Design Of High Speed MOS Multiplier And Divider Using Redundant Binary Representation

Shigeo KUNINOBU,* Tamotsu NISHIYAMA,*
Hisakazu EDAMATSU,* Takashi TANIGUCHI,*
and Naofumi TAKAGI **

* Semiconductor Research Center, Matsushita Electric Industrial Co.,Ltd.
Moriguchi, Osaka, 570 Japan
** Department of Information Science, Faculty of Engineering
Kyoto University, Kyoto, 606 Japan

Abstract

A high speed multiplier and divider for MOS LSI based on a new algorithm is presented. When we implement the multiplier and the divider in LSI, the features such as high speed operation, small number of transistors and easy layout are the most important factors. A computational algorithm using a redundant binary representation has several excellent features such as high speed addition operations. We improved the algorithm and the method of implementation, and designed an advanced multiplier and divider with the above mentioned features. We expect that our multiplier and divider are excellent compared with multipliers using the Booth algorithm and the Wallace tree, and with divider using the SRT method, respectively.

I. Introduction

Rapid advancements in LSI technology demand innovative computational algorithms and hardware structures which fully exploit the useful features of this technology to achieve high performance. A computational algorithm using the redundant binary representation which has a digit set {-1, 0, 1} and its application to multipliers and dividers have been proposed by N. Takagi et al.[1]-[3], and these multiplier and divider have several excellent features. In order to translate this concept into innovative hardware structures for LSI, we improved the algorithm and the method for implementation and designed an advanced multiplier and divider.

A. Multiplier

A multiplier using the Booth algorithm and the Wallace tree is useful for high speed multiplication especially in bit multiplication larger than about 16-bit, so some new multipliers[4],[5] have been fabricated using this method. An n-bit x n-bit multiplication time is proportional to log n, so the multiplication time is small for large n's.

However, the layout is difficult because the Wallace tree requires a carry save adder circuit with three inputs and two outputs, which makes it difficult to realize this multiplier compactly on a LSI chip. Furthermore, this difficulty has a bad effect on high speed multiplication.

An algorithm based on the redundant binary representation also has a multiplication time proportional to log n. However, the layout of the multiplier with this algorithm is easy because a redundant binary adder with two inputs and one output can be used. In the past, the redundant binary representation has required a larger number of transistors than the binary representation, so the number of transistors was larger than that of a multiplier using the Booth algorithm and the Wallace tree since in the redundant binary representation which has a digit set {-1, 0, 1} needed two binary bits to express one redundant binary digit. We present the method to reduce the number of transistors of the multiplier.

As the result, a MOS n-bit multiplier with high speed operation, easy layout and a reduced number of transistors can be performed.

B. Divider

There was little chance to implement a hardware divider by a combinational logic. However, the rapid advancements in LSI technology makes it possible to implement the hardware divider.

The divider using the SRT division method[6] is useful for high speed operation. The division method produces a number of quotient digit in the set {-n,···,-2,-1,0,1,2,···,n} per step (in the set {-1, 0, 1} in the case of a radix two SRT non-restoring division per step).

In the algorithm with the redundant binary representation, all internal calculations, such as the quotient determination step and the partial remainder determination step, can be executed in the redundant binary representation. By adopting the full redundant binary representation to the divider, an n-bit + n-bit division time is proportional to n, and the number of transistors is proportional to n^2. Furthermore, the layout is easy because of the repeatability of its basic cells. We present the method to reduce the number of the transistors of the divider, and to realize high speed operation.

As the result, a MOS divider with high speed, easy layout and reduced number of transistors can be performed.

II. Redundant Binary Representation

A. Carry-Propagation-Free Addition

The redundant binary representation used in this paper has a digit set {-1, 0, 1}. In the redundant binary number system, carry propagation can be eliminated, so addition of two numbers can be performed in a constant time independent of the word length. Carry-propagation-free addition is performed in the following steps.

In the first step, we determine the intermediate carry c_i ($\in$ {-1, 0, 1}) and the intermediate sum digit s_i ($\in$ {-1, 0, 1}) at i-th order position, satisfying the equation $x_i + y_i = 2c_i + s_i$, where x_i and y_i are the augend and addend digit, respectively. In the second step, we obtain the sum digit z_i ($\in$ {-1, 0, 1}) at i-th order position by adding the intermediate sum digit s_i and the intermediate carry c_{i-1} from the next-lower-order position (the (i-1)-th order position) without generating a carry, namely the equation $z_i = s_i + c_{i-1}$.

In the first step, we determine c_i and s_i so that both s_i and c_{i-1} are not 1's, nor are they -1's. Table I shows a computation rule in the first step. We can know the possibility of a carry from the next-lower-order position by examining the augend and the addend digits x_{i-1} and y_{i-1} at the next-lower-order position. When both x_{i-1} and y_{i-1} are 1's or one of them is 1 and the other is 0, there is a possibility of a "1"-carry (positive carry). When both of them are -1's or one of them is -1 and the other is 0, there is a possibility of a " -1"-carry (negative carry). In the other cases, there is no possibility of a carry. Therefore, c_i and s_i can be dete mined by examining x_i, y_i, x_{i-1} and y_{i-1}.

When we determined c_i and s_i as stated in the above, no carry is generated in the addition of s_i and c_{i-1} in the second step. Thus, each sum digit z_i can be computed from x_i, y_i, x_{i-1}, y_{i-1}, x_{i-2} and y_{i-2}. Namely, z_i depends on only these six digits. This is one of the keys to the high speed operation. Fig.2.1 shows an example of carry propagation-free addition in accordance with this rule.

Table I

Computational Rule For The First Step In Carry-Propagation-Free Addition

Augend digit (xi)	Addend digit (yi)	Digits at the next-lower-order position (xi-1, yi-1)	Intermediate carry (ci)	Intermeiate sum digit (si)
1	1	———	1	0
1	0	Both are nonnegative	1	-1
0	1	Otherwise	0	1
0	0	———	0	0
1	-1	———	0	0
-1	1		0	0
0	-1	Both are nonnegative	0	-1
-1	0	Otherwise	-1	1
-1	-1	———	-1	0

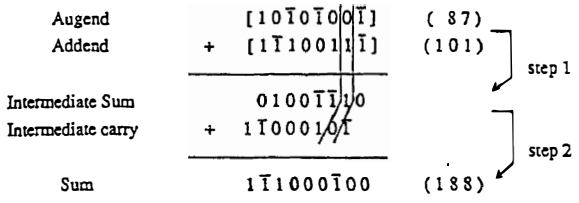

Fig.2.1 Example of Carry-Propagation-Free Addition
(1̄ denotes -1)

B. Conversion From Redundant Binary To Binary Numbers

Conversion from a redundant binary number to a binary number has to be done, because the binary number system is the standard representation system used external to the multiplier. This conversion can be performed easily from the following equation :

$$A\left(=\sum_{i=0}^{n-1} a_i \cdot 2^i\right) = A^+\left(=\sum_{ai=1} a_i \cdot 2^i\right) - A^-\left(=\sum_{ai=-1} (-a_i) \cdot 2^i\right)$$

where $a_i \in \{-1, 0, 1\}$,
each digit of A^+ and $A^- \in \{0, 1\}$.

This conversion is carried out by a carry look ahead adder with a time propotional to log n.

III. Multiply and Divide Algorithm using a Redundant Binary Adder

A. Multiplier

The 2-bit Booth algorithm has been used for high speed operation. In the algorithm, the multiplier is recoded into the radix 4 modified SD (Signed Digit) representation with a digit set {-2, -1, 0, 1, 2 }. Then, the n/2 partial products can be generated in accordance with the n/2 recoded multiplier digit.

We present a new method for partial product generation of a multiplier using the redundant binary representation, which is based on a 2-bit Booth algorithm.

In Booth algorithm, the n/2 partial products can be generated according to the recoded multiplier, on the other hand, in our method, the n/4 partial products can be generated according to the adjacent recoded multiplier group.

This method is as follows:

In 2's complement binary number, the multiplier Y is

$$Y = -y_n \cdot 2^{n-1} + \sum_{j=1}^{n-1} y_j \cdot 2^{j-1} \qquad (3\text{-}1)$$

where y_n = sign bit
$y_{n-1}, y_{n-2}, \cdots, y_1$ = number bit.

Let the number of bits be a multiple of four for convenience' sake. The multiplier Y is changed to

$$Y = \sum_{j=0}^{n/2-1} (y_{2j} + y_{2j+1} - 2y_{2j+2}) \cdot 2^{2j}$$

$$= \sum_{k=0}^{n/4-1} \{(y_{4k} + y_{4k+1} - 2y_{4k+2}) - (2y_{4k+4} - y_{4k+3} - y_{4k+2}) \cdot 2^2\} \cdot 2^{4k} \qquad (3\text{-}2)$$

where $y_0 = 0$,
the first part of terms shows even (j = 2k) and second shows odd (j = 2k + 1) , and if n is not a multiple of four, the second part of the last term is asumed to be 0.

The equation (3-2) is

$$Y = \sum_{k=0}^{n/4-1} \{S_k \cdot B_{k,even} - S_k \cdot B_{k,odd} \cdot 2^2\} \cdot 2^{4k} \qquad (3\text{-}3)$$

where $S_k = \text{sign}\,(y_{4k} + y_{4k+1} - 2y_{4k+2})$

$B_{k,even} = |\, y_{4k} + y_{4k+1} - 2y_{4k+2} \,|$

$B_{k,odd} = S_k \cdot (2y_{4k+4} - y_{4k+3} - y_{4k+2})$.

Let X be the multiplicand, the multiplication X·Y is

$$X \cdot Y = \sum_{k=0}^{n/4-1} \{S_k \cdot (B_{k,even} \cdot X) - S_k \cdot (B_{k,odd} \cdot X) \cdot 2^2\} \cdot 2^{4k} \qquad (3\text{-}4)$$

In this equation, $\{S_k \cdot (B_{k,even} \cdot X) - S_k \cdot (B_{k,odd} \cdot X) \cdot 2^2\} \cdot 2^{4k}$ shows the partial products where the number is about n/4, and each digit of this term in the set {-1, 0, 1 } can be obtained from the subtract of each bit of two terms in the set { 0, 1 }. Namely, the equation (3-4) shows that the number of partial products can be reduced to be half compared with the usual Booth algorithm, and the circuit can be simplified as described in the chapter IV.

In the partial product generation, $B_{k,even} \cdot X$ takes the value 0, +X or +2X and $B_{k,odd} \cdot X$ does 0, +X, -X, +2X or -2X. 2X can be generated by a 1-bit shift, and negative term of the equation like -X or -2X can be realized by adding "1" of 2's complement of the multiplicand X. A 2-bit shift by 2^2 of the term $S_k \cdot (B_{k,odd} \cdot X \cdot 2^2)$ can be implemented by the interconnection at the layout-stage. In order to avoid the carry propagation, a process of adding "1" of 2's complement can be executed by adding "$-S_k \cdot 2^{4k+2}$" at the step of the addition of the partial products.

Furthermore, in the partial products, the first term (the even term) can be the subtrahend or the minuend and the second term (the odd term) can be the minuend or the subtrahend in accordance with $S_k = 1$ or $S_k = -1$, respectively, and the partial products can be the redundant binary number resulting from the subtraction of the above mentioned two terms of the binary number.

Therefore, the multiplication X·Y can be executed by the n/4 partial product addition of the redundant binary representation.

B. Divider

The division algorithm presented in this paper has the following equation of the general shift-subtract/add division method.

$$R^{(j+1)} = r \cdot R^{(j)} - q_j \cdot D \qquad \text{for } j = 0, 1, \cdots\cdots, n-1 \qquad (3\text{-}5)$$

where r = radix (r = 2 in this paper)

D = divisor
q_j = j-th quotient digit from a decimal point
$r \cdot R^{(0)}$ = dividend
$R^{(j+1)}$ = remainder after the decision of q_j.

In this paper, each remainder $R^{(j)}$ is expressed in the redundant binary number {-1, 0, 1}. And the calculation of the equation can also be executed in the redundant binary number. $R^{(j)}$ has 1 digit integral number part and n digit from a decimal point, each quotient digits can be determined by examining the three most significant digit $[r_0^j.r_1^j r_2^j]$ in the set {-1, 0, 1}.

Based on the equation (3-5), the steps of the calculation are as follows :

step 1 : Dividend X ($1/2 \le X < 1$) and divisor Y ($1/2 \le Y < 1$) have the binary number of the set { 0, 1 }. And first order remainder $R^{(1)}$ is calculated in the redundant binary system from the following equation. Namely, each digit of $R^{(1)}$ has the result of the redundant binary number from the subtraction between each bit of X and each bit of Y of the binary numbers, respectively.

$$R^{(1)} \leftarrow X - Y$$
$$q_0 := 1$$

step 2 : Each q_j is determined by examining the three most significant digit $[r_0^j.r_1^j r_2^j]$ as follows :

$$q_j := \begin{cases} -1 & \text{if } [r_0^j.r_1^j r_2^j] < 0 \\ 0 & \text{if } [r_0^j.r_1^j r_2^j] = 0 \\ 1 & \text{if } [r_0^j.r_1^j r_2^j] > 0, \end{cases} \qquad (3\text{-}6)$$

and each $R^{(j+1)}$ is obtained iteratively in the redundant binary system from the following equation.

$$R^{(j+1)} := 2 \cdot R^{(j)} - q_j \cdot Y \qquad (3\text{-}7)$$

step 3 : Finally, we can get the result Z from the redundant binary / binary number conversion as shown in the following equation.

$$Q := [\, q_0 . q_1 \cdots\cdots q_n \,]$$
$$Z \leftarrow Q$$

IV. Implementation

The implementation of CMOS multiplier and divider with high speed operation, a small number of transistors and easy layout is described.

A. Redundant Binary Adder Cell

(1) Simplification of the circuit

According to the computation rule of Table I, the intermediate sum digit s_i ($\in$ {-1, 0, 1}) and the intermediate carry c_i ($\in$ {-1, 0, 1}) have the following relations.

(a) When both the augend digit x_{i-1} and the addend digit y_{i-1} are nonnegative,

(i) the intermediate sum digit s_i is nonpositive (namely, $s_i \in$ {-1, 0}) and

(ii) the intermediate carry c_{i-1} is nonnegative (namely, $c_{i-1} \in$ { 0, 1 }).

(b) When one of x_{i-1} or y_{i-1} is at least negative,

(i) the intermediate sum digit s_i is nonnegative (namely, $s_i \in$ { 0, 1 }) and

(ii) the intermediate carry c_{i-1} is nonpositive (namely, $c_{i-1} \in$ {-1, 0}).

Therefore, in the case of both (a) and (b), s_i and c_{i-1} can be expressed in two values (namely {0, 1} or {-1, 0}) from the redundant binary set {-1, 0, 1}. Namely, by using the two-value s_i and c_{i-1}, it is possible to simplify the redundant binary adder cell.

We introduce the valuable P_{i-1} ($\in$ {0, 1}) as follows :

$P_{i-1} = 0$ if (a) is satisfied, and $P_{i-1} = 1$ if (b) is satisfied.

The intermediate sum digit s_i and the intermediate carry c_{i-1} can be changed to the two-value variable u_i and v_{i-1} by the following equation.

$$u_i = P_{i-1} - s_i$$
$$v_{i-1} = P_{i-1} + c_{i-1} \qquad (4\text{-}1)$$

where s_i and $c_{i-1} \in$ {-1, 0, 1}

u_i, v_{i-1} and $P_{i-1} \in$ { 0, 1 }.

The use of the variables u_i and v_{i-1} in stead of the variables s_i and c_{i-1} makes it possible to realize the simple circuit configuration.

(2) The addition of two general redundant binary numbers - adder cell (I)

We represent a redundant binary digit t_i by two binary bits t_i^s, t_i^a, and assign "11", "00" or "01" to $t_i^s t_i^a$ according as t_i is "-1", "0" or "1", respectively, where t_i is x_i, y_i, s_i or z_i. P_i, u_i and v_i can be decided by the following logic equation.

$$P_i = x_i^s + y_i^s$$
$$u_i = s_i^a \oplus P_{i-1} \qquad (4\text{-}2)$$
$$v_i = \overline{(s_i^a . P_{i-1})} \cdot \overline{(x_i^s . y_i^s)} \cdot (x_i^a + y_i^a)$$

where s_i^a shows the absolute value of the intermediate sum digit s_i, and

$$s_i^a = x_i^a \oplus y_i^a. \qquad (4\text{-}3)$$

Finally, the i-th order of addition z_i can be decided by the following logic equation.

$$z_i^s = u_i \cdot \overline{v_{i-1}}$$
$$z_i^a = u_i \oplus v_{i-1} \qquad (4\text{-}4)$$

From the equations (4-2), (4-3) and (4-4), a CMOS redundant binary adder cell can be implemented as shown in Fig.4.1. The number of transistors of the circuit is 42 and much reduced compared with the former cell.[7] Furthermore, the critical path of the circuit is only 5 gates when the logic path of the exclusive-nor gate is assumed to be 1.5 gates.

(3) The addition of a redundant binary number and a binary number - adder cell (II)

When the augend digit or the addend digit is nonnegative (namely, binary number) in the above mentioned redundant binary adder cell (I), the adder circuit can be further simplified. In this case, the addition of two "-1's " never occurs and the intermediate carry ci-1 can be "0" when the augend digit xi and the addend digit yi are "0" and "-1", so the intermediate c_i is always nonnegative. Therefore, from the computation rule of Table I, we can ignore the case " $P_{i-1} = 1$ ", and the u_i and v_{i-1} are as follows from the equation (4-1),

$$u_i = -s_i$$
$$v_{i-1} = c_{i-1}$$

and we can get the following equation

$$u_i = s_i^a$$
$$v_i = \overline{x_i^s} \cdot (x_i^a + y_i^a)$$

The adder cell is as shown in Fig.4.2. The number of transistors of the cell is 22, so it can reduced to half compared with the cell shown in Fig.4.1. And critical logic path is only 3 gates.

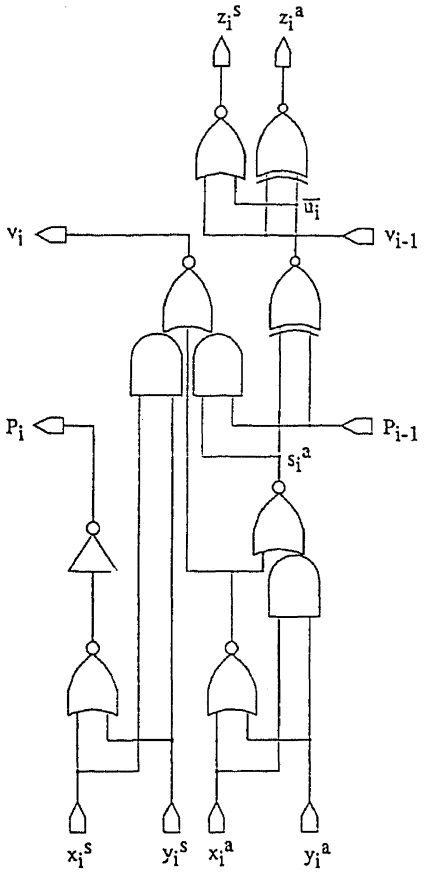

Fig.4.1 Redundant Binary Adder Cell (I)
- Addition of Two General Redundant Binary Numbers -

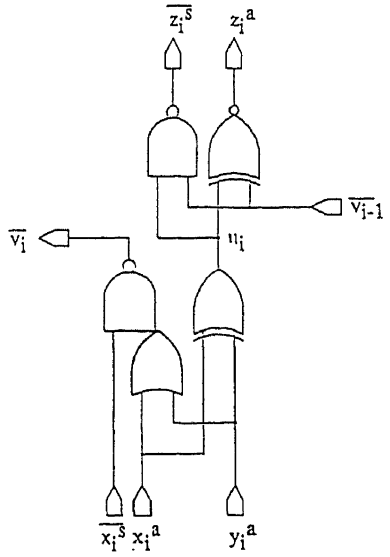

Fig.4.2 Redundant Binary Adder Cell (II)
- Addition of Redundant Binary And Binary Number -

(4) The subtraction of two binary numbers - adder cell (III)

When one of the augend digits and the addends digit is nonnegative and another is nonpositive, the other simple adder cell can be implemented. Namely, this cell is the redundant binary subtract circuit with two inputs of a subtrahend and a minuend, and the equation (4-4) is applicable to this circuit. By changing v_{i-1} and u_i to x_i and y_i in the equation (4-4), the following logic equation can be obtained.

$$z_i^s = y_i \cdot \overline{x_i}$$

$$z_i^a = y_i \oplus x_i$$

The redundant binary number resulting from the subtraction of two binary numbers can be implemented by the simple carry-propagation-free cell as shown in Fig.4.3. The cell has only 10 transistors and the critical logic path is only 1.5 gates. This cell can be used as the sub-circuit of the multiplier and the divider mentioned later.

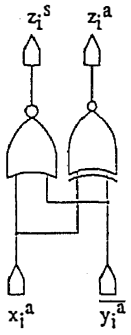

Fig.4.3 Redundant Binary Adder Cell (III)
- Subtract of Two Binary Numbers -

B. Multiplier

(1) Block diagram

The block diagram of the 2's complement multiplier is shown in Fig.4.4. There are four main blocks for recoder, partial product generator, redundant binary adder tree and converter from redundant binary to binary number. Each blocks have the following features :

(a) the recoder is based on Booth algorithm.

(b) the partial product generator can be reduced to n/4 generator as mentioned in the chapter III.

(c) the levels of adder tree can be reduced to $(\log_2 n/4)$, namely $\{(\log_2 n) - 2\}$.

(d) the converter can be designed by Unger's[8] method which is a kind of carry look ahead adder.

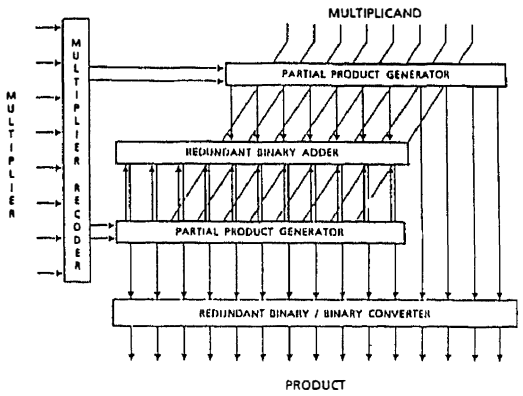

Fig.4.4 Block Diagram of multiplier (the case of 8-bit word length)

(2) Circuit design

We present the CMOS multiplier circuits.

(a) the recoder : We recode the each multiplier bits y_{2j+2}, y_{2j+1} and y_{2j} (j = 2k or j = 2k+1 from the equation (3-2)) to the three bits $r_j^{(s)}$, $r_j^{(1)}$ and $r_j^{(2)}$ as shown in Table II.

(b) the partial product generator : In the equation (3-3), S_k, $B_{k,even}$ and $B_{k,odd}$ are expressed by the set { $r_{2k}^{(s)}$, $r_{2k}^{(1)}$, $r_{2k}^{(2)}$, $r_{2k+1}^{(s)}$, $r_{2k+1}^{(1)}$, $r_{2k+1}^{(2)}$ }. Namely, S_k is implemented by use of $r_{2k}^{(s)}$, $B_{k,even}$ is by use of $r_{2k}^{(1)}$ and $r_{2k}^{(2)}$, and $B_{k,odd}$ is by use of $r_{2k}^{(s)} \oplus \overline{r_{2k+1}^{(s)}}$, $r_{2k+1}^{(1)}$ and $r_{2k+1}^{(2)}$. In the equation (3-4), the partial product generator can be obtained from the subtraction of two terms, so the adder cell (III) of Fig.4.3 is applicable to the subtract circuit. The cell for partial product generator is shown in Fig.4.5.

(c) the adder tree : the redundant binary adder cell (I) shown in Fig.4.1 constitutes the adder tree.

C. Divider

(1) Block diagram

The block diagram of the divider for the combinational circuit is shown in Fig.4.6. There are three main blocks for quotient digit determinator, partial remainder generator and converter from redundant binary to binary number. Each block has the following features :

Table II

Recode between y_{2j+2}, y_{2j+1} and y_{2j}, and $r_j^{(s)}$, $r_j^{(1)}$ and $r_j^{(2)}$

	y_{2j+2}	y_{2j+1}	y_{2j}	$r_j^{(s)}$	$r_j^{(1)}$	$r_j^{(2)}$
2	0	1	1	0	0	1
1	0	1	0	0	1	0
1	0	0	1	0	1	0
0	0	0	0	0	0	0
-0	1	1	1	1	0	0
-1	1	1	0	1	1	0
-1	1	0	1	1	1	0
-2	1	0	0	1	0	1

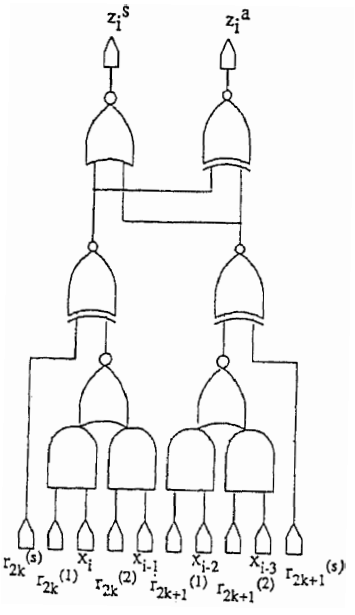

Fig.4.5 Partial Product Generator Cell

(a) quotient digits q_j (j = 1, ····· , n) are determined from the equation (3-6).

(b) partial remainders $R^{(j+1)}$ (j = 1, ····· , n) are determined iteratively from the equation (3-7).

(c) the converter is basically the same with the converter for the multiplier.

(2) Circuit design

We present the CMOS divider circuits.

(a) We represent a redundant binary digit q by two binary bits q^+, q^- and assign "01", "00" or "10" according as q is "-1", "0" or "1", and assign "11", "10" or "01" regarding two binary bits r^s, r^a according as r is "-1", "0" or "1". From these assignment and the equation (3-6), we get quotient digit determining cell as shown in Fig.4.7.

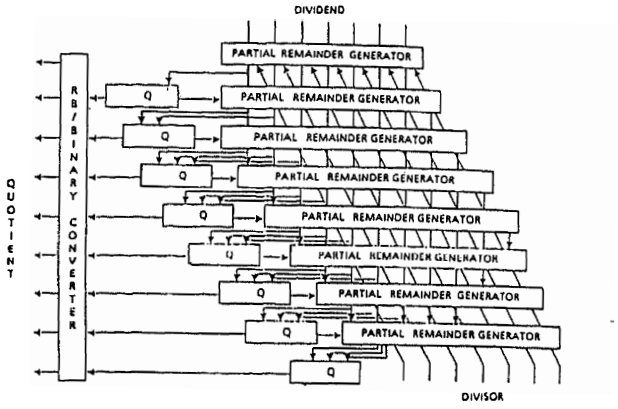

Fig.4.6 Block Diagram of Divider (the case of 8-bit word length and Q shows quotient-digit determinator)

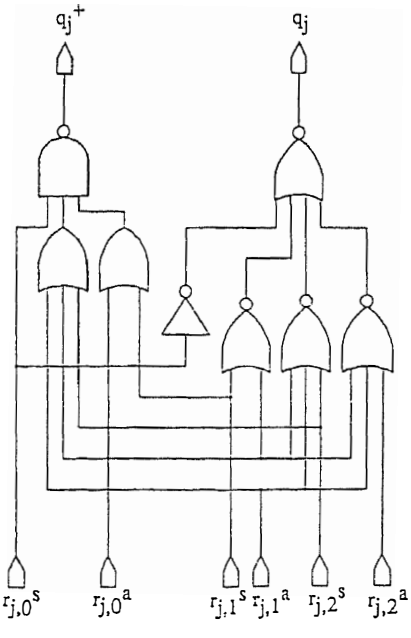

Fig.4.7 Quotient Digit Determining Cell

(b) Redundant binary add/subtract cell for partial remainder determination is shown in Fig.4.8. The redundant binary adder cell (II) shown in Fig.4.2 is applicable to this cell in the same manner. Namely, by adding the following relation to one input with binary numbers in Fig.4.2, we get Fig.4.8.

Relation : one input with binary number is

$$\begin{cases} \text{divisor } y_i & \text{if } q_j = -1 \ (\text{or } q_j^- = 1) \\ \text{divisor } \overline{y_i} & \text{if } q_j = 1 \ (\text{or } q_j^+ = 1) \\ 0 & \text{if otherwise.} \end{cases}$$

The assignment and the redundant binary add/subtract cells are the main factors to realize the MOS divider with high speed operation and a small number of transistors.

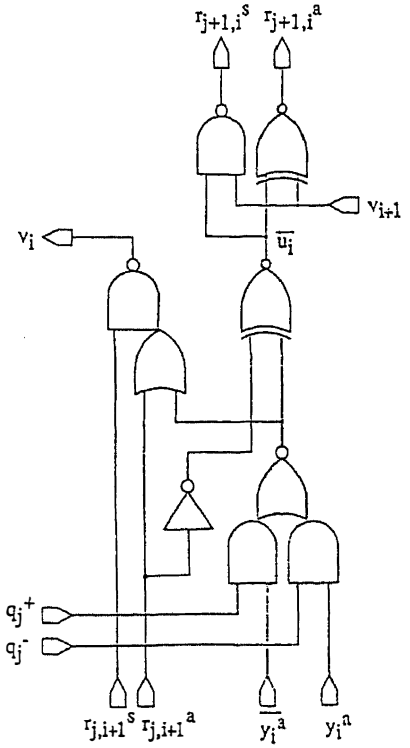

Fig.4.8 Redundant Binary Add/Subtract Cell

V. Comparison with Other Methods

In order to realize the MOS multiplier and the MOS divider, the most important features are operation speed, number of transistors and ease of layout. Therefore, we compare the multiplier and the divider using redundant binary representation with other multiplier and divider from the point of these features.

A. Multiplier

So far multipliers using the Booth algorithm and the Wallace tree have been one of the most advanced one in the areas mentioned above. Table III shows a comparison of our multiplier and a multiplier using the Booth algorithm and the Wallace tree for 64-bit word length for the critical logic path and the number of CMOS transistors. As shown in Table III, by adopting this multiplier, we can reduce the critical logic path to 90% and reduce the number of transistors to 75% compared with multipliers using the Booth algorithm and the Wallace tree. Furthermore, the layout is easier than these conventional multipliers.

The features of our multiplier in Table III are mainly based on the following reasons:

(a) Carry propagation-free addition with two inputs and one outputs (results from redundant binary representation).

(b) Reduction of the number of partial products (results from this paper).

(c) Simplification of the cells (result from this paper).

Table III

Comparison With Two Type Multipliers (64bitX64bit)

	Logic path (gates)	Number of transistors	Layout
Our multiplier	34	90k	easy
Multiplier using Booth & Wallace	38	120k	Difficult

B. Divider

The divider using the SRT division method is advanced especially in the feature of high speed operation for combinational logic. Table IV shows a comparison of our divider and a divider using the SRT division method for 64-bit word length for the critical logic path and the number of the CMOS transistors. There are very few examples of the implementation of the SRT division method, therefore this is just one example. As shown in Table IV, by adopting this divider, we can reduce the critical logic path to about 42% and reduce the number of transistors to about 90% compared with a divider using the SRT division method. Furthermore, the layout is easy because of its cell repeatability.

The features of our divider in Table IV are mainly based on the following reasons:

(a) Nonrestoring division and all internal calculation with signed digit (result from redundant binary representation).

(b) Simplification of the cells (results from this paper).

Table IV

Comparison With Two Type Dividers (64bit÷64bit)

	Logic path (gates)	Number of transistors	Layout
Our Divider	396	110k	easy
Divider using SRT	938	120k	easy

VI. Summary

We presented the design for a MOS high speed multiplier and divider using redundant binary representation. This multiplier and divider have the features of higher speed operation, smaller number of transistors and easier layout compared with the conventional advanced multipliers and dividers. From these features, we expect this multiplier and divider to represent significant advances in the field of algorithmic processing implemented in MOS LSI.

Acknowledgement

The authors wish to thank Prof. S. Yajima of Kyoto University for instructive discussions. The authors also wish to thank Dr. H. Mizuno, a managing director of Matsushita Electric and Dr. S. Horiuchi, a director of Advanced Devices Lab. of Matsushita for giving the opportunity to perfom this research. Finally, the authors wish to thank the staff of the LSI design group for supporting this work.

References

[1] N. Takagi et al., " A VLSI-Oriented High-Speed Multiplier U· 'ng A Redundant Binary Addition Tree, " Trans. of IECE Japan, vol. J66-D, no. 6, pp. 683-690, June 1982 (in japanese)

[2] N. Takagi et al., " A VLSI-Oriented High-Speed Divider Using Redundant Binary Representation, " Trans. of IECE Japan, vol. J67-D, no. 4, pp. 450-457, April 1984 (in Japanese)

[3] N. Takagi et al., " High-Speed VLSI Multiplication Algorithm With a Redundant Binary Addition Tree, " IEEE Trans. on Computers, vol. C-34, no. 9, pp. 789-796, September 1985

[4] Y. Kaji et al., " A 45ns 16x16 CMOS Multiplier," 1984 IEEE International Solid-State Circuits Conference, pp. 84-85, 1984

[5] W. M. McAllister et al., " An NMOS 64b Floating-Point Chip Set," 1986 IEEE International Solid-State Circuits Conference, pp. 34-35, 1986

[6] G. S. Taylor,"Radix 16 SRT Divider With Overlapped Quotient Selection Stages, " IEEE Proceedings 7th Symposium on Computer Arithmetic, pp. 64-71, 1985

[7] Y. Harata et al., " High-Speed Multiplier Using A Redundant Binary Adder Tree, " 1984 IEEE International Conference On Computer Design, pp. 165-170, 1984

[8] S. Unger., " Tree Realization Of Iterative Circuit, " IEEE Trans. on Computers, vol. C-26, no.6, pp. 365-383, April 1977