High-Radix Multiplier-Dividers: Theory, Design, and Hardware

Alaaeldin Amin, Member, IEEE, and Waleed Shinwari, Member, IEEE.

Abstract—This paper describes the theory and design of digital high-radix multiplier-dividers\(^1\). The theory of high-radix division is extended to high-radix multiplier-dividers that can perform fused multiplication and division operations using a single recurrence relation. With the fused implementation of multiplication and division, the two operations can be executed using a single instruction, implying only a single rounding operation. The recurrence relation is described, the quotient digit selection function derived, and important design parameters together with their optimal values and relations are defined. Efficient design procedure and implementation hardware are described and important system parameter values for various radix systems computed. Compared to pure dividers, the multiplier-divider requires a slightly more complex data path and quotient digit selection function.

Index Terms—Computer arithmetic, division, quotient digit selection, SRT, multiplier-divider.

I. INTRODUCTION

To achieve high performance in application-specific processors, the current System-on-a-Chip (SoC) technology generally augments the programmable general-purpose cores with application-specific functional units (AFUs) or hardware accelerators. This is a cost-effective way to simultaneously speed up execution and reduce energy consumption through delegating time consuming tasks of applications to dedicated hardware accelerators, leaving less critical tasks to traditional software execution. A recent trend in SoC technology is to allow the extension of existing instruction sets by special instructions for performance-critical operations [1]. This is achieved by adding application-specific instruction set extensions (ISEs) to the processor instruction set for executing the critical portions of the application on the AFUs [2]. This has lead microprocessor intellectual property (IP) vendors to license configurable and extensible processor cores to their customers [3]. For example, an optimized multiply-and-accumulate (MAC) unit that can compute \(a \times b + c + d\) using only one 4-operand instruction has been recently reported [4]. Another [5], reported a \(GF(2^m)\) 3-operand computation of \(ab/c\) performing both multiplication and inversion.

This paper describes new formulae, algorithm and hardware which can efficiently compute \(A \times B / D\) by performing simultaneous multiplication and division operations thus requiring only a single rounding operation. Such unit, which will be referred to as multiplier-divider, requires a 3-operand instruction and can be implemented as part of a special Floating-point unit (FPU) or as a stand-alone AFU for compute-intensive applications which utilize this operation. The multiplier-divider can perform either a single multiplication operation, a single division operation or a simultaneous combined multiplication and division operations. All operations have the same execution time with one digit of the result produced each cycle starting with the most significant digit. Zurawski and Gosling [6] reported a more restricted approach to build a radix-4 unit for multiply-divide and square-root. Ercegovac and Lang [7] have also reported a module that can perform radix-2 multiplication, division, and square root. Compared to the multiplier-divider reported here, this module works only for radix 2 and can only perform one of the three operations but none of their combinations. Likewise, McIlhenny and Ercegovac [8] have proposed a 3-operand module that can perform two simultaneous multiplications \((A \times B \times C)\). Further, Antelo, et. al. [9], reported a very high radix processor that computes combined division and square root operations \((\sqrt{X/d})\).

There exists quite extensive literature that describes the theory and design of high-speed multiplication and division algorithms [10], [11]. The theory of high-radix multiplier-dividers may be considered as an extension of the high-radix digit-recurrence division algorithm. Based on the different hardware operations used in their implementations, e.g. multiplication, subtraction, and table look-up, division algorithms are divided into five classes [12]: digit recurrence, functional iteration, very high radix, table lookup, and variable latency. Digit recurrence is the oldest class of high speed division algorithms and, as a result, a significant number of publications can be found in the literature proposing digit recurrence algorithms, implementations, and techniques. The most common implementation of digit recurrence division in modern processors has been the SRT method [10].

Digit recurrence division algorithms use iterative methods to calculate quotients one digit per iteration. One quotient digit \((m\text{-bits})\) is retired each iteration using a quotient-digit selection function [13], [14], [15]. Typically, for a system with radix \(r\), the quotient digits are selected from a redundant signed digit set \(D_{\alpha} = \{-\alpha, -\alpha + 1, \ldots, -1, 0, 1, \ldots, \beta - 1, \beta\}\) whose size \((|\alpha + \beta + 1|)\) is greater than \(r\) with both negative and positive digits. It is fairly common to choose a symmetric digit set where \(\beta = \alpha\) in which case the size of the digit set \((2\alpha + 1) > r\) implying that \(\alpha \geq \lceil \frac{r}{2} \rceil\).

The degree of redundancy is measured by the redundancy factor \(h\), where \(h = \frac{\alpha}{\lceil \frac{r}{2} \rceil}\). Redundancy is maximal when \(\alpha = r - 1\) in which case \(h = 1\), while it is minimal when \(\alpha = r/2\) (i.e., \(\frac{1}{2} < h \leq 1\)).

\(^{1}\)Patent Pending

Manuscript received July 22, 2007; revised March 1, 2008.

Alaaeldin Amin is with the Computer Engineering Department of King Fahd University, Dharan, Saudi Arabia.

Waleed Shinwari is with the Department of Electrical and Computer Engineering, McMaster University.

IEEE TRANSACTIONS ON COMPUTERS, MANUSCRIPT ID, VOL. 1, NO. 8, JULY 2008 1
The fundamental choices in the design of digit recurrence dividers are the radix, the allowed quotient digits, and the representation method of the partial remainder (residue). The radix determines the number of quotient bits retired per iteration, which determines the required number of iterations. Larger radices can reduce the latency, but increase the time for each iteration. Judicious choice of the allowed quotient digits can reduce the time for each iteration, but with a corresponding increase in complexity and hardware. Similarly, different representations of the partial remainder (residue) can reduce iteration time, with a corresponding increase in complexity. This paper describes a high-radix digit recurrence algorithm and formulae which can efficiently compute \( S = A/B \) by performing simultaneous accumulation of partial products and subtraction of a proper divisor multiple in each iteration.

The structure of the paper is as follows. In section 2, we present the digit-recurrence relation of the multiplier-divider for both the integer and fractional formats, define constraints imposed on input operands, determine the size of the multiplier-divider processor, and the required number of iterations. In section 3, the recurrence relation is derived, upper and lower bounds of the residue computed, and the quotient digit selection function defined. In section 4, the optimal multiplier-divider design parameters are derived. Section 5 outlines the overall design procedure while section 6 provides hardware implementation of high-radix multiplier-dividers. In section 7 we discuss results with conclusions provided in section 8.

II. HIGH RADIX MULTIPLIER-DIVIDERS

This work presents the design of a digital multiplier-divider unit which can efficiently compute \( S = (A \times B + D) \), where the multiplicand \( A \), the multiplier \( B \), and the divisor \( D \) are \( \ell \)-bit unsigned numbers. Computing \( S \) yields an \( \ell \)-bit quotient \( Q \) and a remainder \( R \) such that:

\[
AB = QD + R, \quad \text{and} \quad |R| < |D| \tag{1}
\]

Conventionally, \( S \) is computed using two independent operations; a multiplication operation and a division operation. Recurrence relations for these two operations have been proposed and are in common use by digital processors. In this work, we propose a general radix single recurrence relation to perform the multiplication and division operations in a fused manner which allows efficient computation of \( S \). The recurrence relation is used as a basis to extend the theory of high-radix division to multiplication-division. The quotient digit selection function for this case is presented, a design procedure is outlined, major design parameters for these systems are defined and their optimal values and relations are derived. Restricted versions and/or utilization of equivalent recurrence relations, however, have reportedly been used in different contexts. For example, to compute the modular product, Takagi [16] has used a radix-4 equivalent version of the recurrence relation while a software implementation by Tang [17] computes the modular multiplication of long multi-precision integers using a similar recurrence relation in its inner loop.

A. Multiplier-Divider Recurrence Relation

To speed-up the computation of \( S = (A \times B \div D) \), the proposed recurrence relation uses a high radix \( r = 2^m \) where \( m \geq 1 \). Initially, consider the operands \( A, B, \) and \( D \) to be \( n \)-digit integers, i.e. \( A = (a_{n-1}, \ldots, a_1, a_0) \), \( B = (b_{n-1}, \ldots, b_1, b_0) \), and \( D = (d_{n-1}, \ldots, d_1, d_0) \), where \( n = \lceil \ell/m \rceil \) and \( a_i, b_i, \) and \( d_i \) are radix \( r \) digits. The proposed multiply-divide recurrence relation is given by:

\[
R_j = rR_{j-1} - q_{n-j}Dr^n + b_{n-j-1}Ar^{n-1} \tag{3}
\]

where,

\[
q_i \text{ is the } i^{th} \text{ quotient digit}
\]

\[
b_i \text{ is the } i^{th} \text{ digit of } B
\]

\[
b_{-1} = 0
\]

\[
R_j \text{ is the } j^{th} \text{ running partial remainder}
\]

\[
R_0 = b_{n-1}Ar^{n-1}
\]

The final results are the quotient \( Q \) and the remainder \( R \) where:

\[
Q = q_{n-1}q_{n-2} \ldots q_2q_1q_0 = \sum_{j=0}^{n-1} q_j r^j \tag{4}
\]

\[
R = R_n \tag{5}
\]

if \( R_n < 0 \) then the following correction step should be performed:

- \( Q = Q - ulp \), where \( ulp \) is a unit in least position, and
- \( R_n = R_n + D \) with \( R = \frac{R_n}{r^m} \)

1) Proof of the recurrence relation: Executing the \( n \) iterations of the proposed recurrence relation yields the desired \( Q \) and \( R \) values as defined by Equations 1 and 2.

\[
R_j = rR_{j-1} - q_{n-j}Dr^n + b_{n-j-1}Ar^{n-1}
\]

\[
R_0 = b_{n-1}Ar^{n-1}
\]

\[
R_1 = r^n b_{n-1} A - q_{n-1} Dr^n + r^{n-1} b_{n-2} A
\]

\[
R_2 = A(r^{n+1} b_{n-1} + r^n b_{n-2} - Dr^{n+1} q_{n-1} - q_{n-2} Dr^n + b_{n-3} Ar^{n-1})
\]

\[
= A(r^{n+1} b_{n-1} + r^n b_{n-2} + r^{n-1} b_{n-3})
\]

\[
- D(r^{n+1} q_{n-1} + q_{n-2} q_{n-2})
\]

\[
R_3 = A(r^{n+2} b_{n-1} + r^{n+1} b_{n-2} + r^n b_{n-3} + r^{n-1} b_{n-4})
\]

\[
- D(r^{n+2} q_{n-1} + r^{n+1} q_{n-2} + r^n q_{n-3})
\]

\[
\vdots
\]

\[
R_n = A \sum_{i=1}^{n+1} r^{2n-i} b_{n-i} - D \sum_{j=1}^{n} r^{2n-j} q_{n-j}
\]

with \( b_{-1} = 0, R_n = r^n (AB - DQ) \).

Thus, \( R = \frac{R_n}{r^m} = AB - DQ, \) and \( AB = DQ + R \)

If the digits of \( Q \) are chosen such that the magnitude of the partial residue \( R_j \) is maintained less than the magnitude of \( D \), then \( Q \) is effectively the required quotient of the operation \( AB/D \). Since \( AB = DQ + R \) and \( |R| < |D| \), then \( R \) is indeed the final remainder.
2) Fractional form of the recurrence relation: The recurrence relation of Equation 3 can be rewritten assuming $A$, $B$, $D$, and $Q$ to be normalized fractions of the form $D = 0.d_1d_2 \cdots d_n$, and $Q = 0.q_1q_2 \cdots q_n$, with minimum value ($D_{\min}$ or $Q_{\min}$) of 0.5. The integer operations can be readily mapped to the fractional form. The fractional formulas are more convenient in mathematical representation, however, since they are readily adaptable to floating point representations. The fractional form is obtained from the integer form as follows:

$$A = A_{\text{integer}} r^{-n}, \quad B = B_{\text{integer}} r^{-n}, \quad D = D_{\text{integer}} r^{-n}, \quad R = R_{\text{integer}} r^{-2n} \quad (6)$$

Following is the modified fractional multiply-divide recurrence relation:

$$R_j = rR_{j-1} - q_{j-1}D + b_{j-1}A r^{-1} \quad (7)$$

where,

$$R_0 = b_{-1}A r^{-1}$$
$$b_{-j} = 0 \quad \text{for} \quad i > n$$
$$R_n = r^n R$$

The final quotient $Q$ and remainder $R$ are given by:

$$Q = 0.q_1q_2 \cdots q_n = \sum_{j=1}^{n} q_{j} r^{-j} \quad (8)$$

$$R = \frac{R_n}{r^n} \quad (9)$$

The above multiplier-divider recurrence relation may alternately be used with $R_0 = 0$ in which case, an extra iteration step is needed. Thus;

$$R_j = rR_{j-1} - q_{j-1}D + b_{j-1}A r^{-1} \quad (10)$$

where,

$$R_0 = 0$$
$$q_0 = 0$$
$$b_{-j} = 0 \quad \text{for} \quad j > n$$
$$R_{n+1} = r^n R$$

In this case, the quotient $Q$ and the remainder $R$ are given by:

$$Q = 0.q_1q_2 \cdots q_n = \sum_{j=1}^{n} q_{j} r^{-j} \quad (11)$$

$$R = \frac{R_{n+1}}{r^n} \quad (12)$$

The above formulae will also support operands that follow the IEEE 754 mantissa/significand format, in which case the operand minimum value (e.g. $D_{\min}$) is 1.0.

Example $r = 2$, $n = 8$, $A = 0.10101011_{\text{binary}}$, $B = 10101101_{\text{binary}}$, and $D = 0.10111101_{\text{binary}}$.

The solution steps are shown in Fig. 1. Note that the integer operands were mapped to their fractional form. Throughout this work, we assume fractional operands, with the understanding that integer operations can be directly mapped to the fractional form. The width of the operations is 11 bits; 8-bits is the original operand size, 1-bit for the sign, 1-bit to accommodate left-shift operation and 1-bit for the right shift of the term $b_{-j}A r^{-1}$. The quotient digits (bits in this case) were chosen such that the remainder will always lie in the range $(-D,+D)$ and not just $[0,+D)$. Allowing negative remainders can cause the selected quotient digit to be negative. A final remark to be made about the example of Fig. 1 is that since all operands ($A$, $B$ and $D$) are positive, the final remainder should be positive as well. Since we are allowing negative remainders (with magnitude less than $D$), we may require a final correction step if the final remainder turns out to be negative. The correction step would add the divisor $D$ to the partial remainder $R_n$ with a corresponding correction to the quotient value by subtracting a $ulp$.

As in the case of division, $AB < D$ is the condition to guarantee that no overflow may occur ($AB = DQ + R$). The following analysis assumes the use of Equations (7)-(9).
with the understanding that similar analysis holds true if the alternative formulae of Equations (10)-(12) are used instead.

B. Pre-Processing andOperand Value Constraint

Referring to the recurrence relation of equation (7), each iteration consists of the following steps:

1) Determination of the next quotient digit \( q_{j} \) using some quotient digit selection function (\( q_{j} = SEL(rR_{j-1}, D) \)). The selection function may typically be implemented as a look-up table.

2) Perform the triple addition of \( rR_{j-1}, (-q_{j}D) \) and \( b_{j-1}A_{j} \). The resulting partial residue \( R_{j} \) must guarantee that \( |R_{j}| < |D| \). The condition \(|R_j|<|D|\) depends on the proper choice of the quotient digit \( q_{j} \).

When performing a multiply-divide operation, we are adding a multiple of the input operand \( A \) in each step. The resulting residue thus obtained \( (R_j)\) cannot be known as predictably as the case of high radix division. However, we may still restrict the value range of \( (R_j) \) by placing some restrictions on the value of \( A \). One possible restriction is to impose the constraint \(|A|<|D|\). Thus, we assume that \( A = \omega D \) where, \( \omega < 1 \).

Assuming \( Max(h_i) = B^+ \), Equation (7) yields:

\[
R_{j_{\text{max}}}(q_j) = rR_{j-1} - q_{j}D + \frac{B^+}{r} \omega D, \\
R_{j_{\text{min}}}(q_j) = R_{j_{\text{division}}} + \omega TD
\]

where, \( T = \left( \frac{B^+}{r} \right) < 1, \omega < 1 \), and \( R_{j_{\text{division}}} \) is the residue of regular high radix division. This shows that the deviation in the remainder curve of the Robertson diagram [10, 11] from the case of pure division can be as high as \( \omega TD \).

The upper bound of \( \omega = A_{j} / D \), which equals \( A_{\text{max}} / D_{\text{min}} \) must be less than 1. To guarantee satisfaction of this constraint, a pre-processing step shifting \( A \) by \( Z \)-bits to the right is performed. Thus, if the input operand is \( A' \), processing is actually performed on \( A = A'/2^Z \) rather than the input operand \( A' \) itself. Accordingly, our proposed methodology computes \( S = AB/D \) and a post-processing step computes \( S' = A'B/D = S + A'/A = S2^Z \). For the adopted operand fractional formats \( D_{\text{min}} = 0.5 \) or \( D_{\text{min}} = 1.0 \), an \( x \)-bit normalized significand has a ratio of \( A_{\text{max}} / D_{\text{min}} \) which equals \( [2 - 2^{-x+1}] \) and accordingly, the upper bound of \( \omega \) is given by:

\[
\omega \leq \frac{A_{\text{max}} / D_{\text{min}}}{2^z} = \frac{2(1 - 2^{-nm})}{2^Z}
\]

Since, for typical values of \( n \) and \( m \), \( 2^{-nm} \ll 1 \) we define the parameter \( \omega_{\text{max}} = 2^{1-Z} \) as the upper bound for \( \omega \) such that \( \omega < \omega_{\text{max}} \) where:

\[
\omega_{\text{max}} = 2^{1-Z} \leq 1
\]

C. Size of the Multiplier-Divider Processor

The multiply-divide recurrence relation can be implemented in hardware using shift and add operations. Although the problem size is \( \ell \) bits (\( \ell = nm \)), the minimum possible size in radix \( r \) implementations is \( [(n+2)m + Z + 1] \) bits where \( r = 2^m \). Referring to the high-radix multiplier-divider recurrence relation (Equation (7)), a total of \( n \)-digits are needed to accommodate the input operand size, two more digits are needed to account for the left and right shifts (\( rR_{j-1} \) and \( b_{j-1}A_{j}^{-1} \)) respectively, \( Z \) extra bits are needed since computations are performed on the constrained parameter \( A \) \( (A = A'/2^Z) \) rather than the multiplicand \( A' \), and a sign bit is required since the partial residue \( R_{j} \) may be either positive or negative.

D. Post-Processing and the Number of Iterations

Due to the pre-processing step where the input multiplicand \( A' \) is shifted right by \( Z \) bit positions, i.e. \( A = A'/2^Z \), a post-processing step where the result \( S \) is shifted left by \( Z \) bit positions is needed, i.e. \( S' = S2^Z \). In other words, since the resulting quotient and remainder values \( (Q \) and \( R) \) satisfy the relation \( AB = QD + R \), i.e. \( (A' + 2^Z)B = QD + R \), the true quotient \( Q' \) and remainder \( R' \) which satisfy \( A'B = Q'D + R' \) are computed in a post-processing step as: \( Q' = Q2^Z \), and \( R' = R2^Z \). Thus, it is expected that the first \( Z \) bits of the resulting quotient \( (Q) \) to be zeroes. Accordingly, if \( n \)-significant digits of \( Q' \) are needed, the number of required iterations of the recurrence relation (Equation (7)) must be raised to \( n + [\frac{Z}{m}] \). Thus, Zero output digits are produced for the first \( [\frac{Z}{m}] \) clock cycles. This initial delay is quite similar to the \( \delta \) on-line delay of on-line arithmetic [10, 18]. As such, this algorithm carries some resemblance to an on-line division algorithm where the dividend is received in an on-line digit-serial manner while the divisor is available in parallel.

III. QUOTIENT DIGIT SELECTION

To define the quotient digit selection function, we need to determine the upper and lower bounds of the shifted partial residue \( P = rR_{j-1} \) for which a given quotient digit value may be selected such that \( |R_{j}| < |D| \). The assumptions under which these bounds will be derived are:

1) \( R_{j} \) is kept bounded, i.e. \( |R_{j}| < |D| \), by defining the negative and positive range limiting factors \( h^- \) and \( h^+ \) such that: \( -h^- D \leq R_{j} \leq h^+ D \) where \( h^- , h^+ < 1 \).

2) The radix \( r \) is a power of 2, i.e. \( r = 2^m \).

3) The multiplier-divider operand \( A \) is obtained by shifting the input operand \( A' \) by \( Z \)-bits to the right, i.e. \( A = A'/2^Z \).

4) The magnitude of the multiplicand \( A \) is smaller than the magnitude of the divisor \( D \) \( (A = \omega D, \) where \( \omega < \omega_{\text{max}} = 2^{1-Z} \leq 1) \).

5) The multiplier \( B \) is represented by radix \( r \) digits \( b_i \) either in a \( [0, r-1] \) non-redundant digit set, or in a \( [-B^-, +B^+] \) redundant signed digit set. In the following analysis, we will use a generalized signed-digit set \([19]\), where \( b_i \) falls in the range \([ -B^-, +B^+ ] \).

6) For the quotient digits, we use a redundant balanced signed-digit set \( \mathcal{P}_q = \{-\alpha, -) (\alpha - 1), \ldots, -1, 0, +1, \ldots, +\alpha\}, \) with \( r/2 \leq \alpha \leq (r - 1) \).
7) For the \( j^{th} \) iteration, an acceptable choice of \( q_{-j} \) is one which satisfies the condition \(-h^- D \leq R_j \leq +h^+ D\).

**A. Range Limiting Factors**

For a feasible implementation of the high-radix multiplier-divider recurrence relation (Equation (7)), when the shifted partial residue \( rR_{j-1} \) equals its maximum value \((r h^+ D)\) and \( b_{-j-1} \) is also maximum \((= +B^+)\), a value of \( q_{-j} = \alpha \) should guarantee that \( R_j < h^+ D \), thus:

\[
(r h^+ D - \alpha D + \frac{B^+}{r} \omega D) \leq h^+ D,
\]

thus,

\[
h^+ \leq \frac{\alpha}{(r-1)} - \frac{B^+}{(r-1)} \frac{\omega}{r}.
\]

Replacing \( \omega \) by \( \omega_{\text{max}} \) in the above equation, we obtain a lower bound expression for \( h^+ \) which guarantees that \( R_j < h^+ D \). Thus, \( h^+ \) is taken as:

\[
h^+ = \frac{\alpha}{(r-1)} - \frac{B^+}{(r-1)} \frac{\omega_{\text{max}}}{r}
\]

(14)

\[
h^+ = h - \frac{\omega_{\text{max}}}{r} h_B^+
\]

(15)

where,

\[
h_B^+ = \frac{B^+}{(r-1)}
\]

(16)

Likewise, when the shifted partial residue \( rR_{j-1} \) equals its minimum value \((-r h^- D)\) and \( b_{-j-1} \) is also minimum \((= -B^-)\), a value of \( q_{-j} = -\alpha \) should guarantee that \( R_j \geq -h^- D \), thus:

\[
(-r h^- D - \alpha D - \frac{B^-}{r} \omega D) \geq -h^- D, \text{ or}
\]

\[
h^- \leq \frac{\alpha}{(r-1)} - \frac{B^-}{(r-1)} \frac{\omega}{r}
\]

Replacing \( \omega \) by \( \omega_{\text{max}} \) in the above equation, we obtain a lower bound expression for \( h^- \) which guarantees that \((-h^- D \leq R_j)\). Thus, \( h^- \) is taken as:

\[
h^- = \frac{\alpha}{(r-1)} - \frac{B^-}{(r-1)} \frac{\omega_{\text{max}}}{r}
\]

(17)

\[
h^- = h - \frac{\omega_{\text{max}}}{r} h_B^-
\]

(18)

where,

\[
h_B^- = \frac{B^-}{(r-1)}
\]

(19)

**B. Recurrence Revisited**

Assuming fractional multiplication-division, the error in the resulting quotient must be bounded by:

\[
|\epsilon| = \left| \frac{AB}{D} - Q \right| \leq r^{-n}
\]

(20)

For a meaningful multiplication-division operation, the partial products have to be accumulated ahead of the division process by at least one iteration. Thus, the quotient error at the \( j^{th} \) iteration is defined by:

\[
\epsilon_j = \frac{AB[j + 1]}{D} - Q[j]
\]

(21)

where, \( B[j + 1] = 0.b_{-1}b_{-2} \cdots b_{-j-1} \) and \( Q[j] = 0.q_{-1}q_{-2} \cdots q_{-j} \). It can be easily shown that:

\[
\epsilon_{j+1} = \epsilon_j + r^{-(j+1)} \left\{ \frac{A b_{-(j+2)}}{D} - q_{-(j+1)} \right\}
\]

(22)

For convergence, the quotient digit \( q_{-(j+1)} \) should be chosen so that the error \( \epsilon_{j+1} \) even under worst case values of \( b_{-(j+2)} \). Assuming a multiplier digit set in the range \([-B^-, +B^+]\), convergence worst case values of \( b_{-(j+2)} \) are \( B^+ \) if \( \epsilon_j > 0 \) and \(-B^- \) otherwise. In either case, the worst case value of \( A/D \) is \( \omega_{\text{max}} \). Thus, for convergence, \( \epsilon_j \) can be expressed in terms of the error after the \( n^{th} \) iteration \( \epsilon_n \) under worst case as:

\[
\epsilon_j \leq \epsilon_n - \sum_{i=j+1}^{n} r^{-i} \left( \omega_{\text{max}} \frac{B^+}{r} \right) + \max_{i=j+1}^{n} q_i r^{-i}
\]

(23)

Assuming a balanced quotient digit set in the range \([-\alpha, +\alpha]\), Equation (23) becomes:

\[
\epsilon_j \leq \epsilon_n + \sum_{i=j+1}^{n} r^{-i} \left( \alpha - \omega_{\text{max}} \frac{B^+}{r} \right)
\]

(24)

\[
\epsilon_j \leq \epsilon_n + \left( \alpha - \omega_{\text{max}} \frac{B^+}{r} \right) \left( r^{-j} - r^{-n} \right)
\]

(25)

\[
\epsilon_j \leq \epsilon_n + \left( h - \omega_{\text{max}} \frac{h_B^+}{r} \right) \left( r^{-j} - r^{-n} \right)
\]

(26)

\[
\epsilon_j \leq \epsilon_n + h^+ (r^{-j} - r^{-n})
\]

(27)

where \( h^+ \) and \( h^- \) are defined in Section III-A.

From Equation (20), since \( |\epsilon_n| \leq r^{-n} \), Equation (27) yields an upper bound solution of \( \epsilon_j \leq h^+ r^{-j} \) for all possible values of \( h \) and \( h_B^+ \). Likewise, it can be shown that the lower bound solution is \( \epsilon_j \geq h^- r^{-j} \) with \( h^- \) as defined in Section III-A. Thus, for iterations to converge to a solution, the iteration quotient error must be bounded by:

\[
h^- r^{-j} \leq \epsilon_j \leq h^+ r^{-j}
\]

(28)

Or,

\[
h^- r^{-j} \leq \frac{AB[j + 1]}{D} - Q[j] \leq h^+ r^{-j}
\]

(29)

We define the residual \( R_j \) such that its upper and lower bounds are independent of \( j \), as:

\[
R_j = r^j (AB[j + 1] - DQ[j])
\]

(30)

Equation (30) yields the recurrence relation:

\[
R_j = rR_{j-1} - q_{-j} D + b_{-j-1} Ar^{-1}
\]

(31)

such that:

\[
-h^- D \leq R_j \leq h^+ D
\]

(32)

The initial value \( R_0 \) is obtained from Equation (30) as:

\[
R_0 = AB[1] = Ab_{-1} r^{-1}
\]

(33)
As indicated by Equation (28), the final error may be negative which would require a correction step where \( D \) is added to \( R_n \) and \( Q \) is decremented by a \textit{ulp}.

C. P-D Diagrams

Here, we determine the selection interval defined by the upper (\( U_k \)) and lower (\( L_k \)) bounds of the shifted partial residue (\( P = rR_{j-1} \)) for which a given quotient digit value \( (q_{j-1} = k) \) may be selected such that the next partial residue \( (R_j) \) satisfies \(-h^-D \leq R_j \leq +h^+D\). From Equation (7), we can write \( P = rR_{j-1} = R_j + q_{j-1}D - b_{j-1}Ar^{-1} \). Thus;

\[
U_k = rR_{j-1} = R_{l_{max}} + kD - MAX\{b_{j-1}Ar^{-1}\}, \quad \text{i.e.}
\]

\[
U_k = h^+D + kD - \frac{B^+}{r} \omega_{max} D
\]

\[
= \left(k + h^+ - \frac{B^+}{r} \omega_{max}\right) D,
\]

thus,

\[
U_k(D) = (k + \rho^+) D, \quad (34)
\]

where,

\[
\rho^+ = h^+ - \frac{B^+}{r} \omega_{max}, \quad \text{or}
\]

\[
\rho^+ = h - \omega_{max} h_B
\]

Likewise;

\[
L_k(D) = (k - \rho^-) D, \quad (36)
\]

where,

\[
\rho^- = h - \omega_{max} h_B
\]

Equations (35) and (37), clearly show that \( \rho^+ = \rho^- \) if a balanced signed-digit set is used for \( B \). In this case, the P-D diagram will be symmetric with \( U_k(D) = -L_k(D) \) which would allow the utilization of only the 1st quadrant of the P-D diagram considerably reducing the storage requirements of the quotient digit selection function \([14], [15]\).

D. The Selection Function

Using all bits of \( P \) and \( D \) \((2\ell + 2m + Z + 1) \) bits as input to the quotient digit selection function \( SEL(P, D) \) requires huge ROM or PLA sizes. Accordingly, it is advantageous to minimize the number of input bits to the quotient digit selection function. Thus, we use truncated values of \( P \) and \( D \) as input to the quotient digit selection function. Let these truncated values be \( P_t \) and \( D_t \) and let the number of fractional bits of these parameters be \( n_P \) and \( n_D \) respectively. Thus, the maximum truncation error values for \( P \) and \( D \) are \( 2^{-n_P} \) and \( 2^{-n_D} \) respectively. Using a 2’s complement representation the introduced truncation errors are always positive, i.e. \( P \geq P_t \) and \( D \geq D_t \). Accordingly, any given value of \( P_t \) represents a range of \( P \) that is defined by: \( P_t \leq P < P_t + 2^{-n_P} \). Likewise, a given value of \( D_t \) represents a range of \( D \) defined by \( D_t \leq D < D_t + 2^{-n_D} \). As \( P_t \) and \( D_t \) are the only inputs to the selection function, with a total number of bits \((m + n_P + n_D + 2 + (\lfloor D_{min} \rfloor))\). For the same system, i.e. same \( r \) and \( \alpha \), the number of input bits to the selection function is the same independent of the representation format of the input fractions (i.e. whether \( D_{min} = 0.5 \) or \( D_{min} = 1.0 \)). Note that the values of \( n_P \) and \( n_D \) in case \( D_{min} = 1.0 \) are smaller by one bit compared to their values for the case where \( D_{min} = 0.5 \). To reduce the hardware complexity of the selection function, \((n_P + n_D)\), henceforth designated as \( n_{Tot} \), should be minimized. The selection function defines for each interval of the divisor \( D \left[D_t(D_{t+1})\right] \), where \( D_{t+1} = D_t + 2^{-n_D} \), comparison constants \( m_k(i) \) within the overlap regions for all values of \( k \) such that:

- The set of \textit{comparison constants} for each range of \( D \) is determined such that a given value of \( P_t \) is compared to these constants based on which a proper value of \( q_{j-1} \) is chosen, e.g. \( m_k(i) \leq P_t < m_{k+1}(i) \Rightarrow q_j = k \).
- If a symmetric multiplier digit set is used \((B^+ = B^-)\), symmetry of the P-D diagram can be utilized \([14]\) and only comparison constants \( m_k(i) \) for the 1st quadrant may be defined, i.e. for \( k = 0, 1, +2, \cdots +\alpha \).
- The comparison constants \( m_k(i) \) are chosen within the overlap regions where a choice of a \( q_{j-1} \) value of either \( k \) or \( k - 1 \) satisfies the constraint \(-h^-D \leq R_j \leq +h^+D \).
- Since any value within the overlap region may be used as a comparison constant, the choice is made such that \((n_P + n_D)\) is minimized.

For the \( i^{th} \) selection interval \([D_t(D_{t+1})]\), when determining the comparison constant \( m_k(i) \), two Conditions must be satisfied \([10], [11]\):

1. \textbf{Containment}: where \( L_k \leq m_k(i) \leq U_k \), and
2. \textbf{Continuity}: If \( P_t = (m_k(i) - 2^{-n_P}) \), then \( q_{j-1} \) must equal \( k - 1 \) which implies that \((m_k(i) - 2^{-n_P}) \leq U_{k-1} \). Written differently, we must have \( m_k(i) \leq U_{k-1} + 2^{-n_P} \) as well as satisfy the containment constraint. Accordingly, \( m_k(i) \) should satisfy \( L_k \leq m_k(i) \leq (U_{k-1} + 2^{-n_P}) \).

For the \( i^{th} \) selection interval \([D_t(D_{t+1})]\), the uncertainty in the value of \( P \) for a given value of \( P_t \), has an upper bound of \( \Delta_P = 2^{-n_P} \), i.e. \( P_t \leq P < P_t + \Delta_P \). Accordingly, the upper bound of the comparison constant \( m_k(i) \) must be reduced by \( \Delta_P \) and hence \( m_k(i) \) should satisfy:

\[
L_k \leq m_k(i) \leq U_{k-1} \quad (38)
\]

IV. Optimal Design Parameters

The objective of this section is to derive expressions for optimal values of \( n_P, n_D \) and \( Z \). Using 2’s complement binary system, we derive these expressions for two cases; one with the shifted partial residue \((P = rR_{j-1})\) represented in a non-redundant binary format, and another in a redundant carry-save format.

A. Using Non-Redundant Binary Representation

For a feasible \( m_k(i) \) value, the height of the overlap region \((\Delta y)\) at a given divisor value \((D)\) must be greater than the minimum grid \( 2^{-n_P} \), thus;

\[
\Delta y = U_{k-1} - L_k = (\rho^+ + \rho^- - 1) D
\]
At $D = D_{\text{min}}$, the height of the overlap region $\Delta y$ is minimum $\Delta y_{\text{min}} = U_{k-1} - L_k = (\rho^+ + \rho^- - 1) D_{\text{min}}$. Accordingly the minimum value of $n_P (n_P(\text{min}))$ is the smallest integer satisfying:

$$2^{-n_P(\text{min})} < (\rho^+ + \rho^- - 1) D_{\text{min}} \quad (39)$$

The lower bound of $n_P(\text{min})$ is reached at very high values of $Z$ ($Z \rightarrow \infty$ leading to $\omega_{\text{max}} \rightarrow 0$) in which case $\rho^+ = \rho^- = h$. This lower bound is the smallest integer value satisfying the following equation:

$$2^{-n_P(\text{Low Bound})} < (2h - 1) D_{\text{min}} \quad (40)$$

where $h = \frac{\alpha}{r-1}$ is the quotient digit set redundancy factor.

Defining $Z_1$ as the value of $Z$ at which $n_P(\text{min})$ is equal to its lower bound value, Equation (39) shows that $Z_1$ is the minimum integer satisfying:

$$2^{Z_1} > \frac{2 (h_B^+ + h_B^-)}{(2h - 1 - 2^{-n_P(\text{Low Bound})} D_{\text{min}})} \quad (41)$$

To exploit symmetry of the P-D diagram, we adopt the approach outlined in [14, 15] where the comparison constants $m_k(i)$ have been defined to stress the symmetric nature of the diagram. This makes it possible to only utilize 1st quadrant of the P-D diagram significantly reducing the hardware complexity of the quotient digit selection logic. In case of pure division, Fig. 2 shows P-D diagram for $r = 4, \alpha = 2, D_{\text{min}} = 0.5$, and $P$ is represented in non-redundant format. The comparison constants are defined as follows:

1) $m_0(i) = 0 \forall i$, i.e. for all ranges of $D$.
2) For $P > 0$:
   - For $1 \leq k \leq \alpha$ define:
     - $U_k(D) = (k + h)D, \quad L_k(D) = (k - h)D$
     - $P^+_{\text{lower}} \leq m_k(i) \leq P^+_{\text{upper}}$, where:
       - $P^+_{\text{upper}} = U_{k-1}(D) = (k - 1 + \rho^+) D$
       - $P^+_{\text{lower}} = L_k(D + 2^{-n_P}) = (k - \rho^-)(D + 2^{-n_D})$
     - $m_k(i) \leq P_i < m_{k+1}(i) \rightarrow q_{-j} = k$
   - $P_{\text{lower}} = L_{k-1}(D) = (k - 1 - \rho^-)D$
   - $m_{-(k+1)}(i) < P_i \leq m_{-k}(i) \rightarrow q_{+j} = -k$

In this context, the overlap region for a given divisor value $\Delta y^+$ for $P > 0$ and $\Delta y^-$ for $P < 0$ are given by:

$$\Delta y^+ = P^+_{\text{upper}} - P^+_{\text{lower}} = (\rho^+ + \rho^- - 1) D - (k - \rho^-)2^{-n_D} > 0 \quad (42)$$

$$\Delta y^- = P^-_{\text{upper}} - P^-_{\text{lower}} = (\rho^+ + \rho^- - 1) D - (k - \rho^-)2^{-n_D} > 0 \quad (43)$$

**Notes:**

1) The overlap range ($\Delta y^+$ or $\Delta y^-$) is smaller for smaller values of $D$.
2) Higher values of $k$ yield smaller overlap regions (for both $\Delta y^+$ and $\Delta y^-$).
3) For worst case analysis, the smallest values of $\Delta y^+$ ($\Delta y_{\text{min}}^+$) and for $\Delta y^-$ ($\Delta y_{\text{min}}^-$) occur at $D = D_{\text{min}}$ and $k = \alpha$. Thus:

$$\Delta y_{\text{min}}^+ = (\rho^+ + \rho^- - 1) D_{\text{min}} - (\alpha - \rho^-)2^{-n_D} > 0 \quad (44)$$

$$\Delta y_{\text{min}}^- = (\rho^+ + \rho^- - 1) D_{\text{min}} - (\alpha - \rho^-)2^{-n_D} > 0 \quad (45)$$

4) With a balanced multiplier digit set ($B^+ = B^-$), $\Delta y^+ = \Delta y^-$. However, if a $[0, r-1]$ non-redundant digit set is used, $B^+ (= 1)$ is greater than $B^- (= 0)$ in which case $\Delta y_{\text{min}}^- < \Delta y_{\text{min}}^+$, and worst case analysis should be performed on $\Delta y_{\text{min}}^+$. Equation (45) shows that the minimum value of $n_{D}(n_{D}(\text{min}))$ is the smallest integer value satisfying:

$$2^{-n_{D}(\text{min})} < \frac{(\rho^+ + \rho^- - 1)D_{\text{min}}}{(\alpha - \rho^+)} \quad (46)$$

The lower bound value of $n_{D}(\text{min})$ (Equation (46)) is reached at very high values of $Z$ (i.e., $Z \rightarrow \infty, \omega_{\text{max}} \rightarrow 0$) in which case $(\rho^+ = \rho^- = h = \frac{\alpha}{r-1})$. Such low bound value ($n_{D}(\text{Low Bound})$) is the smallest integer satisfying:

$$2^{-n_{D}(\text{Low Bound})} < \frac{(2h - 1)D_{\text{min}}}{(\alpha - h)} \quad (47)$$

Defining $Z_2$ as the value of $Z$ at which $n_{D}(\text{min})$ is equal to its lower bound value, Equation (46) shows that $Z_2$ is the
The lower bound of the minimum allowed value of

\[ 2^n P \]

To have a feasible solution, the numerator of Equation (55) shows that the heights of the overlap regions are lower by \( \Delta \) in this case, the minimum height of the overlap regions of \( P \) components of \( P \) used as input to the CPA is \( n_P \), the error introduced due to the use of CSA’s is less than \( 2^{n_P} \). In this case, the upper bounds for the comparison constants (\( P_{upper}^+ \) and \( P_{upper}^- \)) should be reduced by the same amount:

\[ P_{upper}^+ = U_{k-1}(D) - 2^{n_P} \]
\[ P_{upper}^- = U_{k}(D + 2^{n_D}) - 2^{n_P} \]

In this case, the minimum height of the overlap regions of Equations (44) and (45) become as follows:

\[ \Delta y_{min}^+ = (\rho^+ + \rho^- - 1)D_{min} - (\alpha - \rho^-)2^{n_D} - 2^{n_P} \]
\[ \Delta y_{min}^- = (\rho^+ + \rho^- - 1)D_{min} - (\alpha - \rho^+ )2^{n_D} - 2^{n_P} \]

Comparing Equations (52) and (53) with Equations (42) and (43) shows that the heights of the overlap regions are lower by \( 2^{n_P} \) in the carry-save redundant representation case.

To derive mathematical expressions for optimal parameters, we use the inequality \( \Delta y_{min}^- \geq 0 \).

\[ \Delta y_{min}^- = (\rho^+ + \rho^- - 1)D_{min} - (\alpha - \rho^+)2^{n_D} - 2^{n_P} > 0 \]

This yields the following constraint on the allowed minimum value of \( n_D \):

\[ 2^{-n_D} < \frac{[\rho^+ + \rho^- - 1]D_{min} - 2^{n_P}}{\alpha - \rho^+} \]

To have a feasible solution, the numerator of Equation (55) must be strictly greater than zero defining the minimum allowed value of \( n_P \) (\( n_P(min) \)) as the smallest integer value satisfying:

\[ 2^{-n_P(min)} < (\rho^+ + \rho^- - 1)D_{min} \]

The lower bound of the minimum \( n_P \) value \( (n_P(Low\_Bound)) \) is reached at very high values of \( Z \) (\( Z \rightarrow \infty \)) and is defined by the minimum integer value that satisfies the following equation:

\[ 2^{-n_P(Low\_Bound)} < (2h - 1)D_{min} \]

Defining \( Z_1 \) as the minimum value of \( Z \) at which \( n_P(min) = n_P(Low\_Bound) \) Equation (56) yields:

\[ 2^{2Z_1} > \frac{2 \left( h_B^+ + h_B^- \right)}{2h - 1 - 2^{-n_P(Low\_Bound)}} \]

Multiplying both sides of Equation (55) by \( 2^{-n_P} \), we obtain:

\[ 2^{-(n_P+|n_D|)} < \frac{[\rho^+ + \rho^- - 1]D_{min}2^{-n_P} - 2^{-2n_P}}{\alpha - \rho^+} \]

To reduce the complexity of the quotient digit selection hardware \( (n_P + n_D) \) must be minimized or equivalently the value of \( 2^{-(n_P+n_D)} \) must be maximized. Defining \( Y = 2^{-(n_P+n_D)} \), and \( X = 2^{-n_P} \), Equation (59) is rewritten as:

\[ Y < \frac{[\rho^+ + \rho^- - 1]D_{min}X - X^2}{\alpha - \rho^+} \]

Differentiating Equation (60) w.r.t. \( X \), the optimal value of \( n_P \) (\( n_P(opt) \)) is clearly the smallest value satisfying:

\[ 2^{(n_P(opt)-1)} > \frac{1}{(\rho^+ + \rho^- - 1)D_{min}} \]

It is clear that the computed value of \( n_P(opt) \) (Equation (61)) is higher by 1 than the minimum \( n_P \) value defined by Equation (56). Further, by using Equation (55) it is seen that the optimum \( n_D \) value is the minimum integer value satisfying:

\[ 2^{-(n_D(opt)-1)} < \frac{[\rho^+ + \rho^- - 1]D_{min}}{\alpha - \rho^+} \]

The lower bound of \( n_D(opt) \) \( (n_D(Low\_Bound)) \) is reached at very high values of \( Z \) (\( Z \rightarrow \infty \)) and is equal to the smallest integer value satisfying:

\[ 2^{-(n_D(Low\_Bound)-1)} < \frac{(2h - 1)D_{min}}{\alpha - h} \]

Let \( Z_2 \) be the minimum value of \( Z \) at which \( n_D(opt) = n_D(Low\_Bound) \). Using Equation (62), it can be shown that \( Z_2 \) is the smallest integer satisfying:

\[ 2^{(Z_2-1)} > \frac{h_B^+ \left( 1 + \frac{2^{-n_D(Low\_Bound)-1}}{D_{min}} \right) + h_B^-}{(2h - 1) - (\alpha - h)2^{-n_D(Low\_Bound)-1}} \]

The optimal value of \( Z \) is chosen as the larger value of \( Z_1 \) or \( Z_2 \), i.e.

\[ Z = MAX(Z_1, Z_2) \]

C. Determining \( Z \)

The value of \( Z \) is chosen as the higher of the two values; \( Z_1 \) and \( Z_2 \) which are derived from the lower bound values of \( n_P \) and \( n_D \). Expressions for \( Z_1 \) and \( Z_2 \) have been derived for both the non-redundant and redundant representation cases. In either case, the value of \( Z_1 \) is the same since the expressions for \( n_P(Low\_Bound) \) and \( Z_1 \) are identical in both cases as
can be readily seen when comparing equations (40) and (41) on one hand with Equations (57) and (58) on the other. For the case of redundant carry-save representation, the low bound value of $n_D$ as given by Equation (63) is higher by 1-bit than its value for the non-redundant representation case as given by Equation (47). Thus, it can be readily seen (Equations (48) and (64)) that the value of $Z_2$ will also be the same in both cases.

Accordingly, the value of $Z$ is independent of the representation format of $P$. The equations to derive $Z$ are summarized below:

1) Compute $n_P(Low\_Bound)$ as the minimum integer value satisfying $2^{-n_P(Low\_Bound)} < (2h - 1)D_{min}$
2) Compute $Z_1$ as the minimum integer value satisfying $2^{-Z_1} > 2\left(h_0^+ + h_0^-\right)\left(2h - 1 - 2^{-n_P(Low\_Bound)}\right)$
3) Compute $g$ as the minimum integer satisfying $2^{-g} < \left(\frac{2-D_{min}}{\alpha-h}\right)$
4) Compute $Z_2$ as the minimum integer value satisfying $2^{-Z_2} > 2\left(h_0^+\left[1+\frac{g}{2\alpha}\right]+h_0^-\right)\left(2h - 1 - 2^{-n_P(Low\_Bound)}\right)$
5) Compute $Z = \text{MAX}(Z_1 , Z_2)$

In special cases, analytical expressions for the values of $Z_1$, $Z_2$ and $Z$ can be derived. For example, for the case where $r = 2^m \geq 4$, $\alpha = r - 1$, $h = 1$, $2^{-n_P(Low\_Bound)} = 0.5$, $2^{-g} = 1/r$, $Z_1 = 3$, and $Z = Z_2 = m + 1$ for minimally redundant symmetric multiplier digit set ($B^+ = B^- = r/2$) as well as non-redundant multiplier digit set ($B^+ = B^- = 0$). Similarly, for the case where $\alpha = r/2$, $h = 0.5(1 + \frac{1}{r-1})$, $2^{-n_P(Low\_Bound)} = 1/r$, $2^{-g} = 2h/r$, $Z_1 = 2m + 2$, and $Z_2 = 2m + 1$ for minimally redundant symmetric multiplier digit set while for non-redundant multiplier digit set $Z = Z_1 = 2m + 1$, and $Z_2 = 2m$.

For values of $\alpha = r/2, r - 2$, and $r - 1$, Figures 3 and 4 show the values of $Z$ for both the non-redundant and the minimally redundant balanced multiplier digit sets respectively.

V. HIGH-RADIX MULTIPLIER-DIVIDER DESIGN PROCEDURE

Optimal values of $n_P$, $n_D$, and $Z$ are required for the design of a multiplier-divider of a given $r$, $\alpha$, $D_{min}$ and a $[-B^-, +B^+]$ multiplier digit set. To determine if a given set of $n_P$, $n_D$, and $Z$ values represent a valid solution, as well as determine the values of the comparison constants $m_k(i)$ for this solution, we follow the same approach detailed in [14]. This approach defines a validity function that can test for the validity of a given solution as well as an almost closed form to define the comparison constants of a given solution. We will define these expressions for the case of multiplier dividers for the general case which can be easily mapped for the more feasible special case of a balanced multiplier digit set to exploit features of symmetry in the P-D diagram.

The $i^{th}$ truncated divisor value $D_i$ and the comparison constants $m_k(i)$ of the $i^{th}$ divisor range $[D_i, D_i+1)$ are integer multiples of $2^{-n_P}$ and $2^{-n_P}$ respectively, these two parameters are represented as follows:

$$D_i = d_i \cdot 2^{-n_D}$$

$$m_k(i) = m_{k,i} \cdot 2^{-n_P} \quad \text{for} \quad P > 0$$

$$m_{-k}(i) = -m_{-k,i} \cdot 2^{-n_P} \quad \text{for} \quad P < 0$$

where the coefficients $m_{k,i}$, $m_{-k,i}$, and $d_i$ are positive integer values with $D_{min}2^{n_D} \leq d_i < D_{min}2^{n_D+1}$. For $P > 0$, the comparison constants have to satisfy:

$$P_{lower}^+ \leq m_k(i) \leq P_{upper}^+ \quad \text{or}$$

$$(k - \rho^-)(d_i + 1)2^{-n_P} \leq m_k(i)2^{-n_P} \leq (k - 1 + \rho^+d_i)2^{-n_P} - 2^{-n_P} \quad \text{or}$$

$$[2^{n_P-n_D}(k - \rho^-)(d_i + 1)] \leq m_{k,i} \leq [2^{n_P-n_D}(k + \rho^+1)d_i - 1]$$

As in [14], Equation (69) is used as the basis for defining the $m_{k,i}$ coefficients and the validity function $V(n_P, n_D, d_i)$ which verifies the validity of a given solution of $n_P$, $n_D$, and $Z$ as follows:

$$m_{k,i} = [2^{n_P-n_D}(k - \rho^-)(d_i + 1)] \quad \text{where} \quad D_{min}2^{n_D} \leq d_i < D_{min}2^{n_D+1}$$

Fig. 4. Values of $Z$ for different radices and minimally redundancy symmetric multiplier digit set.

Fig. 3. Values of $Z$ for different radices and non-redundant multiplier digit set.
\[ V(n_P, n_D, d_i) = [2^{n_P-n_D}(\alpha + \rho^+ - 1)d_i - 1] \]
\[ -[2^{n_P-n_D}(\alpha - \rho^-)(d_i + 1)] \] (redundant carry – save format) (71)
\[ V(n_P, n_D, d_i) = [2^{n_P-n_D}(\alpha + \rho^+ - 1)d_i - 1] \]
\[ -[2^{n_P-n_D}(\alpha - \rho^-)(d_i + 1)] \] (non-redundant binary format) (72)

Likewise, for \( P < 0 \), the \( m_{-k,i} \) coefficients and the validity functions are defined as:
\[ m_{-k,i} = [2^{n_P-n_D}(k - \rho^+)(d_i + 1) + 1] \] where \( D_{min}2^{n_D} \leq d_i < D_{min}2^{n_D+1} \) (73)
\[ V(n_P, n_D, d_i) = [2^{n_P-n_D}(\alpha + \rho^- - 1)d_i - 1] \]
\[ -[2^{n_P-n_D}(\alpha - \rho^+)(d_i + 1)] \] (redundant carry – save format) (74)
\[ V(n_P, n_D, d_i) = [2^{n_P-n_D}(\alpha + \rho^- - 1)d_i - 1] \]
\[ -[2^{n_P-n_D}(\alpha - \rho^+)(d_i + 1)] \] (non-redundant binary format) (75)

A solution (a given set of \( n_P, n_D \) and \( Z \)) is considered feasible if:
1) \( V(n_P, n_D, D_{min}2^{n_D}) \geq 1 \), or
2) \( V(n_P, n_D, d_i) = 0 \) for \( d_i = D_{min}2^{n_D}, \ldots, d_i - 1 \) and \( V(n_P, n_D, d_i) \geq 1 \).

In case of a \([0, r-1]\) non-redundant multiplier digit set, the P-D diagram is not symmetric and \( \Delta y_{min}^- < \Delta y_{min}^+ \) which means that Equations (74) or (75) should be used to verify the validity of a given solution. If, however, a balanced multiplier digit set is used, then \( B^+ = B^- \), \( h_B^+ = h_B^- = h_B \), and \( \rho^+ = \rho^- = \rho \) leading to a symmetric P-D diagram. In this case, the symmetry of the P-D diagram can be exploited to allow for the use of only the 1\(^{st} \) quadrant which leads to significant reduction in the hardware complexity of the digit selection logic [14], [15]. It should also be noted that the larger the size of the multiplier digit set is the narrower the overlap regions will be which means that a minimally redundant digit set is the best choice for multiplier \( B \).

Example
Let \( r = 4, \alpha = 2, D_{min} = 0.5, \) and \( B^+ = B^- = 2 \), we compute \( Z = 6 \) (section IV-C). This yields \( n_P(\min) = 3 \) with a corresponding \( n_D = 7 \) which is not a feasible solution with \( V(3, 7, 2^6) = -1 \). However, a feasible solution exists at \( Z = 6, n_P = 5, \) and \( n_D = 4 \) with \( V(5, 4, d_i) = 0 \) for \( d_i = 8, 9 \) but with \( d_i = 10 \) \( V(5, 4, 10) = 1 \). The coefficients \( (m_{k,i}) \) of the selection constants for this solution are given in Table I.

To reduce the area of the lookup table of the quotient digit selection function, the total number of address bits of this table must be reduced. Accordingly, an important objective of the design process of multiplier-dividers is to minimize \( n_Tot = n_P + n_D \). To come up with an optimal set of \( n_P, n_D, \) and \( Z, \) some choices/trade-offs are needed. In addition to the fundamental choices that are essential for the design of digit recurrence dividers, e.g. the radix, the quotient digit set and the representation method of the partial remainder [12], other choices are needed for the design of multiplier-dividers. For example, higher values of \( Z \) increase the number of iterations as well as the processor size while, at the same time, yielding lower \( n_P \) and \( n_{Ttot} \) values which would reduce the delay of each iteration as well as the area of the quotient digit selection function. The proposed method to compute \( Z \) as outlined in section IV-C, invariably yields solutions optimized for lower values of \( Z \) and \( n_Tot \). However, slightly higher values of \( Z \) may, in some cases, yield solutions with the same \( n_Tot \) but with lower \( n_P \) values. Lower \( n_P \) values have somewhat lower iteration delays and hardware complexity. For example, in the case of \( r = 4, \alpha = 2, D_{min} = 0.5, \) and \( B^+ = B^- = 2 \), the computed \( Z \) value is 6 which yields the solution \( n_P = 5, \) and \( n_D = 4 \). With \( Z \) increased by one bit to 7, we obtain another solution having the same \( n_Tot \) but with a lower \( n_P \). In other cases, values lower than the computed \( Z \) by one bit may yield the same solution allowing for a lower value of \( Z \) to be used. For example, the case of \( r = 64, \alpha = 32, \) and \( B^+ = B^- = 32 \), the computed value of \( Z \) is 14 yields a solution of \( n_P = 9, \) and \( n_D = 13 \). With \( Z \) smaller by one bit (\( Z = 13 \)), the same solution is obtained. Thus, if the target system parameters are to be optimized, it is advisable to investigate solutions for the computed \( Z \) as well as for \( Z - 1, \) and \( Z + 1 \). Another choice for multiplier-dividers is the \([-B^-, +B^+] \) range of the multiplier digit set. If a non-redundant digit set in the \([0, r-1] \) range is used, generation of \((A/r)\)-multiples would require a number of pre-computations for \( r \geq 4 \). If, however, a minimally redundant symmetric digit set \((B^+ = B^- = \frac{r}{2}) \) is used not only would that lead to fewer number of such pre-computations but it will also yield a symmetric P-D diagram that allows for the use of only the 1\(^{st} \) quadrant significantly reducing the hardware complexity of the selection function. It should also be pointed out that the height of overlap regions is slightly larger in case of a non-redundant multiplier digit set. However, with minimally redundant digit set the height reduction is almost insignificant and has practically no effect on the optimal \( n_P \) and \( n_D \) parameter values.

<table>
<thead>
<tr>
<th>Table I</th>
<th>1(^{st} ) Quadrant Comparison Constants Coefficients ( (m_{k,i}) ) versus ( n_D ) for ( r = 4, \alpha = 2, D_{min} = 0.5, B^+ = B^- = 2, Z = 6, n_P = 5, ) and ( n_{Tot} = 4 ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_i )</td>
<td>( 8 )</td>
</tr>
<tr>
<td>( k = 1 )</td>
<td>( m_{1,d_i} )</td>
</tr>
<tr>
<td>( k = 2 )</td>
<td>( m_{2,d_i} )</td>
</tr>
</tbody>
</table>

Based on the above, given the system radix \( r \), the quotient digit set parameter \( \alpha \), and the multiplier digit set parameters \( B^+ \) and \( B^- \), the optimal parameters for the high-radix multiplier-divider may be determined as follows:
1) Compute \( h = \frac{\alpha}{r-1}, h_B^+ = \frac{B^+}{r-1} \) and \( h_B^- = \frac{B^-}{r-1} \).
2) Determine \( Z \) as detailed in section IV-C.
3) For \( Z_1 = Z - 1 \) To \( Z + 1 \) do
   a) Compute \( \omega_{max} = 2^{1-Z} \).
b) Compute \( n_P(min) \) (Equation (39) or Equation (56)).

c) For \( n_P = n_P(min) \) To \( n_P(min) + 2 \)
   i) Compute the current \( n_D \) value (\( n_{DC} \)) corresponding to the current \( n_P \) value (Equation (46) or Equation (55))
   ii) For \( n_D = n_{DC} \) To \( n_{DC} + 2 \) do
      • If the current \( n_p, n_D \) and \( Z_i \) constitute a feasible solution then store \( Z_i, n_P, n_D \) and \( n_{Tot} = n_P + n_D \).

4) Select the solution which yields the smallest \( n_{Tot} \). In case of more than one solution having the same \( n_{Tot} \), choose the one with the smallest \( n_P \).

5) Generate the comparison constants \( m_k(i) \) for the selected solution (Equations (70) and (73)).

The described design procedure has been used to compute the design parameters for multiplier-dividers of various values of \( r \) and \( \alpha \). For a minimally redundant symmetric multiplier digit set, Fig. 4, Fig. 5, Fig. 6, and Fig. 7 show the derived values of \( Z \), \( n_P \), \( n_D \), and \( n_{Tot}(= n_P + n_D) \) respectively versus different radices for \( \alpha = r/2, \alpha = (r - 2) \) and \( \alpha = (r - 1) \).

It should be pointed out that the described multiplier-divider design procedure will generally yield an optimal \( n_{Tot} \) value which is higher than its corresponding value for a pure divider hardware. Consider, for example, the case of balanced multiplier digit set where expressions for \( U_k \) and \( L_k \) are identical to those of the pure divider case only if \( Rho^+ = Rho^- = h \). Theoretically, however, \( Rho^+ \) and \( Rho^- \) will equal \( h \) only at \( Z = \infty \). When \( Z1 \) and \( Z2 \) have feasible values, the proposed method to compute \( Z \) will only ensure that the values of \( n_P(min) \) and \( n_D(min) \) of the multiplier-divider equal those of the corresponding pure divider. This yields solutions which have either the same value of \( n_{Tot} \) or higher by just one bit at reasonable values of \( Z \). In some cases where the multiplier-divider has a larger \( n_{Tot} \) value, solutions having the same \( n_{Tot} \) as pure dividers are possible to obtain but at impractically large values of \( Z \). This is clear in the case where \( r = 4, \alpha = 2 \) where the multiplier-divider design procedure yields a solution of \( n_{Tot} = 9 \) at \( Z = 5 \) whereas for a pure divider \( n_{Tot} = 8 \). If, however, the value of \( Z \) is raised to 48 or higher, a solution exists where \( n_{Tot} = 8 \). In such case, even though the reduced \( n_{Tot} \) yields a smaller lookup table, the sizable increase in \( Z \) results in unacceptably larger number of iterations as well as processor size. Section VII gives the major design parameter values (\( Z, n_P \) and \( n_D \)) of various multiplier-divider systems.

As shown in Section IV-C, \( Z \) is highest (= \( 2m + 2 \)) when minimally redundant digit sets are used for both the quotient and multiplier. Thus, as a worst-case, \( Z \) may assume values as high as \( 2m + 3 \) since the above described design procedure investigates values up to \( Z + 1 \). Thus, the initial delay of \( \lceil Z \rceil \) clock cycles has a theoretical upper-bound value of 3 clock cycles for \( r \in \{4, 8\} \) and 2 cycles for \( r > 8 \).

VI. HARDWARE IMPLEMENTATION FOR REDUNDANT REPRESENTATION

Fig. 8 shows a possible hardware implementation of our proposed high-radix multiplier-divider of a balanced multiplier digit set for the case of \( D_{min} = 0.5 \). It has the following features:
A counter is used to hold the number of iterations to be performed ($n_{iterate} = \frac{\ell + Z}{m}$).

The most significant bits of the $\ell$-bit $B$-register are passed to combinational logic that generates the desired digit set, e.g., a balanced signed digit set, the output is the current digit ($b_{\ell-1}$) of $B$. In each iteration register $B$ is shifted left by $m$-bits.

Utilizing symmetry of the P-D diagram, depending on the sign of $P_t$, $m+n_P$ bits of either $P_t$ or its $1's$ complement are passed as address to the lookup table of the quotient digit selection function.

The selection function is implemented either as a ROM or a PLA where the truncated values of $P$ and $D$ ($P_t$ and $D_t$) are the input to this ROM (or PLA) for a total of $(m+n_P) + (-1)^{2D_{min}}$ bits. The output of the ROM/PLA is a $1^{st}$-quadrant quotient digit of $[Log_2(\alpha+1)] = m$-bits.

The value of $P_r = rR_t$ uses a redundant representation in the form of a SUM component ($PS$), and a CARRY component ($PC$) which are held in the registers $PSR$ and $PCR$ respectively. Accordingly there are 4 quantities that need to be added each iteration: $PS$, $PC$, ($-q_jD$), and ($b_{\ell-1}A/r$).

- The multiplexer $MUX_a$ generates the $\ell + m$ bits of $|b_{\ell-1}A'|$ or its 1’s complement depending on the sign of $b_{\ell-1}$. This is sign-extended by $1 + Z + m$ bits.
- To generate the signed 2’s complement of $(b_{\ell-1}A/r)$, where $A = A'/2^\ell$, the sign of $b_{\ell-1}$ is fed as the least significant bit of the left-shifted copy of PC.
- The output $\ell + m + 1$ bits of $MUX_d$ is either ($|q_jD|$) or its signed 1’s complement depending on the sign of $q_j$. The output of $MUX_x$ is left appended by $Z + m$ bits each having a value that equals the sign bit of $MUX_d$ output (sign-bit extension). To add the signed 2’s complement of ($-q_jD$), the carry-in of the (4:2) compressor is fed with the sign of ($-q_jD$).
- A Carry Lookahead Adder (CLA) is used to add the $(1 + m + n_P)$ most significant bits of the sum and carry components of the shifted partial residue ($PS$ and $PC$). The resulting summation is the truncated $P_t$ value used as input to the ROM/PLA.
- Adding the four quantities $PS$, $PC$, ($-q_jD$), and ($b_{\ell-1}A'$) is done using a (4:2) compressor yielding two outputs; a partial sum component (Sum), and a partial carry component (Cry).
- An $m$-bit left-shifted version of Sum and Cry are stored in two registers ($PSR$ and $PCR$) to represent ($rR_t$). The outputs of $PSR$ and $PCR$ are fed-back as input to the (4:2) compressor representing the shifted partial residue ($rR_t$) while the $(1 + m + n_P)$ most significant bits of $PSR$ and $PCR$ are added using the CLA to yield the value of $P_t$.
- At the last iteration, a CPA is used to assimilate the sum and carry components of the shifted partial residue ($PS$ and $PC$) to yield the value of $P$. This CPA may (or may not) utilize the $(1 + m + n_P)$-bit CLA to yield the $(1 + m + n_P)$ most-significant bits of the result as shown in Fig. 8.
- Using higher radices results in larger implementation areas. If the selection function is implemented as a ROM, its total capacity in bits would be $2^{m+n_P} + (-1)^{2D_{min}} \times m$ bits. Referring to Fig. 7, under worst case scenario, $n_{rod}$ increases linearly with $m$, i.e. $n_{rod} = \lambda m + \sigma$ in which case the ROM capacity is $2^{\sigma-1} \frac{\lambda}{\sigma+1} \times m$. Thus, assuming a minimally redundant multiplier digit set, for $\alpha = r/2$ where $\lambda = 3$ and $\sigma = 4$ the ROM capacity is of order $O(r^4 Log_2 r)$, while for $\alpha = r - 1$ where $\lambda = 1$ and $\sigma = 5$ the ROM capacity is of order $O(r^5 Log_2 r)$. Sizes of other data path modules, e.g., the (4:2) compressor, registers, adds, and multiplexers depend on $m$, $Z$, or $n_P$. Since, under worst case scenario, $Z$ and $n_P$ increase linearly with $m$ as shown in Fig. 3, 4, and 5, the area complexity of these multiplier-divider data path modules is $O(Log_2 r)$ under worst-case scenario.

VII. RESULTS AND DISCUSSION

The described multiplier-divider design procedure and hardware have been modeled and verified using VHDL. The model has been used to compute the major design parameters for various values of $r$, and $\alpha$ as well as for a minimally redundant balanced multiplier digit set and a $[0, r - 1]$ non-redundant
digit set. Table II lists the obtained results where \( Z_{\text{bal}} \) is the value of \( Z \) when a balanced minimally redundant digit set is used for \( B \), while \( Z_{\text{un}} \) is the value of \( Z \) when a \([0, r - 1]\) non-redundant digit set is used.

### Table II

<table>
<thead>
<tr>
<th>( r )</th>
<th>( \alpha )</th>
<th>( Z_{\text{bal}} )</th>
<th>( Z_{\text{un}} )</th>
<th>( n_P )</th>
<th>( n_D )</th>
<th>( \text{Initial Delay} )</th>
<th>( \text{bal / unbal} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>4 / -</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>- / 3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3 / 2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2 / 3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>48</td>
<td>49</td>
<td>4</td>
<td>4</td>
<td>24 / 24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1 / 2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>2 / 2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>1 / 1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td>2 / 2</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>1 / 1</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>11</td>
<td>10</td>
<td>8</td>
<td>11</td>
<td>2 / 2</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>31</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>1 / 1</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>32</td>
<td>13</td>
<td>12</td>
<td>9</td>
<td>13</td>
<td>2 / 2</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>63</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>1 / 1</td>
<td></td>
</tr>
</tbody>
</table>

For all tested combinations of \( r, \alpha \) and \( D_{\text{min}} \), with the exception of the \( r = 2 \) case, it was found that both the balanced minimally redundant and the non-redundant multiplier digit sets yield the same \( n_P \) and \( n_D \) values. However, the values of \( Z \) are generally slightly lower in the non-redundant case \( (Z_{\text{un}}) \) as can be seen from Table II. This is not unexpected since the overlap region height is slightly larger in this case compared to the case when a balanced digit set is used. Parameter values were also obtained for the case where \( D_{\text{min}} = 1.0 \) and, as expected, exactly the same results were obtained as listed in Table II but with \( n_P \) and \( n_D \) lower by one bit.

The case of \( r = 4, \alpha = 3 \), yields \( n_P \) and \( n_D \) values that are identical to those of a pure divider. For the case of \( r = 4, \alpha = 2 \), the multiplier-divider design procedure described in Section V yields a solution of \( Z = 5 \) and \( n_{\text{Trot}} = 9 \) higher by one bit than that of a pure divider [15]. In the same case, however, the same values of \( n_P \) and \( n_D \) as those of a pure divider are obtained only at \( Z \geq 48 \).

For the case of \( r = 2 \) and \( D_{\text{min}} = 0.5 \), even though the design equation of \( Z_2 \) (Equation (64)) yields infeasible values, a solution is possible by assigning \( Z \) the computed value of \( Z_1 \) in this case. For a balanced minimally redundant multiplier digit set, \( Z = Z_1 = 4 \) and we obtain a solution at \( n_P = 3 \) and \( n_D = 1 \). A better solution is obtained for a \([0, r - 1]\) non-redundant binary multiplier digit set where \( Z = Z_1 = 3 \), \( n_P = 2 \) and \( n_D = 1 \) with comparison constants at \( m_0(i) = -0.5 \) and \( m_1(i) = 0 \).

The last column of Table II shows the computed initial delay \((= \lfloor \frac{Z}{m} \rfloor)\) for a balanced minimally redundant multiplier digit set as well as a non-redundant set. It is clear that for \( r > 4 \) this delay is only either 1 or 2 clock cycles depending on whether the quotient digit set is maximally or minimally redundant balanced set respectively. It may be worth while to point out that a multiplier-divider that is built by cascading an on-line divider to an on-line multiplier will have an overall on-line delay that is equal to the sum of the individual on-line delays of the multiplier and the divider. While on-line multipliers have a typical on-line delay of 3 [10], the on-line delay of on-line dividers varies based on the the used radix, quotient digit set, and the selection function. Depending on the complexity of the implementation, various on-line delays for dividers ranging from as small as 3 [20], to as large as 8 [21] have been reported. Accordingly, a cascaded on-line multiplier-divider would have an on-line delay in the range of 6-11 clock cycles.

## CONCLUSION

A general radix recurrence relation and a design methodology which describes how to efficiently design fast digital multiplier-dividers have been developed. The proposed single recurrence relation performs simultaneous multiplication and division. With the fused implementation of multiplication and division, the two operations can be executed using a single instruction, implying only a single rounding. The design methodology is based on an analytical model with a set of equations fully defining and relating lower bounds and optimal values for the number of fractional bits of both the divisor and the shifted partial remainder to be used as input to the quotient digit selection logic. The developed design strategy and method has been modeled using VHDL and used in the design of multiplier-dividers for variety of radix and quotient digit set values. A proposed hardware implementation has also been modeled and verified.

## ACKNOWLEDGMENT

The authors would like to acknowledge the support of the Computer Engineering Department of King Fahd University of Petroleum & Minerals (KFUPM) and King Abdul-Aziz City for Science & Technology (KACST). This work was partially supported by a grant from King Abdul-Aziz City for Science & Technology under grant AR 22-17. We’d like to also acknowledge the reviewers of the paper as well as the associate editor professor Elisardo Antelo for their insightful comments and suggestions that have further improved the quality of the work presented here.

## REFERENCES


Alaeldin Amin Dr. Amin obtained his PhD from the University of Utah in 1987. He was a member of the MOS memory R & D group of National Semiconductor corporation for over 8 years before joining the Computer Engineering Department of King Fahd University of Petroleum & Minerals in 1988. Dr. Amin holds four US patents with three more pending applications. Dr. Amin research interests include computer arithmetic, VLSI circuit design, digital system testing and design for testability, digital system modeling and synthesis, memory testing, design automation and computer architecture.