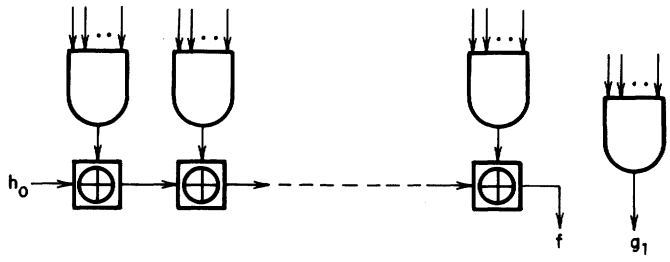

Fig. 5. Modified realization of f with an additional observable output.

$g \in F_t$, i.e., the test set will detect t s-a-faults at the AND gates of Fig. 1.

Definition: If x is a real number then $[x]$ denotes the integer part of x.

Theorem 3: If a function f is realized in complement-free form (as given in Fig. 1) then any t s-a-faults at the input or output of AND gates can be detected by applying all input vectors having 0-weight less than or equal to $[\log_2 2t]$.

Proof: If $g \in F_t$, then by Lemma 3 $|f \oplus g| \leq 2t$. By Theorem 2 there exists an input vector $\boldsymbol{E}$ such that $ow(\boldsymbol{E}) \leq [\log_2 2t]$ and $(f \oplus g)(\boldsymbol{E}) = 1$. Therefore $f(\boldsymbol{E}) \neq g(\boldsymbol{E})$ and the input vector $\boldsymbol{E}$ will detect the fault corresponding to g in the realization of f. Hence if all input vectors $\boldsymbol{X}$ with $ow(\boldsymbol{X})$ less than or equal to $[\log_2 2t]$ are applied to the network then any t or less faults in AND gates will be detected. Q.E.D.

Algorithm 1

1) First apply the test set T_1 thus detecting any number of faults in the collector row of the network.

2) Fix h_0 = constant term in the realization of RMC form f.

3) Apply all input vectors having 0-weights less than or equal to $[\log_2 2t]$ and call this set T_2.

Theorems 1 and 3 guarantee us that any t s-a-faults on all leads other than the primary input leads will be detected by the application of test inputs in $T_1 \cup T_2$. Therefore the number of tests sufficient to detect t faults at the input and output of AND gates and any number of faults in the collector row is

$$4 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{[\log_2 2t]}.$$

It can be shown that if in addition to the network of Fig. 1, an extra AND gate and one observable output are available as shown in Fig. 5, then the test set given by Reddy [1] to detect single faults will also detect any number of faults in the primary inputs. But this test set is included in $(T_1 \cup T_2)$ given in the present correspondence. Therefore if an extra AND gate and an extra observable output are provided then the test set $(T_1 \cup T_2)$ will detect t or less faults.

III. CONCLUSIONS

In this correspondence we have extended the earlier result due to Reddy [1] and obtained a fault detecting test set for multiple faults in RMC networks, realizing an n-variable logic function. We have shown that to detect t s-a-faults in RMC networks, one needs to apply a predetermined test set, independent of the function being realized, whose cardinality is

$$4 + \sum_{i=1}^{[\log_2 2t]} \binom{n}{i}.$$

The number of tests sufficient to detect all single and multiple faults can be reduced to $n + 4$ by adding extra observable outputs [6].

REFERENCES

[1] S. M. Reddy, "Easily testable realizations for logic functions," *IEEE Trans. Comput.*, vol. C-21, pp. 1183–1188, Nov. 1972.
[2] D. E. Muller, "Application of Boolean algebra to switching circuit design and error detection," *IEEE Trans. Electron. Comput.*, vol. EC-3, pp. 6–12, Sept. 1954.
[3] J. P. Hayes, "A study of digital network structure and its relation to fault diagnosis," Coordinated Sci. Lab., Univ. Illinois, Urbana, Rep. R-467, May 1970.
[4] K. K. Saluja and S. M. Reddy, "Multiple faults in Reed–Muller canonic networks," in *Proc. IEEE 13th Annu. Symp. Switching and Automata Theory*, Oct. 1972.
[5] W. H. Kantz, "Testing faults in combinational cellular logic arrays," in *Proc. 8th Annu. Symp. Switching and Automata Theory*, Oct. 1971, pp. 161–174.
[6] K. K. Saluja, "A study of combinational networks based on Reed–Muller canonic forms," Ph.D. dissertation, Dep. Elec. Eng., Univ. Iowa, Iowa City, Aug. 1973.

Arithmetic Algorithms in a Negative Base

DHARMA P. AGRAWAL

Abstract*—Algorithms are described for the basic arithmetic operations in a negative base. These algorithms are simpler, faster, and more general than those proposed by Sankar *et al.

Also, problems associated with the division and square-rooting operations, are treated in a more efficient way.

***Index Terms*—Algorithms, basic arithmetic operations, interim carry, multiple operand addition, negative base, polarization, twin carry.**

I. INTRODUCTION

Recently, Sankar *et al.* [1] have described arithmetic algorithms in a negative base. They have also given a comprehensive list of the references available in this field. Since then, Zohar [2]–[5] has published several papers on the possible use of negative base; while Kanani and O'Keefe [6] have compared the hardware requirements for the conditional-sum adders in $+2$ and -2 base. More recently, various circuits for -2 base arithmetics, have been described in the literature [7]–[9]. This correspondence presents new algorithms for the basic arithmetic operations in a general negative base. For convenience, the notations of [1] are followed here.

Let a and b be the two operands in base $-\beta$ (where β is a positive integer) given by the expressions

Manuscript received August 26, 1974; revised December 24, 1974.
The author is with the Digital Calculators Laboratory, Federal Institute of Technology, Lausanne, Switzerland.

$$a = \sum_{i=0}^{m} a_i(-\beta)^i \tag{1}$$

$$b = \sum_{j=0}^{n} b_j(-\beta)^j \tag{2}$$

where $0 \leq a_i < \beta$ and $0 \leq b_j < \beta$ with a_m and $b_n \neq 0$. Thus, a (b) is positive or negative, as m (n) is even or odd, respectively.

II. ADDITION

As mentioned by Sankar *et al.*, the negative radix addition requires "twin carriers" and these are denoted by c_i and d_i. To obtain the sum of two operands a and b, start the addition from the least-significant-digit (lsd) and use the logic for addition shown in Table I. The sum can be expressed as

$$s = \sum_{i=0}^{\max(m,n)+2} s_i(-\beta)^i \tag{3}$$

where max (m,n) means the larger value between m and n.

It may be noted that addition algorithms of Sankar *et al.* are valid for two numbers, while the one proposed here, can be easily extended for multiple operands addition by taking the value of z_i as

$$z_i = a_i + b_i + e_i + f_i + \cdots + c_i + d_i \tag{4}$$

where a, b, e, f, $\cdots$, etc., are the operands for addition. Now, the complexity of Table I (given here and in [1]) can also be compared.

III. POLARIZATION AND SUBTRACTION

The polarization operation is defined by Sankar *et al.* as transformation of b to $-b$ (or vice-versa) in a negative base. Let b be the given number expressed by (2) and $\bar{b}$ be its polarized form, then

$$\begin{aligned} b &= \sum_{j=0}^{n} -(-\beta)^{j+1} + (-\beta)^{j+1} + b_j(-\beta)^j \\ &= -\sum_{j=0}^{n} (-\beta)^{j+1} - \sum_{j=0}^{n} (\beta - b_j)(-\beta)^j \\ &= -[x] - [y] \end{aligned} \tag{5}$$

and hence

$$\bar{b} = x + y \tag{6}$$

where x and y are, respectively, first and second terms of summation on the right-hand side of (5).

Thus polarization requires generation of x and y, followed by addition. The logic for obtaining x and y is given in Table II. Case 1 of this table is also valid for $b_j = 0$. But Case 2 has been included just to avoid the propagation of carries due to redundant terms. The polarization process of [1] is more complex than given here. Moreover, the subtraction can be achieved without completing polarization. To obtain $(a - b)$, just generate x and y for b and add them to a.

IV. MULTIPLICATION

To evaluate P as a multiplication of a and b, the algorithm described for addition can be repeatedly employed. Using Table I, a can be multiplied by the jth multiplier digit b_j, giving the partial sum P^j and the "interim carries" C^j and D^j as

$$P^j = \sum_{i=0}^{m} p_i{}^j(-\beta)^{i+j} \tag{7a}$$

TABLE I
LOGIC FOR ADDITION

	$z_i = a_i + b_i + c_i + d_i$	s_i	c_{i+1}	d_{i+2}
Case 1:	$z_i < \beta$	z_i	0	0
Case 2:	$k\beta \leq z_i < (k+1)\beta$ for $k = 1 \cdots \beta$	$(z_i) \bmod \beta$	$\beta - k$	1
Case 3:	$(t\beta + k)\beta \leq z_i < (t\beta + k + 1)\beta$ for $t = 1 \cdots \beta - 2$ and $k = 1 \cdots \beta$	$(z_i) \bmod \beta$	$\beta - k$	$t + 1$

TABLE II
LOGIC FOR x AND y GENERATION

	b_j	x_{j+1}	y_j
Case 1:	$\neq 0$	1	$\beta - b_j$
Case 2:	$= 0$	0	0

$$C^j = \sum_{i=0}^{m} c_{i+1}{}^j(-\beta)^{i+j+1} \tag{7b}$$

and

$$D^j = \sum_{i=0}^{m} d_{i+2}{}^j(-\beta)^{i+j+2} \tag{7c}$$

with

$$p_i{}^j = (a_i b_j) \bmod \beta \tag{8a}$$

$$c_{i+1}{}^j = \text{value of } c_{i+1} \text{ from Table I for } z_i = a_i b_j \tag{8b}$$

$$d_{i+2}{}^j = \text{value of } d_{i+2} \text{ from Table I for } z_i = a_i b_j \tag{8c}$$

where

$$a_i b_j \equiv \text{digit } a_i \text{ multiplied by digit } b_j. \tag{9}$$

Table I can be used again for final addition of the partial sums and carries. The result can be expressed as

$$P = ab = \sum_{j=0}^{n} P^j + C^j + D^j. \tag{10}$$

In the final step of addition, C^j and D^j are considered as operands and this allows simultaneous addition of all the terms, while the algorithm of [1] gives the addition of two terms at a time.

V. DIVISION

The division operation in a negative radix poses the problem of defining the criteria that decides the end of a particular digit evaluation. Let a, b, and q be the dividend, divisor, and the quotient, respectively, all in base $-\beta$. First, $\bar{b}$ is obtained from b and a leading zero is added to a and most-significant-digits (msd's) of a and $\bar{b}$ are aligned. Now division operation is started and once the computation of the ith digit q_i is completed, start computing q_{i-1} by shifting $\bar{b}$ by one digit to the right. It may be noted that, while evaluating q_i, either of the following three conditions will exist.

Condition 1:

$$\begin{array}{ccccccc} 0 & a_{i+n-1} & a_{i+n-2} & \cdots & a_i \cdots a_0 \\ \bar{b}_n & \bar{b}_{n-1} & \bar{b}_{n-2} & \cdots & \bar{b}_0 \end{array} \tag{11a}$$

i.e., the nonzero msd of a (or the partial remainder) is placed one or more digits right of the nonzero msd of $\bar{b}$.

Condition 2:

$$\begin{array}{cccc} a_{i+n} & a_{i+n-1} & \cdots & a_i \cdots a_0 \\ \bar{b}_n & \bar{b}_{n-1} & \cdots & \bar{b}_0 \end{array} \tag{11b}$$

i.e., the nonzero msd's are aligned.

Condition 3:

$$\begin{array}{ccccc} a_{i+n+1} & a_{i+n} & a_{i+n-1} & \cdots & a_i \cdots a_0 \\ & \bar{b}_n & \bar{b}_{n-1} & \cdots & \bar{b}_0 \end{array} \tag{11c}$$

i.e., the nonzero msd of $\bar{b}$ is placed one digit right of a or the remainder.

When Condition 3 exists, the polarized addition is always possible, while when Condition 2 exists, the polarized addition is never possible. When Condition 1 exists, the decision is slightly complicated. First obtain P_{cri} as

$$P_{cri} = \bar{b}\left(1 - \frac{1}{\beta} + \frac{1}{\beta^2} - \frac{1}{\beta^3} \cdots \frac{1}{\beta^n}\right) \tag{12}$$

and take a_{i+n-1} and msd of P_{cri} for comparison and follow the following steps (for detailed proof, see [10]).

Step 1: Compare the digits under consideration. If the digit of P_{cri} is smaller, the polarized addition is possible; if larger, polarized addition is not possible; and if equal, go to Step 2.

Step 2: If all the digits have already been compared, go to Step 5; otherwise go to Step 3.

Step 3: Compare next digits. If the digit of P_{cri} is larger, the polarized addition is possible; if smaller, polarized addition is not possible; and if equal, go to Step 4.

Step 4: If the comparison of all the digits have already been completed, go to Step 5; otherwise take next significant digits for comparison and go to Step 1.

Step 5: Take $q_i = 0$, $q_{i-1} = 0$ and onwards alternate digits as $(\beta - 1)$ and 0. Final remainder will be zero.

The results of division thus obtained, will have a unique representation in $-\beta$ base and the possibilities of initial overflow and $q_j = +\beta$ (as happens in nonstoring division algorithms proposed by Sankar *et al.* [1]) have been completely eliminated and Conditions 2, 3, and Step 5 behaves as a deterministic algorithm. It is worth mentioning that, once P_{cri} has been evaluated for any specific "i," same value, with proper right shift, can be utilized for onwards computation of lower significant digits. But value of q_{i-2} and onward digits of Step 5, will no longer hold good.

VI. SQUARE-ROOT

The difference of this process from division lies in the fact that the subtrahend changes in successive step of square-rooting. Here, an algorithm which generates polarized addend (rather than subtrahend) is given. The method starts with the selection of $(1\ \beta{-}1)$ as the two digits of the first polarized addend and allow its addition. Onward addend can be evaluated with the help of the following steps.

Step A: Add $(1\ \beta{-}2)$ to the previous addend, with the lsd's aligned. Take the result as a new addend and go to Step B.

Step B: If possible, add the polarized addend and go to Step A; otherwise go to Step C.

Step C: Add $(2\ \beta{-}1)$ to the previous addend, with 2 aligned with the lsd of the previous addend, shift the result one digit to right and take the result as a new addend and go to Step B. (In the case of base -2, take the equivalent value 1 1 0 1.)

The square-rooting process is similar to the division operation and therefore details are not given here. For this operation P_{cri} has to be taken as present value of polarized addend plus (1 1), with the msd of the second aligned with the lsd of first. Also, Step 5 has to be taken as $q_i = 1$, $q_{i-1} = (\beta - 1)$ and onwards digits and remainder as zero. (For detailed proof, refer to [10].)

The possibility of getting either positive or negative root, has already been mentioned by Sankar *et al.* Square-rooting will give positive root only when $(m + 1)/2$ is odd (m is the number of digits of a). If it is even and it is still desired to obtain the positive root directly, the position of the initial polarized addend has to be shifted through two digits, either to the left or to the right and the condition for left shift is given below.

Condition for Left Shift: Take $w = \beta{-}1\ 0\ \beta{-}1\ 0\ \cdots$ with total number of digits equal to $(m - 1)/2$; obtain the square of w and use it as P_{cri} and if comparison allows polarized addition in Step 1 or 3, allow the left shift; otherwise right shift. When $m = 1$, always do left-shift, while evaluating negative root.

VII. CONCLUSION

Faster and more general algorithms for arithmetic operations in a negative radix, have been described in this correspondence and an attempt has been made to solve the problems associated with division and square-rooting operation.

REFERENCES

[1] P. V. Sankar, S. Chakrabarti, and E. V. Krishnamurthy, "Arithmetic algorithms in a negative base," *IEEE Trans. Comput.*, vol. C-22, pp. 120–125, Feb. 1973.

[2] S. Zohar, "New hardware realizations of nonrecursive digital filters," *IEEE Trans. Comput.*, vol. C-22, pp. 328–338, Apr. 1973.

[3] ——, "The counting recursive digital filter," *IEEE Trans. Comput.*, vol. C-22, pp. 338–347, Apr. 1973.

[4] ——, "Fast hardware Fourier transformation through counting," *IEEE Trans. Comput.*, vol. C-22, pp. 433–441, May 1973.

[5] ——, "A/D conversion for radix (-2)," *IEEE Trans. Comput.*, vol. C-22, pp. 698–701, July 1973.

[6] D. V. Kanani and K. H. O'Keefe, "A note on conditional-sum addition for base -2 systems," *IEEE Trans. Comput.* (Corresp.), vol. C-22, p. 626, June 1973.

[7] L. S. Houselander, "Cellular-array negabinary multiplier," *Electron. Lett.*, vol. 10, pp. 168–169, May 1974.

[8] D. P. Agrawal, "Negabinary carry-look-ahead adder and fast multiplier," *Electron. Lett.*, vol. 10, pp. 312–313, July 1974.

[9] ——, "Negabinary complex number multiplier," *Electron. Lett.*, vol. 10, pp. 502–503, Nov. 1974.

[10] ——, "Some aspects of fast arithmetic techniques," Ph.D. dissertation, Fed. Inst. Technol., Lausanne, Switzerland, 1975.

An Algorithm for Finding Nearest Neighbors

JEROME H. FRIEDMAN, FOREST BASKETT, AND LEONARD J. SHUSTEK

***Abstract*—An algorithm that finds the k nearest neighbors of a point, from a sample of size N in a d-dimensional space, with an expected number of distance calculations**

$$E[n_d] \lesssim \pi^{-1/2}[kd\Gamma(d/2)]^{1/d}(2N)^{1-(1/d)}$$

is described, its properties examined, and the validity of the estimate verified with simulated data.

Manuscript received June 24, 1974; revised February 7, 1975. This work was supported in part by the U. S. Atomic Energy Commission under Contract AT(043)515.

J. H. Friedman is with Stanford Linear Accelerator Center, Stanford, Calif. 94305.

F. Baskett and L. J. Shustek are with the Computer Science Department, Stanford University, Stanford, Calif.