REPRINT OF

INFORMATION PROCESSING 1962

PROCEEDINGS OF IFIP CONGRESS 62

INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING

MUNICH

AUGUST 27 TO SEPTEMBER 1, 1962

NORTH-HOLLAND PUBLISHING COMPANY, AMSTERDAM

ON A FLEXIBLE IMPLEMENTATION OF DIGITAL COMPUTER ARITHMETIC

A. AVIŽIENIS

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Cal., USA.

INTRODUCTION

Most implementations of computer arithmetic deal with numerical values which are represented as machine words of a fixed length. If the number of significant digits is less than the word length, non-significant zero digits are used to fill the remaining positions of the word; if it is greater than the word length, multiple-precision operations are programmed. This paper presents a method for a flexible implementation of arithmetical operations in a digital computer. Arithmetical operations are defined for any length (number of significant digits) of the operands. The result of the operation is generated sequentially, with the most significant digits appearing first. An arithmetical operation may be concluded when the required number of significant digits of the result has been generated. Furthermore, the time required by an addition in a parallel adder is constant for any number of digits. The length of the adder may be altered without changing the rules of implementation or the addition time; this permits a reorganization of the adder in case of a failure, or a reapportionment of adder lengths to suit a particular problem in a parallel system with several adders.

2. SIGNED-DIGIT NUMBER REPRESENTATIONS

The class of *signed-digit* number representations is uniquely suitable for a flexible implementation of arithmetic. Signed-digit representations are positional number representations with a constant integer radix $r > 3$, in which the allowed values of the individual digits z_i are a sequence of q ($r + 2 \leq q \leq 2r - 1$) integers: $\{-a, \ldots, -1, 0, 1, \ldots, a\}$. The range of the maximum magnitude of a digit a is $\frac{1}{2}(r_e + 1) \leq a \leq r_e - 1$ for odd integers, $r_e > 3$; and $\frac{1}{2}r_e + 1 \leq a \leq r_e - 1$ for even integers, $r_e > 4$. Both positive and negative digit values are allowed. The individual digits each contain sign information and therefore, a special sign digit is not required. For example, only one set of allowed values exists for radix $r = 3$ (values $-2, -1, 0, 1, 2$), and for $r = 4$ (values $-3, -2, -1, 0, 1, 2, 3$); for radix $r = 10$ there are four sets, from 13 values (-6 to 6) to 19 values (-9 to 9).

Signed-digit representations are *redundant*, that is, each radix-r digit z_i may assume more than r different values. In a *conventional* (non-redundant) representation only r values of a digit ($0, 1, \ldots, r - 1$) are allowed. Signed-digit numbers have *minimal redundancy* (the digits assume $r_e + 2$ or $r_e + 3$ values) when

$$a = a_{\min} = \tfrac{1}{2}(r_e + 1), \quad \text{or} \quad a = a_{\min} = \tfrac{1}{2}r_e + 1;$$

and they have *maximal redundancy* (the digits assume $2r_e - 1$ or $2r_e - 1$ values) when:

$a = a_{\max} = r_e - 1$, or $a = a_{\max} = r_e - 1$. The most important properties of signed-digit representations are the following:

1. The *algebraic value* of a number is: $Z = \sum_{i=-n}^{m} z_i r^{-i}$.
2. Algebraic value of $Z = 0$ if, and only if all $z_i = 0$.
3. *Sign* of the algebraic value Z is the sign of the most significant non-zero digit.
4. To form the representation of $-Z$, change the sign of every non-zero digit z_i.
5. The addition and subtraction of digits may be *totally-parallel*: $s_i = f(z_i, y_i, z_{i+1}, y_{i+1})$ for all positions i, where s_i are digits in the representation of the sum or difference $S = Z \pm Y$.

There are no carry-propagation chains in totally-parallel addition (or subtraction), that is, any digit of the sum is a function of only two adjacent digits of the operands. Subtraction is performed as a change of sign followed by an addition. The time of one addition is independent of the length of the operands and is equal to the addition time of two digits. The procedure and block diagram of a totally-parallel adder are shown in fig. 1. Here t_i is the *transfer digit* and may assume the values 1, 0, and -1; w_i is the *interim sum digit* and may assume the sequence of values:

$$\{-w_{\max}, \ldots, -1, 0, 1, \ldots, w_{\max}\}.$$

For representations of minimal redundancy ($a = a_{\min}$), we have $w_{\max} = a_{\min} - 1$; in all other cases ($a > a_{\min}$) the value of $w_{\max}$ is restricted to the range

$$a_{\min} - 1 \leq w_{\max} \leq a - 1.$$

To perform conversions between conventional (sign and magnitude) and signed-digit representations, we may treat each digit x_i of a conventional number as the sum of two digits of signed-digit numbers and apply the rules of totally-parallel addition ($x_i = w_i + rt_{i-1}$; $z_i = w_i + t_i$). Conversely, we may consider a signed-digit number to be the sum of a positive and a negative number in conventional representations and obtain a single conventional number by adding them in a conventional adder.

The allowed range of the algebraic values

$$Z = \sum_{i=0}^{m} z_i r^{-i}$$

for fixed-point numbers is required to cover the range $1 > Z > -1$. An easily implemented method of over-

ADDITION (two levels):

(A) $z_i + y_i = rt_{i-1} + w_i$

$t_{i-1} = 0$ if $|z_i + y_i| \leq w_{max}$

$t_{i-1} = 1$ if $z_i + y_i > w_{max}$

$t_{i-1} = -1$ if $z_i + y_i < -w_{max}$

(B) $s_i = w_i + t_i$

SUBTRACTION: change signs of all $y_i \neq 0$ and add

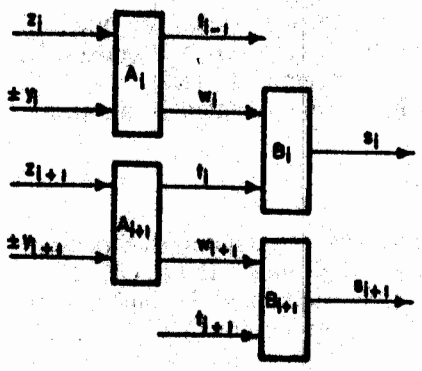

Fig. 1. Totally-parallel addition and subtraction.

flow detection is also necessary. The two most significant digits z_0 and z_1 are inspected to detect overflow.

Positive overflow occurs when:

$$z_0 > 1; \text{ or } z_0 = 1 \text{ and } z_1 > 0. \quad (6)$$

Negative overflow occurs when:

$$z_0 < -1; \text{ or } z_0 = -1 \text{ and } z_1 < 0. \quad (7)$$

where z_m is the least significant digit (the number is $m+1$ digits long). In the range of values of $|Z|$ between these limits, overflow may or may not be indicated, because signed-digit representations are redundant and some algebraic values may be represented in more than one way. For example, given radix $r = 10$ and $a = 6$ (minimal redundancy), we shall have a

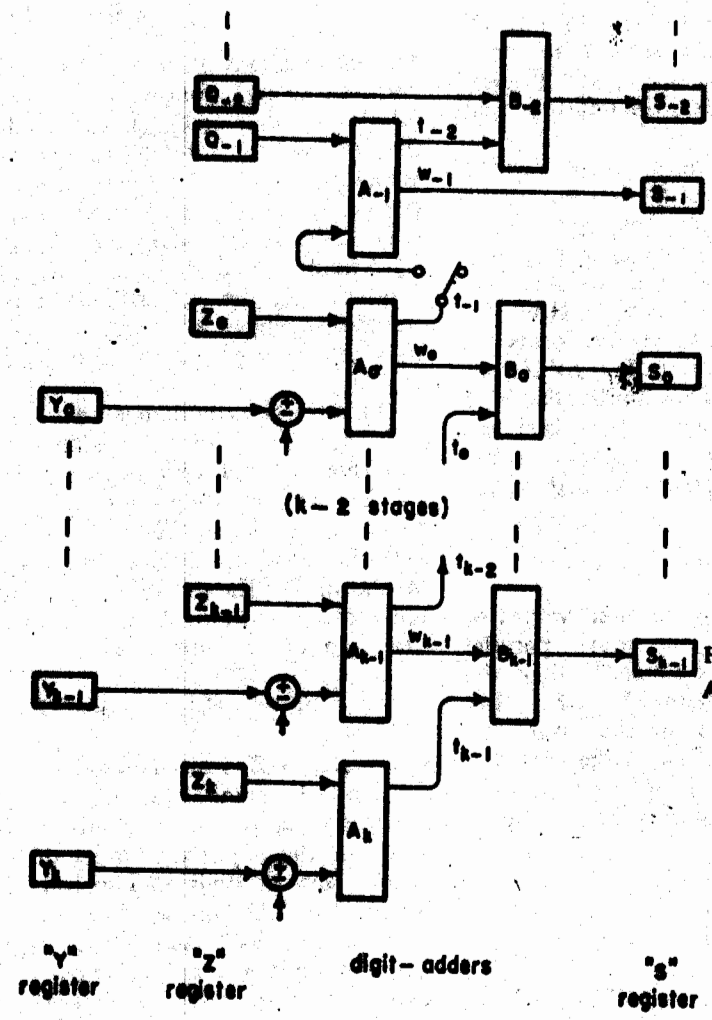

Fig. 2. Adder for k digits.

For these overflow detection rules, no overflow will be indicated for the algebraic values of

$$|Z| < 1 + (1/r) - (a/(r-1))(r^{-1} - r^{-m}),$$

and overflow will always be indicated for

$$|Z| > 1 + (a/(r-1))(r^{-1} - r^{-m}),$$

certain overflow indication for $|Z| > (32/30)$ and no overflow indication for $|Z| < (31/30)$.

In the following sections we employ minimal redundancy representations, which require least storage for the values of one digit and have the least magnitude of a ($a = a_{min}$). The familiar radix 10, with 13 digit

values (— 6 to 6) is the preferred choice for practical applications. A detailed description of signed-digit representations and of conventional arithmetic with signed-digit numbers has been presented in an earlier paper [1]).

3. Implementation of Arithmetic

First we consider the *addition* (including subtraction) of two n digit-long signed-digit numbers ($n > 1$), using an adder of $k > 1$ digits length, and beginning with the k most significant (highest weighted) digits of the operands. The addition consists of a sequence of α steps, where α is the least integer such that $\alpha > (n/k)$. During each step, k digits of the sum are generated; the duration of one step is the unit time interval of the arithmetic unit, independent of k. The most significant digits of the sum are generated first and overflow is immediately detectable. The detection or correction of errors can be independently implemented for each digit-adder circuit. Fig. 2 shows the diagram of an adder for k digits. The adder consists of k digit-adder circuits (A_0 and B_0 to A_{k-1} and B_{k-1}), and registers "Z", "Y" and "S". The two most significant digits which are to be added during the next step (as inputs to A_0) are held in storage locations Z_k and Y_k and serve as inputs to circuit A_k to form t_{k-1}. The circuits A_{-1} and B_{-2} and associated storage locations are used in multiplication only.

The adder is also employed to implement multiplication and division. In these operations, it is necessary to add a multiple $\pm cY$ of the addend $\pm Y$ to the augend Z. Multiples $\pm cy_i (0 \leqslant c \leqslant a_{\min})$ of the digits $\pm y_i$ are to be added to the digits z_i. If the adder of fig. 2 is employed, $\pm Y$ is added c times to Z. However, it is also possible to devise a circuit which forms the multiples $\pm cy_i$ at once. The addition of $\pm cY$ is then completed in one addition time. One digit-adder with a multiple-forming circuit M_i for the digit $\pm y_i$ is shown in fig. 3. Two outputs are generated by the circuit M_i: $c(\pm y_i) = v_i + rh_{i-1}$. These outputs are added to the digit z_i in the circuits A_i* and B_i

$$z_i + v_i + h_i = rt_{i-1} + w_i, \text{ and } w_i + t_i = s_i$$

according to the rules of fig. 1. The allowed values of v_i and h_i are chosen to satisfy the totally-parallel addition requirements. The only difference when the digit-adders of fig. 3 are used to form $Z \pm cY$ is that now $s_i = f(z_i, cy_i, z_{i+1}, cy_{i+1}, cy_{i+2})$ and the input Y_{k+1} must also be available to form t_{k-1}; during the next step the digit from Y_{k+1} is shifted into Y_1. The outputs h_{-1} and t_{-1} are employed only to form product digits.

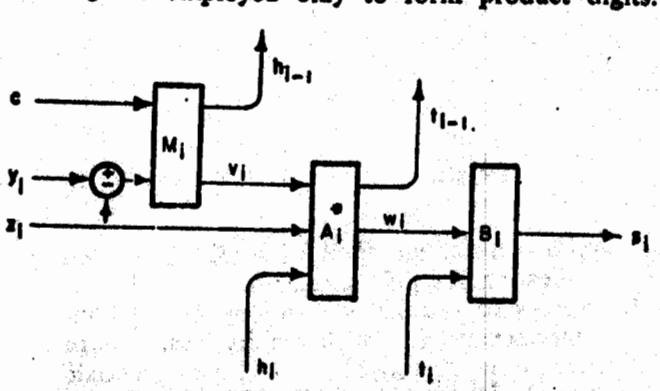

Fig. 3. Digit-adder with a multiple-forming circuit.

Multiplication is implemented as a sequence of additions and shifts, similar to conventional multiplication. The principal difference is that one digit of the product (beginning with the most significant digit) is generated during each step. Consequently, multiplication may be terminated when the required number of product digits has been generated. One step in the multiplication of radix r numbers, X and Y, is described by the algorithm:

$$P_j = P_{j-1} + r^{-j} Y x_j, \quad j = 0, 1, 2, \ldots, m, \qquad (8)$$

where Y is the multiplicand, x_0 to x_m are the digits of the multiplier X, P_j ($j = 0, 1, \ldots, m-1$) are partial products, $P_{-1} = 0$, and $P_m = XY$ is the product.

When the adder of fig. 2 is used to perform multiplication, the "Y" register holds the multiplicand, while the partial products are accumulated in the "Z" register and its extension to the left, i.e., the "Q" register. The multiplier X is held in a shift register "X" and its digits are sensed sequentially, starting with the most significant digit x_0. The product digits are immediately available when (8) is implemented as

$$P_j = r(P_{j-1} + Yx_j);$$

that is, one step of multiplication is performed as the addition of x_jY followed by a left shift of the sum (all S_i into Z_{i-1} or Q_{i-1}). The product digits move into the "Q" register; circuits A_{-1} and B_{-2} add the incoming transfers t_{-1} and h_{-1} during the formation of the next partial product (inputs t_{-1} and h_{-1} are connected to A_{-1} only when y_0 is held in Y_0). After the $(m+1)$th step, "Q" holds the $(m+1)$ most significant digits of the product P_m, the last of which is in Q_{-1}; the less significant digits are held in "Z"; that is, we have $P_m = r^{m+1}XY$ with respect to the radix point location of Y.

Division is also implemented as a sequence of additions and shifts. The Robertson method of division [2]) is most conveniently applicable to signed-digit numbers. In this method the representation of the quotient may be redundant, and then the selection of quotient digit values may be based on the comparison of the approximate magnitudes of the divisor and the dividend, or a partial remainder. Given the radix r, dividend Z, and divisor Y, one step of division is described by the algorithm:

$$R_j = r(R_{j-1} - Yq_j), \quad j = 0, 1, \ldots, m \qquad (9)$$

where $R_{-1} = Z$ is the dividend, R_j ($j = 0, 1, \ldots, m-1$) are partial remainders, $R_m = r^{m+1}(Z - QY)$ is the remainder, q_0 to q_m are digits of the quotient Q, and $m+1$ is the required number of quotient digits.

One step of division consists of the choice of q_j, followed by addition of $-Yq_j$ and a left shift of the sum. During each step of division the value of the quotient digit q_j must be chosen to be such that the partial remainder R_j is within the same allowed range as the previous partial remainder R_{j-1}. This range is a function of the magnitude of the divisor Y and of the allowed values of quotient digits q_j; that is, the requirement $|R_j| < rK \cdot |Y|$ must be satisfied, where K has the range $\frac{1}{2} < K < |q_j|_{\max}/(r-1)$. We allow the quotient digits q_j to assume one of the values $\{-b, \ldots, -1, 0, 1, \ldots, b\}$, where $b = \frac{1}{2}r_0$ for even radices $r_0 > 4$, and $b = \frac{1}{2}(r_0 + 1)$ for odd radices $r_0 > 3$. Since the re-

presentation of the quotient is redundant, the value of q_j is selected on the basis of a comparison between

$$|Y_n|' = |\sum_{i=0}^{3} y_i r^{-i}| \quad \text{and} \quad |R_{j-1}|' = |\sum_{i=0}^{3} z_i r^{-i}|.$$

Here the digits $y_i (i = 0, 1, 2, 3)$ are the first four digits of the normalized divisor: $|Y_n| > 1/r$; and the digits z_i $(i = 0, 1, 2, 3)$ are the first four digits of the partial remainder R_{j-1}.

When the adder of fig. 2 is employed to perform division, the magnitude of R_{j-1} (held in "Z") is diminished by repeated addition of $\pm Y_n$ (held in "Y") until the condition $|R_{j-1} \pm gY_n|' < \frac{1}{2}|Y_n|'$ is detected by a comparison circuit; then $(R_{j-1} \pm gY_n)$ is shifted left. If g additions are performed, $|q_j| = g$; the sign of q_j is chosen to be such that the signs of R_{j-1} and of Yq_j agree. If the digit-adders of fig. 3 are available, one step of division may be completed in one addition-time. We employ a total of b comparison circuits, in which the test $|R_{j-1}|' < (g + \frac{1}{2})|Y_n|'$ is performed for the values of $g = 0, 1, \ldots, b - 1$. The least value of g which satisfies the test gives $|q_j| = g$; if no value of g satisfies the test, $|q_j| = b$. The sign of q_j is chosen as above.

4. Significant Digit Operations

In this section we consider an implementation of arithmetic in which the operations are performed with significant digits only. The propagation of error during a sequence of calculations may be more readily estimated in this case; furthermore, the least time will be taken by an arithmetical operation if a completion signal occurs as soon as the required number of significant digits of a sum, product or quotient has been generated.

To determine the number of significant digits in the result, the number of significant digits in the input operands must be known. This information may be incorporated into the representations of numbers by a special digit φ, designated as the *space-zero*. The non-significant digit positions at the right (low-significance) ends of the input operands and partial results are identified by this special digit φ. The relative locations of the φ digits in all operands supply the required information to conclude an arithmetical operation.

There exist two methods of applying the digit φ. In the first method, addition rules

$$\left.\begin{array}{ll} \text{a)} & z_i \pm \varphi = rt_{i-1} + \varphi \\ \text{b)} & t_i + \varphi = \varphi \end{array}\right\} \text{for all values of } z_i \text{ and } t_i$$
$$\text{(i.e., } w_i = \varphi \text{ and } t_{i-1} = 0, \pm 1) \quad (10)$$

apply to the new digit φ, and the sum of two signed-digit numbers is rounded off by truncation to the length of the number with fewer significant digits. If every allowed value of the digit z_i occurs with the same probability, then the average error which is introduced by truncation is zero, and the round-off is without bias. The end of an addition is signalled by the detection of the digit φ as an input to the adder.

The digit φ may also be interpreted as a zero value:

$$z_i \pm \varphi = z_i \pm 0 \text{ for all values of } z_i \neq \varphi;\ \varphi + \varphi = \varphi. \quad (11)$$

In this case round-off is avoided and the sum has the length of the number with more significant digits. End of addition is signalled by the detection of the digits φ as both inputs to one position of the adder. Rule (11) is applied in the implementation of multiplication and division.

The preceding development leads to a floating-point arithmetic in which the numbers are normalized and the φ digits indicate the precision of the fractional parts. The proposed implementation follows the rules of significant digit arithmetic which have been developed by Metropolis and Ashenhurst [3,4]).

In a *floating-point* (radix r) number system, a numerical value Z^* is represented as a *fractional part* Z and an integral *exponent* E such that the pair (Z, E) represents $Z^* = Zr^E$. We require the fractional part to be *normalized*, since leading zeros are not significant. The fractional part Z is in normal form when 2 conditions are satisfied: a) rules (6) and (7) indicate no overflow; b) either $|z_0| = 1$ or overflow is indicated when z_1 is substituted for z_0, and z_2 is substituted for z_1 in (6) and (7). According to this definition, the range of the normalized fractional part Z is:

$$r^{-1} + r^{-2}[1 - (a/(r-1))] + r^{-m}(a/(r-1)) < |Z| < 1 + r^{-1}(a/(r-1)) - r^{-m}(a/(r-1)). \quad (12)$$

To define the normal form of $Z = 0$, we add the rule that $z_2 = \varphi$ is an *overflow indication* when substituted for z_1 in the test for normalization. Consequently, all fractional parts have at least 2 significant digits (z_0 and z_1), and the fractional part $Z = 0$ is uniquely represented by $z_0 = 0$, $z_1 = 0$ and $z_2 = \varphi$.

The execution of floating-point arithmetic now consists of three parts. The *provisional exponent* E_3' of the result Z_3^* must be calculated. In the addition process $(Z_3^* = Z_1^* + Z_2^*)$, we have $E_3' = \max(E_1, E_2)$; the exponent difference $d = |E_2 - E_1|$ must also be obtained. In multiplication,

$$Z_3^* = Z_1^* Z_2^* \text{ and we have } E_3' = E_1 + E_2,$$

while in division,

$$Z_3^* = Z_1^*/Z_2^* \text{ and we have } E_3' = E_1 - E_2.$$

The fractional part Z_3 of the result Z_3^* is calculated according to the rules of the preceding section, with completion signals provided by the φ digits in Z_1 and Z_2 (fractional parts of the operands). In an addition, the most significant d digits of Z_1 (where $E_1 \geq E_2$) are added to zero, until digit z_d of Z_1 is aligned with the most significant digit z_0 of Z_2; then addition proceeds as described above, and rule (10) applies to the φ digits. In multiplication and division, n_3 (the number of significant digits in Z_3) is determined as $n_3 = \min(n_1, n_2)$, where n_1 and n_2 are the numbers of significant digits in Z_1 and Z_2, respectively. To determine n_3, it is sufficient to scan the operands serially (from the most significant end) during the execution of (8) or (9). The j-th digits are inspected before the execution of the j-th step; if one digit has the value φ, n_3 steps have been completed. The additions of (8) or (9) are performed according to rule (11) to avoid a truncation of the R_j or P_j. Since the dividend $Z = R_{-1}$ is altered by the first step of (9), it should be stored elsewhere for scanning, possibly in the multiplier register "X".

The result Z_3 may become denormalized, that is, the first three digits may indicate overflow or underflow (leading zeros). Then Z_3 must be normalized and the *true exponent* E_3 obtained by a correction of E_3'. One significant digit may be gained or several may be lost during addition; appropriate corrections are also necessary during multiplication and division if denormalization occurs [4]).

5. Conclusion

A signed-digit number of any length consists of a positionally ordered and weighted array of complete one-digit-long numbers, carrying their own sign information. As a result, procedures exist for operations with variable length numbers and for significant digit arithmetic. The most significant digits of a result become available before the entire arithmetical operation is completed. Consequently, it is now possible to consider the organization of arithmetical nets, in which the digits of intermediate results are immediately processed further, and a numerical computation proceeds in a parallel asynchronous manner, at the maximum speed of the components (digit-adders, shift registers and control circuits) of the net. The organization of such arithmetical nets and the logical design of digit-adders and control circuits present interesting problems for further research.

6. Acknowledgment

This paper presents results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract NASw-6, sponsored by the National Aeronautics and Space Administration. A detailed study of flexible implementation will be published as a Technical Report of the Jet Propulsion Laboratory.

7. References

1) Avižienis, A.: *Signed-Digit Number Representations for Fast Parallel Arithmetic.* IRE Trans. on Electronic Computers EC-10 (1961) 389–400.
2) Robertson, J. E.: *A New Class of Digital Division Methods.* IRE Trans. on Electronic Computers EC-7 (1958) 218–222.
3) Metropolis, N. and R. L. Ashenhurst: *Significant Digit Computer Arithmetic.* IRE Trans. on Electronic Computers EC-7 (1958) 265–267.
4) Ashenhurst, R. L. and N. Metropolis: *Unnormalized Floating Point Arithmetic.* J. of the Assoc. for Comput. Machinery 6 (1959) 415–428.

ABSTRACTS

A method is presented for the implementation of arithmetical operations in a digital computer. Addition, subtraction, multiplication and division are performed with operands of variable multiple precision with respect to the length of the adder in an arithmetic unit. The result of the operation is generated sequentially, with the most significant digits appearing first.

A *signed-digit* number representation is employed to represent the operands and results. It is possible to add (or subtract) signed-digit numbers so that any digit of the sum (or difference) is the function of only two adjacent digits of the input operands. Consequently, carry-propagation chains of variable length do not occur during addition or subtraction.

When a special digit is employed to indicate the first non-significant position of each operand, an arithmetical operation can be terminated when all required significant digits of the result have been generated. It is expected that this implementation of arithmetic will be applicable in asynchronous computers, and in complex parallel systems, in which several arithmetic units are required to perform calculations asynchronously at their own maximum speeds.

В данном докладе описывается метод выполнения арифметических операций в цифровой вычислительной машине. Сложение, вычитание, умножение и деление производятся с исходными числами произвольной длины (т.е. с различной переменной точностью) по отношению к длине сумматора в арифметическом блоке. Результат операции (сумма, разность, произведение или частное) выдается последовательно, начиная с старших значащих разрядов. Для представления исходных данных и результатов вычисления используется специальное представление знаковых разрядов. Изображения знаковых разрядов представляют позиционные числа с постоянным основанием системы счисления $r \geq 3$, в которых каждый разряд принимает значения в пределах множества q ($r + 2 \leq q \leq 2r - 1$) целых величин (положительных, равных нулю или отрицательных). Поскольку $q > r$, представления знаковых разрядов обладают избыточностью. Можно складывать или вычитать числа знаковых разрядов так, что любой разряд суммы (или разности) будет являться функцией только двух смежных (находящихся в одной позиции) разрядов входных исходных чисел. Следовательно, во время сложения или вычитания не возникает цепей переноса и становится возможной гибкая работа арифметического устройства.

Так как число значащих разрядов для каждого исходного числа известно, арифметическая операция завершается, когда получено потребное число значащих разрядов результата вычисления. Для того, чтобы указать на незначащие позиции исходного числа, используется специальный разряд. Предполагается, что использование такой самосинхронизации арифметических действий по значащими разрядам будет пригодно для асинхронных вычислительных машин (asynchronous computer) и сложных вычислительных систем параллельного действия, в которых несколько арифметических блоков должны были бы асинхронно производить вычисления при максимальных скоростях этих блоков.

Cette communication présente une méthode pour la mise en œuvre d'opérations arithmétiques dans un calculateur digital. L'addition et la soustraction, la multiplication et la division s'effectuent sur des opérandes de longeur variable. Le résultat d'une opération est élaboré de façon séquentielle, les chiffres les plus significatifs étant traités en premier.

On utilise, pour représenter les opérandes et les résultats, la technique du nombre à *chiffre-signe*. Celle-ci est une représentation où le rang d'un chiffre est significatif, comprenant une base entière constante $r \geqslant 3$, dans laquelle chaque chiffre prend une valeur parmi un ensemble de q valeurs entières. Etant donné que $q > r$, les représentations sont redondantes Il est possible d'additionner (ou de soustraire) des nombres à chiffre-signe de telle sorte que chaque chiffre de la somme (ou de la différence) n'est fonction que de deux chiffres adjacents des opérandes. Ces propriétés suppriment l'existence des chaines de longeur variable de propagation de report en addition ou en soustraction.

Une opération arithmétique peut être terminée dès que le nombre prévu de chiffres-signes du résultat a été engendré, ou un signe spécial peut être employé pour indiquer la première position non significative. On peut espérer que la mise en œuvre de cette arithmétique sera applicable aux calculateurs asynchrones et dans les ensembles complexes parallèles de calcul dans lesquels plusieurs unités arithmétiques exécuteraient des calculs indépendamment, chacun d'eux à sa vitesse maximum.

Es wird eine Methode für die Realisierung der arithmetischen Grundoperation auf einem Digitalrechner beschrieben. Addition, Subtraktion, Multiplikation und Division werden mit Operanden variabler Vielfach-Länge im Verhältnis zu der Länge des Addierwerkes in der Maschine ausgeführt. Das Ergebnis der Operation (Summe, Differenz, Produkt oder Quotient) wird sequentiell gebildet, wobei die führenden Ziffern zuerst erscheinen.

Zur Darstellung der Operanden und Resultate werden *vorzeichenbehaftete* Ziffern verwendet. Zahlen in einer solchen Darstellung lassen sich so addieren oder subtrahieren, dass jede Ziffer der Summe bzw. Differenz nur von zwei benachbarten Ziffern der Operanden abhängt. Ein Übertrag über beliebig viele Ziffern hinweg kann also bei der Addition oder Subtraktion nicht auftreten.

Wenn man mit Hilfe einer Spezialziffer die erste nichtgeltende Ziffer eines jeden Operanden kennzeichnet, dann kann man eine arithmetische Operation beenden, sobald die verlangte Anzahl von geltenden Ziffern des Resultates gebildet ist. Es wird erwartet, dass diese Arithmetik für Asynchronrechner und für komplexe Parallel-Anlagen anwendbar sein wird, die sonst für die Erzielung maximaler Rechengeschwindigkeit mehrere Rechenwerke benötigen würden.

Esta comunicación presenta un metódo para la implementación de operaciones aritméticas en calculadoras digitales. La adición, substración, multiplicación y división se realizan con operandos de longitud variable, y el resultado de la operación se obtiene en forma secuencial y aparecen en primer lugar los dígitos más significativos.

Para la representación de los operandos y los resultados se utilizan números de *dígito-señalado*. Esta es una representación de tipo posicional con una raíz entera constante $r \geqslant 3$, en la que cada dígito toma un valor de un sistema de q valores enteros. Como $q > r$ la representación es redundante. Es posible sumar (o restar) números de dígito de la suma (o de la diferencia) en función solamente de dos dígitos adyacentes de los operandos de entrada.

Consecuentemente, no pueden presentarse cadenas de arrastre de longitud variable durante la adición o substración.

Se puede terminar la operación aritmética cuando se ha generado el número requerido de dígitos significativos del resultado. Alternativamente, se puede indicar la primera posición no significativa. Se espera que esta implementación será aplicable a las calculadoras asincrónicas y en sistemas complejos de cálculo de tipo paralelo, en los que se necesitan varias unidades aritméticas para realizar cálculos en forma asincrónica, cada una a su velocidad máxima de operación.

DISCUSSION

H. Takahasi *(Japan)*. In any redundant number representation, the determination of the sign requires partial or complete propagation of carries. Do you claim that this is not necessary in your system?

A. Avižienis *(USA)*. In the class of signed-digit representations the sign of the algebraic value of a number is the same as the sign of the most significant non-zero digit. For example, in the number of $0\bar{5}\bar{3}4$ the sign is +, since the left-most non-zero digit is $+5$.

There exist two methods of determining the sign:

1. The numbers are held in their normalized form, that is, without leading zero digits. The sign is always available in the left-most position of the number.
2. When normalization is not desirable, the number is scanned serially from the left until the first non-zero digit is encountered.

K. L. Johnson *(Germany)*. A property of signed-digit representations is that, to form $-Z$, you change the sign of every non-zero digit of Z. A practical simplification would ensue if the restriction to non-zero digits were removed. Has this been found to be so in practical trials on a computer? Also, is your method restricted to fixed point systems?

A. Avižienis *(USA)*. Yes, the use of both $+0$ and -0 as digit values is quite practical when sufficient storage is available. For instance, when 4 binary storage elements are employed to store one decimal digit with 13 values (-6 to $+6$), one storage element can be allocated to the storage of the sign, while the next three hold the values 0, 1, .., 6 and φ.

The use of floating-point arithmetic is quite desirable. One reason is that the sign of the algebraic value of a number is readily available when the numbers are normalized. Furthermore, normalization eliminates leading zeros and thus yields the fastest completion of significant digit operations.

C. J. Tunis *(USA)*. Do you use the redundancy of this

representation to obtain error detection or correction of the arithmetic processes carried out?

A. Avižienis *(USA)*. The original purpose of redundancy in this class of number representations was to limit the propagation of carries during addition. When there are no carry-propagation chains, the problem of error detection and correction is considerably simplified, since error detection or correction can be implemented independently for each digit-adder circuit. An investigation of this problem is now in progress.