MULTIPLE ADDITION AND PARALLEL COUNTER IN GENERALIZED BINARY AND NEGABINARY SYSTEMS

L. Bhuyan and Dharma P. Agrawal

Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202

Abstract

This paper explores the possibility of utilizing counters for reducing the partial products generated from multiplication of two numbers represented in a generalized binary system. Algorithm for multioperand addition of Koren's generalized number system are worked out. A carry look-ahead adder is developed and hardware aspects of possible counters for this class of binary number system are discussed. Negabinary counters are also introduced in line with those proposed for binary multiplication by Stenzel et al.

I. Introduction

In a binary multiplication, the complexity and time increase with increased number of bits. Generation of a matrix of partial products with subsequent reducton of the matrix by means of pseudo-adders has been looked into by several authors. Matrix generation and compression schemes are much faster for larger operands. Traditional forms of operand reduction scheme use full adders to reduce the matrix. Dadda[1] introduced the notion of a (c,d) counter as a combinatorial network which receives c bits of equal weight as input and produces a corresponding d bit sum as output. This class of counters were extended recently by Stenzel et al.[2] to include counters which receive columns from several successive radix positions of the input and their sum is produced by taking the corresponding weights into account. They have successively implemented a 24x24 bit multiplier incorporating ROMs as (5,5;4) table lookup counters. This class of counters gives rise to a compact, high speed and less expensive parallel multiplication scheme.

On the other hand, considerable amount of work has been done in different number systems and in deriving arithmetic algorithms for them. Because of the unique advantage of representing the positive and negative numbers without a sign bit, several authors have looked into the negative radix number systems. Sankar et al.[3] described algorithms for basic arithmetic operations in negative base. Some simpler, faster and more general algorithms were also proposed for negative radix number systems.[4] Yuen[5] offered a new (8,-4,-2,1) representation for decimal numbers to simplify the carry generation process in B.C.D. addition. Zohar[6] has concluded that the negabinary system has irrefutable advantage over the conventional binary system in special applications such as digital filtering and signal processing in general. In a more recent presentation, Koren and Maliniak[7] discussed a unified approach to a broad class of number systems, characterized by $\langle\beta,\Lambda\rangle$ and successfully derived algorithms for different arithmetic operations.

In this paper, an approach has been made to look into possible reduction of partial products in multiplication of two numbers, represented by $\langle 2,\Lambda\rangle$ system. Algorithm for multioperand addition in $\langle\beta,\Lambda\rangle$ system is derived in Section II to make the process feasible. A carry look-ahead adder for $\langle 2,\Lambda\rangle$ system has been developed in Section III for fast addition of final two rows. Section IV discusses the possibility of a generalized binary counter and its hardware implications. Finally in section V, negabinary counters are presented in line with those presented in Ref. 2.

II. Generalized Binary Number System

Koren and Maliniak[7] suggested a generalized number system characterized by the pair $\langle\beta,\Lambda\rangle$, where β is a positive integer and Λ is a vector $(\lambda_{n-1},\lambda_{n-2},\ldots,\lambda_0)$, $\lambda_i\varepsilon\{-1,+1\}$. The algebraic value x of an n-tuple $(x_{n-1},x_{n-2},\ldots x_0)$, $x_i\varepsilon\{0,1,\ldots,\beta-1\}$ is defined by

$$X = \sum_{i=0}^{n-1} \lambda_i x_i \beta^i \qquad (1)$$

A generalized binary system is one for which $\beta=2$, $\lambda_i\varepsilon(-1,+1)$, $x_i\varepsilon\{0,1\}$, $\Lambda = (1,1,..,1)$ for binary; $(1,-1,1,\ldots,-1,1)$ for negabinary; $(-1,1,1,\ldots 1)$ for 2's complement; and $(1,-1,-1,1,\ldots,1,-1,-1,1)$ for 8,-4,-2,1 code as suggested by Yuen and so on.

We are primarily interested to show that the partial products generated by multiplying two $\langle 2,\Lambda\rangle$ numbers can also be reduced by the counter methods.

Algorithm for Multioperand Addition

Considering the ith column of m operands x_i's in radix β and carries c_i's and borrows b_i's from preceding columns,

$$z_i = \lambda_i \Sigma x_i + \Sigma c_i - \Sigma b_i \quad (2)$$

or

$$z_i = \lambda_i s_i + \beta(c_{i+1} - b_{i+1}) + \beta^2(c_{i+2} - b_{i+2}) + \ldots + \beta^j(c_{i+j} - b_{i+j}) \quad (3)$$

where,

j is the no. of carries or borrows from the ith column, and

s_i is the sum bit.

This is subject to the following condition,

$$\beta^{j+1} - 1 \geq \sum_m \beta + \sum c_i, \quad c_i \in \{0,1,\ldots,\beta-1\} \quad (4)$$

or,

$$j \geq \frac{\log(1+\Sigma\beta+\Sigma c_i)}{\log \beta} - 1 \quad (5)$$

Restricting the above equations to two carries or two borrows,

$$s_i = (\lambda_i z_i) \bmod \beta \quad (6)$$

Case I

$$z_i \geq \lambda_i s_i$$

$$b_{i+1} = b_{i+2} = 0$$

$$c_{i+1} = \left(\frac{1}{\beta}(z_i - \lambda_i s_i)\right) \bmod \beta$$

$$c_{i+2} = \frac{1}{\beta^2}(z_i - \lambda_i s_i) - \frac{1}{\beta} c_{i+1} \quad (7)$$

Case II

$$z_i < \lambda_i s_i$$

$$c_{i+1} = c_{i+2} = 0$$

$$b_{i+1} = \left(\frac{1}{\beta}(\lambda_i s_i - Z_i)\right) \bmod \beta \quad (8)$$

$$b_{i+2} = \frac{1}{\beta 2}(\lambda_i s_i - Z_i) + \frac{1}{\beta} b_{i+1}$$

For a generalized binary system with $x = \sum_{i=0}^{n-1} \lambda_i x_i 2^i$, $\lambda_i \varepsilon \{-1,+1\}$, $x_i \varepsilon \{0,1\}$ the above set of equations can be modified by substituting $\beta=2$.

III. Carry Look Ahead Adder for <2,Λ> System

Reduction of two operands is usually done by the carry-look ahead adders[8] and hence we consider the design of such adders for <2,Λ> system. To do this, a full adder for two operands with one carry and one borrow inputs is given in Fig. 1a and n such full adders can be connected to give a n-bit parallel adder in <2,Λ> system.

The logical expressions are given by[7],

$$s_i = x_i \oplus y_i \oplus (c_i + b_i) \quad (9)$$

$$c_{i+1} = \gamma_i(x_i+y_i)c_i + \bar{\gamma}_i\bar{x}_i\bar{y}_i c_i + \gamma_i x_i y_i \bar{b}_i$$

$$b_{i+1} = \bar{\gamma}_i(x_i+y_i)b_i + \gamma_i\bar{x}_i\bar{y}_i b_i + \bar{\gamma}_i x_i y_i \bar{c}_i$$

where, x and y are two operands

and $\gamma_i = 0$ for $\lambda_i = -1$

$= 1$ for $\lambda_i = +1$

The above set of equations (9) is not valid for the case $b_i=c_i=1$ and the carries ($\surd$) and borrows (x), generated are shown in Fig. 1b. These equations are also too complex and their implementation in the form of Carry Look Ahead Adder is extremely difficult.

In order to simplify the equations for a Carry Look Ahead Adder, the hypothetical case of $b_i=c_i=1$, which is equivalent to $b_i=c_i=0$, is introduced giving rise to an extra row in the K-map of Fig. 1b. Further simplification is obtained by introducing additional carry-borrow ($\surd$x) null pairs and this leads to a complete K-map drawn in Fig. 1c. The following logical expressions are derived.

$$c_{i+1} = \bar{\gamma}_i\bar{x}_i\bar{y}_i + \gamma_i(x_i+y_i) = \gamma_i\overline{\oplus(x_i+y_i)} = p_i \quad (10)$$

$$b_{i+1} = (Q_i+\bar{c}_i)b_i + Q_i\,\bar{c}_i \quad (11)$$

where $Q_i = \bar{\gamma}_i(x_i y_i + \bar{x}_i\bar{y}_i) + \gamma_i(x_i\bar{y}_i + \bar{x}_i y_i)$

$$= \overline{\gamma_i \oplus x_i \oplus y_i}$$

$$b_{i+1} = (Q_i+\bar{p}_{i-1})b_i + Q_i\,\bar{p}_{i-1}$$

$$= Q_i\,\bar{p}_{i-1} + (Q_i+\bar{p}_{i-1})b_i \quad (12)$$

and $c_{i+1} = p_i$

In terms of the previous output,

$$b_{i+1} = Q_i\bar{p}_{i-1} + Q_{i-1}\bar{p}_{i-2}(Q_i+\bar{p}_{i-1}) + (Q_i+\bar{p}_{i-1})(Q_{i-1}+\bar{p}_{i-2})b_{i-1} \quad (13)$$

Expanding further,

$$b_{i+1} = Q_i\bar{p}_{i-1} + Q_{i-1}\bar{p}_{i-2}(Q_i+\bar{p}_{i-1}) + Q_{i-2}\bar{p}_{i-3}(Q_i+\bar{p}_{i-1})(Q_{i-1}+\bar{p}_{i-2}) + (Q_i+\bar{p}_{i-1})(Q_{i-1}+\bar{p}_{i-2}) \cdots (Q_1+\bar{p}_0)Q_0$$

with $b_1 = Q_0$, $c_1 = p_0$ for $b_0 = c_0 = 0$ (14)

Hence, with the help of logic gates, eqn. (10) and (14) can be implemented to generate the carry and borrow inputs c_{i+1} and b_{i+1} to the (i+1)th column at a faster rate than would be expected out of an ordinary parallel adder using eqn. (9).

IV. Generalized Binary Counter

For a (c,d) counter[1] with c inputs and d outputs, the number of output bits must be sufficient to represent all possible sums of c bits, hence,

$2^d - 1 \geq c$, for a n operand addition with (c-n) carries.

$$\text{The output } z_i = \sum_{j=1}^{d} c_{i+j}\, 2^{i+j} + s_i \quad \text{for } \lambda_i = 1$$

$$= -\sum_{j=1}^{d-1} b_{i+j}\, 2^{i+j} - s_i \quad \text{for } \lambda_i = -1 \qquad (15)$$

Obviously, for one addition, there have to be (d-1) carry outputs and (d-1) borrow outputs besides s_i for taking into consideration both λ_i=+1 or -1. Since carry and borrow outputs cannot occur simultaneously, they can be viewed as a single output for each column. Figure 2a shows a (7.3) counter giving rise to two carry outputs or two borrow outputs corresponding to λ_i=+1 or -1 respectively.

A (5,5;4) Generalized Binary Counter

The idea of a (c;d) counter can as well be extended to counter of type $(c_{k-1}, c_{k-2}, \ldots, c_o; d)$, where k is the number of successive input columns, c_i is the number of input bits in ith column corresponding to a weight 2^i and d is the required number of output bits. The value of the output is

$$V = \sum_{i=0}^{k-1} \lambda_i \sum_{j=0}^{c_{i-1}} x_{ij}\, 2^i \qquad (16)$$

and again subject to the condition that

$$2^d - 1 \geq \sum_{i=0}^{k-1} c_i\, 2^i \qquad (17)$$

A (5,5;4) counter is shown in Fig. 2b adding 5 inputs each at two successive columns and giving 2 sum outputs and 2 carry or borrow outputs for the next two bit positions.

In case of a (5,5;4) counter, the maximum value occurs when both λ_is are +1 and the minimum value occurs when both λ_is are -1. Fig. 2c shows both the cases producing different carry and borrow outputs. It should be kept in mind that the carry output has to be added where as the borrow output has to be subtracted in the corresponding columns irrespective of the value of λ_is.

For a (5,5;4) counter of this type to be used in reduction of partial products, there have to be provision to take into account the extra outputs available from the previous counter. As carry and borrow outputs cannot occur simultaneously, this poses no problem to the usefulness of a counter. It is worth mentioning that either the carry or the borrow output of the counter will have the same sign as the corresponding input bits and hence, it can be treated as input bits. Thus, the number of inputs to the ROM is reduced to 5+5+2 for the two unaccounted carries/borrows. In otherwords, a 4K by 6 bits ROM can be used for a (5,5;4) counter with 12 inputs and 4+2=6 outputs. Again, to take into account four different combinations of λ_is, there have to be ROMs with four different types of contents to reflect the four possible combinations of the two successive λ_is. This still seems to be quite expensive. However, keeping in view the complexity of the problem involved, this is quite expected. Fig. 3a shows the connection between two ROMs in the reduction of partial product matrix of very high number of operands. For the cases where c_i<5 the ROMs have only 5+5=10 inputs and only one level of counters is necessary as shown in Fig. 3b. The last two columns can be added by using the carry look ahead adder developed in the preceding section.

V. Negabinary Counters

The algorithm developed in[4] for multioperand addition in a negative base is reproduced in Table 1 for easy understanding.

Clearly, for negabinary system and for cases k>β=2, the addition will generate more carries c_{i+3}, c_{i+4}, etc. as the number of operand increases. We restrict our observations to k≦2, giving only twin carries. This results in a (c;d) counter where c=5 and d=3. +5 represented by 101 and +3 represented by 111.

A (5,5;4) negabinary counter

As suggested by Stenzel et al.[2] for binary numbers, equal column counters can also be employed for reducing the negabinary partial products. Maximum efficiency is achieved by introducing (5,5;4) negabinary counters which reduces a matrix of 5 rows high to 2 rows high. The final two rows can be summed through negabinary carry look ahead (c.l.a.) adders[8] or modifed binary c.l.a. adders[9]. As an example of a (5,5;4) counter, fig. 4(a) shows the addition of 5 negabinary numbers of weight 1,-2 to produce a 4 bit output. The num-

bers involved are of maximum negative value representable in a 2 bit negabinary system. This also holds good for any two adjacent columns. Fig. 4(b) shows the reduction of an arbitrary 5 bit negabinary addition employing (5,5;4) counters. A generalized $(c_{k-1}, c_{k-2}, \cdots, c_0;d)$ type counter can also be worked out in a negabinary system, where k is the number of input columns, c_i is the number of input bits in the column of weight 2^i and d is the number of bits in the output word. Table II verifies that (5,5;4), (4,5,5;5), (4,4,5,5;6), ..., etc. negabinary counters are possible as in the binary case.

A simple (5,5;4) negabinary counter can be implemented by 1K by 4 bit ROM with 10 address lines representing two columns of 5 bit inputs, as in the case of binary counters suggested by Stenzel et al., but the output of the ROM will be completely different.

VI. Conclusion

The results presented in this paper provide a unified methodology for performing multiple addition and for implementing parallel counter when the numbers are represented in a generalized binary system. One special case of negabinary system is observed to require the same amount of hardware as needed by binary parallel counters and system looks to be attractive for special purpose applications.

References

1. L. Dadda, "Some schemes for parallel multipliers," Alta. Freq., Vol. 19, pp. 349-356, May 1965.
2. W.J. Stenzel, W.J. Kubitz and G.H. Garcia, "A compact high speed parallel multiplication scheme," IEEE Trans. on Computers, Vol. C-26, pp. 948-957, Oct. 1977.
3. P.V. Sankar, S. Chakravarti and E.V. Krishnamurty, "Arithmetic Algorithms in a Negative Base," IEEE Trans. on Computers, Vol. C-22, pp. 120-125, Feb. 1973.
4. D.P. Agrawal, "Arithmetic Algorithms in a Negative Base," IEEE Trans. on Computers, Vol. C-24, pp. 998-1000, Oct. 1975.
5. C.K. Yuen, "A new representation for decimal numbers," IEEE Trans. on Computers, Vol. C-26, pp. 1286-1288, Dec. 1977.
6. S. Zohar, "New hardware realizations of recursive digital filters," IEEE Trans. on Computers, Vol. C-22, pp. 328-338, April 1973.
7. I. Koren and Y. Maliniak, "A unified approach to a class of number systems," IEEE 4th Computer Arithmetic Symposium, Santa Monica, pp. 25-28, Oct. 1978.
8. D.P. Agrawal, "Negabinary Carry Look-Ahead Adder and Fast Multiplier," Electronic Letters, Vol. 10, pp. 312-313, July 25, 1974.
9. D.P. Agrawal, "On Negabinary-Binary Arithmetic Relationships and their Hardware Reciprocity," IEEE Trans. of Computers, Vol. C-29, No. 11, pp. 1032-1035, Nov. 1980.

TABLE I

	$z_i=\Sigma x_i$	s_i	c_{i+1}	c_{i+2}
Case I	$z_i < \beta$	z_i	0	0
Case II	$k\beta \leq z_i < (k+1)\beta$ for $k=1,\ldots\beta$	$(z_i) \bmod \beta$	$\beta-k$	1
Case II	$(t\beta+k)\beta \leq z_i < (t\beta+k+1)\beta$ for $t=1,\ldots\beta-2$, $k=1,\ldots\beta$	$(z_i) \bmod \beta$	$\beta-k$	$t+1$

where, β=radix, x_i=sum bit, c_{i+2} and c_{i+1} are twin carriers.

Table II

# of outputs d	Possible Absolute Max Value	Possible Absolute Min Value	Limit of Operands	Counters Type
4	$(-2)^{i+3}+(-2)^{i+1}$	$(-2)^{i+2}+(-2)^{i}$	$5(-2)^{i+1}$;$5(-2)^{i}$	(5,5;4)
5	$(-2)^{i+4}+(-2)^{i+2}+(-2)^{i}$	$(-2)^{i+3}+(-2)^{i+1}$	$4(-2)^{i+2}+5(-2)^{i}$; $5(-2)^{i+1}$	(4,5,5;5)
6	$(-2)^{i+5}+(-2)^{i+3}+(-2)^{i+1}$	$(-2)^{i+4}+(-2)^{i+2}+(-2)^{i}$	$4(-2)^{i+3}+5(-2)^{i+1}$; $4(-2)^{i+2}+5(-2)^{i}$	(4,4,5,5;6)

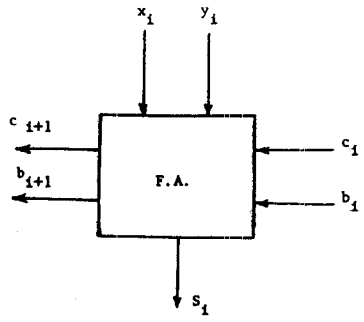

Fig. 1a. A generalized binary Full Adder

b_ic_i \ x_iy_i	$\gamma_i=0$ 00	01	11	10	$\gamma_i=1$ 10	11	01	00
00	✓x		x		✓x	✓	✓x	
01	✓				✓	✓	✓	
11	✓x		x		✓x	✓	✓x	
10	✓x	x	x	x	✓x	✓x	✓x	x

✓ - carries

x - borrows

Fig. 1c. Complete K-map for carry and borrow outputs.

b_ic_i \ x_iy_i	$\gamma_i=0$ 00	01	11	10	$\gamma_i=1$ 10	11	01	00
00			x			✓		
01	✓				✓	✓	✓	
11								
10		x	x	x				x

✓ - carries

x - borrows

Fig. 1b. Carry and borrow outputs from ith column

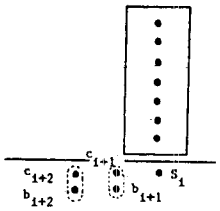

Fig. 2a. A (7,3) Generalized binary counter.

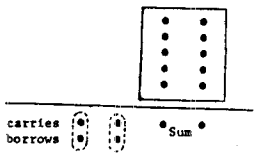

Fig. 2b. A (5,5,4) Generalized binary counter.

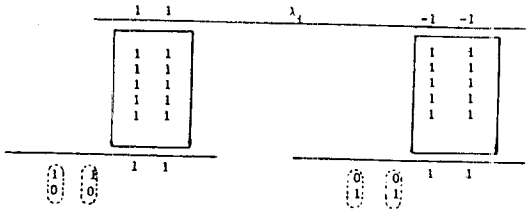

Fig. 2c. Maximum and minimum situations in a (5,5,4) ccunter.

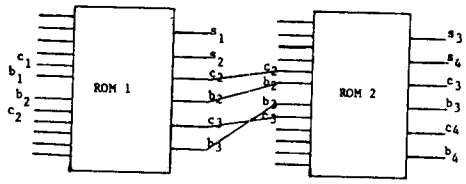

Fig. 3a. Interconnection between ROMs.

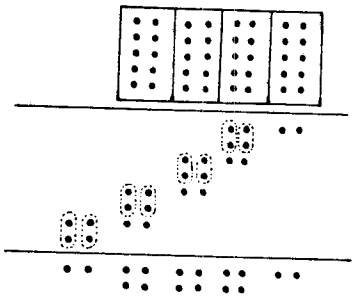

Fig. 3b. Effect of adjoint counters.

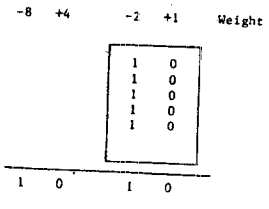

Fig. 4a. A (5,5,4) negabinary counter.

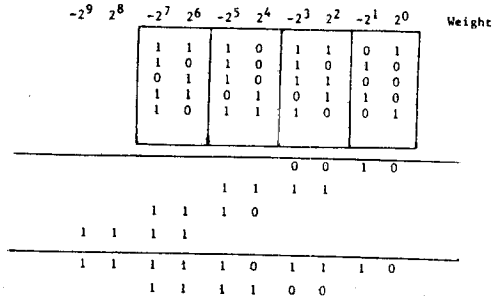

Fig. 4b. Effect of adjoint negabinary counters.