a request on behalf of port 3 or port 4. Consequently, there is always an overlap where one of the secondary arbiters is making a decision while the other is in use. The primary arbiter is guaranteed to always have a request waiting to be serviced. A wide class of priority networks can be simulated exactly or approximated by trees of two input arbiters. Fig. 12(b) has almost the same priority behavior as the network in Fig. 10. Thus, the two input arbiter (Fig. 11) is a basic building block for more complicated arbiters.

VI. The Basic Arbiter

In the design as presented, each port contained "buffer" logic. Referring to Fig. 5, we will now consider driving the port logic from the port (R_j', A_j') and will compare its operation with what was described for the complete arbiter.

An up transition on R_j' commands the arbiter to establish a connection from port$_j$ to the resource, use the resource, and finally, signal completion of the resource with an up transition on A_j'. The connection through the arbiter remains established, locking the resource from use by other ports until port$_j$ sends a reset command $\overline{R_j'}$. When the buffer circuit is used, the completion of the resource automatically initiates the dissolving of the connection without waiting for an $\overline{R_j}$. When $\overline{R_j}$ does come, it is simply absorbed.

Thus, the (R_j', A_j') port presents a "basic arbiter" which is useful in the construction of large, multiple-server arbiters. However, for typical applications dealing with current computer systems the automatic dismissing of the resource (i.e., the inclusion of the buffer circuit) is desirable.

Acknowledgment

The author wishes to thank Prof. J. B. Dennis, Prof. S. Patil, J. A. McKenzie, and Miss A. Rubin, who all contributed to this paper.

References

[1] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, 275–303, Mar. and Apr. 1954.

[2] ——, "Design and use of hazard-free switching networks," *J. Ass. Comput. Mach.*, vol. 4, pp. 47–72, Jan. 1957.

[3] S. H. Unger, "Hazards and delays in asynchronous sequential switching circuits," *IRE Trans. Circuit Theory*, vol. CT-6, pp. 12–25, Mar. 1959.

[4] G. A. Maley and J. Earle, *The Logical Design of Transistor Digital Computers*. Englewood Cliffs, N. J.: Prentice-Hall, 1963.

[5] R. McNaughton, "Badly timed elements and well timed nets," Moore School of Engineering, Philadelphia, Pa., Rep. 65-02, June 10, 1964.

[6] E. J. McCluskey, *Introduction to the Theory of Switching Circuits*. New York: McGraw-Hill, 1965.

[7] R. E. Miller, *Switching Theory*, vol. 2. New York: Wiley, 1965.

[8] D. B. Armstrong, A. D. Friedman, and P. R. Menon, "Design of asynchronous circuits assuming unbounded gate delays," *IEEE Trans. Comput.*, vol. C-18, pp. 1110–1120, Dec. 1969.

[9] T. H. Bredt and E. J. McCluskey, "A model for parallel computer systems," Stanford Digital Syst. Lab., Stanford Univ., Stanford, Calif., Rep. 5, Apr. 1970.

A Suggestion for a High-Speed Parallel Binary Divider

RENATO STEFANELLI

Abstract—A family of four procedures to compute the inverse $1/X$ of a given binary number X normalized between 0.5 and 1 is described. The quotient is obtained in redundant binary form, i.e., in a base 2 code in which digits can assume any positive or negative integer value. All methods here described can be implemented by combinatorial networks; the dividers realized in this way are very fast because all carry propagations take place at the same time.

The first procedure is based on a simple principle but leads to a very expensive and impractical realization; the other procedures are obtained from the first by successive modifications that simplify the divider structure. The fourth procedure, the more economical one, can be realized by a two-dimensional cellular array of parallel counters.

Finally a converter from redundant binary to binary code is described.

Index Terms—Binary divider, binary redundant code, carry propagation, cellular array, combinatorial network, high-speed dividers, parallel adders, parallel counters, parallel dividers, parallel subtractors.

Manuscript received October 1, 1968; revised September 28, 1970. This work was supported in part by the Comitato di Ingegneria, Comitato Nazionale delle Ricerche (Italian Patent 844218).

The author is with the Istituto di Elettrotecnica ed Elettronica del Politecnico di Milano, Milan, Italy.

I. Introduction

It is well known that, in the solution of numerical problems, the speed of a digital computer usually is limited by the time required to compute a product and a quotient.

Computational speed could be substantially increased if multiplier and divider circuits, with computation times comparable to that of a parallel adder were available.

The first problem, i.e., the one of designing high-speed parallel multipliers, has already been studied and some very interesting circuits have been described [1], [2]; one of them [3], which was realized with medium-speed threshold gates (delay times of about 50 ns), obtains the product of two 12-bit numbers in 550 ns.

On the contrary, many difficulties have been encountered in implementing high-speed dividers; the aim of this paper is to contribute to the solution of this problem.

The long execution time required by a division in conventional computers is due to the fact that division is normally performed by iterated subtractions; this method performs one comparison and possibly one subtraction for every bit of the quotient; both operations require a carry propagation and can be performed only when the carry propagation of the preceding operation has been completed.

Most of the procedures proposed so far to speed up the execution time of a division, such as methods based on the conventional approach, that obtain more than one bit at the same time by means of a more complex comparison [4]; the methods based on the power-series decomposition of the reciprocal of the divisor [2], [5]; the iterative [6]; and the divide-and-correct methods [7], [8], obtain a reduction of the execution time by reducing, as far as possible, the number of elementary operations required; each elementary operation—addition, subtraction, comparison, and product—still requires a carry propagation.

A rapid divider, involving very few carry propagations, can be realized using circuits that calculate approximate logarithms [9] but the accuracy thus obtained is not very high.

In this paper a family of methods for performing division will be presented, that require only a minimum number of nonsimultaneous carry propagations and can be implemented by highly parallel circuits.

II. Introductory Remarks on the Proposed Methods

In the following paragraphs some methods will be described to compute the inverse $y=1/x$ of a given number x; the methods proposed here can be easily extended to the computation of the ratio of two numbers z/x.

In all these methods the numerator is approximated, in binary form, by the number $0.11 \cdots 1$; the divisor x must be expressed in binary form and normalized between 0.5 and 1, i.e.,

$$0.1000 \cdots \leq x \leq 0.1111 \cdots .$$

The network that computes $1/x$, shown in Fig. 1, can be considered as consisting of two parts. *The first part*, the divider, gives the inverse of x in a "redundant binary" form that will be defined. *The second part* converts this representation of the inverse to the binary form.

In the binary system, a number $x<1$ is represented by the ordered set of digits $x_1, x_2, \cdots, x_n$, each being either 0 or 1, such that

$$x = x_1 2^{-1} + x_2 2^{-2} + \cdots + x_n 2^{-n}. \tag{1}$$

We define the redundant binary form of a number x as an ordered set $x_1 \cdots x_n$ of digits such that (1) still holds, but where each x_i can assume any integer value, whether positive or negative.[1],[2] For example, the decimal number 0.78125,

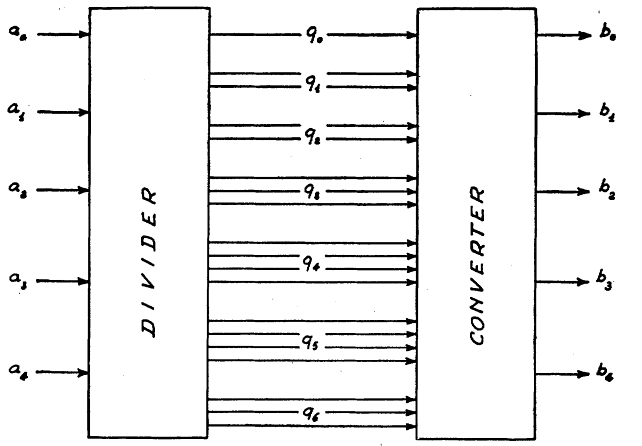

Fig. 1. Decomposition of a binary divider into a circuit that obtains the quotient $1/A$ in redundant binary code and a redundant binary to binary code converter.

coded in binary form as 0.110010, can be coded in redundant binary form as $0.102\bar{1}14$ or as $0.2\bar{2}3\bar{5}62$.[3] In the described dividers, each digit q_i of the quotient, as shown in Fig. 1, is represented by a certain number of binary signals in a suitable code.

In the next paragraph, a basic procedure is discussed that allows one to compute the inverse of a given number in redundant binary form; a possible implementation of this procedure is also proposed.

In the following paragraphs, some algorithms derived from the basic one, but resulting in simpler and faster structures, will be presented; finally, the converter from a redundant binary code to a binary code will be discussed.

III. The Basic Procedure

Let $A=[a_1, a_2, \cdots, a_n]$ be the number to be inverted and $Q=[q_0, q_1, q_2 \cdots q_m]$ be the quotient; the product AQ, as said previously, is $0.1111 \cdots$.

It is well known that in order to compute the product of the two numbers A and Q, some summands must be added together.

These summands, which appear as the rows of the matrix reproduced in Table I, are obtained by shifting right the multiplicand A by $1, 2, 3 \cdots i$ places and multiplying it by the corresponding digits $q_0, q_1, q_2, \cdots, q_{i-1}$ of the multiplier Q.

Generally, if the sum is computed column by column, some carries are obtained. In this way the sum of the digits in a column also depends on the columns to the right and, in turn, it affects the sums of the columns on its left.

If a binary redundant code is used to represent quotient Q, it is possible to choose the values of digits q_i in such a way that the sum of the digits appearing in a column be exactly 1. In this way no further carry propagation between columns is needed.

The values of digits q_i can be easily computed starting

[1] Redundant binary codes have already been used [4], [10] but only values 1, 0, −1 were allowed for each digit.

[2] This representation is "binary" since it uses base 2; it is "redundant" because more than one set of digits $x_1 \cdots x_n$ may correspond to a given number x.

[3] In this paper an overbar denotes negative digits.

[4] If A is normalized between 0.5 and 1, then $1 \leq Q \leq 2$ and the first digit q_0 of Q has exponent 2^0.

TABLE I

THE SUM OF THE ROWS IS THE PRODUCT $AQ=0\cdot 11111\cdots$

				a_1q_3	$\cdots$	$a_{n-3}q_3$	$a_{n-2}q_3$	$a_{n-1}q_3$	a_nq_3	
			a_1q_2	a_2q_2	$\cdots$	$a_{n-2}q_2$	$a_{n-1}q_2$	a_nq_2		
		a_1q_1	a_2q_1	a_3q_1	$\cdots$	$a_{n-1}q_1$	a_nq_1			
	a_1q_0	a_2q_0	a_3q_0	a_4q_0	$\cdots$	a_nq_0				
$0\cdot$	1	1	1	1	$\cdots$	1	1	1	1	$\cdots$

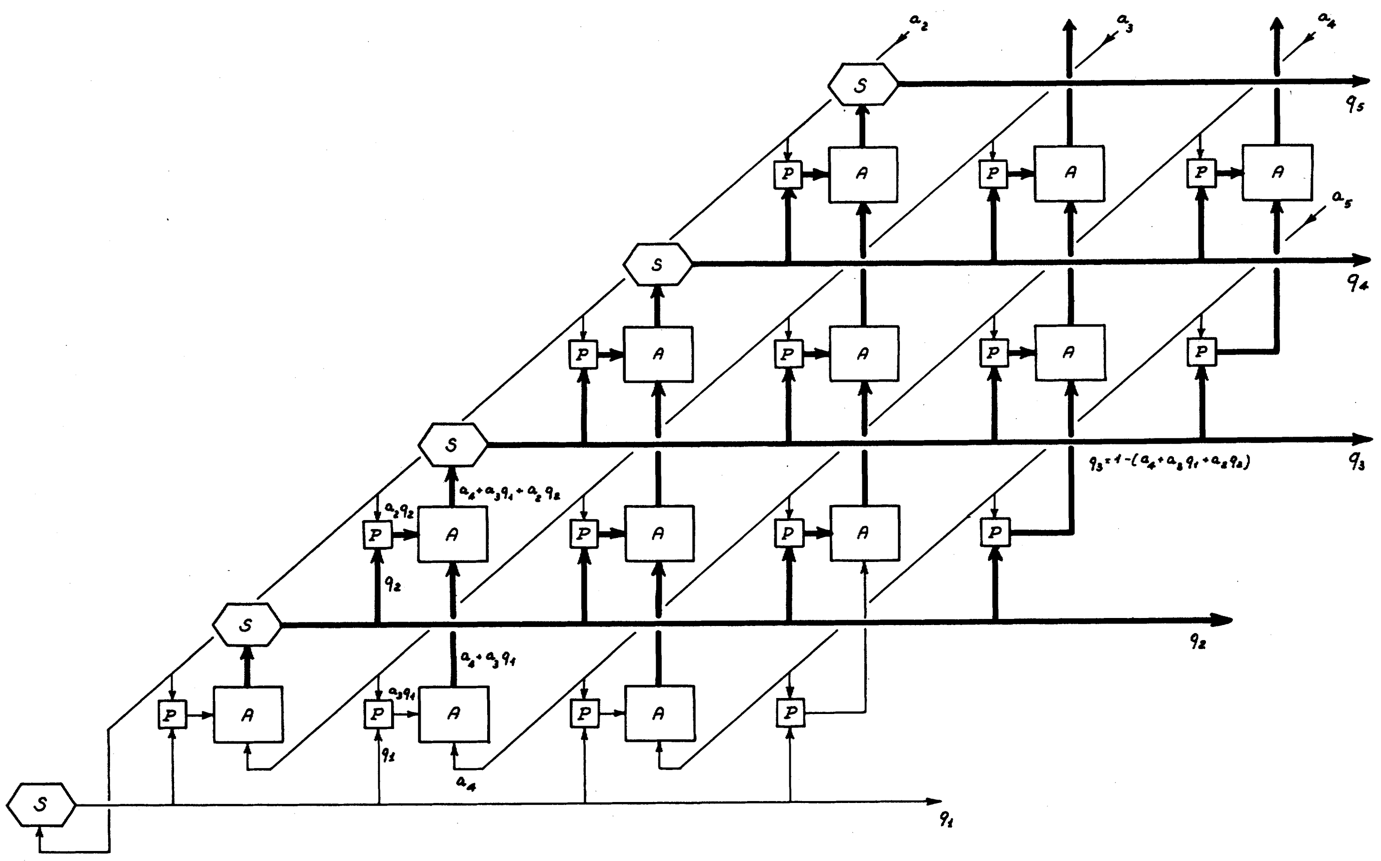

Fig. 2. Scheme of a divider implementing the basic procedure. Thin lines represent binary signals, thick lines represent redundant binary signals. The divider is formed by a regular array of multiplying circuits (P), parallel adders (A), and parallel subtractors (S).

from the first column on the left of Table I; since $a_1=1(0.5 \leq A<1)$ we obtain

$$q_0 = 1. \tag{2}$$

Considering the second and the other columns and considering also (2), we have

$$q_1 = 1 - a_2$$
$$q_2 = 1 - (a_2q_1 + a_3)$$
$$q_3 = 1 - (a_2q_2 + a_3q_1 + a_4)$$
$$\vdots$$
$$q_i = 1 - (a_2q_{i-1} + a_3q_{i-2} + \cdots + a_iq_1 + a_{i+1}). \tag{2'}$$

As an example, the inverse of the number $A=0.1101$ is computed. By (2) and (2′) we obtain

$$q_0 = 1 \quad q_1 = 0 \quad q_2 = 1 \quad q_3 = \bar{1} \quad q_4 = 2$$
$$q_5 = \bar{2} \quad q_6 = 4 \quad q_7 = \bar{5} \quad q_8 = 8 \quad q_9 = \overline{11} \cdots.$$

If we take into account only the first six digits ($Q=1.01\bar{1}2\bar{2}$) quotient Q expressed in binary form, is equal to 1.0011; product AQ is 0.11110111 and approximates number 1 by the first four bits after the binary point. Evidently, if we consider more digits in the representation of Q, we obtain a better accuracy.

A. The Divider Structure

Fig. 2 shows the scheme of the divider whose operating principle is based on the already mentioned algorithm; binary signals are represented by thin lines while those in redundant binary code, as for example digits q_i, are represented by thick lines; the divider is a regular array with three kinds of elementary circuits.

1) Multiplying Circuits (Denoted by P in Fig. 2): They are used to compute the products $a_i q_j$; each digit q_i can be represented by a set of binary signals while a_i is a single bit; each multiplying circuit therefore consists of a number of AND gates equal to the number of bits representing digit q_i.

2) Two-Input Parallel Adders (Denoted by A in Fig. 2).

TABLE II

Maximum and Minimum Values of the Output of the Adders in Fig. 2 for a 12-Bit Divider

																						386
																						*
																					10'761	17'388
																					-248	-385
																				204	249	6627
																				-6660	*	-366
																			4122	6661	248	6627
																			-155	-203	-4100	-267
																		120	156	2539	142	2505
																		-2551	-4121	-112	-4100	-155
																	1579	2552	156	2539	100	2505
																	-80	-119	-1570	-104	-1549	-107
																66	81	973	110	960	92	926
																-977	-1578	-114	-1570	-78	-1549	-424
															605	978	77	973	66	960	295	926
															-44	-65	-601	-62	-593	-56	-572	-316
														37	45	373	53	368	43	355	207	321
														-374	-604	-65	-601	-54	-593	-203	-572	-407
													232	375	41	373	41	368	142	355	260	321
													-28	-36	-230	-46	-227	-30	-219	-134	-198	-130
												20	29	143	30	141	29	136	88	123	82	89
												-143	-231	-36	-230	-28	-227	-97	-219	-164	-198	-137
											89	144	29	143	19	141	68	136	105	123	86	89
											-15	-19	-88	-19	-87	-18	-84	-57	-76	-51	-55	-15
										12	16	55	18	54	17	52	37	47	32	34	11	
										-55	-88	-15	-88	-14	-87	-46	-84	-66	-76	-53	-55	
									34	56	16	55	15	54	32	52	42	47	33	34		
									-8	-11	-33	-13	-33	-14	-32	-24	-29	-20	-21	-8		
								6	9	22	12	21	10	20	16	18	13	13	6			
								-21	-33	-11	-33	-11	-33	-22	-32	-27	-29	-21	-21			
							13	22	8	22	7	21	15	20	17	18	13	13				
							-5	-5	-12	-6	-12	-9	-12	-10	-11	-8	-8	-5				
						3	6	9	7	9	6	8	7	7	6	5	3					
						-8	-12	-5	-12	-6	-12	-10	-12	-11	-11	-8	-8					
					5	9	6	9	5	9	7	8	7	7	5	5						
					-2	-2	-4	-5	-5	-5	-5	-5	-5	-5	-4	-2						
				2	3	4	4	5	5	6	5	4	3	2	2							
				-3	-4	-2	-4	-3	-4	-4	-4	-4	-4	-3	-3							
			2	4	3	4	4	5	5	5	4	3	2	2								
			-1	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-1								
		1	2	3	3	4	4	4	4	4	3	2	1									
		-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1									
	1	2	2	3	3	3	3	3	3	3	2	1										
	0	0	0	0	0	0	0	0	0	0	0	0										
1	1	2	2	2	2	2	2	2	2	2	1											
0	0	0	0	0	0	0	0	0	0	0	0											

Note: An asterisk denotes a value lower than −10.000. The upper entries in the columns are the maximum and minimum values of the quotient digits q_i.

3) *Parallel Subtractors (Denoted by S in Fig. 2):* They perform the operation $1-x$ where x is the only input of the circuit. As shown in Fig. 2, column i implements the ith relation in (2′).

The number of stages of each parallel adder depends on the maximum and minimum values of the adder outputs; these values, which were computed by simulating a 12-bit divider on a digital computer, are listed in Table II. The upper entries in the columns in Table II are the maximum and minimum values of the subtractor outputs, i.e., of digits q_i.

The adders in the rightmost columns in Fig. 2 have the largest number of stages. In order to limit the cost of the divider, the minimum number of digits q_i required to obtain a quotient with as many bits as the divider must be computed.

By simulating the algorithm on a computer, the relative truncation error was calculated; the results are reported in Fig. 3, in which the truncation error is plotted versus the number N_1 of redundant binary digits.

The same figure shows, for comparison purposes, the

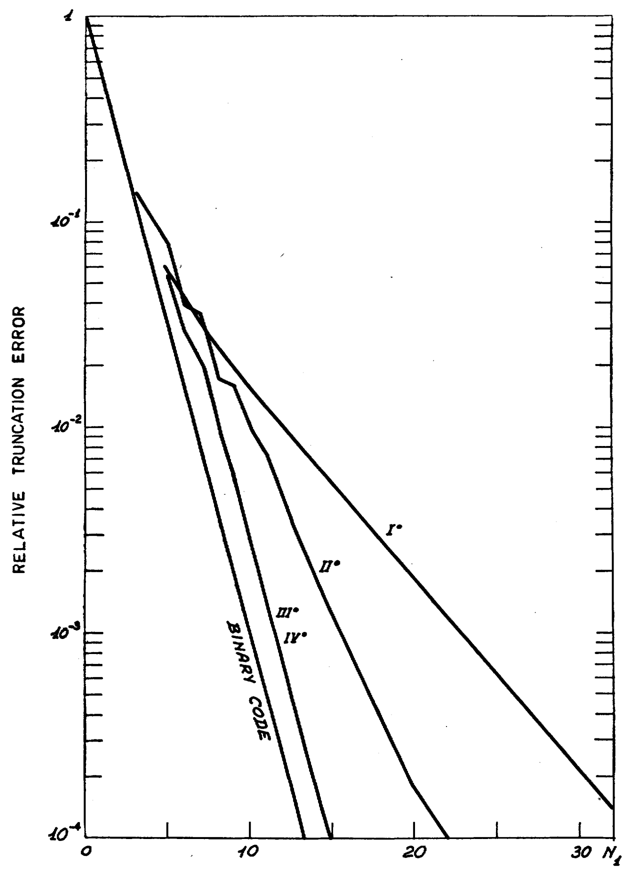

Fig. 3. Relative truncation error, related to the number N_1 of digits q_i, obtained with basic, second, third, and fourth procedures. For comparison, the truncation error obtainable by a binary coded quotient is reported.

truncation error obtained with N_1 binary digits. Fig. 4 reports the number N_1 of redundant binary digits that are needed in order to obtain N binary digits. The following approximate relation, which is valid for $N > 5$, can be obtained from Fig. 4

$$N_1 = 3.2\,(N - 3).$$

B. Delay Time Evaluation

The scheme of the divider, shown in Fig. 2, shows that all the signals propagate from the bottom to the top of each column and at the same time in every column; in fact, as soon as digit q_i is available, all the products $a_j q_i$ are obtained at the same time and, also at the same time, they are applied at the input of all the adders that are on the ith row.

The slower signal chain is the one on the left border of the divider structure. As the digit q_i is computed, in order to obtain q_{i+1}, the signal must go through one multiplying circuit, one adder, and one subtractor.

Both adders and subtractors have carry propagation, but an operation can be started before the completion of the preceding one since both operations require carry propagation in the same direction from the least to the most significant bit.

Therefore, the total delay time required to obtain a number N of digits q_i is

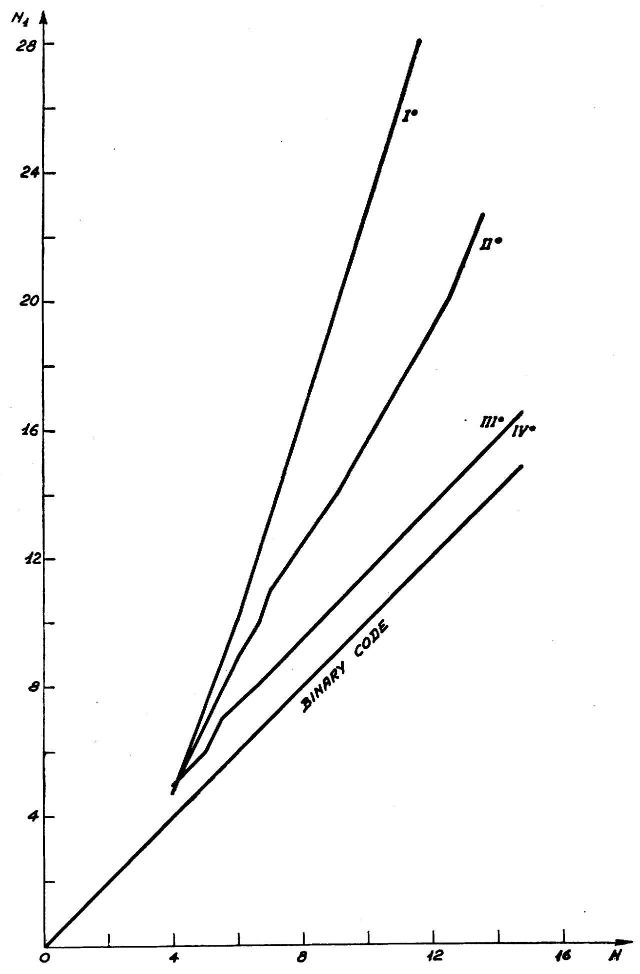

Fig. 4. Number N_1 of binary redundant digits q_i that are needed to compute an N-bit quotient with the basic, second, third, and fourth procedures.

$$\tau = N\tau_a + N\tau_s + N\tau_r + \tau_0$$

where τ_a, τ_s, and τ_r are, respectively, the delay time between input and output of *one* stage of an adder, subtractor and multiplying circuit, and τ_0 is the carry propagation time of the adder or subtractor with the highest number of stages.

IV. Introductory Remarks on the Other Methods

The basic procedure that was described in the preceding paragraph leads to a redundant binary quotient whose digits q_i may have very high values. As shown in the example reported in Section III, something like an instability phenomenon takes place. If digit q_i is high, generally the next digit q_{i+1} will take a larger value with opposite sign.

This fact leads to a very expensive implementation of the divider for these main reasons.

1) A large number of adders is required because, in order to obtain an N-bit quotient, almost $3N$ redundant digits q_i must be computed.

2) Each adder requires a large number of stages.

3) A large and expensive redundant binary to binary converter is required.

For these reasons, the algorithm proposed in the preceding paragraph has been modified as will be shown in what follows.

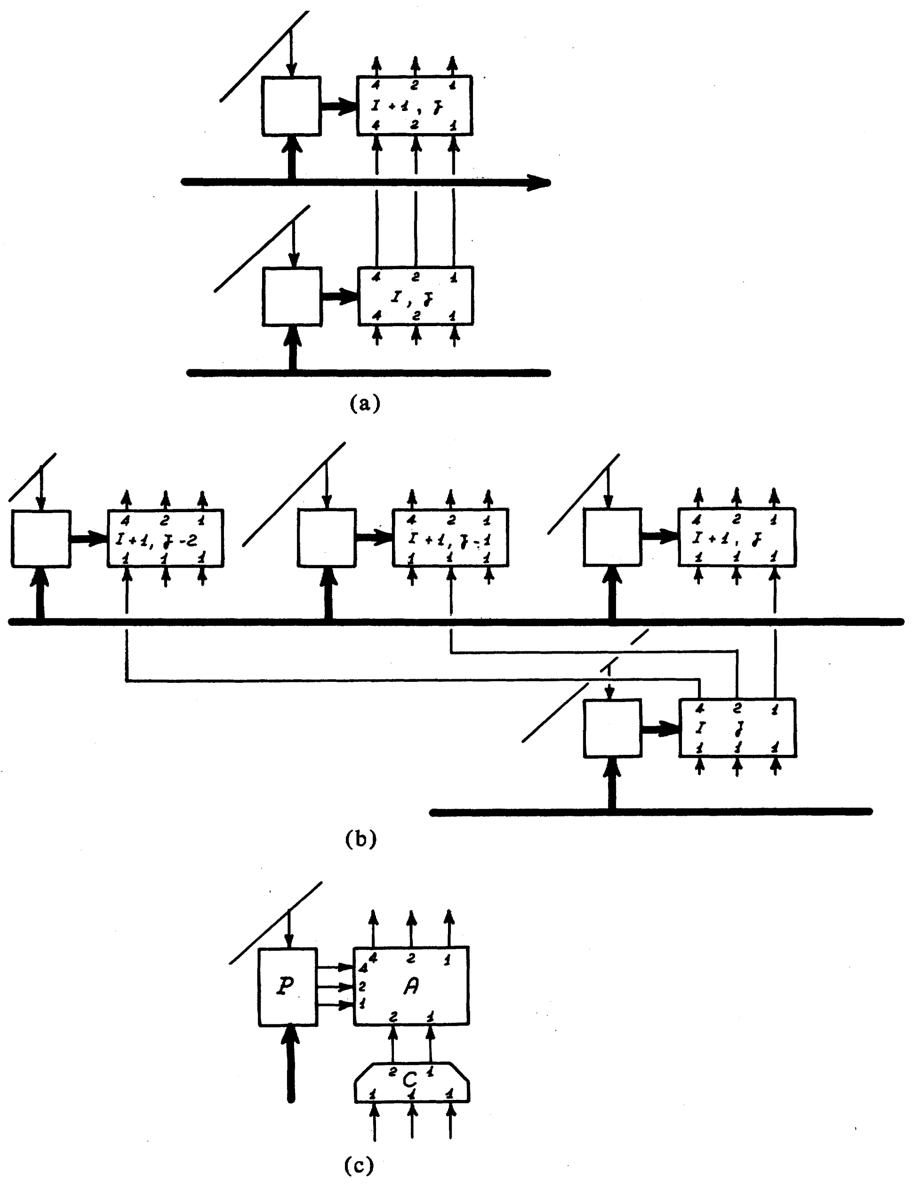

Fig. 5. (a) Connections between two parallel adders in the basic procedure. (b) Connections between parallel adders in the second procedure. (c) Decomposition of an elementary circuit into a parallel adder and a parallel counter.

A. The Second Procedure

In the description of the basic procedure, the choice of a particular negative number representation inside the divider structure is not necessary because this representation is relevant only to the design of each adder or subtractor circuit and not to the operating principle of the divider. The choice of the negative number representation becomes essential when the second and following procedures are described.

The representation that leads to the maximum simplification of the divider structure is a kind of redundant binary code in which digits 1, 0, $\bar{1}$ are used. Positive numbers are represented by the conventional binary code using only the digits 0 and 1 ($13_{10}=1101$); negative numbers are represented by the same coded form as the corresponding positive numbers, where $\bar{1}$ has been substituted for 1 ($-13_{10}=\bar{1}\bar{1}0\bar{1}$).[5]

[5] This can be obtained either by representing each digit by its sign and its absolute value or by employing ternary logic.

From now on, this code will be called simple redundant binary code.

A first modification to the basic procedure can be deduced from the following. Let $q_0 \cdots q_{i-1}, q_i \cdots$ be a redundant representation of the quotient; equivalent representation is obtained by subtracting (adding) from digit q_i a number K and adding (subtracting) to digit q_{i-1} the number $K/2$.

In the basic procedure the output of adder (I, J) (where I refers to the rows of the structure in Fig. 2, from bottom up, and J to the columns, from left to right), represented in simple redundant binary form, consists of a set of signals, with values $\bar{1}$, 0, 1 and with associated weights 1, 2, 4, 8 $\cdots$.

These signals, as illustrated in Fig. 5(a), showing the connection between two adders, are applied at the same weight inputs of adder $(I+1, J)$.

From the preceding remark it follows that the divider still works properly if, as shown in Fig. 5(b), the output of circuit (I, J) having weight 2 is applied with weight 1 at the input

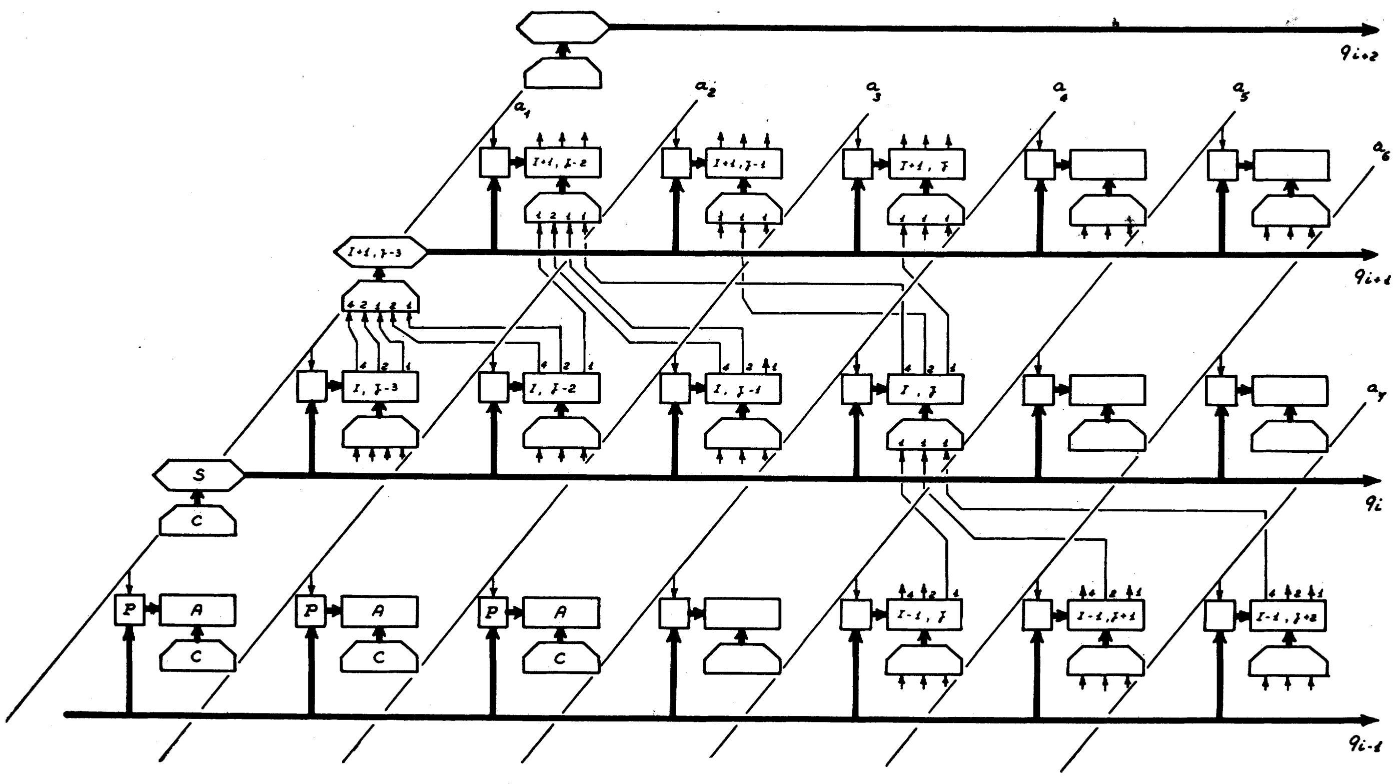

Fig. 6. Scheme of a divider implementing the second procedure.

of the circuit $(I+1, J-1)$; the weight 4 output of circuit (I, J) is applied with weight 1 at circuit $(I+1, J-2)$, etc.

This type of connection leads to an advantage with respect to the basic procedure since, by reducing the weight of the adder input signals, the outputs of each adder also decrease in magnitude and hence the absolute values of digits q_i in the quotient also decrease.

Each elementary circuit of Fig. 5(b) can be decomposed, as shown in Fig. 5(c), into a parallel adder and a parallel counter (circuit c); a parallel counter is defined to be a combinatorial circuit whose outputs, considered as a simple redundant binary number, represent the sum of the signals present at the input.[6]

Fig. 6 shows a part of a divider scheme, based on the procedure previously described. For simplicity, only some connections are reported. We have also supposed that the output of every adder is limited between -7 and $+7$ (three-output simple redundant binary signals).

The scheme in Fig. 6 shows the input and output connections to the circuit (I, J): the inputs are connected to the outputs of circuits $(I-1, J)$, $(I-1, J+1)$, and $(I-1, J+2)$; the outputs are connected to the inputs of circuits $(I+1, J)$, $(I+1, J-1)$, and $(I+1, J-2)$.

Also shown are the connections for the circuits on the left border of the structure: the outputs of circuit $(I, J-3)$ must be applied to the inputs of circuit $(I+1, J-3)$ with weights 4, 2, and 1; in the same way, the circuit $(I+1, J-3)$ will have as its inputs, with weights 2 and 1, the outputs with weight 4 and 2 of circuit $(I, J-2)$.

Circuit $(I+1, J-2)$ gets as its inputs, with weights 2 and 1, the weight 4 and weight 2 outputs of circuit $(I, J-1)$; and with weight 1, the weight 1 output of circuit $(I, J-2)$ and the weight 4 output of circuit (I, J).

The scheme in Fig. 6, is not the only possible one for the connections to the left-border circuits; for example, the weight 4 output of $(I, J-1)$ could be applied with weight 1 at $(I+1, J-3)$, rather than with weight 2 at $(I+1, J-2)$. The scheme in Fig. 6, however, has given the best results in the computer simulation of the various possible schemes.

In Table III the maximum and minimum values of the quotient digits q_i for a 12-bit divider are shown; it can be noted that these results are better than the ones obtained with the basic procedure.

In Fig. 3, the relative truncation error is plotted as a function of the number N_1 of digits q_i (curve II). Fig. 4 (curve II) plots the relation between N_1 and the number N of binary digits that can be obtained. From this curve the following approximate relation can be derived.

$$N_1 = 1,78\,(N-1).$$

B. The Third Procedure

The second procedure, like the basic one described in Section III, also leads to a divider structure that requires adders and subtractors with a large number of stages. A third procedure will be described here that greatly simplifies the divider structure. This procedure is based on the following. Let $q_0, q_1, \cdots q_i, q_{i+1}, \cdots$ be a redundant binary code

[6] Parallel counters have already been studied and implemented [11], [12] for positive numbers. They were used to realize parallel multipliers [1]-[3]. It may be remarked that the well-known half-adder and full-adder circuits are parallel counters with, respectively, two and three inputs. The problem of the implementation of high-speed ternary parallel counters, which can be used to implement dividers, will be illustrated in [13].

TABLE III
MAXIMUM AND MINIMUM VALUES OF THE QUOTIENT DIGITS q_i FOR A 12-BIT DIVIDER BASED ON THE SECOND PROCEDURE

i	Minimum q_i	Maximum q_i
1	0	1
2	0	1
3	−1	1
4	−1	1
5	−2	2
6	−2	4
7	−5	3
8	−4	7
9	−9	5
10	−5	14
11	−19	8
12	−11	26
13	−32	17
14	−23	41
15	−55	34
16	−49	76
17	−102	74
18	−108	139
19	−187	161
20	−240	256
21	−360	361
22	−540	540
23	−808	810

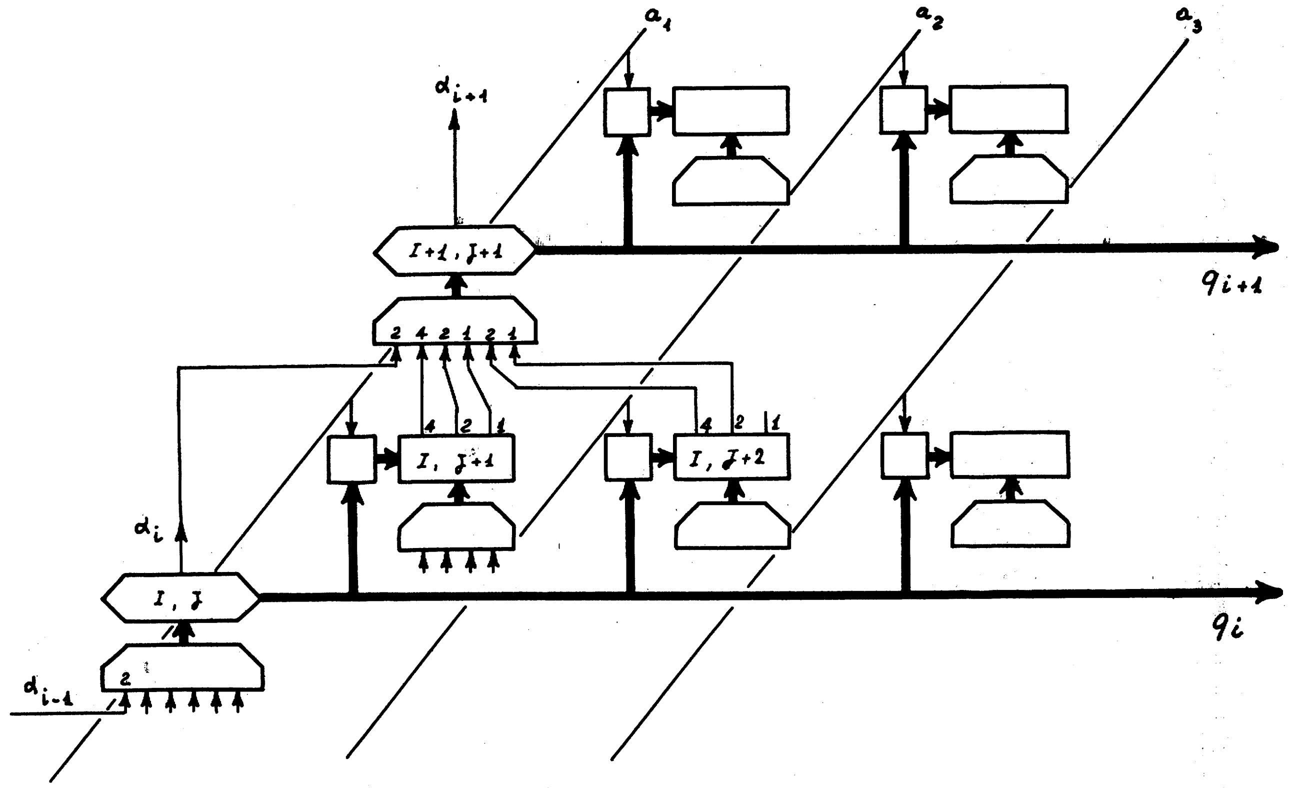

Fig. 7. Modification of the divider shown in Fig. 6, in order to implement the third procedure.

of the quotient; an equivalent code is obtained if a number K is added (subtracted) to digit q_i and the number $2K$ is subtracted (added) to digit q_{i+1}.

The third procedure is similar to the second one with the only differences being that, if q_i is positive and higher than a given threshold s_p, then 1 is subtracted from q_i and 2 is added to q_{i+1}. If digit q_i is negative and its absolute value is higher than a given threshold s_n, then 1 is added to q_i and 2 is subtracted from q_{i+1}. Since generally if q_i has a high value, digit q_{i+1} has a higher value with opposite sign, the third procedure reduces the values of both digits.

Both the test on the two thresholds and the correction of digit q_i may be performed by the subtracting circuit, as is shown in Fig. 7, which displays a part of a divider. The sub-

TABLE IV

Maximum and Minimum Values of the Output of the Adders of a 12-Bit Divider Based on the Third Procedure

																							2 / -1
																						3 / -3	0 / 0
																					3 / -3	4 / -3	4 / -3
																				4 / -3	5 / -3	4 / -4	5 / -3
																			4 / -2	4 / -3	5 / -3	6 / -3	5 / -2
																		4 / -2	4 / -3	4 / -3	5 / -4	4 / -3	5 / -3
																	3 / -3	3 / -2	3 / -3	3 / -4	4 / -4	3 / -3	3 / -3
																4 / -3	4 / -3	4 / -3	5 / -3	3 / -3	3 / -2	3 / -2	4 / -3
															3 / -3	4 / -3	5 / -3	5 / -3	4 / -3	4 / -3	4 / -4	4 / -3	3 / -3
														3 / -4	4 / -3	4 / -4	5 / -3	5 / -5	5 / -2	5 / -3	4 / -3	4 / -3	3 / -4
													4 / -3	4 / -3	4 / -3	4 / -3	4 / -2	4 / -3	4 / -3	5 / -2	4 / -3	3 / -2	4 / -3
												4 / -3	4 / -2	4 / -3	5 / -3	4 / -4	4 / -3	4 / -3	4 / -2	4 / -3	6 / -4	4 / -4	4 / -3
											3 / -3	3 / -3	3 / -3	4 / -4	4 / -3	3 / -3	4 / -3	3 / -3	4 / -3	4 / -2	3 / -2	3 / -3	
										3 / -3	4 / -3	5 / -2	5 / -5	4 / -3	3 / -4	3 / -3	4 / -3	4 / -3	4 / -3	3 / -3	3 / -3		
									3 / -3	4 / -3	4 / -3	4 / -3	3 / -2	4 / -3	4 / -3	4 / -2	4 / -3	4 / -2	3 / -3	3 / -3			
								2 / -2	4 / -2	4 / -3	4 / -3	4 / -2	4 / -3	4 / -2	4 / -3	4 / -3	4 / -3	3 / -2	2 / -2				
							3 / -2	3 / -2	4 / -3	4 / -3	4 / -2	4 / -2	3 / -2	4 / -3	4 / -3	4 / -3	3 / -3	3 / -2					
						2 / -2	4 / -2	3 / -2	4 / -1	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2						
					2 / -1	3 / -1	4 / -2	4 / -2	3 / -2	4 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -1							
				2 / -1	3 / -1	3 / -1	4 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -1								
			1 / -1	2 / -1	2 / -1	2 / -1	3 / -1	3 / -1	3 / -1	3 / -1	3 / -1	2 / -1	2 / -1	1 / -1									
		1 / -1	2 / 0	2 / -1	3 / -1	3 / -1	3 / -1	3 / -1	3 / -1	3 / -1	2 / -1	2 / -1	1 / -1										
	1 / 0	1 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	1 / 0											
1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0												

Note: The upper entries in the columns are the maximum and minimum values of the quotient digits q_i.

tractor has one input X_i and two outputs q_i and α_i; the relation between input and outputs is

$$q_i = \begin{cases} 2 - X_i \\ 1 - X_i \\ -X_i \end{cases} \text{ and } \alpha_i = \begin{cases} 1 \\ 0 \\ -1 \end{cases} \text{if } \begin{cases} X > s_n + 1 \\ (1 - s_p) \leq X \leq (1 + s_n) \\ X < 1 - s_p. \end{cases}$$

The correction of digit q_{i+1} is obtained by applying the simple redundant binary signal α_i, with weight 2, at the parallel counter of circuit $(I+1, J+1)$. (See Fig. 7.)

A 12-bit divider, based on the third procedure, has been simulated on a digital computer and many threshold values were tried; the best results have been obtained with threshold values $s_n = 1$ and $s_p = 2$.

The minimum and maximum output values of each adder of that divider are listed in Table IV; the leftmost entries in

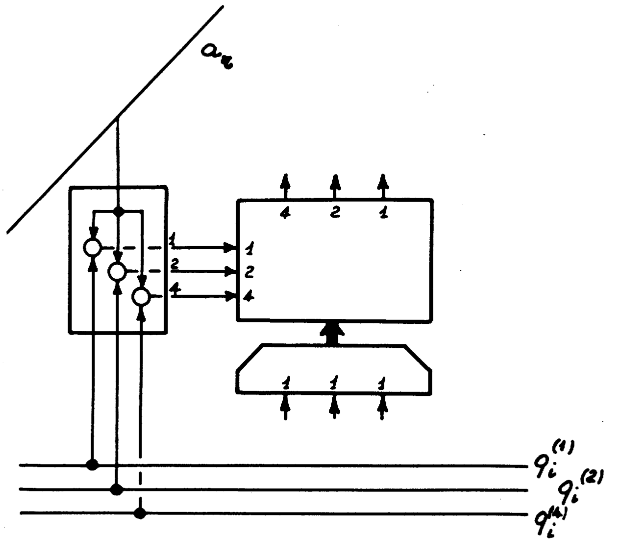
Fig. 8. The internal scheme of a multiplying circuit; each circle represents a ternary gate.

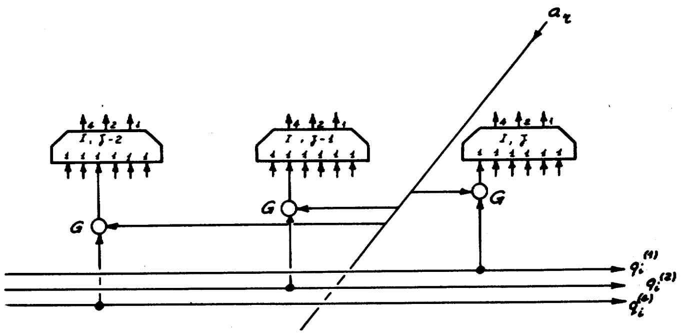
Fig. 9. Connections of the outputs of a multiplying circuit in the fourth procedure.

each row of the same table are the minimum and maximum values of digits q_i.

Table IV shows that the third procedure is much better than the first and the second ones. For a 12-bit divider, the output of each elementary circuit lies between -5 and $+6$. In this way a large reduction of the number of stages of each adder is obtained. Another advantage is the great reduction of the number N_1 of digits q_i that are required in order to obtain N binary digits of the quotient. From Fig. 4, (curve III) the following approximate relation for N_1 is obtained:

$$N_1 = N + 2.$$

C. The Fourth Procedure

In Fig. 8 an elementary circuit used to implement the second and third procedures is shown. Both the simple redundant signals $q_i^{(1)}$, $q_i^{(2)}$, $q_i^{(4)}$, which represent digit q_i $(-7 \leq q_i \leq +7)$, and the internal structure of multiplying circuits that compute the product $q_i a_r$ are illustrated in this figure.[7]

Each multiplier output, with weight 1, 2, and 4, is connected to the adder input having the same weight. The fourth procedure is based on the application of the same statement that is the basis of the second procedure to the outputs of each multiplying circuit. The three outputs of the multiplier in Fig. 8 can be connected, as shown in Fig. 9, all with weight 1, to the inputs of circuits (I, J), $(I, J-1)$ and $(I, J-2)$.

This type of connection presents two main advantages. The weights of many signals are reduced so that the minimum and maximum values of each elementary circuit output are also decreased; only one parallel counter is needed to implement an elementary circuit, since now all input signals have weight 1.

In Fig. 10 a part of a divider based on the fourth procedure is shown. It is realized by a regular array of parallel counters, gates, and subtractors. We have assumed that counter output values and digits q_i lie between -3 and $+3$; only on the left border of the structure three-output parallel counters are used. These hypotheses are justified, as will be seen later, by the results provided by simulation. In the scheme presented in Fig. 10 all input and output connections to counters (I, J), $(I, J-3)$, and $(I+1, J-3)$ are shown. Circuit (I, J), which is an example of a circuit internal to the cellular structure, receives input signals from circuits $(I-1, J)$ and $(I-1, J+1)$. Its outputs are applied at circuits $(I+1, J-1)$ and $(I+1, J)$; at its inputs signals $a_4 q_i^{(1)}$ and $a_5 q_i^{(2)}$ are also fed.

Circuit $(I, J-3)$ which is on the border, is connected to

[7] a_r is a binary signal, but every simple redundant signal $q_i^{(j)}$ can take on one of the three values: $\bar{1}$, 0, 1; thus the circuit that computes the product $a_r\, q_i^{(j)}$ is a ternary gate if three-level electrical signals are used to represent simple redundant digits; or consists of two binary gates if each simple redundant digit is coded by two binary signals.

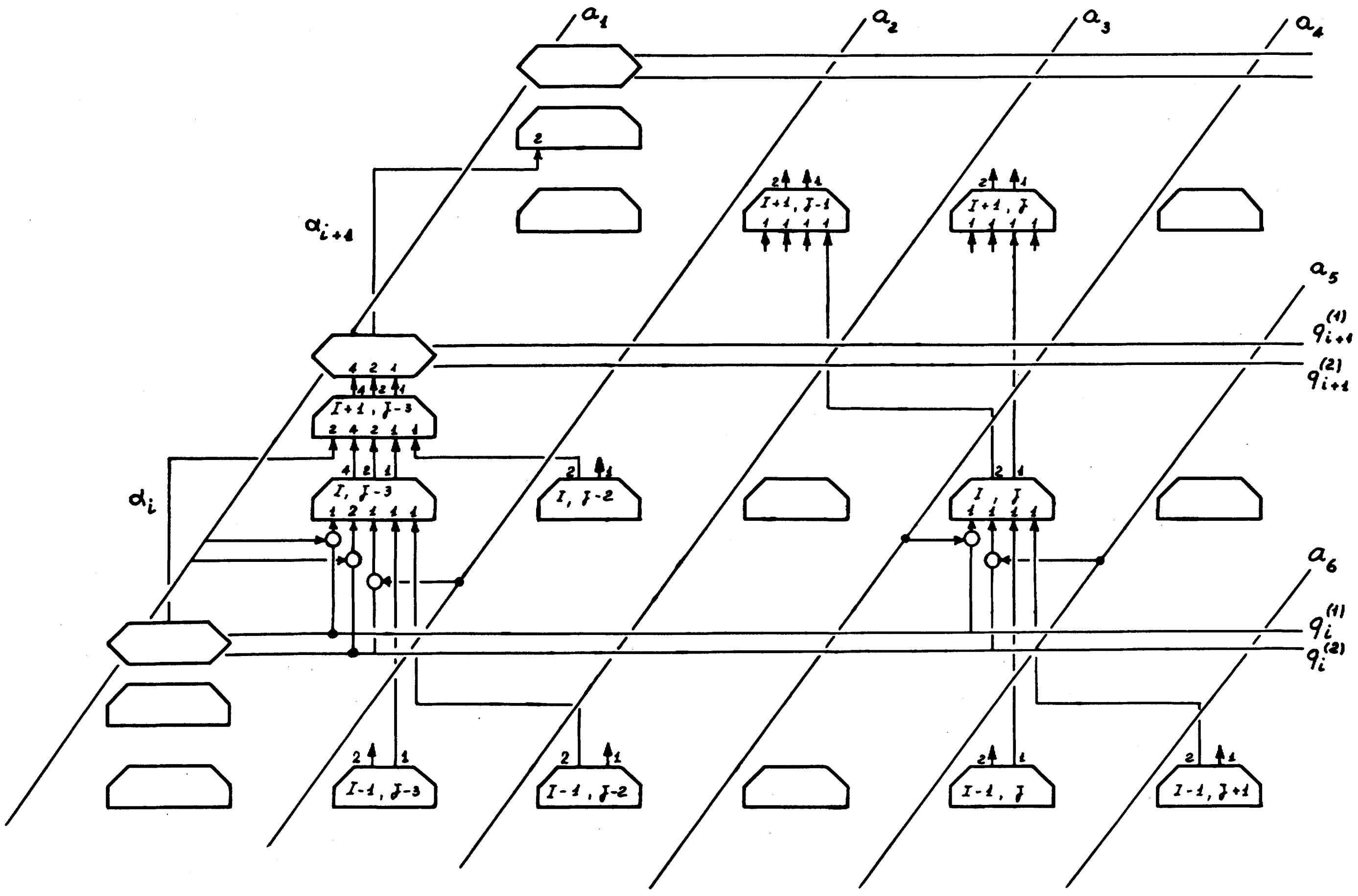

Fig. 10. Structure of a divider implementing the fourth procedure.

outputs of $(I-1, J-3)$ and $(I-1, J-2)$ and receives the signals $a_1q_i^{(1)}$ and $a_2q_i^{(2)}$; signal $a_1q_i^{(2)}$ is applied, with weight 2, at circuit $(I, J-3)$ because there are no counters on the left of this circuit.

Output signals from $(I, J-3)$ and signal α_i are applied at circuit $(I+1, J-3)$, according to the rules of the third procedure.

Table V shows the minimum and maximum output values of each counter for a 12-bit divider; the leftmost entries in each row of this table are the minimum and maximum values of digits q_i. Thresholds $s_n=1$ and $s_p=2$ were used.

This divider scheme has the advantage over the one derived from the third procedure of constraining the values of the q_i digits and of the output values of parallel counters (with the exception only of the leftmost counter in each row) to lie between -3 and $+3$.

Truncation errors (Fig. 3, curve IV) and the relation between N_1 and N (Fig. 4, curve IV) are the same as for the third procedure.

V. The Redundant Binary to Binary Converter

As we have already said in preceding sections, the quotient Q is obtained in redundant binary form. In order to obtain a binary coded quotient, a suitable code converter must follow each of the divider schemes described above.

In this section a code converter will be described consisting of two parts. The *first* converts the redundant binary quotient into two simple redundant numbers Q' and Q'' whose sum equals the quotient; these numbers are applied at the inputs of the *second* part which produces the binary coded quotient.

As will be seen in what follows, the first part is needed only when there is at least one digit q_i that may take on a value higher than 3 or lower than -3; in this way, if the fourth procedure is adopted, only the second part is required.

The first part will now be described by means of the example reported in Table VI. Position index i and digits q_i are reported in the first and second rows, respectively. Matrix A is obtained from digits q_i. As an example digit $q_7=13$ is coded by 1 1 0 1. These four digits, having weights $2^3, 2^2, 2^1, 2^0$, respectively, are displayed in matrix A, left shifted by 3, 2, 1, 0 places with respect to position $i=7$. In the same way the five simple redundant digits $\bar{1}\ 0\ \bar{1}\ \bar{1}\ \bar{1}$ representing digit $q_{10}=\overline{23}$ are recorded in matrix A in columns 6–10.[8] Only five simple redundant digits are required to code the highest value digit of $Q(q_{13}=\overline{31})$; then matrix A has five rows.

The sum of the five addends, each corresponding to a row of matrix, still represents quotient Q. In order to reduce the number of addends, a generalization of a procedure already used [1], [2] to perform parallel multiplication will be used.

[8] No circuit is needed to construct matrix A, which is obtained only by a suitable disposition of signals $q_i^{(j)}$ representing digits q_i.

TABLE V

MAXIMUM AND MINIMUM VALUES OF THE OUTPUT OF THE ADDERS OF A 12-BIT DIVIDER BASED ON THE FOURTH PROCEDURE

																							2 / -1
																						3 / -2	0 / 0
																					3 / -3	3 / -4	2 / -2
																				3 / -3	5 / -4	4 / -3	2 / -2
																			3 / -3	5 / -4	3 / -3	3 / -3	2 / -2
																		3 / -2	4 / -3	3 / -3	3 / -3	3 / -2	2 / -2
																	3 / -2	4 / -4	3 / -3	3 / -2	3 / -2	3 / -2	2 / -2
																3 / -3	4 / -4	3 / -3	3 / -3	3 / -3	3 / -3	3 / -2	2 / -2
															3 / -3	4 / -4	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2
														3 / -3	4 / -4	3 / -3	3 / -3	3 / -3	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2
													3 / -2	4 / -4	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2
												3 / -3	4 / -4	3 / -3	3 / -2	3 / -3	3 / -2	3 / -2	3 / -2	3 / -3	2 / -2	2 / -2	1 / -1
											3 / -2	4 / -3	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2	2 / -2	1 / -1	
										3 / -3	4 / -4	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2	1 / -1		
									3 / -2	4 / -4	3 / -2	3 / -3	3 / -2	3 / -2	3 / -3	3 / -2	3 / -2	2 / -2	2 / -2	1 / -1			
								2 / -2	4 / -3	3 / -3	3 / -3	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2	1 / -1				
							3 / -2	4 / -3	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2	1 / -1					
						2 / -2	4 / -3	3 / -2	3 / -1	3 / -2	3 / -2	3 / -2	2 / -2	3 / -2	2 / -2	2 / -2	1 / -1						
					2 / -2	4 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2	1 / -1							
				2 / -1	3 / -1	3 / -1	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	3 / -2	2 / -2	2 / -2	1 / -1								
			1 / -1	2 / -1	2 / -1	2 / -1	3 / -1	3 / -1	3 / -1	3 / -1	3 / -1	2 / -1	2 / -1	1 / -1									
		1 / -1	2 / 0	2 / -1	3 / -1	3 / -1	3 / -1	3 / -1	3 / -1	3 / -1	2 / -1	2 / -1	1 / -1										
	1 / 0	1 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	1 / 0											
1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0	1 / 0												

If the sum of the five rows in matrix A is performed column by column, Q_a is then obtained (see Table VI). Matrix B can be derived from Q_a applying the same rules used to obtain matrix A from Q.

The rowwise sum of a matrix can be obtained by means of one parallel counter per column; as an example, the five digits in matrix A, column $i=7$, are applied at the inputs of a parallel counter; its output signals with values 1 0 0 representing number 4, are recorded in matrix B in columns 5–7.

In the same way from matrix B another equivalent form Q_b of the quotient is obtained and then a matrix C with two rows which are the two output numbers Q' and Q'' of the first part of the converter. As mentioned in the preceding section, each output digit q_i of a divider based on the fourth procedure is represented only by two simple redundant digits. Thus matrix C is directly obtained and the first part

TABLE VI

i	1	2	3	4	5	6	7	8	9	10	11	12
Q	1	0	$\bar{2}$	3	7	$\bar{9}$	13	11	20	$\overline{23}$	28	$\overline{31}$
MATRIX A	1	0	0	1	1	$\bar{1}$	1	1	0	$\bar{1}$	0	$\bar{1}$
		$\bar{1}$	1	1	0	0	1	0	$\bar{1}$	0	$\bar{1}$	
			1	0	1	0	1	$\bar{1}$	1	$\bar{1}$		
			$\bar{1}$	1	1	0	0	1	$\bar{1}$			
					1	$\bar{1}$	1	$\bar{1}$				
Q_a	1	$\bar{1}$	1	3	4	$\bar{2}$	4	0	$\bar{1}$	$\bar{2}$	$\bar{1}$	$\bar{1}$
MATRIX B	1	$\bar{1}$	1	1	0	0	0	0	$\bar{1}$	0	$\bar{1}$	$\bar{1}$
			1	0	$\bar{1}$	0			$\bar{1}$			
			1		1							
Q_b	1	$\bar{1}$	3	1	0	0	0	0	$\bar{2}$	0	$\bar{1}$	$\bar{1}$
MATRIX C												
Q'	1	$\bar{1}$	1	1	0	0	0	0	0	0	$\bar{1}$	$\bar{1}$
Q''	0	1	0	0	0	0	0	$\bar{1}$	0	0	0	0

Note: First part of a redundant binary to binary conversion. From a redundant binary coded quotient Q a matrix A is obtained. Then an equivalent representation Q_a of the quotient can be derived by adding all the entries in each column in matrix A. The reduction procedure stops when a two-row matrix is obtained.

of the converter is not required. The second part of the code converter may be realized in many ways; the following seems to the author to be the most convenient one. Each number Q' and Q'' is converted into binary form by means of two simple redundant to binary converters whose outputs are applied at the inputs of a conventional parallel adder.

The design of a simple redundant to binary converter can be based on the following statement. Any sequence formed by a group of digits 1 preceded by $\bar{1}$ and followed by 2 ($\cdots 000\bar{1}111 \cdots 112000 \cdots$) is a redundant code of number zero so that it may be added to every number without changing its value. This sequence will be called a *zero sequence.*

Digits of number Q to be converted are successively examined starting from the least significant one; the construction of a zero sequence is started when the first $\bar{1}$ is encountered and is stopped when the first 1 is found. All the zero sequences obtained this way are added to number Q. The resulting sum contains 0's and 1's only. In Table VII an example is presented that requires two zero sequences.

A simple redundant to binary converter based on the principle described above can be implemented by an iterative one-dimensional network (Fig. 11). Table VIII shows the transition table of the network.

The first part of the converter, when needed, can be made very fast because no carry propagations are required; the second part is composed of three circuits, two single redundant to binary converters and one binary parallel adder, each requiring a carry propagation; however, since carries propagate in the same direction, from the least to the most significant digit, the time delay of the converter corresponds to the time required for only one carry propagation.

VI. Quotient of Two Numbers

All procedures described above can be modified in order to produce the quotient $Q=C/A$ of two given normalized numbers C and A. Only the first procedure will be discussed here since the results of the discussion can be easily extended to the other procedures.

TABLE VII
An Example of Simple Redundant Binary to Binary Conversion

1	1	0	0	−1	−1	0	−1	0	1	1	−1	1	0	1
0	−1	1	1	1	1	1	2	0	0	−1	2	0	0	0
1	0	1	1	0	0	1	1	0	1	0	1	1	0	1

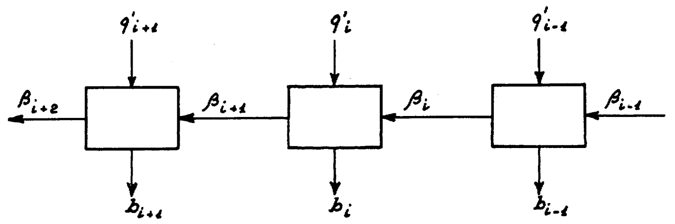

Fig. 11. Simple redundant binary to binary code converter.

TABLE VIII
Transition Table of Each Cell in Fig. 11

q_i'	β_i	b_i	β_{i+1}
1	0	1	0
0	0	0	0
−1	0	1	1
1	1	0	0
0	1	1	1
−1	1	0	1

The first procedure is still valid with only the following difference. In Table I the number 0.1 1 1 1 1, $\cdots$, sum of all the rows of the matrix, must be replaced by the number $[c_1, c_2, c_3, \cdots]$ which represents, in binary code, the dividend C. Equations (2) and (2′) are then replaced by

$$\begin{aligned} q_0 &= 1 \\ q_1 &= c_2 - a_2 \\ q_2 &= c_3 - (a_3 + a_2q_1) \\ q_3 &= c_4 - (a_4 + a_3q_1 + a_2q_2) \\ &\vdots \\ q_i &= c_{i+1} - (a_{i+1} + a_iq_1 + a_{i-1}q_2 + \cdots + a_2q_{i-1}). \end{aligned}$$

The scheme shown in Fig. 2 can still be employed, provided that the function performed by subtractor circuits becomes:

$$q_i = c_{i+1} - X.$$

Tables II–V are no longer valid. Higher positive and lower negative values have to be expected for the adder outputs and for digits q_i. This results in a more expensive divider implementation.

VII. Conclusion

Division is the slowest fundamental arithmetic operation performed by digital computers because it generally consists of many elementary operations—sums, subtractions, and comparisons—each requiring a carry propagation. Most of the methods already described reduce the computation time by reducing the number of elementary operations.

In this paper four dividers are described where high-speed operation is achieved by means of two-dimensional combinatorial ternary networks requiring only two carry propagations.

First the problem of computing the inverse $1/A$ of a given number A is discussed, then modifications are described in order to compute the quotient C/A of two numbers C and A.

Each divider consists of two distinct parts. The first computes the quotient in redundant binary form, i.e., in a base 2 code where digits can take every positive and negative integer value. The second is a converter from redundant binary to binary code. Each of these two parts requires only one carry propagation.

The first divider is interesting only because of its very simple operating principle, but its implementation is very expensive and impractical. The cost is largely reduced when the other dividers are considered. These circuits are obtained by successive modifications introduced in the first divider.

The fourth dividing circuit, the most interesting one both from the cost and speed viewpoints, is a cellular array of parallel binary counters, i.e., of combinatorial circuits whose outputs represent, in binary coded form, the sum of input signals.

The cost of this divider is proportional to the square of the quotient and divisor length, because of the two-dimensional structure of the circuit. When the quotient of two numbers with a high number of bits is to be computed, a divider for shorter operands may be used if one of the well-known multiple-precision division methods is employed.

References

[1] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, May 1965.

[2] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 14–17, Feb. 1964.

[3] D. Ferrari, "Un moltiplicatore numerico parallelo sperimentale," in *Proc. 67th Meeting AEI* (Alghero, Italy), Sept. 1966.

[4] J. B. Wilson and R. S. Ledley, "An algorithm for rapid binary division," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 662–670, Dec. 1961.

[5] D. Ferrari, "A division method using a parallel multiplier," *IEEE Trans. Electron. Comput.* (Short Notes), vol. EC-16, pp. 224–226, Apr. 1967.

[6] R. E. Gilman, "A mathematical procedure for machine division," *Commun. Ass. Comput. Mach.*, vol. 12, pp. 10–12, Apr. 1959.

[7] M. L. Stein, "Divide-and-correct methods for multiple precision division," *Commun. Ass. Comput. Mach.*, vol. 7, p. 472, Aug. 1964.

[8] Ṡ. K. Nandi and E. W. Krishnamurthy, "A simple technique for digital division," *Commun. Ass. Comput. Mach.*, vol. 10, pp. 299–301, May 1967.

[9] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, vol. EC-11, pp. 512–517, Aug. 1962.

[10] G. Metze, "A class of binary divisions yielding minimally represented quotients," *IRE Trans. Electron. Comput.*, vol. EC-11, pp. 761–764, Dec. 1962.

[11] S. Morelli and R. Stefanelli, "Alcuni circuiti contatori di tipo parallelo," *Alta Frequenza*, vol. 34, pp. 836–846, Dec. 1965.

[12] M. Cifariello and R. Stefanelli, "Un circuito ordinatore e suo uso nella sintesi di funzioni logiche simmetriche," in *Proc. 67th Meeting AEI* (Alghero, Italy), Sept. 1966.

[13] ——, "Alcuni contatori parallelo ternari," to be published.