# Correspondence

## On the Complexity of Table Lookup for Iterative Division

BEHROOZ PARHAMI

*Abstract*—**We show that except for a few special cases allowing smaller tables, the lookup table used for achieving $k$ digits of convergence after the initial multiplication (or for obtaining the approximate reciprocal of the divisor with $k - 1$ digits of accuracy) in iterative division methods must have at least $(r - 1)r^k$ words of $k + 1$ digits, $r$ being the number representation base. In the important special case of $r = 2$ with $k \geq 5$, a $2^k$-word table with $k$-bit entries can be used, since the initial digit is always 1. It is also shown that a table of this optimal size can always be constructed. The special cases corresponding to $r = 3$ with $k = 1$, and $r = 2$ with $k \leq 4$, allow smaller tables than the general case and are thus handled separately.**

*Index Terms*—**Arithmetic algorithms, computer arithmetic, division algorithm, iterative division, quadratic convergence, reciprocal approximation, table lookup.**

### I. Introduction

Iterative division methods, used in high-speed computers, yield the quotient of two $b$-digit fractions (values less than 1) in $O(\log b)$ multiplication steps [1]–[9]. This can be much faster than the $O(b)$ addition steps of algorithms which obtain the quotient one digit at a time, provided that an extremely fast multiplier (preferably pipelined to handle two multiplications concurrently) is available.

Iterative division algorithms have been known for at least 35 years [9]. The table lookup enhancement method for such algorithms has been in practical use for about two decades [1]. While some sort of validation and optimization must have accompanied each of the many actual implementations, no general discussion of the required table size appears in the literature. Waser and Flynn [8, pp. 195–196] analyze the ROM table size for $r = 2$. But even in this special case, no prior proof of optimality is known to the author.

Given two base-$r$ fractions $d$ and $z$, with $z$ normalized (i.e., $z \geq 1/r$) and $d < z$, the quotient $d/z$ can be computed by repeated multiplications as

$$d/z \approx da_0a_1a_2 \cdots a_f, \tag{1}$$

provided that the $a_i$'s are chosen such that

$$za_0a_1a_2 \cdots a_f \approx 1.0.$$

In practice, $a_0$ is selected as a function of $z$, using a lookup table, to yield the initial product $za_0$ satisfying

$$1 - r^{-k} \leq za_0 < 1 + r^{-k}. \tag{2}$$

Then, in the $j$th step, $a_j$ is computed from

$$a_j = 2 - za_0a_1 \cdots a_{j-1} \tag{3}$$

or some approximation thereof, based on truncated values.

With the exact value for $a_j$, the proof that (1) holds proceeds as follows. First, we rewrite (2) as

$$|1 - za_0| \leq r^{-k}.$$

Then, using (3), we find

$$1 - za_0a_1 = 1 - [1 - (1 - za_0)][1 + (1 - za_0)] = (1 - za_0)^2.$$

Thus,

$$|1 - za_0a_1| \leq r^{-2k}.$$

We obtain, by induction on $j$

$$|1 - za_0a_1 \cdots a_j| \leq r^{-2^jk}.$$

Given $b$-digit fractions, the number of required multiplications is $2f + 1$ (the last multiplication, yielding $za_0a_1a_2 \cdots a_f$, need not be performed), where

$$f = \lceil \log_2 (b/k) \rceil.$$

As an example, for $b = 56$ and $k = 7$, seven multiplications are needed to complete our iterative division algorithm.

There are many practical variations on this basic algorithm, to speed up the required operations for particular systems. For example, working with a truncated value for $za_0a_1 \cdots a_{j-1}$, one obtains an approximate value for $a_j$ from (3). This can speed up the subsequent multiplication due to a reduction in the number of multiplier digits but reduces the rate of convergence (e.g., [1]). With proper selection of parameters, the net result may be a faster algorithm, given a certain level of hardware complexity.

In this paper, we focus on the initial table lookup procedure to obtain $a_0$ as a function of $z$. Thus, the details of the above-mentioned variations are not relevant to our discussion.

### II. The Lookup Table

The selection of $a_0$ is based on several high-order digits of $z$. The following theorem defines the lookup table needed for this purpose.

*Theorem 1:* For any normalized base-$r$ fraction

$$z = 0.x_1x_2x_3 \cdots x_{k+1}x_{k+2} \cdots x_b,$$

there exists a number less than $r$, represented as

$$a_0 = y_0.y_1y_2y_3 \cdots y_k$$

and obtainable as a function of $x_1, x_2, \cdots, x_{k+1}$, such that (2) holds. In other words, for (2) to hold, a lookup table containing $r^{k+1}$ words of $k + 1$ base-$r$ digits is sufficient. The lookup table size can actually be $(r - 1)r^k$, since $x_1 \neq 0$.

*Proof:* Let $x_1x_2x_3 \cdots x_{k+1}$ denote the integer $m$ in base $r$. The fraction $z$ being normalized, we have

$$r^k \leq m < r^{k+1} \tag{4}$$

and

$$mr^{-(k+1)} \leq z < (m + 1)r^{-(k+1)}. \tag{5}$$

Let $y_0y_1y_2 \cdots y_k$ denote the integer $n$ in base $r$, obtainable as a function of $m$:

$$n = g(m) = a_0r^k. \tag{6}$$

Manuscript received September 6, 1985.

The author is with the School of Computer Science, Carleton University, Ottawa, Ont., Canada K1S 5B6; on leave from Sharif University of Technology, Tehran 14584, Iran.

IEEE Log Number 8713998.

0018-9340/87/1000-1233$01.00 

We obtain from (5) and (6)

$$mnr^{-(2k+1)} \le za_0 < (m+1)nr^{-(2k+1)}.$$

Thus, for (2) to hold, we must have

$$1 - r^{-k} \le mnr^{-(2k+1)}$$

$$(m+1)nr^{-(2k+1)} \le 1 + r^{-k}.$$

In other words, $n$ must be an integer solution of

$$r^{k+1}(r^k - 1)/m \le n \le r^{k+1}(r^k + 1)/(m+1). \quad (7)$$

For (7) to yield some integer solution, we must have

$$r^{k+1}(r^k - 1)/m \le \lfloor r^{k+1}(r^k + 1)/(m+1) \rfloor. \quad (8)$$

It is now sufficient to prove that (8) holds for all the values of $m$ satisfying (4).

Let $q$ be the quotient and $s \le m$ be the remainder of dividing $r^{k+1}(r^k + 1)$ by $m + 1$. Thus,

$$r^{k+1}(r^k + 1) = (m+1)q + s. \quad (9)$$

Obtaining $m$ from (9) and substituting in the left-hand side of (8), and noting that the right-hand side of (8) is $q$, we obtain the following condition:

$$q + s \le 2r^{k+1}. \quad (10)$$

But, inequality (10) always holds, since

$$m + 1 \ge r^k + 1$$

$$q = \lfloor r^{k+1}(r^k + 1)/(m+1) \rfloor \le r^{k+1}$$

$$s \le m < r^{k+1}.$$

This concludes the proof. □

*Example 1:* For $r = 10$ and $k = 1$, taking from (7) the solution

$$n = g(m) = \lceil r^{k+1}(r^k - 1)/m \rceil = \lceil 900/m \rceil,$$

we obtain a lookup table with 90 eight-bit words (Table I). □

## III. Proof of Optimality

We now show that the table lookup procedure implied by Theorem 1 is optimal for $r > 3$, in the sense that neither the number of lookup digits of $z$ (i.e., $x_1, x_2, \cdots, x_{k+1}$) nor the number of digits of $a_0 = y_0.y_1y_2 \cdots y_k$ can be reduced.

*Theorem 2:* The lookup table implied by Theorem 1 is optimal for $r > 3$; i.e., no smaller table can yield the values of $a_0$ satisfying (2) for all the values of $z$.

*Proof:* We first show that the lookup table cannot have fewer, i.e., $(r - 1)r^{k-1}$, words. Let $x_1x_2x_3 \cdots x_k$ denote the integer $m'$ in base $r$. Then, following the steps in the proof of Theorem 1, with $m'$ substituted for $m$, inequality (7) becomes

$$r^k(r^k - 1)/m' \le n \le r^k(r^k + 1)/(m' + 1). \quad (11)$$

To complete this part of our proof, we must show that for some value of $m'$ in the admissible range $r^{k-1} \le m' < r^k$, no integer $n$ can satisfy (11). Let $m' = r^{k-1}$. Substituting this value for $m'$ in (11), we obtain

$$r(r^k - 1) \le n \le r^k(r^k + 1)/(r^{k-1} + 1)$$

which is easily transformed into

$$r(r^k - 1) \le n \le r(r^k - 1) - r(r^k - 2r^{k-1} - 1)/(r^{k-1} + 1). \quad (12)$$

TABLE I
THE LOOKUP TABLE FOR EXAMPLE 1 ($r = 10$, $k = 1$)

| $0.x_1x_2$ | $y_0.y_1$ | $0.x_1x_2$ | $y_0.y_1$ | $0.x_1x_2$ | $y_0.y_1$ |
|---|---|---|---|---|---|
| 0.10 | 9.0 | 0.40 | 2.3 | 0.70 | 1.3 |
| 0.11 | 8.2 | 0.41 | 2.2 | 0.71 | 1.3 |
| 0.12 | 7.5 | 0.42 | 2.2 | 0.72 | 1.3 |
| 0.13 | 7.0 | 0.43 | 2.1 | 0.73 | 1.3 |
| 0.14 | 6.5 | 0.44 | 2.1 | 0.74 | 1.3 |
| 0.15 | 6.0 | 0.45 | 2.0 | 0.75 | 1.2 |
| 0.16 | 5.7 | 0.46 | 2.0 | 0.76 | 1.2 |
| 0.17 | 5.3 | 0.47 | 2.0 | 0.77 | 1.2 |
| 0.18 | 5.0 | 0.48 | 1.9 | 0.78 | 1.2 |
| 0.19 | 4.8 | 0.49 | 1.9 | 0.79 | 1.2 |
| 0.20 | 4.5 | 0.50 | 1.8 | 0.80 | 1.2 |
| 0.21 | 4.3 | 0.51 | 1.8 | 0.81 | 1.2 |
| 0.22 | 4.1 | 0.52 | 1.8 | 0.82 | 1.1 |
| 0.23 | 4.0 | 0.53 | 1.7 | 0.83 | 1.1 |
| 0.24 | 3.8 | 0.54 | 1.7 | 0.84 | 1.1 |
| 0.25 | 3.6 | 0.55 | 1.7 | 0.85 | 1.1 |
| 0.26 | 3.5 | 0.56 | 1.7 | 0.86 | 1.1 |
| 0.27 | 3.4 | 0.57 | 1.6 | 0.87 | 1.1 |
| 0.28 | 3.3 | 0.58 | 1.6 | 0.88 | 1.1 |
| 0.29 | 3.2 | 0.59 | 1.6 | 0.89 | 1.1 |
| 0.30 | 3.0 | 0.60 | 1.5 | 0.90 | 1.0 |
| 0.31 | 3.0 | 0.61 | 1.5 | 0.91 | 1.0 |
| 0.32 | 2.9 | 0.62 | 1.5 | 0.92 | 1.0 |
| 0.33 | 2.8 | 0.63 | 1.5 | 0.93 | 1.0 |
| 0.34 | 2.7 | 0.64 | 1.5 | 0.94 | 1.0 |
| 0.35 | 2.6 | 0.65 | 1.4 | 0.95 | 1.0 |
| 0.36 | 2.5 | 0.66 | 1.4 | 0.96 | 1.0 |
| 0.37 | 2.5 | 0.67 | 1.4 | 0.97 | 1.0 |
| 0.38 | 2.4 | 0.68 | 1.4 | 0.98 | 1.0 |
| 0.39 | 2.4 | 0.69 | 1.4 | 0.99 | 1.0 |

Inequality (12) can yield a value for $n$ only if

$$r^k - 2r^{k-1} - 1 \le 0.$$

This is impossible for $r > 3$. The fact that (12) does not even have a noninteger solution indicates that a lookup table with fewer words does not exist for any word size.

We next show that the word size of our lookup table cannot be reduced to $k$ digits. Let $y_0y_1y_2 \cdots y_{k-1}$ denote the integer $n'$ in base $r$, obtainable as a function of $m$. Then, following the steps in the proof of Theorem 1, with $n'$ substituted for $n$, inequality (8) becomes

$$r^k(r^k - 1)/m \le \lfloor r^k(r^k + 1)/(m+1) \rfloor. \quad (13)$$

To complete this part of our proof, we must show that for some value of $m$ satisfying (4), inequality (13) does not hold. Let $m = r^{k+1} - r^k - 1$. Substituting this value for $m$ in (13), we obtain

$$r^k(r^k - 1)/(r^{k+1} - r^k - 1) \le \lfloor (r^k - 1 + 2)/(r - 1) \rfloor,$$

which, noting that $r^k - 1$ is divisible by $r - 1$ and that $2 < r - 1$ for $r > 3$, can be changed into

$$r^k(r^k - 1)/(r^{k+1} - r^k - 1) \le (r^k - 1)/(r - 1). \quad (14)$$

Simplifying (14), we arrive at $r^k \le 0$, which is impossible. This concludes our proof. □

*Example 2:* For $r = 10$ and $k = 1$, the lookup procedure cannot be based solely on $x_1$, nor can it yield single-digit values for $a_0$ as a function of $x_1$ and $x_2$. Let $a_0 = h(x_1)$. Then, if $z = 0.1x_2x_3x_4 \cdots$, we must have $h(1) \ge 9$ (in case $x_2 = x_3 = x_4 = \cdots = 0$) and $h(1) \le 5.5$ (in case $x_2 = x_3 = x_4 = \cdots = 9$). Clearly, no such $h$ exists. Next, let $a_0 = h'(x_1, x_2)$ be a single-digit integer. Then, if $z = 0.89x_3x_4 \cdots$, we must have $h'(8, 9) > 1$ and $h'(8, 9) < (1 + 10^{-1})/z < 1.24$. Clearly, no such $h'$ exists. □

## IV. The Two Special Cases

Base-2 and base-3 numbers were excepted in the proof of Theorem 2. The reason is that we encounter some irregularities for $r = 2$ and $r = 3$. Thus, we handle these special cases by Theorems 3 and 4, respectively.

*Theorem 3:* For $r = 2$, the lookup table implied by Theorem 1 (with the high-order digits removed to obtain $k$-digit table entries) is

optimal for $k \geq 5$. For $k \leq 4$, either the table size can be cut in half or its word size can be reduced by one bit (sometimes both) to obtain a smaller lookup table.

*Proof:* Let $m' = 2^{k-1} + 1$ in the proof of Theorem 2. Then, we can write (11) as

$$2^k(2^k-1)/(2^{k-1}+1) \leq n \leq 2^k(2^k+1)/(2^{k-1}+2). \quad (15)$$

Tedious, but straightforward, manipulation transforms (15) into

$$6/(2^{k-1}+1) \leq n+6-2^{k+1} \leq 12/(2^{k-1}+2). \quad (16)$$

The left-hand side of (16) is always greater than zero and the right-hand side is less than 1 for $k \geq 5$. Thus, the middle part, which is an integer, cannot lie between these two values. Table II, which shows the values of $a_0$ for $k \leq 4$, completes the first part of our proof. Here, $z_k = m'/2^k$ is the $k$-bit truncated version of $z$.

Next, let $m = 2^{k+1} - 6$ in the proof of Theorem 2. Then, we write (13) as

$$2^k(2^k-1)/(2^{k+1}-6) \leq \lfloor 2^k(2^k+1)/(2^{k+1}-5) \rfloor. \quad (17)$$

Once again, straightforward manipulation converts (17) to

$$6/(2^{k+1}-6) \leq \lfloor 3/4 + 35/(2^{k+3}-20) \rfloor. \quad (18)$$

For $k \geq 5$, the left-hand side of (18) is greater than zero, while its right-hand side is zero. Thus, inequality (18), and hence inequality (13), does not hold for $k \geq 5$. As a result, $a_0$ cannot have less than $k + 1$ bits. However, the high-order bit of $a_0$ is always equal to 1. Thus, the lookup table for $r = 2$ and $k \geq 5$ consists of $2^k$ words of length $k$.

Table III, which shows the values of a shorter $a_0$ for $k \leq 4$, completes our proof. Here, $z_{k+1} = m/2^{k+1}$ is the $(k + 1)$-digit truncated version of $z$. Note that for $k \leq 3$, the table size and the number of bits in $a_0$ can be reduced at the same time. This is true since $y_k$ is zero in Table II and $a_0'$ is independent of $x_{k+1}$ in Table III for $k \leq 3$. For $k = 4$, either the table size or the number of bits in $a_0$ can be reduced, with the former option yielding a smaller table (40 total bits compared to 64 bits). □

*Theorem 4:* For $r = 3$, the lookup table implied by Theorem 1 is optimal for $k \geq 2$. For $k = 1$, the table size can be reduced by a factor of 3 and its word size reduced by one digit.

*Proof:* Let $m' = 3^{k-1}$ in the proof of Theorem 2. Then, we can write (11) as

$$3^k(3^k-1)/3^{k-1} \leq n \leq 3^k(3^k+1)/(3^{k-1}+1). \quad (19)$$

Straightforward manipulation transforms (19) into

$$0 \leq n - 3(3^k-1) \leq 6/(3^{k-1}+1) - 3. \quad (20)$$

For $k \geq 2$, the right-hand side of (20) becomes negative, leading to a contradiction. Noting that for $k = 1$, the values of $a_0 = 2.0$ and $a_0 = 1.0$ can be selected for $x_1 = 1$ and $x_1 = 2$, respectively, completes the first part of our proof.

Next, let $m = 3^{k+1} - 4$ in the proof of Theorem 2. Then, we write (13) as

$$3^k(3^k-1)/(3^{k+1}-4) \leq \lfloor 3^k(3^k+1)/(3^{k+1}-3) \rfloor. \quad (21)$$

Subtracting $3^{k-1}$ from both sides of (21), we obtain after some manipulation

$$1/9 + 4/(3^{k+3}-36) \leq \lfloor 2/3 + 2/(3^{k+1}-3) \rfloor. \quad (22)$$

For $k \geq 2$, the left-hand side of (22) is greater than zero, while its right-hand side is zero. Thus, inequality (22), and hence (13), does not hold for $k \geq 2$. Noting that the values of $a_0$, given in the first part of the proof for $k = 1$, are actually single-digit integers completes our proof. □

TABLE II
A SET OF LOOKUP TABLES FOR $r = 2$ AND $k \leq 4$

| $k$ | $z_k$ | $a_0$ |
|---|---|---|
| 1 | 0.1 | 1.0 |
| 2 | 0.10 | 1.10 |
| 2 | 0.11 | 1.00 |
| 3 | 0.100 | 1.110 |
| 3 | 0.101 | 1.100 |
| 3 | 0.110 | 1.010 |
| 3 | 0.111 | 1.000 |
| 4 | 0.1000 | 1.1110 |
| 4 | 0.1001 | 1.1011 |
| 4 | 0.1010 | 1.1000 |
| 4 | 0.1011 | 1.0110 |
| 4 | 0.1100 | 1.0100 |
| 4 | 0.1101 | 1.0011 |
| 4 | 0.1110 | 1.0010 |
| 4 | 0.1111 | 1.0000 |

TABLE III
ANOTHER SET OF LOOKUP TABLES FOR $r = 2$ AND $k \leq 4$

| $k$ | $z_{k+1}$ | $a_0'$ |
|---|---|---|
| 1 | 0.10 | 1. |
| 1 | 0.11 | 1. |
| 2 | 0.100 | 1.1 |
| 2 | 0.101 | 1.1 |
| 2 | 0.110 | 1.0 |
| 2 | 0.111 | 1.0 |
| 3 | 0.1000 | 1.11 |
| 3 | 0.1001 | 1.11 |
| 3 | 0.1010 | 1.10 |
| 3 | 0.1011 | 1.10 |
| 3 | 0.1100 | 1.01 |
| 3 | 0.1101 | 1.01 |
| 3 | 0.1110 | 1.00 |
| 3 | 0.1111 | 1.00 |
| 4 | 0.10000 | 1.111 |
| 4 | 0.10001 | 1.111 |
| 4 | 0.10010 | 1.110 |
| 4 | 0.10011 | 1.101 |
| 4 | 0.10100 | 1.100 |
| 4 | 0.10101 | 1.100 |
| 4 | 0.10110 | 1.011 |
| 4 | 0.10111 | 1.011 |
| 4 | 0.11000 | 1.010 |
| 4 | 0.11001 | 1.010 |
| 4 | 0.11010 | 1.010 |
| 4 | 0.11011 | 1.001 |
| 4 | 0.11100 | 1.001 |
| 4 | 0.11101 | 1.001 |
| 4 | 0.11110 | 1.000 |
| 4 | 0.11111 | 1.000 |

## V. CONCLUSION

Except for a few special cases handled by Theorems 3 and 4, Theorems 1 and 2 show that a lookup table with $(r - 1)r^k$ words of $k + 1$ base-$r$ digits is necessary and sufficient for the iterative division algorithm, if $k$ digits of convergence are to be achieved with the initial multiplication. Theorem 1 also provides a method for constructing the lookup table, as illustrated by Example 1. In general, $n$ can be selected as

$$n = \lceil r^{k+1}(r^k-1)/m \rceil \quad (23)$$

from (7). Considering the definitions of $z$ and $a_0$ in the statement of Theorem 1 and denoting the $(k + 1)$-digit truncated value of $z$ by $z_{k+1}$, we can transform (23) into

$$a_0 = \lceil (r^k-1)/z_{k+1} \rceil / r^k. \quad (24)$$

Equation (24) can be used to construct lookup tables for iterative division. Lookup tables for the special cases of $r = 3$ with $k = 1$ and $r = 2$ with $k \leq 4$ have been provided in this paper as constructive proofs for parts of Theorems 3 and 4.

The results presented here are also valid for division with reciprocal approximation. To see this, we show that $a_0$ is actually an

approximation of $1/z$ with $k - 1$ digits of accuracy. Let

$$a_0 = 1/z + e,$$

where $e$ is the absolute error of the initial reciprocal approximation. Then, $za_0 = 1 + ze$, and (2) can be written as

$$|ze| \leq r^{-k}. \tag{25}$$

The fraction $z$ being normalized, we have $z \geq 1/r$. Thus, we obtain from (25)

$$|e| \leq r^{-(k-1)}.$$

That is to say, as an approximation of $1/z$, $a_0$ is accurate to $k - 1$ digits ($y_0.y_1y_2y_3 \cdots y_{k-2}$, with $y_{k-1}$ and $y_k$ possibly being in error). Thus, to obtain the reciprocal of $z$ with $k - 1$ digits of accuracy, a lookup table with $(r - 1)r^k$ words of $k + 1$ base-$r$ digits is needed in most cases.

ACKNOWLEDGMENT

The research reported in this paper was carried out while the author was with Sharif University of Technology, Tehran 14584, Iran. The author gratefully acknowledges the contribution of referee B for bringing reference [8] to his attention.

REFERENCES

[1] S. F. Anderson *et al.*, "The IBM System/360 Model 91: Floating point execution unit," *IBM J. Res. Devel.*, vol. 11, pp. 34–53, Jan. 1967.
[2] D. Ferrari, "A division method using a parallel multiplier," *IEEE Trans. Comput.*, vol. EC-16, pp. 224–226, Apr. 1967.
[3] M. J. Flynn, "On division by functional iteration," *IEEE Trans. Comput.*, vol. C-19, pp. 702–706, Aug. 1970.
[4] R. Z. Goldschmidt, "Applications of division by convergence," M.S. thesis, Massachusetts Instit. Technol., Cambridge, MA, June 1964.
[5] J. B. Gosling, *Design of Arithmetic Units for Digital Computers.* London, England: Macmillan, 1980.
[6] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design.* New York: Wiley, 1979.
[7] E. V. Krishnamurthy, "On optimal iterative schemes for high speed division," *IEEE Trans. Comput.*, vol. C-19, pp. 227–231, Mar. 1970.
[8] S. Waser and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers.* New York: Holt, Rinehart, and Winston, 1982.
[9] M. V. Wilkes *et al.*, *Preparation of Programs for an Electronic Digital Computer.* Reading, MA: Addison-Wesley, 1951.

## Finite Field Fault-Tolerant Digital Filtering Architectures

G. ROBERT REDINBO

*Abstract*—**Digital filtering architectures that simultaneously offer advantages for VLSI fabrication and contain distributed error control are presented. Such structures require parallelism as well as inherent error-control capabilities because VLSI implementations are susceptible to temporary and intermittent hardware errors. The filtering convolution operation is similar to the formation of cyclic error-correcting codes so that fault-tolerant systems employing finite field arithmetic may be designed containing such codes imbedded directly in the architecture. The interconnection of such systems produces a fault-tolerant system. In addition, the subsystems possess a common design structure which is easily customized to the particular field required, an attractive feature for yield enhancement. Straightforward realizations depending on parallel algebraic decompositions are studied, introducing the locations for fault tolerance and the role of cyclic codes.**

*Index Terms*—**Chinese Remainder Theorem, cyclic codes, digital filtering, error-correcting codes, fault-tolerant computing, finite fields, residue arithmetic, VLSI.**

Manuscript received October 16, 1984; revised September 24, 1985 and March 25, 1986. This work was supported in part by AFOSR Grant 80-153 and by NSF Grant DMC-19843.

The author is with the Department of Electrical and Computer Engineering/Computer Science, University of California, Davis, CA 95616.

IEEE Log Number 8715309.

### INTRODUCTION

The modern trend in digital electronics dictates that systems should be constructed from modular subsystems which are in turn easily replicated for fabrication in very large scale integration (VLSI) technology. Previous work has shown how digital filtering systems can be designed for wafer scale integration (WSI) by using many parallel sections, each employing small finite fields [1]. However, at the same time problems of testing and error control become more difficult while yield appears as a limiting factor. Wafer scale integrated VLSI digital filtering systems require concurrent testing and error-control subsystems distributed throughout their architectures.

A new major source of errors is called soft errors. They are internal momentary errors random in both time and space, characterized by their short duration and infrequent random nature. Soft errors may arise from several sources such as electric field distortions caused by atomic particle injection or random parameter variations due to circuit processing tolerances. Nevertheless, a soft error acts temporarily just as an intermittent fault, but its source cannot be traced to any permanent defect in the electronic structure or design. Classical testing and fault-tolerant approaches are inadequate in coping with soft errors, and future complex VLSI systems must devote a portion of the increased system architectural capability to internal distributed error control to enhance overall system performance. In addition, these same error-control features increase the yield of complete systems by providing full external system performance even though internal electronic devices are not functioning properly.

A new approach for fault tolerance involves cyclic codes which can be imbedded in a distributed fashion throughout the architectures, greatly increasing the overall reliability. These architectures take full advantage of VLSI capabilities while still allowing powerful error-control features to be built directly into all the parts of the system because the inherent algebraic structures of error-correcting codes, over general finite fields, match the fundamental operation in filtering, convolution. Special structures associated with cyclic codes called minimal ideals have very important algebraic properties not only affording implicit error protection but leading to fast maximally parallel implementation algorithms [2].

Digital filtering operations, after the proper sampling, scaling, rounding, and sequence segmentation are viewed as being performed over a ring of integers modulo $M$ where $M$ is a large positive integer [1]–[4]. Furthermore, it is possible to decompose the implementation into many parallel realizations by using residue number system techniques [5], [6]. When the modulus $M$ is a product of only distinct primes, the parallel decomposition results in subsections operating with finite field arithmetic [7] where in this case the integer $M$ has the following form.

$$M = p_1 p_2 \cdots p_s.$$

This situation is depicted in Fig. 1, where each parallel subsection operates over the distinct finite fields denoted by $GF(p_i)$, $i = 1, 2, \cdots, s$. The arithmetic decomposition and recombination operations shown, respectively, at the input and output in Fig. 1 are a result of the Chinese Remainder Theorem [8], [9].

There are known techniques [10]–[15] for protecting the arithmetic reconstruction shown at the output in Fig. 1. However, the major