ as functions of M and N. In the next section we give a high level overview of the results obtained in later sections.

4. Overview of results

Given a $CSA_{k\to\ell}$ with delay matrix M, we want to construct $n \to \ell$ networks with asymptotically minimal delay. This in particular will give us the value of $\delta(M)$.

Let us assume at first that the delay matrix M is *modular*, i.e., that each pair of rows (and columns) in it differs by a constant. Alternatively, M is modular if and only if it can be represented as $b - a^T$ where $b \in \mathbb{R}^\ell$ and $a \in \mathbb{R}^k$, i.e., $m_{ij} = b_i - a_j$. The time characteristics of *CSA* units with modular delay matrices could be depicted schematically by strangely-shaped boxes, as was done in Figures 2.2(c) and 3.1.

Given a modular *CSA*, our goal is to build networks in which copies of this *CSA* unit are packed together tightly (in time) so that in most connections no delays are introduced. As seen from the optimal $6 \to 2$ and $8 \to 2$ networks of Figure 3.1, delays are sometimes inevitable. We will see however that we can always connect the *CSA*'s together in such a way that the asymptotic effect of all these delays is negligible.

If $M = b - a^T$ is a modular matrix and all the entries of M are positive (this means that the first output is obtained only after the last input is supplied) then its *characteristic equation* $P_{a,b}(\lambda) = \sum \lambda^{a_j} - \sum \lambda^{b_i} = 0$ has a unique solution $\lambda = \lambda(a,b)$ in the interval $(1,\infty)$. We will prove that in d units of time, the best that we can do is to add about λ^d numbers. This immediately shows that $\delta(M) = 1/\log \lambda(a,b)$.

We next turn our attention to non-modular gadgets. We have already encountered one (see Figures 2.4 and 2.5) in our attempts to construct a U_2-FA_3. Although we cannot represent the behaviour of a non-modular gadget by a single strangely-shaped box, we can still do so for each of its outputs separately, as was done in Figure 2.5(a).

In a sense, a non-modular gadget can never be utilised fully. If a *CSA* with a non-modular matrix M gets its inputs at times specified by the vector x, and produces its outputs at times specified by the vector $y = M \diamond x$, then it essentially functions as a modular gadget with delay matrix $M' = y - x^T$. The matrix M' *dominates* M, i.e., every entry in M' is at least as large as the corresponding entry in M. If M is indeed non-modular, then at least one entry in M' is strictly larger than the corresponding entry in M.

Note that the collection of modular matrices dominating M forms a convex set. We shall see that it is in fact a polyhedron and we call it the *modular polyhedron over M*.

In a network composed of *CSA*'s with a non-modular matrix M, the behaviour of each such unit can be described by a modular matrix dominating M. These modular matrices need not all be the same. The variety of forms in which a non-modular *CSA* could function in a network is schematically obtained by superimposing the boxes describing each of its components, allowing arbitrary shifts between them. An attempt to depict this process was made in Figures 2.5(b) and (c), where the two extremal shift positions are shown.

In the case of the gadget considered in Figure 2.5 all the delays are integral. If we assume that all the inputs are available at some integral time, say time 0, then the behaviour of each such gadget in a network will be dominated by at least one of the two modular gadgets of Figure 2.5(b),(c). We already know the best result that we could get by using each of these two gadgets separately. Can we do better by using both together?

Things could be even more complicated. If some of the delays of a non-modular gadget were non-integral then intermediate, non-extremal shifts between the different components might appear useful. If some of the delays were irrational we could have an infinite number of possible shifts to consider.

Fortunately, the situation is not so complicated. We shall see that the optimal way of using a non-modular matrix M is always to choose one modular matrix M' dominating it for which $\delta(M')$ is maximal (in certain degenerate cases there may be more than one such matrix), and then to use M' optimally on its own. Furthermore, M' could always be chosen from amongst the finitely many matrices with extremal relative shifts between the different components of M, i.e., amongst the vertices of the modular polyhedron over M.

What happens if we have at our disposal a collection of r *CSA* units with delay matrices $M_1, \ldots, M_r$? Again, it turns out that optimal asymptotic results could be obtained using a single *CSA* from this collection with a maximal value of $\delta(M_i)$ in this set.

We next consider the formula size case. Given a occurrence matrix N, we want to construct $n \to \ell$ networks in which the outputs have minimum asymptotic formula size. In the depth case we had to find the optimal relative delays that inputs to the same *CSA* unit should have. Analogously, we now have to find the optimal ratios between the formula sizes of inputs that are to be fed into the same *CSA* unit.

Given a vector $x \in \mathbb{R}^k$ and a number $p > 0$, the L_p-*norm* (or *pseudo-norm* if $p < 1$) is defined to be $\|x\|_p = (\sum |x_i|^p)^{1/p}$. The L_p-norm of an $\ell \times k$ matrix N is defined to be $\|N\|_p = \max_{x \neq 0} \frac{\|Nx\|_p}{\|x\|_p}$. The norm of a matrix gives a bound on the expansion that can occur as a result of applying N (note however that x and Nx do not lie in the same space). We need an opposite measure that we call *anti-norm* that bounds the shrinkage that can occur as a result of applying N. We define the L_p-anti-norm of N to be $\langle\langle N \rangle\rangle_p = \min_{x>0} \frac{\|Nx\|_p}{\|x\|_p}$ where $x > 0$ means that all the components of x are positive.

If N is a occurrence matrix then the anti-norm $\langle\langle N \rangle\rangle_p$ is strictly increasing with p. There is therefore a unique number $p = p(N)$ for which $\langle\langle N \rangle\rangle_p = 1$. We will show that $\epsilon(N) = 1/p(N)$. Furthermore, the unique direction for which we have $\|Nx\|_p = \|x\|_p$ gives the formula size ratios to be used in the optimal construction for N.

5. Lower bounds

Define the following functions :

$$\begin{aligned} \delta'(M) &= \max\Big\{\delta > 0 : \begin{array}{c} \forall\, x \in \mathbb{R}^k \text{ and } y = M \diamond x \\ \sum_{j=1}^k 2^{x_j/\delta} \leq \sum_{i=1}^\ell 2^{y_i/\delta} \end{array} \Big\}, \\ \epsilon'(N) &= \max\Big\{\epsilon > 0 : \begin{array}{c} \forall\, x \in (\mathbb{R}^{\geq 0})^k \\ \|x\|_{1/\epsilon} \leq \|Nx\|_{1/\epsilon} \end{array} \Big\}. \end{aligned}$$

The motivation for these definitions will become clearer in the statement and proof of Theorem 5.2 below. The following technical lemma establishes the existence of $\delta'(M)$ and $\epsilon'(N)$.

Lemma 5.1 *The maxima in the definitions of $\delta'(M)$ and $\epsilon'(N)$ exist.*

Proof : For every $x \in \mathbb{R}^k$ and $y = M \diamond x$, let $A_x = \{\delta > 0 : \sum 2^{x_j/\delta} \leq \sum 2^{y_i/\delta}\}$. It is easy to see that the set $A'_x = A_x \cup \{0\}$ is closed. Let $A = \cap_{x\in\mathbb{R}^k} A_x$ and $A' = \cap_{x\in\mathbb{R}^k} A'_x = A \cup \{0\}$. The set A', being an intersection of closed sets, is also closed.

Let $m = \max\{m_{ij}\}$ be a maximal entry in M. If we choose $x = 0$ then $y_i \leq m$. If $\delta \in A_0$ then $k = \sum 2^{x_j/\delta} \leq \sum 2^{y_i/\delta} \leq \ell \cdot 2^{m/\delta}$ and, as a consequence, $\delta \leq m/\log(k/\ell)$. Thus A_0, and therefore A', is bounded. Consequently A' is compact and has a maximum.

Let $m' = \min\{m_{ij}\}$ be a minimal entry in M. We assume that all the entries in M are positive and therefore $m' > 0$. Let $x \in \mathbb{R}^k$ and $y = M \diamond x$. Let

$x_r = \max\{x_j\}$. There exists $y_s \geq x_r + m'$, so for $\delta \leq m'/\log k$ we have $\sum 2^{x_j/\delta} \leq k \cdot 2^{x_r/\delta} \leq 2^{y_s/\delta} \leq \sum 2^{y_i/\delta}$. In particular we get that $m'/\log k \in A_x$ for every $x \in \mathbb{R}^k$, and therefore $m'/\log k \in A'$.

Thus $\max A' > 0$, and therefore $\max A = \max A'$ and their common value is $\delta'(M)$. The existence of $\epsilon'(N)$ is proved in a similar manner. □

More direct proofs for the existence of these maxima could be derived from the arguments used in Sections 6 and 7.

In this section we show that $\delta'(M), \epsilon'(N)$ are lower bounds for $\delta(M), \epsilon(N)$. In Sections 6 and 7 we will show that they are also upper bounds, thus establishing that $\delta(M) = \delta'(M)$ and $\epsilon(N) = \epsilon'(N)$.

Theorem 5.2

(i) $D_M(n) \geq \delta'(M)\log(n/\ell)$;

(ii) $F_N(n) \geq (n/\ell)^{\epsilon'(N)}$.

Proof : Consider an $n \to \ell$ network composed of *CSA*'s with delay matrix M. If the inputs to a *CSA* in the network have delays $x_1, \ldots, x_k$ then the outputs will have delays $y_1, \ldots, y_\ell$ where $y = M \diamond x$. The definition of $\delta = \delta'(M)$ ensures that $\sum_{j=1}^{k} 2^{x_j/\delta} \leq \sum_{i=1}^{\ell} 2^{y_i/\delta}$. Using induction we get a similar relation for the inputs and outputs of the whole network. The n inputs have delay 0. If the ℓ outputs all have delay at most d then we get that $n \leq \ell \cdot 2^{d/\delta}$ or equivalently $d \geq \delta \log(n/\ell)$.

Similarly, if the *CSA*'s in the network have occurrence matrix N we get that the sizes of the inputs and the outputs to every *CSA* in the network satisfy the relation $\|x\|_{1/\epsilon} \leq \|y\|_{1/\epsilon}$ where $\epsilon = \epsilon'(N)$. Using induction we get a similar relation for the whole network. The n inputs have size 1. If the ℓ outputs all have size at most f then we get that $f \geq (n/\ell)^{\epsilon}$. □

The argument used in this proof is similar to the one used in a proof that a binary tree of depth ℓ can have at most 2^ℓ leaves using Kraft's inequality, that $\sum 2^{-\ell_i} \leq 1$ where the ℓ_i's are the depths of the leaves in a binary tree.

An immediate consequence is the following result.

Corollary 5.3

(i) $\delta'(M) \leq \delta(M)$;

(ii) $\epsilon'(N) \leq \epsilon(N)$.

6. The delay problem

In this section we show how to compute $\delta'(M)$ $(= \delta(M))$ for every delay matrix M. We start by showing that, for a modular delay matrix M, we can obtain $\delta'(M)$ by solving one polynomial equation. We then show that for general, non-modular delay matrices we simply have to consider a finite number of modular matrices, compute δ' for each one of these and take the maximum. The modular matrices that we have to consider are the vertices of the modular polyhedron over M (formal definitions of all these concepts appear below).

If $x \in \mathbb{R}^k$ and $y \in \mathbb{R}^\ell$, we denote by $y - x^T$ the $\ell \times k$ matrix whose elements are $y_i - x_j$. Matrices of this form are called *modular*. We define

$$P_{x,y}(\lambda) = \sum_{j=1}^{k} \lambda^{x_j} - \sum_{i=1}^{\ell} \lambda^{y_i}$$

to be the *characteristic polynomial* of the modular delay matrix $y - x^T$.

Lemma 6.1 *If* $\max x_j < \min y_i$ *and* $\ell < k$ *then the equation* $P_{x,y}(\lambda) = 0$ *has a unique root* $\lambda(x, y)$ *in the interval* $(1, \infty)$.

Proof : If $\lambda = 1$ then $P_{x,y}(1) = k - \ell > 0$ and if $\lambda \to \infty$ then $P_{x,y}(\lambda) \to -\infty$. Thus the equation has at least one root in the interval $(1, \infty)$.

Since a translation of x and y by the same amount leaves the roots invariant, we may assume, without loss of generality that $\max_j x_j < 0 < \min_i y_i$. Every positive contribution to $P_{x,y}(\lambda)$ is now decreasing in λ, while every negative contribution is increasing. Therefore $P_{x,y}(\lambda)$ as a whole is decreasing and the uniqueness of the root is guaranteed. $\square$

If M', M are two matrices and $m'_{ij} \geq m_{ij}$ for every i, j, we say that M' *dominates* M and we write $M' \geq M$.

If we replace δ in the definition of $\delta'(M)$ by $\lambda = 2^{1/\delta}$ then since $y = M \diamond x$ implies $y - x^T \geq M$, we get that $\delta'(M) = 1/\log \lambda(M)$ where

$$\lambda(M) = \min\Big\{ \lambda \geq 1 : \begin{array}{c} \forall\, x \in \mathbb{R}^k, y \in \mathbb{R}^\ell, y - x^T \geq M \\ \sum_{j=1}^{k} \lambda^{x_j} \leq \sum_{i=1}^{\ell} \lambda^{y_i} \end{array} \Big\}.$$

If M is a delay matrix with positive entries and $y - x^T \geq M$ then clearly $\max_j x_j < \min_i y_i$ and the uniqueness of $\lambda(x, y)$ follows. We therefore get that $\sum \lambda^{x_j} \leq \sum \lambda^{y_i}$ holds if and only if $\lambda \geq \lambda(x, y)$. We can thus state the definition of $\lambda(M)$ in the following form

$$\lambda(M) = \max\{\lambda(x, y) : y - x^T \geq M, x \in \mathbb{R}^k, y \in \mathbb{R}^\ell\}.$$

A pair (x, y) of vectors $x \in \mathbb{R}^k$ and $y \in \mathbb{R}^\ell$ satisfying $y - x^T \geq M$ will be called a *schedule* for M. If we impose the additional requirement that $x_1 = 0$ we get a one-to-one correspondence between schedules and modular matrices dominating M. This set of modular matrices dominating M will be denoted by $P(M)$ and will be called the *modular polyhedron* over M.

If $M = b - a^T$ is a modular matrix and $y - x^T \geq M$ then $y_i \geq \max_j\{b_i - a_j + x_j\} = b_i + c$ where $c = \max_j\{x_j - a_j\}$. If we define $x'_j = c + a_j$ then we still have $y - x'^T \geq M$, although $x' \geq x$ and therefore $\lambda(x', y) \geq \lambda(x, y)$. Since translating both x and y by the same amount leaves the roots invariant, we may assume that $c = 0$. So $x' = a, y = b$, and we have proved the following theorem.

Theorem 6.2 *If M is modular, $M = b - a^T$ say, then $\lambda(M) = \lambda(a, b)$.*

As an immediate consequence of this theorem we get that

$$\lambda(M) = \max\{\lambda(M') : M' \in P(M)\}.$$

The set $P(M)$ is defined using a finite set of linear inequalities and it is therefore a polyhedron. As mentioned before, we can identify a point $M' \in P(M)$ with the unique schedule (x, y) that satisfies $x_1 = 0$ and $y - x^T = M'$.

For every schedule (x, y), define a bipartite graph $\Gamma(x, y)$ in which the elements of x and y are the nodes and in which x_j and y_i are connected by an edge if and only if $y_i - x_j = m_{ij}$. A vertex of a polyhedron is an extremal point of it, that is, a point which is not a convex combination of any other two points in the polyhedron. It is easy to check that (x, y) is a vertex of the polyhedron $P(M)$ if and only if the graph $\Gamma(x, y)$ is connected. The polyhedron $P(M)$ has only a finite number of vertices. We denote this finite set of vertices by $P^*(M)$. Our aim is to prove that the maximum in the previous relation for $\lambda(M)$ is attained at some vertex of $P^*(M)$.

Suppose that $(x, y) \in P(M)$ is not a vertex of $P(M)$, so the graph $\Gamma(x, y)$ is disconnected. We will show that there exists a schedule (x', y') such that $\Gamma(x', y')$ is connected and $\lambda(x', y') \geq \lambda(x, y)$. Suppose that A is the set of variables in the nonempty connected component of $\Gamma(x, y)$ containing x_1. Let

B denote the complementary set of variables. We can split up the definition of $P_{x,y}(\lambda)$ in the following way

$$P_{x,y}(\lambda) = \underbrace{\left(\sum_{x_j \in A} \lambda^{x_j} - \sum_{y_i \in A} \lambda^{y_i}\right)}_{P_A(\lambda)} + \underbrace{\left(\sum_{x_j \in B} \lambda^{x_j} - \sum_{y_i \in B} \lambda^{y_i}\right)}_{P_B(\lambda)}.$$

Let $\lambda = \lambda(x, y)$, so that $P_{x,y}(\lambda) = 0$. If $P_B(\lambda)$ is negative, then decreasing the variables of B by a common constant increases $P_B(\lambda)$. While if $P_B(\lambda)$ is non-negative, then increasing these variables by a common constant does not decrease $P_B(\lambda)$. In either case the variables of B can be shifted in the appropriate direction until one more of the constraints $y_i - x_j \geq m_{ij}$ is satisfied with equality. The result is a schedule (x', y') for which $\Gamma(x', y')$ has all the edges of $\Gamma(x, y)$ together with at least one edge between A and B. Furthermore $P_{x',y'}(\lambda) \geq 0$ for $\lambda = \lambda(x, y)$, so that $\lambda(x', y') \geq \lambda(x, y)$.

By repeating this procedure we arrive at a schedule for which the graph is connected (so that the schedule corresponds to a vertex of $P(M)$), without ever decreasing λ.

We have thus proved

Theorem 6.3 *For any delay matrix M with positive entries*

$$\lambda(M) = \max\{\lambda(M') : M' \in P^*(M)\}.$$

As a simple example we may consider the delay matrix $\begin{pmatrix} 2 & 4 & 4 \\ 2 & 3 & 3 \end{pmatrix}$ corresponding to the *CSA* unit of Figure 2.5. The two gadgets in Fig. 2.5(b) and (c) are the vertices of the modular polyhedron. The gadget in (c) turns out to be the better and thus $\delta'\begin{pmatrix} 2 & 4 & 4 \\ 2 & 3 & 3 \end{pmatrix} = \delta'\begin{pmatrix} 3 & 4 & 4 \\ 2 & 3 & 3 \end{pmatrix} \simeq 5.42$. More details on this example may be found in Section 10.

As mentioned earlier, the results of this section could be generalised to cover the case in which a finite set of basic gadgets, each with an associated delay matrix, is given to us. In order to get optimal asymptotic performance it is always enough to use only one of the available gadgets.

7. The formula size problem

Our aim in this section is to prove that if N is an occurrence matrix and $\epsilon = \epsilon'(N)$ then there exists a unique direction $x \in (\mathbb{R}^{>0})^k$ (where $\mathbb{R}^{>0} = (0, \infty)$) for which $\|x\|_{1/\epsilon} = \|Nx\|_{1/\epsilon}$. The existence of such a positive direction will be needed in Section 9 where constructions with formula size achieving the lower bound from Section 5 are obtained. This special direction gives the optimal ratios between the sizes of inputs that are to be fed into a *CSA* unit with occurrence matrix N.

We will need the following lemma.

Lemma 7.1 *If N is an occurrence matrix then $\epsilon'(N) > 1$.*

Proof : It is clear that if $N' \geq N$ then $\epsilon'(N') \geq \epsilon'(N)$. The $k \times \ell$ occurrence matrix $\mathbf{1}$ all of whose entries are 1 is dominated by every other $k \times \ell$ occurrence matrix. A direct computation shows that $\epsilon'(\mathbf{1}) = \log_{k/\ell} k > 1$. □

We are now ready to prove

Theorem 7.2 *If N is an occurrence matrix and $\epsilon = \epsilon'(N)$ then there exists a unique direction $x \in (\mathbb{R}^{>0})^k$ for which $\|x\|_{1/\epsilon} = \|Nx\|_{1/\epsilon}$.*

Proof : Consider the function

$$f(x) = (\|Nx\|_{1/\epsilon})^{1/\epsilon} = \sum_i (\sum_j n_{ij} x_j)^{1/\epsilon}$$

over the set $B = \{x \in (\mathbb{R}^{\geq 0})^k : \|x\|_{1/\epsilon} = 1\}$. The function f is continuous and the set B is compact so the function f has a minimal point x^* in B. If $f(x^*) < 1$ we get a contradiction to the requirement that $\|x\|_{1/\epsilon} \leq \|Nx\|_{1/\epsilon}$ for every $x \in (\mathbb{R}^{\geq 0})^k$. If $f(x^*) > 1$ we get a contradiction to the maximality of ϵ. Therefore $f(x^*) = 1$. This establishes the existence of a direction in $(\mathbb{R}^{\geq 0})^k$ with the required equality.

In order to prove the uniqueness of x^*, define $x'_j = x_j^{1/\epsilon}$ and $n'_{ij} = n_{ij}^{1/\epsilon}$. We now have

$$f(x) = f'(x') = \sum_i (\sum_j (n'_{ij} x'_j)^\epsilon)^{1/\epsilon}$$

and the domain is $B' = \{x \in (\mathbb{R}^{\geq 0})^k : \sum_j x'_j = 1\}$. A minimum point of f on B corresponds to a minimum point of f' on B'. Lemma 7.1 tells us that $\epsilon > 1$. It is well known that for $\epsilon > 1$ the L_ϵ norm, $\|x\|_\epsilon = (\sum_j |x_j|^\epsilon)^{1/\epsilon}$, is strictly convex, that is, $\|\alpha x + (1-\alpha) y\|_\epsilon < \alpha \|x\|_\epsilon + (1-\alpha)\|y\|_\epsilon$ for $0 < \alpha < 1$, provided that x is not a scalar multiple of y. For every i the function $f'_i(x') = (\sum_j (n'_{ij} x'_j)^\epsilon)^{1/\epsilon}$ is obtained from $\|x\|_\epsilon$ by scaling and it is therefore also strictly convex. The function $f'(x')$ is obtained by summing strictly convex functions and it is therefore strictly convex also. The set B' is convex and therefore f' has a unique minimum point on it.

Finally, we prove that the minimum point x^* lies in the interior of B' so that none of its co-ordinates is 0. Let $f''_i(x') = (f'_i(x'))^\epsilon = \sum_j (n'_{ij}x'_j)^\epsilon$. Suppose, on the contrary, that one of the co-ordinates of x^* were zero. We know of course that at least one of the co-ordinates of x^* is non-zero. Without loss of generality assume that $x^*_1 = 0, x^*_2 > 0$. It is easy to check that the function $(n_{i1}\Delta)^\epsilon + (n_{i2}(x_2 - \Delta))^\epsilon$ has a negative derivative at $\Delta = 0$. For small values of Δ we would therefore get that $f''_i(x^*_\Delta) < f''_i(x^*)$, or equivalently $f'_i(x^*_\Delta) < f'_i(x^*)$, where $x^*_\Delta = (\Delta, x_2 - \Delta, \ldots, x_n)$. Since this holds for every i we get that for sufficiently small Δ, $f'(x^*_\Delta) < f'(x^*)$ which contradicts the minimality of x^*. $\square$

The strict convexity of the functions involved makes the numerical task of finding $\epsilon = \epsilon'(N)$ and the direction $x = x(N)$ satisfying $\|x\|_{1/\epsilon} = \|Nx\|_{1/\epsilon}$ an easy one.

8. Depth constructions

As we saw in Section 6, for every delay matrix M there exists a modular delay matrix M' which dominates it and for which $\lambda(M) = \lambda(M')$. It is therefore enough to consider in this section only modular gadgets. A general modular gadget is shown in Figure 8.1. It has the modular delay matrix $M = b - a^T$ where we assume that $0 = a_1 \le a_2 \le \ldots a_k < b_1 \le b_2 \le \ldots \le b_\ell$ and $\ell < k$. We assume here that all the outputs are produced after all the inputs have been supplied, which is the case if all entries in the delay matrix are positive. As mentioned in Section 3, the results of this section could be extended to cover more general cases but we shall not do so here.

The characteristic equation of this gadget is

$$\sum_{j=1}^{k} \lambda^{a_j} - \sum_{i=1}^{\ell} \lambda^{b_i} = 0.$$

We know that this equation has a unique root in the interval $(1, \infty)$ which we denote by $\lambda = \lambda(a, b)$.

Our goal in this section is to prove the following theorem.

Theorem 8.1 *If $M = b - a^T$ is a modular delay matrix then $D_M(n) \le (1 + o(1)) \log_\lambda n$ where $\lambda = \lambda(a, b)$.*

This immediately gives us the following.

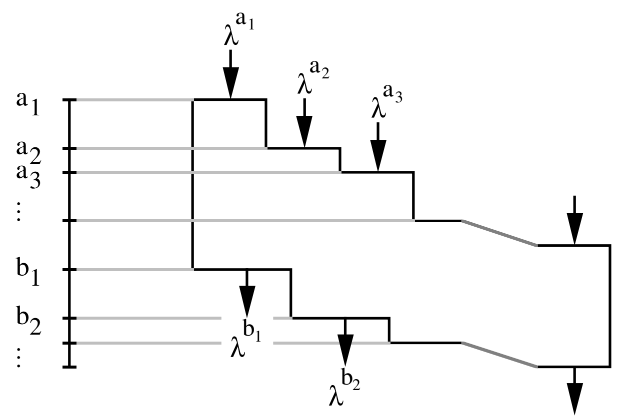

Figure 8.1. A general modular gadget.

Corollary 8.2 *For every delay matrix M*

$$\delta(M) = \delta'(M) = \max\Big\{\delta > 0 : \begin{array}{c} \forall\, x \in \mathbb{R}^k \text{ and } y = M \diamond x \\ \sum_{j=1}^{k} 2^{x_j/\delta} \leq \sum_{i=1}^{\ell} 2^{y_i/\delta} \end{array}\Big\}.$$

Proof (of Theorem 8.1) :

We first consider an important special case.

Case 1. The elements of a, b are *integral*.

A *CSA* is said to lie at *level* d in a network if its inputs are supplied at times $d + a_1, \ldots, d + a_k$. Its outputs are then available at times $d + b_1, \ldots, d + b_\ell$. Since we assume here that all the delays are integral we only have to consider integral levels.

We can describe the essentials of a network by specifying the number C_d of level-d *CSA*'s used in it. The *balance* of signals generated and consumed at time d is given by

$$Bal_d = \underbrace{\sum_{j=1}^{k} C_{d-a_j}}_{\text{consumed}} - \underbrace{\sum_{i=1}^{\ell} C_{d-b_i}}_{\text{generated}} .$$

If $Bal_d > 0$ then the network accepts Bal_d new inputs at time d. If $Bal_d < 0$ then the network yields $|Bal_d|$ outputs at time d. Note that any network specified in such a way uses every signal immediately it is generated.

The choice $C_d = n\lambda^{-d}$ would yield $Bal_d \equiv 0$ and therefore represents an equilibrium. At each time the number of signals produced is equal to the number of signals consumed. The number of signals processed at time d is $\alpha \cdot n\lambda^{-d}$ where $\alpha = \sum \lambda^{a_j} = \sum \lambda^{b_i}$. The number of signals processed at time 0 is therefore $\Omega(n)$ and the number of signals processed at time $\log_\lambda n$ is $O(1)$. To turn this into an actual construction we have to tackle two problems. First, the above process is infinite, it begins at time $-\infty$ and never ends. Secondly, the number of gadgets that should be used at each time unit is generally not integral. These problems are however easy to overcome.

We choose

$$C_d = \begin{cases} \lceil n\lambda^{-d} + c\rceil & \text{if } 1 \le n\lambda^{-d} \le n \\ 0 & \text{otherwise} \end{cases}$$

where $c = (\ell - 1)/(k - \ell)$.

We now verify the following facts.

(i) If $0 \le d < b_1$ then

$$Bal_d = \textstyle\sum_{a_j \le d} \lceil n\lambda^{a_j - d} + c\rceil \ge n\lambda^{-d},$$

(since $a_1 = 0$) so we may input at least n inputs at time 0, and even some more at times $1, \ldots, a_k$.

(ii) If $b_1 \le d \le \lfloor \log_\lambda n \rfloor$ then

$$\begin{aligned} Bal_d &\ge \textstyle\sum_{j=1}^{k} \lceil n\lambda^{a_j - d} + c\rceil - \sum_{i=1}^{\ell} \lceil n\lambda^{b_i - d} + c\rceil \\ &> (n\lambda^{-d} \textstyle\sum_{j=1}^{k} \lambda^{a_j}) + kc - (n\lambda^{-d} \sum_{i=1}^{\ell} \lambda^{b_i}) - \ell(c+1) \\ &= kc - \ell(c+1) = -1. \end{aligned}$$

Since Bal_d is integral we get that $Bal_d \ge 0$. In other words no outputs are produced at times less than or equal to $\lfloor \log_\lambda n \rfloor$.

(iii) If $\lfloor \log_\lambda n \rfloor < d \le \lfloor \log_\lambda n \rfloor + b_\ell$ then

$$\begin{aligned} Bal_d &\ge -\textstyle\sum_{i=1}^{\ell} C_{d-b_i} \\ &\ge -\textstyle\sum_{i=1}^{\ell} \lceil n\lambda^{b_i - d} + c\rceil \\ &\ge -\lambda^{-d'} \textstyle\sum_{i=1}^{\ell} \lambda^{b_i} - \ell(c+1) \quad \text{where} \quad d' = d - \lceil \log_\lambda n \rceil. \end{aligned}$$

We therefore get a total of at most $L = \frac{1}{\lambda - 1} \sum_{i=1}^{\ell} \lambda^{b_i} + \ell b_\ell (c+1)$ outputs at times less than or equal to $\lfloor \log_\lambda n \rfloor + b_\ell$. Note that L is independent of n.

(iv) Finally, if $d > \lfloor \log_\lambda n \rfloor + b_\ell$ then $Bal_d = 0$ so no more outputs are produced.

A final stage with $O(1)$ delay could now be used to reduce the L outputs obtained to only ℓ. Thus, if a, b are integral then we even have $D_M(n) \le \log_\lambda n + O(1)$.

Note that without the assumption that $a_1 \le a_2 \le \ldots \le a_k < b_1 \le \ldots \le b_\ell$ this construction may fail. It will not always be guaranteed that $Bal_d \ge 0$ for $1 \le d \le b_\ell$.

Case 2. The elements of a, b are arbitrary positive real numbers.

Consider the integer schedule

$$\begin{aligned}\overline{a} &= (\lfloor qa_1 \rfloor, \lfloor qa_2 \rfloor, \ldots, \lfloor qa_k \rfloor),\\ \overline{b} &= (\lceil qb_1 \rceil, \lceil qb_2 \rceil, \ldots, \lceil qb_\ell \rceil),\end{aligned}$$

where $q = q(n) \to \infty$ as $n \to \infty$. Note that $\overline{\lambda} = \lambda(\overline{a}, \overline{b}) \to \lambda^{1/q}$ as $q \to \infty$. The construction given in Case 1 produces $\overline{L} = O(q)$ outputs, all within time $\lfloor \log_{\overline{\lambda}} n \rfloor + \overline{b}_\ell = q(\log_\lambda n + O(1))$. Reducing the timescale in this construction by a factor of q throughout, produces a network of delay $\log_\lambda n + O(1)$ with $O(q)$ outputs. The final stage to reduce these outputs to ℓ requires a delay of $O(\log q)$. This construction satisfies the theorem provided that q is chosen so that $\log q = o(\log n)$. □

9. Formula size constructions

Let N be an occurrence matrix with entries greater than or equal to one. Choose a vector $x \in (\mathbb{R}^{>0})^k$ and define the following two vectors

$$\begin{aligned}a_j(x) &= \log x_j && \text{for } 1 \le j \le k,\\ b_i(x) &= \log \textstyle\sum_{j=1}^k n_{ij} x_j && \text{for } 1 \le i \le \ell.\end{aligned}$$

Associate with every *CSA* with occurrence matrix N, a delay matrix $M(x) = b(x) - a(x)^T$. Note that all the entries of $M(x)$ are positive.

Suppose that Γ is a network composed of *CSA*'s with occurrence matrix N, and therefore with delay matrix $M(x)$. To every wire w in Γ we can now assign both a size $f(w)$ and a delay $d(w)$. We generalise the observation that the size of a formula is at most 2 to the power of its depth.

Lemma 9.1 *For any wire w in Γ we have $f(w) \le 2^{d(w)}$.*

Proof : If w is an input wire then $f(w) = 1$ and $d(w) = 0$, so the relation holds with equality. Suppose now that $u_1, \ldots, u_k$ are the inputs of some *CSA* in the network, that $f(u_j) \le 2^{d(u_j)}$ for every j, and that $v_1, \ldots, v_\ell$ are the outputs of this *CSA*. Let $t = \max_j\{d(u_j) - a_j\}$. Then t may be regarded as the 'time' at which this *CSA* is activated. The delays of the inputs satisfy

$d(u_j) \le t + a_j$, while the delays of the outputs satisfy $d(v_i) = t + b_i$. The size of v_i will now be

$$f(v_i) = \sum_j n_{ij} f(u_j) \le \sum_j n_{ij} 2^{d(u_j)} \le \sum_j n_{ij} 2^{t+a_j} = 2^t \sum_j n_{ij} x_j = 2^{t+b_i} = 2^{d(v_i)}$$

and the claim of the lemma follows by induction. □

Now let $x \in (\mathbb{R}^{>0})^k$ be a vector that satisfies $\|x\|_{1/\epsilon} = \|Nx\|_{1/\epsilon}$ where $\epsilon = \epsilon'(N)$. The existence of such a vector was proved in Section 7. The constructions of the previous section give $n \to \ell$ networks composed of CSA's with delay matrix $M = M(x)$ (and occurrence matrix N) with depth $(\delta(M) + o(1)) \log n$. The previous lemma shows that these networks have formula size $n^{\delta(M)+o(1)}$.

Finally, we have $\delta(M) = \delta'(M) = \epsilon'(N)$ since M is a modular matrix, and therefore $\delta'(M) = \delta$ satisfies $\sum 2^{a_j/\delta} = \sum 2^{b_i/\delta}$, or equivalently $\sum x_j^{1/\delta} = \sum \left(\sum_j n_{ij} x_j\right)^{1/\delta}$. We thus get

Theorem 9.2 *For every occurrence matrix N we have $F_N(n) \le n^{\epsilon'(N)+o(1)}$.*

An immediate consequence is the following result.

Corollary 9.3 *For every occurrence matrix N*

$$\epsilon(N) = \epsilon'(N) = \max\{\epsilon > 0 : \forall\, x \in (\mathbb{R}^{\ge 0})^k,\ \|x\|_{1/\epsilon} \le \|Nx\|_{1/\epsilon}\}.$$

10. Numerical examples

The delay matrix of the FA_3 described in Fig. 2.2(a) is the modular matrix $M = (2\ 3)^T - (1\ 0\ 0)$. Therefore $\lambda(FA_3)$ is the unique positive root of the cubic equation $\lambda^3 + \lambda^2 - \lambda - 2 = 0$. We can verify that $\lambda \simeq 1.2056$ and that $\log_\lambda n \simeq 3.71 \log n$.

The delay matrix of the FA_3 described in Fig. 2.4 is the non-modular matrix $\left(\begin{smallmatrix} 2 & 4 & 4 \\ 2 & 3 & 3 \end{smallmatrix}\right)$. The two vertices of the modular polyhedron are $\left(\begin{smallmatrix} 2 & 4 & 4 \\ 2 & 4 & 4 \end{smallmatrix}\right) = (4\ 4)^T - (2\ 0\ 0)$ and $\left(\begin{smallmatrix} 3 & 4 & 4 \\ 2 & 3 & 3 \end{smallmatrix}\right) = (4\ 3)^T - (1\ 0\ 0)$ (see Fig. 2.5). The characteristic equation of the first vertex is $2\lambda^4 - \lambda^2 - 2 = 0$ and its root is $\lambda_1 = \sqrt{\frac{1+\sqrt{17}}{2}} \simeq 1.1317$. The characteristic equation of the second vertex is $\lambda^4 + \lambda^3 - \lambda - 2 = 0$ and its root is $\lambda_2 \simeq 1.1365$. The second possibility is clearly better and the circuits that we could build would have depth $\log_{\lambda_2} n \simeq 5.42 \log n$.

Khrapchenko [17] designed a U_2-FA_7 with delay matrix $\begin{pmatrix} 4 & 6 & 6 & 6 & 6 & 6 & 6 \\ 5 & 6 & 6 & 7 & 7 & 7 & 7 \\ 5 & 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix}$. Using ad hoc methods he was able to construct with it networks of depth $5.12 \log n$. The delay matrix of Khrapchenko's FA_7 is non-modular. The optimal vertex in the modular polyhedron of this matrix is $\begin{pmatrix} 5 & 6 & 6 & 6 & 6 & 6 & 6 \\ 6 & 7 & 7 & 7 & 7 & 7 & 7 \\ 5 & 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix}$ and therefore $\lambda(FA_7)$ is the unique positive root of the equation $\lambda^7 + 2\lambda^6 - \lambda - 6 = 0$. We find that $\lambda \simeq 1.1465$ and that $\log_\lambda n \simeq 5.07 \log n$. We can thus improve Khrapchenko's construction even using his own gadget. We can further reduce the depth of the U_2-circuits to $4.95 \log n$ using the novel design of $CSA_{11\to 4}$ described in [23],[24].

The size of the optimal formulae for multiple carry save addition that can be obtained using CSA's based on the FA_3 described in Fig. 2.2 is $n^{\epsilon+o(1)}$ where

$$\epsilon = \max \left\{ \frac{1}{p} \ : \ \begin{array}{c} \forall x_1, x_2, x_3 \geq 0 \\ x_1^p + x_2^p + x_3^p \leq (x_1 + x_2 + x_3)^p + (x_1 + x_2 + 3x_3)^p \end{array} \right\}$$

A numerical solution gives $\epsilon \simeq 3.2058$ and equality is achieved when $x_1 = x_2 = 1$, $x_3 \simeq 0.3926$. This yields formulae of size $O(n^{3.21})$. As mentioned before we can get better results using more complicated CSA's.

11. Concluding remarks

Many related open problems still remain. The hardest of them all is probably to determine the exact depth and formula size of multiplication, multiple addition and multiple carry save addition. In this paper upper bounds and a restricted form of lower bounds on these complexities were obtained. Although the upper bounds obtained may be close to the real values, they can probably be further improved by devising better basic building blocks.

The best non-restricted lower bounds currently known for the formula size of the majority function are $\Omega(n \log n)$ in the B_2 case, due to Fischer, Meyer and Paterson [10], and $\Omega(n^2)$ in the U_2 case due to Khrapchenko [14],[15] (see also [38]). From these, $\Omega(m \cdot n \log n)$ and $\Omega(m \cdot n^2)$ lower bounds on the B_2 and U_2 formula sizes of one of the most significant output bits in the result of a multiple addition of n m-bit numbers are easily obtained. Note that such a bit, regarded as a function of the $n \times m$ input matrix, contains as a subfunction the majority function of nearly every column in the input matrix.

For multiple carry save addition a trivial $\Omega(n \log n)$ lower bound can be obtained by noticing that each set of two consecutive output pairs must depend on almost all the $n \log n$ input bits within $\log n$ positions to the right.

It can be checked that if $2^{n+1} \equiv 1 \pmod{k}$, then the $(n+1)$-st output bit of $x \cdot \left\lfloor \frac{2^{n+1}}{k} \right\rfloor$ is 1 if and only if $x > 0$ and $x \bmod k = 0$, or if $x \bmod k > \left\lceil \frac{k}{2} \right\rceil$. Using the methods of Fischer, Meyer and Paterson and of Khrapchenko, referred to earlier, we can therefore get an $\Omega(n \log n)$ B_2 lower bound and an $\Omega(n^2)$ U_2 lower bound for the formula size of multiplication.

In the U_2 case, the $\Omega(n^2)$ formula size lower bounds on these problems imply non-trivial $2 \log n - O(1)$ lower bounds for depth.

As can be seen, there are still large gaps between the upper and lower bounds and narrowing these gaps, especially by improving the lower bounds, is a challenging problem.

References

[1] Aho A.V., Hopcroft J.E., Ullman J.D., *The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974.*

[2] Avizienis A., *Signed-digit number representation for fast parallel arithmetic. IEEE Trans. Elec. Comp. Vol. EC10 (1961), pp. 389-400.*

[3] Boppana R.B., *Amplification of probabilistic Boolean formulas. Advances in Computing Research, Vol. 5 on Randomness and Computation, Editor S. Micali, JAI Press, Greenwich, Conn., pp. 27-45.*

[4] Brent R., *On the addition of binary numbers. IEEE Trans. on Comp., Vol. C-19 (1970), pp. 758-759.*

[5] Brent R., Kuck D.J., Maruyama K., *The parallel evaluation of arithmetic expressions without division. IEEE Trans. on Comp., Vol. C-22 (1973), pp. 532-534.*

[6] Chin A., *On the depth complexity of the counting functions. Information Processing Letters 35 (1990) 325-328.*

[7] Dadda L., *Some schemes for parallel multipliers. Alta Frequenza, Vol. 34 (1965), pp. 343-356.*

[8] Dadda L., *On parallel digital multipliers. Alta Frequenza, Vol. 45 (1976), pp. 574-580.*

[9] Dunne P.E., *The Complexity of Boolean Networks. Academic Press, 1988.*

[10] Fischer M.J., Meyer A.R., Paterson M.S., $\Omega(n \log n)$ *lower bounds on length of Boolean formulas. SIAM J. Comput., Vol. 11 (1982), pp. 416-427.*

[11] Friedman J. *Constructing $O(n \log n)$ size monotone formulae for the k-th threshold function on n Boolean variables. SIAM J. Comput., Vol. 15 (1986), pp. 641-654.*

[12] Karatsuba A., Ofman Y., *Multiplication of multidigit numbers on automata. Soviet Physics Dokl., Vol. 7 (1963), pp. 595-596.*

[13] Khrapchenko V.M., *Asymptotic estimation of addition time of a parallel adder. Problemy Kibernet., Vol. 19 (1967), pp. 107-122* (in Russian). English translation in *Syst. Theory Res., Vol. 19 (1970), pp. 105-122.*

[14] Khrapchenko V.M., *Complexity of the realization of a linear function in the class of π-circuits. Mat. Zametki, Vol. 9 (1971), pp. 35-40* (in Russian). English translation in *Math. Notes Acad. Sciences USSR, Vol. 9 (1971), pp. 21-23.*

[15] Khrapchenko V.M., *Methods of determining lower bounds for the complexity of π-schemes. Mat. Zametki, Vol. 10 (1972), pp. 83-92* (in Russian). *Math. Notes Acad. Sciences USSR, Vol. 10 (1972), pp. 474-479* (English translation).

[16] Khrapchenko V.M., *The complexity of the realization of symmetrical functions by formulae. Mat. Zametki Vol. 11 (1972) pp. 109-120* (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR Vol. 11 (1972) pp. 70-76.*

[17] Khrapchenko V.M., *Some bounds for the time of multiplication. Problemy Kibernet., Vol. 33 (1978), pp. 221-227* (in Russian).

[18] Knuth D.E., *The Art of Computer Programming, Vol. 2, Seminumerical algorithms (second edition). Addison-Wesley.*

[19] Mehlhorn K., Preparata F.P., *Area-time optimal VLSI integer multiplier with minimum computation time. Information and Control, Vol. 58 (1983), pp. 137-156.*

[20] Ofman Y., *On the algorithmic complexity of discrete functions. Doklady Akademii Nauk SSSR, 145 pp. 48-51* (in Russian). English translation in *Sov. Phys. Doklady, Vol. 7 (1963) pp. 589-591.*

[21] Paterson M.S., *New bounds on formula size. Proc. of 3rd GI conference on Theoretical Computer Science 1977, Lecture Notes in Computer Science 48, Springer-Verlag 1977, pp. 17-26.*

[22] Paterson M.S., Pippenger N., Zwick U., *Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions. Proceedings of the 31st IEEE Symposium on the Foundations of Computer Science (FOCS), St. Louis 1990.*

[23] Paterson M.S., Zwick U., *Shallow multiplication circuits. Proceedings of the 10th IEEE Symposium on Computer Arithmetic (ARITH), pp. 28-34, Grenoble 1991.*

[24] Paterson M.S., Zwick U., *Shallow circuits and concise formulae for multiple addition and multiplication.* In preparation.

[25] Peterson G.L., *An upper bound on the size of formulae for symmetric Boolean functions. TR 78-03-01, University of Washington.*

[26] Pippenger N., *Short formulae for symmetric functions. IBM report RC-5143, Yorktown Heights, NY (November 20, 1974).*

[27] Pratt V.R., *The effect of basis on size of Boolean expressions. Proc. 16th IEEE Symposium on FOCS (1975), pp. 119-121.*

[28] Preparata F.P., Muller D.E., *Efficient parallel evaluation of Boolean expressions. IEEE Trans. Comp., Vol. C-25 (1976), pp. 548-549.*

[29] Red'kin N.P., *Minimal realizations of a binary adder. Problemy Kibernet., Vol. 38 (1981), pp. 181-216,272* (in Russian).

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

[31] Spira P.M., *On time-hardware complexity tradeoffs for Boolean functions. Proc. 4th Hawaii Int. Symp. on System Sciences (1971), 525-527.*

[32] Stockmeyer L., *On the combinational complexity of certain symmetric Boolean functions. Math. Syst. Theory, Vol. 10 (1977), pp. 323-336.*

[33] Valiant L.G., *Short monotone formulae for the majority function. Journal of Algorithms, Vol. 5 (1984), pp. 363-366.*

[34] Van Leijenhorst D.C., *A note on the formula size of the "mod k" functions. Info. Proc. Letters, Vol. 24 (1987) pp. 223-224.*

[35] Wallace C.S., *A suggestion for a fast multiplier. IEEE Trans. Electronic Comp. EC-13 (1964) pp. 14-17.*

[36] Wegener I., *The Complexity of Boolean Functions. Wiley-Teubner Series in Computer Science, 1987.*

[37] Zwick U., *A* $4n$ *lower bound on the combinational complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions. SIAM J. Comput., Vol. 20 (1991), pp. 499-505.*

[38] Zwick U., *An extension of Khrapchenko's theorem.* Information Processing *Letters 37 (1991) 215-217.*