REDUNDANCY IN NUMBER REPRESENTATIONS AS AN ASPECT OF COMPUTATIONAL COMPLEXITY OF ARITHMETIC FUNCTIONS

Algirdas Avižienis
Computer Science Department
University of California, Los Angeles
Los Angeles, California 90024

Introduction

Recent research has led to the derivation of bounds for the time required to perform arithmetic operations by means of logical elements with a limited number of inputs [1]-[4]. The model of a (d, r) logical circuit C employed in these studies consists of a set of (d,r) logical elements and a rule of interconnection 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 the operands is constrained by 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 have been derived as functions of d, r, and of the range N of the input arguments. To obtain upper bounds, (d,r) circuits for a given operation are devised. The studies have considered radix-d positional encodings of the operands as well as the more general cases of arbitrary (but non-redundant) encodings.

An earlier study of digital arithmetic [5] led to the development of an addition algorithm which requires the constant time T = 2 to compute the sum of two operands regardless of their range. The coding of the operands and of the results employs a positional, radix b > 2 "signed digit" redundant form in which the allowed digit values are $\{-a, \ldots, -1, 0, 1, \ldots, a\}$ with $b-1 \geq a \geq \lceil (b+1)/2 \rceil$. (Here $\lceil x \rceil$ designates the smallest integer not smaller than x.) Further studies led to constant time T = 1 addition circuits as well as signed digit algorithms for multi-operand addition, multiplication, and division [6], [7].

The purpose of this paper is to consider the differences encountered in computing arthmetic functions with redundant and non-redundant encodings of the results and to establish the quantity of redundancy 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. The results are based on the signed-digit algorithms presented in [7].

Two-Operand Constant-Time Addition

The principal observation here is that the introduction of a limited amount of redundancy into the representation of the operands makes the time of two-operand addition independent of their range. The remaining constraint is that certain minimum values of d and r are required for the (d,r) element.

The least redundancy which satisfies the "T = 2" addition algorithm [5] consists of two additional values in each radix b (with $b \geq 3$) digital position, giving a redundancy 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 distinct forms while the (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 additional values above the original range, while the remaining forms provide alternate (redundant) representations for the results of additions in which the non-redundant representation would require carry propagation across one or more digits. The "T = 2" addition algorithm for the radix b = d-2 requires the cascading of two (d,r) logical elements. Constant addition time T = 2 (independent of the range of the operands) is attained with (d,r) elements having r = 2 and $d \geq 5$.

The constant addition time T = 2 is not attained with $d < 5$; however, it has been shown [5] that a "T = 3" addition algorithm exists for any radix $b \geq 2$ and b+1 digit values. Constant addition time T = 3 is attained with (d,r) elements having r = 2 and $d \geq 3$ when the radix b = d-1 is used to encode the operands. In this "constant time T = 3" addition algorithm a cascade of three logic elements forms the addition circuit. This algorithm requires only b+1 values in each position of the radix $b \geq 2$ positional encoding; three values of the transfer digit are again required. The "T = 3" addition algorithm has the redundancy of (b+1)/b per digit. 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 additional values, and the remaining are redundant.

Constant time T = 1 addition algorithms can be derived from the previous T = 2 and T = 3 algorithms by increasing the number r of input lines to the (d,r) element [7]. The minimal complexity of the (d,r) elements is as follows:

(a) The "T = 2" algorithm yields the "T = 1" algorithm with the minimal input complexity r = 4 and $d \geq 5$, given the radix b = d-2.
(b) The "T = 3" algorithm yields the "T = 1" algorithm with the minimal input complexity r = 6 and $d \geq 3$, given the radix b = d-2.

Another measure of complexity is the number of (d,r) elements required to perform the addition. The total count of the (d,r) logical elements used in the addition circuit of 2 n-digit radix b operands is readily determined because of the simple structure of the circuit:

(a) The "T = 1" algorithms 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.

Other Algorithms

This section considers some other algorithms that were developed for redundant number representations [5] -[7].

Additive Inverse. The additive inverse of a signed-digit operand is generated by changing the signs of 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) is required in the "T = 2" algorithm and the related "T = 1" algorithm, when the radix b (≥ 3) is even;
(b) A total of b+2 digit values (i.e., d = b+2) is required in the "T = 3" algorithm 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. The multi-operand algorithm developed for the signed-digit number system consists of two parts:

(a) The original k operands are summed to represent the sum as two numbers in the same representation;
(b) The two-operand addition algorithm is applied to get the final result.

The radix b algorithm for the summation of r digits [7] from the position i of r operands ($x^r, \ldots, x^1$) has the form:

$$\text{for } r \le b+1; \quad \sum_{j=1}^{r} x_i^j = bu_{i+1} + v_i$$

The limit $r \le b+1$ is imposed to guarantee that the result digits u_{i+1} and v_i have the same (redundant) set of values as the input digits $x_i^r, \ldots x_i^1$. In terms of (d,r) logical elements, two (d,r) elements with d = b+2 (or d = b+1) and $r \le b+1$ 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:

$$r\lfloor r/2 \rfloor + (r-2\lfloor r/2 \rfloor) \ge k > r$$

Generally, time T = j units is required when:

$$K(j) \ge k > K(j-1)$$

where: $K(1) = r$
$K(2) = r\lfloor r/2 \rfloor + (r-2\lfloor r/2 \rfloor)$
$K(j) = r\lfloor K(j-1)/2 \rfloor + (K(j-1) - 2\lfloor K(j-1)/2 \rfloor)$

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

$$K(j) = 2(r/2)^j$$

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

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

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.

Multiplication. A two-digit product algorithm has been developed previously [6]. This algorithm uses (d,r) logic elements with r = 2 and computes the product of two n-digit numbers in the form of $2n^2$ digits which then are 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 T (add 2n-1 operands) + 1. Other 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 [7]. 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 when r = 6 inputs are provided.

Conclusions

The main result is 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. The price for circumventing Winograd's lower bound [1] for non-redundant encodings has been established for all three cases. We have also considered the time required for K operand addition and two-operand multiplication using 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 for two n-digit operands is 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. The known 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.

Multiplication time may be reduced to the constai times 3, 2, or 1 when the operands are represented by the values of exponents in their prime power represen tation as discussed in [4]. The exponents are represented by radix b redundant encodings and multiplication is performed by adding the corresponding exponen using the constant-time algorithms discussed previous ly.

The results of this study suggest that redundanc in encoding of numbers is also an aspect of computational complexity, and that the general notion of cor puting an arithmetic function should encompass redundant encodings as well as the 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), pp. 277-285.

[2] Winograd, S., "On the Time Required to Perform Multiplication," Journal of the ACM, Vol. 14, No. 4, (1967), pp. 793-802.

[3] Brent, R., "On the Addition of Binary Numbers," IEEE Transactions on Computers, Vol. C-19, No. 8, (1970), pp. 758-759.

[4] Spira, P. M., "Computation Times of Arithmetic and Boolean Functions in (d,r) Circuits," IEEE Transactions on Computers, Vol. C-22, No. 6, (1973), pp. 552-555.

[5] Avižienis, A., "Signed-Digit Number Representations for Fast Parallel Arithmetic," IRE Transactions, Vol. EC-10, No. 3, (1961), pp. 389-400.

[6] Avižienis, A., "Arithmetic Microsystems for the Synthesis of Function Generators," Proceedings of the IEEE, Vol. 54, No. 12, (1966), pp. 1910-1919.

[7] Avižienis, A. and C. Tung, "A Universal Arithmetic Building Element (ABE) and Design Methods for Arithmetic Processors," IEEE Transactions on Computers, Vol. C-19, No. 8, (1970), pp. 733-745.

Acknowledgment

This research was supported by the National Science Foundation, Grant No. DCR72-03633 A03.