ON THE PROBLEM OF COMPUTATIONAL TIME AND COMPLEXITY OF ARITHMETIC FUNCTIONS

Algirdas Avižienis
University of California, Los Angeles, California

Summary

The time and incremental complexity required to perform two-operand addition using logical circuitry are compared for nonredundant and minimally redundant encodings of the operands. The comparison is extended to multi-operand addition and two-operand multiplication.

Introduction and Review

The times required to perform the arithmetic operations of addition and multiplication using logical circuitry have been investigated by Winograd[1,2]. The model of a logical circuit C employed in these studies consists of a set of (d,r) logical elements and a rule of interconnections with designated sets of input and output lines. The (d,r) logical element has r input lines and one output line; these lines can assume one of d distinct states. The (d,r) logical element has a unit time delay; that is, the state of the output line at the time t+1 is a function of the states of the input lines at time t.

The circuit C is said to be capable of computing the function f in time T if the specified result in coded representation is observed on its output lines T time units after the arguments were applied (in coded representation) to the input lines. At the time of application of the inputs the circuit C is set to a standard internal state, and the inputs are held fixed until the time T. The method of encoding of the operands is not specified except for the requirement that the output encoding should be one-one; that is, redundancy of representation is excluded in the notion of computation. Lower bounds for the time required are then derived as functions of d, r, and of the range N of the input arguments, and computation schemes are devised which approach the bounds. Precise definitions of the (d,r) logical element, logical circuit, and computation are found in the references[1,2]; a familiarity with these references is assumed in this discussion.

An earlier study of digital arithmetic by Avizienis[3] led to the empirical development of an addition algorithm which requires a constant time T=2 to compute the sum of two operands regardless of their range. The coding of the operands and of the result employs a positional, constant radix b>2 "signed-digit" redundant form in which the allowed digit values are $\{-a,\ldots,-1,0,1,\ldots,a\}$ with $b-1 \le a \le \lceil (b+1)/2 \rceil$ where $\lceil x \rceil$ is the smallest integer not smaller than x.

The first pair of logical elements (with d = 2a+1 and r=2 according to Winograd's model[1]) accepts the operand digits x_i and y_i (indexing: n-1,...,i,...,1,0) and forms the output "transfer digit" t_{i+1} with three allowed values $\{-1,0,1\}$ and the output "interim sum" digit w_i in the range $\{-(a-1),\ldots,a-1\}$ such that

$$x_i + y_i = b\, t_{i+1} + w_i \qquad (1)$$

is satisfied (i.e., $t_{i+1} = 1$ if $x_i + y_i \ge a$; $t_{i+1} = -1$ if $x_i + y_i \le -a$, and $t_{i+1} = 0$ if $a > x_i + y_i > -a$. The second logical element accepts the inputs w_i and t_i and forms the sum digit s_i such that

$$s_i = w_i + t_i \text{ (with } a \ge s_i \ge -a). \qquad (2)$$

The coding used for the operands is preserved in the result, and the amount of redundancy is not increased.

Measures of Complexity: Two-Operand "Constant Time" Addition

The purpose of the present discussion is to consider the differences encountered in computing arithmetic functions with redundant and non-redundant encodings of the results and to establish the measure of additional or "incremental" complexity which represents the cost of holding the two-operand addition time to constant values T=3, 2, and 1. The summation of several operands and the multiplication of two operands are considered subsequently.

Minimal Redundancy with T=2

The existing set of signed-digit algorithms[3] was devised to satisfy practical design constraints of arithmetic processors. Among these, a convenient additive inverse algorithm, a unique representation of zero, and a convenient range test algorithm were needed and led to the choice of the symmetric sets $\{-a,\ldots,a\}$ for digit values, and $\{-1,0,1\}$ for transfer digit values. In general, the two-operand "constant time T=2" addition algorithm (1), (2) requires the minimum redundancy of b+2 digit values for any radix b>2. Negative digit values are avoided when the transfer digit values $\{0,1,2\}$ and the digit set $\{11,10,\ldots,1,0\}_b$ one used any b≥3; however, the transfer value set $\{-1,0,1\}$ and b+2 values around zero e.g., $\{-(b+1)/2$ to $(b+1)/2\}$ for odd b≥3 and $\{\mp(b/2)$ to $\pm(1+b/2)\}$ for even b≥4 offer the advantages of a simple range test algorithm and a unique representation of zero modulo b^n, both of which are not available with digit values $x_i \ge b$.

The least redundancy which satisfies the "T=2" addition algorithm (1), (2) consists of two additional values in each radix b digital position, giving an <u>incremental redundancy</u> ratio of (b+2)/b per digit. Considering a complete radix b positional encoding of n digits length, we observe that the conventional (non-redundant) representation has b^n unique forms while a minimally redundant representation has a total of $(b+2)^n$ forms representing $b^n + 2(b^n-1)/(b-1)$ distinct integers. The introduction of redundancy adds $(b+2)^n - b^n = b^n((1+2/b)^n -1)$ new forms, of which $2(b^n-1)/(b-1)$ represent new values, while the remaining forms provide alternate (redundant) representation for the results of additions in which the non-redundant representation requires "carry propagation" across one or more digits.

Minimal Redundancy with T=3

A modified "two-transfer" or "constant time T=3" addition algorithm has also been devised[3], in which a cascade of three logic elements form the addition circuit. This algorithm requires only b+1 values in each position of the radix b≥2 positional encoding; three values of the transfer digit are again required. The "T=3" addition algorithm has the incremental redundancy of (b+1)/b per digit. The same considerations affect the choice of digit values around zero: {-b/2 to b/2} for even b≥2 and {∓(b-1)/2 to ±(b+1)/2} for odd b≥3. The total redundancy count shows $(b+1)^n$ forms representing $b^n + (b^n-1)/(b-1)$ distinct integers. The redundancy adds

$$(b+1)^n - b^n = b^n((1+1/b)^n-1)$$

new forms, of which $(b^n-1)/(b-1)$ represent new values, and the remaining are redundant.

Minimal Complexity of Logical Elements

The originally developed "T=2" addition algorithm (1), (2) requires the cascading of two (d,r) logical elements, giving T=2 (independent of the range of the operands) with r=2 and d≥5 (for radix b=d-2≥3). The constant addition time T=2 is not available with d<5; however, it has been shown[3] that the "T=3" addition algorithm applies for any radix b≥2 with b+1 digit values, giving T=3 with r=2 and d≥3 (for radix b=d-1 ≥2).

It is evident that $s_i = f(x_i, y_i, x_{i-1}, y_{i-1})$ holds for the "T=2" algorithm (1), (2), and that $s_i = f(x_i, y_i, x_{i-1}, y_{i-1}, x_{i-2}, y_{i-2})$ holds for the "T=3" algorithm. An increase in the number r of input lines (without altering d) permits the restatement of both algorithms in terms of a single (d,r) logical element for every digit of the sum.

Two "T=1" algorithms are possible:

(1a) The "T=2" algorithm yields the "T=1" algorithm with the minimal complexity r=4 and d≥5 for the radix b=d-2.
(1b) The "T=3" algorithm yields the "T=1" algorithm with the minimal complexity r=6 and d≥3 for the radix b=d-1.

Total Complexity of the Addition Circuit

The total count of the (d,r) logical elements used in the addition circuit of two n-digit radix b operands is readily determined because of the simple structure of the circuit:
(a) The "T=1" algorithms (1a) and (1b) require n+1 (d,r) elements.
(b) The "T=2" algorithm requires 3n-1 (d,r) elements.
(c) The "T=3" algorithm requires 5n-4 (d,r) elements.
These total complexity counts offer a measure for the evaluation of the relative cost of non-redundant addition schemes.

Measures of Complexity: Other Algorithms

Additive Inverse

The additive inverse of a signed-digit operand is generated by changing the signs of all nonzero digit values. One additional digit value at the inputs of the addition circuit is required in the cases in which the digit set is not symmetrical around zero:
(a) A total of b+3 digit values (i.e., d=b+3) are required in the "T=2" algorithm and the related "T=1" algorithm (1a) when the radix b(≥3) is even;
(b) A total of b+2 digit values (i.e., d=b+2) are required in the "T=3" algorithm and the related "T=1" algorithm (1b) when the radix b(≥2) is odd.
The one-unit increase in the value of d causes corresponding changes in redundancy and logical element complexity. The other parameters (time and total complexity) remain unchanged.

Multi-Operand Addition

This section considers the time and complexity of a logical circuit which performs the addition of k>2 operands. The multi-operand addition algorithm developed for the signed-digit number systems consists of two parts:
(a) The original k operands are summed to yield the sum in the form of two encoded numbers;
(b) The two-operand addition algorithm is applied to get the result.
The time and complexity of the circuit required for part (a) is considered next; part (b) has been considered in the previous section.

The algorithm for the summation of r digits from the position i of r operands $(x^r, \ldots, x^1)$ has the form:

$$\text{for } r \le b+1, \sum_{j=1}^{r} x_i^j = bu_{i+1} + v_i \qquad (3)$$

with the limit r≤b+1 imposed to guarantee that the digits in the two results u and v have the same (redundant) set of values as the input digits the operands of $x^r, \ldots x^1$. In terms of (d,r) logical elements, two (d,r) elements (with d=b+2, or d=b+1 and r≤b+1) for every position i will reduce r encoded operands to a sum represented by two similarly encoded results in one time unit (T_r=1).

For a total of k operands, when k>r, time T>1 is required. Time T=2 is required when:

k is in the range $\max_2(k) \geq k > \max_1(k)$, with

$$\max_2(k) = r\lfloor r/2 \rfloor + (r-2\lfloor r/2 \rfloor), \text{ and } \max_1(k) = r$$

Generally, time T=j is required when:

$$\max_j(k) \geq k > \max_{j-1}(k), \text{ where}$$

$$\max_j(k) = \lfloor \max_{j-1}(k)/2 \rfloor r + (\max_{j-1}(k) - 2\lfloor \max_{j-1}(k)/2 \rfloor)$$

For the case of an even value of r, the above expression reduces to

$$\max_j(k) = 2(r/2)^j \text{ (for } r \geq 4\text{, even)}$$

The time required for a complete addition with any value of k and an even $r \geq 4$ then is given by the expression (with $r \leq b+1$):

$$T(k) = j+1 = \lceil \log_{r/2} \lceil k/2 \rceil \rceil + 1$$

where the first term j gives the time for the circuit which computes two results and the constant 1 accounts for the last 2-operand addition.

The total complexity of the circuit which precedes the last 2-operand addition and requires the time T=j depends on k,r, and the range (in digits) of the redundantly encoded input operands. For example consider a given time j and the simpler case of an even $r \geq 4$. If the number of inputs $k = \max_j(k) = 2(r/2)^j$, then the one-digit cross section of the circuit shows (beginning with the circuit's output):

$$P(j) = 1 + (r/2) + (r/2)^2 + \ldots + (r/2)^{j-1} + (r/2)^j =$$

$$((r/2)^{j+1} - 1)/((r/2) - 1)$$

pairs of (d,r) logical elements. If $k < \max_j(h)$, then

$$P = P(j-1) + D = \{((r/2)^j - 1)/(r/2) - 1\} +$$

$$\lceil (k - 2(r/2)^{j-1})/(r-2) \rceil$$

where P(j-1) is the count for the j-1 complete levels and D is the count for the incomplete first level of the circuit. Range considerations show that when the input level has n-digit operands, the length of the "u" output of (3) increases by one in each consecutive level, and the "u" output has a range of n+j digits. An upper bound for the total complexity is

$$N' = 2 \cdot (n+j) \cdot P$$

(d,r) logical elements. The actual count N is obtained by considering the exact lengths of the consecutive levels of the circuit.

The preceding discussion shows that the total complexity of circuits for the addition of k>2 operands is obtained by a straightforward counting process.

Multiplication

A two-digit product algorithm

$$a_i x_j = bp_{i+j+1} + q_{i+j} \tag{4}$$

has been developed previously[4]. This algorithm uses (d,r) logic elements with r=2 and computes the product in the form of $2n^2$ digits which must be summed in a multi-operand summation circuit with a provision to accept 2n-1 inputs in the positions n and n-1 of the 2n digits long product. The time of multiplication is then 1+T (add 2n-1 operands).

It is observed that the algorithm (4) requires only two inputs, i.e., r=2. Two new algorithms for digit multiplication have been devised which reduce the number of summands (and the time) for the multi-operand addition circuit by taking advantage of more inputs. The number of summands is reduced from 2n-1 to n when four inputs (r=4) are allowed and the algorithm then consists of forming three products according to (4):

$$a_i x_{j-2},\ a_i x_{j-1}, \text{ and } a_i x_j$$

and then forming the digit y_{i+j} of the product $a_i x$ (x is the n-digit operand) as follows:

$$y_{i+j} = (p_{i+j} + q_{i+j}) - bt_{i+j+1} + t_{i+j} \tag{5}$$

Here algorithm (1) is used to determine t_{i+j+1} from p_{i+j} and q_{i+j}, and t_{i+j} from p_{i+j-1} and q_{i+j-1}. The terms p_{i+j+1} and q_{i+j-2} need not be formed.

A further reduction of the number of summands to $\lceil (2n-1)/3 \rceil$ can be achieved by an extension of the above developed algorithm (5) when r=6 inputs are provided $(a_i, a_{i-1}, a_{i-2}, x_j, x_{j-1}, x_{j-2})$.

The total complexity of the multiplication circuit is the sum of contributions from the digit multiplication, multi-operand summation and two-operand summation circuits. The (d,r) element count of the digit multiplication circuit is $2n^2$ for algorithm (4) and (n+1)n for algorithm (5). It must be noted that the internal complexity of one (d,r) element for (5) is considerably higher (about 3-4 times) than for (4).

Concluding Observations

The primary objective of this note has been the identification of the additional complexity, (expressed in terms of redundancy and of the minimal complexity required for (d,r) logic elements), which must be accepted in order to attain two-operand addition in constant time of 3,2, and 1 units respectively. This "price" for circumventing Winograd's lower bound[1] for non-redundant encodings has been established for all three cases.

The same considerations have been applied also to the additive inverse, multi-operand addition, and two-operand multiplication using logical circuitry and redundant encodings for the operands and results. In (k>2)-operand addition the time is shown to be a function of k,r, and d, and independent of the range of the operands. The algorithm is a generalization of the "carry-save" principle of binary addition. The multiplication time is shown to be a function of the range of the operands, since it is carried out using k-operand summation, with $k=2n-1$, n, and $\lceil(2n-1)/3\rceil$ for increasing complexity of the (d,r) element which carries out the digit multiplication preceding the summation. Winograd's lower bound[2] has not been reached; however, it is interesting to note that the algorithms use the same encoding, and that the full product, its most significant half, and its least significant half are obtained simultaneously.

The total complexity (number of (d,r) elements used) also has been considered. The simple structure of the circuits (which is due to the redundancy and the use of the constant radix b) facilitates the counting of elements. A model for the measurement of total complexity for non-redundant as well as redundant encodings is being developed; the redundant case is expected to provide a reference point for complexity vs. time comparisons.

The results of this study suggest that redundancy in encoding of numbers is also an aspect of computational complexity, and that the general notion of computing an arithmetic function should encompass redundant encodings as well as Winograd's[1,2] special case of non-redundant encoding of the results.

References

1. Winograd, S., "On the time required to perform addition", Journal of the ACM, Vol. 12, no. 2, (1965), 277 -285.

2. Winograd, S., "On the time required to perform multiplication", Journal of the ACM, Vol. 14, no. 4, (1967), 793 - 802.

3. Avižienis, A., "Signed-digit number representations for fast parallel arithmetic", IRE Transactions, Vol. EC-10, no. 3. (1961), 389 - 400.

4. Avižienis, A., "Arithmetic microsystems for the the synthesis of function generators," Proc. of the IEEE, Vol. 54, no. 12 (1966), 1910 - 1919.