Shallow Multiplication Circuits [*]

Michael S. Paterson

Department of Computer Science
University of Warwick
Coventry, CV4 7AL, England

Uri Zwick

Mathematics Institute
University of Warwick
Coventry, CV4 7AL, England

Abstract

Ofman, Wallace and others used carry save adders to design multiplication circuits whose total delay is proportional to the logarithm of the length of the two numbers multiplied. An extension of their work is presented here.

The first part presents a general theory describing the optimal way in which given carry save adders can be combined into carry save networks.

In the second part, two new designs of basic carry save adders are described. Using these building blocks and the above general theory, the shallowest known theoretical circuits for multiplication are obtained.

1 Introduction

We examine theoretical ways to speed up multiplication circuits. The general approach used is the one suggested by Ofman [10] and Wallace [13]. The present paper extends previous results reported in [11],[12].

The model we use is that of *dyadic Boolean circuits.* A dyadic Boolean circuit is an acyclic circuit composed of dyadic (i.e., two-input) Boolean gates.

There are ten non-trivial types of dyadic gates: eight of them of the form $(x^a \wedge y^b)^c$ (where $x^0 = x$ and $x^1 = \bar{x}$ and $\wedge$ denotes the AND operation), and the other two of the form $(x \oplus y)^c$, where $\oplus$ denotes the XOR operation. The first eight types include the usual AND, OR, NAND and NOR gates.

Two different cases may be considered. In the first we assume that all gate types can be used and in the second that only the eight AND-like types can be used. Our method can also be used to construct fast multiplication circuits in cases where fewer gate types, e.g., only NAND gates, are allowed.

We allow the gates to be connected in an arbitrary acyclic manner and assume that each has unit delay, i.e., the output of a gate is stabilised one unit of time after its two inputs are stabilised. The total delay of a circuit is the time from the moment at which all the inputs are stable until all the outputs are stable. In this model the total delay corresponds to the length of the longest directed path from an input to an output.

This model ignores many practical considerations. No attention is paid for example to possible VLSI layouts of these circuits. Simplifying assumptions are made: that no delays are introduced on connecting wires and that the delay of a gate is not influenced by its surroundings. The model enables however a theoretical investigation of the inherent delay needed to perform multiplication. Subsequent work may reveal ways of making the constructions described in this work more practical. The basic ideas of Ofman and Wallace, on which this work is based, are already of practical use (see e.g. [1]).

The above Boolean circuit model is one of the principal models used in the theory of computational complexity. For a summary of the extensive literature available on this subject the reader is referred to [4],[6],[14].

The first step in the Ofman-Wallace approach is to design a *carry save adder (CSA).* The simplest *CSA* receives three input numbers and avoids carry propagation by outputting the sum of them as the sum of two numbers. Such a device will be called a $CSA_{3\to2}$. The striking discovery of Ofman, Wallace and others (see also [2],[5]) was that *CSA*'s can have constant delay, independent of the size of the input numbers. The second step is to construction a network of *CSA*'s that reduces the sum of n input numbers to the sum of only two. Such a network requires only a logarithmic number of layers and its total delay is therefore logarithmic. The two remaining numbers may be added using a *carry look ahead* adder (see [3],[8]) which also has logarithmic delay.

Since multiplication of two n-bit numbers involves little more than adding n numbers, the above approach yields logarithmic depth multiplication circuits.

In sections 4 and 5 of this work we present improved designs of *CSA*'s. In section 3 we describe a general method of combining *CSA*'s into shallow networks.

2 Bit Adders and Carry Save Adders

The simplest $CSA_{3\to2}$ is obtained by using an array of FA_3's (3-bit full adders) as shown in Fig. 2.1. In fact any *bit adder (BA)* could be used to construct a *CSA.*

A bit adder is a unit with k input bits and ℓ output bits, where $\ell < k$. Each input and output bit has an as-

*This work was partially supported by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM).

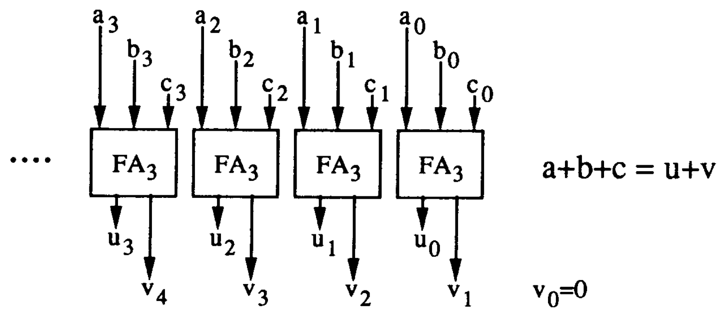

Figure 2.1: Constructing a $CSA_{3\to2}$ using FA_3's.

sociated significance. If the k input bits are denoted by $x_1,\dots,x_k$ and their significances are $a_1,\dots,a_k$, and if the ℓ output bits are denoted by $y_1,\dots,y_\ell$ and their significances are $b_1,\dots,b_\ell$ then the relation $\sum_{i=1}^{\ell} y_i 2^{b_i} = \sum_{j=1}^{k} x_j 2^{a_j}$ holds.

The simplest bit adder is the 3-bit full adder FA_3. The significance of the three input bits is 0 and the significances of the two outputs bits are 0 and 1 respectively. More generally, if $k = 2^m - 1$ then an FA_k which receives k input bits, and outputs m bits containing the binary representation of their sum could be constructed. The significance of all the inputs to an FA_k is again 0, and the significances of the m outputs are $0, 1, \dots, m-1$.

The fastest *CSA* networks which can be constructed using $CSA_{3\to2}$'s have depths asymptotic to $3.71\log_2 n$ or $5.42\log_2 n$, depending on whether all dyadic gates or only the AND-like gates are used (see [11],[12]). In order to get our best-performing *CSA*'s we need to consider slightly more general *BA*'s. If $\sum_{i=0}^{r} c_i 2^i = 2^m - 1$ we denote by $FA_{c_0,\dots,c_r}$ the bit adder with $k = \sum_{i=0}^{r} c_i$ inputs, where c_0 of them have significance 0, c_1 of them have significance 1, and so on. The unit $FA_{c_0,\dots,c_r}$ will have m outputs with significances $0, 1, \dots, m-1$. Since every number $0 \le x < 2^m$ has a unique binary representation of length m, the output of this *BA* for any input vector is uniquely defined.

In section 4 we describe an efficient implementation of an $FA_{5,1}$. A $CSA_{6\to3}$ could be built using this $FA_{5,1}$ as illustrated in Fig. 2.2. The *CSA*'s constructed in this way give rise to the shallowest known multiplication circuits. These constructions use both AND-like and exclusive-or (XOR) gates. Their asymptotic delay is about $3.57\log_2 n$ time units (excluding the time needed for the final addition).

In section 5 we describe an efficient implementation of an $FA_{7,4}$ using only AND-like gates. A $CSA_{11\to4}$ could be built using this $FA_{7,4}$ in a similar way to that shown in Fig. 2.2. The *CSA*'s constructed in this way yield the shallowest known multiplication circuits that use only AND-like gates. Their asymptotic delay is about $4.95\log_2 n$ time units (excluding again the time needed for the final addition).

3 Constructing *CSA* networks

We are given a $CSA_{k\to\ell}$ unit G that accepts its k inputs at times $x_1,\dots,x_k$ and delivers its ℓ outputs at times $y_1,\dots,y_\ell$, where $k > \ell$. Our task is to compose copies of G into a network that reduces the sum of n numbers to the sum of only ℓ. We would like the delay of this network to be as small as possible.

Without loss of generality we may assume that $0 = x_1 \le \dots \le x_k$ and $y_1 \le \dots \le y_\ell$. In a real 'causal' device no useful output can be given until after the first input is received, and no input can be relevant unless it precedes the final output. Hence we may assume that $y_1 > x_1$ and $y_\ell > x_k$.

The *characteristic polynomial of G* is defined to be $g(z) = \sum_{j=1}^{\ell} z^{y_j} - \sum_{i=1}^{k} z^{x_i}$. It is easy to see that $g(1) = \ell - k < 0$ and $g(\infty) = \infty$. Hence the equation $g(z) = 0$ has at least one real root greater than 1. We call the smallest such root the *principal root of G* and denote it by λ_G. The asymptotic depth of the networks we construct depends on the principal root: the larger this root, the shallower the circuit.

As a consequence of the causality, if all the inputs up to time t are zero, then all the outputs up to time t are zero as well. If the number of inputs still to be given after time t is larger than the number of outputs still to be produced, then we can get a *CSA* unit $G^{[t]}$ by fixing all the inputs to G up to time t to zero and ignoring the outputs up to that time. If for some $t > 0$ a unit $G^{[t]}$ can be obtained for which $\lambda_{G^{[t]}} \ge \lambda_G$ we say that G could be *improved*, since we could use the smaller unit $G^{[t]}$ instead of G to construct circuits which are asymptotically at least as shallow. If G cannot be improved in this way then we say that it is *reduced*.

For a polynomial $f(z) = \sum_{i=0}^{m} f_i z^i$ we define $f \succ 0$ (respectively $f \succeq 0$) if $f_i > 0$ (respectively $f_i \ge 0$) for $0 \le i \le m$ and also write, for example, $f \preceq g$ if $(g - f) \succeq 0$.

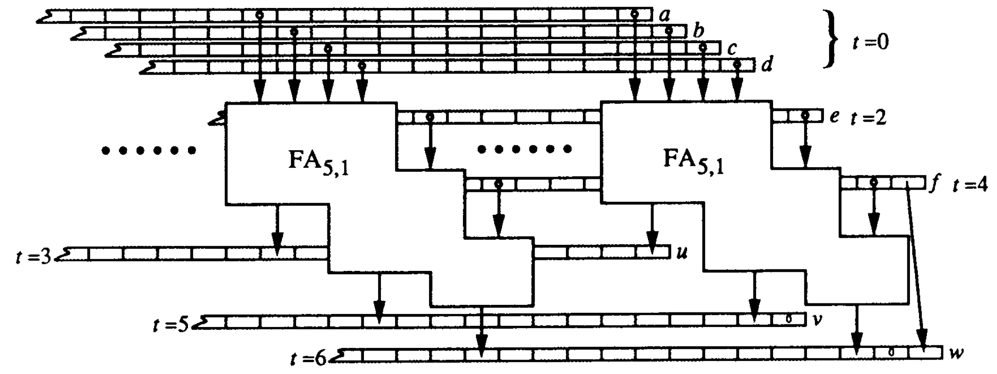

Figure 2.2: Constructing a $CSA_{6\to 3}$ using $FA_{5,1}$'s.

Lemma 3.1 *If G is reduced and $g(z) = (z - \lambda_G)h(z)$ then $h \succ 0$.*

Proof : For a polynomial $f(z) = \sum_{i=0}^{m} f_i z^i$ we define $f^{[t]}(z) = \sum_{i=t}^{m} f_i z^i$. Note that if $G^{[t]}$ is a functional *CSA* then its characteristic polynomial is $f^{[t]}(z)$.

We have $h_{m-1} = g_m > 0$ where $m = \deg(g)$. Suppose that for some t, $0 \le t < m-1$, we have $h^{[t+1]} \succ 0$ but $h_t \le 0$. Then

$$g^{[t+1]}(z) = h_t z^{t+1} + (z - \lambda_G)h^{[t+1]}(z) < 0$$

for $1 \le z < \lambda_G$. Hence $\lambda_{G^{[t+1]}} \ge \lambda_G$, which contradicts the assumption that G is reduced. □

We can interpret the equation $\lambda_G h(z) + g(z) = z h(z)$ as an assertion that if $\lambda_G h_i$ data items are available at time i, for $0 \le i < m$, and (some of them) are input to a copy of the unit G, then the result is that h_i items are available at time $i+1$ for $0 \le i < m$. This suggests that we may be able to reduce the number of items by a factor of λ_G at every time unit. The delay of a network for the carry save addition of n numbers in this case would be about $\log_{\lambda_G} n$ time units.

There is however a small obstacle to be overcome. The numbers $\lambda_G h_i$ and h_i are in general non-integral. We will use integers to approximate the real numbers that we encounter and show that this does not affect the validity of our results.

A copy of G that receives its inputs at times $t+x_1, \ldots, t+x_k$ (and yields its outputs at times $t+y_1, \ldots, t+y_\ell$) is said to be *based* at time t. The essentials of a network could be described by specifying the number of *CSA* units that are based at each given time. The outputs produced at time t can be supplied as inputs at time t in an arbitrary manner. If more items are consumed at time t than produced then the network may receive some external inputs at time t. If more items are produced than consumed, then the network produces some external outputs at time t.

For every $N > 0$ we construct a *CSA*-network in which $c_t = \lceil A/\lambda^t + b \rceil$ copies of G are based at time t for $0 \le t \le T$ where $\lambda = \lambda_G$, $A = N/(\lambda h_0)$, $b = 1/(\lambda - 1)$ and $T = \lceil \log_\lambda A \rceil \le \lceil \log_\lambda N \rceil$. The coefficient of z^t in the *characteristic function*, $\left(\sum_{t=0}^{T} c_t z^t\right) g(z)$, of the network, gives the number of inputs or outputs required or supplied at time t. A negative number denotes inputs while a positive number denotes outputs.

We claim that this network accepts at least N inputs at or after time 0, and produces only a constant number of outputs, no later than time $T + m \le \lceil \log_\lambda N \rceil + m$. This follows from the next lemma.

Lemma 3.2

$$N + \left(\sum_{t=0}^{T} c_t z^t\right) g(z) \preceq z^{T+1}(b+2)h(z) .$$

Proof :

$$\begin{aligned} & N + \sum_{t=0}^{T} \lceil A/\lambda^t + b \rceil z^t g(z) = \\ & N + \sum_{t=0}^{T} \lceil A/\lambda^t + b \rceil z^{t+1} h(z) - \sum_{t=0}^{T} \lceil A/\lambda^t + b \rceil \lambda z^t h(z) \\ &= N + \lceil A/\lambda^T + b \rceil z^{T+1} h(z) - \lceil A + b \rceil \lambda h(z) + \\ & \qquad \sum_{t=1}^{T} \left(\lceil A/\lambda^{t-1} + b \rceil - \lceil A/\lambda^t + b \rceil \lambda \right) z^t h(z) \\ &\preceq \lceil A/\lambda^T + b \rceil z^{T+1} h(z) \\ &\preceq z^{T+1}(b+2)h(z). \end{aligned}$$

We used the fact that $\lceil B + b \rceil < \lceil B/\lambda + b \rceil \lambda$ for any $B > 0$. The number b was chosen to ensure this. □

The constant number of outputs produced by these networks could be reduced to two using an additional fixed delay, independent of N.

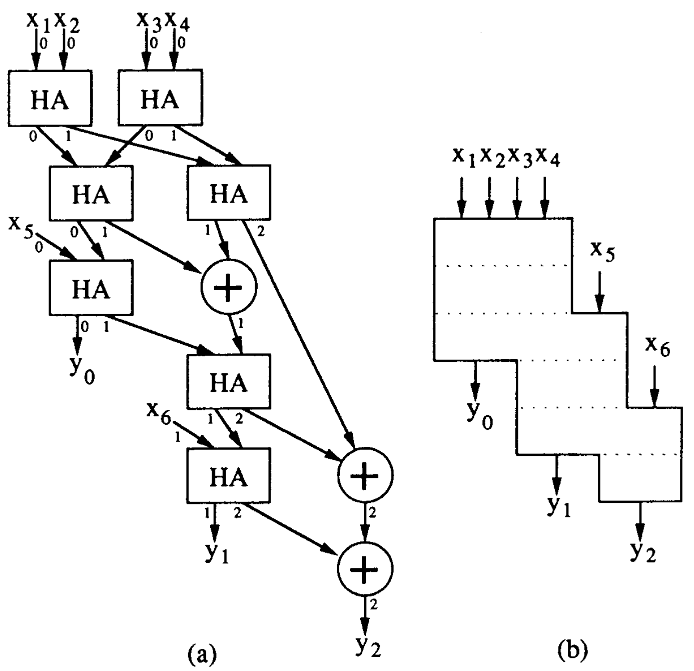

Figure 4.1: An implementation of an $FA_{5,1}$.

4 A 6 → 3 Carry Save Adder

An implementation of an $FA_{5,1}$ is given in Fig. 4.1(a). The implementation uses seven *half adders (HA)* and three XOR gates. A *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). The three separate XOR gates used in Fig. 4.1(a) could in fact be replaced by OR gates.

In order to verify the validity of this implementation we imagine at first that the three XOR gates are replaced by *HA*'s. Notice that the connections between the *HA*'s respect the significances of the inputs and outputs. The significance associated with each wire in the circuit is written next to that wire in Fig. 4.1(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 that we used to replace the XOR gates are always zero so the *HA*'s could be replaced by XOR gates or by OR gates.

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 then obtained one unit of time later, even before x_6 needs to be supplied. The outputs y_1 and y_2 are obtained one and two units of time after x_6 is supplied. This behaviour is depicted in Fig. 4.1(b). The $CSA_{6\to3}$ constructed using this $FA_{5,1}$ will have the same delay characteristics.

The results of the previous section give us the optimal way of combining 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 root of the polynomial equation $\lambda^6 + \lambda^5 - \lambda^4 + \lambda^3 - \lambda^2 - 4 = 0$.

5 An 11 → 4 Carry Save Adder

The $CSA_{6\to3}$ described in the previous section relied heavily on the use of XOR gates. An XOR gate could always be replaced by three AND-like gates with a total delay of two time units. Better results are obtained however by using a completely different design.

Khrapchenko [9] gave the design of an FA_7 with which a $CSA_{7\to3}$ with the characteristics given in Fig. 5.2 could be constructed. He also described networks based on this *CSA* with asymptotic delay of $5.12 \log_2 n$. The networks that he described were not optimal however. Using the designs of section 3, or even the less general designs described in [11],[12], better networks of delay $5.07 \log_2 n$ can be obtained.

In this section we give an implementation of an $FA_{7,4}$ using which the preceding results can be further improved. Since the design of this unit is based on Khrapchenko's design, we give a concise summary of his construction in Fig. 5.1.

In Figs. 5.1 and 5.3 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 on 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.

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

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

Figure 5.1: Khrapchenko's construction of an FA_7.

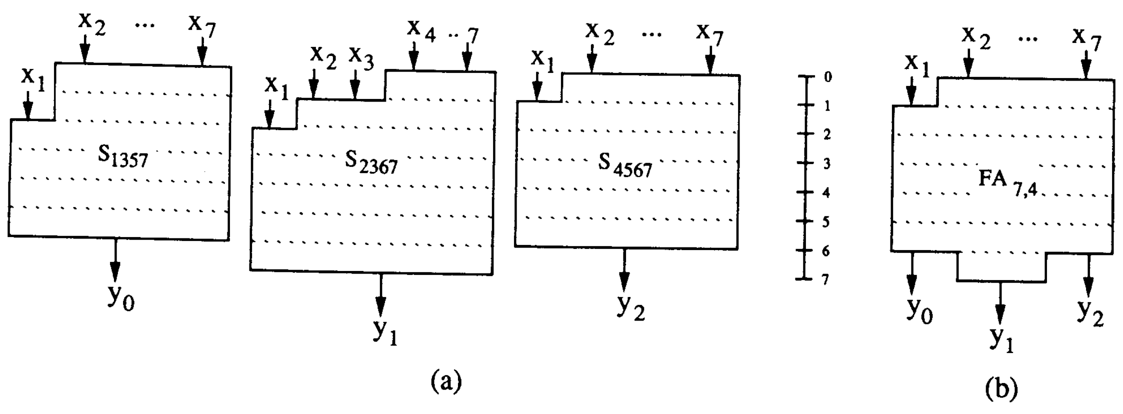

Figure 5.2: The delay characteristics of Khrapchenko's construction.

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

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

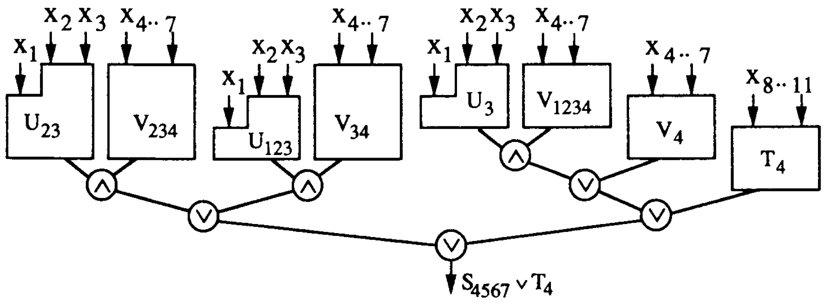

Figure 5.4: The final stages in the computation of $S_{4567} \vee T_4$.

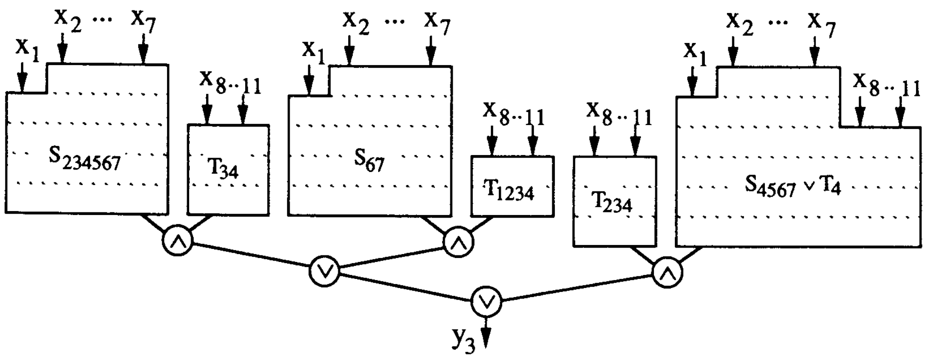

Figure 5.5: The final stages in the computation of y_3.

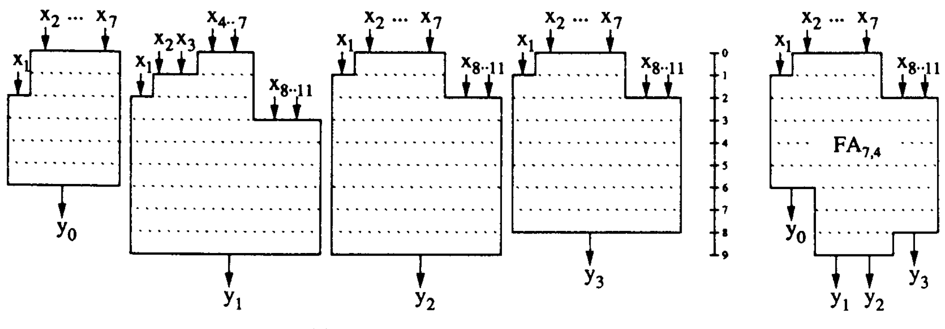

Figure 5.6: The delay characteristics of the new $FA_{7,4}$.

The numbers given to the left of each formula in Fig. 5.1 are the depths of the variables in that formula. They will be called ***delay vectors***. The delay characteristics of the three output bits y_0, y_1, y_2 that compose Khrapchenko's FA_7 are described in Fig. 5.2(a). In [12] it is shown that the optimal way of combining these three units into a single unit is as presented in Fig. 5.2(b).

The construction of our new $FA_{7,4}$ is given in Fig. 5.3. Figures 5.4 and 5.5 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 Fig. 5.6(a) and we see that we can fit them all into the unit outlined in Fig. 5.6(b).

By the results of section 3, we can combine the new $CSA_{11\to 4}$'s into networks of asymptotic depth $\log_\lambda n \simeq 4.95 \log_2 n$ where $\lambda \simeq 1.15041$ is the principal root of the equation $2\lambda^9 + \lambda^8 + \lambda^6 - 4\lambda^2 - \lambda - 6 = 0$.

6 Concluding remarks

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. There is a polynomial time algorithm which, for any *CSA G* and any n, gives an optimal-depth network of G's for the carry save addition of n inputs.

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

References

[1] El Gamal A., Gluss D., Ang P-H., Greene J., Reyneri J., "A CMOS 32 bit Wallace tree multiplier-accumulator," *1986 ISSCC Digest of Technical Papers*, pp. 194-195.

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

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

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

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

[6] Dunne P.E., *The complexity of Boolean networks*, Academic Press, 1988.

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

[8] 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.

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

[10] 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.

[11] Paterson M.S., Pippenger N., Zwick U., "Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions," *Proceedings of the 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, St. Louis 1990.

[12] Paterson M.S., Pippenger N., Zwick U., "Optimal carry save networks," *Boolean function complexity: Selected papers from the LMS symposium, Durham 1990.* To appear, Cambridge Univ. Press, 1991.

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

[14] Wegener I., *The complexity of Boolean functions*, Wiley-Teubner Series in Computer Science, 1987.