A Tutorial on Digit Recurrence Algorithms for Division and Square Root

Joonsang Yoon

July 28, 2025

Abstract

Digit recurrence algorithms, particularly the SRT (Sweeney, Robertson, and Tocher) family, form the backbone of high-speed division and square root units in digital computers. They achieve high performance by employing a redundant number system for the result digits. This core principle of redundancy introduces tolerance for imprecision, allowing the critical digit selection step to be performed with fast, truncated estimates of the internal state, thereby avoiding the performance bottleneck of full-width carry-propagating additions. This paper provides a comprehensive tutorial on the theory and practice of these algorithms. We present a step-by-step derivation of the fundamental recurrence relations for both division and square root. The tutorial meticulously explains how to establish the convergence conditions and then guides the reader through the complete design process for the digit selection logic. To ground the theory in practice, we develop detailed Radix-2 and Radix-4 examples, demonstrating how to determine key design parameters and derive the final selection logic from first principles.

1 Introduction

Division and square root are fundamental arithmetic operations in modern microprocessors, essential for a wide range of applications in scientific computing, signal processing, and graphics. Unlike addition and multiplication, which can be implemented with relatively straightforward combinatorial logic, division and square root are inherently iterative. The most prevalent and efficient hardware methods for these operations are digit recurrence algorithms.

The core idea of a digit recurrence algorithm is to compute the result (quotient or square root) one digit at a time, from most significant to least significant. Each iteration involves two main steps: selecting a new digit of the result and calculating a new partial remainder, or *residual*. The performance of the entire algorithm hinges on the speed of this iterative loop. A naive implementation faces a critical bottleneck in the digit selection step. To select a correct digit, one must compare the current residual against multiples of the divisor (or partial root). This comparison requires the precise value of the residual, which is updated in every cycle. Calculating this precise value would necessitate a full-width, carry-propagating addition, making the loop unacceptably slow.

The SRT family of algorithms, named after their independent inventors Sweeney, Robertson, and Tocher, provides an elegant solution to this bottleneck. The key innovation is the use of a *redundant digit set* for the result. For example, in Radix-2, instead of the standard digits $\{0, 1\}$, a redundant set such as $\{-1, 0, 1\}$ is used. This redundancy relaxes the constraints on digit selection, creating an overlap between the valid selection regions for adjacent digits. This overlap is the crucial feature that enables high performance: because the regions overlap, a correct digit can be chosen using a low-precision, truncated estimate of the residual. This eliminates the need for a slow, full-width addition in the critical path. The iterative loop can thus be made very fast, typically limited only by a short carry-propagate addition on a few most-significant bits and a small lookup table.

This paper serves as a tutorial on the design and analysis of digit recurrence algorithms, providing a thorough and clear exposition of their mathematical foundations. The structure of the paper is as follows:

- section 2 introduces the concept of redundant digit sets and the redundancy factor ρ, the parameter that quantifies the overlap essential to these algorithms.

- section 3 provides a complete derivation of the digit recurrence algorithm for division. We establish the residual recurrence, the containment condition, and the digit selection intervals, then show how to design practical selection logic using truncated operands for Radix-2 and Radix-4.
- section 4 extends the methodology to the square root operation. We derive the corresponding recurrence relations and selection intervals, highlighting the similarities and key differences compared to division, and again provide detailed Radix-2 and Radix-4 design examples.
- section 5 summarizes the key principles and design trade-offs discussed throughout the paper.

2 Redundancy in Digit Sets

The cornerstone of high-speed digit recurrence algorithms is the use of a redundant number system for the digits of the result. Instead of a standard non-redundant set for radix B, such as $\{0, 1, \ldots, B-1\}$, these algorithms employ a symmetric, redundant digit set.

At each step j of the algorithm, the selected digit (q_{j+1} for division or s_{j+1} for square root) is chosen from the set:

$$q_{j+1}, s_{j+1} \in \{-a, -a+1, \ldots, a-1, a\} \tag{1}$$

where a is an integer representing the largest possible digit magnitude. The degree of redundancy is quantified by the *redundancy factor*, ρ, defined as the ratio of the largest digit in the redundant set to the largest digit in a non-redundant set:

$$\rho = \frac{a}{B-1} \tag{2}$$

The fundamental requirement for a high-speed SRT-style algorithm is that this redundancy factor must be greater than one-half:

$$\rho > \frac{1}{2} \tag{3}$$

This condition is critical because it guarantees that the *selection intervals* for adjacent digits overlap. As we will derive in section 3, the choice of a digit k is valid only for a certain range of residual values. The condition $\rho > 1/2$ ensures that the valid range for selecting digit k and the valid range for selecting digit $k+1$ have a non-empty intersection. This overlap means there are values of the residual for which either choice is correct. It is precisely this flexibility that allows the digit selection to be based on a fast, truncated estimate of the residual, rather than its exact, slowly-computed value. A non-redundant system corresponds to $\rho = 1/2$, where selection intervals meet at precise boundaries, demanding an exact comparison.

The constraints on the digit magnitude a are a direct consequence of the required range for ρ. For a practical and efficient implementation, ρ is bounded as:

$$\frac{1}{2} < \rho \leq 1 \tag{4}$$

Substituting the definition of ρ from eq. (2) into this inequality yields the required range for a:

$$\frac{B-1}{2} < a \leq B-1 \tag{5}$$

The lower bound, $a > (B-1)/2$, is the mathematical requirement to ensure $\rho > 1/2$ and thus guarantee the essential overlap. The upper bound, $a \leq B-1$, is a practical hardware constraint. While a larger a provides more redundancy, choosing $a > B-1$ complicates the logic needed to form the term $q_{j+1}D$ in the recurrence, offering diminishing returns for the added complexity. A choice of $a = B-1$ (where $\rho = 1$) is termed "maximally redundant," while choices of a closer to $(B-1)/2$ (where $\rho \to 1/2$) are "minimally redundant."

3 Division by Digit Recurrence

We begin by developing the algorithm for division. The fundamental relationship is:

$$N = DQ + R \tag{6}$$

where N is the dividend, D is the divisor, Q is the quotient, and R is the final remainder. For fractional arithmetic, it is standard practice to normalize the operands. We assume the following ranges:

$$N \in \left[\frac{1}{4}, \frac{1}{2}\right) \qquad \text{(Dividend)}$$

$$D \in \left[\frac{1}{2}, 1\right) \qquad \text{(Divisor)}$$

These ranges guarantee that the quotient Q is also in a predictable range, preventing overflow and simplifying the algorithm:

$$Q \in \left(\frac{N_{\min}}{D_{\max}}, \frac{N_{\max}}{D_{\min}}\right) = \left(\frac{1/4}{1}, \frac{1/2}{1/2}\right) = \left(\frac{1}{4}, 1\right) \tag{7}$$

The quotient Q is computed iteratively over n steps, where $j = 0, 1, \ldots, n-1$. Let Q_j be the quotient estimate after j digits $(q_1, \ldots, q_j)$ have been determined. The quotient is refined at each step by adding a new term:

$$Q_{j+1} = Q_j + q_{j+1} B^{-(j+1)} \tag{8}$$

Unfolding this recurrence shows the composition of the quotient at step j:

$$\text{At step } j, \quad Q_{j+1} = Q_0 + \sum_{i=0}^{j} q_{i+1} B^{-(i+1)} \tag{9}$$

After n iterations, the final quotient is computed to the desired precision:

$$\text{At step } j = n-1, \quad Q_n = Q_0 + \sum_{i=0}^{n-1} q_{i+1} B^{-(i+1)} = Q \tag{10}$$

The core of the algorithm is ensuring that the process converges to the true quotient. At any step j, the error in the current approximation Q_{j+1} is given by:

$$\varepsilon_{j+1} = \frac{N}{D} - Q_{j+1} = \sum_{i=j+1}^{\infty} q_{i+1} B^{-(i+1)} \tag{11}$$

The magnitude of this error is bounded by the largest possible value of the remaining undiscovered digits. Using the digit bound a:

$$\left| \sum_{i=j+1}^{\infty} q_{i+1} B^{-(i+1)} \right| \leq \sum_{i=j+1}^{\infty} a B^{-(i+1)} = a B^{-(j+2)} \sum_{k=0}^{\infty} B^{-k} = a B^{-(j+2)} \frac{1}{1 - B^{-1}} = \frac{a}{B-1} B^{-(j+1)} \tag{12}$$

This gives the fundamental error bound in terms of the redundancy factor ρ:

$$|\varepsilon_{j+1}| \leq \rho B^{-(j+1)} \tag{13}$$

Substituting the definition of the error, we get:

$$\left| \frac{N}{D} - Q_{j+1} \right| \leq \rho B^{-(j+1)} \tag{14}$$

To work with integer-like quantities in hardware, we define a scaled residual, R_{j+1}. Multiplying by D and scaling by B^{j+1} gives:

$$\left| B^{j+1} (N - D Q_{j+1}) \right| \leq \rho D \tag{15}$$

We define the scaled residual at the end of step j as:

$$R_{j+1} = B^{j+1}(N - DQ_{j+1}) \tag{16}$$

This leads to the crucial *containment condition*: the residual must always be bounded by a multiple of the divisor.

$$|R_{j+1}| \leq \rho D \tag{17}$$

The final, unscaled remainder R is related to the final residual R_n by:

$$\text{At step } j = n-1, \quad R_n = B^n(N - DQ_n) = B^n R \tag{18}$$

To implement the algorithm, we need an iterative way to compute R_{j+1} from R_j. We derive this recurrence from its definition in eq. (16):

$$\begin{aligned}
R_{j+1} &= B^{j+1}(N - DQ_{j+1}) && \text{Definition of } R_{j+1} \\
&= B^{j+1}\left(N - D\left(Q_j + q_{j+1}B^{-(j+1)}\right)\right) && \text{Using eq. (8)} \\
&= B^{j+1}(N - DQ_j) - B^{j+1}Dq_{j+1}B^{-(j+1)} && \text{Distributing } D \\
&= B\left(B^j(N - DQ_j)\right) - q_{j+1}D && \text{Simplifying} \\
&= BR_j - q_{j+1}D && \text{Using the definition of } R_j
\end{aligned}$$

This gives the central recurrence relation for division:

$$R_{j+1} = BR_j - q_{j+1}D \tag{19}$$

At each step j, the algorithm performs two main tasks: (1) select a digit q_{j+1} based on the current residual R_j and the divisor D, and (2) compute the next residual R_{j+1} using this recurrence. The selection of q_{j+1} must be done such that the next residual R_{j+1} satisfies the containment condition in eq. (17).

3.1 Selection Intervals for a Positive Divisor

For a positive divisor, $D \in \left[\frac{1}{2}, 1\right)$, the containment condition from eq. (17) is:

$$|R_{j+1}| \leq \rho D \implies -\rho D \leq R_{j+1} \leq \rho D \tag{20}$$

We substitute the recurrence relation from eq. (19) into this inequality:

$$-\rho D \leq BR_j - q_{j+1}D \leq \rho D \tag{21}$$

To find the condition that BR_j must satisfy to select a particular digit q_{j+1}, we add $q_{j+1}D$ to all parts of the inequality:

$$(q_{j+1} - \rho)D \leq BR_j \leq (q_{j+1} + \rho)D \tag{22}$$

This equation defines a selection interval for each possible digit q_{j+1}. The goal of the digit selection logic is to determine which interval the shifted residual BR_j falls into. The fact that $\rho > 1/2$ ensures these intervals overlap, making the selection robust to small errors.

3.2 Selection Intervals for a Negative Divisor

For a negative divisor, $D \in \left[-1, -\frac{1}{2}\right)$, the magnitude is $|D| = -D$. The containment condition is still based on the magnitude:

$$|R_{j+1}| \leq \rho|D| \implies |R_{j+1}| \leq -\rho D \tag{23}$$

This expands to:

$$\rho D \leq R_{j+1} \leq -\rho D \tag{24}$$

Substituting the recurrence relation from eq. (19):

$$\rho D \leq BR_j - q_{j+1}D \leq -\rho D \tag{25}$$

Adding $q_{j+1}D$ yields the selection interval:

$$(q_{j+1} + \rho)D \leq BR_j \leq (q_{j+1} - \rho)D \tag{26}$$

3.3 Initialization

The first step of the algorithm ($j = 0$) requires initial values for the quotient approximation, Q_0, and the residual, R_0. These values must be chosen such that the first shifted residual, BR_0, falls within a valid selection region for the first quotient digit, q_1. This ensures the recurrence can start correctly and that the subsequent residual, R_1, will satisfy the containment condition.

A common and robust initialization strategy is to set the initial quotient approximation to $Q_0 = 0$ and the initial residual to $R_0 = N/B$. This choice has the convenient property that the first shifted residual is simply the dividend itself:

$$BR_0 = B\left(\frac{N}{B}\right) = N \tag{27}$$

With the assumed operand ranges, for a positive divisor $D \in \left[\frac{1}{2}, 1\right)$, the dividend is $N \in \left[\frac{1}{4}, \frac{1}{2}\right)$, so the shifted residual BR_0 is also in the range $\left[\frac{1}{4}, \frac{1}{2}\right)$.

To verify this initialization, we must show that the range of BR_0 is fully covered by the union of the selection intervals for q_1, as defined in eq. (22) for a positive divisor. The selection condition for q_1 is:

$$(q_1 - \rho)D \leq BR_0 \leq (q_1 + \rho)D \tag{28}$$

Since BR_0 is positive, we only need to check the intervals for non-negative digits ($q_1 \in \{0, 1, \ldots, a\}$).

In the Radix-2 case ($B = 2, \rho = 1$), the shifted residual range is $2R_0 = N \in \left[\frac{1}{4}, \frac{1}{2}\right)$. We check the selection intervals for $q_1 = 0$ and $q_1 = 1$:

- For $q_1 = 1$: The interval is $[(1 - 1)D, (1 + 1)D] = [0, 2D]$.
- For $q_1 = 0$: The interval is $[(0 - 1)D, (0 + 1)D] = [-D, D]$.

The union of these intervals is $[-D, 2D]$. For any $D \in \left[\frac{1}{2}, 1\right)$, this union covers at least $\left[-\frac{1}{2}, 1\right]$. This range safely contains the required range of $2R_0 \in \left[\frac{1}{4}, \frac{1}{2}\right)$, so the initialization is valid.

In the Radix-4 case ($B = 4, \rho = 2/3$), the shifted residual range is $4R_0 = N \in \left[\frac{1}{4}, \frac{1}{2}\right)$. We check the intervals for $q_1 \in \{0, 1, 2\}$:

- For $q_1 = 2$: The interval is $\left[\left(2 - \frac{2}{3}\right)D, \left(2 + \frac{2}{3}\right)D\right] = \left[\frac{4}{3}D, \frac{8}{3}D\right]$.
- For $q_1 = 1$: The interval is $\left[\left(1 - \frac{2}{3}\right)D, \left(1 + \frac{2}{3}\right)D\right] = \left[\frac{1}{3}D, \frac{5}{3}D\right]$.
- For $q_1 = 0$: The interval is $\left[\left(0 - \frac{2}{3}\right)D, \left(0 + \frac{2}{3}\right)D\right] = \left[-\frac{2}{3}D, \frac{2}{3}D\right]$.

The union of these intervals is $\left[-\frac{2}{3}D, \frac{8}{3}D\right]$. For any $D \in \left[\frac{1}{2}, 1\right)$, this union covers at least $\left[-\frac{1}{3}, \frac{4}{3}\right]$. This range safely contains the required range of $4R_0 \in \left[\frac{1}{4}, \frac{1}{2}\right)$, confirming the initialization is also valid for the Radix-4 case.

For a negative divisor, the analysis is analogous using the selection intervals from eq. (26). The principle of ensuring the initial shifted residual falls within the union of valid selection regions remains the same.

3.4 Example: Radix-2 SRT Division

Let's consider the simplest and most common case: Radix-2 ($B = 2$) with a maximally redundant digit set $\{-1, 0, 1\}$, which means we choose $a = 1$. The redundancy factor is:

$$\rho = \frac{a}{B - 1} = \frac{1}{2 - 1} = 1 \tag{29}$$

3.4.1 Positive Divisor

With $\rho = 1$ and $B = 2$, the selection intervals from eq. (22) become:

$$q_{j+1} = \begin{cases} 1 & \text{if } 0 \leq 2R_j \leq 2D \\ 0 & \text{if } -D \leq 2R_j \leq D \\ -1 & \text{if } -2D \leq 2R_j \leq 0 \end{cases} \tag{30}$$

These intervals overlap. For example, if $2R_j = 0.2D$, both $q_{j+1} = 1$ and $q_{j+1} = 0$ are valid choices. This overlap is what allows for a simplified implementation.

In hardware, the residual R_j is typically kept in carry-save form to avoid slow carry propagation in the main loop. To perform the comparisons in the selection logic, we would need to compute the full value of $2R_j$, which requires a fast adder. To avoid this, we compute an estimate of $2R_j$ by summing only a few of its most significant bits.

Let the full-precision shifted residual be $2R_j = r_{j,2\ldots0}.r_{j,-1\ldots-\infty}$. This value is the sum of two bit-vectors in carry-save representation: $Y_j = y_{j,2\ldots0}.y_{j,-1\ldots-\infty}$ and $Z_j = z_{j,2\ldots0}.z_{j,-1\ldots-\infty}$.

$$r_{j,2\ldots0}.r_{j,-1\ldots-\infty} = y_{j,2\ldots0}.y_{j,-1\ldots-\infty} + z_{j,2\ldots0}.z_{j,-1\ldots-\infty} \tag{31}$$

We form a truncated estimate t_j by adding the top bits of Y_j and Z_j using a small carry-propagate adder (CPA). Let's say we use α fractional bits.

$$t_{j,2\ldots0}.t_{j,-1\ldots-\alpha} = y_{j,2\ldots0}.y_{j,-1\ldots-\alpha} + z_{j,2\ldots0}.z_{j,-1\ldots-\alpha} \tag{32}$$

This truncation introduces a one-sided error. The sum of the truncated parts is less than or equal to the true sum.

$$0 \le 2R_j - t_{j,2\ldots0}.t_{j,-1\ldots-\alpha} < 2 \cdot 2^{-\alpha} = 2^{-(\alpha-1)} \tag{33}$$

Similarly, we use a truncated version of the divisor D, using its first β fractional bits. We define integer-valued estimates for use in a simple lookup table (PLA):

$$\tau_j = \mathrm{Int}(2^{\alpha} \cdot t_{j,2\ldots0}.t_{j,-1\ldots-\alpha}) \in \left\{-2^{\alpha+2}, \ldots, 2^{\alpha+2} - 1\right\} \qquad (\tau_j \le 2^{\alpha+1} R_j < \tau_j + 2) \tag{34}$$

$$\delta = \mathrm{Int}(2^{\beta} \cdot 0.1d_{j,-2\ldots-\beta}) \in \left\{2^{\beta-1}, \ldots, 2^{\beta} - 1\right\} \qquad (\delta \le 2^{\beta} D < \delta + 1) \tag{35}$$

The selection logic must choose a digit $k \in \{-1, 0, 1\}$ based on τ_j and δ.

To ensure a correct digit can be selected using the truncated estimates, the selection regions must have sufficient overlap to absorb the uncertainty introduced by truncation. This is formalized by the *continuity condition*. Let $L_k(D) = (k - \rho)D$ and $U_k(D) = (k + \rho)D$ be the lower and upper bounds of the selection interval for digit k. The condition requires that for any adjacent digits k and $k - 1$, the region for $k - 1$ must overlap with the region for k even under the worst-case truncation errors. Specifically, the minimum possible value of the upper bound for digit $k - 1$ must be greater than or equal to the maximum possible value of the lower bound for digit k:

$$\min(U_{k-1,\mathrm{trunc}}) \ge \max(L_{k,\mathrm{trunc}}) \tag{36}$$

To evaluate this, we must find the extrema of the bounds over the uncertainty intervals of D and $2R_j$. The truncation of the residual introduces a one-sided error of up to $2^{-(\alpha-1)}$, which lowers the minimum value of the upper bound. The truncation of the divisor means $D \in \left[2^{-\beta}\delta, 2^{-\beta}(\delta + 1)\right)$. This leads to the following expressions for the truncated bounds (assuming positive coefficients for D):

$$\min(U_{k-1,\mathrm{trunc}}) = \min_D(U_{k-1}(D)) - 2^{-(\alpha-1)} = (k - 1 + \rho)(2^{-\beta}\delta) - 2^{-(\alpha-1)} \tag{37}$$

$$\max(L_{k,\mathrm{trunc}}) = \max_D(L_k(D)) = (k - \rho)(2^{-\beta}(\delta + 1)) \tag{38}$$

Substituting these into the continuity condition gives the inequality that must be solved to find the required precision parameters α and β. For negative coefficients, the roles of δ and $\delta + 1$ are swapped. We analyze this for the boundaries between selection regions:

- For $k = 1$ (U_0, L_1): $(0+1)(2^{-\beta}\delta) - 2^{-(\alpha-1)} \ge (1-1)(2^{-\beta}(\delta+1))$. This simplifies to $2^{-\beta}\delta - 2^{-(\alpha-1)} \ge 0$.
- For $k = 0$ (U_{-1}, L_0): $(-1 + 1)(2^{-\beta}(\delta + 1)) - 2^{-(\alpha-1)} \ge (0 - 1)(2^{-\beta}\delta)$. This simplifies to $-2^{-(\alpha-1)} \ge -2^{-\beta}\delta$, or $2^{-\beta}\delta \ge 2^{-(\alpha-1)}$.

The condition $2^{-\beta}\delta \ge 2^{-(\alpha-1)}$ must hold for the worst case (smallest δ). Since $D \in [1/2, 1)$, the minimum δ is $2^{\beta-1}$. $2^{-\beta}(2^{\beta-1}) \ge 2^{-(\alpha-1)} \implies 1/2 \ge 2^{-(\alpha-1)}$. This implies $\alpha - 1 \ge 1$, so we need $\alpha \ge 2$. Choosing $\alpha = 2, \beta = 2$:

$$\tau_j = \mathrm{Int}(4 \cdot t_{j,2\ldots0}.t_{j,-1}t_{j,-2}) \in \{-16, \ldots, 15\} \qquad (\tau_j \le 8R_j < \tau_j + 2) \tag{39}$$

$$\delta = \mathrm{Int}(4 \cdot 0.1d_{j,-2}) \in \{2, 3\} \qquad (\delta \le 4D < \delta + 1) \tag{40}$$

The final selection logic is a table mapping (τ_j, δ) to q_{j+1}. This table is derived from the selection intervals, adjusted for truncation error.

$$q_{j+1} = \begin{cases} 1 & \text{if } 0 \le \tau_j < 2(\delta+1) \\ 0 & \text{if } -\delta \le \tau_j \le \delta - 2 \\ -1 & \text{if } -2(\delta+1) - 2 < \tau_j \le -2 \end{cases} \tag{41}$$

3.4.2 Negative Divisor

The analysis for a negative divisor, $D \in [-1, -1/2)$, is analogous to the positive case. The primary difference is that the selection intervals from eq. (26) are inverted:

$$q_{j+1} = \begin{cases} 1 & \text{if } 2D \le 2R_j \le 0 \\ 0 & \text{if } D \le 2R_j \le -D \\ -1 & \text{if } 0 \le 2R_j \le -2D \end{cases} \tag{42}$$

The continuity condition becomes $\min(U_{k,\text{trunc}}) \ge \max(L_{k-1,\text{trunc}})$, where $U_k = (k-\rho)D$ and $L_k = (k+\rho)D$. Care must be taken with the bounds of products involving negative numbers. The analysis reveals that the same precision, $\alpha \ge 2$, is required. Choosing $\alpha = 2, \beta = 2$, the integer estimates are defined as:

$$\tau_j = \text{Int}(4 \cdot t_{j,2\ldots0}.t_{j,-1}t_{j,-2}) \in \{-16, \ldots, 15\} \qquad (\tau_j \le 8R_j < \tau_j + 2) \tag{43}$$

$$\delta = \text{Int}(4 \cdot 1.0d_{j,-2}) \in \{-4, -3\} \qquad (\delta \le 4D < \delta + 1) \tag{44}$$

The resulting selection logic is:

$$q_{j+1} = \begin{cases} 1 & \text{if } 2\delta - 2 < \tau_j \le -2 \\ 0 & \text{if } \delta + 1 \le \tau_j \le -(\delta+1) - 2 \\ -1 & \text{if } 0 \le \tau_j \le -2\delta \end{cases} \tag{45}$$

3.5 Example: Radix-4 SRT Division

For Radix-4 ($B = 4$), a common choice for the digit set is $\{-2, -1, 0, 1, 2\}$, so $a = 2$. The redundancy factor is less than one:

$$\rho = \frac{a}{B-1} = \frac{2}{4-1} = \frac{2}{3} \tag{46}$$

3.5.1 Positive Divisor

The selection intervals for $q_{j+1} \in \{-2, \ldots, 2\}$ are given by eq. (22):

$$q_{j+1} = \begin{cases} 2 & \text{if } \frac{4}{3}D \le 4R_j \le \frac{8}{3}D \\ 1 & \text{if } \frac{1}{3}D \le 4R_j \le \frac{5}{3}D \\ 0 & \text{if } -\frac{2}{3}D \le 4R_j \le \frac{2}{3}D \\ -1 & \text{if } -\frac{5}{3}D \le 4R_j \le -\frac{1}{3}D \\ -2 & \text{if } -\frac{8}{3}D \le 4R_j \le -\frac{4}{3}D \end{cases} \tag{47}$$

The implementation uses truncated estimates of the shifted residual $4R_j$ and the divisor D. The residual $4R_j$ is held in carry-save form and a truncated estimate t_j is computed with α fractional bits, introducing an error $0 \le 4R_j - t_j < 2^{-(\alpha-1)}$. The integer estimates are:

$$\tau_j = \text{Int}(2^\alpha \cdot t_{j,2\ldots0}.t_{j,-1\ldots-\alpha}) \in \left\{-2^{\alpha+2}, \ldots, 2^{\alpha+2} - 1\right\} \qquad (\tau_j \le 2^{\alpha+2}R_j < \tau_j + 2) \tag{48}$$

$$\delta = \text{Int}(2^\beta \cdot 0.1d_{j,-2\ldots-\beta}) \in \left\{2^{\beta-1}, \ldots, 2^\beta - 1\right\} \qquad (\delta \le 2^\beta D < \delta + 1) \tag{49}$$

The continuity condition $\min(U_{k-1,\text{trunc}}) \ge \max(L_{k,\text{trunc}})$ must hold for $k \in \{-1, 0, 1, 2\}$. Substituting $\rho = 2/3$ and simplifying gives the conditions:

- For $k = 2$: $\frac{1}{3} \cdot 2^{-\beta}\delta - \frac{4}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} \geq 0$.
- For $k = 1$: $\frac{1}{3} \cdot 2^{-\beta}\delta - \frac{1}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} \geq 0$.
- For $k = 0$: $\frac{1}{3} \cdot 2^{-\beta}\delta - \frac{1}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} \geq 0$.
- For $k = -1$: $\frac{1}{3} \cdot 2^{-\beta}\delta - \frac{4}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} \geq 0$.

The most restrictive conditions are for $k = 2$ and $k = -1$. We need to find α, β that satisfy these for the smallest δ. A common choice is $\alpha = 5, \beta = 4$.

$$\tau_j = \text{Int}(32 \cdot t_{j,2\ldots0}.t_{j,-1\ldots-5}) \in \{-128, \ldots, 127\} \qquad (\tau_j \leq 128R_j < \tau_j + 2) \tag{50}$$

$$\delta = \text{Int}(16 \cdot 0.1d_{j,-2\ldots-4}) \in \{8, \ldots, 15\} \qquad (\delta \leq 16D < \delta + 1) \tag{51}$$

The selection logic is a more complex table, derived from the selection intervals and precision analysis.

$$q_{j+1} = \begin{cases} 2 & \text{if } \frac{8}{3}(\delta+1) \leq \tau_j < \frac{16}{3}(\delta+1) \\ 1 & \text{if } \frac{2}{3}(\delta+1) \leq \tau_j \leq \frac{10}{3}\delta - 2 \\ 0 & \text{if } -\frac{4}{3}\delta \leq \tau_j \leq \frac{4}{3}\delta - 2 \\ -1 & \text{if } -\frac{10}{3}\delta \leq \tau_j \leq -\frac{2}{3}(\delta+1) - 2 \\ -2 & \text{if } -\frac{16}{3}(\delta+1) - 2 < \tau_j \leq -\frac{8}{3}(\delta+1) - 2 \end{cases} \tag{52}$$

3.5.2 Negative Divisor

For a negative divisor $D \in [-1, -1/2)$, the selection intervals are reversed:

$$q_{j+1} = \begin{cases} 2 & \text{if } \frac{8}{3}D \leq 4R_j \leq \frac{4}{3}D \\ 1 & \text{if } \frac{5}{3}D \leq 4R_j \leq \frac{1}{3}D \\ 0 & \text{if } \frac{2}{3}D \leq 4R_j \leq -\frac{2}{3}D \\ -1 & \text{if } -\frac{1}{3}D \leq 4R_j \leq -\frac{5}{3}D \\ -2 & \text{if } -\frac{4}{3}D \leq 4R_j \leq -\frac{8}{3}D \end{cases} \tag{53}$$

The analysis proceeds analogously to the positive divisor case, using the continuity condition $\min(U_{k,\text{trunc}}) \geq \max(L_{k-1,\text{trunc}})$. The same precision parameters ($\alpha = 5, \beta = 4$) are sufficient. The integer estimates are:

$$\tau_j = \text{Int}(32 \cdot t_{j,2\ldots0}.t_{j,-1\ldots-5}) \in \{-128, \ldots, 127\} \qquad (\tau_j \leq 128R_j < \tau_j + 2) \tag{54}$$

$$\delta = \text{Int}(16 \cdot 1.0d_{j,-2\ldots-4}) \in \{-16, \ldots, -9\} \qquad (\delta \leq 16D < \delta + 1) \tag{55}$$

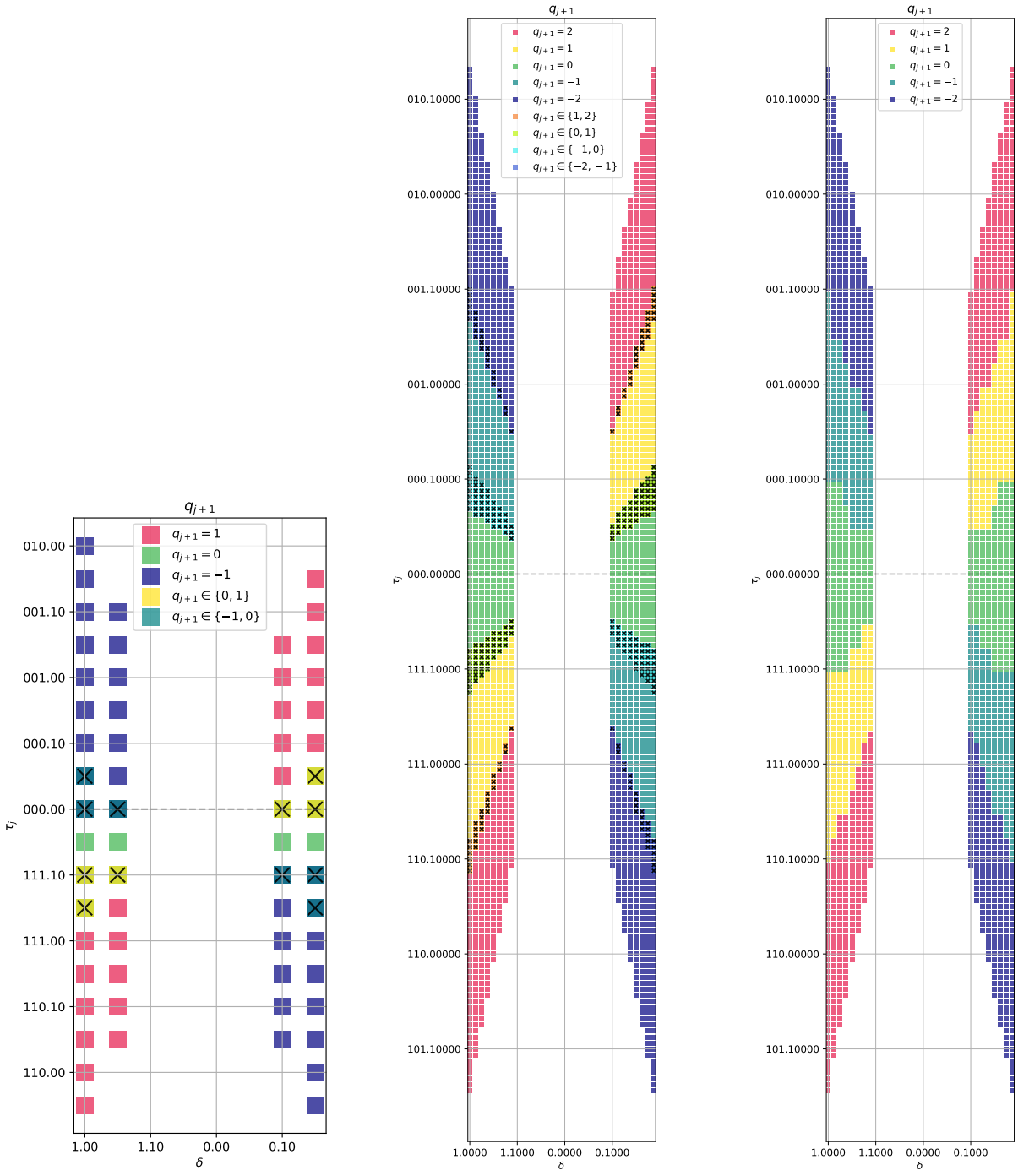
The corresponding selection logic is:

$$q_{j+1} = \begin{cases} 2 & \text{if } \frac{16}{3}\delta - 2 < \tau_j \leq \frac{8}{3}\delta - 2 \\ 1 & \text{if } \frac{10}{3}(\delta+1) \leq \tau_j \leq \frac{2}{3}\delta - 2 \\ 0 & \text{if } \frac{4}{3}(\delta+1) \leq \tau_j \leq -\frac{4}{3}(\delta+1) - 2 \\ -1 & \text{if } -\frac{2}{3}\delta \leq \tau_j \leq -\frac{10}{3}(\delta+1) - 2 \\ -2 & \text{if } -\frac{8}{3}\delta \leq \tau_j \leq -\frac{16}{3}\delta \end{cases} \tag{56}$$

4 Square Root by Digit Recurrence

The digit recurrence methodology can be adapted for square root extraction. The fundamental relationship is:

$$X = S^2 + R \tag{57}$$



(a) Radix-2 Basic QDS (b) Radix-4 Basic QDS (c) Radix-4 Optimized QDS

Figure 1: Selection regions for the Quotient Digit Selector (QDS). The y-axis is the truncated residual estimate τ_j and the x-axis is the truncated divisor estimate δ. The plots show how the choice of quotient digit q_{j+1} depends on these estimates.

where X is the radicand, S is the square root, and R is the final remainder. We again use normalized operands:

$$X \in \left[\frac{1}{4}, 1\right) \qquad \text{(Radicand)}$$

$$S \in \left[\sqrt{\frac{1}{4}}, \sqrt{1}\right) = \left[\frac{1}{2}, 1\right) \qquad \text{(Square Root)}$$

The root S is computed iteratively for $j = 0, 1, \ldots, n-1$. Let S_j be the root estimate after j digits $(s_1, \ldots, s_j)$ have been determined.

$$S_{j+1} = S_j + s_{j+1} B^{-(j+1)} \tag{58}$$

Unfolding the recurrence:

$$\text{At step } j, \quad S_{j+1} = S_0 + \sum_{i=0}^{j} s_{i+1} B^{-(i+1)} \tag{59}$$

The final root after n iterations is:

$$\text{At step } j = n-1, \quad S_n = S_0 + \sum_{i=0}^{n-1} s_{i+1} B^{-(i+1)} = S \tag{60}$$

The error at step j is $\varepsilon_{j+1} = \sqrt{X} - S_{j+1}$. As with division, this error is bounded by the maximum possible value of the remaining digits:

$$\varepsilon_{j+1} = \sum_{i=j+1}^{\infty} s_{i+1} B^{-(i+1)} \tag{61}$$

$$\left| \sum_{i=j+1}^{\infty} s_{i+1} B^{-(i+1)} \right| \leq \sum_{i=j+1}^{\infty} a B^{-(i+1)} = a B^{-(j+2)} \sum_{k=0}^{\infty} B^{-k} = a B^{-(j+2)} \frac{1}{1 - B^{-1}} = \frac{a}{B-1} B^{-(j+1)} \tag{62}$$

This gives the same error bound form as in division:

$$|\varepsilon_{j+1}| \leq \rho B^{-(j+1)} \implies \left| \sqrt{X} - S_{j+1} \right| \leq \rho B^{-(j+1)} \tag{63}$$

To find the containment condition, we rearrange and square the inequality:

$$S_{j+1} - \rho B^{-(j+1)} \leq \sqrt{X} \leq S_{j+1} + \rho B^{-(j+1)} \tag{64}$$

$$S_{j+1}^2 - 2 S_{j+1} \rho B^{-(j+1)} + \rho^2 B^{-2(j+1)} \leq X \leq S_{j+1}^2 + 2 S_{j+1} \rho B^{-(j+1)} + \rho^2 B^{-2(j+1)} \tag{65}$$

Isolating the term $X - S_{j+1}^2$:

$$\rho^2 B^{-2(j+1)} - 2 S_{j+1} \rho B^{-(j+1)} \leq X - S_{j+1}^2 \leq \rho^2 B^{-2(j+1)} + 2 S_{j+1} \rho B^{-(j+1)} \tag{66}$$

We define the scaled residual $R_{j+1} = B^{j+1}(X - S_{j+1}^2)$. Multiplying by B^{j+1}:

$$\rho^2 B^{-(j+1)} - 2 \rho S_{j+1} \leq B^{j+1}(X - S_{j+1}^2) \leq \rho^2 B^{-(j+1)} + 2 \rho S_{j+1} \tag{67}$$

This gives the containment condition for the square root residual:

$$\rho^2 B^{-(j+1)} - 2 \rho S_{j+1} \leq R_{j+1} \leq \rho^2 B^{-(j+1)} + 2 \rho S_{j+1} \tag{68}$$

The final unscaled remainder R is related to the final residual R_n by:

$$\text{At step } j = n-1, \quad R_n = B^n (X - S_n^2) = B^n R \tag{69}$$

Next, we derive the residual recurrence relation:

$$\begin{aligned}
R_{j+1} &= B^{j+1}(X - S_{j+1}^2) \\
&= B^{j+1}\left(X - \left(S_j + s_{j+1}B^{-(j+1)}\right)^2\right) \\
&= B^{j+1}\left(X - \left(S_j^2 + 2S_j s_{j+1}B^{-(j+1)} + s_{j+1}^2 B^{-2(j+1)}\right)\right) \\
&= B^{j+1}(X - S_j^2) - B^{j+1}\left(2S_j s_{j+1}B^{-(j+1)} + s_{j+1}^2 B^{-2(j+1)}\right) \\
&= B\left(B^j(X - S_j^2)\right) - \left(2S_j s_{j+1} + s_{j+1}^2 B^{-(j+1)}\right) \\
&= BR_j - s_{j+1}\left(2S_j + s_{j+1}B^{-(j+1)}\right)
\end{aligned}$$

The central recurrence for square root is:

$$R_{j+1} = BR_j - s_{j+1}\left(2S_j + s_{j+1}B^{-(j+1)}\right) \tag{70}$$

To find the selection intervals, we substitute this recurrence into the containment condition from eq. (68):

$$\rho^2 B^{-(j+1)} - 2\rho S_{j+1} \le BR_j - s_{j+1}\left(2S_j + s_{j+1}B^{-(j+1)}\right) \le \rho^2 B^{-(j+1)} + 2\rho S_{j+1} \tag{71}$$

Substituting $S_{j+1} = S_j + s_{j+1}B^{-(j+1)}$ into the bounds:

$$\rho^2 B^{-(j+1)} - 2\rho\left(S_j + s_{j+1}B^{-(j+1)}\right) \le BR_j - s_{j+1}\left(2S_j + s_{j+1}B^{-(j+1)}\right) \le \rho^2 B^{-(j+1)} + 2\rho\left(S_j + s_{j+1}B^{-(j+1)}\right) \tag{72}$$

Adding $s_{j+1}\left(2S_j + s_{j+1}B^{-(j+1)}\right)$ to all parts and simplifying leads to the selection interval for BR_j:

$$2(s_{j+1} - \rho)S_j + (s_{j+1} - \rho)^2 B^{-(j+1)} \le BR_j \le 2(s_{j+1} + \rho)S_j + (s_{j+1} + \rho)^2 B^{-(j+1)} \tag{73}$$

Unlike division, these bounds depend on the partial result S_j, which changes at every step. This makes the digit selection logic more complex.

4.1 Initialization

The first step of the algorithm ($j = 0$) requires initial values for the root approximation, S_0, and the residual, R_0. These values must be chosen such that the first shifted residual, BR_0, falls within a valid selection region for the first root digit, s_1. This ensures the recurrence can start correctly and that the subsequent residual, R_1, will satisfy the containment condition.

A common and robust initialization strategy is to set the initial root approximation to $S_0 = 1$. While this value is outside the final root's range of $S \in [1/2, 1)$, it is a valid starting point due to the self-correcting nature of the algorithm. With this choice, the initial residual is defined as:

$$R_0 = X - S_0^2 = X - 1 \tag{74}$$

Since the radicand X is in the range $\left[\frac{1}{4}, 1\right)$, the initial residual R_0 is always non-positive, with a range of $R_0 \in \left[-\frac{3}{4}, 0\right)$. A negative residual forces the selection of non-positive digits ($s_k \le 0$) until the root approximation S_k is brought down into the correct range.

To verify this initialization, we must show that the range of BR_0 is fully covered by the union of the selection intervals for s_1. The selection condition for s_1 is derived by substituting $j = 0$ and $S_0 = 1$ into the general selection interval formula from eq. (73):

$$2(s_1 - \rho) + (s_1 - \rho)^2 B^{-1} \le BR_0 \le 2(s_1 + \rho) + (s_1 + \rho)^2 B^{-1} \tag{75}$$

Since BR_0 is negative, we only need to check the intervals for non-positive digits ($s_1 \in \{0, -1, \ldots, -a\}$).

In the Radix-2 case ($B = 2, \rho = 1$), the shifted residual range is $2R_0 \in \left[-\frac{3}{2}, 0\right)$. We check the selection intervals for $s_1 = 0$ and $s_1 = -1$:

- For $s_1 = 0$: The interval is $[2(0-1) + (0-1)^2 2^{-1}, 2(0+1) + (0+1)^2 2^{-1}] = [-1.5, 2.5]$.
- For $s_1 = -1$: The interval is $[2(-1-1) + (-1-1)^2 2^{-1}, 2(-1+1) + (-1+1)^2 2^{-1}] = [-2, 0]$.

The union of these intervals covers the range $[-2, 2.5]$, which safely contains the required range of $2R_0 \in [-1.5, 0)$. Thus, the initialization is valid.

In the Radix-4 case ($B = 4, \rho = 2/3$), the shifted residual range is $4R_0 \in [-3, 0)$. We check the intervals for $s_1 \in \{0, -1, -2\}$:

- For $s_1 = 0$: The interval is $\left[2\left(0 - \frac{2}{3}\right) + \left(0 - \frac{2}{3}\right)^2 4^{-1}, 2\left(0 + \frac{2}{3}\right) + \left(0 + \frac{2}{3}\right)^2 4^{-1}\right] = \left[-\frac{4}{3} + \frac{1}{9}, \frac{4}{3} + \frac{1}{9}\right] \approx [-1.22, 1.44]$.
- For $s_1 = -1$: The interval is $\left[2\left(-1 - \frac{2}{3}\right) + \left(-1 - \frac{2}{3}\right)^2 4^{-1}, 2\left(-1 + \frac{2}{3}\right) + \left(-1 + \frac{2}{3}\right)^2 4^{-1}\right] = \left[-\frac{10}{3} + \frac{25}{36}, -\frac{2}{3} + \frac{1}{36}\right] \approx [-2.64, -0.64]$.
- For $s_1 = -2$: The interval is $\left[2\left(-2 - \frac{2}{3}\right) + \left(-2 - \frac{2}{3}\right)^2 4^{-1}, 2\left(-2 + \frac{2}{3}\right) + \left(-2 + \frac{2}{3}\right)^2 4^{-1}\right] = \left[-\frac{16}{3} + \frac{16}{9}, -\frac{8}{3} + \frac{4}{9}\right] \approx [-3.56, -2.22]$.

The union of these intervals covers the range $[-3.56, 1.44]$, which safely contains the required range of $4R_0 \in [-3, 0)$. This confirms the initialization is also valid for the Radix-4 case.

4.2 Example: Radix-2 SRT Square Root

We use $B = 2$, $a = 1$, so $\rho = 1$. The digit set for s_{j+1} is $\{-1, 0, 1\}$. The selection intervals from eq. (73) become:

$$s_{j+1} = \begin{cases} 1 & \text{if } 0 \le 2R_j \le 4S_j + 4 \cdot 2^{-(j+1)} \\ 0 & \text{if } -2S_j + 2^{-(j+1)} \le 2R_j \le 2S_j + 2^{-(j+1)} \\ -1 & \text{if } -4S_j + 4 \cdot 2^{-(j+1)} \le 2R_j \le 0 \end{cases} \tag{76}$$

The selection depends on both the shifted residual $2R_j$ and the partial root S_j. We need truncated estimates of both. The residual $2R_j = r_{j,3\ldots0}.r_{j,-1\ldots-\infty}$ is handled as before, with a truncated estimate t_j having an error $0 \le 2R_j - t_j < 2^{-(\alpha-1)}$. The integer estimates are:

$$\tau_j = \mathrm{Int}(2^\alpha \cdot t_{j,3\ldots0}.t_{j,-1\ldots-\alpha}) \in \left\{-2^{\alpha+3}, \ldots, 2^{\alpha+3} - 1\right\} \qquad (\tau_j \le 2^{\alpha+1} R_j < \tau_j + 2) \tag{77}$$

$$\sigma_j = \mathrm{Int}(2^\beta \cdot 0.1 s_{j,-2\ldots-\beta}) \in \left\{2^{\beta-1}, \ldots, 2^\beta - 1\right\} \qquad (\sigma_j \le 2^\beta S_j \le \sigma_j + 1 - 2^\beta \mathrm{ulp}) \tag{78}$$

The partial root S_j is in the range $[1/2, 1)$, and its ulp (unit in the last place) is 2^{-j}. So the bound on the estimate is $\sigma_j \le 2^\beta S_j \le \sigma_j + 1 - 2^\beta 2^{-j}$.

The continuity condition analysis is more complex than for division. Let $L_k(S_j)$ and $U_k(S_j)$ be the lower and upper bounds from eq. (73). The condition is $\min(U_{k-1,\mathrm{trunc}}) \ge \max(L_{k,\mathrm{trunc}})$. With $\rho = 1, B = 2$:

- For $k = 1$: $\min(U_{0,\mathrm{trunc}}) \ge \max(L_{1,\mathrm{trunc}})$. This gives $2 \cdot 2^{-\beta}\sigma_j - 2^{-(\alpha-1)} \ge 0$. This holds for $\alpha \ge 1$ since $\sigma_j \ge 2^{\beta-1}$.
- For $k = 0$: $\min(U_{-1,\mathrm{trunc}}) \ge \max(L_{0,\mathrm{trunc}})$. This gives $2 \cdot 2^{-\beta}\sigma_j - 2^{-(\alpha-1)} - B^{-(j+1)} \ge 0$.

The condition for $k = 0$ is more restrictive due to the $-B^{-(j+1)}$ term. For the worst case (smallest $\sigma_j = 2^{\beta-1}$), we need $2 \cdot 2^{-\beta}(2^{\beta-1}) - 2^{-(\alpha-1)} - 2^{-(j+1)} = 1 - 2^{-(\alpha-1)} - 2^{-(j+1)} \ge 0$. This requires $\alpha \ge 2$. Choosing $\alpha = 2, \beta = 2$:

$$\tau_j = \mathrm{Int}(4 \cdot t_{j,3\ldots0}.t_{j,-1}t_{j,-2}) \in \{-32, \ldots, 31\} \qquad (\tau_j \le 8R_j < \tau_j + 2) \tag{79}$$

$$\sigma_j = \mathrm{Int}(4 \cdot 0.1 s_{j,-2}) \in \{2, 3\} \qquad (\sigma_j \le 4S_j \le \sigma_j + 1 - 4 \cdot 2^{-j}) \tag{80}$$

The selection logic becomes:

$$s_{j+1} = \begin{cases} 1 & \text{if } 0 \le \tau_j < 4(\sigma_j + 1) \\ 0 & \text{if } -2(\sigma_j - 1) \le \tau_j \le 2\sigma_j - 2 \\ -1 & \text{if } -4(\sigma_j + 1) - 2 < \tau_j \le -2 \end{cases} \tag{81}$$

4.2.1 Special Case: Radicand Close to 1

A special condition arises in the initial iterations of the square root algorithm if the first few selected digits are zero. If $s_1, \ldots, s_j$ are all 0, the partial root S_j remains at its initial value, $S_j = S_0 = 1$. This simplifies the selection logic, as the partial root estimate σ_j becomes a constant. This scenario is particularly relevant for radicands X that are close to 1.

For the Radix-2 case, we use precision parameters $\alpha = 2$ and $\beta = 2$. The integer estimate of the scaled residual $2R_j$ is τ_j. Since the initial residual $R_0 = X - 1$ is negative, and the recurrence simplifies to $R_k = BR_{k-1}$ when $s_k = 0$, the residual R_j remains negative as long as only zero-digits are selected, making its estimate τ_j also negative. The integer estimate of the scaled partial root S_j is σ_j. Since $S_j = 1$, its value is constant.

$$\tau_j = \text{Int}(4 \cdot 1t_{j,2\ldots0}.t_{j,-1}t_{j,-2}) \in \{-32, \ldots, -1\} \qquad (\tau_j \leq 8R_j < \tau_j + 2) \tag{82}$$

$$\sigma_j = \text{Int}(4 \cdot 1.00) = 4 \qquad (4S_j = \sigma_j) \tag{83}$$

The selection logic becomes:

$$s_{j+1} = \begin{cases} 0 & \text{if } -2(\sigma_j - 1) \leq \tau_j \leq -1 \\ -1 & \text{if } -4\sigma_j - 2 < \tau_j \leq -2 \end{cases} \tag{84}$$

4.3 Example: Radix-4 SRT Square Root

We use $B = 4$, $a = 2$, so $\rho = 2/3$. The digit set for s_{j+1} is $\{-2, \ldots, 2\}$. The selection intervals from eq. (73) are:

$$s_{j+1} = \begin{cases} 2 & \text{if } \frac{8}{3}S_j + \frac{16}{9}4^{-(j+1)} \leq 4R_j \leq \frac{16}{3}S_j + \frac{64}{9}4^{-(j+1)} \\ 1 & \text{if } \frac{2}{3}S_j + \frac{1}{9}4^{-(j+1)} \leq 4R_j \leq \frac{10}{3}S_j + \frac{25}{9}4^{-(j+1)} \\ 0 & \text{if } -\frac{4}{3}S_j + \frac{4}{9}4^{-(j+1)} \leq 4R_j \leq \frac{4}{3}S_j + \frac{4}{9}4^{-(j+1)} \\ -1 & \text{if } -\frac{10}{3}S_j + \frac{25}{9}4^{-(j+1)} \leq 4R_j \leq -\frac{2}{3}S_j + \frac{1}{9}4^{-(j+1)} \\ -2 & \text{if } -\frac{16}{3}S_j + \frac{64}{9}4^{-(j+1)} \leq 4R_j \leq -\frac{8}{3}S_j + \frac{16}{9}4^{-(j+1)} \end{cases} \tag{85}$$

The integer estimates are:

$$\tau_j = \text{Int}(2^\alpha \cdot t_{j,3\ldots0}.t_{j,-1\ldots-\alpha}) \in \left\{-2^{\alpha+3}, \ldots, 2^{\alpha+3} - 1\right\} \qquad (\tau_j \leq 2^{\alpha+2}R_j < \tau_j + 2) \tag{86}$$

$$\sigma_j = \text{Int}(2^\beta \cdot 0.1s_{j,-2\ldots-\beta}) \in \left\{2^{\beta-1}, \ldots, 2^\beta - 1\right\} \qquad (\sigma_j \leq 2^\beta S_j \leq \sigma_j + 1 - 2^\beta 4^{-j}) \tag{87}$$

The continuity condition $\min(U_{k-1,\text{trunc}}) \geq \max(L_{k,\text{trunc}})$ must hold for $k \in \{-1, 0, 1, 2\}$. Substituting $\rho = 2/3$ and simplifying gives the conditions:

- For $k = 2$: $\frac{2}{3} \cdot 2^{-\beta}\sigma_j - \frac{8}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} \geq 0$.
- For $k = 1$: $\frac{2}{3} \cdot 2^{-\beta}\sigma_j - \frac{2}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} \geq 0$.
- For $k = 0$: $\frac{2}{3} \cdot 2^{-\beta}\sigma_j - \frac{2}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} - \frac{4}{9}B^{-(j+1)} \geq 0$.
- For $k = -1$: $\frac{2}{3} \cdot 2^{-\beta}\sigma_j - \frac{8}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} - \frac{25}{9}B^{-(j+1)} \geq 0$.

A significant complication arises in the continuity analysis for square root that is not present in division. The selection bounds in eq. (73) contain terms proportional to $B^{-(j+1)}$, originating from the $(s_{j+1} \pm \rho)^2$ components. These terms cause the overlap between selection regions to depend on the iteration number j. Consequently, the available overlap can shrink, making the digit selection more difficult. This problem is most acute for early iterations (small j) where the $B^{-(j+1)}$ factor is largest, and for negative choices of the digit k where the $(k \pm \rho)^2$ coefficients are largest. The most restrictive case is the boundary between selecting $k = -1$ and $k = -2$. For this boundary, the continuity condition becomes:

$$\frac{2}{3} \cdot 2^{-\beta}\sigma_j - \frac{8}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} - \frac{25}{9}B^{-(j+1)} \geq 0.$$

This inequality must hold for the selection logic to be valid, but the final term may cause it to fail for small values of j. For instance, with parameters $\alpha = 4, \beta = 4$, radix $B = 4$, and a worst-case partial root estimate ($\sigma_j = 8$), the constant part of the inequality is $\frac{1}{24}$. The condition simplifies to:

$$\frac{1}{24} - \frac{25}{9}4^{-(j+1)} \geq 0.$$

This is only met for $j \geq 3$. The failure for the initial iterations ($j = 0, 1, 2$) means they require a more precise selection mechanism or a special startup sequence. For $j \geq 3$, we can use the parameters $\alpha = 4, \beta = 4$:

$$\tau_j = \text{Int}(16 \cdot t_{j,3\ldots0}.t_{j,-1\ldots-4}) \in \{-128, \ldots, 127\} \qquad (\tau_j \leq 64R_j < \tau_j + 2) \tag{88}$$

$$\sigma_j = \text{Int}(16 \cdot 0.1s_{j,-2\ldots-4}) \in \{8, \ldots, 15\} \qquad (\sigma_j \leq 16S_j \leq \sigma_j + 1 - 16 \cdot 4^{-j}) \tag{89}$$

The selection logic for $j \geq 3$ is:

$$s_{j+1} = \begin{cases} 2 & \text{if } \frac{8}{3}(\sigma_j + 1) \leq \tau_j < \frac{16}{3}(\sigma_j + 1) \\ 1 & \text{if } \frac{2}{3}(\sigma_j + 1) \leq \tau_j \leq \frac{10}{3}\sigma_j - 2 \\ 0 & \text{if } -\frac{4}{3}\left(\sigma_j - \frac{1}{48}\right) \leq \tau_j \leq \frac{4}{3}\sigma_j - 2 \\ -1 & \text{if } -\frac{10}{3}\left(\sigma_j - \frac{5}{96}\right) \leq \tau_j \leq -\frac{2}{3}(\sigma_j + 1) - 2 \\ -2 & \text{if } -\frac{16}{3}(\sigma_j + 1) - 2 < \tau_j \leq -\frac{8}{3}(\sigma_j + 1) - 2 \end{cases} \tag{90}$$

4.3.1 Startup Issues and Iteration-Dependent Selection

As with the Radix-2 case, we can analyze the special condition where $s_1, \ldots, s_j = 0$, which implies $S_j = 1$. However, the analysis reveals a complication: the validity of the selection logic becomes dependent on the iteration index j.

The most restrictive continuity condition for Radix-4 square root is for selecting between $s_{j+1} = -1$ and $s_{j+1} = -2$. As derived previously, this condition is:

$$\frac{2}{3} \cdot 2^{-\beta}\sigma_j - \frac{8}{3} \cdot 2^{-\beta} - 2^{-(\alpha-1)} - \frac{25}{9}B^{-(j+1)} \geq 0$$

Let's test this inequality for our special case, substituting $B = 4$, $\alpha = 4$, $\beta = 4$, and the constant value $\sigma_j = 16$:

$$\begin{aligned} \frac{2}{3} \cdot 2^{-4}(16) - \frac{8}{3} \cdot 2^{-4} - 2^{-(4-1)} - \frac{25}{9}4^{-(j+1)} &\geq 0 \\ \frac{2}{3} - \frac{1}{6} - \frac{1}{8} - \frac{25}{9}4^{-(j+1)} &\geq 0 \\ \frac{3}{8} - \frac{25}{9}4^{-(j+1)} &\geq 0 \end{aligned}$$

This final inequality must hold for the selection logic to be valid. However, it clearly depends on j.

- For $j = 0$: $\frac{3}{8} - \frac{25}{9 \cdot 4} = \frac{3}{8} - \frac{25}{36} = \frac{27-50}{72} = -\frac{23}{72} \not\geq 0$. The condition fails.
- For $j = 1$: $\frac{3}{8} - \frac{25}{9 \cdot 16} = \frac{3}{8} - \frac{25}{144} = \frac{54-25}{144} = \frac{29}{144} \geq 0$. The condition holds.

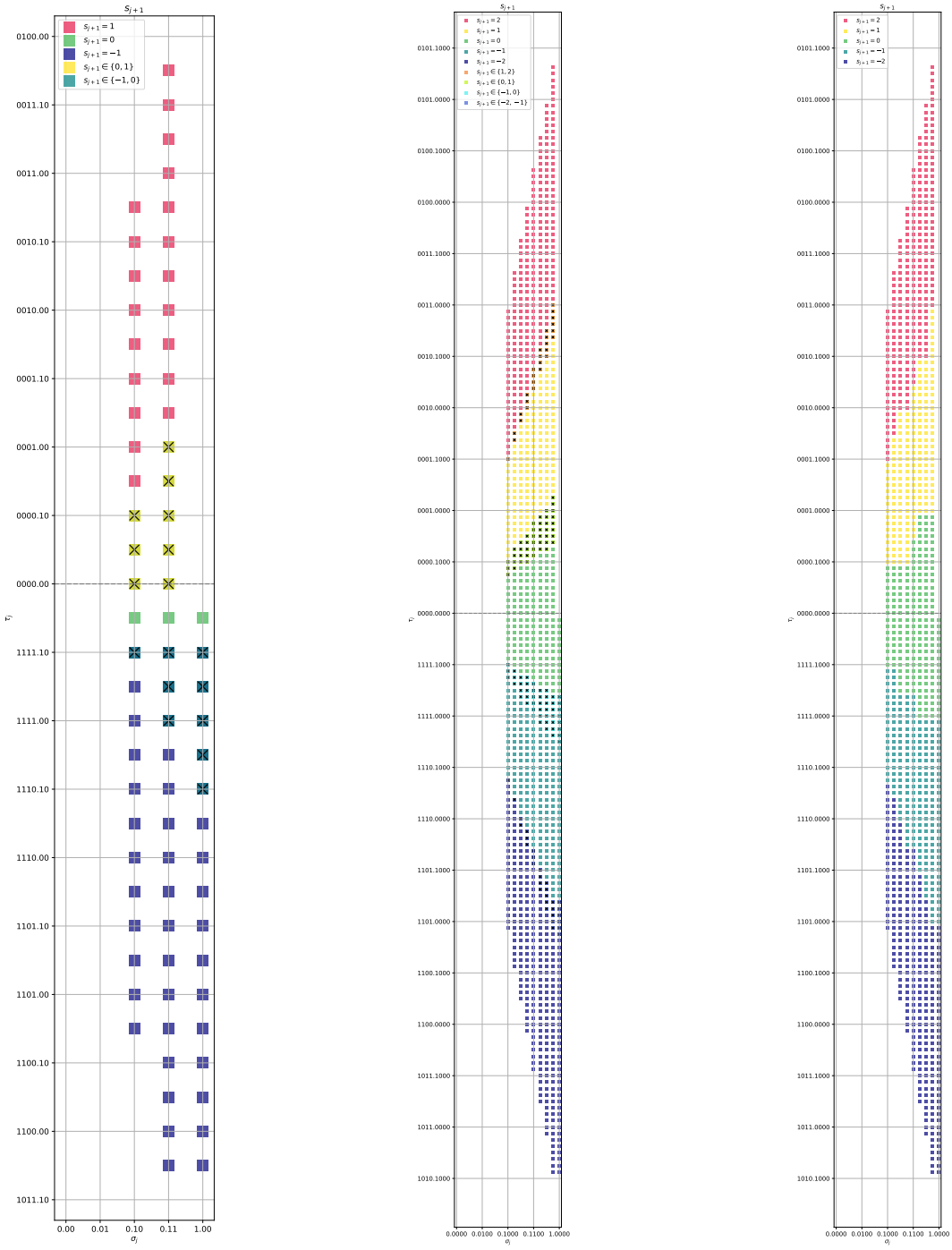
This demonstrates that the standard selection logic is not guaranteed to work for the first iteration, necessitating a special, more precise startup mechanism. For $j \geq 1$, we can use the parameters $\alpha = 4, \beta = 4$:

$$\tau_j = \text{Int}(16 \cdot 1t_{j,2\ldots0}.t_{j,-1\ldots-4}) \in \{-128, \ldots, -1\} \qquad (\tau_j \leq 64R_j < \tau_j + 2) \tag{91}$$

$$\sigma_j = \text{Int}(16 \cdot 1.0000) = 16 \qquad (16S_j = \sigma_j) \tag{92}$$

The selection logic for $j \geq 1$ is:

$$s_{j+1} = \begin{cases} 0 & \text{if } -\frac{4}{3}\left(\sigma_j - \frac{1}{3}\right) \leq \tau_j \leq -1 \\ -1 & \text{if } -\frac{10}{3}\left(\sigma_j - \frac{5}{6}\right) \leq \tau_j \leq -\frac{2}{3}\sigma_j - 2 \\ -2 & \text{if } -\frac{16}{3}\sigma_j - 2 < \tau_j \leq -\frac{8}{3}\sigma_j - 2 \end{cases} \tag{93}$$



(a) Radix-2 Basic RDS (b) Radix-4 Basic RDS (c) Radix-4 Optimized RDS

Figure 2: Selection regions for the Root Digit Selector (RDS). The y-axis is the truncated residual estimate τ_j and the x-axis is the truncated partial root estimate σ_j. The plots show how the choice of root digit s_{j+1} depends on these estimates.

Table 1: Example selection logic constants for a Radix-4 implementation. The table shows the required integer range for the truncated residual estimate τ_j to select a given digit, based on the truncated divisor/root estimate (δ or σ_j).

τ_j	$q_{j+1}, s_{j+1} = 2$	$q_{j+1}, s_{j+1} = 1$	$q_{j+1}, s_{j+1} = 0$	$q_{j+1}, s_{j+1} = -1$	$q_{j+1}, s_{j+1} = -2$
$\delta, \sigma_j = 8$	24 to 49	8 to 23	-8 to 7	-26 to -9	-50 to -27
$\delta, \sigma_j = 9$	28 to 53	8 to 27	-8 to 7	-28 to -9	-56 to -29
$\delta, \sigma_j = 10$	32 to 59	8 to 31	-12 to 7	-32 to -13	-60 to -33
$\delta, \sigma_j = 11$	32 to 65	8 to 31	-12 to 7	-34 to -13	-66 to -35
$\delta, \sigma_j = 12$	36 to 71	12 to 35	-12 to 11	-36 to -13	-72 to -37
$\delta, \sigma_j = 13$	40 to 75	16 to 39	-16 to 15	-40 to -17	-76 to -41
$\delta, \sigma_j = 14$	40 to 81	16 to 39	-16 to 15	-44 to -17	-82 to -45
$\delta, \sigma_j = 15$	48 to 85	16 to 47	-16 to 15	-48 to -17	-88 to -49

5 Conclusion

Digit recurrence algorithms provide an elegant and efficient framework for implementing division and square root in hardware. They resolve a fundamental conflict at the heart of iterative arithmetic: the need for high precision in the critical path inherently limits speed. As this tutorial has demonstrated, the core principle of redundancy elegantly resolves this conflict. By employing a redundant digit set for the result, characterized by a redundancy factor $\rho > 1/2$, these algorithms create overlapping selection regions. This overlap is the key that unlocks high performance, as it permits the critical selection of the next result digit to be based on fast, low-precision estimates of the internal state.

We have systematically derived the fundamental recurrence relations and shown how the theoretical guarantee of redundancy is translated into a practical hardware implementation. The continuity condition serves as the crucial bridge, providing a rigorous method to determine the minimum required precision for the truncated operands. This analysis ensures that the uncertainty introduced by truncation is always less than the overlap provided by redundancy, guaranteeing a correct choice can always be made. The journey from the abstract recurrence relations to the concrete selection logic tables for Radix-2 and Radix-4 illustrates a complete and robust design methodology.

The power of this methodology was shown to extend from the relatively straightforward case of division, with its static divisor D, to the more intricate square root operation. The latter's reliance on a dynamic, evolving partial root S_j introduces greater complexity, particularly for higher-radix designs. As we saw in the Radix-4 square root example, iteration-dependent terms can shrink the available overlap, demanding higher precision or special startup logic. This highlights the inherent trade-offs between performance (radix), complexity, and area that designers must navigate.

Ultimately, digit recurrence methods exemplify a powerful design paradigm where algorithmic properties are exploited to optimize hardware performance. By trading the simplicity of a non-redundant number system for the flexibility of redundancy, they break the performance bottleneck of traditional iterative approaches. They are not merely a topic of academic interest but form the foundation of essential, high-performance arithmetic units in virtually all modern processors, standing as a cornerstone of practical computer arithmetic.