Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

A FAST DIGITAL CIRCUIT FOR ITERATIVE ADDITIONS IN HNS

Giuseppe Alia and Enrico Martinelli

Nota interna B4-31
Settembre 1994

A FAST DIGITAL CIRCUIT FOR ITERATIVE ADDITIONS IN HNS

Giuseppe Alia* and Enrico Martinelli‡

ABSTRACT

Repeated modular additions and overflow detection are practicable in Hybrid Number Systems. In this paper, an adding, overflow-detecting procedure is described and evaluated by statistical methods; a circuit is proposed allowing a mean addition time less than 8.9 gate delays for numbers having a magnitude order normally distributed in $[-2^{33}, 2^{33}-1]$.

KEYWORDS

Combinational problems, computational complexity, computer arithmetic, weighted and residue number systems, overflow detection, fast accumulator.

---

* Dipartimento di Ingegneria dell'Informazione - Facoltà di Ingegneria dell'Università di Pisa - Via Diotisalvi, 2 - 56126 Pisa - Italy

‡ Istituto di Elaborazione dell'Informazione del Consiglio Nazionale delle Ricerche - Via S. Maria, 46 - 56126 Pisa - Italy

1 - INTRODUCTION

Recently, RNS arithmetic received a considerable attention. However, lengthy intermodular operations are necessary to detect sign or overflow and to compare magnitudes. Many solutions have been proposed to speed up intermodular operations [1, 2, 3, 4, 6]. Alternatively, several authors tried to add a "magnitude index" to the residue representation [5, 7]. This approach is reconsidered here recalling the definition of Redundant Hybrid Number System.

An adding, overflow-detecting procedure, strongly reducing the need for intermodular operations, is described and evaluated by simulation in terms of mean rate of intermodular operations. This procedure could be implemented fruitfully by a device cooperating with a CPU when massive quantities of data must be accumulated, such as statistical computations. In particular, even if an optimal design was not aimed, a circuit was devised for a specific-size RHNS, featuring a mean response time from 8.9 down to about 8 gate delays/addition, for iterative additions in the range $[-2^{33}, 2^{33}-1]$, against 14 gate delays for traditional 32 bit adders, when integers have magnitude order values normally distributed, with the maximum value of probability selected in the range from ≈3% to 1/128 ≈.8% (uniform distribution).

2 - ADDITION AND OVERFLOW DETECTION IN HNS

Residue addition and multiplication are very fast, but comparison and overflow detection require the knowledge of the magnitude of results and time consuming intermodular operations are necessary. In a Hybrid Number System (HNS) an integer X is represented as $\{R_X, W_X\}$, with $R_X = |X|_\mu$, $W_X = \lfloor X/\mu \rfloor$ and assuming that a) $-P \le W_X < P$, where P is a positive integer, locates X into the interval $[W_X \cdot \mu, (W_X+1)\cdot\mu)$ and is represented in a *weighted* notation;

b) R_X is represented in a *residue system* of range μ. Then $X = W_X \cdot \mu + R_X$.

In this section the properties of addition and overflow detection in HNS are recalled from [11, 12] and well tuned procedures are presented for adding and normalizing in redundant HNS. Consider t integers $X_1 \equiv \{R_{X_1}, W_{X_1}\}, \ldots, X_t \equiv \{R_{X_t}, W_{X_t}\}$. Their sum $S = \sum_{i=1,t} R_{X_i} + \mu \cdot \sum_{i=1,t} W_{X_i}$ has the representation $S \equiv \{R_S, W_S\}$, where $R_S = \left| \sum_{i=1,t} R_X \right|_\mu$ and $W_S = \lfloor S/\mu \rfloor = \sum_{i=1,t} W_{X_i} + \left\lfloor \left(\sum_{i=1,t} R_{X_i} \right) / \mu \right\rfloor = \sum_{i=1,t} W_{X_i} + \delta_S$ with

$$\delta_S \leq \lfloor (t(\mu - 1)) / \mu \rfloor \qquad (1)$$

To avoid the computation of δ_S the residue range μ is extended by a set of moduli whose product m_R is such that μm_R contains the sum of the residue parts of t operands; μ and m_R are relatively prime. The representation of X in a *Redundant Hybrid Number System* (RHNS) will be $X \equiv \{\mathcal{R}_X, \mathcal{W}_X\}, 0 \leq \mathcal{R}_X < \mu m_R, \mathcal{W}_X = (X - \mathcal{R}_X)/\mu$, respectively. Denoting $\mathcal{R}'_X = \mathcal{R}_X + k\mu, \mathcal{W}'_X = \mathcal{W}_X - k$, the same integer X can be also represented as $X \equiv \{\mathcal{R}'_X, \mathcal{W}'_X\}, 0 \leq \mathcal{R}'_X < \mu m_R, \mathcal{W}_X = (X - \mathcal{R}'_X)/\mu$, with k is a positive integer, its maximum value depending on $(\mu m_R - 1) - \mathcal{R}_X$. An RHNS representation will coincide with the HNS representation if $0 \leq \mathcal{R}_X < \mu$ and $\{\mathcal{R}_X, \mathcal{W}_X\} = \{R_X, W_X,\}$ will be referred to as the *normalized representation.*

Consider the addition in RHNS. Given the normalized representations of t integers in $[-P\mu, P\mu)$: $X_1 \equiv \{R_{X_1}, W_{X_1}\}, \ldots, X_t \equiv \{R_{X_t}, W_{X_t}\}$, m_R must satisfy the condition $\mu m_R - 1 \geq t(\mu - 1)$ to obtain: $\mathcal{R}_S = \sum_{i=1,t} \mathcal{R}_{X_i} = \sum_{i=1,t} R_{X_i}$, $\mathcal{W}_S = \sum_{i=1,t} \mathcal{W}_{X_i} = \sum_{i=1,t} W_{X_i}$.

It is easy to see that the following inequalities hold:

$$0 \le \mathcal{R}_S \le t(\mu\text{-}1),\ -tP \le \mathcal{W}_S \le t(P\text{-}1) \tag{2}$$

On the contrary, from the condition $\mu m_R\text{-}1 \ge t(\mu\text{-}1)$, for given m_R

$$t^* = m_R + \lfloor (m_R - 1) / (\mu - 1) \rfloor \tag{3}$$

represents an upper bound to the number of normalized integers which can be added without overflow from the extended residue range. Then, if conditions (2) hold and the weighted range is assumed as $[-tP, t(P\text{-}1)]$, an RHNS representation of the sum is $S \equiv \{\mathcal{R}_S, \mathcal{W}_S\}$. Overflow from $[-P\mu, P\mu)$ can be checked by reconstructing the normalized weighted part W_S, as $W_S = \mathcal{W}_S + \delta_S$; an overflow will be detected iff $W_S < -P$ or $W_S > P\text{-}1$, that is $\mathcal{W}_S < -P\text{-}\delta_S$ or $\mathcal{W}_S > P\text{-}1\text{-}\delta_S$. These conditions can be restated only as sufficient conditions, by means of (1):

$$\mathcal{W}_S < -P - \lfloor t(\mu - 1) / \mu \rfloor \quad \text{or} \quad \mathcal{W}_S > P - 1. \tag{4}$$

It follows that testing for overflow requires the knowledge of the number t of added normalized representations. Then a control part C_X is added to the RHNS representation. It will be assumed that: a) any normalized representation has a control part $C_X = 1$; b) the control part C_S of the sum is obtained by adding the control parts of the operands; c) all operands have normalized representations or they result as a sum of normalized representations. Conditions (4) will take the form $\mathcal{W}_S < -P - \lfloor C_S(\mu - 1) / \mu \rfloor$ or $\mathcal{W}_S > P\text{-}1$.

From (3), C_S cannot exceed the quantity $C_{max} = m_R + \lfloor (m_R - 1) / (\mu - 1) \rfloor$. These relations are further simplified if the redundancy does not exceed the duplication of the

residue range ($m_R < \mu$); in that case one obtains

$$\mathcal{W}_S < -P - (C_S + \lfloor -C_S/\mu \rfloor) = -P - C_S + 1 \quad \text{or} \quad \mathcal{W}_S > P - 1 \tag{5}$$

and $C_{max} = m_R$. Previous considerations prove the following procedures.

ADDING PROCEDURE (mostly parallel)

1. Add residue, weighted and control parts of operands to obtain $\mathcal{R}_S$, $\mathcal{W}_S$, C_S;
2. If condition (5) is satisfied an overflow is detected, else
3. If $C_S \leq C_{max}$ then $S \equiv \{\mathcal{R}_S, \mathcal{W}_S, C_S\}$, else $\mathcal{R}_S$ may exceed the extended residue range and a preliminary normalization of operands is required before repeating the procedure.

NORMALIZING PROCEDURE (intermodular)

1. Convert residue digits $|\mathcal{R}_X|_{m_i}$ back to the integer $\mathcal{R}_X$ in the range $[0, \mu m_R\text{-}1]$;
2. Update the value $\mathcal{W}_X$ by an increment $\lfloor \mathcal{R}_X/\mu \rfloor$;
3. Restore the correct value R_X in $[0, \mu\text{–}1]$ and evaluate $|R_X|_{m_i} = |\mathcal{R}_X - \lfloor \mathcal{R}_X/\mu \rfloor \mu|_{m_i}$.

3 - SOME CONSIDERATIONS ON PERFORMANCE

The model selected here to generate sequences of additions assumes that sequences of normalized representations of integers in the range $[-P\mu, P\mu)$ with a given distribution are summed up. Each sequence ends when an overflow is detected or a normalization phase is necessary. Consequently, all the sequences which end with a normalization request have the same length $C_{max}+1$. The probability p_N of a normalization request has been computed for several probability distributions, for $m_R = 2^i$, $i = 1, ..., 7$ and for $2P = 2^j$, $j = 1, ..., 6$. The reason

for considering $2P$ is that overflow detection is related to this range when $m_R < \mu$ is assumed. Consequently, for computational purposes, it is sufficient to process the weighted parts of numbers only. In this case, Gauss-like distributions for the weighted parts of operands were chosen, that is $p(\mathcal{W}) = A \cdot \exp\left(-B\frac{(\mathcal{W}+0.5)^2}{(2P)^2}\right)$, with the following conditions: $-P \leq \mathcal{W} \leq P-1$, i.e. the mean $\overline{\mathcal{W}} = -0.5$, and $2P \leq 64$ to limit the number of simulation experiments. Constant A was chosen to normalize the distribution, while B was given values in the set {0, 15, 30, 45}, to produce a wide set of shapes, as shown in Figure 1 for $2P = 64$. The behaviour of p_N is shown in Figures 2a-2d for the same values of B and of the weighted range $2P$ as in Figure 1. It is immediate to see that the number of normalizations reduces to a negligible rate as soon as m_R takes non trivial values. Moreover:

- the probability p_N equals $1/m_R$ for $2P = 2$, whatever the distribution shape. This occurs as the weighted part of operands can take only values -1 and 0, and neither positive nor negative overflow can be detected before the m_Rth addition is performed.

- p_N decreases as $2P$ increases. This happens since the greater the value of $2P$, the more likely overflow detection is.

- p_N decreases for increasing values of m_R and the value $1/m_R$ represents a maximum for p_N. In fact, at least m_R additions can be performed without normalization requests.

- As parameter B increases, p_N tends to its maximum $1/m_R$, for any value $2P$. In fact, as B increases, the probability of weighted values near zero increases and, correspondingly, the probability of overflow detection decreases.

In order to perform a quantitative evaluation in an example of specific size, let us consider an RHNS with the weighted part in [-64, +63], the residue part expressed by means moduli m_1

= 113, m_2 = 121, m_3 = 125 and m_4 = 126, and by the redundant modulus m_R = 127. As $\prod_{i=1}^{4} m_i$ is beyond $215 \cdot 10^6$, the RHNS can represent integers in $[-2^{33}, 2^{33}-1]$. To evaluate sums in this range efficiently, let us represent each residue digit modulo m_i by means of an RNS of only two moduli m_A = 16 and m_B = 15, so that residue digits $r_{i,j}$, $i \in \{1, 2, 3, 4, R\}$, $j \in \{A, B\}$, can be processed by five 4-bit binary adders and by five pairs of 4-bit binary adders followed by a 2×1, 4-bit multiplexer with output clear, as shown in Figure 3. Consequently, any sum in the range $[-2^{33}, 2^{33}-1]$ requires a total of 8 gate delays (Δg), versus 14 Δg necessary to 32-bit binary adders. (We assume an addition time of 6+4(h-1) Δg for operands of 4^h bits, exploiting carry look ahead techniques [8]). On the other hand it may be necessary to normalize operands at a rate depending on the value of parameter B. From the shape of p_N in Figs. 2a÷2d normalization rates for this case are likely to be as shown in Table I, where the probability values for the mean magnitude order of integers $\overline{W}$ = -0.5 are also reported ;

B	$p_{\overline{W}}$	normalizations/additions
45	0.02957	$\approx 2\times10^{-3}$
30	0.02414	$\approx .6\times10^{-3}$
15	0.01718	$\approx .3\times10^{-4}$
0	0.00781	$\approx 10^{-10}$

Table I

Referring to Figure 4, let's evaluate the cost of normalization without taking care of the possibility of designing special hardware optimally for this operation. First, 5 pairs of 4-bit residue digits must be converted to five 7-bit residue digits. Furthermore, the 7-bit residue digits must be converted to obtain $\mathcal{R}_X$, in $[0, \mu m_R)$, according to the Chinese Remainder Theorem $\mathcal{R}_X = \left| \sum_{i \in \{1, \ldots, 4, R\}} \hat{m}_i \cdot \left| r_i / \hat{m}_i \right|_{m_i} \right|_{\mu m_R}$ where $\hat{m}_i = \prod_{s \in \{1, \ldots, 4, R\}, s \neq i} m_s$ and

$\left|1/\hat{m}_i\right|_{m_i}$ is the multiplicative inverse of $\hat{m}_i$ modulo m_i. For the whole conversion from $r_{i,j}$, $j \in \{A, B\}$ to $\hat{m}_i \left|r_i/\hat{m}_i\right|_{m_i}$ five groups of nine 256×4 ROM's, obtaining a word length of 36 bits sufficient to represent numbers in $[0, \mu m_R)$, can be used. ROM's such as MCM10150AL with an access time as low as about 10 Δg can be chosen. The summation of these five values can be carried out in less than 3×(14+2+14) = 90 Δg, including a further subtractions of μm_R, possibly required after any summation to keep the result modulo μm_R. Thus about 100 Δg are necessary to obtain the value of $\mathcal{R}_X$ from $r_{i,j}$. Moreover, the increment of the magnitude index, which is less than m_R, must be evaluated by $\lfloor \mathcal{R}_X/\mu \rfloor$, that can be rewritten as $(\mathcal{R}_X - \left|\mathcal{R}_X\right|_\mu)/\mu = \left|\left(\mathcal{R}_X - \left|\mathcal{R}_X\right|_\mu\right)\right|_{m_R} \cdot \left|\frac{1}{\mu}\right|_{m_R}\Big|_{m_R}$. To evaluate $\left|\mathcal{R}_X\right|_\mu$, a mod μ operation can be carried out as in [9]. Since $\mathcal{R}_X$ is 35 bits long and assuming that fast multipliers are available, consuming [8] about $\lceil 2\times(\log_2 n-1)/(\log_2 3-1)\rceil+2+4\times\lceil 1/2\times\log_2 n\rceil$ Δg, about 87 Δg are required to produce the value $\left|\mathcal{R}_X\right|_\mu$. Furthermore, a 35 bit subtraction (≈ 14 Δg), a mod m_R operation (≈ 69 Δg) and a mod m_R multiplication [10] which takes about 77 Δg must be performed to obtain $\lfloor \mathcal{R}_X/\mu \rfloor$. Further 10 Δg to increment the magnitude index $\mathcal{W}_X$ would be accounted for, but such an operation can be performed in parallel with another one, so a total of about 247 Δg must be spent to evaluate W_X from $\mathcal{R}_X$. Finally, the new residue r_R is evaluated as $\left|R_X\right|_{m_R} = \left|\left|\mathcal{R}_X\right|_{m_R} - \left|\lfloor \mathcal{R}_X/\mu \rfloor\right|_{m_R} \cdot \left|\mu\right|_{m_R}\right|_{m_R}$; in fact, residue digits r_i, $i = 1, \ldots, 4$, remain unchanged, as a multiple of μ is subtracted from $\mathcal{R}_X$. This computation consists of a 7×7 binary multiplication (≈ 17 Δg) and of a 14 bit subtraction (≈ 10 Δg) from $\left|\mathcal{R}_X\right|_{m_R}$. The

obtained value must be converted to residue digits $|R_X|_{m_R}$ by means of a mod m_R converter in about 59 Δg. Thus about 86 Δg are necessary to conclude the normalization. Before going on to add input data, r_R must be reconverted into a pair of residue digits, mod 15 and mod 16; this can be obtained by two 128×4 ROM's. Therefore, the normalization phase globally amounts to about 100+247+86+10 = 443 Δg. Comparing this value with normalization rates of Table I and with the saving amount (6 Δg/addition), it can be concluded that, in the worst simulated case B = 45, the mean addition time, including the normalization, is about 8.9 Δg against 14 Δg for traditional 32 bit long adders. Note that in the sketched structure an increment of the residue range obtained by adding further moduli of the same length slightly affects the normalization time, increasing the depth of the adder tree and the length of its adders. In particular, adding three 7-bits moduli in the example does not modify the normalization time at all, while the range of integers is enhanced by a factor of about 2^{20}.

REFERENCES

[1] Alia, G. and Martinelli, E. *A VLSI algorithm for direct and reverse conversion from weighted binary number system to residue number system*, IEEE Trans. Circuits and Systems, Vol. CAS-31, 1984, pp. 1033-1039.

[2] Alia, G., Barsi, F. and Martinelli, E. *A fast VLSI conversion between binary and residue systems*, Information Processing Letters, Vol 18, n. 3, 1984, pp. 141-145.

[3] Bayoumi, M. A., Jullien, G. A. and Miller, W. C. *A VLSI implementation of residue adders*, IEEE Trans. Circuits and Systems, Vol. CAS-34, 1987, pp. 284-288.

[4] Jullien, G. A. *Residue number scaling and other operations using ROM arrays*, IEEE Trans. Comput., Vol. C-27, 1978, pp. 325-336.

[5] Rao, T. R. N. and Trehan, A. K. *Binary logic for residue arithmetic using magnitude index*, IEEE Trans. Comput., Vol. C-19, 1970, pp. 752-757.

[6] Taylor, F. J. and Huang, C. H. *An autoscaler residue multiplier*, IEEE Trans. Comput., Vol. C-31, 1982, pp. 321-325.

[7] Ulman, Z. D. *Sign detection and implicit-explicit conversion of numbers in residue arithmetic*, IEEE Trans. Comput., Vol. C-32, 1983, pp. 590-594.

[8] N. R. Scott, *Computer Number Systems and Arithmetic*, Prentice-Hall International Editions, Englewood Cliffs, NJ, 1985.

[9] Alia, G. and Martinelli, E. *A VLSI Structure for X mod m Operations,* The Journal of VLSI Signal Processing, 1, 1990, pp. 257-264.

[10] Alia, G. and Martinelli, E. *A VLSI Modulo m Multiplier*, IEEE Trans. Comput., Vol. C-40, 1991, pp.873-878.

[11] Alia, G., Barsi, F. and Martinelli, E., *Addition and Overflow Handling in a Class of Redundant RNS with Magnitude Index*, Technical Report IEI-CNR B4-33, Dec. 1987.

[12] Barsi, F., Martinelli, E., *A VLSI Architecture for RNS with MI Adders,* INTEGRATION, the VLSI Journal, Vol. 11, 1991, pp. 67-83.

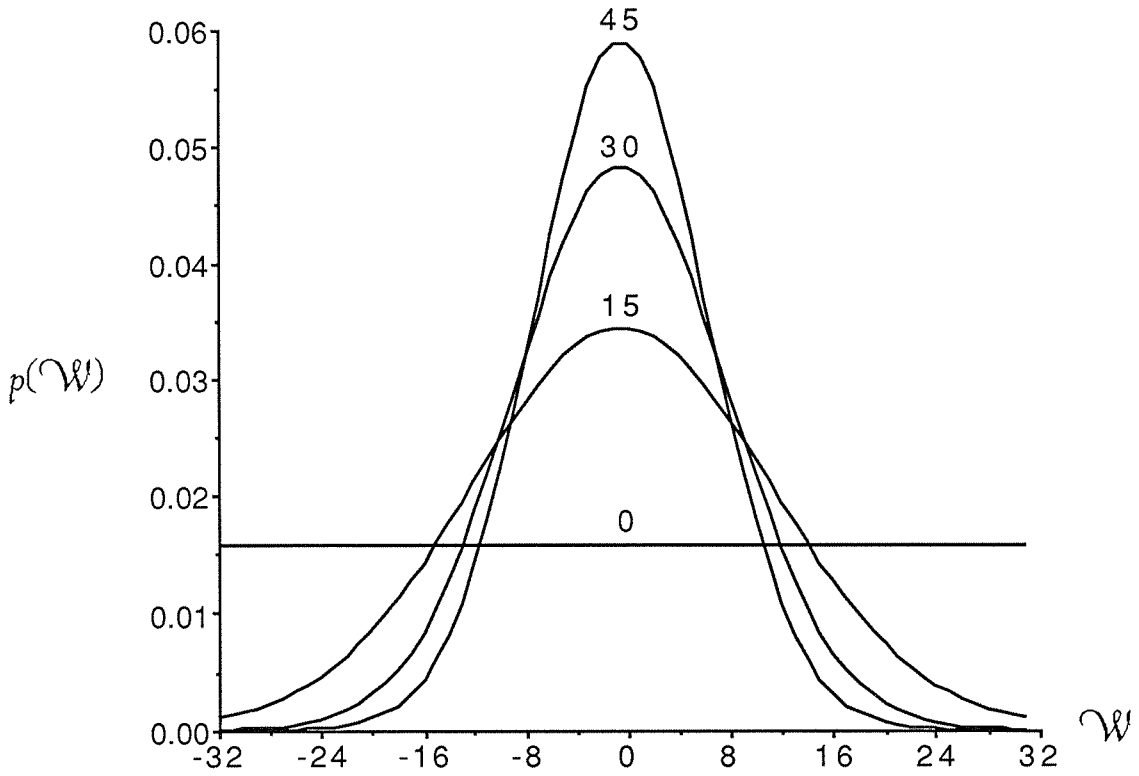

Fig. 1 - Probability distribution of the weighted part $\mathcal{W}$ of numbers

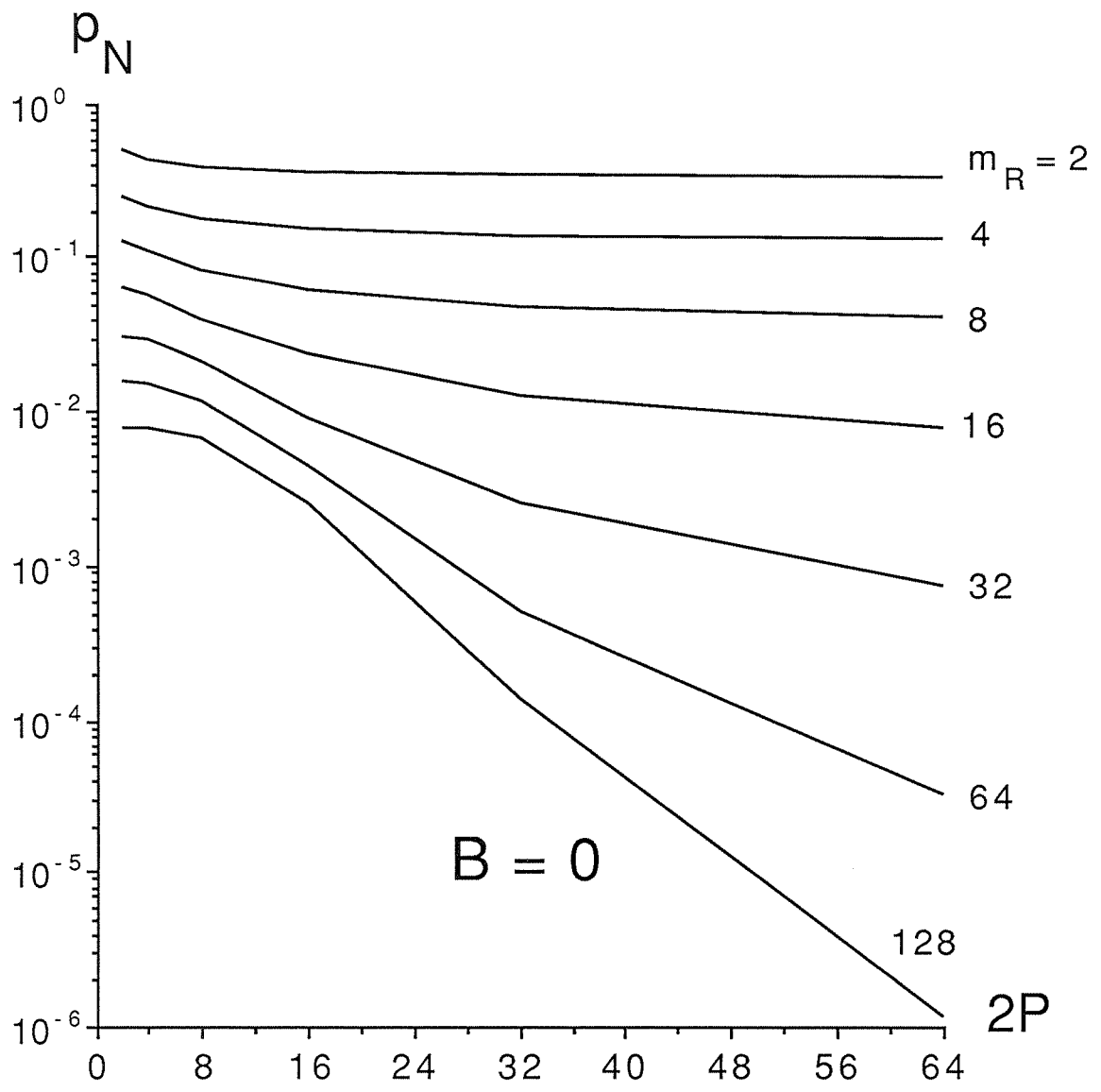

Fig. 2a - Shapes of the normalization request probability

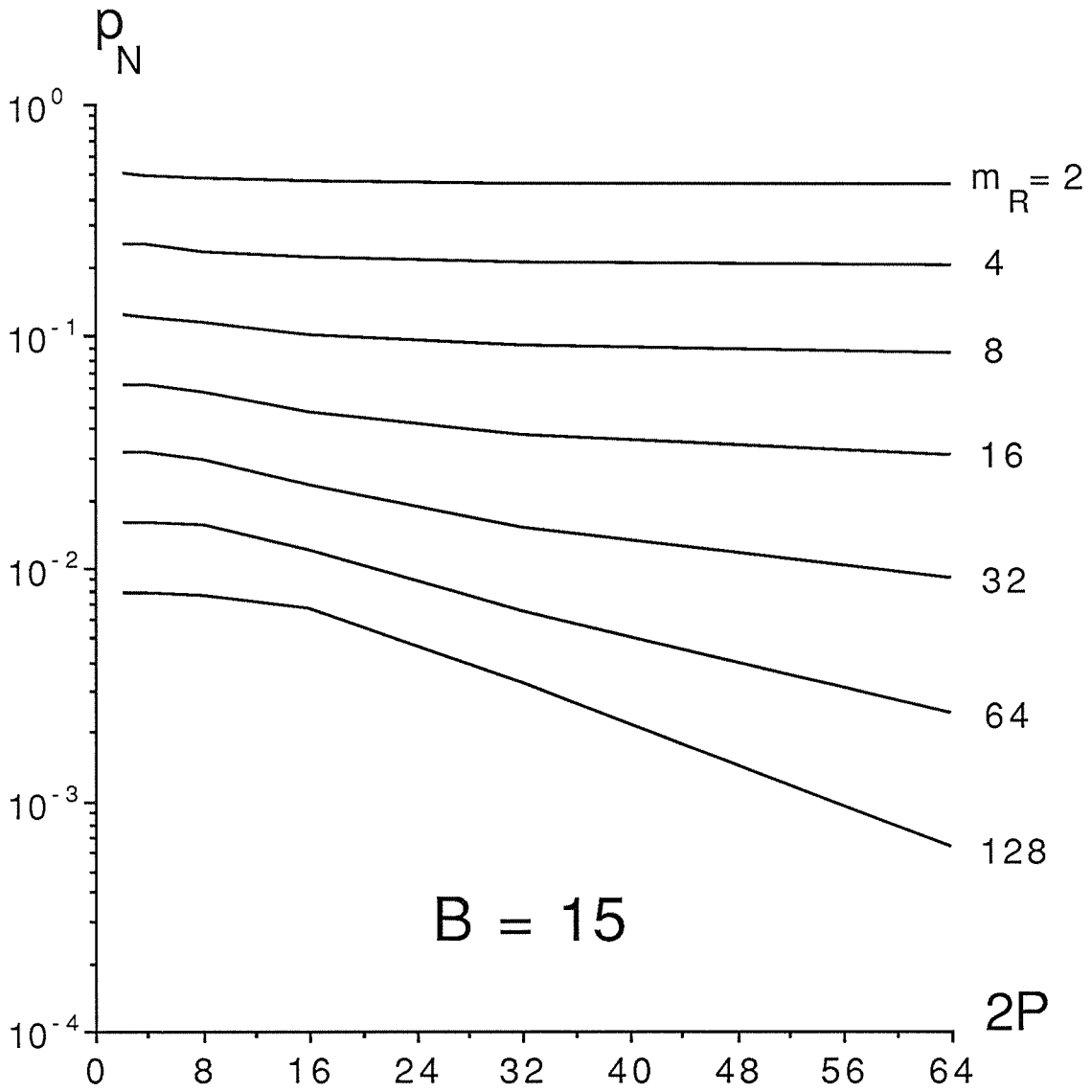

Fig. 2b (continued)

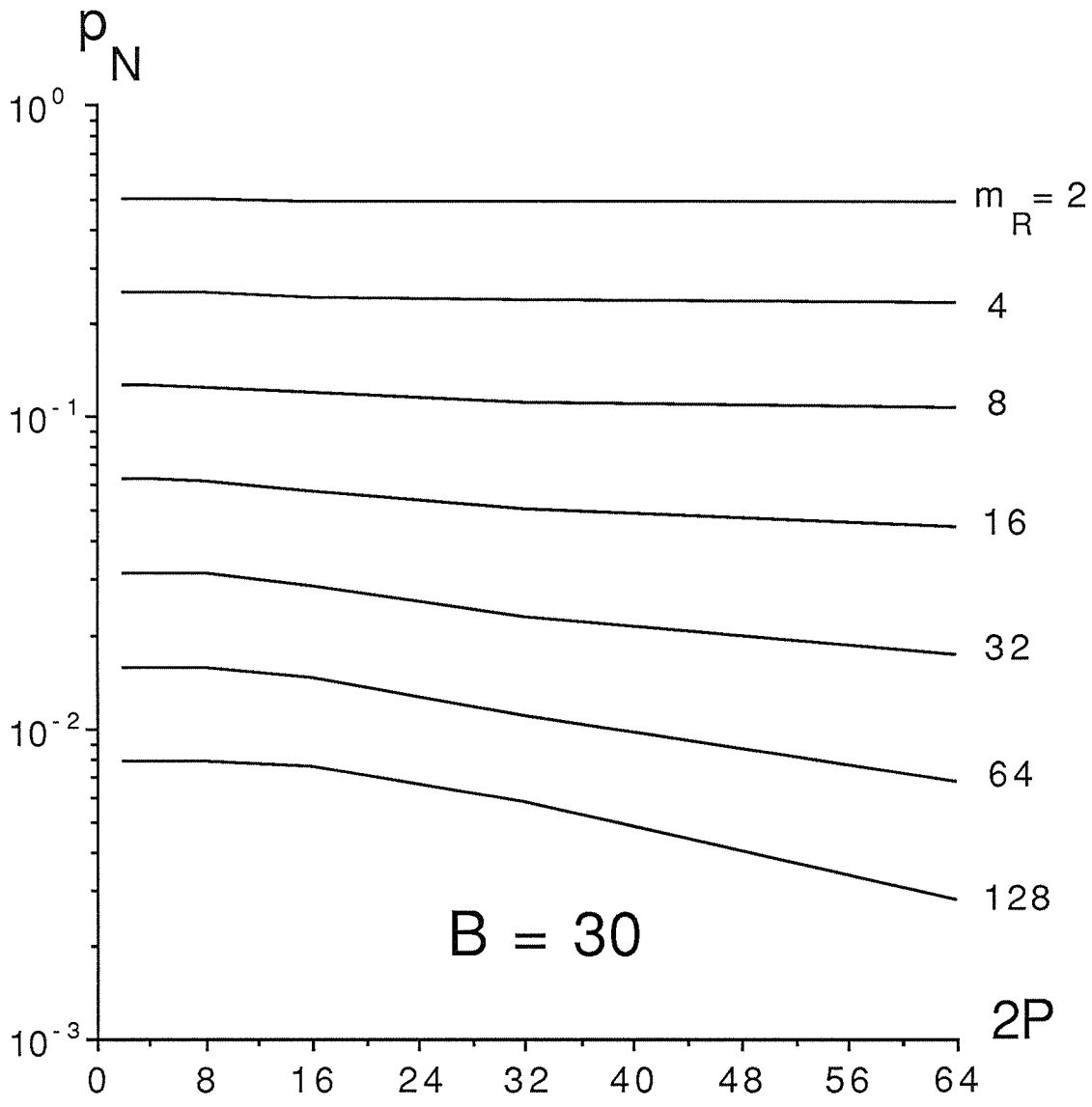

Fig. 2c (continued)

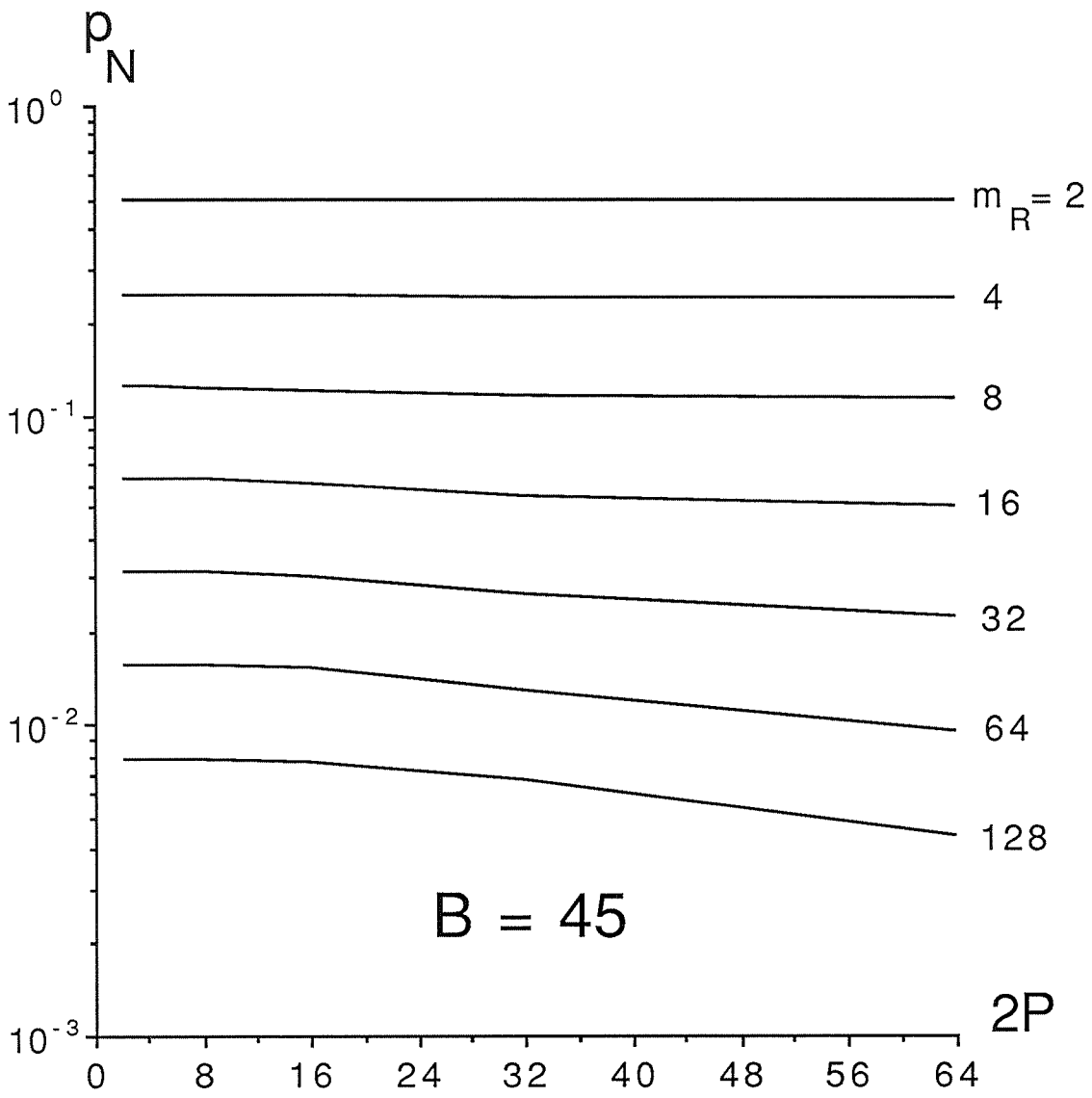

Fig. 2d - (continued)

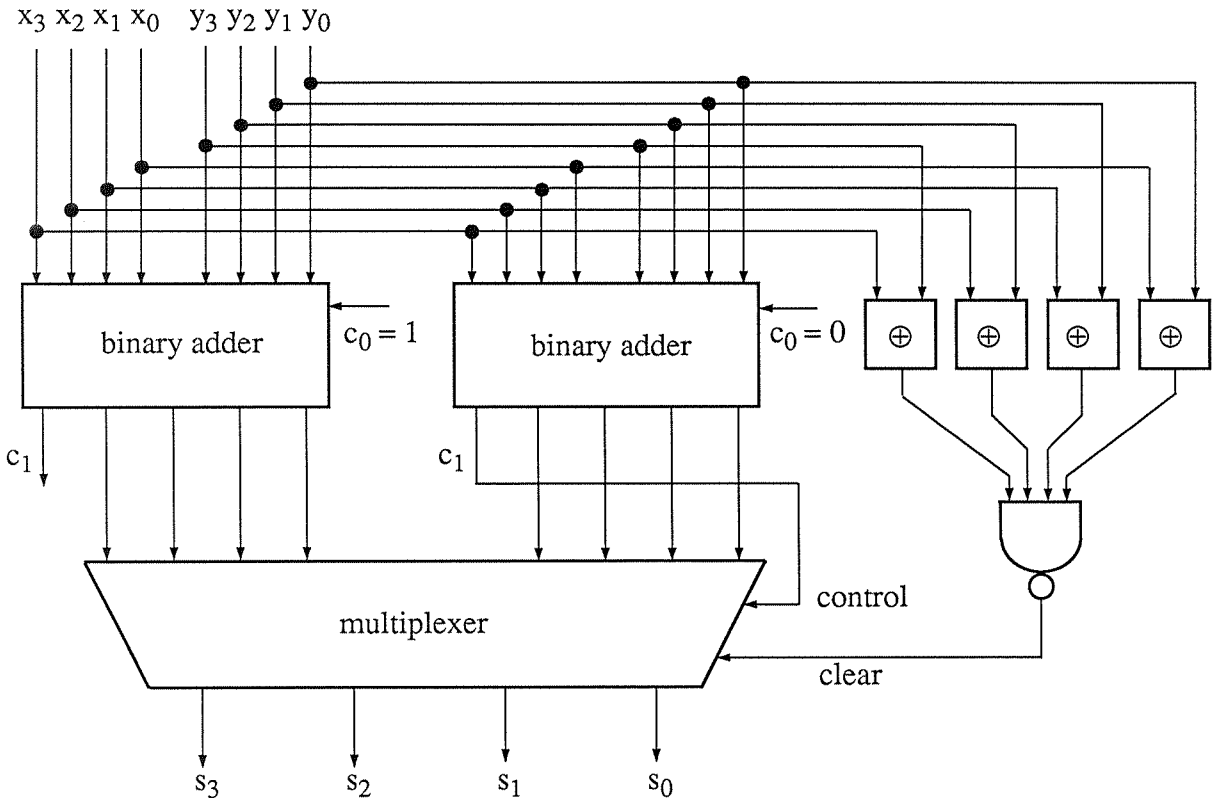

Fig. 3 - A scheme for a mod 15 adder

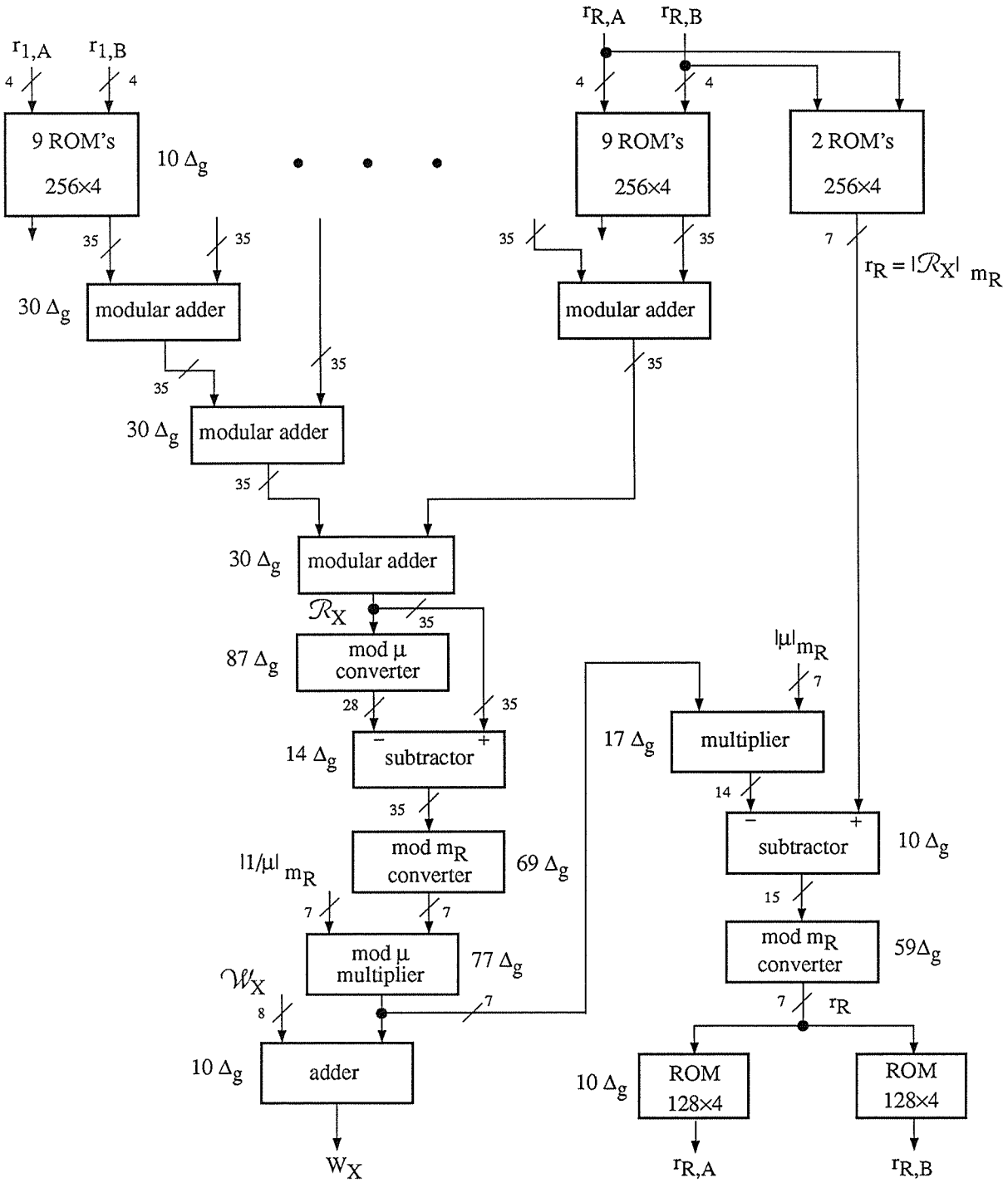

Fig. 4 - Layout of the normalization structure