SHALLOW CIRCUITS AND CONCISE FORMULAE FOR MULTIPLE ADDITION AND MULTIPLICATION

Michael Paterson, Uri Zwick

Abstract. A theory is developed for the construction of carry-save networks with minimal delay, using a given collection of carry-save adders each of which may receive inputs and produce outputs using several different representation standards.

The construction of some new carry-save adders is described. Using these carry-save adders optimally, as prescribed by the above theory, we get $\{\wedge, \vee, \oplus\}$-circuits of depth $3.48 \log_2 n$ and $\{\wedge, \vee, \neg\}$-circuits of depth $4.95 \log_2 n$ for the carry-save addition of n numbers of arbitrary length. As a consequence we get multiplication circuits of the same depth. These circuits put out two numbers whose sum is the result of the multiplication. If a single output number is required then the depth of the multiplication circuits increases respectively to $4.48 \log_2 n$ and $5.95 \log_2 n$.

We also get $\{\wedge, \oplus, \neg\}$-formulae of size $O(n^{3.13})$ and $\{\wedge, \neg\}$-formulae of size $O(n^{4.57})$ for all the output bits of a carry-save addition of n numbers. As a consequence we get formulae of the same size for the majority function and many other symmetric Boolean functions.

Key words. Multiplication, carry-save addition, circuits, formulae.
Subject classifications. 68Q25, 06E30, 94C10.

1. Introduction

A carry-save adder (or *CSA* for short) is a unit with k inputs and ℓ outputs ($\ell < k$ and each input or output is a number) with the property that the sum of the outputs is always equal to the sum of the inputs.

It has long been known (Avizienis 1961, Dadda 1965, Karatsuba & Ofman 1963, Wallace 1964) that constant depth carry-save adders can be built and can be used to obtain logarithmic depth circuits, and therefore also polynomial size formulae, for multiple addition, multiplication and related problems. These circuits and formulae are obtained by constructing networks of *CSA* units that

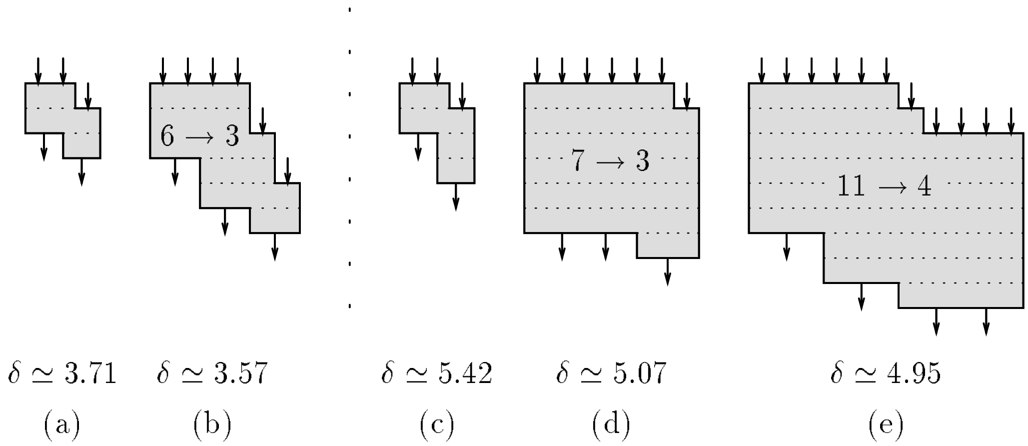

Figure 1.1: Old and new *CSA* units and their time characteristics

reduce the sum of n input numbers to the sum of only two numbers. Such networks need only have a logarithmic number of *CSA* levels. The two output numbers produced could be added, again in logarithmic depth, yielding logarithmic depth multiple addition circuits. Multiplication, in turn, is easily reduced in constant depth to a problem of multiple addition. All circuits and formulae considered in this paper are composed of dyadic (i.e., 2-input) Boolean gates. For more information on this model the reader is referred to (Boppana & Sipser 1990), (Dunne 1988), or (Wegener 1987).

Since the multiplication of two n-bit integers is such a basic task it is interesting to investigate the exact depth it requires. In particular, it would be interesting to obtain bounds on the minimal constant δ^* for which there exist multiplication circuits of depth $(\delta^* + o(1))\log_2 n$. We allow the multiplication circuits considered in this paper to output the result as a sum of two numbers. These two ($2n$-bit) numbers could be added, if required, using Carry-Look-Ahead addition circuits (Brent 1970, Khrapchenko 1970) with an additional depth of $(1 + o(1))\log_2 n$.

In Figure 1.1 we illustrate the timing properties of some of the *CSA* units referred to in this paper. For example, Figure 1.1(a) shows a $CSA_{3\to2}$ which is given two inputs at time 0 and one input at time 1 and yields outputs at times 2 and 3. The first substantial attempt to obtain exact upper bounds for δ^* was made by Khrapchenko (1978). He built a $CSA_{7\to3}$ with the time characteristics depicted in Figure 1.1(d) using which he constructed circuits over the *unate* basis, $\{\wedge, \vee, \neg\}$, of depth $5.12\log_2 n$ for multiple addition and multiplication.

Hence Khrapchenko showed that $\delta^* \leq 5.12$. The only non-trivial lower bound currently known for δ^* is $\delta^* \geq 2$ over the unate basis. This lower bound follows from a lower bound on formula size also obtained by Khrapchenko (1972a).

Most *CSA*'s (e.g., those of Figure 1.1) do not require all their inputs at the same time. Nor do they produce all their outputs at the same time. In (Paterson *et al.* 1992), ways of taking full advantage of this fact were described. Given a collection of *CSA* units, the theory of Paterson *et al.* (1992) describes the optimal way in which these units can be combined to form carry-save networks. An application of this theory to the standard $CSA_{3\to 2}$ unit of Figure 1.1(a) yields $\{\wedge, \oplus\}$-circuits for multiple addition and for multiplication of depth $3.71 \log_2 n$. An application of this theory to Khrapchenko's $CSA_{7\to 3}$ yields unate circuits of depth $5.07 \log_2 n$. Khrapchenko, as it turns out, did not use his $CSA_{7\to 3}$ unit optimally. In this work we construct the $CSA_{6\to 3}$ unit of Figure 1.1(b) using which $\{\wedge, \oplus\}$-circuits of depth $3.57 \log n$ for multiplication can be obtained. We also construct the $CSA_{11\to 4}$ of Figure 1.1(e), using which we obtain unate multiplication circuits of depth $4.95 \log_2 n$.

The stated constants are easily extracted from the schematic descriptions of these *CSA* units given in Figure 1.1. Each unit has an attached *characteristic equation*. The characteristic equation of the $CSA_{3\to 2}$ of Figure 1.1(a), for example, is $2 + x - x^2 - x^3 = 0$. The optimal circuits constructed using this unit would have asymptotic depth $(\delta + o(1)) \log_2 n$ where $\delta = 1/\log_2 \lambda$, and $\lambda \simeq 1.20557$ is the *principal root* of this characteristic equation. The principal root of a characteristic equation is its maximal real root. Characteristic equations of well-behaved units have a unique root $\lambda > 1$ which is the principal root.

What we consider to be the main result of the present work is an extension of the theory developed in (Paterson *et al.* 1992) that allows us to consider collections of *CSA* units each of which may receive inputs and produce outputs using several different representation standards. The nature of this extension is best explained by an example.

Assume that we have at our disposal the collection of $CSA_{3\to 2}$'s given in Figure 1.2. The terms s_0, s_1 and s_2 labeling the inputs and outputs of these units describe the *standard* in which each input is required and in which each output is produced. We do not specify for the time being what these different standards are. We are not allowed to connect, say, an s_2-output directly into an s_0-input (this may cause a short-circuit). We have however the means of translating between the different standards, if we so wish. In this example we assume that we can use the conversion units shown in Figure 1.3. Some of these translations are done instantaneously (such as translating an s_0 to an s_1

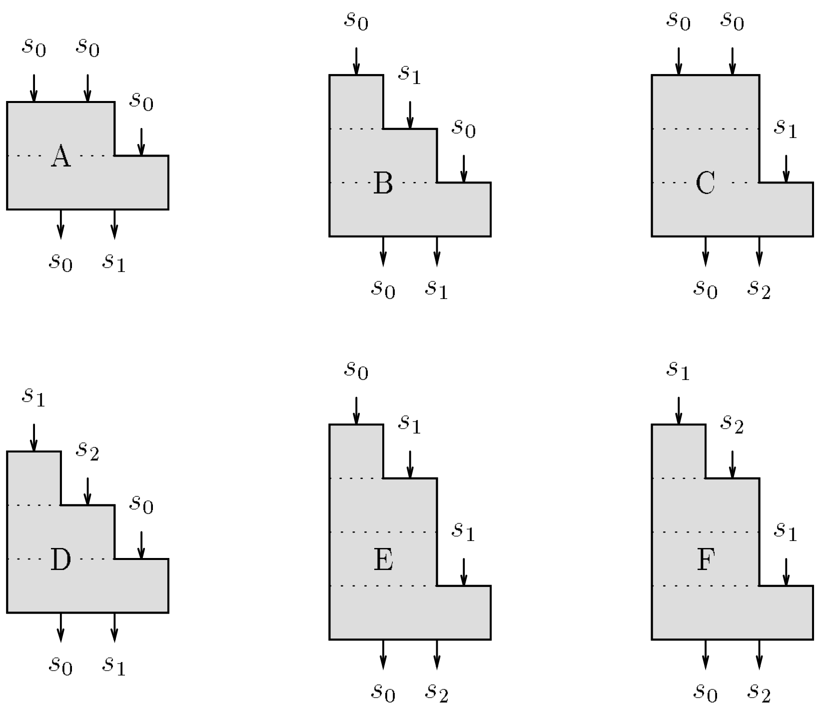

Figure 1.2: A collection of *CSA* units that use three different standards

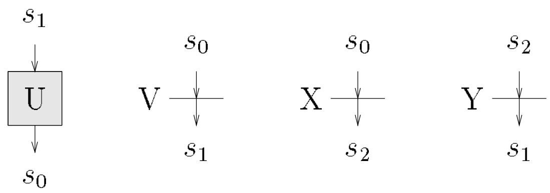

Figure 1.3: Conversion units between the three different standards

or s_2) but others take time.

We are now given n input numbers, in the s_0 standard, and we want to build a network that returns two numbers, again in the s_0 standard, whose sum is equal to the sum of the n original numbers. What is the minimal depth that such a network would require? The theory we develop here gives a general answer to questions of this type. An application of this general theory to the collection of Figure 1.2 and Figure 1.3 gives an answer of about $3.48 \log_2 n$.

It may seem at first that this consideration of multi-standard *CSA* units is a useless generalization and that the use of several standards will only hamper attempts to construct shallow networks and circuits. Surprisingly perhaps, we show that this is not the case. We have found three transmission standards s_0, s_1 and s_2 for which each of the units of Figure 1.2 and Figure 1.3 is realisable using 2-input AND, OR and XOR gates. As a consequence we get $\{\wedge, \vee, \oplus\}$-circuits for multiple addition and multiplication of depth $3.48 \log_2 n$.

The use of *CSA* units corresponds to the introduction of redundant number representation. The three representation standards we use to implement the units of Figure 1.2 and Figure 1.3 introduce redundant *bit* representations. Each bit is coded in these standards by a tuple of four bits that flow, at different times, over four different wires.

An interesting feature of the theory dealing with multi-standard *CSA* units is the resort to the Min-Max Theorem of game theory (von Neumann & Morgenstern 1944) in the proof that the lower bounds and the constructions presented do indeed match each other.

In (Paterson & Zwick 1992), where a preliminary version of some of these results appeared, an analogy between multi-standard *CSA* units and multi-currency financial plans was presented. We do not pursue this analogy any further here. Previous reports on some of the results obtained in this paper have also appeared in (Paterson *et al.* 1990) and (Paterson & Zwick 1991).

A modular gadget is a unit whose time characteristics can be described by a diagram of the form used in Figures 1.1, 1.2 and 1.3. There also exist non-modular gadgets whose time characteristics cannot be described by such a single diagram. A separate diagram should be used instead for each of the non-modular unit's outputs. It was shown in (Paterson *et al.* 1992) that the best performance of a non-modular gadget can be matched using a modular gadget obtained from the non-modular gadget by adding to it some internal delays. Thus, non-modular gadgets offer no advantages over modular gadgets if they can only be used as 'black boxes'. In Section 12 we show however that a non-modular design is sometimes an indication that a better unit, yielding shallower networks, could be designed. The $CSA_{11\to4}$ unit of Figure 1.1(e) is actually obtained by adding internal delays to a non-modular *CSA* unit (see Figure 10.6). As a consequence we get that this unit is not optimal and can be further improved. The implied improvement is minute however.

Most of this paper deals with depth. In (Paterson *et al.* 1992) an analogy between formula size and depth was presented. This analogy was exploited to describe the optimal way in which given *CSA* units can be used to obtain the shortest formulae for each output bit of a carry-save addition operation on n numbers. Using a specially designed $CSA_{6\to3}$ unit (different from the one used in the depth constructions) and the $CSA_{11\to4}$ of Figure 1.1(e), we get $\{\wedge,\oplus,\neg\}$-formulae of size $O(n^{3.13})$ and $\{\wedge,\neg\}$-formulae of size $O(n^{4.57})$ for each of these output bits. As a consequence majority formulae of the same size are obtained. This improves the results of (Paterson *et al.* 1992) and several earlier results (Khrapchenko 1972b, Pippenger 1974, Paterson 1977, Peterson 1978).

2. Collections of *CSA* units

The time characteristics of a general k-input ℓ-output carry-save adder (or $CSA_{k\to\ell}$ for short) are described by an $\ell\times k$ *delay matrix* D. The entry d_{ij} of the matrix gives the relative delay of the i-th output with respect to the j-th input. In particular, it is assumed that if the k inputs to the unit are available at times $x_1,\ldots,x_k$ then the i-th output will be ready at time $y_i=\max_{1\le j\le k}\{d_{ij}+x_j\}$.

It was shown in (Paterson *et al.* 1992) that units with general delay matrices offer no advantages over units with *modular* delay matrices. An $\ell\times k$ matrix D is modular if there exist two vectors $a\in\mathbb{R}^k$ and $b\in\mathbb{R}^\ell$ such that $D=b-a^T$, i.e., $d_{ij}=b_i-a_j$. In view of this we restrict our treatment here (with the exception of Section 12) to units with modular delay matrices. We do allow however negative entries in delay matrices, although this was not allowed in (Paterson *et al.* 1992), as we need this added generality to obtain some of our carry-save addition circuits. A unit has negative entries in its delay matrix if it can yield some of its outputs before seeing all its inputs. The $CSA_{6\to3}$ of Figure 1.1(b) is an example of such a unit. To incorporate matrices with negative entries in our treatment we introduce a *causality* assumption which is described in the next section. The causality assumption will also be needed in the treatment of multi-standard *CSA* units. Given a modular delay matrix $D=b-a^T$, we assume that all the entries of a and b are integral and that the minimal element of a is zero. We attach to a modular unit G with delay matrix $D=b-a^T$ the *characteristic polynomial* $f_G(x)=\sum_{j=1}^{k}x^{a_j}-\sum_{i=1}^{\ell}x^{b_i}$.

A *carry-save network* is an acyclic network composed of *CSA* units. Each input of a *CSA* unit used in the network is either fed by an output of another *CSA* unit or else considered to be an input of the network. Each output of a *CSA* unit used in the network is either feeding an input of another *CSA* unit or else considered to be an output of the network. An $n\to\ell$ *network* is a network with n inputs and ℓ outputs. The inputs to such a network are assumed to be available at time 0. The times at which the outputs are available are computed using the delay matrices of the *CSA*'s used in the network. The *delay of a network* is the time at which the last output of the network is obtained. Any carry-save network, like any individual *CSA* unit, has the property that the sum of its outputs is equal to the sum of its inputs.

Given a collection $\mathcal{G} = \{G_1, \ldots, G_m\}$ of *CSA* units, our task is to construct $n \to \ell$ carry-save networks (where ℓ is some fixed constant) with asymptotically minimal delay. We denote by $\delta(\mathcal{G})$ the minimal constant for which such networks of depth $(\delta(\mathcal{G}) + o(1)) \log_2 n$ could be constructed. In (Paterson *et al.* 1992) this problem was solved for collections of units in which all inputs and outputs conform to the same standard. It was shown there that if G is a modular unit with characteristic polynomial $f_G(x)$, then $\delta(G) = 1/\log_2 \lambda(G)$ where $\lambda(G) > 1$ is the principal root of the *characteristic equation* $f_G(x) = 0$ of G. If the delay matrix of G has no negative entries (or, as we shall see, if G is causal), then the characteristic equation has a unique root $\lambda(G) > 1$. It was further shown there that if $\mathcal{G} = \{G_1, \ldots, G_m\}$ is a collection of units all of which use the same standard then $\delta(\mathcal{G}) = \min\{\delta(G_i) : 1 \le i \le m\}$. In this work we solve the problem for collections that may use several different standards. As we shall see, it is no longer the case that a single *CSA* type from each collection can be used to obtain optimal constructions.

We first extend the definition of characteristic polynomials to the case in which more than one standard is present. Each input (output) of standard s_j required (produced) at time i contributes the term $+s_j x^i$ $(-s_j x^i)$ to the characteristic polynomial of its unit. It is assumed here that the earliest input to a unit is required at time 0. The symbols s_j are regarded as indeterminates. The characteristic polynomials of the units shown in Figures 1.2 and 1.3, for example, are:

$$\begin{aligned}
A &= (2 + x - x^2)\cdot s_0 - x^2\cdot s_1 \\
B &= (1 + x^2 - x^3)\cdot s_0 + (x - x^3)\cdot s_1 \\
C &= (2 - x^3)\cdot s_0 + x^2\cdot s_1 - x^3\cdot s_2 \\
D &= (x^2 - x^3)\cdot s_0 + (1 - x^3)\cdot s_1 + x\cdot s_2 \\
E &= (1 - x^4)\cdot s_0 + (x + x^3)\cdot s_1 - x^4\cdot s_2 \\
F &= -x^4\cdot s_0 + (1 + x^3)\cdot s_1 + (x - x^4)\cdot s_2
\end{aligned}$$

and

$$\begin{aligned}
U &= -x\cdot s_0 + s_1 \\
V &= s_0 - s_1 \\
X &= s_0 - s_2 \\
Y &= s_2 - s_1
\end{aligned}$$

Carry-save networks may include internal delays. An internal delay occurs when an output of a *CSA* unit that feeds an input of a second *CSA* unit

$$M_{\mathcal{G}_0}(x) = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ U \\ V \\ X \\ Y \end{array} \begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{pmatrix} 2+x-x^2 & -x^2 & 0 \\ 1+x^2-x^3 & x-x^3 & 0 \\ 2-x^3 & x^2 & -x^3 \\ x^2-x^3 & 1-x^3 & x \\ 1-x^4 & x+x^3 & -x^4 \\ -x^4 & 1+x^3 & x-x^4 \\ -x & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \end{pmatrix} \end{array} .$$

Figure 2.1: The characteristic matrix of the collection of Figures 1.2 and 1.3.

is ready before it can be consumed by the second unit or when an external input is available before it can be consumed. We may transform a network with internal delays into one without such delays by introducing explicit delay units. For simplicity we will assume that units capable of delaying an input of any given standard by one time unit are always available. Note that in the concrete example of Figures 1.2 and 1.3, delay units for every standard are easily obtained by combining two or more converters.

The important parameters of a network, i.e., the number of inputs and outputs of each standard it has in each time unit are described by the *characteristic polynomial* of the network. The characteristic polynomial of a network contains, as in the case of single *CSA* units, a $+s_jx^i$ term for every s_j-input required at time i and a $-s_jx^i$ term for every s_j-output produced at time i. It is again assumed that the earliest input is required at time 0. The same characteristic polynomial is obtained by considering the network to be a single *CSA* unit. Characteristic polynomials provide an adequate and convenient algebraic representation for our constructions.

The characteristic polynomial of a network constructed from units $G_1, \ldots,$ G_m with characteristic polynomials $g_1(x;\mathbf{s}), \ldots, g_m(x;\mathbf{s})$ (where $\mathbf{s}$ denotes the vector of standards) can be written in the form

$$N(x;\mathbf{s}) = \sum_{i=1}^{m} f_i(x) g_i(x;\mathbf{s})$$

where each $f_i(x)$ is a univariate polynomial in x with non-negative integer coefficients that specify the number of copies of unit G_i that are initiated at each time unit. The exact interconnections used are not important, as long as the s_j-outputs produced at a certain time i are connected, in a one to one manner, to the s_j-inputs required at time i. Any excess inputs required at some time become inputs of the whole network. Any excess outputs produced become outputs of the whole network.

It is sometimes convenient to arrange the characteristic polynomials of the units belonging to a certain collection in a matrix. The *characteristic matrix* $M_{\mathcal{G}}(x)$ of a collection $\mathcal{G} = \{G_1, \ldots, G_m\}$ of gadgets that use standards $s_0, \ldots,$ s_{p-1} is an $m \times p$ matrix in which the entry $m_{i,j}$ is the coefficient of the indeterminate s_j in $g_i(x; \mathbf{s})$, for $0 \le i \le p-1$, $1 \le j \le m$. The characteristic matrix of the collection $\mathcal{G}_0$ of Figures 1.2 and 1.3, for example, is given in Figure 2.1.

3. The causality assumption

To get constructions that match the lower bounds of Section 4, we need to assume that the gadgets with which we are dealing are causal.

DEFINITION 3.1. *A unit G is said to be* causal *if an output produced at time t depends only on inputs supplied at times strictly before t.*

Any gadget implemented using Boolean gates or any other conceivable technology will certainly be causal, as it cannot predict its future inputs. Any causal carry-save adder can produce a useful, non-constant, output only after receiving at least one input and it cannot receive any non-constant input after all the outputs have been produced. A further consequence of causality, that will be used in Section 5, is the following property: if all the inputs supplied to a causal unit before time t are zero, then all the outputs at or before time t are constants.

The conversion units V, X and Y of Figure 1.3 are not strictly causal as they produce their outputs instantaneously. Any combination of these units with other strictly causal units will however yield strictly causal units and the theory we develop can be easily shown to apply in such cases.

If G is a gadget, we denote by $G^{[t]}$ the gadget obtained by removing from G all the inputs before time t and all the outputs at or before time t. Note that if G is causal, then $G^{[t]}$ could be emulated by feeding zeros to all the inputs required before time t and ignoring all the outputs that are produced at or before time t (as they are all constant anyway).

4. Lower bounds for the depth of *CSA* networks

We begin our treatment of multi-standard networks by presenting a lower bound on their depth.

Theorem 4.1. *Let $\mathcal{G} = \{G_1, \ldots, G_m\}$ be a collection of multi-standard units with characteristic matrix $M_{\mathcal{G}}(x)$. We assume that $\mathcal{G}$ contains conversion units which can be used to translate any standard to any other standard by a sequence of conversions. If $\lambda > 1$, $\mathbf{v} \in (\mathbb{R}^+)^p$, $\mathbf{v} \neq \mathbf{0}$, and $M_{\mathcal{G}}(\lambda)\cdot\mathbf{v} \leq \mathbf{0}$ then any network composed of units from $\mathcal{G}$, that takes n s_0-inputs and produces ℓ s_0-outputs (and no outputs of any other standard) has a delay of at least $\log_\lambda(n/\ell)$.* ($\mathbb{R}^+$ denotes the set of *non-negative* real numbers.)

Proof. Let N be a network with characteristic polynomial

$$N(x;\mathbf{s}) = \sum_{i=1}^{m} f_i(x) g_i(x;\mathbf{s})$$

where each $f_i(x)$ has non-negative (integral) coefficients. The condition $M_{\mathcal{G}}(\lambda)\cdot\mathbf{v} \leq \mathbf{0}$ means that $g_i(\lambda;\mathbf{v}) \leq 0$ for $1 \leq i \leq m$, and so $N(\lambda;\mathbf{v}) \leq 0$. By introducing delays if necessary, we may assume that all the inputs to the network N are required at time 0. If N has n s_0-inputs at time 0 and only ℓ s_0-outputs at times $d_1 \leq \ldots \leq d_\ell$ (and no other outputs), then

$$N(x;\mathbf{s}) = (n - \sum_{i=1}^{\ell} x^{d_i})\cdot s_0 \ .$$

Since $N(\lambda;\mathbf{v}) \leq 0$ we get that $(n - \sum_{i=1}^{\ell} \lambda^{d_i})\cdot v_0 \leq 0$. The row of $M_{\mathcal{G}}(\lambda)$ corresponding to a conversion unit from s_j to s_k yields an inequality of the form $v_j - \lambda^r v_k \leq 0$, and so $v_j > 0$ implies $v_k > 0$. The assumption of translatability between any two standards ensures that $v_j > 0$ for $0 \leq i \leq p-1$. In particular $v_0 > 0$ and therefore $n - \sum_{i=1}^{\ell} \lambda^{d_i} \leq 0$, which implies that $d \geq \log_\lambda(n/\ell)$, where $d = d_\ell$ is the delay of the network. □

For $M_{\mathcal{G}_0}(x)$, the characteristic matrix corresponding to the collection of Figures 1.2 and 1.3, the minimal $\lambda > 1$ for which there exists a non-zero vector $\mathbf{v} \in (\mathbb{R}^+)^3$ that satisfies $M_{\mathcal{G}_0}(\lambda)\cdot\mathbf{v} \leq \mathbf{0}$ turns out to be the principal root of the equation $6 + 2x - 2x^2 - 3x^3 = 0$. A corresponding vector is $\mathbf{v} = (1, v_1, v_2)$ where $v_1 = -1 + \lambda^{-1} + 2\lambda^{-2}$ and $v_2 = -1 - \lambda^{-1} + \lambda^{-2} + 4\lambda^{-3}$. Thus $\lambda \simeq 1.22096$, $v_1 \simeq 1.16064$, $v_2 \simeq 1.04941$ and $\log_\lambda n \simeq 3.47201 \log_2 n$. These values are obtained by solving the three simultaneous equations: $A(\lambda;\mathbf{v}) = C(\lambda;\mathbf{v}) = D(\lambda;\mathbf{v}) = 0$.

5. Optimal constructions of *CSA* networks

We now present our constructions. In the next section we will show that the upper bounds obtained from these constructions match the lower bounds of the previous section. These constructions are therefore optimal.

THEOREM 5.1. *Let $\mathcal{G} = \{G_1, \ldots, G_m\}$ be a collection of* causal *multi-standard units with characteristic matrix $M_{\mathcal{G}}(x)$. If $\lambda > 1$, $\mathbf{u} \in (\mathbb{R}^+)^m$, $\mathbf{u} \neq \mathbf{0}$, and $\mathbf{u}^T \cdot M_{\mathcal{G}}(\lambda) \geq \mathbf{0}$, then there exist networks, composed of units from $\mathcal{G}$, with $\Omega(n)$ s_0-inputs at time 0, and $O(1)$ s_0-outputs all produced before time $\log_\lambda n + O(1)$.*

PROOF. Let us ignore at first integrality problems and consider 'networks' that may use a non-integral number of gadgets at each level. Let H be the 'gadget' obtained by taking u_i disjoint copies of G_i for $1 \leq i \leq m$. H has the characteristic polynomial

$$h(x;\mathbf{s}) = \sum_{i=1}^{m} u_i g_i(x;\mathbf{s}) = \sum_{j=0}^{p-1} h_j(x) \cdot s_j \quad .$$

Let Δ be the degree of x in $h(x;\mathbf{s})$. At time 0, the 'gadget' H takes in some inputs (possibly a fractional number of them) but produces no outputs (this is a consequence of causality). At times $1, \ldots, \Delta - 1$, the 'gadget' H may have both inputs and outputs. At time Δ, the 'gadget' H produces some outputs and does not take in any inputs (this is again a consequence of causality).

The condition $\mathbf{u}^T \cdot M_{\mathcal{G}}(\lambda) \geq \mathbf{0}$ is equivalent to the condition $h_j(\lambda) \geq 0$ for $0 \leq j \leq p-1$. We divide each $h_j(x)$ by $1 - x/\lambda$ and get that

$$h_j(x) = (1 - x/\lambda) h'_j(x) + h''_j$$

where $h''_j \geq 0$ for $0 \leq j \leq p-1$.

The networks satisfying the requirements of the theorem are obtained by placing a geometrically decreasing number of H's at the different levels of the networks. More precisely, let $d = \lceil \log_\lambda n \rceil$ and

$$P_{n,\lambda}(x) = n \cdot \sum_{k=0}^{d-1} \lambda^{-k} x^k = n \cdot \frac{1 - (x/\lambda)^d}{1 - (x/\lambda)} \quad ,$$

and consider the network N_n with the characteristic polynomial

$$N_n(x;\mathbf{s}) = P_{n,\lambda}(x) \cdot h(x;\mathbf{s}) \quad .$$

In this network $n\lambda^{-k}$ H units are initiated at time k for $0 \le k < d$. How many inputs and outputs does this network have? A simple manipulation yields

$$\begin{aligned} N_n(x;\mathbf{s}) &= n\cdot\frac{1-(x/\lambda)^d}{1-(x/\lambda)}\cdot h(x;\mathbf{s}) \;=\; n\cdot\frac{1-(x/\lambda)^d}{1-(x/\lambda)}\cdot\sum_{j=0}^{p-1} h_j(x)\cdot s_j \\ &= n\cdot(1-(x/\lambda)^d)\cdot\underbrace{\sum_{j=0}^{p-1} h'_j(x)\cdot s_j}_{h'(x;\mathbf{s})} + n\cdot\frac{1-(x/\lambda)^d}{1-(x/\lambda)}\cdot\underbrace{\sum_{j=0}^{p-1} h''_j\cdot s_j}_{h''(\mathbf{s})} \quad . \end{aligned}$$

All the coefficients of $h''(\mathbf{s})$ are non-negative and correspond therefore to inputs. As $n\lambda^{-d} \le 1$, the term $-n(x/\lambda)^d\cdot h'(x;\mathbf{s})$ corresponds to a bounded (perhaps fractional) number of inputs and outputs all within time $d+(\Delta-1)$, as $\Delta-1$ is the degree of x in $h'(x;\mathbf{s})$.

We are left with the term $n\cdot h'(x;\mathbf{s})$. As we have noticed earlier, the 'gadget' H produces some outputs, and takes no inputs, at time Δ. The coefficients of the $x^\Delta s_j$ terms in $h(x;\mathbf{s})$ are therefore all non-positive, with at least one of them negative. As a consequence, the coefficients of the $x^{\Delta-1}s_j$ terms in $h'(x;\mathbf{s})$ are all non-negative, with at least one of them strictly positive. Therefore the term $n\cdot h'(x;\mathbf{s})$ specifies cn inputs, where $c>0$ is some constant, at time $\Delta-1$.

Finally, consider the 'network' $N_n^{[\Delta-1]}$ obtained by feeding zeros to all the inputs of N_n required before time $\Delta-1$ and ignoring all outputs at or before time $\Delta-1$. This network has $\Omega(n)$ inputs at time $\Delta-1$, and perhaps some additional inputs at other times, but only $O(1)$ outputs, all within time $\log_\lambda n + O(1)$. These inputs and outputs do not necessarily conform to the s_0 standard but this can be easily resolved using converters. The networks thus obtained satisfy all our requirements, except that they usually involve a fractional number of gadgets at each level.

The solution to the integrality problem is quite simple. Consider the network

$$N_n^*(x;\mathbf{s}) = \sum_{i=1}^{m} \lceil u_i P_{n,\lambda}(x)\rceil g_i(x;\mathbf{s})$$

where we define $\lceil \sum_{k=0}^q a_k x^k \rceil = \sum_{k=1}^q \lceil a_k \rceil x^k$. The coefficients in N_n^* differ from the corresponding coefficients in N_n by at most an additive constant. Thus the network N_n^* will still have $\Omega(n)$ inputs at the start (time $\Delta-1$), but may now have $O(1)$ additional outputs at every time unit up to $\log_\lambda n + O(1)$. Since these additional outputs trickle out very slowly (only a fixed number at each time unit), it is easy to reduce their sum to the sum of a fixed number of outputs at time $\log_\lambda n + O(1)$ by adding a fixed number of carry-save adders at each level. This establishes the validity of the theorem. □

Let $\lambda \simeq 1.22096$ be the principal root of the equation $6+2x-2x^2-3x^3 = 0$. It can be verified that

$$(2-\lambda^3)\cdot A(\lambda;\mathbf{s}) + C(\lambda;\mathbf{s}) + \lambda^2\cdot D(\lambda;\mathbf{s}) = 0$$

where $2-\lambda^3 \simeq 0.17985$. This corresponds to a mixture of units A, C and D using which networks of depth $\log_\lambda n \simeq 3.47201 \log_2 n$ can be constructed. These networks make optimal use of the gadgets shown in Figures 1.2 and 1.3. Note that units B, E, F are not used at all by these networks and that the converters U, V, X, Y are needed (if at all) primarily at the initial and final stages.

6. Matching the lower and upper bounds

An easy consequence of the von Neumann/Morgenstern Min-Max Theorem of game theory (von Neumann & Morgenstern 1944) is the following:

THEOREM 6.1. *For every $m\times p$ matrix M of real numbers, there exists a vector $\mathbf{v} \in (\mathbb{R}^+)^p, \mathbf{v} \neq \mathbf{0}$, such that $M\cdot\mathbf{v} \leq \mathbf{0}$ or a vector $\mathbf{u} \in (\mathbb{R}^+)^m, \mathbf{u} \neq \mathbf{0}$, such that $\mathbf{u}^T\cdot M \geq \mathbf{0}$.*

This theorem establishes the optimality of the lower bounds and constructions presented in the two previous sections. For every $\lambda > 1$, we either get a lower bound of $\log_\lambda n - O(1)$ or an upper bound of $\log_\lambda n + O(1)$, or both. Continuity considerations show that, for the maximum λ for which we get a $\log_\lambda n + O(1)$ upper bound, we also get a corresponding $\log_\lambda n - O(1)$ lower bound.

7. Bit-Adders and Carry-Save Adders

Carry-save adders are usually built from *bit-adders*. A bit-adder is a unit with k input bits and ℓ output bits, where $\ell < k$. Each input and output bit has an associated positional *significance*. If the k input bits are denoted by $x_1, \ldots, x_k$ and their significances are $a_1, \ldots, a_k$, and if the ℓ output bits are denoted by $y_1, \ldots, y_\ell$ and their significances are $b_1, \ldots, b_\ell$ then the relation $\sum_{i=1}^{\ell} y_i 2^{b_i} = \sum_{j=1}^{k} x_j 2^{a_j}$ must hold.

The simplest bit-adder is the 3-bit full adder FA_3 that inputs three bits with significance 0 and outputs two bits with significances 0 and 1. More generally, we denote by $FA_{k_0,\ldots,k_r}$ the bit-adder with $k = \sum_{i=0}^{r} k_i$ inputs, where k_0 of them have significance 0, k_1 of them have significance 1, and so on, and ℓ outputs with significances $0, 1, \ldots, \ell-1$, where ℓ is minimal such that $\sum_{i=0}^{r} k_i 2^i \leq 2^\ell - 1$.

An implementation of an FA_3 is given in Figure 7.1(a) and its time characteristics are given in Figure 7.1(b). A 3-input 2-output carry-save adder (or $CSA_{3\to2}$ for short) is easily implemented in constant depth using an array of FA_3's as shown in Figure 7.2. A more schematic description of the same process is given in Figure 7.3(a). Note that the $CSA_{3\to2}$ obtained would also have the time characteristics shown in Figure 7.1(b).

Any bit-adder can be used to construct a carry-save adder. In Figure 7.3(b) and (c) we see, for example, how to construct a $CSA_{6\to3}$ and a $CSA_{11\to4}$ using arrays of $FA_{5,1}$ and $FA_{7,4}$ units respectively. In (b) are shown six binary numbers above the line, the sixth having a trailing zero which is not shown. The three (output) numbers below the line, again with trailing zeros, have the same sum as the six input numbers. This is achieved by giving the 5+1 input bits in each chain above the line to an $FA_{5,1}$. Its three outputs are identified with bits of the three output numbers in a way that respects their significances, as shown by the triple below each chain. The sum invariance of each full adder guarantees the sum invariance of the carry-save adder.

The shallowest networks for the carry-save addition of n numbers (of arbitrary length) that can be constructed using $CSA_{3\to2}$'s obtained from the FA_3's of Figure 7.1 have depth of about $3.71 \log_2 n$. In Section 8 we describe an efficient implementation of an $FA_{5,1}$. Using $CSA_{6\to3}$ units built using arrays of such $FA_{5,1}$'s we get networks with depth of about $3.57 \log_2 n$. In Section 11 we construct a family of FA_3 units that use bit-redundant encodings for some of their inputs and outputs. Using this family of gadgets we get our best upper bound, of about $3.48 \log_2 n$, for the depth required for the carry-save addition of n numbers.

All the units mentioned above use XOR gates. In most technologies XOR gates cannot be directly implemented and it is therefore interesting to see what depth is needed for carry-save addition if the use of XOR gates is not allowed. An implementation of an FA_3 using only unate gates is presented in Figure 7.4. Note that this implementation is not modular. The shallowest networks for carry-save addition that can be constructed using $CSA_{3\to2}$'s based on these FA_3's have depth of about $5.42 \log_2 n$. Khrapchenko (1978) gave a unate implementation of an FA_7. A description of this implementation will

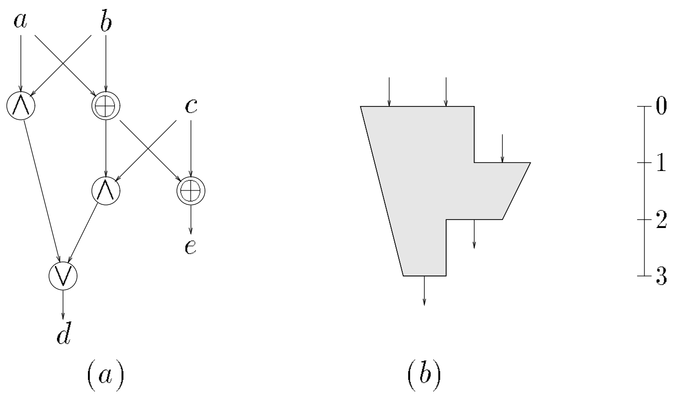

Figure 7.1: A 3-bit full-adder and the resulting $3 \rightarrow 2$ carry-save adder

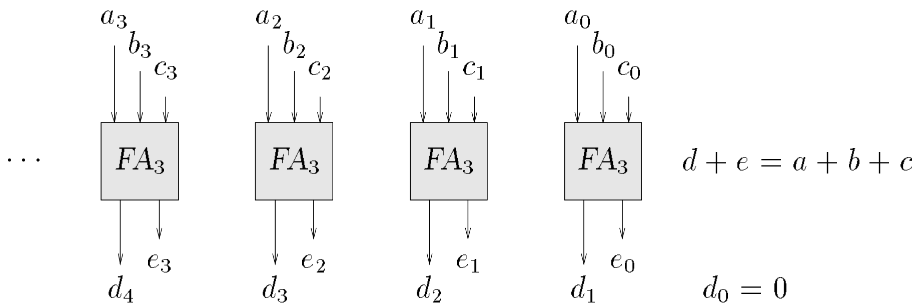

Figure 7.2: Constructing a $CSA_{3\rightarrow 2}$ using FA_3's.

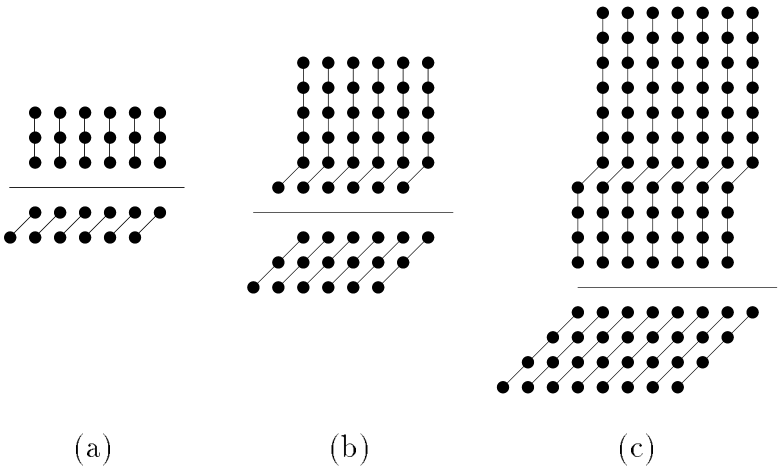

Figure 7.3: Converting bit adders to carry-save adders.

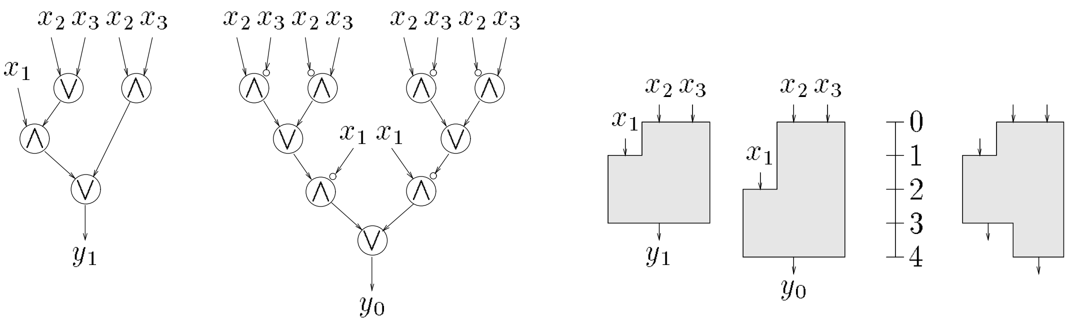

Figure 7.4: A unate implementation of a 3-bit full-adder

appear in Section 10 as our best unit is based on it. Khrapchenko showed how to use his unit to get networks with depth of about $5.12\log_2 n$. Making optimal use of Khrapchenko's FA_7 we get networks with depth of about $5.07\log_2 n$. In Section 10 we describe a design of an $FA_{7,4}$ using which, networks with depth of about $4.95\log n$ are obtained. This is currently our best bound over the unate basis.

We take the *formula size* of a formula to be the total number of occurrences of variables in it. Using the FA_3's of Figure 7.4, we can get formulae of size $O(n^{4.70})$ for each output bit in the result of a carry-save addition of n numbers. Khrapchenko (1972b) showed how to use his FA_7's to get such formulae of size $O(n^{4.62})$. Using his unit optimally, the formula size can be slightly reduced to $O(n^{4.60})$. Using our $FA_{7,4}$ an additional reduction to $O(n^{4.57})$ is obtained. It is interesting to note that in the unate case we use the *same* units to obtain our best results for both depth and formula size.

The FA_3 of Figure 7.1 yields $\{\wedge,\oplus\}$-formulae of size $O(n^{3.21})$ for each output bit of carry-save addition. (Note that the $\vee$ connective may be replaced by $\oplus$.) To get our best non-unate formulae we use however a second $CSA_{6\to3}$ unit, designed especially for this purpose, using which $\{\wedge,\oplus,\neg\}$-formulae of size $O(n^{3.13})$ can be obtained.

8. A depth efficient $6 \to 3$ Carry-Save Adder

An $\{\wedge,\oplus\}$-implementation of an $FA_{5,1}$ is given in Figure 8.1(a). The implementation uses seven *half adders (HA)* and three XOR gates. An *HA* is composed of an XOR gate and an AND gate. The left output of an *HA* with inputs a, b is $a \oplus b$ (the sum) and the right output is $a \wedge b$ (the carry).

To verify the validity of this implementation, imagine at first that the three XOR gates are replaced by *HA*'s. The connections between the *HA*'s respect the significances of the inputs and outputs (the significance associated with each wire is written next it in Figure 8.1(a)). The inputs $x_1, \ldots, x_5$ have significance 0 while x_6 has significance 1. It is easy to check that the carry output of the three *HA*'s replacing the XOR gates are identically zero so the AND gates of these *HA*'s are redundant. Removing these AND gates we get back the XOR gates we started with.

Note that the input x_5 is supplied to this unit two units of time after $x_1, \ldots, x_4$ are supplied and that y_0 is obtained one unit of time after this, even before x_6 is supplied. The outputs y_1 and y_2 are obtained one and two units of time after x_6 is supplied. This behavior is depicted in Figure 8.1(b). The $CSA_{6\to3}$ constructed using this $FA_{5,1}$ will have the same delay characteristics.

The results of the previous sections give us the optimal way of combining

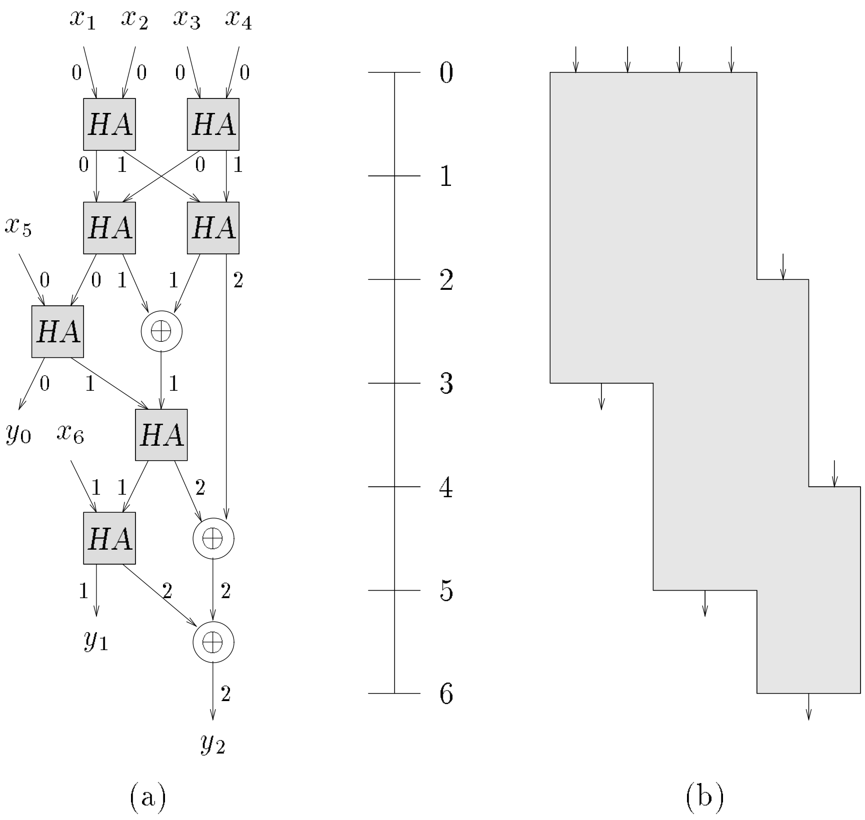

Figure 8.1: An implementation of an $FA_{5,1}$.

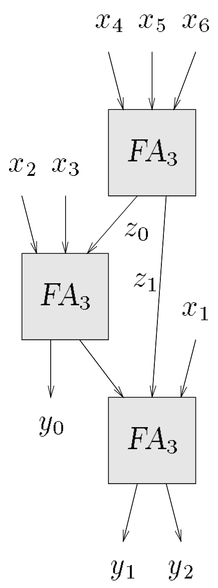

Figure 9.1: An implementation of an $FA_{5,1}$ using three FA_3's.

these $CSA_{6\to3}$'s into networks. The delay of these networks for the (carry-save) addition of n numbers will be approximately $\log_\lambda n \simeq 3.57 \log_2 n$ time units where $\lambda \simeq 1.21486$ is the principal root of the characteristic equation $4 + x^2 - x^3 + x^4 - x^5 - x^6 = 0$.

9. A formula size efficient $6 \to 3$ Carry-Save Adder

The $FA_{5,1}$ implementation presented in the previous section was designed to minimize delays. It is not very efficient however with respect to formula size. We now describe an implementation with the opposite features.

Our starting point is the implementation of an $FA_{5,1}$ using three FA_3's shown in Figure 9.1. We denote the inputs by $x_1, \ldots, x_6$ (where x_1 is this time the input with significance 1), the outputs by y_0, y_1, y_2 (with significances $0, 1, 2$ respectively) and the two intermediate results by z_0, z_1. The bits z_0 and z_1 can be expressed using the following formulae

$$\begin{aligned} z_0 &= x_4 + x_5 + x_6 \\ z_1 &= M(x_4, x_5, x_6) = (x_4 + x_6)(x_5 + x_6) + x_6 \end{aligned}$$

where M denotes the majority function. For clarity we use $+$ to denote XOR connectives. Implicit multiplications correspond to AND connectives. Composing these formulae we get

$$
\begin{aligned}
y_0 &= x_2 + x_3 + (x_4 + x_5 + x_6) \\
y_1 &= M(x_4, x_5, x_6) + M(x_4 + x_5 + x_6, x_2, x_3) + x_1 \\
y_2 &= M(\ x_1,\ M(x_4 + x_5 + x_6, x_2, x_3),\ M(x_4, x_5, x_6)\ ) \\
&= \Big(x_1 + M(x_4, x_5, x_6)\Big) \cdot \Big(M(x_4 + x_5 + x_6, x_2, x_3) + M(x_4, x_5, x_6)\Big) \\
&\quad + M(x_4, x_5, x_6) \quad .
\end{aligned}
$$

The term $M(x_4, x_5, x_6) + M(x_4 + x_5 + x_6, x_2, x_3)$ appears in the formulae for both y_1 and y_2. Expanding it we get

$$
\begin{aligned}
&M(x_4, x_5, x_6) + M(x_4 + x_5 + x_6, x_2, x_3) \\
&\quad = \Big((x_4 + x_6)(x_5 + x_6) + x_6\Big) \ + \ \Big((x_4 + x_5 + x_6 + x_3)(x_2 + x_3) + x_3\Big) \\
&\quad = (x_4 + x_6 + 1)(x_5 + x_6 + 1) \ + \ (x_4 + x_5 + x_6 + x_3)(x_2 + x_3 + 1) \ + \ 1 \quad .
\end{aligned}
$$

Two occurrences of x_3 and x_6 in the first expression were replaced in the second by four occurrences of the costless constant 1, which could be removed by using negations. Putting everything together, we get the following expanded formulae for y_1, y_2 and y_3

$$
\begin{aligned}
y_0 &= x_2 + x_3 + x_4 + x_5 + x_6 \\
y_1 &= (x_4 + x_6 + 1)(x_5 + x_6 + 1) + (x_3 + x_4 + x_5 + x_6)(x_2 + x_3 + 1) + x_1 + 1 \\
y_2 &= \Big((x_4 + x_6 + 1)(x_5 + x_6 + 1) + (x_3 + x_4 + x_5 + x_6)(x_2 + x_3 + 1) + 1\Big) \cdot \\
&\quad \Big(x_1 + (x_4 + x_6)(x_5 + x_6) + x_6\Big) \ + \ (x_4 + x_6)(x_5 + x_6) + x_6
\end{aligned}
$$

This gives us an $\{\wedge, \oplus, \neg\}$-implementation of an $FA_{5,1}$, and therefore also of a $CSA_{6\to3}$, with *occurrence matrix* $C_{6\to3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 3 \\ 1 & 1 & 2 & 4 & 4 & 9 \end{pmatrix}$. An occurrence matrix specifies the number of times each variable appears in each of the formulae defining the output bits.

It was shown in (Paterson *et al.* 1992) that the smallest carry-save addition formulae that can be constructed using a $CSA_{k\to\ell}$ unit with occurrence matrix C are of size $O(n^{\epsilon(C)})$ where $\epsilon(C) = 1/p(C)$ and $p(C)$ is the largest p for which there exists a vector $\mathbf{x} \in (\mathbb{R}^+)^k$, $\mathbf{x} \neq \mathbf{0}$, such that $\|C\mathbf{x}\|_p = \|\mathbf{x}\|_p$. It can be checked that for $C_{6\to3}$ we have $\epsilon = 1/p \simeq 3.1225317$. A corresponding vector is

$$
\mathbf{x} \simeq (1.000000, 0.327835, 0.186265, 0.128724, 0.128724, 0.059360) \quad .
$$

The components of this vector correspond to the ratios between the formula sizes of inputs that are fed to the same unit in the optimal construction. Thus by using the $CSA_{6\to3}$ just designed we can obtain carry-save addition formulae, and as a consequence also majority formulae, of size $O(n^{3.13})$.

It is interesting to note that the formula used above for y_2 is not of minimal size. It could be shortened to

$$\begin{aligned} y_2' &= \Big((x_3+x_4+x_5+x_6)(x_2+x_3+1)+(x_4+x_6+1)(x_5+x_6+1)\Big)\cdot \\ &\quad \Big(x_1+(x_4+x_6)(x_5+x_6)+x_6\Big) \quad + \quad x_1 \quad . \end{aligned}$$

Using this formula for y_2 we get however worse results: formulae of size about $n^{3.17}$. The reason is that the five occurrences of x_4, x_5, x_6 that were removed carry much less weight than the new occurrence of x_1 which is introduced.

10. An $11 \to 4$ Carry-Save Adder

The $CSA_{6\to3}$'s described in the two previous sections rely heavily on the use of XOR gates. An XOR gate can always be replaced by three AND-like gates with a total delay of two time units. Better results are obtained however by using different designs.

Khrapchenko (1978) gave a design of an FA_7 which yields a $CSA_{7\to3}$ with the characteristics given in Figure 1.1(d). A more detailed description of the time characteristics of Khrapchenko's unit, which is actually non-modular, is given in Figure 10.2. Khrapchenko describes networks based on his *CSA* unit with asymptotic delay of about $5.12\log_2 n$. These networks do not use his unit optimally. Using the results of the preceding sections or even the less general results in (Paterson *et al.* 1992), we can obtain better networks with a delay of about $5.07\log_2 n$.

In this section we give an implementation of an $FA_{7,4}$ using which a further improvement is possible. Since the design of this new unit is based on Khrapchenko's design, we give a concise summary of Khrapchenko's $CSA_{7\to3}$ unit in Figure 10.1. In Figure 10.1 (and in Figure 10.3) we use the following notation. We denote by S_k^A the symmetric function of k variables which takes the value 1 for inputs $x_1,\ldots,x_k$ if and only if $\sum x_i \in A$. For example, S_7^{4567} stands for the majority function of seven variables. For conciseness we write U_A for $S_3^A(u)$ where $u=(x_1,x_2,x_3)$, and V_A for $S_4^A(v)$ where $v=(x_4,x_5,x_6,x_7)$, and so on.

On the left of each formula appearing in Figure 10.1 we give its delay and occurrence vectors. The 'c' in the occurrence vector of y_1 stands for the number

Notation: $x = \underbrace{x_1x_2x_3}_{u}\underbrace{x_4x_5x_6x_7}_{v}$

delays	*occ'nces*			
4666666	4888888	y_0	$=$	$\mathbf{S}_7^{1357} = U_{02}V_{13} \vee U_{13}V_{024}$
5667777	688cccc	y_1	$=$	$\mathbf{S}_7^{2367} = U_{23}V_{04} \vee U_{12}V_1 \vee U_{01}V_2 \vee U_{03}V_3$
5666666	3446666	y_2	$=$	$\mathbf{S}_7^{4567} = V_4 \vee U_{123}V_{34} \vee U_{23}V_{234} \vee U_3V_{1234}$
233	122	U_{01}	$=$	$\overline{U}_{23}$
244	244	U_{02}	$=$	$\overline{U}_{13}$
233	222	U_{12}	$=$	$\overline{U}_{03}$
122	111	U_3	$=$	$x_1(x_2x_3)$
233	222	U_{03}	$=$	$\overline{x}_1(\overline{x}_2\overline{x}_3) \vee x_1(x_2x_3)$
244	244	U_{13}	$=$	$\overline{x}_1(\overline{x}_2x_3 \vee x_2\overline{x}_3) \vee x_1(\overline{x}_2\overline{x}_3 \vee x_2x_3)$
233	122	U_{23}	$=$	$x_1(x_2 \vee x_3) \vee x_2x_3$
122	111	U_{123}	$=$	$x_1 \vee (x_2 \vee x_3)$
4444	3333	V_1	$=$	$\overline{x}_4\overline{x}_5(\overline{x}_6x_7 \vee x_6\overline{x}_7) \vee (\overline{x}_4x_5 \vee x_4\overline{x}_5)\overline{x}_6\overline{x}_7$
4444	4444	V_2	$=$	$V_{234}\overline{V}_{34}$
4444	3333	V_3	$=$	$(\overline{x}_4x_5 \vee x_4\overline{x}_5)x_6x_7 \vee x_4x_5(\overline{x}_6x_7 \vee x_6\overline{x}_7)$
4444	4444	V_{13}	$=$	$\overline{V}_{024}$
2222	1111	V_4	$=$	$x_4x_5x_6x_7$
3333	2222	V_{04}	$=$	$\overline{x}_4\overline{x}_5\overline{x}_6\overline{x}_7 \vee x_4x_5x_6x_7$
3333	2222	V_{34}	$=$	$(x_4x_5 \vee x_6x_7)(x_4x_6 \vee x_5x_7)$
4444	4444	V_{024}	$=$	$(\overline{x}_4\overline{x}_5 \vee x_4x_5)(\overline{x}_6\overline{x}_7 \vee x_6x_7) \vee (\overline{x}_4x_5 \vee x_4\overline{x}_5)(\overline{x}_6x_7 \vee x_6\overline{x}_7)$
3333	2222	V_{234}	$=$	$(x_4 \vee x_5)(x_6 \vee x_7) \vee (x_4 \vee x_6)(x_5 \vee x_7)$
2222	1111	V_{1234}	$=$	$x_4 \vee x_5 \vee x_6 \vee x_7$

Figure 10.1: Khrapchenko's construction of an FA_7.

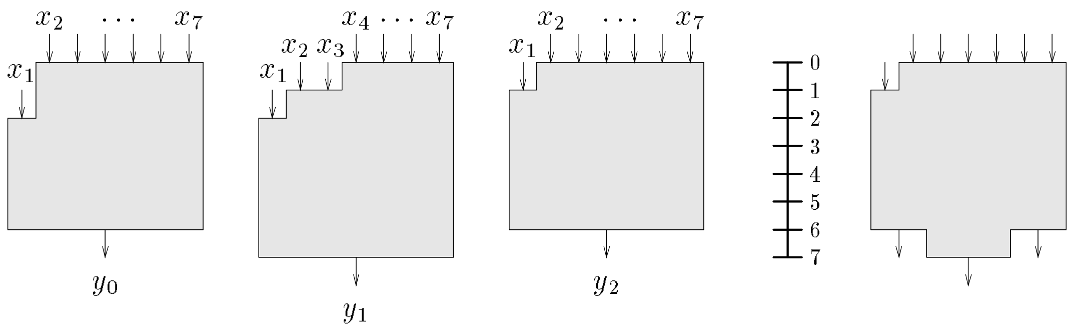

Figure 10.2: The delay characteristics of Khrapchenko's construction.

delays Notation: $x = \underbrace{\overbrace{x_1x_2x_3}^{u}\overbrace{x_4x_5x_6x_7}^{v}}_{s}\underbrace{x_8x_9x_{10}x_{11}}_{t}$

4666666	y_0	$=$	S_{1357}
78899996666	y_1	$=$	$S_{0145}T_{13} \vee S_{2367}T_{024}$
89999997777	y_2	$=$	$(S_{4567}T_{04} \vee S_{2345}T_1) \vee (S_{0123}T_2 \vee S_{0167}T_3)$
78888886666	y_3	$=$	$(S_{234567}T_{34} \vee S_{67}T_{1234}) \vee T_{234}(S_{4567} \vee T_4)$
56666664444	$S_{4567} \vee T_4$	$=$	$(U_{23}V_{234} \vee U_{123}V_{34}) \vee ((U_3V_{1234} \vee V_4) \vee T_4)$
5667777	S_{0145}	$=$	$\overline{S}_{2367}$
5666666	S_{0167}	$=$	$\overline{S}_{2345}$
5666666	S_{0123}	$=$	$\overline{S}_{4567}$
5666666	S_{2345}	$=$	$S_{234567}\overline{S}_{67}$
4555555	S_{234567}	$=$	$U_{123}V_{1234} \vee (U_{23} \vee V_{234})$
4555555	S_{67}	$=$	$U_{23}V_4 \vee U_3V_{34}$

Figure 10.3: The new $FA_{7,4}$ construction.

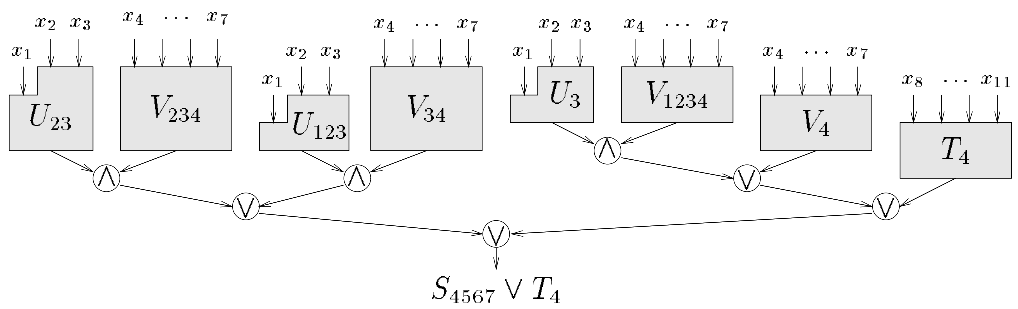

Figure 10.4: The final stages in the computation of $S_{4567} \vee T_4$.

12. Note that negations are not considered to cause any delays. If we use the de Morgan rules, all the negations can be pushed to the lowest levels of the circuits adding a total delay of at most one time unit. The delay characteristics of the three output bits y_0, y_1, y_2 of Khrapchenko's FA_7 are given schematically in Figure 10.2. It is shown in (Paterson *et al.* 1992) that the optimal way of combining these three units into a single modular gadget is the way presented on the right in Figure 10.2.

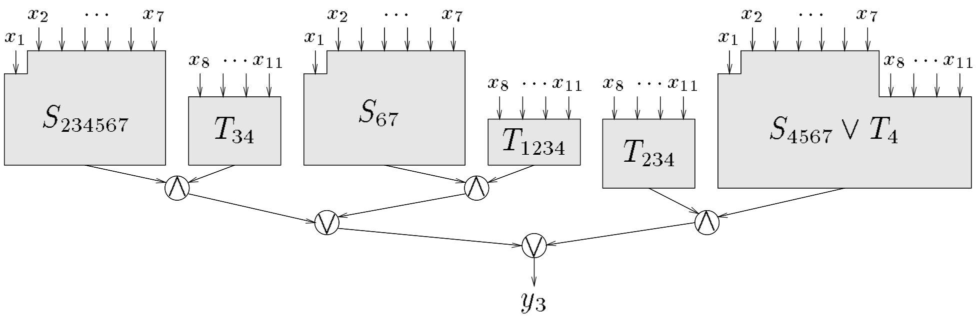

Figure 10.5: The final stages in the computation of y_3.

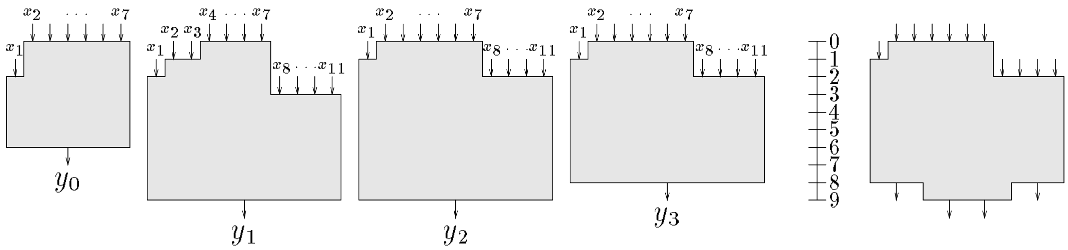

Figure 10.6: The delay characteristics of the new $FA_{7,4}$.

The construction of our new $FA_{7,4}$ is given in Figure 10.3. Figures 10.4 and 10.5 depict the final stages in the construction of y_3. The delay characteristics of y_0, y_1, y_2, y_3 are shown in Figure 10.6 and we see that they fit into the modular unit shown on the right in that figure. It can be checked using the methods of (Paterson *et al.* 1992) that this is the optimal 'packing' of these units. By the results of Section 5, we can combine the new $CSA_{11\to 4}$'s into networks with an asymptotic depth of about $\log_\lambda n \simeq 4.95 \log_2 n$, where $\lambda \simeq 1.15041$ is the principal root of the characteristic equation $6 + x + 4x^2 - x^6 - x^8 - 2x^9 = 0$.

It can be checked that the occurrence matrix of the new $FA_{7,4}$ unit is

$$C_{11\to 4} = \begin{pmatrix} 4 & 8 & 8 & 8 & 8 & 8 & 8 & 0 & 0 & 0 & 0 \\ 12 & 16 & 16 & 24 & 24 & 24 & 24 & 8 & 8 & 8 & 8 \\ 14 & 20 & 20 & 24 & 24 & 24 & 24 & 12 & 12 & 12 & 12 \\ 7 & 10 & 10 & 12 & 12 & 12 & 12 & 6 & 6 & 6 & 6 \end{pmatrix} \quad .$$

It can be further checked that $\epsilon(C_{11\to 4}) \simeq 4.56255$, and carry-save addition $\{\wedge, \neg\}$-formulae of size $O(n^{4.57})$ can therefore be constructed using this unit, since $\vee$-connectives can be removed using the de Morgan rules.

11. Bit-redundant $3 \to 2$ Carry-Save Adders

Consider the standard implementation of an FA_3 given in Figure 7.1(a). A delay is introduced on the left input wire to the OR gate computing the carry output d. This seems to suggest that the design could be improved, but how?

The answer is simple. We do not have to insist on producing the output d as a single entity. We can simply remove the bottom OR gate and get two output bits $d_1^\vee, d_2^\vee$ such that $d = d_1^\vee \vee d_2^\vee = d_1^\vee \oplus d_2^\vee$. (Note that $d_1^\vee = a \wedge b$ and $d_2^\vee = (a \oplus b) \wedge c$ are never 1 simultaneously.) If we are then required to OR (or XOR) d with another bit f supplied at the time $d_1^\vee$ is produced, we can reassociate $(d_1^\vee \vee d_2^\vee) \vee f$ as $(d_1^\vee \vee f) \vee d_2^\vee$, thereby gaining one time unit. It is the accumulation of these time units that will enable us to reduce the delay of carry-save addition and multiplication circuits.

But what will we do when required to AND d with another bit? It seems that we would then have to OR the two fractions $d_1^\vee, d_2^\vee$, losing the time unit we tried to gain. To overcome this problem we note that we can easily get, within the same time limits used to obtain the pair $d_1^\vee, d_2^\vee$, another pair $d_1^\wedge, d_2^\wedge$ such that $d = d_1^\wedge \wedge d_2^\wedge$. To get this pair we simply use the following alternative majority formula $d = (a \vee b) \wedge ((a \wedge b) \vee c)$. We may therefore take $d_1^\wedge = a \vee b$ and $d_2^\wedge = (a \wedge b) \vee c$.

A description of the unit we get, which we call unit A, is given in Figure 11.1. Note that the output bit d is coded over four wires $(d_1^\vee, d_1^\wedge, d_2^\vee, d_2^\wedge)$ and that the information on the wires $d_2^\vee, d_2^\wedge$ flows one unit of time after the information on the wires $d_1^\vee, d_1^\wedge$.

We consider normal input and output bits to conform to standard s_0, and split bits, like d in the previous paragraphs, to conform to standard s_1. We consider a split bit $(d_1^\vee, d_1^\wedge, d_2^\vee, d_2^\wedge)$ to be produced (or supplied) at time t if $d_1^\vee, d_1^\wedge$ are produced (or supplied) at time $t-1$ and $d_2^\vee, d_2^\wedge$ are produced (or supplied) at time t. We will also encounter split bits of standard s_2 in which the time difference between the availability of the two pairs $d_1^\vee, d_1^\wedge$ and $d_2^\vee, d_2^\wedge$ is two time units. With these conventions we see that the time characteristics of unit A of Figure 11.1 do correspond to those of unit A of Figure 1.2.

The output bits $d_2^\vee = (a \oplus b) \wedge c$, $d_2^\wedge = (a \wedge b) \vee c$ and $e = (a \oplus b) \oplus c$ were all obtained using formulae with the generic form $(a \circ b) \circ c$. Other generic forms that may be used to design split-bit FA_3 units with no internal delay are given in Table 11.1. To get concrete formulae for $d_2^\vee, d_2^\wedge$ and e from these generic forms, the generic connective $\circ$ should be replaced by an appropriate sequence of $\vee, \wedge$ and $\oplus$ connectives. To get the concrete formulae for e for example, all

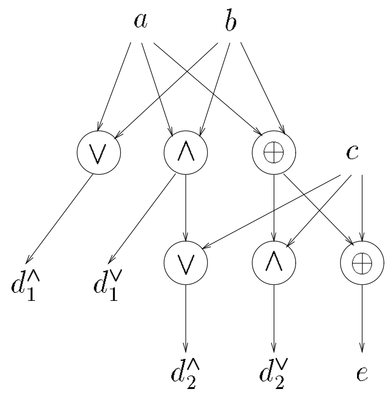

Figure 11.1: An implementation of unit A

the connectives should be taken to be $\oplus$ (and $a_1, a_2, b_1, b_2, c_1, c_2$ should be taken to be $a_1^\vee, a_2^\vee, b_1^\vee, b_2^\vee, c_1^\vee, c_2^\vee$). The generic expressions for the fractions $d_1^\vee, d_1^\wedge$ are obtained from those of Table 11.1 by removing the outer level (or two levels) of nesting involving c (or c_1 and c_2).

Detailed constructions of units C and D are given in Figure 11.2. The construction of units B, E and F goes along the same lines and requires no new ideas. Although these last three units are not used in our best solution to date, they are included here for completeness. The set $\{A, \ldots, F\}$ seems to form a natural collection of simple units which may have potential value.

The conversion units U, V, X, Y are also easily implemented. Note that we may convert an s_0-bit a into a split bit $b = (b_1^\vee, b_1^\wedge, b_2^\vee, b_2^\wedge)$ in standard s_1 or s_2 instantaneously by taking $b_2^\vee = b_2^\wedge = a$ and by letting $b_1^\vee = 0$ and $b_1^\wedge = 1$. This takes care of units V and X. Units U and Y are simple to design.

We have therefore shown that the collection of multi-standard units of Fig-

A	:	$(a \circ b) \circ c$
B	:	$((a \circ b_1) \circ b_2) \circ c$
C	:	$((a \circ b) \circ c_1) \circ c_2$
D	:	$(((a_1 \circ b_1) \circ a_2) \circ b_2) \circ c$
E	:	$(((a \circ b_1) \circ b_2) \circ c_1) \circ c_2$
F	:	$((((a_1 \circ b_1) \circ a_2) \circ b_2) \circ c_1) \circ c_2$

Table 11.1: Generic forms of formulae used to compute the outputs $d_2^\vee, d_2^\wedge, e$.

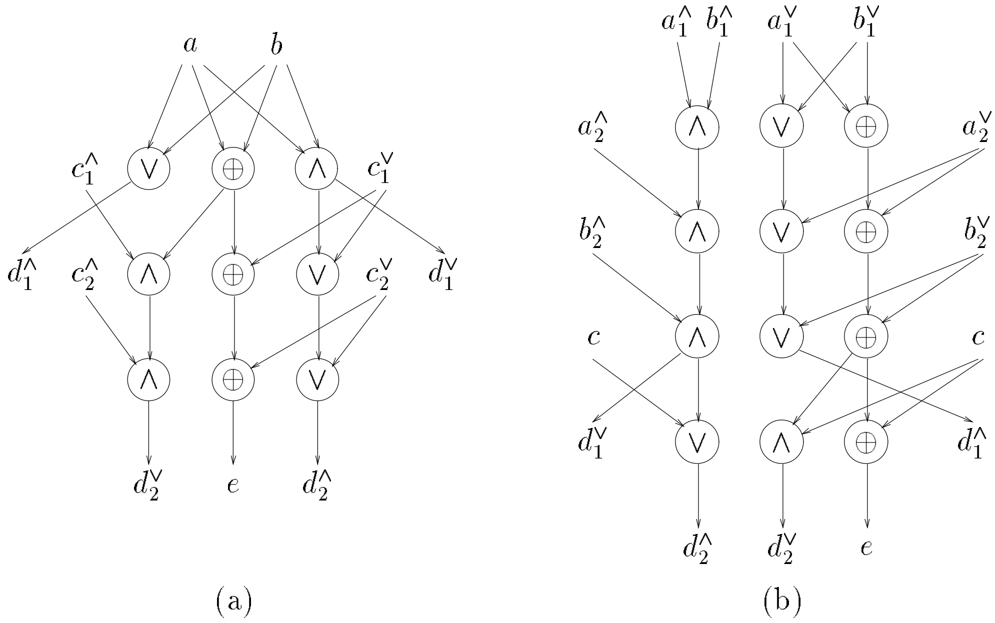

Figure 11.2: Implementations of units C and D

ures 1.2 and 1.3 can be implemented over the basis $\{\wedge, \vee, \oplus\}$ with the indicated time characteristics. Theorem 5.1 gives a depth of about $3.48\log_2 n$ for the resulting carry-save addition circuits. Theorem 4.1 shows that this is indeed the best that can we can obtain from this collection.

12. Trees of bit-adders

Consider the unate FA_3 unit constructed in Figure 7.4. It has the non-modular delay matrix $\left(\begin{smallmatrix} 2 & 4 & 4 \\ 2 & 3 & 3 \end{smallmatrix}\right)$. So far we had to consider this unit as having the characteristic polynomial $2 + x - x^3 - x^4$. The principal root of this polynomial is $\lambda \simeq 1.136528$, and the networks obtained are of depth $5.42\log_2 n$.

This did not take into account the fact that for the computation of the parity bit, it would still be acceptable to delay the third input by one additional time unit. Is there any way of making use of that fact? If we only use such unate FA_3 units to construct $CSA_{3\to 2}$'s which are then regarded as 'black boxes' then the answer is 'no'. We will see however that we do have more freedom.

Consider the tree of unate FA_3 units given in Figure 12.1. This tree, considered now as a single unit, has the characteristic polynomial

$$(4 + x + 4x^2 + 4x^3 + x^4 + 2x^6 + x^7 + x^{10})(2 + x - x^3 - x^4)$$

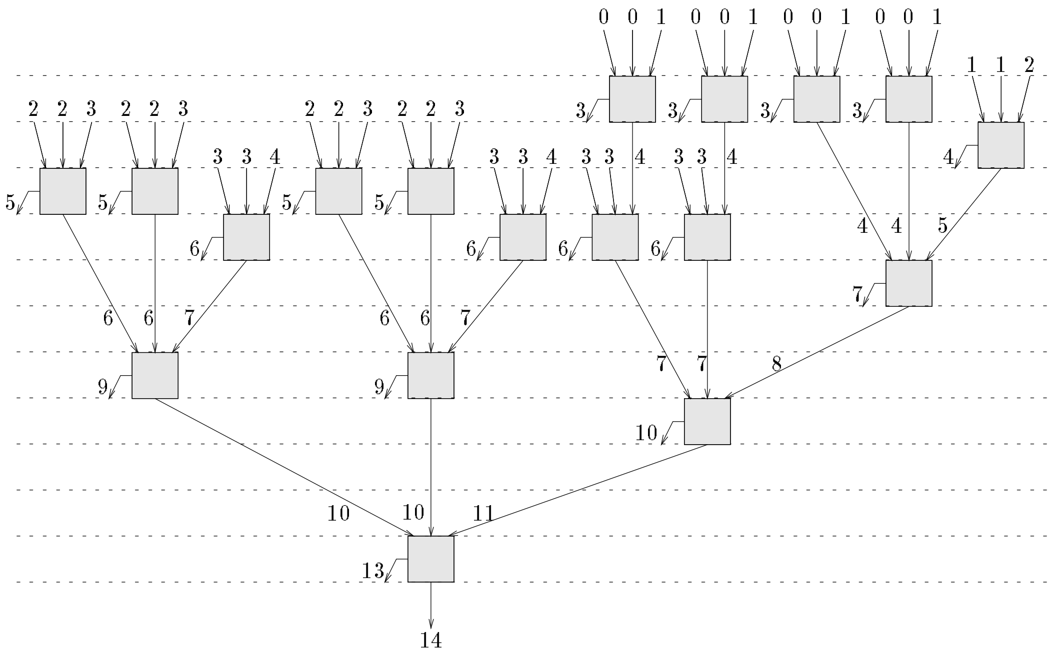

Figure 12.1: A tree of U_2-FA_3 units.

$$= \quad 8 + 6x + 9x^2 + 8x^3 + x^4 - 4x^5 - 4x^6 - x^7 - 2x^9 - x^{10} - x^{13} - x^{14} \quad .$$

The principal root of this characteristic polynomial is equal to the principal root of the characteristic polynomial $2 + x - x^3 - x^4$ for a single unate FA_3 unit. Therefore we can still get networks of depth $5.42 \log_2 n$, even if we only use the unate FA_3 units combined into such tree structures.

However, the final output of such trees is just the parity of the 37 input bits. (The sum of the positive coefficients in the above characteristic polynomial adds up to only 32 because five of the input terms are cancelled by corresponding outputs.) It is easily verified that there is a unate formula for the parity of these 37 bits, each supplied at its prescribed time, which yields its output at time $t = 13$, i.e., one unit of time before the time it is obtained in the tree. This means that we can implement a unit with characteristic polynomial

$$8 + 6x + 9x^2 + 8x^3 + x^4 - 4x^5 - 4x^6 - x^7 - 2x^9 - x^{10} - 2x^{13} \quad .$$

The principal root of this polynomial is $\lambda \simeq 1.139576$ and the networks obtained have depth $5.31 \log_2 n$, a significant improvement on the performance of a single FA_3 unit.

The same process could of course be carried out for the $FA_{7,4}$ constructed in Section 10, as delays are also introduced there in the computation of the parity bit. The improvement obtained in this way would be very small however.

13. Concluding remarks

The multipliers of many present day CPU units already follow Wallace's suggestion (Wallace 1964) and use networks of carry-save adders. Although our results are presented in the idealized Boolean circuit model, we believe that some of the ideas used here may have practical value in the future.

We have presented a general construction and some specific designs which yield circuits for carry-save addition which are faster than those previously published. Although we have only given asymptotic results here, the same methods provide efficient networks for small numbers of inputs. Further, there is a polynomial time algorithm which, for any CSA unit G and any n, gives an optimal-depth network of G's for the carry-save addition of n numbers.

From the theoretical point of view many questions remain unanswered. Is the depth of multiplication about equal to the depth of multiple addition? [1] Is there a finite collection of gadgets using which optimal-depth circuits for multiple addition can be obtained?

How far from optimal are our circuits? The best lower bound currently known for the depth of multiplication over the full binary basis is $\log_2 n + \Omega(\log\log n)$ (Fischer *et al.* 1982). Over the unate basis, a $2\log_2 n$ lower bound on the depth of multiplication follows from Khrapchenko's lower bound for formula size (Khrapchenko 1972a). Obtaining better lower bounds for multiplication is a challenging problem.

Our constants will no doubt be improved before long [2], but the techniques provide a simple construction method which may be of more durable value.

Acknowledgements

The first author was partially supported by the ESPRIT II BRA Programme of the EC under contracts # 3075 (ALCOM) and # 7141 (ALCOM II). A part of

[1] Eddie Grove (1993) has recently shown that the depths of (carry-save) multiplication, multiple addition and counting differ by only $o(\log n)$.

[2] Eddie Grove (1993) has also improved our constants. By modifying Khrapchenko's FA_7, he produced unate multiplication circuits of depth $4.93\log_2 n$. He also constructed a family of $CSA_{5\to3}$'s using which he obtains $\{\wedge, \vee, \oplus\}$ multiplication circuits of depth $3.44\log_2 n$.

this work was carried out while he was visiting Tel Aviv University. The second author was partially supported by a grant from THE BASIC RESEARCH FOUNDATION administrated by THE ISRAEL ACADEMY OF SCIENCES AND HUMANITIES. A part of this work was carried out while he was visiting the University of Warwick.

References

A. Avizienis, Signed-digit number representation for fast parallel arithmetic, *IEEE Trans. Elect. Comput.* **EC-10** (1961), 389-400.

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

R. Boppana, M. Sipser, The complexity of finite functions, in *Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity*, J. van Leeuwen, ed., Elsevier/MIT Press, 1990, 757-804.

L. Dadda, Some schemes for parallel multipliers, *Alta Frequenza* **34** (1965), 343-356.

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

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

E. Grove, *Proofs with potential*, Ph.D. thesis, U.C. Berkeley, May 1993.

A. Karatsuba, Y. Ofman, Multiplication of multidigit numbers on automata, *Soviet Phys. Dokl.* **7** (1963), 595-596.

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

V. M. Khrapchenko, A method of determining lower bounds for the complexity of π-schemes, *Mat. Zametki* **10** (1972a), 83-92 (in Russian). *Math. Notes Acad. Sciences USSR* **10** (1972), 474-479 (English translation).

V. M. Khrapchenko, The complexity of the realization of symmetrical functions by formulae, *Mat. Zametki* **11** (1972b), 109-120 (in Russian). English translation in *Math. Notes of the Acad. of Sci. of the USSR* **11** (1972), 70-76.

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

J. von Neumann, O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton Univ. Press, 1944.

M. S. Paterson, New bounds on formula size, *Proc. 3rd GI Conf. Theoret. Comput. Sci. 1977, Lecture Notes in Computer Science* **48**, Springer-Verlag 1977, 17-26.

M. S. Paterson, N. Pippenger, and U. Zwick, Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions, *Proc.*

31st Ann. IEEE Symp. Found. Comput. Sci., 1990, 642-650.

M. S. Paterson, N. Pippenger, and U. Zwick, Optimal carry save networks, in *Boolean function complexity*, M. S. Paterson, ed., LMS Lecture Note Series **169**, Cambridge University Press, 1992, 174-201.

M. S. Paterson, U. Zwick, Shallow multiplication circuits, *Proc. ARITH-10, (10th Ann. IEEE Symp. Computer Arithmetic)*, 1991, 28-34.

M. S. Paterson, U. Zwick, Shallow multiplication circuits and wise financial investments, *Proc. 24th Ann. ACM Symp. Theor. Comput.*, 1992, 429-437.

G. L. Peterson, *An upper bound on the size of formulae for symmetric Boolean functions*, TR 78-03-01, University of Washington, 1978.

N. Pippenger, *Short formulae for symmetric functions*, IBM report RC 5143, Yorktown Heights, N.Y., 1974.

C. S. Wallace, A suggestion for a fast multiplier, *IEEE Trans. Elect. Comput.* EC-13 (1964), 14-17.

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

Manuscript received September 17, 1992

Michael Paterson
Department of Computer Science
University of Warwick
Coventry, CV4 7AL, England
msp@dcs.warwick.ac.uk

Uri Zwick
Department of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
zwick@math.tau.ac.il