Correspondence

A Combinatoric Division Algorithm for Fixed-Integer Divisors

DAVID H. JACOBSOHN

Abstract—A procedure is presented for performing a combinatoric fixed-integer division that satisfies the division algorithm in regard to both quotient and remainder. In this procedure, division is performed by multiplying the dividend by the reciprocal of the divisor. The reciprocal is, in all nontrivial cases, of necessity a repeating binary fraction, and two treatments for finding the product of an integer and repeating binary fraction are developed. Two examples of the application of the procedure are given.

Index Terms—Combinatoric division, division algorithm, fixed-integer division, hardware division, modular division.

Introduction

In certain operations such as that of base conversion and that of locating elements in a multidimensional list, the availability of a combinatoric algorithm to perform a division by a fixed-integer divisor that will supply both an exact quotient and an exact remainder is of value. This correspondence will discuss the general techniques for developing such a combinatoric division algorithm and the evaluation of such an algorithm for several divisors.

Two specific examples to be considered are the conversion of base 2 integers into binary coded decimal (BCD) and the mapping of the bits of a memory organized into 48-bit words into a contiguous bit-addressable array.

General Description and Approach

Division by binary integers that are integral powers of 2 is trivial. It is accomplished by partitioning the dividend. Given a divisor[1]

$$B = b_n b_{n-1} \cdots b_i \cdots b_1 b_0$$

where

$$b_k = 1 \text{ and } b_i = 0, \qquad \text{for } i \neq k$$

and dividend

$$A = a_m a_{m-1} \cdots a_1 a_0$$

the quotient is

$$Q = q_{m-k} q_{m-k-1} \cdots a_0$$

where

$$q_{m-k-i} = a_{m-i}$$

and the remainder is

$$R = a_{k-1} a_{k-2} \cdots a_1 a_0.$$

In the more general case where B is not an integral power of 2, B can be partitioned into an odd number and an integral power of 2 as follows:

$$B = b_n b_{n-1} \cdots b_i \cdots b_1 b_0.$$

Divide B by $B'' = 2^k$ where k is the largest value such that 2^k divides B and leaves a zero remainder. Then

$$B = (B/B'')B''.$$

Let

$$B' = B/B''$$

then

$$B' = b'_{n-k} b'_{n-k-1} \cdots b'_0$$

where

$$b'_{n-k-i} = b_{n-i}$$

and

$$B = B' \cdot B'' = b'_{n-k} b'_{n-k-1} \cdots b'_0 \cdot 2^k.$$

Having shown that the divisor B can be partitioned into a B' and B'' trivially, it now remains to be shown that the division of A by B can also be partitioned trivially. The following should suffice as proof.

Given A and $B = B' \cdot B''$ where $B'' = 2^k$ as above then

$$A/B = A/(B'B'')$$

let

$$A/B'' = Q'' + R''/B''$$

and

$$Q''/B' = Q + R/B'$$

then

$$\frac{A}{B''B'} = Q + R/B' + \frac{R''}{B'B''}$$

$$\frac{A}{B} = Q + \frac{B''R}{B'B''} + \frac{R''}{B'B''} = Q + \frac{B''R + R''}{B}.$$

In this expression for A/B, only Q and R are generated nontrivially.

Having shown that Q and R above are the only values that must be generated nontrivially, it becomes obvious that the division process to be implemented by the combinatoric algorithm need only be concerned with odd integers.

Division by Fixed Odd Integers

Now that the total process has been reduced to division by fixed odd integers, it is appropriate to outline the basic approach for implementation.

In order to generate $A/B = Q + R/B$, the following procedure is taken.

1) The reciprocal I of B' is generated.

2) A combinatoric multiplier is constructed based upon the fixed constant I and having A as a variable input parameter. The output parameters of this multiplier are Q and a number related to R.

The significant point to observe is that the value of I above must be represented as a repeating binary of value less than 1. The cycle length of this repeating binary must be less than the value of B.[2] The problems that remain are to construct a fixed finite multiplier to represent a repeating binary, to extract the integer portion Q of the result, and to convert the fractional portion of the result to the appropriate value for R. More specifically, the problem is to generate the product $A \cdot I$ where A is a variable, and extract Q and R from that product. Since I is a repeating binary fraction of cycle n, it can be represented as

Manuscript received July 25, 1972; revised November 2, 1972. This work was performed under the auspices of the U.S. Atomic Energy Commission.

The author is with the Applied Mathematics Division, Argonne National Laboratory, Argonne, Ill. 60439.

[1]Subscripted literals assume positional significance throughout the text.

[2]When performing an integer division by B, all of the partial remainders must be integers of values less than B. Upon the repetition of any partial remainder, the cycle will repeat, and should the partial remainder zero occur, the process will terminate; therefore, there can at most be $B - 1$ possible partial remainders limiting the cycle to a maximum of $B - 1$ in length.

$$I = I' \sum_{j=0}^{\infty} 2^{-jn}.$$

Therefore, the product

$$A \cdot I = A \cdot I' \sum_{j=0}^{\infty} 2^{-jn}.$$

The product $P = A \cdot I'$ can be generated easily by combinatoric methods, such as Wallace's,[3] reducing the problem to generating

$$P \cdot \sum_{j=0}^{\infty} 2^{-jn}.$$

(Subsequently Σ will be used to represent $\Sigma_{j=0}^{\infty} 2^{-jn}$.) P is a product of finite length and can be represented as

$$P = P_m P_{m-1} \cdots P_1 P_0$$

where each P_i element is of n bits in length (i.e., a digit base 2^n).

The following illustrates the generation of $P \cdot \Sigma$:

$$\begin{array}{l} P_m P_{m-1} \cdots P_1 P_0 \\ \underline{\times\ (2^0 + 2^{-n} + 2^{-2n} + \cdots)} \\ (P_m \quad P_{m-1} \cdots P_1 \quad P_0) \\ \quad + (P_m \quad P_{m-1} \cdots P_1 \quad P_0) \\ \qquad + (P_m \quad P_{m-1} \cdots P_1 \quad P_0) \\ \qquad\qquad \ddots \\ \hline P \cdot \Sigma = \cdots . \end{array}$$

Reorganizing the above expression, we get

$$\begin{array}{l} P_m P_{m-1} \cdots P_1 P_0 \\ \underline{\times\ (2^0 + 2^{-n} + 2^{-2n} + \cdots)} \\ P_m \quad P_m \quad P_m \cdots \\ \quad + P_{m-1}\, P_{m-1} \cdots \\ \qquad \ddots \\ \qquad\qquad + P_0\, P_0 \cdots \\ \hline P \cdot \Sigma = \cdots . \end{array}$$

In the preceding product, we can replace P_m by P'_m, and each prior P_i by P'_i, where

$$P'_i = P_i + P'_{i+1} + \delta$$

with $\delta = 1$ when $P_i + P'_{i+1} \geqslant 2^n - 1$ and $\delta = 0$ otherwise. (δ is the carry when adding two digits in base 2^n.) Then

$$P \cdot \Sigma = P'_m\, 2^{-n(m-1)} + P'_{m-1}\, 2^{-n(m-2)} + \cdots + P'_1 + P'_0 \sum_{i=1}^{\infty} 2^{-in}.$$

Since P'_i can have values greater than $2^n - 1$, the above may be resolved to

$$P \cdot \Sigma = P''_m P''_{m-1} \cdots P''_1 \cdot (P''_0 \Sigma)$$

(where again the P''_i are integers base 2^n). $P''_m P''_{m-1} \cdots P''_1$, the portion to the left of the binary point is the exact integer quotient Q of A/B.

The portion to the right of the binary point is a repeating fraction of period n representing the quantity R/B. Multiplying R/B by B with suitable rounding gives the exact integer value of R, hence $R = B\, \Sigma_{i=1}^{\infty} 2^{-in}$.

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

Alternate $P \cdot \Sigma$ Generator

An alternate and more economical approach for the generation of $P \cdot \Sigma$ exists and is similar in concept to the process of casting out nines.[4] In this approach the single adder array shown as follows is used.

$$\begin{array}{lcccccccc} & P_m & P_{m-1} & P_{m-2} & \cdots & P_i & P_{i-1} & \cdots & P_1 \; P_0 \; 0 \\ & +0 & P''_m & P''_{m-1} & \cdots & P''_{i+1} & P''_i & \cdots & P''_2 \; P''_1 \; P''_0 \\ & + & & & & & & & 1 \\ \hline P \cdot \Sigma = & P''_m & P''_{m-1} & P''_{m-2} & \cdots & P''_i & P''_{i-1} & \cdots & P''_1 \cdot P''_0 \; X \end{array} .$$

This adder can be considered to be blocked into $m + 2$ units, each of n bits. Into the leftmost block P_m will be inputted. Each successive block to the right will have the next P_i inputted with i going from m to 0. Therefore, the rightmost block will have no P_i input. If we now name the outputs of the various blocks of the adder P''_i where each P''_i output corresponds positionally to a P_i input, we will get $P''_m \cdots P''_0\, X$ as the output. The generation of the $P \cdot \Sigma$ product is obtained by inputting the P''_ith terms into the remaining inputs of the $i - 1$st block of the adder. A carry is inserted into the right end of the adder to accomplish rounding, and the previously unspecified inputs at blocks m and -1 of the adder receive zero's. $P''_m \cdots P''_i \cdots P''_1 \cdot P''_0$ is the product $P \cdot \Sigma$. Of this product $P''_m \cdots P''_1$ is the integral part of the quotient, and P''_0 the fractional remainder. This alternate method reduces the hardware requirements for the generation of $P \cdot \Sigma$ to a third of that of the method contained in the correspondence.

Example I: Base 2 to Base 10 Conversion

In this process, a binary integer is to be divided by 10 base 2. The remainder will be the BCD rightmost decimal digit of the conversion. The quotient residues are repeatedly divided by 10 until the final quotient residue is less than 10. At each step, an additional digit to the left of the previous digit is generated.

For this illustration of method, we shall assume a 9-bit binary number is to be converted. Given

$$A = a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$$
$$B = 1010. \text{ base } 2$$

then

$$\begin{aligned} B' &= 101. \\ B'' &= 10. \\ Q'' &= a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1 \\ R'' &= a_0 \\ I &= 0.00110011 \cdots = \frac{1}{B'} \\ I' &= 0.0011 \\ n &= 4. \end{aligned}$$

P then equals $Q'' \cdot I'$ which is generated as follows:

$$\begin{array}{r} a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1 \\ \underline{+\ a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1} \\ P = a'_9\, a'_8\, a'_7\, a'_6\, a'_5\, a'_4\, a'_3\, a'_2\, a'_1\, a'_0 \end{array}$$

and

$$P = P_2 P_1 \cdot P_0$$

where

$$\begin{aligned} P_0 &= a'_3\, a'_2\, a'_1\, a'_0 \\ P_1 &= a'_7\, a'_6\, a'_5\, a'_4 \\ P_2 &= 0 \quad 0 \quad a'_9\, a'_8. \end{aligned}$$

[4]D. E. Knuth, *Seminumerical Algorithms. Vol. 2: The Art of Computer Programming.* Reading, Mass.: Addison-Wesley, 1969, p. 253.

$P \cdot \Sigma$ is then generated by the basic approach as follows:

$$P_2' = P_2$$

$$P_1' = P_2' + P_1 + \delta = c_7\, a_7''\, a_6''\, a_5''\, a_4''$$

which is generated as follows:

$$\begin{array}{rcccccccc}
 & 0 & 0 & a_9' & a_8' & 0 & 0 & a_9' & a_8' \\
+ & a_7' & a_6' & a_5' & a_4' & a_7' & a_6' & a_5' & a_4' \\
 & & & & & & & & +1 \\
\hline
P_1' = c_7 & a_7'' & a_6'' & a_5'' & a_4'' & & & &
\end{array}$$

and

$$P_0' = P_1' + P_0 + \delta = c_3\, a_3''\, a_2''\, a_1''\, a_0''$$

which is generated as follows:

$$\begin{array}{rcccccccc}
 & a_7'' & a_6'' & a_5'' & a_4'' & a_7'' & a_6'' & a_5'' & a_4'' \\
+ & a_3' & a_2' & a_1' & a_0' & a_3' & a_2' & a_1' & a_0' \\
 & & & & & & & & +1 \\
\hline
P_0' = c_3 & a_3'' & a_2'' & a_1'' & a_0'' & & & &
\end{array} .$$

Then

$$\begin{array}{rccccccccccc}
 & & & & & & c_3 & & a_3'' & a_2'' & a_1'' & a_0'' \\
 & & & + & c_7 & a_7'' & a_6'' & a_5'' & a_4'' & & & \\
+ & 0 & 0 & 0 & 0 & a_9' & a_8' & & & & & \\
\hline
P \cdot \Sigma = & & a_{10}''' & a_9''' & a_8''' & a_7''' & a_6''' & a_5''' & a_4''' \cdot a_3'' & a_2'' & a_1'' & a_0''
\end{array} .$$

Alternately, $P \cdot \Sigma$ can be generated as follows:

$$\begin{array}{cccccccccccccc}
 & & & & & & & & & & & & & +1 \\
a_9' & a_8' & a_7' & a_6' & a_5' & a_4' & a_3' & a_2' & a_1' & a_0' & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & a_{10}''' & a_9''' & a_8''' \cdot & a_7''' & a_6''' & a_5''' & a_4''' & a_3'' & a_2'' & a_1'' & a_0'' \\
\hline
a_{10}''' & a_9''' & a_8''' & a_7''' & a_6''' & a_5''' & a_4''' \cdot & a_3'' & a_2'' & a_1'' & a_0'' & X & X & X & X
\end{array} .$$

Now reassembling the parts

$$Q = a_{10}'''\, a_9'''\, a_8'''\, a_7'''\, a_6'''\, a_5'''\, a_4'''$$

and

$$R = B' \cdot (a_3''\, a_2''\, a_1''\, a_0'') \sum_{i=1}^{\infty} 16^{-i}.$$

Hence

$$A/B = Q + \frac{2R + a_0}{B}.$$

Example II: Bit Location in a Word-Organized Memory

In this example it is desired to locate a bit in a 64-word memory of 48-bit words. The bits are to be addressed contiguously independent of word boundaries. The memory has 64×48 bits in total. The location of a bit is achieved by dividing the bit address by 48, accepting the quotient as the word address, and the remainder as the bit address within that word. Given

$$A = a_{11}\, a_{10}\, a_9 \cdots a_1\, a_0$$

$$B = 110000. \text{ base 2}$$

then

$$B' = 11.$$

$$B'' = 10000.$$

$$Q'' = a_{11}\, a_{10} \cdots a_5\, a_4$$

$$R'' = a_3\, a_2\, a_1\, a_0$$

$$I = 0.0101 \cdots = \frac{1}{B'}$$

$$I' = 0.01$$

$$n = 2.$$

P then equals $Q'' \cdot I'$, which is

$$P = a_{11}\, a_{10}\, a_9\, a_8\, a_7\, a_6 \cdot a_5\, a_4$$

and

$$P = \qquad P_3\, P_2\, P_1 \cdot P_0$$

where

$$P_0 = a_5\, a_4$$

$$P_1 = a_7\, a_6$$

$$P_2 = a_9\, a_8$$

$$P_3 = a_{11}\, a_{10}.$$

$P \cdot \Sigma$ is then generated by the basic approach as follows:

$$P_3' = P_3$$

$$P_2' = P_3' + P_2 + \delta = c_9\, a_9'\, a_8'$$

which is generated as follows:

$$\begin{array}{rcccc}
 & a_{11} & a_{10} & a_{11} & a_{10} \\
+ & a_9 & a_8 & a_9 & a_8 \\
 & & & & +1 \\
\hline
\end{array}$$

$$c_9 \quad a_9'\, a_8'$$

and similarly

$$P_1' = P_2' + P_1 + \delta = c_7\, a_7'\, a_6'$$

and

$$P_0' = P_1' + P_0 + \delta = c_5\, a_5'\, a_4'.$$

$$\begin{array}{rcccccccc}
 & & & & & c_5 & & a_5' & a_4' \\
 & & & + & c_7 & a_7' & a_6' & & \\
 & + & c_9 & a_9' & a_8' & & & & \\
+ & a_{11} & a_{10} & & & & & & \\
\hline
P \cdot \Sigma = & a_7'' & a_6'' & a_5'' & a_4'' & a_3'' & a_2'' \cdot & a_1'' & a_0''
\end{array}$$

Alternately, $P \cdot \Sigma$ can be generated as follows:

$$\begin{array}{rcccccccccc}
 & a_{11} & a_{10} & a_9 & a_8 & a_7 & a_6 & a_5 & a_4 & 0 & 1 \\
+ & 0 & 0 & a_7'' & a_6'' & a_5'' & a_4'' & a_3'' & a_2'' & a_1'' & a_0'' \\
\hline
 & a_7'' & a_6'' & a_5'' & a_4'' & a_3'' & a_2'' \cdot & a_1'' & a_0'' & X & X
\end{array} .$$

Now reassembling the parts

$$Q = a_7''\, a_6''\, a_5''\, a_4''\, a_3''\, a_2''$$

and

$$R = B'(a_1''\, a_2'') \sum_{i=1}^{\infty} 4^{-i}.$$

Hence

$$A/B = Q + \frac{16R + a_3\, a_2\, a_1\, a_0}{B}.$$

Conclusion

A method for developing combinatoric structures for performing divisions by fixed integers has been demonstrated. In the examples shown, the divisions were performed through the use of full adders only, thus illustrating that this technique can be implemented using standard off-the-shelf MSI components.