Radix-16 Signed-Digit Division*

Tony M. Carter
University of Utah
Dept. of Computer Science
3190 Merrill Engg. Building
Salt Lake City, Utah 84112
(801) 581-7686
carter@cs.utah.edu

James E. Robertson
University of Illinois
Dept. of Computer Science
1304 West Springfield
Urbana, Illinois 61801
(217) 333-1456

30 January 1989
Tony Carter
UUCS-88-004

VLSI Systems Research Group, Technical Report UUCS–88–004
University of Utah, Department of Computer Science

Radix-16 Signed-Digit Division

TONY M. CARTER* (*carter@cs.utah.edu*)
University of Utah
Dept. of Computer Science
3190 Merrill Engineering Building
Salt Lake City, Utah 84112

JAMES E. ROBERTSON† (*rob@m.cs.uiuc.edu*)
University of Illinois
Dept. of Computer Science
1304 West Springfield
Urbana, Illinois 61801

Keywords: Variable Precision Arithmetic, Signed-Digit Division, Computer Arithmetic

Abstract. For use in the context of a linearly scalable arithmetic architecture supporting high/variable precision arithmetic operations (integer or fractional), a two-stage algorithm for fixed point, radix-16 signed-digit division is presented. The algorithm uses two limited precision radix-4 quotient digit selection stages to produce the full radix-16 quotient digit. The algorithm requires a two digit estimate of the (initial) partial remainder and a three digit estimate of the divisor to correctly select each successive quotient digit.

The normalization of redundant signed-digit numbers requires accommodation of some fuzziness at one end of the range of numeric values that are considered normalized. A set of general equations for determining the ranges of normalized signed-digit numbers is derived. Another set of general equations for determining the precisions of estimates of the divisor and dividend required in a limited precision SRT model signed-digit division are derived. These two sets of equations permit design tradeoff analyses to be made with respect to the complexity of the model division.

The specific case of a two-stage radix-16 signed-digit division is presented. The staged division algorithm used can be extended to other radices as long as the signed-digit number representation used has certain properties.

1 Introduction

There are several significant mathematical problems for which solutions currently involve algorithms that rely on variable precision arithmetic [15]. Two of these are the combination of three-dimensional surfaces represented using b-splines [26] and Gröbner bases solutions to complex systems of equations

*Supported by DARPA under contract number DAAK11-84-K-0017.
†Supported by NSF under grant number MCS-83-08576.

[19]. The maximum precisions required may vary from a few tens of decimal digits to a thousand or more. These two algorithms currently rely upon Lisp bignums [29] implemented in software. Given machines with fixed (and not easily extended) word-widths for numeric operands, arithmetic operations on high/variable precision numbers (integers or fractions) are necessarily serial in nature. In general, the serial nature of arithmetic calculations on numbers whose precision exceeds the precision of the machine word is unavoidable.

It is in the context of developing a specialized computer architecture [3] to support arithmetic operations on high/variable precision numeric operands that this division algorithm arises. This specialized computer architecture will permit significant acceleration of algorithms that currently depend on high/variable precision arithmetic implemented in software. It will also provide flexibility in space as well as in time. The other commonly practiced techniques for implementing arithmetic units in hardware (with the exception of carry save) do not support linear expansion in space with no additional cost in time. In other words, one has the option of taking additional time with no space cost to compute with high precision but not to use additional hardware with no time cost. In general, flexibility is readily available in time, but not in space.

The structure of the architecture as shown in figure 1 is based on an arithmetic digit-slice derived from one presented by Chow [8], [7], [23], [6]. The digit-slice directly supports addition, subtraction and multiplication. It uses a redundant, radix-16 signed-digit number representation for reasons described in [8], which are summarized as follows:

- to simplify normalization (the use of maximally redundant digit sets is necessarily prohibited which eliminates the use of binary techniques such as carry save), and
- to accommodate variable precision multiplication that produces the most significant digit first (in which case the already produced product digits must not be affected by carry propagation from subsequent additions — this requires that the radix be at least 7).
- to simplify quotient digit selection which can be done in two limited precision radix-4 steps.

The use of such a number representation has some significant advantages as described in [2],[27],[28] over many other methods. Among these is the ability to design a constant-time, linear area-cost adder that permits free tradeoffs of time versus space. In general, SRT divisions are implemented as a low-precision model of the real division. Both the real division and the model division should be as simple as possible, but tradeoffs involving space and time

are necessary. In the scalable architecture shown in figure 1, there is a single *control module* that contains the hardware for the model division and a linearly scalable arithmetic unit consisting of one or more *arithmetic modules*. There are several important architectural and circuit design considerations that are impacted directly by the division algorithm:

1. tradeoffs in time and pin count for transmitting estimates of the (initial) partial remainder from the arithmetic modules to the control module,
2. the area and time complexities of normalization hardware,
3. the area in the control module consumed by the model division,
4. the complexity of the circuit designs and the arithmetic algorithms (which significantly impact design time), and
5. the architectural flexibility provided by the circuits.

One of the significant design considerations in implementing a higher radix SRT division algorithm is the required precisions of the (initial) partial remainder and the divisor in the model division[1]. In a scalable architecture such as the one shown in figure 1, a new estimate of the current partial remainder must be provided from the arithmetic modules to the control module during every cycle of the division algorithm (that is, for every quotient digit produced). In general, the use of more precise estimates of the partial remainder to the control module will require either more pins (a premium VLSI commodity) or more time.

The impact of different normalization and estimation (of the divisor and partial remainders) strategies on the model division and the architecture as a whole can, as shown in section 2.3, be assessed. In general, a lower radix of normalization implies the use of smaller estimates of the divisor and the partial remainder and therefore a simpler quotient digit selector in the model division. Normalization also carries costs in both area and time that must be considered in the design process.

The following discussions treat the aspects of division that are critical to the development of a chip set to be used in implementing a scalable, variable precision arithmetic processor. In these discussions we refer to digit sets. A digit set is a set of consecutive integers (that includes 0) from which a digit in a number may take its value. A digit-set is classified by its diminished cardinality (δ) and its offset (ω) (notation from [21] and [22]). The diminished cardinality is the number of distinct values that can be represented by a digit, minus one, and the offset is the distance from the smallest value to zero. We use the notation $< \delta.\omega >$ to represent a digit-set. As noted before, Chow's dissertation [8] details the reasons (most having to do with division) for selecting a radix of 16 and a twenty-one valued symmetric digit

set, $< 20.10 >$ (i.e., $\{\overline{10},\overline{9},\overline{8},\overline{7},\overline{6},\overline{5},\overline{4},\overline{3},\overline{2},\overline{1},0,1,2,3,4,5,6,7,8,9,10\}$ where $\overline{n}$ means $-n$). Of particular significance is that the $< 20.10 >$ symmetric digit set can also be represented as the sum of two identical radix 4 digits (i.e., $< 20.10 >= 4 < 4.2 > + < 4.2 >$ where $< 4.2 >= \{\overline{2},\overline{1},0,1,2\}$) which suggests and is critical to the two-stage division algorithm presented in this paper. The only other redundant radix-4 digit set, $< 6.3 >$, is eliminated from consideration since it is maximally redundant.

Let us first examine some general theory on the division of signed digit numbers.

2 Division of Symmetric Signed-Digit Numbers

Division is frequently implemented as an iterative process that produces one quotient digit per cycle according to the following recurrence equation

$$p_{j+1} = rp_j - q_{j+1}d \tag{1}$$

where p_j is the current partial remainder, p_{j+1} is the next partial remainder, r is the radix, q_{j+1} is the next quotient digit, and d is the divisor. The dividend (or initial partial remainder) is p_0. For division resulting in a redundant quotient, estimates of the divisor ($\widehat{d}$) and the partial remainder ($\widehat{p_j}$) may be used rather than their full-precision values in quotient digit selection [1].

Division involves two principal constraints that are enforced prior to evaluation of recurrence equation 1:

1. The quotient digit selector is simplified by restricting the range of possible divisor values through normalization of the divisor.
2. A stable algorithm is achieved by ensuring that the divisor value is no smaller than approximately $1/r$ times the dividend. Thus, an upper bound on the value of the dividend and each successive value of the partial remainder is imposed. This is assured for the dividend by pre-scaling it if necessary (the same hardware used for normalizing the divisor can be used for this pre-scaling operation). During division, the bound is enforced by proper selection of quotient digits and the evaluation of equation 1.

In the following discussions fractional numbers are assumed to be of the form

$$\mathbf{X} = \sum_{i=0}^{m} r^{-i} x_i$$

where the digits x_i are selected from the symmetric digit set

$$\{\overline{n}, \overline{n-1}, \cdots, 0, \cdots, n-1, n\}.$$

It should be noted that the maximum value for such a number is

$$|\mathbf{X}_{max}| = n\sum_{i=0}^{m} r^{-i} < n \lim_{m\to\infty} \sum_{i=0}^{m} r^{-i} = \frac{n}{1-\frac{1}{r}} = \frac{rn}{r-1}.$$

2.1 Normalization of Symmetric Signed-Digit Numbers

In SRT division algorithms for floating point numbers, the divisor and the dividend are maintained in normalized (or pre-scaled) form. That is, the leading non-zero digits of fractions or the mantissas of floating-point numbers are aligned so that the shift and subtract operation of division will result in

$$|p_{j+1}| \leq \left(\frac{n}{r-1}\right) d = kd \tag{2}$$

where the constant $k = \left(\frac{n}{r-1}\right)$ is called the coefficient of redundancy. If $k = 1$ the number is maximally redundant and if $k = \frac{\lfloor \frac{r}{2} \rfloor}{r-1}$ the number is minimally redundant. If $k = \frac{1}{2}$ the number is non-redundant, but this can occur only for odd radices.

In SRT division of variable precision integers, the divisor is normalized but it is not necessary to pre-scale the dividend (using the normalization hardware); however in higher radix division the cost for not doing so will be a more expensive division. Given the number of digits in the divisor and dividend and the number of shifts required to "normalize" each one, the number of significant digits in the quotient can be precomputed. In essence, leading zeroes in the quotient are computed and discarded during the normalization/ pre-scaling process which involves only a controlled shift rather than in the considerably more expensive quotient digit selection process and evaluation of equation 1. In integer SRT division, the remainder will need to be denormalized following the division in any case; this represents no extra cost for the proposed algorithm.

In non-redundant systems, a number $\mathbf{X}$ is considered normalized if $\frac{1}{r} \leq \mathbf{X} \leq 1$. In redundant systems, there is some fuzziness introduced at either the upper or lower bound so that the range of normalized numbers is extended somewhat due to multiple representations of a single arithmetic value, some of which are normalized and some of which are not. This is understood by noting that in a redundant system the value range of the part not examined for determining normalization is greater than in a non-redundant system. In short, normalization of symmetric signed-digit numbers for a given radix r may be achieved in two ways [20], [5]:

- Guarantee a lower bound of $\frac{1}{r}$, allowing an upper bound in excess of 1.
- Guarantee an upper bound of 1, allowing a lower bound of less than $\frac{1}{r}$.

We choose the first method since it will result in a larger minimum value for a normalized divisor, potentially simplifying quotient digit selection.

In general, a fractional radix-r number $\mathbf{X}$, composed of symmetric signed digits, is normalized (ignoring the special case where $\mathbf{X} \equiv 0$, i.e. $x_i = 0 (i \in \{0, 1, \ldots, M\})$) if

$$x_0 = 0 \text{ and } |r^{-1}x_1 + r^{-2}x_2| > r^{-1} \tag{3}$$

or

$$|x_0| = 1 \text{ and } |x_0 + r^{-1}x_1| \leq 1. \tag{4}$$

Equation 3 requires the $r^{-2}x_2$ term since the sum of all other remaining digits can be either positive or negative. If, for example, $x_0 = 0$, $x_1 = 1$, $x_2 = 0$ and $x_3 = \overline{1}$ and if x_2 is not examined, the value of the number would be less than r^{-1} and the number would not be normalized. Since

$$r^{-2} > n\sum_{i=3}^{m} r^{-i}$$

the contribution of $x_2 = 1$ to the value of the number cannot be fully eliminated even if the contributions of the remaining digits are of opposite sign. Similarly, the x_1 term is necessary in equation 4 in that a negative value for x_1 will reduce the value of the number below 1 and the positive contributions of all the remaining digits cannot make it greater than 1. Equation 4 means that the sign of x_0 must not be equal to the sign of x_1.

When redundant number representations are used for the divisor and/or dividend of an SRT division [24, 14], normalization becomes a problem when maximal redundancy is used. For example, consider the maximally redundant radix-10 number $1.\overline{999}$ (where $\overline{9}$ means -9). Under the definitions of normalized numbers (see equations 3 and 4), $1.\overline{999}$ is considered normalized, but its true value 0.001 is definitely not normalized.

It is essential [8] that $\lceil \frac{r-1}{2} \rceil \leq n \leq r-2$ since if $n = r-1$ the normalization procedure must incorporate recoding of small values. Equation 4 dictates that n is less than $r-1$. In this equation, when $x_0 = 1$ and all other $x_i = \overline{n}$, the minimum value for a normalized, infinitely long number will be $1 - (\frac{rn}{r-1} - n)$. If $n = r-1$ this value approaches zero as the word width approaches infinity. For all finite word widths, this value is given by a 1 in the least significant digit which is the smallest representable number. If maximal redundancy is allowed, normalization would require the use of a recoding procedure to eliminate such representations. If we limit, as Chow suggests in [8], $n \leq r-2$ the value of the smallest normalized number is $1 - (\frac{r(r-2)}{r-1} - (r-2)) = \frac{1}{r-1} > \frac{1}{r}$ and no recoding is necessary. Therefore, no maximally redundant technique, including carry save, can be used.

Whether or not shifts are required for normalization is easily determined by examining only the three most significant digits of a number as dictated by equations 3 and 4. Right shifts are required when

$$|x_0| > 1 \text{ or } (|x_0| = 1 \text{ and } |x_0 + r^{-1}x_1| \geq 1)$$

and left shifts are required when

$$x_0 = 0 \text{ and } |r^{-1}x_1 + r^{-2}x_2| \leq r^{-1}.$$

We must demonstrate that the normalization process is stable and we must determine bounds on the values of normalized numbers to compute the precisions of estimates required in the model division. The following analysis gives both of these by determining the ranges of numbers that absolutely need normalization, may need normalization, or which are certainly normalized.

Equation 3 tells us that if a number is less than $\frac{1}{r}$ then it will require left shifts to become normalized. The maximum unnormalized value for X occurs when $x_0 = 0, x_1 = 1, x_2 = 0$ and all the remaining $x_i = n$. For values greater than this, no left shift is required. For values less than this, left shifts *may or may not* be required. This value is given by the equation:

$$r^{-1} + n\sum_{i=3}^{m} r^{-i} < r^{-1} + n\left(\frac{r}{r-1} - 1 - \frac{1}{r} - \frac{1}{r^2}\right) = \frac{1}{r} + \frac{n}{r^2(r-1)} = S_{lmax}. \quad (5)$$

Furthermore, the minimum value that may be normalized occurs when $x_0 = 0, x_1 = 1, x_2 = 1$ and all the remaining $x_i = \overline{n}$. For values greater than this a left shift *may or may not* be required and for values less than this left shift(s) are required for normalization. This value is given by the equation:

$$r^{-1} + r^{-2} + \overline{n}\sum_{i=3}^{m} r^{-i} < r^{-1} + r^{-2} - n\left(\frac{r}{r-1} - 1 - \frac{1}{r} - \frac{1}{r^2}\right) = \frac{1}{r} + \frac{1}{r^2}\left[1 - \frac{n}{r-1}\right] = S_{lmin}. \quad (6)$$

Equation 4 gives us the upper bounds on normalized numbers. First, if $x_0 = 1$ and $x_1 = 0$ and all other $x_i = n$, the upper bound on normalized numbers is given by:

$$1 + n\sum_{i=2}^{m} r^{-i} < 1 + n\left(\frac{r}{r-1} - 1 - \frac{1}{r}\right) = 1 + \frac{n}{r(r-1)} = S_{rmax} \quad (7)$$

and right shift(s) will be required to avoid overflow for all numbers above this value.

Furthermore, the lower bound on numbers requiring a right shift occurs when $x_0 = 1, x_1 = 1$ and all other $x_i = \overline{n}$. This value is given by

$$1+\frac{1}{r}-n\sum_{i=2}^{m} r^{-i} \;<\; 1+\frac{1}{r}-n(\frac{r}{r-1}-1-\frac{1}{r}) \;=\; 1+\frac{1}{r}-\frac{n}{r(r-1)} \;=\; S_{rmin} \quad (8)$$

In general,

if (		$\lvert X\rvert$	$< S_{lmin}$	) left shift(s) are required
if (	S_{lmin}	$< \lvert X\rvert$	$< S_{lmax}$	) a left shift may or may not be required
if (	S_{lmax}	$< \lvert X\rvert$	$< S_{rmin}$	) **X** is certainly normalized
if (	S_{rmin}	$< \lvert X\rvert$	$< S_{rmax}$	) a right shift may or may not be required
if (	S_{rmax}	$< \lvert X\rvert$		) right shift(s) are required.

Given this method of determining whether or not a shift is to be made, the values of S_{lmin} and S_{rmax} represent the bounds of numbers which may be normalized (although some numbers within those bounds may not be). They are important in determining the precisions required in the model division. It is important to note that

$$rS_{lmax} = S_{rmax}$$

$$\frac{1}{r}S_{rmin} = S_{lmin},$$

indicating that a left (or right) shift of a number within the range of normalized numbers but which is not considered normalized will not result in the number being outside the range of normalized numbers. This demonstrates that the normalization process is indeed stable.

2.2 Quotient Digit Selection

Once the divisor and dividend (initial partial remainder) are normalized, equation 1 can be evaluated until the desired number of quotient digits q_i have been generated. In general, quotient digit selection is a process of comparing the magnitude of the divisor with that of the partial remainder and deciding which value(s) of the quotient digit will result in the next partial remainder adhering to equation 2.

Consider the P-D plot in figure 2. This plot indicates the various regions in which certain values for the quotient digit must or may be selected. Note that there are *uncertainty regions* in which one of two quotient digit values may be selected. In the process of quotient digit selection, we must concern ourselves primarily with the behavior of quotient digit selection in these uncertainty regions. The lines bounding these regions are given by a set of equations

$$rp_j \;=\; (\varphi \pm k)d, \; (\varphi \in \{\overline{n}, \overline{n-1}, \cdots, 0, \cdots, n-1, n\})$$

It should be noted that the lines

$$rp_j = (\varphi \pm \frac{1}{2})d$$

run exactly down the middle of each uncertainty region. If $k = \frac{1}{2}$, there are no uncertainty regions since there is no redundancy. If $k = 1$, there are only uncertainty regions (except for the regions outside $rp_j = \pm nd$) since there is maximal redundancy.

Although we could use stairstep solutions or compare partial remainder values to any set of lines within the uncertainty regions, we prefer to compare rp_j with $(\varphi \pm \frac{1}{2})d$ to determine the quotient digit since this is simple, gives a relatively even distribution of quotient digits and simplifies the formation of the divisor multiples in the model division relative to other possible values.

In selecting a quotient digit, we can (except in the case of non-redundant numbers) use estimates of the divisor $(\widehat{d})$ and the partial remainder $(\widehat{p_j})$. The precisions required of these estimates is governed by the angular width (α) of the uncertainty regions. As the angular width of the uncertainty regions decreases, the required precisions of the divisor and partial remainder estimates increase. As the redundancy coefficient increases, the required precision of these estimates decreases because the uncertainty regions become wider. It is obvious from this discussion and figure 2 that the worst case for precision of the estimates is in selecting between n and $n-1$ (or between $-n$ and $-n+1$) as a quotient digit since the angular width of this uncertainty region is at a minimum.

The equations for the lines bounding the uncertainty region for selection between quotient digit values n and $n-1$ are

$$rp_j = (n - \frac{n}{r-1})d$$

and

$$rp_j = (n - 1 + \frac{n}{r-1})d.$$

Figure 3 shows these lines and the *comparison line*

$$rp_j = (n - \frac{1}{2})d$$

that will be used in making the comparison for quotient digit selection. Also shown is a rectangle centered about a specific value of d and its intersection with the comparison line. This box represents the area of this uncertainty region that is covered by estimates of d and rp_j. Obviously, it has its smallest dimensions (and therefore its largest precision requirement) when $d = d_{min}$.

There are three principal contributors in determining the precision needed for the estimates $\widehat{d}$ and $\widehat{p_j}$: divisor normalization, the technique used to form the estimates, and the comparison value (σ) used to select between quotient digit values. The smallest value of a normalized divisor, $d_{min} = S_{lmin}$, is also used in determining the precision required of the estimates.

2.3 Precision of Estimates for Divisors and Partial Remainders

The estimates ($\widehat{d}$ for d and $\widehat{p_j}$ for p_j) are assumed to be made by truncation such that

$$|d - \widehat{d}| \leq k\rho^{-\epsilon}$$
$$|p_j - \widehat{p_j}| \leq k\varrho^{-\eta}$$

In these equations, ϵ and η respectively indicate the required precisions of the divisor and partial remainder. ρ and ϱ indicate the way in which the divisor and partial remainder are normalized. As seen before, normalization also has an effect on the value of d_{min}.

If we consider the rectangle in figure 3, we note that the left and right edges of the rectangle are set by the value of ϵ and the top and bottom edges are set by the value of η. The following two equations place a bound on the length of the left and right edges respectively (they can be read: the y coordinate of the top left corner minus the height of the rectangle must be on or above the comparison line, and the y coordinate of the bottom right corner plus the height of the rectangle must be on or below the comparison line). In these equations, $v = (n - 1 + \frac{n}{r-1})$ and $w = (n - \frac{n}{r-1})$. Note that $\epsilon, \eta \geq 1$ indicating that (at whatever radix) at least one digit to the right of the radix point is required in the model division.

$$v(d_{min} - k\rho^{-\epsilon}) - k\varrho^{-\eta} \geq \sigma d_{min} \implies d_{min} \geq \frac{k}{v - \sigma}(v\rho^{-\epsilon} + \varrho^{-\eta}) \quad (9)$$

$$w(d_{min} + k\rho^{-\epsilon}) + k\varrho^{-\eta} \leq \sigma d_{min} \implies d_{min} \geq \frac{k}{w - \sigma}(w\rho^{-\epsilon} + \varrho^{-\eta}) \quad (10)$$

For any given signed-digit number system, we can compute the values of k, d_{min}, v, and w, choose a value for σ, pick values for ρ and ϱ based on the normalization technique used, and then iteratively compute a set of acceptable minimum values for ϵ and η. In general, it is best to pick $\sigma = n - \frac{1}{2}$ since $\frac{1}{2}\widehat{d}$ can be simply formed.

3 Radix-16 Signed Digit Division

Having examined some of the general theory for signed digit division, let us turn our attention to the specific case of radix-16 division for the < 20.10 > =

$4 < 4.2 > + < 4.2 >$ digit set.

3.1 Radix-16 Normalization

For radix 16 division, we will perform radix 16 normalization on both the dividend and the divisor. For this, $r = 16$, $n = 10$, and from the equations for S_{lmin} and S_{rmax} we are assured that, after radix-16 normalization, $\frac{25}{24} \geq \mathbf{X} \geq \frac{49}{768}$. From equations 3 and 4 and without respect to how the numbers are represented, right shifts are performed for radix-16 normalization when (the x_i are radix-16 digits)

$$((x_0 > 1)\ or\ (x_0 < -1)\ or\ (x_0 = 1\ and\ x_1 > 0)\ or\ (x_0 = -1\ and\ x_1 < 0))$$

and left shifts are performed when

$$((x_0 = 0)\ and\ ((x_1 = 0)\ or\ (x_1 = 1\ and\ x_2 \leq 0)\ or\ (x_1 = -1\ and\ x_2 \geq 0))).$$

3.2 Radix-4 Normalization

When each digit in a radix-16 number is represented by the sum $4 < 4.2 > + < 4.2 >$, we can easily perform radix-4 normalization by shifting. This will further restrict the range of divisor values and thereby simplify quotient digit selection and the model division. We have two options: either perform radix 16 normalization before radix 4 normalization or perform only radix 4 normalization. In either case, if both the initial partial remainder and the divisor are normalized to radix 4 there can be a mismatch in the number of shifts required for normalization. This could cause either the initial partial remainder or the divisor to be multiplied by an extra factor of four, requiring either that the other one be multiplied by four prior to normalization and division or that the quotient and remainder be suitably adjusted after division. When using the $4 < 4.2 > + < 4.2 >$ digit set, it is obvious that multiplication by 4 is easily accomplished by a single half-digit left shift.

We suggest that a radix-16 normalization step precede radix-4 normalization. Since $\frac{25}{24} < \frac{13}{12}$, no right shifts of a number will ever be required for radix-4 normalization after radix-16 normalization has taken place. In addition, at most two left shifts will be required.

We also suggest that radix-4 normalization be carried out only on the divisor. If left half-digit shifts are required for radix-4 normalization of the divisor, then the as yet unnormalized dividend should also shifted left correspondingly. If this is done, no post-processing of the quotient will be required and the only denormalization of the remainder is required.

For radix-4 normalization, $r = 4$, $n = 2$, and from the equations for S_{lmin} and S_{rmax} we are assured that, after radix-4 normalization, $\frac{7}{6} \geq \mathbf{X} \geq \frac{13}{48}$. From

equations 3 and 4 and without respect to how the numbers are represented, left shifts are performed for radix 4 normalization when (the x_i are radix 4 $< 4.2 >$ digits)

$$((x_0 = 0)\ and\ ((x_1 = 0)\ or\ (x_1 = 1\ and\ x_2 \leq 0)\ or\ (x_1 = -1\ and\ x_2 \geq 0))).$$

3.3 Radix-2 Normalization

It is also conceivable to further restrict range of the divisor by performing a radix-2 normalization step. The number representation $4 < 4.2 > + < 4.2 >$ does not lend itself as readily to radix-2 normalization, but conditional doubling of a number is easily accomplished by adding it to itself.

As with radix-4 normalization, if radix-2 normalization requires that the divisor be multiplied by two, then the unnormalized dividend should also be pre-scaled by a factor of two to eliminate any unnecessary post-processing of the quotient and remainder. Also, it is assumed that radix-2 normalization would only be done after radix-4 normalization so that only a single multiplication by two would be required.

Before radix-2 normalization, a number $\mathbf{X}$ would already be in the range $\frac{13}{48} \leq \mathbf{X} < \frac{7}{6}$. Numbers require doubling for radix-2 normalization if

$$\frac{1}{2} \geq |x_0 + 4^{-1}x_1 + 4^{-2}x_2| \geq \frac{1}{4} \tag{11}$$

Thus, doubling is required for radix 2 normalization when (the x_i are $< 4.2 >$ digits)

$$\begin{aligned}((x_0 = 0 \quad and \quad & ((x_1 = 2\ and\ x_2 \leq 0)\ or\ (x_1 = 1\ and\ x_2 > 0)\ or \\ & (x_1 = -1\ and\ x_2 < 0)\ or\ (x_1 = -2\ and\ x_2 \geq 0)))\ or \\ (x_0 = 1 \quad and \quad & x_1 = -2\ and\ x_2 \leq 0)\ or \\ (x_0 = -1 \quad and \quad & x_1 = 2\ and\ x_2 \geq 0)).\end{aligned} \tag{12}$$

The minimum positive value of $\mathbf{X}$ after radix 2 normalization occurs when $x_0 = 0, x_1 = \overline{2}, x_2 = 1$ or when $\frac{29}{48} \geq \mathbf{X} \geq \frac{25}{48}$. Thus, after radix-2 normalization, $\frac{7}{6} \geq \mathbf{X} \geq \frac{25}{48}$ will hold.

4 Radix-16 Quotient Digit Selection

The quotient digit selector for radix-16 division is based on estimates $\hat{d}$ of the divisor and $\hat{p_j}$ of the partial remainder. It involves a limited precision *model division* rather than a full precision division. This is essential when

multi-word numeric arguments are used. Given that the radix 16 digit-set is represented as $< 20.10 > = 4 < 4.2 > + < 4.2 >$, the following two methods of quotient digit selection come to mind. The first is straightforward and complex to implement while the second is somewhat less straightforward and much simpler to implement.

Algorithm 16

1. Present $\widehat{d}$ to the model division to the precision for radix 16 division and set $j = 0$.
2. Present $\widehat{p_j}$ to the model division to the precision for radix-16 division.
3. Select q_{j+1} using $\widehat{d}$ and $\widehat{p_j}$ to the precision for radix-16 division.
4. Perform a full precision evaluation of equation 1.
5. Until enough quotient digits have been generated, increment j and go to step 2.

Algorithm 16 generates a radix-16 quotient digit in a single step. It has the advantage of not requiring any intermediate sequential steps. It has a distinct disadvantage in that $r\widehat{p_j}$ must be compared with twenty different numbers $(\alpha+\frac{1}{2}\widehat{d},\ \alpha \in \{-10,\cdots,0,\cdots,9\})$. This represents a prohibitively large amount of parallel hardware required for fast quotient digit selection. Additionally, the larger multiples of the divisor are relatively difficult to compute.

Algorithm 16.4

1. Present $\widehat{d}$ to the model division to the precision for radix-16 division and set $j = 0$.
2. Present $\widehat{p_j}$ to the model division to the precision for radix-16 division.
3. Select q'_{j+1} using $\widehat{d}$ and $\widehat{p_j}$ to the precisions for radix-4 division.
4. Evaluate $\widehat{p'_{j+1}} = r\widehat{p_j} - q'_{j+1}\widehat{d}$ to the precision for radix-16 division.
5. Select q''_{j+1} using $\widehat{d}$ and $\widehat{p'_{j+1}}$ to the precisions for radix-4 division.
6. Form $q_{j+1} = 4q'_{j+1} + q''_{j+1}$.
7. Perform a full precision evaluation of equation 1 using q_{j+1} as determined in step 6.

8. Until enough quotient digits have been generated, increment j and go to step 2.

Algorithm 16.4 involves a two-stage quotient digit selector. It generates a radix-16 quotient digit in two sequential, limited precision, radix-4 steps. It has the advantage of comparing $r\widehat{p_j}$ to only four different numbers ($\alpha + \frac{1}{2}\widehat{d}$, $\alpha \in \{-2, -1, 0, 1\}$). In reality, when the signs of the divisor and partial remainder are known (as when both are normalized), only two comparisons need be made. In this case where the divisor is normalized to a smaller radix than the dividend, intermediate partial remainders with many leading zeros are not a problem since when the magnitude of the partial remainder is less than that of the divisor the quotient digit selector will generate a zero. Only when a non-zero digit has been shifted into x_1 (for which the sign is easily known) will rp_j be sufficiently large to require generation of a non-zero quotient digit. Two comparisons represent a reasonable amount of parallel comparison hardware for quotient digit selection.

Algorithm 16.4 has been extensively simulated using a functional model of the hardware required to implement it. The model uses radix-16 normalization, followed by radix-4 and radix-2 normalizations as described in sections 3.1, 3.2 and 3.3. Quotient digit selection is performed using 2 radix-16-digit estimates of the partial remainder and 3 radix-16-digit estimates of the divisor. The model is written in Common Lisp and the results of the functional model applied to inputs of random length and value were compared with results from the standard bignum package in Common Lisp. In fact, during the simulations an error in the bignum package was discovered.

In forming $\frac{1}{2}d$ (by doubling d and then shifting one half-digit to the right) for use in the comparison circuitry of the quotient digit selector, only a few of the divisor's most significant digits need be used. The formation of $\frac{1}{2}d$ is performed only once at the beginning of the division algorithm.

As was noted before, the methods used for normalization and estimation of both d and p_j are significant in determining the complexity of the quotient digit selector. They are also important in determining any extra pre-division operations that must be done on the initial partial remainder. Tables 1 and 2 illustrate some of the vast number of possibilities available. Table 1 contains required precisions and annotations of required pre-division operations for the generation of quotient digits by Algorithm 16. Table 2 contains required precisions and annotations of required pre-division operations for the generation of quotient digits by Algorithm 16.4. In these tables, the following column labels and notations are used:

- The **Name** columns refers to the total number of radix-16 digits in the model division.

- The r_{norm} columns give the normalization radices for the divisor and the dividend.
- The r_{est} columns give the radix at which the estimates of d and p_j are taken.
- The $r = 16$ columns give the best solutions to equations 9 and 10 for radix-16 division. These values represent the number of digits at the estimation radix for, respectively, d and p_j.
- The $r = 4$ columns give the best solutions to equations 9 and 10 for radix-4 division. These values represent the number of digits at the estimation radix for, respectively, d and p_j.
- The D_{model} columns indicate the number of radix-16 digits of d and p_j required in the model division.
- The $D_{compare}$ columns indicate the number of radix-16 digits of the d andp_j required in the radix-4 comparisons for Algorithm 16.4.
- The **Preprocessing** column contains three possible entries:
  - **none**: no dividend preprocessing other than radix-16 normalization is required:
  - **4R**: the dividend may need to be multiplied by 4 or 16 before normalization if the divisor required shifts for radix-4 normalization.
  - **2R**: the dividend may need to be doubled before normalization if the divisor was doubled to provide radix-2 normalization.

In these algorithms, only the divisor is directly subjected to radix-4 or radix-2 normalization. It may be that multiplications by 4 and/or 2 are required to normalize the divisor. If this is the case, then we similarly preprocess the dividend by multiplying by 4 and/or 2 to avoid any complex post-division processing of the quotient or remainder. Radix-16 normalization is performed on the dividend only after these pre-multiplications so that the initial partial remainder is within the normalized number range. Of course, the remainder can be denormalized simply by shifting right by the number of radix-16 left-shifts performed in normalizing the dividend.

In building tables 1 and 2 many clearly unacceptable alternatives were noted. These are not included. For Algorithm 16, there are 141 possible implementations, of which only the fifteen best are shown. For Algorithm 16.4, there are 276 possible implementations. This is reduced to 132 after eliminating those that require more digits of either the divisor or the partial remainder in the comparison hardware than in the model. Of these 132, only the 28 best are shown in the table. The "best" ones were selected because they required the fewest radix-4 or radix-2 normalization steps.

4.1 Length of the Radix 16 Signed-Digit Quotient

In division with a fixed word width, one simply generates that number of quotient digits (or perhaps one more as a "guard" digit). For division of variable precision numbers, we must be able to compute the required number of quotient digits prior to the division operation. Assuming we know the number of digits (excluding leading zeros) in an integer, the number of quotient digits can be easily estimated for non-redundant systems from $N_q = 1 + (N_{p_0} - N_d)$ (N represents the number of digits in a number). This is unfortunately not adequate for redundant systems. Consider the radix-16 number $1\overline{8}_{16}$ which can also be represented as 08_{16}. This demonstrates that if the leading two non-zero digits of a number can be represented with fewer digits then the number of quotient digits will need to be adjusted. Furthermore, since we use normalization methods for aligning the divisor and dividend, it is possible to get an extra shift in either (when one results in a non-zero leading digit and the other results in a zero valued leading digit), thereby requiring a further adjustment in the number of quotient digits.

4.2 Restoration in Radix-16 Signed-Digit Division

It is required that the sign of any remainder from a division be equivalent to the sign of the initial partial remainder. Thus, at the end of the division operation the sign of the remainder must be compared with the sign of the initial partial remainder. If there is a sign change, the quotient must be incremented or decremented by one (depending on the direction of the sign change and the sign of the divisor) and the divisor must be appropriately added to or subtracted from the remainder. Finally, in integer division the remainder must be returned to integer form through a denormalization process. This can be done by keeping track of the number of left shifts applied to the partial remainder both for normalization and quotient digit generation. At the conclusion of the division operation, a right shift is performed for each left shift to return the number to integer form.

5 Conclusions

This radix-16 division algorithm, although apparently complex when compared to most binary or radix-4 techniques, was developed in the very specific context of a variable precision processor and is subject to some rather stringent constraints:

- All variable precision operations may produce results *most significant digit first.* This requires a radix of no less than seven. To achieve simplicity in the implementation, a symmetric signed-digit number represen-

tation is required.

- All operands are redundant. While all SRT division algorithms result in a redundant quotient and may use the redundant carry-save form of partial remainders, none that we know of use a redundant divisor.

Given these constraints, the division algorithms recently proposed by Magenheimer for the HP Precision Architecture [17, 18], by Ercegovac [10] in which redundant radix-2 quotient digits are converted on the fly to the non-redundant two's complement form, by Fandrianto [12] in which a radix-4 SRT divider is combined with a radix-4 square-root extractor (although beyond the scope of this paper, the division hardware required for our division algorithm can be easily modified to support radix-16 square-root extraction), by Ercegovac and Lang [9] in which a radix-4 divider is based on a different recurrence equation [11], $p_{j+1} = r(p_j - q_{j+1}d)$, in which both the partial remainder and the divisor are shifted simultaneously and by Kuninobu [16] in which an array divider is based on a redundant radix-2 number representation are not applicable to the problem of division in a variable precision processor. Each of these has its initial operands in non-redundant, two's complement form and requires removal of redundancy from the quotient and remainder for storage. Furthermore, all assume fixed precision arithmetic units and none appear to be applicable to higher radix (radix-16) division.

It might be argued that operands and results could be stored in carry save form, however as shown in section 2.1 and in [8], maximally redundant number presentations (which binary carry save is) are very difficult to normalize for SRT type divisions.

Taylor describes the only other radix-16 division algorithm in the literature [25]. One of his techniques is based on overlapping two successive stages of radix-4 quotient digit selection with the formation of the divisor multiple and the next partial remainder. Using this technique, the time involved in a fractional division can be potentially reduced to the time required for quotient digit selection. Although the implementation discussed by Taylor for division and quotient digit selection in particular cannot be used in the variable precision processor, the technique for overlapping the two successive radix-4 stages of quotient digit generation may be applicable in implementing the division algorithm described in this paper.

The equations relating to and techniques for normalization are applicable to signed-digit number representations of any radix, as are the equations which compute the required precisions of the divisor and dividend in a model SRT division. The multi-level normalization process (in this case radix-16, followed by radix-4, followed by radix-2) is also applicable to any radix. The signed-digit number representation chosen will, however, have a significant impact on the difficulty of performing the normalization operations.

One-stage generation of higher radix quotient digits becomes prohibitively expensive since far too many comparisons must be made. When they are made sequentially the time cost is too high and when they are made in parallel the hardware cost is too high. When each digit of a number can be represented as a sum of equally weighted identical digit-sets with smaller diminished cardinality, then the staged division algorithm presented here is directly applicable. For example, radix-64 division could be implemented using three successive radix-4 quotient digit generation stages in the model division when the number representation is

$$< 84.42 >= 16 < 4.2 > +4 < 4.2 > + < 4.2 > .$$

Radix-36 division could be implemented using two successive radix-6 quotient digit generation stages in the model division when the number representation is

$$< 56.28 >= 6 < 8.4 > + < 8.4 > .$$

Of course, higher-radix division does not make much sense unless hardware is available for computing the next partial remainder (as in equation 1) in a single multiply/subtract operation.

Algorithm 16.4, developed specifically for use in a specialized arithmetic hardware [3, 4] for operating on high/variable precision numbers using a redundant signed-digit number representation, is well suited for implementation in VLSI since the hardware required to implement the recursion gives constant addition/subtraction time with a linear increase in area and a very simple interconnection strategy. All of the division recursion hardware is directly used for other supported arithmetic operations. It is being implemented as part of a two chip CMOS VLSI chip-set from which a variable precision arithmetic processor will be constructed.

Numbers are stored in radix-16 redundant form, eliminating the need for costly conversions to and from two's complement while incurring only a 25% increase in storage requirements. The cycle time for the generation of each quotient digit will consist of the time to evaluate the full-precision radix-16 recurrence equation (essentially a multiply-subtract operation which has a time complexity of O(1) [3]), plus the time for shifting the partial remainder one digit to the left, plus twice the time to generate a radix-4 quotient half-digit, plus the time to evaluate a limited precision radix-4 recurrence equation.

Given the variable precision architecture in which this division algorithm operates, a number of optimizations in the control module are possible that make division less costly on average. First, it need never generate more quotient digits than are necessary. If two numbers of roughly similar size are the operands for division, very few quotient digits will be generated. In fact, if a divisor is more than one digit larger than the dividend the division algorithm

will not be invoked and a constant zero will be returned. Second, the remainder of a division need not be denormalized if it will not be subsequently used. Third, division by powers of four (or sixteen) can be optimized to shifts even though division by powers of two cannot. Fourth, optimization of division by two can be achieved by adding a number to itself and then shifting right by one half-digit.

When compared to the cost of the variable-precision arithmetic modules, the model division portion of the proposed division algorithm represents only a small portion of the cost of the control module and of the system as a whole. Using the proposed algorithm, special-purpose variable-precision hardware can be used to compute division of very-high precision integers several orders of magnitude more quickly than with available software packages [4].

References

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

2. A. Avizienis, "Signed-digit number representations for fast parallel arithmetic", *IRE Trans. on Electronic Computers*, vol. EC-10, No. 9, Sep. 1961, pp. 389-400.

3. T. M. Carter, "Cascade: Hardware for High/Variable Precision Arithmetic", *Proceedings of the* 9^{th} *Symposium on Computer Arithmetic*, Sep. 1989.

4. T. M. Carter, "Cascade: A Hardware Alternative to Bignums", Tech. Report UUCS-89-006, Univ. of Utah, Dept. of Computer Science, Apr. 1989.

5. T. M. Carter and J. E. Robertson, "Radix-16 Signed Digit Division", Tech. Report UUCS-88-004, Univ. of Utah, Dept. of Computer Science, Mar. 1988.

6. T. M. Carter and L. A. Hollaar "The Implementation of a Radix-16 Digit-Slice Using a Cellular VLSI Technique", *Proceedings of ICCD 1983*, Nov. 1983, pp. 688-691.

7. C. Y. F. Chow, "A Variable Precision Processor Module", *Proc. IEEE Int'l Conf. on Computer Design*, Nov. 1983, pp. 692-695.

8. C. Y. F. Chow, *A Variable Precision Processor Module*, Ph.D. Diss., Univ. of Ill. at Urbana-Champaign, Dept. of Comp. Science, July 1980.

9. M. D. Ercegovac and T. Lang, "A Division Algorithm with Prediction of Quotient Digits", *Proc. 7th Symp. on Computer Arithmetic*, June 1985, pp. 51-56.

10. M. D. Ercegovac and T. Lang, "An Area-Time Efficient Binary Divider", *Proc. IEEE Int'l Conf. on Computer Design*, 1987, pp. 645-648.

11. M. D. Ercegovac, "A Higher-Radix Division with Simple Selection of Quotient Digits", *Proc. 6th Symp. on Computer Arithmetic*, June 1983, pp. 94-98.

12. J. Fandrianto, "Algorithm for High Speed Shared Radix 4 Division and Radix 4 Square Root", *Proc. 8th Symp. on Computer Arithmetic*, May 1987, pp. 73-79.

13. M. Hill et al, "Design Decisions in SPUR", *IEEE Computer*, vol. 19, no. 10, Nov. 1986, pp. 8-22.

14. K. Hwang, *Computer Arithmetic Principles, Architecture and Design*, Wiley and Sons, 1979.

15. D. W. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, Reading, MA: Addison-Wesley, 1981, pp. 250-265.

16. S. Kuninobu et al, "Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation", *Proc. 8th Symp. on Computer Arithmetic*, May 1987, pp. 80-86.

17. D. J. Magenheimer, L. Peters, K. W. Pettis and D. Zuras, "Integer Multiplication and Division on the HP Precision Architecture", *IEEE Trans. on Computers*, vol. C-37, no. 8, Aug. 1988, pp. 980-990.

18. D. J. Magenheimer, L. Peters, K. W. Pettis and D. Zuras, "Integer Multiplication and Division on the HP Precision Architecture", *Proc. Symp. on Arch. Supp. for Prog. Lang. and Systems II*, Oct. 1987, pp. 90-99.

19. H. Melenk, H. M. Möller and W. Neun,"On Gröbner Bases Computation on a Supercomputer Using REDUCE", Preprint SC 88-2, Jan. 1988, FB Mathematik und Informatik der Fernuniversität Hagen, 18 p.

20. J. E. Robertson, "Normalization and Quotient Digit Selection in a Variable Precision Arithmetic Unit", Report UIUCDCS-R-86-1229, Univ. of Ill. at Urbana-Champaign, 1986, 14 p.

21. J. E. Robertson, "A Theory of Decomposition of Structures for Binary Addition and Subtraction", Report UIUCDCS-R-81-1004, Univ. of Ill. at Urbana-Champaign, Jan. 1983.

22. J. E. Robertson, "A Systematic Approach to the Design of Structures for Arithmetic", *Proc. of the* 5^{th} *Symp. on Computer Arithmetic*, May 1981, pp. 35-41.

23. J. E. Robertson, "Design of the Combinational Logic for a Radix-16 Digit-Slice for a Variable Precision Processor Module", *Proc. of ICCD 1983*, Nov. 1983, pp. 696-699.

24. J. E. Robertson, "A New Class of Digital Division Methods", *IRE Trans. on Electronic Computers*, Vol. EC-7, Sep. 1958, pp. 218-222.

25. G. S. Taylor, "Radix 16 SRT Dividers with Overlapped Quotient Selection Stages", *Proc.* 7^{th} *Symp. on Computer Arithmetic*, June 1985, pp. 64-71.

26. S. W. Thomas, *Modeling Volumes Bounded by B-Spline Surfaces*, Ph.D. Diss., Univ. of Utah, Dept. of Computer Science, 1984.

27. C. Tung, "A Division Algorithm for Signed-Digit Arithmetic", *IEEE Trans. on Computers*, Vol. C-17, No. 9, Sep. 1968, pp. 887-889.

28. C. Tung, "Signed-Digit Division Using Combinational Arithmetic Nets", *IEEE Trans. on Computers*, Vol. C-19, No. 8, Aug. 1970, pp. 746-748.

29. J. L. White, "Reconfigurable, Retargetable Bignums: A Case Study in Efficient, Portable Lisp System Building", *Proc. ACM Conf. on Lisp and Functional Prog.*, 1986, pp. 174-191.

Table 1: **Quotient Digit Selection by Algorithm 16**

Name	r_{norm}		r_{est}		$r = 16$		D_{model}		$D_{compare}$		Pre Processing
	d	p_j	d	p_j	ϵ	η	d	p_j	d	p_j	
5.50	16	16	16	4	4	3	4.00	1.50	4.00	1.50	none
5.25	16	16	2	4	15	3	3.75	1.50	3.75	1.50	none
5.00	16	16	16	16	3	2	3.00	2.00	3.00	2.00	none
4.75	16	16	16	2	3	7	3.00	1.75	3.00	1.75	none
4.50	16	16	4	16	5	2	2.50	2.00	2.50	2.00	none
4.00_a	4	16	16	16	2	2	2.00	2.00	2.00	2.00	$4R$
4.00_b	4	16	16	16	3	1	3.00	1.00	3.00	1.00	$4R$
3.75_a	2	16	2	16	7	2	1.75	2.00	1.75	2.00	$4R, 2R$
3.75_b	4	16	2	16	11	1	2.75	1.00	2.75	1.00	$4R, 2R$
3.75_c	2	16	16	2	3	3	3.00	0.75	3.00	0.75	$4R, 2R$
3.50_a	4	16	16	4	2	3	2.00	1.50	2.00	1.50	$4R$
3.50_b	2	16	2	2	11	3	2.75	0.75	2.75	0.75	$4R, 2R$
3.25_a	2	16	2	4	7	3	1.75	1.50	1.75	1.50	$4R, 2R$
3.25_b	2	16	2	2	7	5	1.75	1.25	1.75	1.25	$4R, 2R$
3.00	2	16	16	16	2	1	2.00	1.00	2.00	1.00	$4R, 2R$

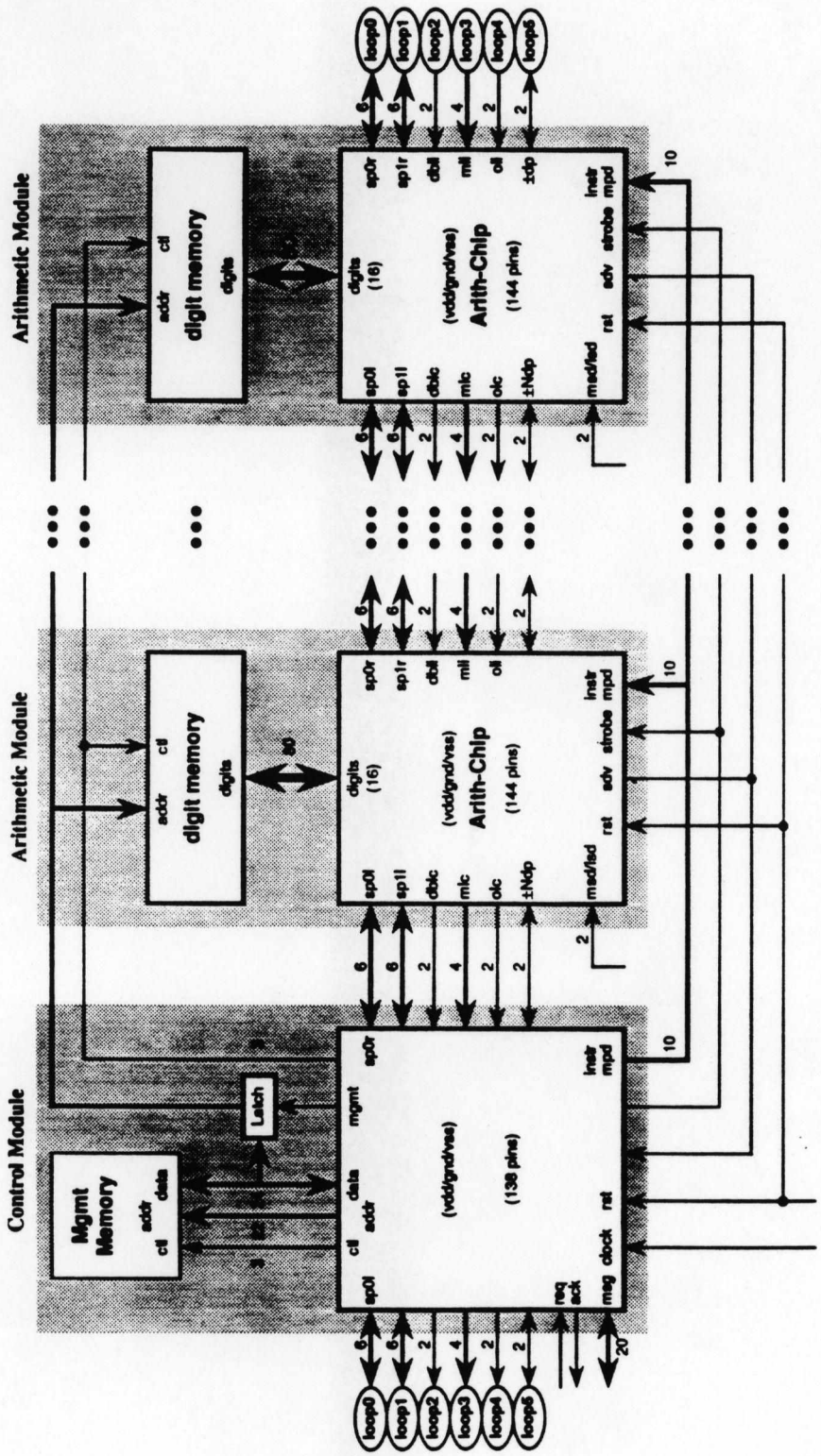

Figure 1: An Architecture for Variable Precision Arithmetic Hardware

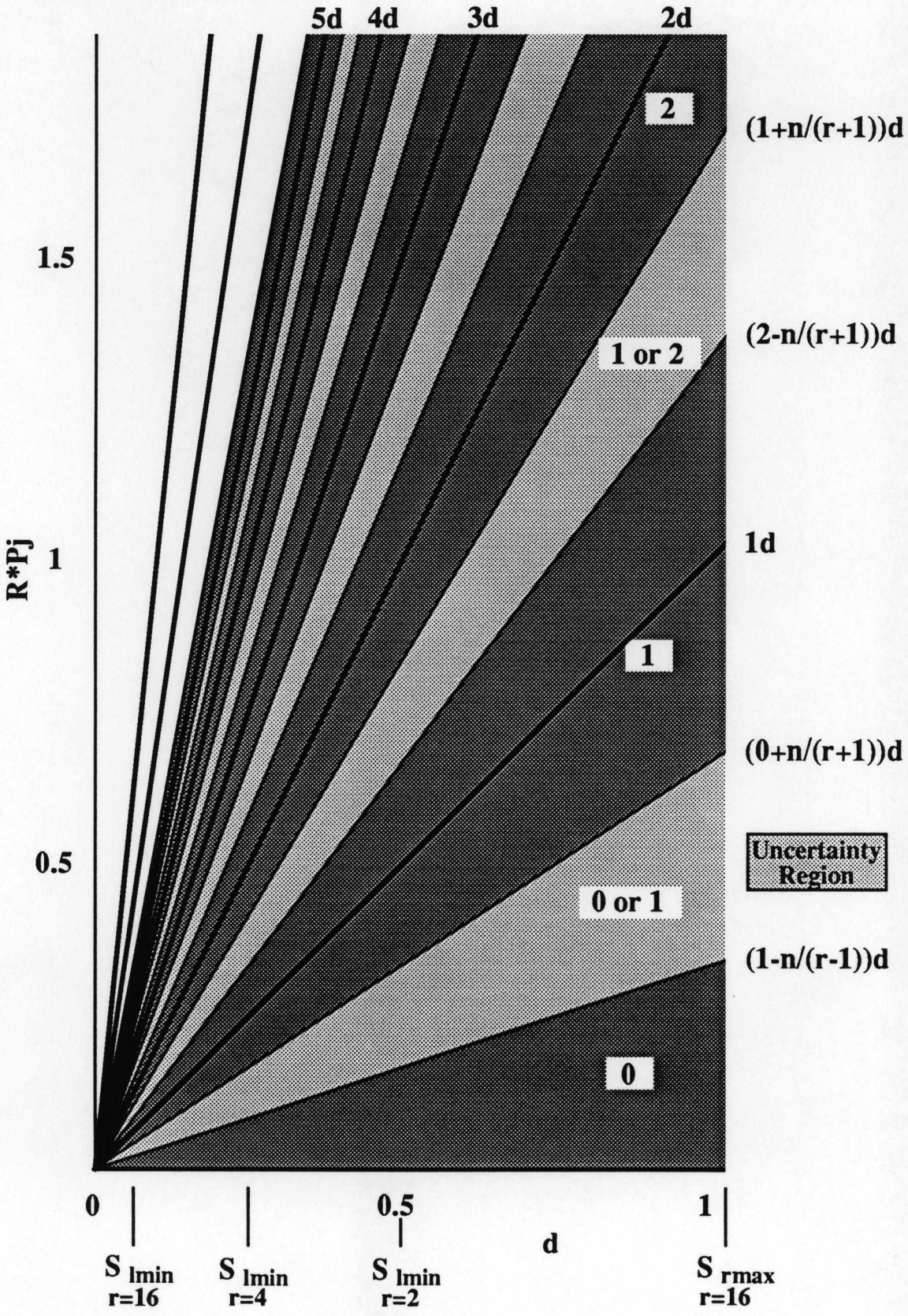

FIGURE 2: P-D PLOT FOR QUOTIENT DIGIT SELECTION

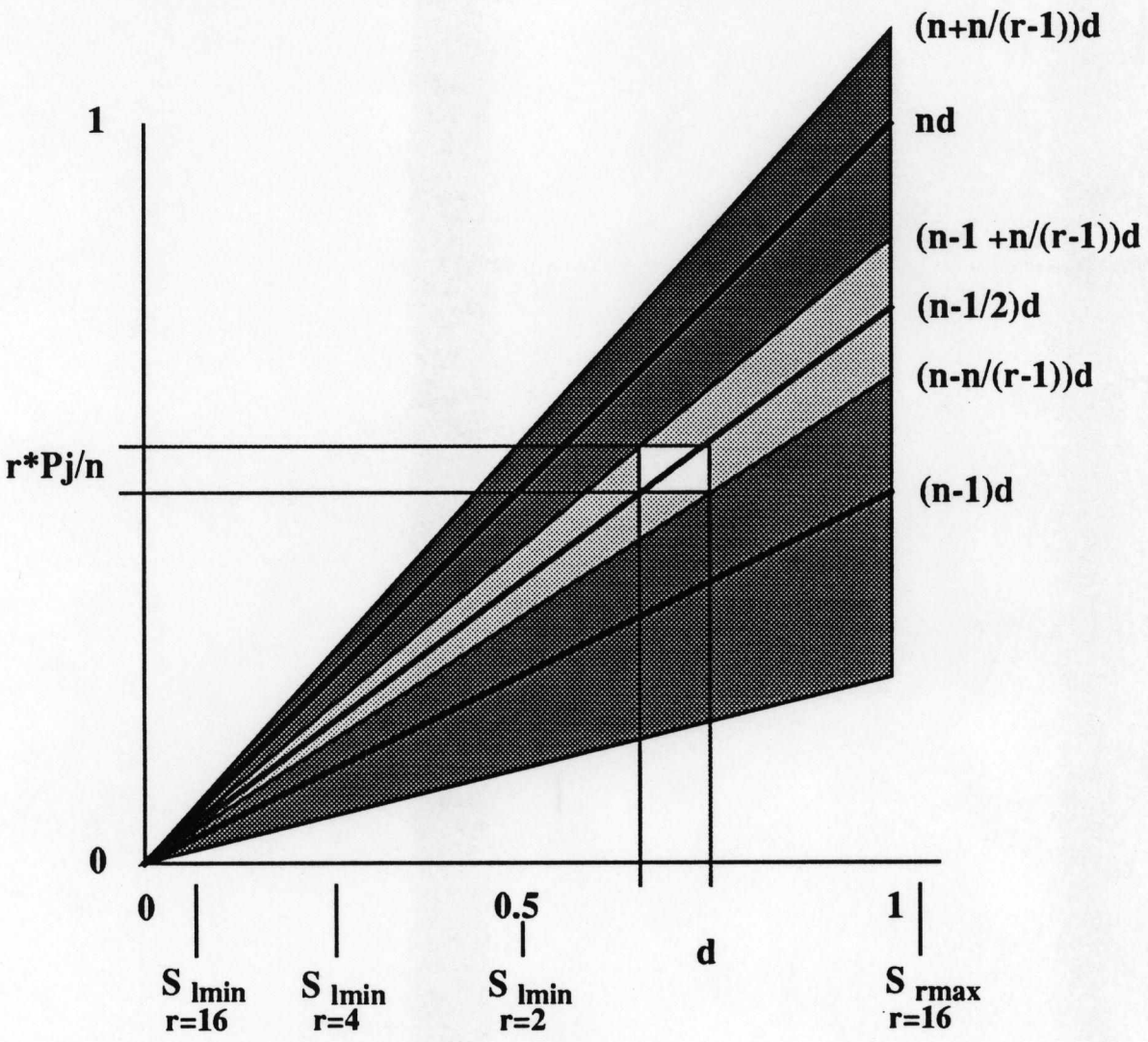

Figure 3: PD Plot for Selecting Between n and $n-1$

Table 2: **Quotient Digit Selection by Algorithm 16.4**

Name	r_{norm}		r_{est}		$r = 16$		$r = 4$		D_{model}		$D_{compare}$		**Pre Processing**
	d	p_j	d	p_j	ϵ	η	ϵ	η	d	p_j	d	p_j	
5.50_a	16	16	16	4	4	3	4	3	4.00	1.50	4.00	1.50	none
5.50_b	16	16	4	4	8	3	7	3	4.00	1.50	3.50	1.50	none
5.25	16	16	2	4	15	3	13	3	3.75	1.50	3.25	1.50	none
5.00	16	16	16	16	3	2	2	2	3.00	2.00	2.00	2.00	none
4.75	16	16	16	2	3	7	2	7	3.00	1.75	2.00	1.75	none
4.50_a	16	16	4	16	5	2	4	2	2.50	2.00	2.00	2.00	none
4.50_b	16	16	4	2	5	8	4	7	2.50	2.00	2.00	1.75	none
4.00_a	4	16	16	16	2	2	2	2	2.00	2.00	2.00	2.00	$4R$
4.00_b	4	16	4	16	4	2	3	2	2.00	2.00	1.50	2.00	$4R$
4.00_c	4	16	2	16	8	2	5	2	2.00	2.00	1.25	2.00	$4R$
4.00_d	4	16	16	16	3	1	3	1	3.00	1.00	3.00	1.00	$4R$
4.00_e	4	16	4	16	6	1	5	1	3.00	1.00	2.50	1.00	$4R$
3.75_a	2	16	2	16	7	2	4	2	1.75	2.00	1.00	2.00	$4R, 2R$
3.75_b	2	16	2	16	7	2	5	1	1.75	2.00	1.25	1.00	$4R, 2R$
3.75_c	4	16	2	16	11	1	9	1	2.75	1.00	2.25	1.00	$4R$
3.75_d	2	16	16	2	3	3	3	3	3.00	0.75	3.00	0.75	$4R, 2R$
3.75_e	2	16	4	2	6	3	5	3	3.00	0.75	2.50	0.75	$4R, 2R$
3.50	2	16	2	2	11	3	9	3	2.75	0.75	2.25	0.75	$4R, 2R$
3.50_a	4	16	16	4	2	3	2	3	2.00	1.50	2.00	1.50	$4R$
3.50_b	4	16	16	2	2	6	2	5	2.00	1.50	2.00	1.25	$4R$
3.50_c	4	16	4	4	4	3	3	3	2.00	1.50	1.50	1.50	$4R$
3.50_d	4	16	2	4	8	3	5	3	2.00	1.50	1.25	1.50	$4R$
3.50_e	4	16	4	2	4	6	3	5	2.00	1.50	1.50	1.25	$4R$
3.25_a	2	16	2	4	7	3	4	3	1.75	1.50	1.00	1.50	$4R, 2R$
3.25_b	2	16	2	4	7	3	5	2	1.75	1.50	1.25	1.00	$4R, 2R$
3.00_a	2	16	16	16	2	1	2	1	2.00	1.00	2.00	1.00	$4R, 2R$
3.00_b	2	16	4	16	4	1	3	1	2.00	1.00	1.50	1.00	$4R, 2R$
3.00_c	2	16	2	16	8	1	5	1	2.00	1.00	1.25	1.00	$4R, 2R$