Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

A VLSI ALGORITHM FOR DIRECT AND REVERSE CONVERSION FROM WEIGHTED BINARY NUMBER SYSTEM TO RESIDUE NUMBER SYSTEM

G. Alia, E. Martinelli

Rapporto F82-01

Progetto finalizzato Informatica

Mumicro-VLSI

ABSTRACT

Residue Number Systems (RNS) are proved to be useful in many applications, as for example in signal processing. In this work a VLSI computing architecture is proposed for converting an unsigned integer number N from the weighted binary representation into and out a residue code based on s moduli.

For this architecture a possible layout is given and its complexity is evaluated in terms of area and time. Under several hypotheses on RNS parameters, constructive upper bounds ranging from $O(n^2 \log n))$ to $O(n^2 \log\log n)$ and from $O(\log^2 n)$ to $O(\log n)$ for area and time respectively have been obtained for the direct conversion. On the contrary, constructive upper bounds $A=O(n^2 \log n)$ and $T=O(\log^2 n)$ have been found independent of the formed hypotheses, for the reverse conversion.

1. INTRODUCTION

In the last years many authors have directed their attention to the residue number systems (RNS) for the implementation of fast digital signal processing hardware as, for example, discrete Fourier transform processors and digital filters /1,2,3,4/. The interest proceeds from the fact that arithmetic units based on a residue number system are fast and simple, at least for applications requiring algorithms where mainly additions and multiplications are used, as the class of applications mentioned above.

One of the problems to deal with when considering arithmetic units based on RNS is the conversion of input data from the weighted system to the residue number system and the reverse conversion of output data.

In fact the double conversion is an overhead which would offset the advantage of high speed if a suitable algorithm is not devised and its implementation well designed /5,6/.

In this work a VLSI computing architecture is proposed for converting an unsigned integer number N, represented in the weighted binary system, into and out a residue code based on s moduli.

For this architecture a possible layout is given and its complexity is evaluated in terms of the silicon area required for its implementation and of the time spent for the execution of the conversion algorithm.

The following notation is used to denote complexity relations:
let $f(n)$ and $g(n)$ be two functions of n

$f(n)=O(g(n))$ there exists a positive constant k for which $f(n) \leq kg(n)$ for all sufficiently large n. This means that $g(n)$ is an upper bound of $f(n)$;

$f(n)=\theta(g(n))$ there exist two positive constants k_1, k_2 for which $k_1 g(n) \leq f(n) \leq k_2 g(n)$, for all sufficiently large n. This means that g(n) and f(n) are of the same order;

$f(n)=\Omega(g(n))$ there exists a positive constant k for which $f(n) \geq k g(n)$ for all sufficiently large n. This means that g(n) is a lower bound of f(n).

The VLSI model of computation we choose to derive a complexity measure is based on the following assumptions, generally accepted /7,8,9/

a) the area required to store 1 bit of information is θ(const.);

b) wires have minimal width $\lambda = \theta$(const.);

c) the distance between parallel wires is $\theta(\lambda)$;

d) only two wires may cross at a point;

e) wires run horizontally and vertically;

f) the time required to propagate a binary signal along a wire is θ (const.). Long wires, of length L, meet this assumption if driven by properly dimensioned drivers with area A=O(L).

Section 2 is devoted to a brief resume of residue number systems, limited to what is needed by this work, and to introducing some assumptions about their characteristics. Next two sections describe the algorithm and the corresponding VLSI structure for the direct and reverse conversion between the weighted binary system and the RNS, respectively, and provide also the complexity relations. In the last section the influence of the RNS parameters on the previously derived complexity relations is discussed.

2. RESIDUE NUMBER SYSTEMS

Let N be an unsigned integer number, represented, in the weighted binary system, in a field of length $n=2^p$ bits:

$$N \triangleq \{ b_{n-1}, b_{n-2}, \ldots, b_2, b_1, b_0 \} = \sum_{j=0}^{n-1} b_j 2^j \qquad b_j \in \{0,1\}$$

In a residue number system, N is represented by s numbers α_i:

$$N \triangleq \{ \alpha_1, \alpha_2, \ldots, \alpha_s \}$$

where

$$\alpha_i \triangleq |N|_{m_i} = N - \left\lfloor \frac{N}{m_i} \right\rfloor . m_i \quad ,$$

$\lfloor N/m_i \rfloor$ is the largest integer not exceeding N/m_i, and $\{ m_i \}$ is the set of moduli. If all the m_i are relatively prime, it can be shown that /10/ there is a unique representation for each number in the range

$$0 \leq N < \prod_{i=1}^{s} m_i \triangleq M$$

Moreover, the binary operations of addition, subtraction and multiplication can be performed separately on α_i; for example:

$$|A+B|_{m_i} = |A|_{m_i} + |B|_{m_i}$$

and this allows very fast parallel implementations of special arithmetic units based on RNS.

In this work we assume that

$$m_i = \theta(m) \qquad 1 \leq i \leq s$$

and

$$2^n \leq \prod_{i=1}^{s} m_i \leq 2^{n+1}$$

From these assumptions it follows that, for large n

$$2^n = \theta(m^s) , \qquad n \approx s \log m .$$

The latter relation shows that the sum of the fields lengths of the weighted representations of α_i is, for large n, about equal to the length of the positional representation of N.

3. THE CONVERSION FROM THE WEIGHTED TO THE RESIDUE REPRESENTATION

The algorithm used for converting N from the weighted number system to the residue number system is based on the following considerations.

Let us consider the RNS representations of the powers of two modulo m_i, that is

$$r_j^i = |2^j|_{m_i} \qquad 0 \leq j \leq n-1$$

then the RNS representation of N is given by

$$\left\{ \left| \sum_{j:b_j=1} r_j^i \right|_{m_i} , \qquad 1 \leq i \leq s \right\}$$

The structure we propose performs this computation with an high degree of parallelism while retaining features as modularity and regularity, requested by the VLSI technology. In such structure, for

each modulus m_i, n storage cells of $\log m_i$ bits are pre-loaded with the values of r_j^i and are enabled to output their contents or zero's depending on the value 1 or 0 of the j-th bit of N. A simple processing element is associated to each cell corresponding to odd j. It is supplied with an accumulator register and is able to perform binary addition and subtraction and sign test operations, necessary to implement a modulo m_i addition.

The conversion is carried out in $\theta(\log n)$ steps. In Fig. 1 the time behaviour of the VLSI algorithm is represented, with active processing elements put into evidence at each step. At the first step, the outputs of each pair of adjacent cells are added modulo m_i and all the n/2 processing elements are active; at step h the results of step h-1 are added modulo m_i in pairs and $n/2^h$ processing elements are active; finally, $\alpha_i = |N|_{m_i}$ is contained in the accumulator register corresponding to j=n-1.

In Fig. 2 the layout of the proposed VLSI structure is drawn. It is arranged as an array of identical circuit modules; only the first row and the first column of the array are shown for the sake of simplicity. Each circuit module consists of two one-bit registers R, a one-bit processing element PE with its own accumulator A and a control gate G, as illustrated in Fig. 3.

Data processed at a module are propagated through a data bus, which can be partitioned by means of control gates so that active modules are independent one another.

Times at which adjacent bus sections are connected are decided by the zero-crossing of a $\log\log n$-counter which is initialized before each conversion run with a wired configuration.

The control line Z_R connects registers R's to the data bus at the first step, while puts their outputs in high impedance in the next steps.

The layout area can be valuated supposing that the processing elements of a row need a total area O(logmloglogm) /11/ and considering that s structures are necessary, all equal to the just defined one and placed each at the side of the other (see Fig. 4):

$$A = O(snloglogm(logm + loglogn)) = O(loglogm(n^2 + snloglogn)).$$

As the time required to add modulo m_i two numbers less than m_i by means of a binary adder is of the same order as the time of a binary addition, that is O(loglogm) /11/, the conversion time is

$$T = O(logn\ loglogm)$$

4. THE CONVERSION FROM THE RESIDUE TO THE WEIGHTED REPRESENTATION

Consider the RNS representation of $N=\{\alpha_1, \alpha_2, \ldots, \alpha_s\}$; it can be observed that N is given in the weighted representation by

$$\sum_{i=1}^{s} N_i$$

where N_i is the n-bits weighted representation of $(0,0,\ldots,\alpha_i,\ldots,0,0)$.

In turn, N_i can be obtained by

$$N_i = \sum_{k=1}^{\lceil logm_i \rceil} b_k^{(i)} \cdot q_{ik}$$

where $b_k^{(i)}$ is the bit value in the k-th position of α_i and q_{ik} is the n-bits weighted representation of $(0,0,\ldots,2^{k-1},\ldots,0,0)$.

Hence

$$N = \sum_{i=1}^{s} \sum_{k=1}^{\lceil \log m_i \rceil} b_k^{(i)} \cdot q_{ik} \; ;$$

it is then possible to use the same approach as for the direct conversion, defining a VLSI structure with $\sum_{i=1}^{s} \log m_i$ storage cells of n bits. Each cell is pre-loaded with the proper q_{ik}. In this case Processing Elements require only binary n-bit adders. The procedure for the computation of N is similar to the procedure described for the direct conversion and consequently

$$A = O(s \log n \log m (n + \log\log(s \log m))) = O(n^2 \log n)$$

and

$$T = O(\log(s \log m) \log n) = O(\log^2 n)$$

5. INFLUENCE OF RESIDUE NUMBER SYSTEM FEATURES ON THE COMPLEXITY OF THE VLSI STRUCTURE

The complexity of the VLSI structure has been obtained as a function of n, the number of bits in the weighted system, s and m_i, that are the number and the values of moduli in the residue number system. The choice of s and m_i is essential for the performance of any computing system based on RNS arithmetic, and, consequently, it is not convenient that it depends only on the design of direct and reverse conversion circuits. For these reasons, only hypotheses can be formed at this level on the values that s and m_i can take.

In Table 1 A(n) and T(n) for the direct and reverse conversions are given according to the following hypotheses:

a) $s = \theta(\text{const.})$; $\log m = \theta(n)$

b) $s = \theta(\log n)$; $\log m = \theta(n/\log n)$

c) $s = \theta(n/\log n)$; $\log m = \theta(\log n)$

d) $s = \theta(n)$; $\log m = \theta(\text{const.})$

It can be remarked that A(n) and T(n) in the reverse conversion have the same upper bounds $O(n^2 \log n)$ and $O(\log^2 n)$ respectively, whatever hypothesis is formed. On the contrary, in the direct conversion, the upper bound for A(n) is $O(n^2 \log n)$ but for the last two hypotheses, for which A(n) is $O(n^2 \log\log n)$. Note that the last hypothesis ($s = \theta(n)$, $\log m = \theta(\text{const.})$) must be considered as a trend hypothesis, because it is impossible to keep logm constant as n arbitrarily increases, in fact for each constant $k > 0$, the numbers less than k form a finite set.

On the other hand the upper bound for T(n) is decreasing from $O(\log^2 n)$ to $O(\log n)$ while s grows from $\theta(\text{const.})$ to $\theta(n)$.

6. CONCLUSIONS

We have presented a method for converting a number N from the weighted binary representation to RNS representation and viceversa. This operation is needed when processors based on RNS arithmetic are chosen for their characteristics of speed, simplicity and easy error recovery. A structure to implement this method has been proposed, which is well suited for integration on very large scale, since it features high modularity and regularity. Upper bounds on the complexity of the conversion problem have been also derived, under several hypotheses on the RNS parameters, separately for area and time. Further work is necessary to find the lower bounds to the problem $A = \Omega(f(n))$ and $T = \Omega(g(n))$; they would allow to evaluate the degree of optimality of our solution. However the relations obtained for A and T are useful to evaluate the convenience of adopting RNS arithmetic in VLSI systems, related to the complexity of the problem they must solve.

REFERENCES

/1/ G.A. Jullien: "Residue Number Scaling and Other Operations Using ROM Arrays", IEEE Trans. on Comp., Vol. C-27, n° 4, April 1978, pp. 325-336.

/2/ W.K. Jenkins and B.J. Leon: "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters", IEEE Trans. on Circuits and Systems, Vol. CAS-24, No. 4, April 1977, pp. 191-201.

/3/ C.H. Huang, D.G. Peterson, H.E. Rauch, J.W. Teague, D.F. Fraser: "Implementation of a Fast Digital Processor Using the Residue Number System", IEEE Trans. on Circuits and Systems, Vol. CAS-28, No. 1, January 1981, pp. 32-38.

/4/ M.A. Soderstrand: "A High-Speed Low Cost Recursive Digital Filter Using Residue Number Arithmetic", IEEE Proc., Vol. 65, No. 7, July 1977, pp. 1065-1067.

/5/ W.K. Jenkins: "Techniques for Residue-to-Analog Conversion for Residue-Encoded Digital Filters", IEEE Trans. on Circuits and Systems, Vol. CAS-25, No. 7, July 1978, pp. 555-562.

/6/ A. Baraniecka and G.A. Jullien: "On Decoding Techniques for Residue Number System Realizations of Digital Signal Processing Hardware", IEEE Trans. on Circuits and Systems, Vol. CAS-25, No. 11, November 1978, pp. 935-936.

/7/ C.D. Thompson: "A Complexity Theory for VLSI", Ph.D. Thesis, Carnegie-Mellon University, Computer Science Department, August 1980.

/8/ R.P. Brent and H.T. Kung: "The Area-Time Complexity of Binary Multiplication", Technical Report CMU-CS-79-136, July 1979.

/9/ C. Mead and L. Conway: Introduction to VLSI Systems, Addison Wesley Reading, MA, 1980.

/10/ N.S. Szabo and R.J. Tanaka: Residue Arithmetic and its Application to Computer Technology, McGraw-Hill, New York, 1967.

/11/ R.B. Johnson, Jr: "The Complexity of a VLSI Adder", Information Processing Letters", Vol. 11, No. 2, October 1980, pp. 92-93.

		DIRECT CONVERSION		REVERSE CONVERSION	
s	log m	A	T	A	T
θ(const.)	θ(n)	$O(n^2 \log n)$	$O(\log^2 n)$	$O(n^2 \log n)$	$O(\log^2 n)$
$\theta(\log n)$	$\theta(n/\log n)$	$O(n^2 \log n)$	$O(\log^2 n)$	$O(n^2 \log n)$	$O(\log^2 n)$
$\theta(n/\log n)$	$\theta(\log n)$	$O(n^2 \log \log n)$	$O(\log n . \log \log n)$	$O(n^2 \log n)$	$O(\log^2 n)$
$\theta(n)$	θ(const.)	$O(n^2 \log \log n)$	$O(\log n)$	$O(n^2 \log n)$	$O(\log^2 n)$

TABLE 1

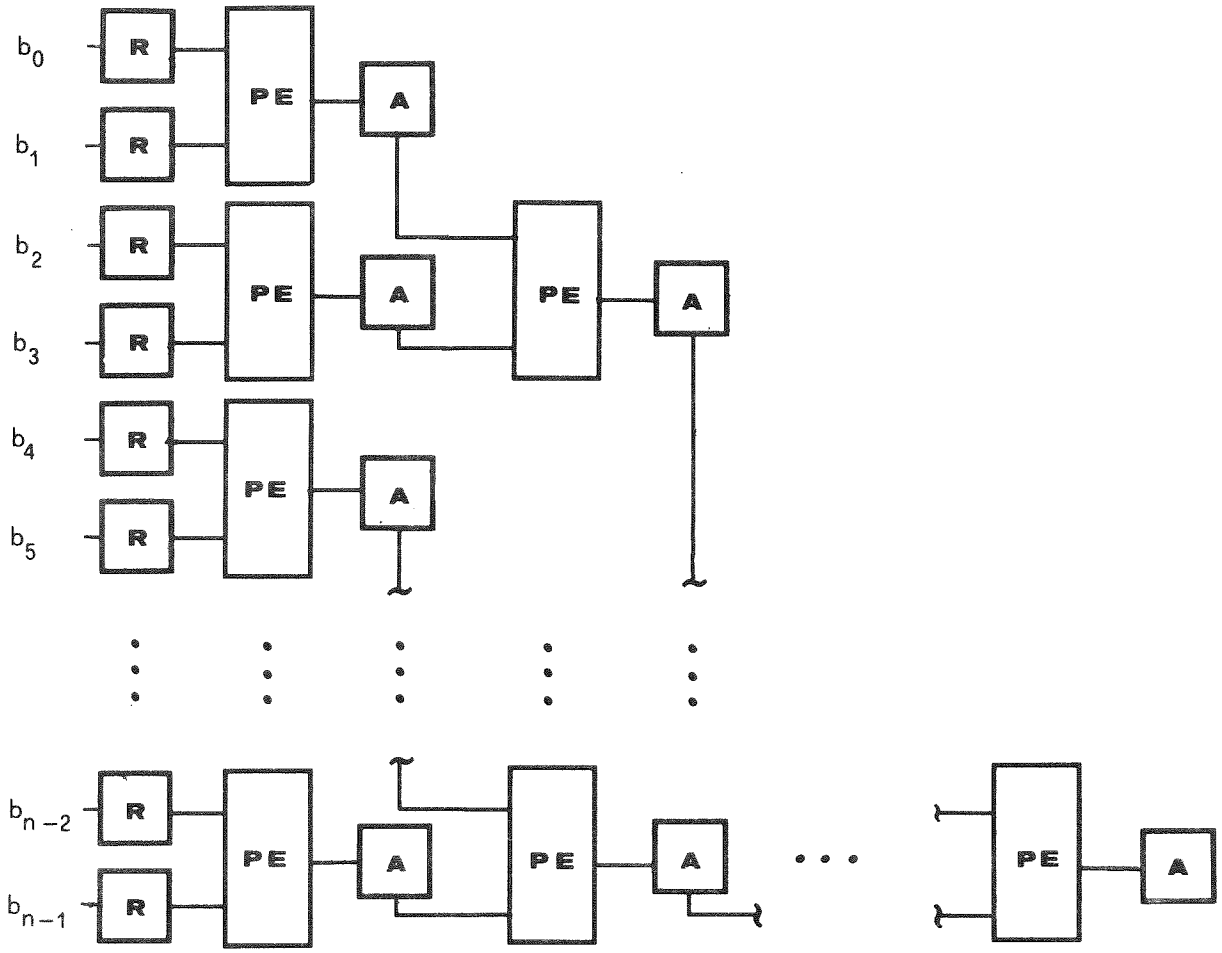

Fig. 1

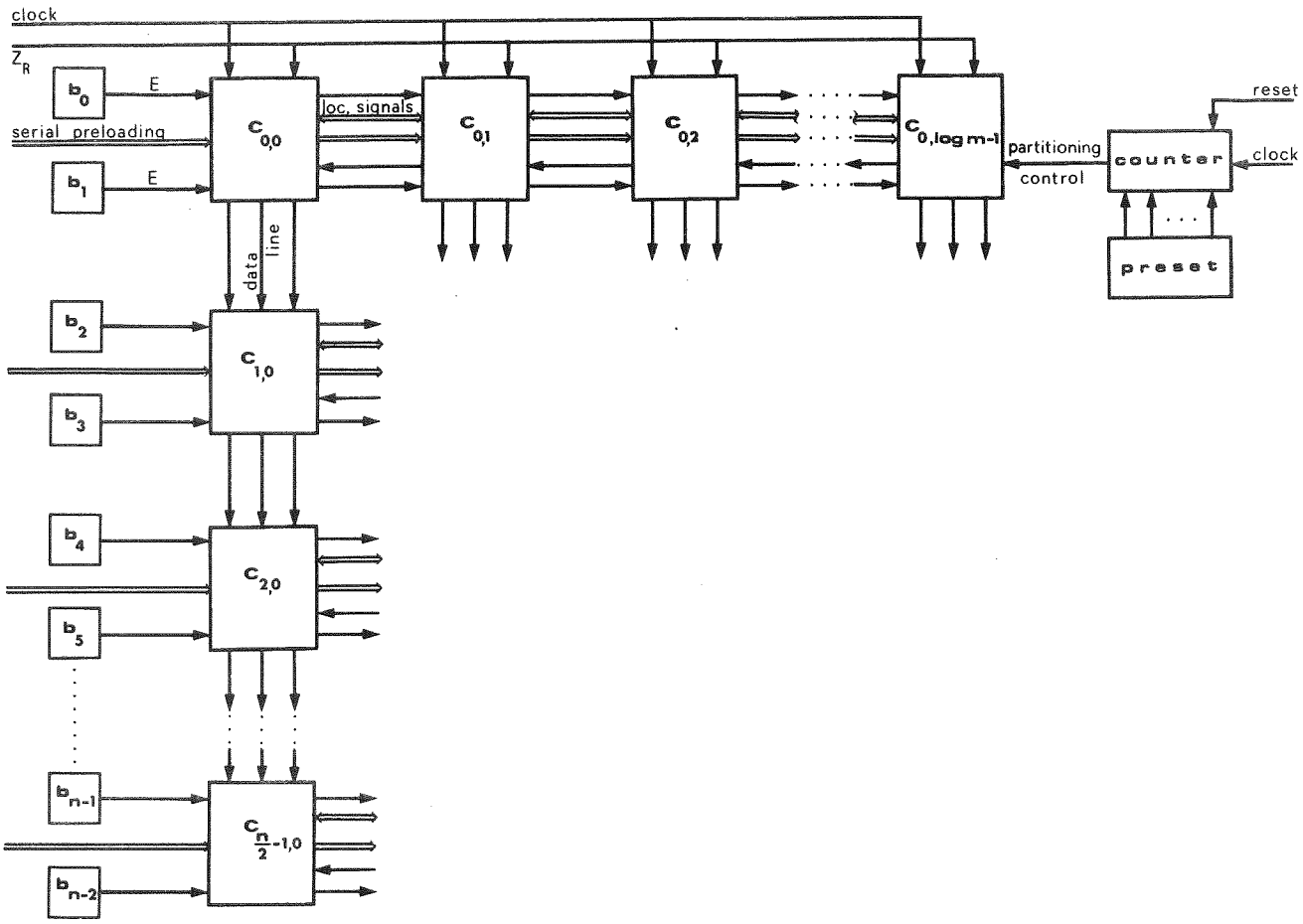
clock
Z_R
b_0
E
serial preloading
b_1
E
$C_{0,0}$
loc. signals
$C_{0,1}$
$C_{0,2}$
$C_{0,\log m-1}$
partitioning
control
counter
reset
clock
preset
data line
b_2
b_3
$C_{1,0}$
b_4
b_5
$C_{2,0}$
b_{n-1}
b_{n-2}
$C_{\frac{n}{2}-1,0}$

Fig. 2

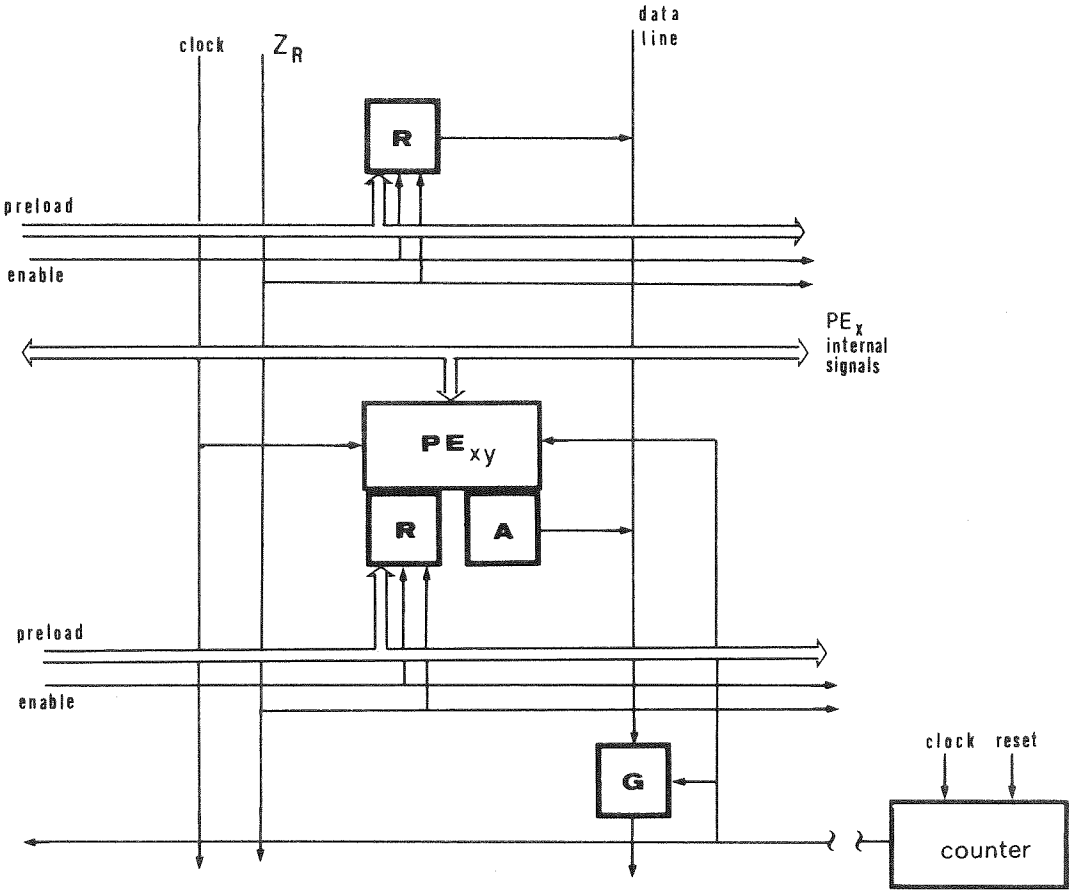
clock
Z_R
data line
R
preload
enable
PE_x internal signals
PE_{xy}
R
A
preload
enable
G
clock
reset
counter

Fig. 3

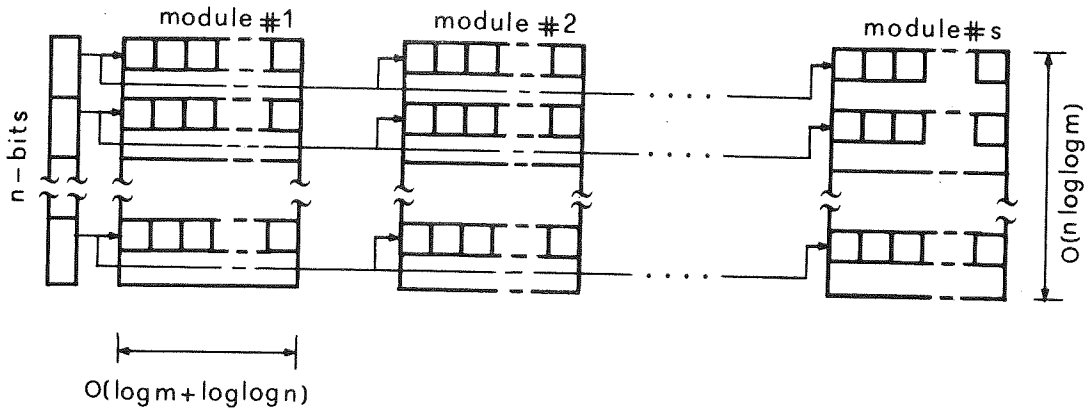

Fig. 4