Tight Upper Bounds on the Minimum Precision Required of the Divisor and the Partial Remainder in High-Radix Division

Behrooz Parhami, *Fellow*, *IEEE*

Abstract—Digit-recurrence binary dividers are sped up via two complementary methods: keeping the partial remainder in redundant form and selecting the quotient digits in a radix higher than 2. Use of a redundant partial remainder replaces the standard addition in each cycle by a carry-free addition, thus making the cycles shorter. Deriving the quotient in high radix reduces the number of cycles (by a factor of about h for radix 2^h). To make the redundant partial remainder scheme work, quotient digits must be chosen from a redundant set, such as [–2, 2] in radix 4. The redundancy provides some tolerance to imprecision so that the quotient digits can be selected based on examining truncated versions of the partial remainder and divisor. No closed form formula for the required precision in the partial remainder and divisor, as a function of the quotient digit set and the range of the partial remainder, is known. In this paper, we establish tight upper bounds on the required precision for the partial remainder and divisor. The bounds are tight in the sense that each is only one bit over a well-known simple lower bound. We also discuss the implications of these bounds for the quotient digit selection process.

Index Terms—Digit-recurrence division, digit-selector PLA, high-radix division, *p*-*d* plot, quotient digit selection, SRT division.

1 Introduction

Division is the most complex, and hence slowest, of the four basic arithmetic operations. Thus, techniques for speeding up division have been extensively studied by researchers in computer arithmetic [9]. Simple dividers are usually based on digit-recurrence algorithms employing repeated additions, while high-performance implementations tend to use multiplier-based convergence schemes. In digit-recurrence division [3], digits of the quotient are derived one at a time, from the most to the least significant. For a *k*-digit quotient, there are at least *k* cycles in the division process; one cycle for producing each radix-*r* quotient digit, plus additional cycles for initialization, possible final correction, and rounding.

There are but two ways to speed up such an algorithm: reducing the number of cycles and making each cycle shorter in duration. The first implies higher radix division, while the second leads to several techniques, including keeping the partial remainder in redundant form so as to eliminate the relatively long delay of carry-propagate addition in each cycle. Regardless of the design choices at the algorithmic level, circuit techniques can be used to improve the divider speed and often have a significant impact on the resulting performance [5]. We do not deal with circuit techniques in this paper.

To make the use of redundant partial remainder possible, quotient digits must be chosen from a redundant set, such as [–2, 2] in radix 4. The redundancy provides some tolerance to imprecision so that the quotient digits can be selected based on examining truncated versions of the divisor and the partial remainder [12], [1]. A redundant partial remainder that is truncated to a few bits can be quickly transformed into an estimate for the partial remainder or its bits used directly as inputs to a PLA or ROM table that spews out quotient digit values. Among other issues, the designer of a fast digit-recurrence divider determines the precision with which the divisor and the partial remainder must be examined in order to deduce the value of the next quotient digit. This precision, stated in the number of bits before and after the radix point, dictates the complexity of the PLA or the size of the ROM table yielding the quotient digit value. The PLA/ROM size influences not only the cost (VLSI area), but also the circuit speed, given that the quotient digit selection block is usually on the critical path.

Many practical aspects of the quotient digit selection process and associated hardware implementations have been discussed in the literature [13], [3], [7]. Particularly relevant to this work are the original analysis by Atkins [1] showing the minimal precision required and more recent contributions by Burgess and Williams [2] and Oberman and Flynn [8] elaborating on the selection criteria and trade offs. These latter contributions will be reviewed in Section 6 dealing with redundant representation of the partial remainder.

No closed form formula for the required precision in the divisor and the partial remainder, as a function of the quotient digit set and the range of the partial remainder, is known. So, the design process is often described as trial and error. In this paper, we establish tight upper bounds on the minimum precision required for the divisor and the partial remainder. The bounds are tight in the sense that each is only one bit over a well-known simple lower bound, leading to the requirement for just four trial points during the design process: the one represented by the lower bounds, one (or two) more bit(s) of precision for *d*, one more bit of precision for *p*. If none of these four points leads to a viable design, then one more bit of precision for both *d* and *p* is guaranteed to work. We also elaborate on the implications of these results on the design of practical SRT dividers that use a redundant representation for the partial remainder. The notation used in our analyses is listed in Table 1 for ready reference.

2 Quotient Digit Selection

We derive our main results in Sections 2-5 assuming a nonredundant partial remainder. We then consider the effects of a redundant partial remainder on these results in Section 6. Even though practical SRT dividers are almost always based on a redundant representation of the partial remainder, this separation and stepwise derivation lead to a simpler and clearer presentation, as well as better understanding of the impact of redundancy.

Radix-*r* digit-recurrence division for the (possibly adjusted) dividend $s^{(0)} = z = 0.z_{-1}z_{-2}2 \dots z_{-k}$ and the normalized divisor $d = 0.1d_{-2}d_{-3} \dots d_{-k}$ is based on the equation

$$s^{(j)} = rs^{(j-1)} - q_{-j}d,$$

where $s^{(j)}$ is the *j*th partial remainder and $q = 0.q_{-1}q_{-2} \dots q_{-k}$ is the developed quotient. We assume $s^{(0)} = |z| \le h|d|$, perhaps ensured through preshifting of the dividend, yielding the adjusted dividend. The parameter h will be defined shortly. Additionally, we assume that the radix is a power of 2 so that everything is done in binary, with a radix-*r* digit corresponding to $\log_2 r$ bits.

To allow the choice of q_{-j} based on truncated versions of d and $p^{(j)} = rs^{(j)}$, we use a redundant quotient digit set $[-a, a]$, with $a \ge r/2$ (usually, we have $a \le r - 1$, although overredundant digit sets satisfying $a \ge r$ are also of some interest).

Assuming $d > 0$, to maintain the invariant $|s| \le hd$, throughout the division process, we must be able to restore a shifted partial remainder, having the worst-case magnitude rhd, to hd or less by subtracting ad from it. This requirement defines h by bounding it from above:

• *The author is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560. E-mail: parhami@ece.ucsb.edu.*

Manuscript received 2 Apr. 2001; revised 16 May 2002; accepted 25 Feb. 2003.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 113915.

TABLE 1
List of Notation

Symbol	Meaning or usage
Δd	Minimum width or horizontal spacing of an overlap region
Δp	Minimum height or vertical spacing of an overlap region
δ	Precision required of the divisor, in bits after the radix point
ε	Precision required of the partial remainder, in bits after the radix point
θ	Angle between the lower boundary of an overlap region and the d axis
a	Maximum magnitude of a radix-r quotient digit
d, d^{trunc}	Divisor, and its truncated version
$d^{\min}$, $d^{\max}$	Minimum (maximum) magnitude for the divisor d
g	Power of 2 constituting the radix ($r = 2^g$) or the period in hybrid redundancy
h	A fraction, such that the partial remainder has a magnitude of at most $h\|d\|$
k	Word width in bits
l	Number of bits after the radix point that are assimilated
lb($\bullet$)	Lower bound
m	Index of vertical grid line
n	Number of vertical grid line
p, p^{trunc}	Shifted partial remainder (rs), and its truncated version
$p(\bullet)$	Partial remainder at a particular point of the p-d plot
q	Quotient
r	Radix
s, $s^{(j)}$	Partial remainder in general, and partial remainder after the jth step
s', s''	Two components of a redundant partial remainder ($s = s' + s''$)
u	Height of a leftmost rectangle of width w enclosed in an overlap region (Fig. 3)
ub($\bullet$)	Upper bound
v	$= \Delta p - u$ (Fig. 3)
W, w	Grid tile width or horizontal spacing
x	Candidate radix-r quotient digit
Y, y	Grid tile height or vertical spacing
z	Dividend

$$rhd - ad \le hd \quad \text{or} \quad h \le a/(r-1).$$

Choosing $h = a/(r-1)$ so as to impose the least possible restriction on the range of s, we can easily determine the two boundaries of the region where x is a valid quotient digit value:

$$-hd \le p - xd \le hd.$$

This leads to the following range for p, across which x is a valid quotient digit value:

$$(x-h)d \le p \le (x+h)d.$$

The ranges associated with the validity of x and $x-1$ as quotient digit values overlap in:

$$(x-h)d \le p \le (x-1+h)d.$$

Fig. 1 contains a graphical representation of this overlap region between the two straight lines $p = (x-h)d$ and $p = (x-1+h)d$. This representation is known as a *p-d* plot. Within the overlap region shown in the partial *p-d* plot of Fig. 1, either $x-1$ or x is a valid quotient digit value. Thus, the boundary for choosing between $x-1$ and x as quotient digit value can be drawn anywhere inside the overlap region.

3 Lower Bounds for the Required Precision

The lower bounds presented in this section are well known. However, given that the methods and notation used in their derivations are essential for understanding the rest of the paper, we include them here for the sake of completeness.

If we truncate d to δ bits after the radix point, the uncertainty in the value of d, based on its truncated version d^{trunc}, will be less than $w = 2^{-\delta}$ In other words, $d^{\text{trunc}} \le d < d_{\text{trunc}} + 2^{-\delta}$ Similarly, if we truncate p to ε bits after the radix point to get p^{trunc}, the uncertainty will be less than $y = 2^{-\varepsilon}$ and we have $p^{\text{trunc}} \le p < p^{\text{trunc}} + 2^{-\varepsilon}$. These two truncations together define a grid or tiling on the *p-d* plot, with vertical (horizontal) grid lines being $w = 2^{-\delta}$ ($y = 2^{-\varepsilon}$) apart.

A valid choice for the quotient digit based on d^{trunc} and p^{trunc} is possible only if none of the $w \times y$ tiles on the *p-d* plot, each one

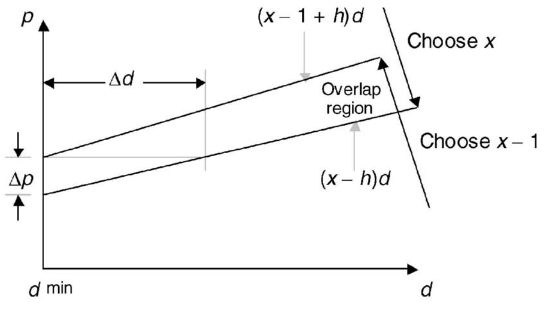

Fig. 1. Part of a *p-d* plot, showing an overlap region along with Δd and Δp.

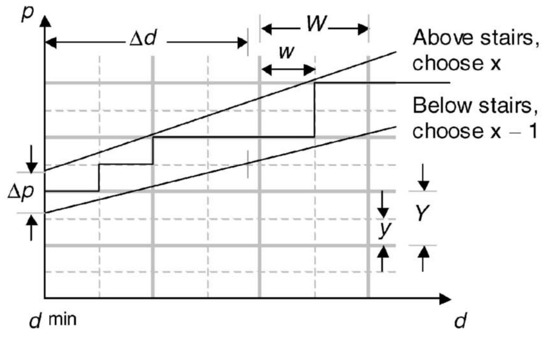

Fig. 2. Tiling the *p-d* plot and embedding stairs in an overlap region.

representing uncertainties in the values of *d* and *p*, intersects both boundaries of an overlap region. Put another way, between each pair of adjacent vertical (horizontal) grid lines, there must exist a horizontal (vertical) grid line that is totally contained in the overlap region. Defining Δd as the minimum horizontal separation between the boundaries of the overlap region, clearly we must have $w \leq \Delta d$. Based on the equations for the two lines bounding the overlap region in Fig. 1, we find

$$\Delta d = d^{\min}(2h-1)/(x-h),$$

which assumes its smallest possible value for $x = a$. Hence, we arrive at the following well-known lower bound (lb) for the precision required of *d* in quotient digit selection:

$$\delta \geq \mathrm{lb}(\delta) = \lceil -\log_2 \Delta d \rceil = \lceil -\log_2 [d^{\min}(2h-1)/(a-h)] \rceil.$$

A similar argument, with the roles of the horizontal and vertical grid lines interchanged, yields:

$$\varepsilon \geq \mathrm{lb}(\varepsilon) = \lceil -\log_2 \Delta p \rceil = \lceil -\log_2 [d^{\min}(2h-1)] \rceil.$$

Of course, in addition to the fractional bits of *d* and *p*, as dictated by the bounds above, integer bits of each operand must also be inspected. Because, even in unsigned division, the partial remainder can go negative, the sign bit of *p* is important as well. In the case of signed division, the sign bit of *d* is also involved in the decision.

Returning to the discussion of the number of fractional bits needed, we endeavor to prove that setting $\delta = \mathrm{lb}(\delta) + 1$ and $\varepsilon = \mathrm{lb}(\varepsilon) + 1$ is always sufficient for quotient digit selection, thus establishing the simultaneous upper bounds $\mathrm{ub}(\delta) = \mathrm{lb}(\delta) + 1$ and $\mathrm{ub}(\varepsilon) = \mathrm{lb}(\varepsilon) + 1$. Fig. 2 represents an example. Here, based on the precision lower bounds provided by Δd and Δp, the coarse grid of $W \times Y$ tiles (solid, heavy gray lines) is built. The resulting precisions are clearly inadequate, as evident from the absence of a solid horizontal grid line spanning the overlap region between the vertical grid lines at $d^{\min}$ and $d^{\min} + W$. Our results will show that halving the grid line spacing in each dimension (adding the broken grid lines in Fig. 2) provides adequate precision in the worst case.

4 Upper Bounds for the Minimum Precision

With the background presented in the previous sections, we are now ready to state and prove our main result [10], which helps in the vast majority of cases where precisions equal to the lower bounds previously derived are inadequate for a hardware divider.

Theorem 1. *To determine the quotient digit in radix-r division, with r a power of 2, it is sufficient to inspect δ bits of d and ε bits of the nonredundant p after their radix points, where:*

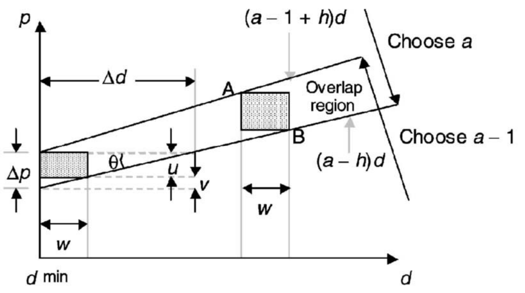

Fig. 3. Part of a *p-d* plot and notation used in the proof of Theorem 1.

$$\delta = \mathrm{lb}(\delta) + 1 = \lceil -\log_2 [d^{\min}(2h-1)/(a-h)] \rceil + 1$$
$$\varepsilon = \mathrm{lb}(\varepsilon) + 1 = \lceil -\log_2 [d^{\min}(2h-1)] \rceil + 1.$$

Proof. We need to show that if vertical grid lines are spaced $w = 2^{-\delta} \leq \Delta d/2$ apart, there will be at least one horizontal grid line (spaced $y = 2^{-\varepsilon} \leq \Delta p/2$ apart) that is entirely contained in the overlap region between two successive vertical grid lines. The proof is based on the geometric representation in Fig. 3 and consists of showing that any rectangle that touches two consecutive vertical grid lines and the boundaries of the overlap region has a height of at least $\Delta p/2$, which is no less than $2^{-\varepsilon}$. We only need to prove this at the leftmost edge of the uppermost overlap region that defines the boundary between choosing $a-1$ or a as the quotient digit value. Corresponding rectangles in other parts of the same overlap region or in other overlap regions are taller, as exemplified by the rectangle near the right end of the overlap region in Fig. 3. Consider the shaded rectangle near the left edge of Fig. 3 whose width is w and which touches both boundaries of the overlap region. Let the height of this rectangle be $u = \Delta p - v$. We want to demonstrate that $u \geq \Delta p/2$ or $v \leq \Delta p/2$. This is easily proven by combining $\tan\theta = v/w = \Delta p/\Delta d$ with $w \leq \Delta d/2$. The latter inequality results from $w = 2^{-\delta} \leq \frac{1}{2} d^{\min}(2h-1)/(a-h) = \Delta d/2$. □

Based on the joint upper bound results of Theorem 1, only the first three cases shown in Table 2 need to be investigated. If none of these three cases proves viable, then the use of $\mathrm{lb}(\delta) + 1$ bits of precision for *d* and $\mathrm{lb}(\varepsilon) + 1$ bits of precision for *p* (per Theorem 1) is called for and is guaranteed to work. Case 4 in Table 2 is included because it might be of some interest when a redundant partial remainder is used (see Section 6).

5 Determining the Minimum Required Precision

Verification for each of the first three cases in Table 2 is relatively easy to mechanize. It consists of checking that, for all of the overlap regions, of which there are a in the first quadrant, and all consecutive pairs of vertical grid lines, of which there are $2^{\delta}(d^{\max} - d^{\min})$, a horizontal grid line exists that is totally contained in the overlap region. Referring to Fig. 3, this means that, for the points A and B of the rectangle associated with the overlap region for $x-1$ and x and the pair of vertical grid lines at $d^{\min} + 2^{-\delta}m$ and $d^{\min} + 2^{-\delta}(m+1)$, the coordinates

$$p(\mathrm{A}) = (x-1+h)(d^{\min} + 2^{-\delta}m)$$
$$p(\mathrm{B}) = (x-h)(d^{\min} + 2^{-\delta}(m+1))$$

must be such that there exists an integer multiple of $2^{-\varepsilon}$ between them, i.e., we must have:

$$\lfloor 2^{\varepsilon} p(\mathrm{A}) \rfloor \geq 2^{\varepsilon} p(\mathrm{B}).$$

TABLE 2
Cases to Be Tried before Using the Upper Bounds of Theorem 1

Case	δ	ε	Notes
1	lb(δ)	lb(ε)	Begin with the lower bounds
2	lb(δ) + 1	lb(ε)	Increase the precision of *d* first, because it is easier to deal with than *p*, which may be in redundant form
3	lb(δ)	lb(ε) + 1	Next, increase the precision of *p*
4	lb(δ) + 2	lb(ε)	Because inspecting an extra bit of *d* may be simpler than one more bit of *p*(in redundant form), try this case also

```
function adequate($r$, $a$, $d^{min}$, $d^{max}$, $\delta$, $\varepsilon$)
$h := a/(r-1)$
$x := 1$                                   /* Index of overlap region */
while $x \le a$ do                         /* Check the overlap between $x-1$ and $x$ */
    $m := 0$                               /* Index of vertical grid line */
    $n := 2^{\delta}(d^{max} - d^{min})$
    $A := 2^{\varepsilon}(x - 1 + h)d^{min}$            /* Initial value for $2^{\varepsilon}p$(A) */
    $B := 2^{\varepsilon}(x - h)(d^{min} + 2^{-\delta})$    /* Initial value for $2^{\varepsilon}p$(B) */
    $S := 2^{\varepsilon-\delta}(x - 1 + h)$            /* Step size for $2^{\varepsilon}p$(A) */
    $T := 2^{\varepsilon-\delta}(x - h)$                /* Step size for $2^{\varepsilon}p$(B) */
    while $m < n$ do                       /* Check between $m$th and $(m+1)$th grid lines */
        if $A - B \ge 1$
        then $m := n$                      /* No need to check the remaining grid lines */
        else  if $\lfloor A \rfloor \ge B$
              then $m := m + 1$            /* Proceed to checking the next grid line */
              else return **false**        /* Precision is inadequate */
              endif
        endif
        $A = A + S$
        $B = B + T$
    endwhile
    $x := x + 1$
endwhile
return **true**                            /* Precision is adequate */
endfunction adequate
```

Fig. 4. Verifying if precisions of δ bits for d^{trunc} and ε bits for p^{trunc} are adequate.

So, the algorithm, presented in Fig. 4, has two nested loops, corresponding to varying x and m, with the inner loop's termination condition being $2^{\varepsilon}p(\text{B}) - 2^{\varepsilon}p(\text{A}) \ge 1$. This latter condition is sufficient for the existence of the required horizontal grid line, but is not necessary; for example, there exists an integer between 2.75 and 3.25, even though their difference is less than 1. Note that the condition is verified only in the first quadrant of the *p-d* plot for, even though the choice of quotient digit is not symmetric about the two axes, the placement of the uncertainty regions, which affects the required precisions, is symmetric.

Note that, for the sake of clarity, the pseudocode in Fig. 4 has not been fully optimized. In an actual coding of the algorithm, the values of *A*, *B*, *S*, and *T* could be initialized for *x* = 1 outside the *x* while loop and then appropriately updated with advancing *x*. Updating of these values inside the while loop would then require only the addition of increment values in much the same way that updating of *A* and *B* in the inner *m* loop is now handled.

Note that the selection constants or, alternatively, the table entries, are obtained as byproducts of the checking function in Fig. 4. In other words, when, for a particular m, we have $\lfloor A \rfloor \ge B$, any integer in $[\lfloor A \rfloor, B]$ can be used as the selection constant to choose between $x - 1$ and x for the divisor interval $[m, m+1)$; i.e., when the truncated divisor, taken as an integer, represents m.

6 Redundant Partial Remainder

Most modern implementations of the SRT division use carry-save or BSD (binary signed-digit or borrow-save) representation of the partial remainder. Both of these representations require 2 bits in each radix-2 position. Given that redundant formats with lower representational redundancy are gaining in popularity, we base our presentation in this section on a more general format that covers binary carry-save and BSD representations as special cases [6].

Let the partial remainder s be represented as $s' + s''$, with the main component s' being a radix-2 unsigned or two's complement number and the auxiliary component s'' taking the form of a radix-2^g number whose digits are in [0, 1] or [–1, 0]. The special case of $g = 1$ covers binary carry-save and BSD representations, while $g > 1$ includes high-radix carry-save representations [4] and hybrid redundancy [11] with the recurring BSD position encoded as a pair of negative and positive bits (see Fig. 5).

Truncating a redundant partial remainder to ε bit positions usually leads to a larger maximum error than that of a nonredundant partial remainder. It is clear, however, that, for all the examples shown in Fig. 5 and, more generally, for any redundant representation that contains at most 2 bits in any given position, inspecting one additional bit of the redundant partial remainder is adequate, provided that these bits are used directly as

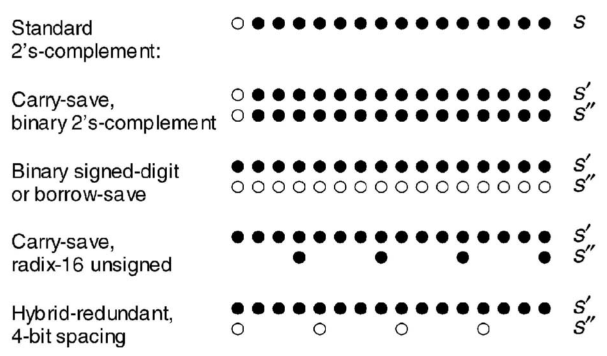
Standard 2's-complement:
s
Carry-save, binary 2's-complement
s′
s″
Binary signed-digit or borrow-save
s′
s″
Carry-save, radix-16 unsigned
s′
s″
Hybrid-redundant, 4-bit spacing
s′
s″

Fig. 5. Dot-notation representation of some 16-position numbers encoded as weighted bit sets, including both positive bits (heavy dots) and negative bits (hollow dots).

inputs to a lookup table. Such a process is practical only for small radices (perhaps only 2 and 4). For larger radices, and sometimes even for radix 2 or 4, a common compromise is to assimilate a portion of the redundant partial remainder into a truncated nonredundant estimate and then use the resulting estimate in the quotient digit selection process as before [2], [8]. This is attractive for both carry-save and BSD partial remainders because the delay due to assimilation of an l-bit segment, which is done through standard binary addition, grows rather slowly with the width l.

The assimilation-based approach needs to be further discussed as it is unclear how our preceding results can be applied in this case. A nonredundant estimate derived from assimilating a section of a redundant partial remainder has two sources of error: 1) A maximum error of just under $2^{-\varepsilon}$ due to the truncated estimate. 2) An additional error due to assimilating only l fractional bits of the redundant partial remainder (complete assimilation would, of course, defeat the purpose of having a redundant partial remainder in the first place). We need only specify how the additional error above affects the selection process.

To show what needs to be done, we redraw the lightly shaded rectangle of Fig. 3, with opposite vertices A and B touching the boundaries of the overlap region at vertical grid lines, in Fig. 6. The function "adequate" in Fig. 4 checks to see if a horizontal grid line spans the entire width of this rectangle. When we assimilate only l fractional bits of a redundant partial remainder, we commit an additional error that is equal to the sum of all the dropped bits. For example, for carry-save representation, the positive bits that are dropped can add to almost 2^{-l+1} and, for BSD representation, the bits dropped can add to a value in the open interval $(2^{-l}, 2^{l})$.

Based on the preceding discussion, we can use the function "adequate" of Fig. 4 to determine if assimilation of l bits of a redundant partial remainder would be adequate, provided A and B are initialized as follows, instead of as in Fig. 4:

$$A := 2^{\varepsilon}(x - 1 + h)d^{\min} - \text{maximum magnitude of dropped positive bits}$$

$$B := 2^{\varepsilon}(x - h)(d^{\min} + 2^{-\delta}) + \text{maximum magnitude of dropped negative bits.}$$

The corrective terms on the right-hand sides of these two expressions are easily derived, given the particular redundant format (as in the examples in Fig. 5) and the assimilation width l.

In fact, rather than test for different values of l to see which one leads to success, we can modify the function "adequate" of Fig. 4 to return the smallest acceptable value for l along with its true/false answer. If the returned value of l is unacceptable, then larger values of ε and/or δ may be tried. Note that, because the magnitude of the additional error term due to the incomplete assimilation of a redundant partial remainder is never more than

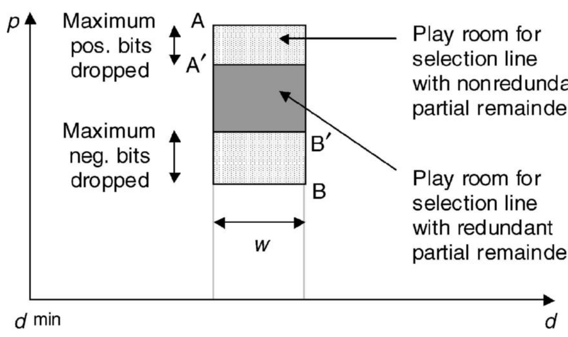
p
Maximum pos. bits dropped
A
A′
Maximum neg. bits dropped
B′
B
w
d min
d
Play room for selection line with nonredundant partial remainder
Play room for selection line with redundant partial remainder

Fig. 6. Effect of a redundant partial remainder on the uncertainty rectangle.

the original $2^{-\varepsilon}$ error term, at most a couple of cases beyond those in Table 2, namely those with $\varepsilon = \text{lb}(\varepsilon) + 2$, need to be considered. Note also that one byproduct of the lower redundancies associated with the high-radix carry-save and hybrid-redundant representations in Fig. 5 is that straight table lookup (i.e., without first assimilating a truncated redundant partial remainder into nonredundant form) might become practical where it is not with binary carry-save or BSD representation.

7 Conclusion

In the literature on computer arithmetic, it is often mentioned that determining the precisions required of truncated versions of d and p to correctly choose the radix-r quotient digits from the digit set $[-a, a]$ involves several iterations over the design space. In this paper, we have proven that the number of cases to be tried is limited to a handful, as listed in Table 2 and further augmented in Section 6, given choices for r (power-of-2 radix) and a (defining a symmetric, redundant digit set). Theorem 1, along with some of its implications to the design of high-radix dividers, constitutes the main contributions of this work.

Considering that the trials mentioned above are easily mechanized, as demonstrated in Fig. 4, and that the choice of the radix r is quite limited by cost/performance requirements, the design process is not as cumbersome as one might have thought. This is true even after variations in the format of the redundant partial remainder (Fig. 5) are taken into account. It would be interesting to see if corresponding results for high-radix square-rooting, which is intimately related to SRT division, can be derived.

A possible area for future research is that of quotient digit-set encoding. In our derivations, the chosen quotient digit q_i was treated as a single indivisible entity. However, a high-radix digit set must be encoded as a collection of bits. Possible choices include 2's-complement representation, gray-code encoding [8], and ad hoc sign-plus-weighted-bits encodings that facilitate the multiple generation process. With any particular choice of encoding, each bit in the representation of the selected quotient digit can be viewed as a separate function of the truncated divisor and partial remainder. Using separate logic circuits or tables to obtain these bits can potentially lead to savings in the total circuit complexity or table size.

Acknowledgments

A preliminary version of this paper appeared in the *Procedings of the 35th Asilomar Conference on Signals, Systems, and Computers* [9]. The anonymous referees supplied detailed comments and a few references. Their contributions to the clarity and completeness of the paper are gratefully acknowledged.

REFERENCES

[1] D.E. Atkins, "Higher Radix Division Using Estimates of the Divisor and Partial Remainders," *IEEE Trans. Computers,* vol. 17, no. 10, pp. 925-934, Oct. 1968.
[2] N. Burgess and T. Williams, "Choices of Operand Truncation in the SRT Division Algorithm," *IEEE Trans. Computers,* vol. 44, no. 7, pp. 933-938, July 1995.
[3] M.D. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations.* Kluwer, 1994.
[4] H.A.H. Fahmy, A.A. Liddicoat, and M.J. Flynn, "Improving the Effectiveness of Floating-Point Arithmetic," *Proc. 35th Asilomar Conf. Signals, Systems, and Computers*, pp. 875-879, Nov. 2001.
[5] D.L. Harris, S.F. Oberman, and M.A. Horowitz, "SRT Division Architectures and Implementations," *Proc. 13th Symp. Computer Arithmetic*, pp. 18-25, July 1997.
[6] G. Jaberipur, B. Parhami, and M. Ghodsi, "Weighted Bit-Set Encodings for Redundant Digit Sets," *Proc. 36th Asilomar Conf. Signals, Systems, and Computers,* pp. 1629-1633, Nov. 2002.
[7] S.F. Oberman and M.J. Flynn, "Division Algorithms and Implementations," *IEEE Trans. Computers,* vol. 46, no. 8, pp. 833-854, Aug. 1997.
[8] S.F. Oberman and M.J. Flynn, "Minimizing the Complexity of SRT Tables," *IEEE Trans. VLSI Systems,* vol. 6, no. 1, pp. 141-149, Mar. 1998.
[9] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs.* Oxford, 2000.
[10] B. Parhami, "Precision Requirements for Quotient Digit Selection in High-Radix Division," *Proc. 35th Asilomar Conf. Signals, Systems, and Computers,* pp. 1670-1673, Nov. 2001.
[11] D.S. Phatak and I. Koren, "Hybrid Signed-Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains," *IEEE Trans. Computers,* vol. 43, no. 8, pp. 880-891, Aug. 1994.
[12] J.E. Robertson, "A New Class of Digital Division Methods," *IRE Trans. Electronic Computers,* vol. 7, pp. 218-222, Sept. 1958.
[13] J.H.P. Zurawski and J.B. Gosling, "Design of a High-Speed Square Root, Multiply, and Divide Unit," *IEEE Trans. Computers,* vol. 36, no. 1, pp. 13-23, Jan. 1987.

New Systolic Architectures for Inversion and Division in $\mathrm{GF}(2^m)$

Zhiyuan Yan, *Student Member*, *IEEE*, and
Dilip V. Sarwate, *Fellow*, *IEEE*

Abstract—We present two systolic architectures for inversion and division in $\mathrm{GF}(2^m)$ based on a modified extended Euclidean algorithm. Our architectures are similar to those proposed by others in that they consist of two-dimensional arrays of computing cells and control cells with only local intercell connections and have $O(m^2)$ area-time product. However, in comparison to similar architectures, both our architectures have critical path delays that are smaller, gate counts that range from being considerably smaller to only slightly larger, and latencies that are identical for inversion but somewhat larger for division. One architecture uses an adder or an $(m+1)$-bit ring counter inside each control cell, while the other architecture distributes the ring counters into the computing cells, thereby reducing each control cell to just two gates.

Index Terms—Finite fields, field arithmetic, inversion, division, systolic, extended Euclidean algorithm.

1 INTRODUCTION

FINITE fields have found applications in areas such as error-control codes [1], cryptography [2], [3], and digital signal processing [4]. Desirable features of finite-field arithmetic units include small critical path delay (CPD), high computation speed (throughput), short computation delay (latency), and small area-time (AT) complexity. Following Parhi [5], we define CPD to be the total delay of the longest path between any two storage elements (or latches) and latency to be the number of clock cycles from the time at which the first bit of the input enters the architecture and the time at which the last bit of the corresponding output exits the architecture. Also, the AT complexity of an architecture is simply the product of the respective orders of the gate count and the number of cycles between successive outputs.

Inversion and division are the most complicated finite-field arithmetic operations and have received the most attention, leading to a variety of algorithms and architectures. We consider only those algorithms and architectures that can be used for any value of m and will not discuss the architectures using the recursive construction approach [15], [16], [17] that are suitable only for certain choices of m. Several of the general architectures have $O(m^2)$ AT complexity and $O(m)$ latency. These include all the previously proposed architectures that are based on the extended Euclidean algorithm (EEA) [6], [7], [8], [9], [10], [11], which have a CPD of four gates delay or more because of the complicated control signals required, the double-basis inversion architecture proposed by Hasan [19] that has a CPD of three 2-input gates delay, and the architectures based on the extended Stein algorithm (ESA) [21], [22], [23], [24], [25], of which the one proposed by Wu et al. [23] has a CPD of one 2-input AND gate delay plus one 3-input XOR gate delay. Some architectures based on Fermat's theorem [7], [12], [13], [14] or on the solution of systems of linear equations [18] achieve even smaller CPDs of two 2-input gates delay, but this is accompanied by an increase in AT complexity

• *The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1308 W. Main St., Urbana, IL 61801-2307. E-mail: yan@ifp.uiuc.edu, sarwate@comm.csl.uiuc.edu.*

Manuscript received 6 Nov. 2001; revised 1 Oct. 2002; accepted 24 Jan. 2003. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 115327.

REFERENCES

[1] D.E. Atkins, "Higher Radix Division Using Estimates of the Divisor and Partial Remainders," *IEEE Trans. Computers,* vol. 17, no. 10, pp. 925-934, Oct. 1968.

[2] N. Burgess and T. Williams, "Choices of Operand Truncation in the SRT Division Algorithm," *IEEE Trans. Computers,* vol. 44, no. 7, pp. 933-938, July 1995.

[3] M.D. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations.* Kluwer, 1994.

[4] H.A.H. Fahmy, A.A. Liddicoat, and M.J. Flynn, "Improving the Effectiveness of Floating-Point Arithmetic," *Proc. 35th Asilomar Conf. Signals, Systems, and Computers,* pp. 875-879, Nov. 2001.

[5] D.L. Harris, S.F. Oberman, and M.A. Horowitz, "SRT Division Architectures and Implementations," *Proc. 13th Symp. Computer Arithmetic,* pp. 18-25, July 1997.

[6] G. Jaberipur, B. Parhami, and M. Ghodsi, "Weighted Bit-Set Encodings for Redundant Digit Sets," *Proc. 36th Asilomar Conf. Signals, Systems, and Computers,* pp. 1629-1633, Nov. 2002.

[7] S.F. Oberman and M.J. Flynn, "Division Algorithms and Implementations," *IEEE Trans. Computers,* vol. 46, no. 8, pp. 833-854, Aug. 1997.

[8] S.F. Oberman and M.J. Flynn, "Minimizing the Complexity of SRT Tables," *IEEE Trans. VLSI Systems,* vol. 6, no. 1, pp. 141-149, Mar. 1998.

[9] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs.* Oxford, 2000.

[10] B. Parhami, "Precision Requirements for Quotient Digit Selection in High-Radix Division," *Proc. 35th Asilomar Conf. Signals, Systems, and Computers,* pp. 1670-1673, Nov. 2001.

[11] D.S. Phatak and I. Koren, "Hybrid Signed-Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains," *IEEE Trans. Computers,* vol. 43, no. 8, pp. 880-891, Aug. 1994.

[12] J.E. Robertson, "A New Class of Digital Division Methods," *IRE Trans. Electronic Computers,* vol. 7, pp. 218-222, Sept. 1958.

[13] J.H.P. Zurawski and J.B. Gosling, "Design of a High-Speed Square Root, Multiply, and Divide Unit," *IEEE Trans. Computers,* vol. 36, no. 1, pp. 13-23, Jan. 1987.

New Systolic Architectures for Inversion and Division in $\mathrm{GF}(2^m)$

Zhiyuan Yan, *Student Member, IEEE*, and
Dilip V. Sarwate, *Fellow, IEEE*

Abstract—We present two systolic architectures for inversion and division in $\mathrm{GF}(2^m)$ based on a modified extended Euclidean algorithm. Our architectures are similar to those proposed by others in that they consist of two-dimensional arrays of computing cells and control cells with only local intercell connections and have $O(m^2)$ area-time product. However, in comparison to similar architectures, both our architectures have critical path delays that are smaller, gate counts that range from being considerably smaller to only slightly larger, and latencies that are identical for inversion but somewhat larger for division. One architecture uses an adder or an $(m+1)$-bit ring counter inside each control cell, while the other architecture distributes the ring counters into the computing cells, thereby reducing each control cell to just two gates.

Index Terms—Finite fields, field arithmetic, inversion, division, systolic, extended Euclidean algorithm.

1 INTRODUCTION

FINITE fields have found applications in areas such as error-control codes [1], cryptography [2], [3], and digital signal processing [4]. Desirable features of finite-field arithmetic units include small critical path delay (CPD), high computation speed (throughput), short computation delay (latency), and small area-time (AT) complexity. Following Parhi [5], we define CPD to be the total delay of the longest path between any two storage elements (or latches) and latency to be the number of clock cycles from the time at which the first bit of the input enters the architecture and the time at which the last bit of the corresponding output exits the architecture. Also, the AT complexity of an architecture is simply the product of the respective orders of the gate count and the number of cycles between successive outputs.

Inversion and division are the most complicated finite-field arithmetic operations and have received the most attention, leading to a variety of algorithms and architectures. We consider only those algorithms and architectures that can be used for any value of m and will not discuss the architectures using the recursive construction approach [15], [16], [17] that are suitable only for certain choices of m. Several of the general architectures have $O(m^2)$ AT complexity and $O(m)$ latency. These include all the previously proposed architectures that are based on the extended Euclidean algorithm (EEA) [6], [7], [8], [9], [10], [11], which have a CPD of four gates delay or more because of the complicated control signals required, the double-basis inversion architecture proposed by Hasan [19] that has a CPD of three 2-input gates delay, and the architectures based on the extended Stein algorithm (ESA) [21], [22], [23], [24], [25], of which the one proposed by Wu et al. [23] has a CPD of one 2-input AND gate delay plus one 3-input XOR gate delay. Some architectures based on Fermat's theorem [7], [12], [13], [14] or on the solution of systems of linear equations [18] achieve even smaller CPDs of two 2-input gates delay, but this is accompanied by an increase in AT complexity

• *The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1308 W. Main St., Urbana, IL 61801-2307. E-mail: yan@ifp.uiuc.edu, sarwate@comm.csl.uiuc.edu.*

Manuscript received 6 Nov. 2001; revised 1 Oct. 2002; accepted 24 Jan. 2003. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 115327.