The Area-Time Complexity of Binary Multiplication

R. P. BRENT

The Australian National University, Canberra, Australia

AND

H. T. KUNG

Carnegie-Mellon University, Pittsburgh, Pennsylvania

ABSTRACT The problem of performing multiplication of n-bit binary numbers on a chip is considered Let A denote the chip area and T the time required to perform multiplication. By using a model of computation which is a realistic approximation to current and anticipated LSI or VLSI technology, it is shown that

$$\left(\frac{A}{A_0}\right)\left(\frac{T}{T_0}\right)^{2\alpha} \geq n^{1+\alpha}$$

for all $\alpha \in [0, 1]$, where A_0 and T_0 are positive constants which depend on the technology but are independent of n. The exponent $1 + \alpha$ is the best possible A consequence of this result is that binary multiplication is "harder" than binary addition More precisely, if $(AT^{2\alpha})_M(n)$ and $(AT^{2\alpha})_A(n)$ denote the minimum area-time complexity for n-bit binary multiplication and addition, respectively, then

$$\frac{(AT^{2\alpha})_M(n)}{(AT^{2\alpha})_A(n)} = \left\{\begin{array}{ll} \Omega(n^{1-\alpha}) & \text{for } 0 \leq \alpha \leq \frac{1}{2} \\ \Omega\left(\frac{n^\alpha}{\log^{2\alpha} n}\right) & \text{for } \frac{1}{2} < \alpha \leq 1 \\ \Omega\left(\frac{n}{\log^{2\alpha} n}\right) & \text{for } \alpha > 1 \end{array}\right\} \quad (= \Omega(n^{1/2}) \quad \text{for all} \quad \alpha \geq 0).$$

KEY WORDS AND PHRASES. area-time complexity, binary multiplication, chip design, chip layout, circuit design, combinational logic, chip complexity, lower bounds, VLSI

CR CATEGORIES: 5 25, 6 1, 6.32

1. *Introduction*

We are interested in the design of multipliers suitable for implementation in VLSI chips. The multiplication problem has been considered by several authors (see, e.g., [8, 10, 17, 19, 25, 27]). Much attention has been paid to the trade-off between time and the number of gates, but until recently little attention has been paid to the

This research was supported in part by the National Science Foundation under Grant MCS 78-236-76 and the Office of Naval Research under Contracts N00014-76-C-0370 and N00014-80-C-0236. Most of this work was carried out at the Australian National University while H. T. Kung was there as a Visiting Fellow during May 1979

Authors' addresses. R P Brent, Department of Computer Science, The Australian National University, Canberra, A C T 2600, Australia, H. T Kung, Department of Computer Science, Schenley Park, Carnegie-Mellon University, Pittsburgh, PA 15213

problem of connecting the gates in an economical and regular way to minimize chip area and design costs. In this paper we give lower and upper bounds on the area-time product for multiplication circuits, assuming a model of computation which is intended to approximate current and anticipated LSI or VLSI technology. Details of the model are given in Section 2.

The lower bound on AT, where A is the chip area and T the time to perform n-bit binary multiplication on the chip, is the special case $\alpha = \frac{1}{2}$ of a more general lower bound

$$AT^{2\alpha} = \Omega(n^{1+\alpha}), \tag{1.1}$$

which is valid for all $\alpha \in [0, 1]$. We establish this general result in Section 3. The case $\alpha = 1$ was established independently by Abelson and Andreae [1] using a more restrictive model than ours (see also [21]). In Section 4 we sketch a design for n-bit multiplication that gives the upper bound

$$AT^{2\alpha} = O(n^{1+\alpha}\log^{1+2\alpha}n), \tag{1.2}$$

for all $\alpha \geq 0$.[1] Thus the exponent $1 + \alpha$ of n in (1.1) and (1.2) is tight for $\alpha \in [0, 1]$.

In [3] we give upper bounds on A and T for the problem of adding n-bit binary numbers. From (1.1) and the results of [3] we conclude in Section 5 that binary multiplication is harder than binary addition if the complexity measure is $AT^{2\alpha}$, for any $\alpha \geq 0$ (see also [7]).

2. *The Computational Model and Basic Assumptions*

We assume the existence of circuit elements or "gates" which compute a logical function of two inputs in constant time and occupy at least a constant minimum area. Gates are connected by wires which have constant minimum width (equivalently, the wires must be separated by at least some minimal spacing). Our measure of the cost of a design is the area rather than the number of gates required. This is an important difference between our model and earlier models of [4, 26] and others. For motivation and discussion of models similar to ours, see [12, 23].

To prove the results of this paper, various subsets of the following assumptions A1 through A8 are used. Comments and justification are given following the statement of each assumption.

A1. The computation is performed in a convex planar region R of area A.

Because of heat-dissipation, packing, and testing requirements, a two-dimensional planar model is reasonable. The convexity assumption is not restrictive in the sense that almost all existing chips or useful modular designs do have convex boundaries for packaging or modularity reasons. (The convexity assumption can be removed for part of Theorem 3.1 below by using a different proof.)

A2. Wires have minimal width $\lambda > 0$.

λ is assumed constant, but in applications of our results it will of course depend on the technology. We also assume R has width at least λ in every direction.

A3. At most $\nu \geq 2$ wires can overlap (or intersect) at any point in R.

A chip may consist of ν layers. Wire crossings through different layers are allowed. In fact, transistors are typically formed by crossovers of wires. Since

[1] Log denotes log to the base 2 throughout.

$\nu \geq 2$, the graph of wires (edges) and gates (nodes) need not be planar in a graph-theoretic sense.

A4. I/O ports each contain a $\lambda \times \lambda$ square and thus have area at least $\rho \geq \lambda^2$. An I/O port can be multiplexed to handle more than one input or output variable.

If R is a complete chip, ρ will be large compared to λ^2. If R is only part of a chip and I/O is to other regions on the chip, ρ could be of order λ^2. We do not require each input (or output) variable to appear in a distinct input (or output) port, as required in [23]. I/O ports may be multiplexed as they often are in practice.

A5. A bit requires minimal time $\tau > 0$ to propagate along a wire or to be transmitted through an I/O port. The time for one gate computation and an arbitrary fanout of the result is included in τ.

Since dimensions are limited by the minimal wire width λ and minimal gate area, a minimal propagation time is reasonable. We do not need to assume that the propagation time increases with the length of the wire. With the (small) sizes of chips we now have or anticipate, the propagation time, which is the time needed to charge or discharge a wire, is limited by the wire capacitance rather than the velocity of light. A longer wire will generally have a larger capacitance and thus require a larger driver to maintain constant propagation time, but the driver area need not exceed a fixed percentage of the wire area and so can be ignored if λ is increased slightly; see [15]. Although it would be reasonable to assume bounded fanout, we do not need this assumption for proving lower bounds. When proving upper bounds, we do assume bounded fanout.

A6. The times and locations at which input and output bits are available are fixed and independent of the values of the input bits.

When proving upper bounds in Section 4, we further assume that if a_i and a_j are any two bits in an operand such that a_i is more significant than a_j, then a_i is not input to (or output from) the chip before a_j, but they are allowed to be input to (or output from) the chip in parallel.

A7. Storage for one bit of information takes area at least $\beta > 0$.

β is typically several times larger than λ^2.

A8. Each input bit is available only once.

There is no free memory outside R. If the same input bit is required at different times, it must be stored within R, taking area at least β (see A7).

3. *Lower Bound Results*

Let $p = p_{2n} \cdots p_1$ be the $2n$-bit product of n-bit integers $a = a_n \cdots a_1$ and $b = b_n \cdots b_1$.

3.1 Lower Bounds for Shifting Circuits. When $b = 2^j$, p is a shifted j bits to the left. Thus any multiplier circuit must also be a shifting circuit capable of performing j-bit shifts for all $0 \leq j \leq n - 1$.

Theorem 3.1. *Under assumptions $A1$–$A6$ of Section 2, any chip that is capable of performing the shifts described above must satisfy*

$$AT^2 \geq K_1 n^2, \tag{3.1}$$

and

$$AT \geq K_2 Ln, \tag{3.2}$$

where

$$K_1 = 2\left[\frac{\lambda\tau(9 - 4\cdot 5^{1/2})}{\nu}\right]^2, \qquad K_2 = \frac{\lambda\tau(9 - 4\cdot 5^{1/2})}{\pi\nu}, \tag{3.3}$$

and L is the perimeter of the chip.

Before proving Theorem 3.1 we need two Lemmas.

LEMMA 3.1. *For any convex planar figure with area A, perimeter L, diameter D, and chord of length C perpendicular to a chord whose length is the diameter D,*

$$A \geq \frac{CD}{2}, \tag{3.4}$$

and

$$A \geq \frac{CL}{2\pi}. \tag{3.5}$$

PROOF. The results follow from well-known inequalities for convex figures. For a proof (and a definition of "diameter," etc.) see, for example, [28]. □

LEMMA 3.2

$$\min_{0 \leq r < 1} \max(2r, (1 - r)^2/8) = 2(9 - 4\cdot 5^{1/2}).$$

PROOF. It is easy to verify that the minimum occurs when $16r = (1 - r)^2$, and the only root of this equation in $[0, 1]$ is $r = 9 - 4\cdot 5^{1/2}$. □

PROOF OF THEOREM 3.1. Consider any chip that can perform j-bit shifts for all $0 \leq j \leq n - 1$. By assumption A1, the chip forms a convex region R. Let D be the diameter of R, and Y a chord of length D.

Let $S = \{p_{2n-1}, \ldots, p_n\}$ and M be the maximum number of elements of S sharing or multiplexing one output port of the chip. By assumption A4, an I/O port has area at least $\rho \geq \lambda^2$. We represent each I/O port by an infinitesimal point on the port. On the basis of these representatives of I/O ports, we partition the chip by a chord X perpendicular to Y as follows. The chord X divides S into two subsets S_1 and S_2 such that representatives of the output ports for elements of S_1 lie on one side of X and those for elements of S_2 lie on the other side of X. (Since representatives of I/O ports are of infinitesimal size, we can assume that by an infinitesimal perturbation from the perpendicular to Y, X does not intersect any of them.) By "sliding" the intersection of X and Y along Y, we can arrange that

$$|S_i| \leq \left\lfloor \frac{n + M}{2} \right\rfloor \tag{3.6}$$

for $i = 1$ and 2. For notational convenience we use d to denote $\lfloor (n + M)/2 \rfloor$. When the j-bit shift is performed, p_{i+j} takes the value of a_i. For $d \leq i \leq n$, the ith row in Table I indicates the p_i's that take the value of a_i under j-bit shifts for all $n - i \leq j \leq n - 1$. Note that in the table all the p_i's belong to S, which is divided into two parts by the chord X. By (3.6), in the ith row of the table there are at most d of

TABLE I THE DEPENDENCE OF THE p_i'S ON THE a_i'S UNDER VARIOUS SHIFTS

	j								
a_i	0	1	2	$\cdots$	$n-d-1$	$n-d$	$\cdots$	$n-2$	$n-1$
a_d						p_n	$\cdots$	p_{n+d-2}	p_{n+d-1}
a_{d+1}					p_n	$\cdots$		p_{n+d-1}	p_{n+d}
$\vdots$									
a_{n-1}		p_n	p_{n+1}		$\cdots$			p_{2n-3}	p_{2n-2}
a_n	p_n	p_{n+1}	p_{n+2}		$\cdots$			p_{2n-2}	p_{2n-1}

the p_i's for which the representatives of the output ports lie on the same side of X as the representative of the input port for a_i. Consequently, in the ith row there are at least $i-d$ of the p_i's for which the representative of the output ports *do not* lie on the same side of X as the representative of the input port for a_i. For all rows in the table, there are a total of at least $\sum_{i=d}^{n}(i-d) \geq (n-M)^2/8$ such p_i's. This implies that one of the n columns in the table, say the jth column, must have at least $(n-M)^2/8n$ such p_i's. In other words, if

$I = \{i \mid i \in \{d, d+1, \ldots, n\}$ and the representative of the input port for a_i does not lie on the same side of X as that of the output port for $p_{i+j}\}$,

then

$$|I| \geq \frac{(n-M)^2}{8n}.$$

For $i \in I$, the input port for a_i or the output port for p_{i+j} may intersect the chord X, although their representatives do not. Define

$I' = \{i \mid i \in I$, and the chord X intersects the input port for a_i or the output port for p_{i+j}, or both$\}$.

Then

$I - I' = \{i \mid i \in \{d, d+1, \ldots, n\}$, and the input port for a_i and the output port for p_{i+j} do not intersect X and they lie on different sides of $X\}$.

Consider the computation of the j-bit shift. Note that the j-bit shift, which maps a_i to p_{i+j} for $i = 1, \ldots, n$, is an identity mapping. Hence, before the shift is complete, at least $|I - I'|$ bits of information about a_i, $i \in I - I'$, must cross X for computing p_{i+j}, $i \in I - I'$, and at least $|I'|$ bits of information about a_i, $i \in I'$, must be input to or output from some I/O ports intersecting X for computing p_{i+j}, $i \in I'$. Suppose that the chord X is of length C. Then by assumptions A2–A4, at most $\nu C/\lambda$ wires or I/O ports cross X. Thus, by assumption A5, the time T to perform the j-bit shift must satisfy the inequality

$$\left(\frac{\nu C}{\lambda}\right)\left(\frac{T}{\tau}\right) \geq |I - I'| + |I'| = |I| \geq \frac{(n-M)^2}{8n},$$

or

$$T \geq \frac{(\lambda\tau/\nu C)n\cdot(1-r)^2}{8}, \tag{3.7}$$

where $r = M/n$. Since M outputs come through one output port, assumption A5 gives

$$T \geq M\tau = \tau n r. \tag{3.8}$$

First suppose $M < n$. Then at least one wire or one I/O port crosses X, and assumptions A2 and A4 give

$$C \geq \lambda. \tag{3.9}$$

By assumption A3, $\nu \geq 2$. Combining this with (3.8) and (3.9) gives

$$T \geq \tau n r = \left(\frac{2C\tau}{2C}\right) n r \geq \left(\frac{\lambda\tau}{\nu C}\right) n \cdot 2r. \tag{3.10}$$

From (3.7) and (3.10) it follows by Lemma 3.2 that

$$T \geq \left(\frac{2K_0}{C}\right) n, \tag{3.11}$$

where

$$K_0 = \frac{\lambda\tau(9 - 4 \cdot 5^{1/2})}{\nu},$$

so by (3.4),

$$AT^2 \geq \left(\frac{CD}{2}\right)\left(\frac{2K_0}{C}\right)^2 n^2 \geq 2K_0^2 n^2, \tag{3.12}$$

since $D \geq C$. Suppose on the other hand that $M = n$. Then $r = 1$. Since there is at least one output port, assumption A4 gives $A \geq \rho \geq \lambda^2$, so by (3.8),

$$AT^2 \geq (\lambda\tau n)^2 > 2K_0^2 n^2. \tag{3.13}$$

Result (3.1) follows from (3.12) and (3.13).

Result (3.2) follows in a similar way. If $M < n$, combining (3.11) with (3.5) gives

$$AT \geq \left(\frac{CL}{2\pi}\right)\left(\frac{2K_0}{C}\right) n = K_2 L n. \tag{3.14}$$

Suppose on the other hand that $M = n$. By assumption A2, R has width at least λ in every direction, so we can choose a chord that is of length $C \geq \lambda$ and is perpendicular to Y. By (3.5) and (3.8) with $r = 1$, we have

$$AT \geq \left(\frac{CL}{2\pi}\right)(\tau n),$$

which gives

$$AT \geq K_2 L n. \qquad \square$$

Since any circuit that performs integer multiplications must also be able to perform shifts, (3.1) and (3.2) hold for any n-bit multiplication chip.

Result (3.2) can sometimes give useful lower bounds which are based on the I/O characteristics of a multiplication or shifting chip. If at one time the chip inputs or outputs a total of z bits along its boundary, then by assumptions A3 and A4, $L \geq z\lambda/\nu$, and (3.2) gives $AT \geq K_2(\lambda z/\nu)n$. Thus for any multiplication scheme that accepts, say $\Omega(n^{1/2})$ input bits simultaneously along the chip boundary, we know immediately that $AT = \Omega(n^{3/2})$ (cf. the multiplication scheme in Section 4).

Result (3.1) (with a smaller constant for K_1) could have been established by a proof parallel to that used by Thompson [23] for the discrete Fourier transform problem. In fact, using his result that relates the area of a graph to its minimum bisection width, one can derive (3.1) without the convexity assumption in A1. Our proof above represents a new approach that incorporates geometric properties of the chip boundary in the lower bound proof. We feel that the extra convexity assumption we make is not restrictive, since most existing chips do have convex boundaries for packaging reasons. Furthermore, we note that the convexity assumption is needed for establishing results such as (3.2) that relate AT to the perimeter L. In [6], under a similar convexity assumption, tight lower bounds on the minimum area required to layout complete binary (or t-ary) trees are obtained.

An interesting corollary of Theorem 3.1 is that lower bounds in (3.1) and (3.2) hold for chips that perform floating-point additions, for which shifts are needed to equalize exponents. This explains why the area–time requirements for floating-point addition are much higher than those for integer addition, as observed in practical implementations. (Charles Leiserson at CMU first pointed out to one of the authors the application of Theorem 3.1 to floating-point addition.)

3.2 A LOWER BOUND ON THE AREA FOR MULTIPLIER CIRCUITS. In Theorem 3.1 we gave lower bounds on AT^2 and AT for shifting circuits. Now, using different techniques, we give a lower bound on A for multiplier circuits.

THEOREM 3.2. *Under assumptions A4 and A6–A8, any n-bit multiplication must satisfy*

$$A \geq A_0 n,$$

where

$$A_0 = \frac{5}{6}\left(\frac{\beta\rho}{\beta + \rho}\right). \tag{3.15}$$

Let $\Phi_N = \{ij \mid 0 \leq i < N, 0 \leq j < N\}$ be the set of all integers which can be written as a product of two factors, each less than N, and let $\mu(N) = |\Phi_N|$ be the cardinality of Φ_N. For example, $\Phi_4 = \{0, 1, 2, 3, 4, 6, 9\}$ and $\mu(4) = 7$. Before proving Theorem 3.2 we need lower bounds on $\mu(N)$ and a related function,

$$\delta(n) = \frac{\lceil \log \mu(2^n) + 1 - n \rceil}{n}. \tag{3.16}$$

LEMMA 3.3

$$\mu(N) \geq \sigma(N),$$

where $\sigma(N) = \sum_{p \in P_{N-1}} p$ and P_{N-1} is the set of prime numbers smaller than N.

PROOF. The numbers pj are distinct if $p \in P_{N-1}$ and $1 \leq j \leq p$. Thus the result follows from the definition of $\mu(N)$. □

LEMMA 3.4. *For all $N \geq 4$,*

$$\mu(N) \geq \frac{N^2}{2 \ln N}.$$

PROOF. Using a slight modification of Theorem 1 and eq. (4.13) of [18], we can show that for all $N \geq 348$,

$$\sigma(N) > \frac{N^2}{2 \ln N}.$$

Thus the result for $N \geq 348$ follows from Lemma 3.3. For $4 \leq N \leq 347$ the result may be verified by a straightforward computation. □

TABLE II. $\mu(2^n)$ AND RELATED FUNCTIONS FOR $1 \le n \le 17$

n	$\mu(2^n)$	$\mu(2^n)/\mu^*(2^n)$	$\delta(n)$
1	2	0.355000	1
2	7	0.748125	1
3	26	0 932329	1
4	90	0 952734	1
5	340	1.006695	1
6	1,238	0 995890	1
7	4,647	0.997629	1
8	17,578	0.995092	1
9	67,592	1.000412	1
10	259,768	0.998846	9/10
11	1,004,348	0.998392	10/11
12	3,902,357	0.999002	11/12
13	15,202,050	0.999089	12/13
14	59,410,557	0 999788	13/14
15	232,483,840	0.999637	14/15
16	911,689,012	0.999788	15/16
17	3,581,049,040	1.000005	16/17

LEMMA 3.5. *If* $\delta(n)$ *is defined by* (3.16), *then for all* $n \ge 1$,

$$\delta(n) \ge \tfrac{5}{6}.$$

PROOF. From Lemma 3.4,

$$\delta(n) \ge \frac{\lceil n - \log(n \ln 2) \rceil}{n}, \tag{3.17}$$

and it is easy to verify that the right side of (3.17) is at least $\frac{5}{6}$ for all $n \ge 18$. (There is equality for $n = 18$ and $n = 24$.) For $1 \le n \le 17$, direct computation shows that $\delta(n) \ge \frac{9}{10}$. □

Table II gives $\mu(2^n)$, $\mu(2^n)/\mu^*(2^n)$, and $\delta(n)$ for $n = 1, 2, \ldots, 17$, where

$$\mu^*(N) = \frac{N^2}{0.71 + \log\log N}$$

is an empirical approximation to $\mu(N)$. For $5 \le n \le 17$, the approximation error is less than 1 percent. If this remained true for $n > 17$, it would follow that $\delta(n) \ge \frac{9}{10}$, and the constant $\frac{5}{6}$ in Lemma 3.5 and Theorem 3.2 could be increased. On the basis of the empirical evidence we conjecture that

$$\lim_{N\to\infty}\left(\frac{\mu(N)\log\log N}{N^2}\right) = 1 \quad \text{and} \quad \delta(n) \ge \frac{9}{10}$$

for all $n \ge 1$.

PROOF OF THEOREM 3.2. If $n = 1$, there is at least one output port, so $A \ge \rho$, and the result holds. Hence, suppose that $n \ge 2$.

Consider the state of the computation just before the last input bit(s) is accepted. Let m be the number of input bits still to be accepted, so $1 \le m \le 2n$.

It is easy to show that there are some inputs a and b such that the output bits $p_{2n}, \ldots, p_n$ are not determined by the $2n - m$ input bits already accepted. Thus, by assumption A6, at most $n - 1$ bits, $p_{n-1}, \ldots, p_1$, have been output.

Suppose that s bits of information are stored in R at this instant. Then we must have, by assumption A8,

$$\mu(2^n) \le 2^{m+(n-1)+s},$$

or the circuit could not produce all $\mu(2^n)$ possible outputs and would fail for certain inputs. Thus

$$m + s \ge \lceil \log \mu(2^n) + 1 - n \rceil = n\delta(n),$$

and, from Lemma 3.5,

$$m + s \ge \frac{5n}{6}. \tag{3.18}$$

By assumption A7,

$$A \ge \beta s. \tag{3.19}$$

Since a port can accept only one bit at a time, the last m bits must be input through m different ports; so assumption A4 gives

$$A \ge \rho m. \tag{3.20}$$

The result follows easily from (3.18)–(3.20). □

3.3 General Lower Bounds for Multiplier Circuits. Theorems 3.1 and 3.2 are the extreme cases $\alpha = 1$ and $\alpha = 0$ of the following result.

Theorem 3.3. *Under assumptions A1–A8, any n-bit multiplication chip must satisfy*

$$\left(\frac{A}{A_0}\right)\left(\frac{T}{T_0}\right)^{2\alpha} \ge n^{1+\alpha}, \tag{3.21}$$

for all $\alpha \in (0, 1)$. Here A_0 is given by (3.15),

$$T_0 = \left(\frac{K_1}{A_0}\right)^{1/2},$$

and K_1 is given by (3.3).

Proof. From Theorem 3.1,

$$\left(\frac{A}{A_0}\right)\left(\frac{T}{T_0}\right)^{2} \ge n^2,$$

so

$$\left(\frac{A}{A_0}\right)^{\alpha}\left(\frac{T}{T_0}\right)^{2\alpha} \ge n^{2\alpha}. \tag{3.22}$$

From Theorem 3.2, since $\alpha \in [0, 1]$,

$$\left(\frac{A}{A_0}\right)^{1-\alpha} \ge n^{1-\alpha}. \tag{3.23}$$

Multiplying (3.22) and (3.23) gives the result. □

The following corollary of Theorem 3.3 seems worth stating separately, for AT is often used as a complexity measure (see, e.g., [16]).

COROLLARY 3.1. *Under assumptions A1–A8, any n-bit multiplication chip must satisfy*

$$AT \geq K_3 n^{3/2},$$

where

$$K_3 = A_0 T_0 = (A_0 K_1)^{1/2}.$$

4. *Upper Bound Results for Multiplication*

It is easy to design practical n-bit multipliers with area $A = O(n)$ and time $T = O(n)$, so

$$AT^{2\alpha} = O(n^{1+2\alpha}). \tag{4.1}$$

For example, the "serial pipeline multipliers" typically used in the implementation of digital filters and signal processors achieve these area and time bounds (see [9, 14]). In this section we sketch the design of a multiplier with $A = O(n \log n)$ and $T = O(n^{1/2} \log n)$, giving

$$AT^{2\alpha} = O(n^{1+\alpha} \log^{1+2\alpha} n), \tag{4.2}$$

which is asymptotically better than (4.1). The design uses the Convolution Theorem to compute the product of two integers in a complex way, and consequently its implementation appears to be difficult. Nevertheless, the design is theoretically interesting because it shows that the exponent $1 + \alpha$ of n in Theorem 3.3 is tight. We do not know if there is any practical design having $AT^{2\alpha} = o(n^{1+2\alpha})$ for $\alpha \in [0, 1]$. Straightforward implementations of "fast" algorithms, for example, the Schonhage–Strassen algorithm [22] or the "3–2 reduction" algorithm [17, 25], seem to require area at least order n^2.

In the remainder of this section we assume that

(a) $n = k^2$ is a perfect square, and
(b) $a_j = b_j = 0$ if $j > n/2$.

(If not, n may be increased sufficiently without affecting the asymptotic results.) Let p be the smallest prime of the form $nq + 1$, $q \geq 1$, F_p the finite field of integers mod p. It is known that $\log p = O(\log n)$ (see [13, 24]) and that F_p has an nth root of unity u (see [2]). Let $w = u^k$, so w is a kth root of unity. Note that in any circuit n is fixed, so we are not concerned with the complexity of finding p, u, w, etc; they will be encoded into the circuit. For facilitating arithmetic in F_p we assume that a $2\lceil \log p \rceil$-bit approximation to $1/p$ is encoded into the circuit.

In steps 1–5 below, all arithmetic is done in F_p. In steps 1–3 we compute the discrete Fourier transform a' of $(a_1, \ldots, a_n)$ and b' of $(b_1, \ldots, b_n)$ over F_p; that is,

$$a'_{j+1} = \sum_{i=0}^{n-1} a_{i+1} u^{ij}$$

for $j = 0, \ldots, n - 1$, etc. In step 4 we multiply the Fourier transforms. In step 5 we take the inverse transform, and in step 6 the final result is computed.

Step 1. Let A, B, U, and W be k by k matrices with elements

$$A_{ij} = a_{(i-1)k+j}, \qquad U_{ij} = u^{(i-1)(j-1)},$$
$$B_{ij} = b_{(i-1)k+j}, \qquad W_{ij} = w^{(i-1)(j-1)}.$$

Perform k by k matrix multiplications to compute

$$A' = WA \qquad \text{and} \qquad B' = WB,$$

using a "systolic array" [11]. All computations are performed in F_p, so each processing element of the systolic array needs to perform multiplication and addition in F_p. Using a serial pipeline multiplier and a serial adder, a multiplication and addition step in F_p requires no more than area $O(\log p)$ and time $O(\log p)$. Thus, step 1 can be done with area $O(n \log n)$ and time $O(n^{1/2} \log n)$.

Step 2. Compute $A'' = A' \circ U$ and $B'' = B' \circ U$, where $\circ$ denotes componentwise multiplication.

Step 3. Compute $A''' = A''W$ and $B''' = B''W$ using the same method as for step 1. It may be shown that A''' and B''' contain the Fourier transforms of $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$; in fact, for $1 \le i, j \le k$,

$$A'''_{ij} = a'_{(j-1)k+i}, \qquad B'''_{ij} = b'_{(j-1)k+i}.$$

Step 4. Compute $C''' = A''' \circ B'''$.

Step 5. Compute $C = W^{-1}(U' \circ (C''' W^{-1}))$ as in steps 1–3. Here $U'_{ij} = u^{-(i-1)(j-1)}$. The matrix C represents the inverse Fourier transform of C'''. Define the c_i's by

$$C_{ij} = c_{(i-1)k+j}.$$

Then by the Convolution Theorem and assumptions (a) and (b) above,

$$c_j = a_1 b_j + a_2 b_{j-1} + \cdots + a_j b_1 \qquad \text{for} \quad 1 \le j \le n.$$

Thus,

$$\sum_{i=1}^{2n} p_i 2^{i-1} = \sum_{i=1}^{n} c_i 2^{i-1}.$$

Grouping the terms on the right-hand side into $k = n^{1/2}$ groups so that the c_i's in each row of the matrix C belong to one group, we obtain

$$\sum_{i=1}^{2n} p_i 2^{i-1} = \sum_{i=1}^{k} R_i 2^{(i-1)k}, \tag{4.3}$$

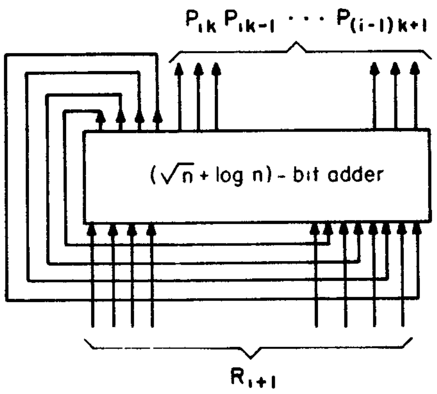
where

$$R_i = \sum_{j=1}^{k} c_{(i-1)k+j} 2^{j-1}.$$

Given that the c_i's are outputs of the systolic array that computes the matrix C, all the R_i's can be formed in area $O(n \log n)$ and time $O(n^{1/2} \log n)$, using the result of Theorem 5.1 of Section 5 regarding addition circuits. Thus the problem of computing $p_{2n}, \ldots, p_1$ has been reduced to the problem of summing $k = n^{1/2}$ terms in the right-hand side of eq. (4.3). Hence, the final step in the computation is

Step 6. Compute $p_{2n}, \ldots, p_1$ from the R_i's. Note that each R_i has at most $n^{1/2} + \log n$ bits. Using (4.3), the p_i's can be computed, $n^{1/2}$ of them at a time, with an $(n^{1/2} + \log n)$-bit adder. This is depicted in Figure 1. At the end of the ith addition, the first $n^{1/2}$ low order bits in the output are output as $p_{ik}, p_{ik-1}, \ldots, p_{(i-1)k+1}$, and the remaining bits in the output are fed back to the adder to be added to the arriving R_i in the $(i + 1)$st addition. With the result of Theorem 5.1 one can easily see that all the p_i's can be computed in area $O(n \log n)$ and time $O(n^{1/2} \log n)$.

This completes our outline of the multiplier with area $A = O(n \log n)$ and time $T = O(n^{1/2} \log n)$, giving $AT^{2\alpha} = O(n^{1+\alpha} \log^{1+2\alpha} n)$.



$P_{ik}\ P_{ik-1} \cdots P_{(i-1)k+1}$
$(\sqrt{n} + \log n)$ - bit adder
R_{i+1}

FIG 1. Computing the p_i's from the R_i's.

For $\alpha \in [0, 1]$, the exponent $1 + 2\alpha$ of $\log n$ can be reduced by using a more complicated design than the one outlined above, but we do not know what its minimal value is. For $\alpha > 1$, a design based on the "3–2 reduction" algorithm gives $AT^{2\alpha} = O(n^2 \log^\delta n)$ for some $\delta > 0$, which is a better upper bound than (4.2).

5. *Concluding Remarks*

In [3] we demonstrate a regular layout for look-ahead adders, giving the following result.

THEOREM 5.1. *Let $1 \leq w \leq n$. Then all the carries in an n-bit addition can be computed in time proportional to $(n/w) + \log w$ and in area proportional to $w \log w +$ 1, and so can the addition.*

Let $(AT^{2\alpha})_M(n)$ and $(AT^{2\alpha})_A(n)$ be the area–time complexity for n-bit integer multiplication and addition, respectively. Note that the serial adder gives $(AT^{2\alpha})_A(n) = O(n^{2\alpha})$, and that for $\alpha > 1$, $(AT^{2\alpha})_M(n) = \Omega(n^2)$, since for multiplication, by (3.1), $A(T/\tau)^{2\alpha} > A(T/\tau)^2 \geq K_1(n/\tau)^2$. These observations together with Theorems 3.3 and 5.1 establish the following result.

THEOREM 5.2 *Under assumptions A1–A8 of Section 2,*

$$\frac{(AT^{2\alpha})_M(n)}{(AT^{2\alpha})_A(n)} = \left\{\begin{array}{ll} \Omega(n^{1-\alpha}) & \text{for } 0 \leq \alpha \leq \frac{1}{2} \\ \Omega\left(\frac{n^\alpha}{\log^{2\alpha} n}\right) & \text{for } \frac{1}{2} < \alpha \leq 1 \\ \Omega\left(\frac{n}{\log^{2\alpha} n}\right) & \text{for } \alpha > 1 \end{array}\right\} (= \Omega(n^{1/2}) \quad \text{for all} \quad \alpha \geq 0).$$

Thus for any $\alpha \geq 0$, the area–time product for multiplication is asymptotically larger than that for addition. We can say that multiplication is harder than addition as far as the area–time complexity is concerned.

For binary division it is easy to deduce a lower bound of the same form as (3.21), using the method of [5], and an upper bound $AT^{2\alpha} = O(n^{1+\alpha} \log^{1+2\alpha} n)$, using Newton's method.

In Section 3 we derived lower bounds on $AT^{2\alpha}$, $\alpha \in [0, 1]$, for binary multiplication. Similar lower bounds on AT^2 have been obtained for computation of the discrete Fourier transform by Thompson [23], and, for matrix multiplication by Savage [20]. It seems that area-time complexity is, in general, a useful measure for establishing the complexity hierarchy of many classes of problems because it captures important attributes of a computation such as time and space, as well as communication. One should expect that more results along this line will be obtained in the near future.

REFERENCES

1. ABELSON, H., AND ANDREAE, P Information transfer and area-time trade-offs for VLSI multiplication *Commun ACM 23,* 1 (Jan 1980), 20-23
2. BONNEAU, R.J. A class of finite computation structures supporting the fast Fourier transform. Tech Rep MAC Tech Memo 31, Project MAC, Massachusetts Institute of Technology, Cambridge, Mass, March 1973
3. BRENT, R P, AND KUNG, H T A regular layout for parallel adders. Tech. Rep CMU-CS-79-131, Dep of Computer Science, Carnegie-Mellon Univ, Pittsburgh, Pa, June, 1979 (to appear in *IEEE Trans. Comput.*).
4. BRENT, R P On the addition of binary numbers *IEEE Trans. Comput C-19* (1970), 758-759.
5. BRENT, R P The complexity of multiple-precision arithmetic In *The Complexity of Computational Problem Solving,* R.S Anderssen and R P Brent, Eds, University of Queensland Press, Brisbane, Australia, 1976, pp. 126-165.
6. BRENT, R.P., AND KUNG, H T On the area of binary tree layouts. *Inf Proc Letters 11,* (1980), 46-48
7. BRENT, R P, AND KUNG, H T. The chip complexity of binary arithmetic Proc 12th Ann ACM Symp on Theory of Computing, Los Angeles, Calif., April 1980, pp 190-200
8. GARNER, H.L A survey of some recent contributions to computer arithmetic *IEEE Trans Comput. C-25* (1976), 1277-1282.
9. JACKSON, L.B., KAISER, S F, AND MCDONALD, H S An approach to the implementation of digital filters. *IEEE Trans Audio Electroacoust. AU-16* (Sept 1968), 413-421.
10. KUCK, D.J. *The Structure of Computers and Computations.* John Wiley & Sons, New York, 1978.
11. KUNG, H.T., AND LEISERSON, C E Systolic arrays (for VLSI). Sparse Matrix Proceedings 1978, Knoxville, Tenn., Society for Industrial and Applied Mathematics, 1979, pp 256-282 (a slightly different version appears in [15, Sec 8 3])
12. LEISERSON, C E. Area-efficient graph layouts (for VLSI) Carnegie-Mellon Univ., Pittsburgh, Pa., Feb. 1980
13. LINNIK, U V On the least prime in an arithmetic progression. I The basic theorem. *Rec. Math 15* (1944), 139-178
14. LYON, R.F Two's complement pipeline multipliers" *IEEE Trans Commun. COM-24,* 4 (April 1976), 418-425.
15. MEAD, C.A., AND CONWAY, L A *Introduction to VLSI Systems* Addison-Wesley, Reading, Mass, 1980.
16. MEAD, C A, AND REM, M. Cost and performance of VLSI computing structures *IEEE J Solid State Circuits SC-14,* 2 (April 1979), 455-462.
17. OFMAN, Y On the algorithm complexity of discrete functions. *Dokl. Akad. Nauk SSSR 145* (1962), 48-51 (in Russian)
18. ROSSER, J B, AND SCHOENFELD, L. Approximate formulas for some functions of prime numbers. *Illinois J Math. 6* (1962), 64-94
19. SAVAGE, J E *The Complexity of Computing* John Wiley & Sons, New York, 1976
20. SAVAGE, J E. Area-time tradeoffs for matrix multiplication and related problems in VLSI models Tech. Rep. CS-50, Brown Univ, Providence, R I, Aug 1979
21. SAVAGE, J E, AND SWAMY, S Space-time tradeoffs for oblivious sorting and integer multiplication Tech Rep CS-37, Brown Univ, Providence, R I, 1978
22. SCHONHAGE A., AND STRASSEN, V. Schnelle Multiplikation grosser Zahlen *Comput. 7* (1971), 281-292
23. THOMPSON, C D Area-time complexity for VLSI Proc 11th Ann ACM Symp. on Theory of Computing, Atlanta, Ga, May 1979, pp 81-88.
24. WAGSTAFF, S S. JR Greatest of the least primes in arithmetic progressions having a given modulus *Math Comp. 33* (1979), 1073-1083.

25. WALLACE, C.S. A suggestion for a fast multiplier. *IEEE Trans. Elec. Comput. EC-13* (1964), 14–17.
26. WINOGRAD, S. On the time required to perform addition. *J. ACM 12*, 2 (April 1965), 277–285.
27 WINOGRAD, S. On the time required to perform multiplication. *J. ACM 14*, 4 (Oct. 1967), 793–802.
28. YAGLOM, I.M., AND BOLTYANSKII, V.G. *Convex Figures*. Holt, Rinehart and Winston, New York, 1961 (translated by P.J. Kelly and L F Walton).

RECEIVED AUGUST 1979; REVISED MARCH 1980, ACCEPTED APRIL 1980