Computation Times of Arithmetic and Boolean Functions in (d,r) Circuits

PHILIP M. SPIRA

Abstract—A (d, r) circuit is a d-valued logical circuit in which each element has fan-in at most r and can compute any r-argument d-valued logical function in unit time. In this paper we review results previously published on the computation time of such circuits for addition and multiplication and for computation of general Boolean functions. We also explicitly state hitherto unpublished but known results on the time necessary to divide in such circuits.

Index Terms—Circuits of elements of limited fan-in, time of arithmetic operations, time of computation.

I. Introduction

THIS paper summarizes recent results on the time necessary to compute arithmetic operations and Boolean functions with switching circuits of elements with limited fan-in. Most of the material has appeared in previous publications of Ofman [1], Karatsuba and Ofman [2], Winograd [3]-[5], Spira [6]-[8], Brent [9], and Marques [10]. The rest of the material has not been previously published though it is known to specialists in the field.

We believe we have summarized most of the relevant literature on the subject with one notable exception—those papers dealing with redundant number representation. The reader interested in this topic is referred to the recent work of Avižienis [11], [12], as a starting-off point.

II. The Model of Computation

In any meaningful discussion of time of computation it is necessary to establish a precise model first. This is especially true when discussing lower bounds. One must define a unit-time computing device. Once this has been done it is possible hopefully to make statements about bounds derived from consideration of the possible ways to interconnect such devices. Lower bounds tend to involve abstract arguments while upper bounds are constructive, i.e., one exhibits a circuit for the particular task of interest. It is, in principal, possible to establish exact results on the time to compute a given function by enumerating all circuits first of time one, then of time two, etc., until one is found that computes the function. Such methods, however, are both aesthetically displeasing and in practice impossible for computations involving large numbers.

As the unit-time computing device the papers surveyed here take modules that compute some d-valued logic function of r variables, i.e., some $f: Z_d^r \to Z_d$ where $Z_d = \{0, 1, \cdots, d-1\}$. The most commonly encountered case in practice is, of course, binary valued logic in which $d = 2$. Although it is not strictly true that all such functions take equal time for a fixed d and a fixed r this is a good model for our purposes. We are mainly interested in nets of these devices that compute functions of n variables for $n >> r$. Our model is perfectly adequate to give growth rates of time to compute such functions as n increases.

Manuscript received January 22, 1973. This work was supported in part by NSF Grant GJ-708 and in part by the U.S. Army Research Office, Durham, N.C., under Contract DA-AROD-D-31-124-71-G174.

The author is with the Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, Calif. 94720.

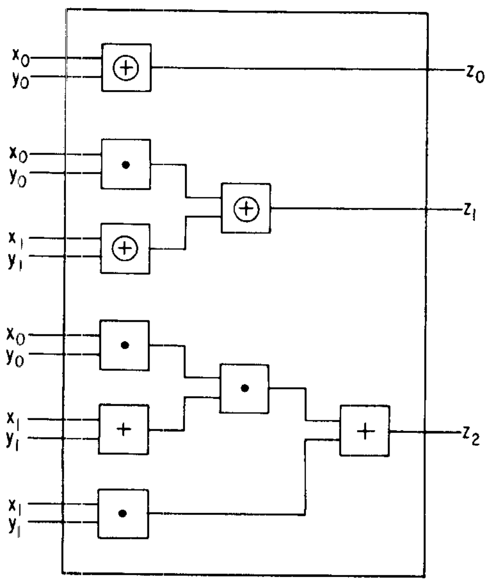


Fig. 1. A circuit for adding two-bit number.

We now define our model avoiding unnecessary formalism where ideas should be intuitively clear.

Definition 1: A (d, r) module is a device with r inputs and one splittable output which in unit time computes a fixed r-variable d-valued function $f: Z_d^r \to Z_d$. A (d, r) circuit C is a circuit composed of such modules. The inputs to C are chosen from Z_d and the computation time of C is the time necessary for the final output values of C to appear, i.e., the longest path through the circuit.

A basic fact about (d, r) circuits is that it is not possible to speed up computation of a function by the use of feedback. Therefore, the reader can consider all circuits to be feedback free in the sequel. As an example of such a circuit consider Fig. 1 which is a (2, 2) circuit to add two-bit binary numbers x_1x_0 and y_1y_0 obtaining their sum $z_2z_1z_0$ in computation time three. The reader will note that we do not have multiple fan-out in this circuit. It is also true that the computation time of a function is unaffected if we restrict ourselves to elements with nonsplittable output lines. We summarize the two preceding observations as follows.

Theorem 1: Let $f:Z_d^n \to Z_d^m$ be any n-input m-output d-valued logic function. Then if there is a (d, r) circuit that computes f in time T there is a (d, r) circuit in which no element of the circuit has a splittable output line computing f in this same time.

Proof: See Winograd [3].

Intuitively speaking, limitation to fan-out one allows us to split the function being computed into its "parallel pieces," making it easier to obtain lower bounds on computation time which nevertheless are valid for multiple fan-out device circuits. On the other hand, when constructing actual circuits, it is useful to have splittable output lines in order to avoid excessive amounts of hardware.

III. Time of Arithmetic Operations

In this section we survey results on the time of arithmetic operations, namely addition, multiplication, and division. These fall into two classes those in which d-ary representation of the numbers is allowed and those in which any nonredundant input and output codes are allowed for the circuit.

The main tool used in deriving the lower bounds that follow is given in Ofman [2] and explicitly stated in Winograd [3]. Because it is so important we give it here with proof as follows.

Theorem 2: Let $f:Z_d^n \to Z_d$ be any n-variable d-valued logical function that is nondegenerate in all of its arguments (i.e., really depends upon all n arguments). Then any (d, r) circuit that computes f has computation time at least $\lceil \log_r n \rceil$.

Proof: At each stage of the circuit the number of lines can be reduced by at most a factor of r. Hence if the computation time is T we must have $r^T \geqslant n$. Q.E.D.

As we shall see in the next section this bound is extremely poor for almost all functions. Amazingly enough, however, it is a tight bound for addition and multiplication time and a good bound for division time.

We now consider arithmetic operations of integers coded in d-ary notation for $d \geqslant 2$. Ofman [1] first considered (d, r) circuits as a model although his terminology was slightly different. He was interested in the time necessary to add n-bit binary numbers. He showed that there is a (3, 2) circuit for addition of n-bit numbers in time $0(\log n) + 1$. Winograd [4] gives a construction of a circuit to add base d numbers of n significant figures coded in d-ary notation in time $c_1 + \lceil c_2 \log_r n \rceil$ by means of a (d, r) circuit where $c_1 = 1 + \log_{\lfloor r+1/2 \rfloor} \lfloor r/2 \rfloor$ and $c_2 = \log_{\lfloor r+1/2 \rfloor} r$. The construction is valid for $r \geqslant 3$. His circuit will compute the sum either numerically or modulo d^n by ignoring the most significant bit of the answer. Since the second most significant bit is a function of all $2n$ inputs Theorem 2 applies to give a lower bound of $\lceil \log_r 2 + \log_r n \rceil$ for any $r \geqslant 2$ for this computation. Brent [9] has shown that this lower bound can be closely approached for the case $d = 2$ and $r \geqslant 2$. He shows that n-bit binary numbers can be added in time $n(1 + \epsilon)$ where ϵ is any number greater than zero if n is large enough. Marques [10] has shown that this result generalizes to bases greater than two.

Multiplication on $(2, r)$ circuits was first considered by Karatsuba and Ofman. They showed a (2, 2) circuit to multiply binary coded n-bit numbers in time $0(\log n)$ + constant. They also noted that this problem is equivalent to the addition of a column of binary numbers that represent the partial products of the multiplication. In general they showed that m n-bit numbers can be summed in $0(mn)$ + constant. Hence the multiplication result follows as a special case. To the author's knowledge no one has published results on the time necessary to multiply base d numbers on circuits when the numbers are coded in d-ary notation. However, it is easy to see that numbers of n significant figures can be multiplied in $0(\log n)$ + constant for any fixed d. Theorem 2 again shows that this is the optimal growth rate for such a computation. The best attainable multiplicative constant for the growth rate remains an open problem.

We now consider division of d-ary numbers. Note that in contrast to addition or multiplication, division has no exact answer. We thus consider the time to divide one d-ary number by another where both numbers and the answer we desire have n significant figures. Now one can see that the most significant figure of the answer depends upon all $2n$ inputs. Hence for (d, r) circuits a lower bound computation time of $\lceil \log_r n \rceil$ is valid for this problem. In contrast to previous results for addition and multiplication it is not known whether this growth rate is attainable. Note that if one can find an n-place reciprocal of an n-bit number in time $0(\log n)$ + constant it will follow that one can divide in time $0(\log n)$ + constant since multiplying the reciprocal of the divisor by the dividend yields the desired answer. Unfortunately the best known means of computing the reciprocal are all $0(\log^2 n)$. Hence the fastest known way to do the division has this growth rate with n as well. We briefly discuss one such method.

Let X and y be n-place d-ary numbers where we assume that $d^{-1} \leqslant y \leqslant 1 - d^{-n}$. Then, letting $Z = 1 - y$

$$\frac{1}{y} = 1 + Z + \cdots + Z^n$$

to n significant figures. Hence the basic idea is to invert y by computing powers of Z and adding. To compute the powers of Z we first compute by successive squaring $Z^2, Z^4, \cdots, Z^m$ where m is the least power of two which is not less than n. Then merely multiply appropriate choices of these powers. Each multiplication takes time $0(\log n)$ + constant from before. The number of multiplications that must be performed sequentially to compute Z^k is hence at most $\lceil \log_2 n \rceil$ plus the number of ones in the binary expansion of k. This latter number is at most $\lceil \log_2 n \rceil$ as well. So the powers of Z are all computable in time $0(\log^2 n)$. The summation of these powers is time $0(\log n)$ + constant. Hence division by this method takes $0(\log^2 n)$. A similar result attains using the "quadratic convergence" method based on the formula

$$\frac{1}{y} \cong (1 + Z)(1 + Z^2)(1 + Z^4) \cdots (1 + Z^{\lceil n/2 \rceil}).$$

We now briefly discuss time to perform arithmetic operations when the input and output codes to the circuit are allowed to be any 1-1 functions. Let $\phi: X \times X \to Y$ be any function with indicated domain and range for finite sets X and Y.

Definition 2: A (d, r) circuit computes $\phi: X \times X \to Y$ with nonredundant coding if there are 1-1 functions f_1 and f_2 from X into the set finite-dimensional d-ary vectors and 1-1 functions f_3 from Y into the set of finite-dimensional d-ary vectors such that if $x_1, x_2 \in X$ then if C has input $[f_1(x_1), f_2(x_2)]$ its output will be $f_3(\phi(x_1, x_2))$.

For example, in the preceding discussion, f_1, f_2, and f_3 have always been codes into d-ary representation of the numbers and X has been Z_{d^n} whereas Y has been Z_{d^n}, $Z_{d^{n+1}}$, or $Z_{d^{2n-1}}$ depending upon the function computed.

By algebraic methods involving group theory Winograd [4], [5] and Spira [6] have derived bounds on the time of circuits that compute specific such ϕ that do not depend upon the codes so long as they are 1-1. We shall not discuss the details here. Though we give a simple example to show the reader that some such ϕ are computable faster using nonstandard number representation. Consider the case $\phi: Z_{d^n} \times Z_{d^n} \to Z_{d^{2n-1}}$ where $\phi(x, y) = xy$, i.e., numerical multiplication. Then from before any (d, r) circuit computing ϕ *using d-ary representation* as its input and output codes must take time $0(\log n)$ to operate. Now consider the case in which the representation of a number N where $0 \leqslant N \leqslant d^n - 1$ contains the d-ary representation of the exponents in the prime power representation of N. Clearly no such exponent will exceed $\lceil \log_2 d^n \rceil$ so it will have at most $\lceil \log_d \lceil n \log_2 d \rceil \rceil$ places in d-ary notation. We now can multiply two numbers N_1 and N_2 so encoded by merely adding their respective exponents. But these additions can be done in parallel in time $C + 0(\log \log n)$ using previous method.

We now summarize the results of [4]-[6] and refer the reader interested in details to the original papers. First some definitions are necessary.

Definition 3: Let G be a finite Abelian group. Let $\alpha(G)$ be the maximum order of any cyclic subgroup of G. For an integer N let $\alpha(N)$ be the maximum prime power dividing N. Note that if G_N is the cyclic group of order N then $\alpha(N) = \alpha(G_N)$.

Definition 4: Let A_N be the group of positive integers less than and relatively prime to N with the group operation being multiplication modulo N. Let $\beta(N) = \alpha(A_N)$.

Definition 5: Let $Q_m = \text{lcm}\ \{1, 2, \cdots, m\}$ and $\gamma(N) = \min \{m: Q_m \geqslant N\}$. Then we summarize lower and upper bounds valid for modular and arithmetic addition and multiplication valid for (d, r) circuits having nonredundant coding. All lower bounds are valid for $r \geqslant 2$ as are all upper bounds for ϕ_1, ϕ_3, and ϕ_4. The upper bound for ϕ_2 is valid for $r \geqslant 3$. Upper bounds for ϕ_1, ϕ_3, and ϕ_4 appear in Spira [6]; all other bounds are in Winograd [4], [5]. The reader can easily see that the bounds are very tight if he notes that $\gamma(4x) \leqslant 2 + \gamma(x)$. They are presented in Fig. 2.

IV. Time to Compute Boolean Functions

In this section we discuss the time to compute Boolean functions. The results here generalize to functions of d-ary logic for $d > 2$ but we shall not worry about that here. Hence we only consider $(2, r)$ circuits.

Definition 6: Let $f: \{0, 1\}^n \to \{0, 1\}$ be a Boolean function of n arguments and let $T_r(f)$ be the minimum computation time of any $(2, r)$ circuit computing f. Let

Function	Lower Bound	Upper Bound
$\phi_1: Z_N \times Z_N \to Z_N$ $\phi_1(x,y) = x+y \bmod N$	$\lceil \log_r 2 \lceil \log_d \alpha(N) \rceil \rceil$	$1 + \lceil \log_2 2 \lceil \log_d \alpha(N) \rceil \rceil$
$\phi_2: Z_N \times Z_N \to Z_N$ $\phi_2(x,y) = xy \bmod N$	$\lceil \log_r 2 \lceil \log_d \beta(N) \rceil \rceil$	$3 + \lceil \log_{\lfloor \frac{r+1}{2} \rfloor} \lceil \frac{1}{\lfloor r/2 \rfloor} \lceil \log_d \beta(N) \rceil \rceil \rceil$
$\phi_3: Z_N \times Z_N \to Z_{2N-1}$ $\phi_3(x,y) = x+y$	$\lceil \log_r 2 \lceil \log_d \gamma(\lceil \frac{N}{2} \rceil) \rceil \rceil$	$1 + \lceil \log_r \lceil \frac{1}{\lfloor r/2 \rfloor} \lceil \log_d \gamma(2N-1) \rceil \rceil \rceil$
$\phi_4: \{1,2,\ldots,N\}^2 \to \{1,2,\ldots,N^2\}$ $\phi_4(x,y) = xy$	$\lceil \log_r 2 \lceil \log_d \gamma(\lceil \frac{\lfloor \log_2 2N \rfloor}{2} \rceil) \rceil \rceil$	$1 + \lceil \log_r \lceil \frac{1}{\lfloor r/2 \rfloor} \lceil \log_d \gamma(2 \lceil \log_2 N \rceil - 1) \rceil \rceil \rceil$

Fig. 2. Bounds for various functions.

$$T_r(n) = \max \{T_r(f): f: \{0, 1\}^n \to \{0, 1\}\}.$$

Then it follows easily from Theorem 2 that $T_r(n) \geqslant \lceil \log_2 n \rceil$.

Winograd has shown [3] by a counting argument in which the number of trees of a given depth are compared to the number of functions of n variables that $T_r(n) \geqslant (n \log_r)(1 - \epsilon)$ for any $\epsilon > 0$ and sufficiently large n. In fact almost all $f: \{0, 1\}^n \to \{0, 1\}$ have $T_r(f) \geqslant (n \log_r 2)(1 - \epsilon)$. On the other hand Spira [7] gives a construction demonstrating that given any $k \geqslant 1$ for n large enough

$$T_2(n) \geqslant n + \underbrace{\log_2 \log_2 \cdots \log_2 n}_{k \text{ iterations}}.$$

This result is based on use of Shannon's expansion [11]. It seems probable that $T_r(n)$ is proportional to n, but this is an unsettled conjecture.

Spira [8] also discusses the time necessary to compute classes of Boolean functions of special interest—namely, symmetric, monotone, and threshold functions. It is shown that symmetric or threshold functions of n variables are computable in time $0(\log n)$ on $(2, r)$ circuits whereas there are monotone functions of n variables requiring time proportional to n to compute.

Many authors have noted that the best known lower bound on $T_r(f)$ for any *specific* Boolean function of n variables is $\lceil \log_r n \rceil$ although almost all such functions require roughly time $n \log_r 2$ to compute. We note in closing that it would be extremely interesting and probably incredibly difficult to find a sequence of functions $\{f_i\}$ where $f_i: \{0, 1\}^i \to \{0, 1\}$ such that $T_r(f_i)$ grows faster than $\log i$. This is the case even though any "randomly" chosen sequence would almost surely have this property.

References

[1] Y. Ofman, "On the algorithmic complexity of discrete functions," *Dokl. Akad. Nauk. SSSR*, vol. 145, no. 1, pp. 48-51, 1962.

[2] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers with computers," *Dokl. Akad. Nauk. SSSR*, vol. 145, no. 2, p. 293, 1962.

[3] S. Winograd, "On the time required to perform computer operations," Ph.D. dissertation, Dep. Math., New York Univ., New York, N.Y.

[4] —, "On the time required to perform addition," *J. Ass. Comput. Mach.*, vol. 12, no. 2, pp. 277-285, 1965.

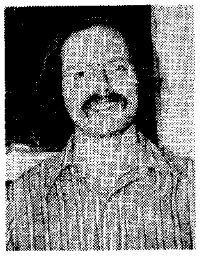
[5] —, "On the time required to perform multiplication," *J. Ass. Comput. Mach.*, vol. 14, no. 4, pp. 793-802, 1967.

[6] P. M. Spira, "The time required for group multiplication," *J. Ass. Comput. Mach.*, vol. 16, pp. 235-243, Apr. 1969.

[7] —, "On the time necessary to compute switching functions," *IEEE Trans. Comput.* (Short Notes), vol. C-20, pp. 104-105, Jan. 1971.

[8] —, "On the computation time of certain classes of Boolean func-

tions," in *Conf. Rec. Ass. Comput. Mach. Theory of Computing*, Marina del Rey, Calif., May 1969.
[9] R. Brent, "On the addition of binary numbers," *IEEE Trans. Comput.* (Short Notes), vol. C-19, pp. 758–759, Aug. 1970.
[10] I. Marques, private communication.
[11] A. Avižienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, Sept. 1961.
[12] —, "On the problem of computational time and complexity of arithmetic functions," in *Conf. Rec. Ass. Comput. Mach. Theory of Computing*, Marina del Rey, Calif., May 1969.
[13] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill, 1970.



Philip M. Spira was born in Worcester, Mass., on June 30, 1941. He received the B.S. degree from the Massachusetts Institute of Technology, Cambridge, in 1963 and the M.S. and Ph.D. degrees from Stanford University, Stanford, Calif., in 1965 and 1968, respectively, all in electrical engineering.

He is currently teaching computer science at the University of California, Berkeley. His research is mainly concerned with complexity of computation and algorithms.

The Status of Investigations into Computer Hardware Design Based on the Use of Continued Fractions

JAMES E. ROBERTSON AND KISHOR S. TRIVEDI

Abstract—**The purpose of this paper is to demonstrate that representations of numbers other than positional notation may lead to practical hardware realizations for digital calculation of classes of algorithms. This paper describes current research in the use of continued fractions. Although practicality has not been demonstrated, theoretical results are promising.**

Index Terms—**Computer arithmetic, continued fractions, continued products, hardware, quadratic equation, radix, representation of numbers, Riccati equation, selection rules.**

I. History and Motivation

THIS paper is essentially a report on research in progress. The fundamental observation is that, currently, virtually all digital hardware calculations are based on the use of positional notation; equivalently, on weighted sums of series. Other representations of numbers exist; the concern here will be with continued products and continued fractions.

The use of positional notation has been limited to addition, subtraction, multiplication, division, and, to a lesser extent, square and higher roots. It has been shown [1] that use of continued products extends the list of implementable algorithms to the logarithm, the exponential, the trigonometric and inverse trigonometric functions, as well as multiply, divide, and square root. Both time of execution and cost of hardware are reasonable with current technology; in comparison with a conventional arithmetic unit, factors of 2 to 3 (depending on the function) for both time and cost are typical. A small read-only memory fast enough to match accumulator rates is also needed. The investigation of the use of continued products was originally limited to the binary case. Higher radix techniques appear promising, and are being investigated [2]. Otherwise, emphasis will be given here to investigations into the use of continued fractions. Results to date are theoretically promising, but not yet practical in the sense of hardware implementation.

There appear to be three fundamental requirements for a proposed representation of numbers to be useful for implementation in hardware. These are the following.

Requirement 1: Conversion to conventional series form (positional notation) must be both possible and simple. Implicit here is the requirement that the set of possible results spans continuously (in the limit of infinite precision) some permissible range of values. For floating-point arithmetic, it seems sufficient to require that the ratio of the upper limit to the lower limit of the range be at least two.

Requirement 2: The set of algorithms should include algorithms that are easily soluble for the representation of numbers employed. Compatibility among algorithms, in the sense of hardware sharing, is also a desirable goal.

Requirement 3: Since most algorithms, other than multiplication, appear to require trial and error procedures in the absence of redundancy, it must be possible to devise techniques such that the selection of each of the successive coefficients is practical (cf., quotient-digit selection in division).

It should be pointed out that the use of the coefficients of a representation is ephemeral, since conversion to positional

Manuscript received August 11, 1972; revised November 21, 1972. This work was supported in part by NSF Grant GJ813, in part by the Applied Mathematics Department of Brookhaven National Laboratory, and in part by the Department of Computer Science, University of Illinois, Urbana, Ill.

The authors are with the Department of Computer Science, University of Illinois, Urbana, Ill. 61801.