Digit Stability Inference for Iterative Methods Using Redundant Number Representation

He Li, *Student Member, IEEE*, Ian McInerney, *Student Member, IEEE*, James J. Davis, *Member, IEEE*, and George A. Constantinides, *Senior Member, IEEE*

Abstract—In our recent work on iterative computation in hardware, we showed that arbitrary-precision solvers can perform more favorably than their traditional arithmetic equivalents when the latter's precisions are either under- or over-budgeted for the solution of the problem at hand. Significant proportions of these performance improvements stem from the ability to infer the existence of identical most-significant digits between iterations. This technique uses properties of algorithms operating on redundantly represented numbers to allow the generation of those digits to be skipped, increasing efficiency. It is unable, however, to guarantee that digits will stabilize, i.e., never change in any future iteration. In this article, we address this shortcoming, using interval and forward error analyses to prove that digits of high significance will become stable when computing the approximants of systems of linear equations using stationary iterative methods. We formalize the relationship between matrix conditioning and the rate of growth in most-significant digit stability, using this information to converge to our desired results more quickly. Versus our previous work, an exemplary hardware realization of this new technique achieves an up-to 2.2× speedup in the solution of a set of variously conditioned systems using the Jacobi method.

Index Terms—Digit stability, stationary iterative methods, redundant number representation, arbitrary-precision computation.

1 Introduction & Motivation

Many scientific, optimization, and machine learning applications require the solution of systems of linear equations [1]. Stationary iterative methods such as Gauss-Seidel, Jacobi, and successive over-relaxation are popular ways to convert such an N-dimensional system, $\boldsymbol{Ax} = \boldsymbol{b}$, into a linear fixed-point iteration. These all take the form $\boldsymbol{x}^{(k+1)} = f\left(\boldsymbol{x}^{(k)}\right)$, where $f : \mathbb{R}^N \to \mathbb{R}^N$ is a computable real function.

When solving such systems conventionally, cases arise where low-magnitude perturbations cause large numbers of digits to change between iterations via carry propagation. Fig. 1a exemplifies this for the toy iteration

$$x^{(k+1)} = 5/4 - 1/4 \cdot x^{(k)}$$

computed from $x^{(0)} = 0$ with nonredundant radix-10 number representation. Here, the method causes oscillations in approximants around the true result, $x^* = 1$. Although the absolute algorithm residue $\left|x^{(k)} - x^*\right|$ decreases monotonically as $k \to \infty$, digits across approximants never stabilize.

By introducing redundancy into our number representation, we can prevent the occurrence of this scenario. Fig. 1b shows the same example, also radix-10, but now with digits able to be selected from the *redundant digit set* $\{-9, -8, \cdots, 8, 9\}$. Here, less-significant digits (LSDs) can be used to correct errors that were previously introduced by digits of higher significance, further allowing those most-significant digits (MSDs) to be declared stable. The computation of these digits—shown in gray—can thus be avoided, increasing computational efficiency.

- *The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, United Kingdom. E-mail: {h.li16, i.mcinerney17, james.davis, g.constantinides}@imperial.ac.uk.*

$x^{(1)}$:	1.2500000	1.2500000
$x^{(2)}$:	1.0156255	1.0156255
$x^{(3)}$:	0.9960933	$1.00\overline{4}1\overline{1}3$
$x^{(4)}$:	1.0009766	$1.000\overline{3}106$
$x^{(5)}$:	0.9997555	$1.0000\overline{3}55$
$x^{(6)}$:	1.0000611	1.0000061
	(a) Nonredundant form.	(b) Redundant form.

Fig. 1. Approximants of toy iteration $x^{(k+1)} = 5/4 - 1/4 \cdot x^{(k)}$ with both nonredundant and redundant radix-10 number representation. In (b), digits in gray are known to have stabilized, and thus do not need to be recomputed. Digits with over-bars represent negative values: $\overline{i} = -i$.

In our previous work, ARCHITECT, we introduced the first hardware architecture capable of computing solutions of systems of linear equations to arbitrary accuracy [2], [3], [4]. ARCHITECT uses redundant number representation to allow approximants to be computed from MSD first with *online arithmetic*, enabling earlier approximants to be refined as needed. With the knowledge that some D MSDs are common to approximants k and $k+1$, ARCHITECT is able to deduce the number that will also appear in approximant $k+2$. Since that number is always smaller than D, however, this technique is unfortunately unable to infer digit stability.

Also using online arithmetic, Ercegovac's E-method produces the digits of its results from MSD first, one more per iteration [5]. As exemplified in Fig. 2, this technique therefore enables the inference of digit stability. The E-method, however, is a specialized Jacobi iteration and imposes strict conditions on its inputs: particularly a well conditioned $\boldsymbol{A}$.

In this article, we revisit ARCHITECT's MSD elision, combining knowledge of MSDs shared between successive

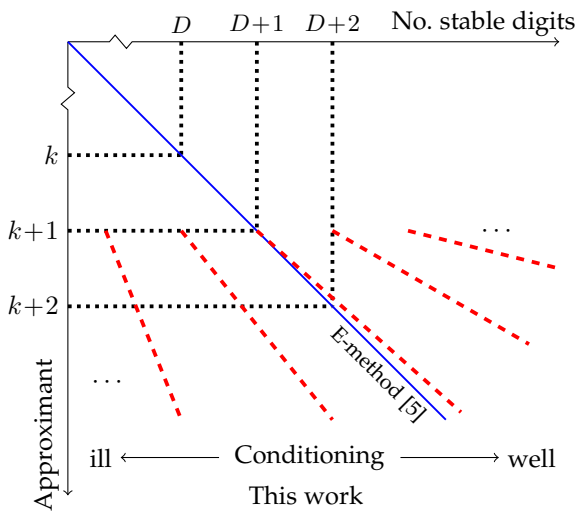

Fig. 2. A sketch of guaranteed digit stability. The E-method produces one new digit of lower significance per iteration; these, whose boundary is represented by the solid blue line, therefore remain stable across all future approximants. With knowledge that approximants k and $k+1$ share D identical MSDs, our technique is able to infer the numbers of stable digits within the $k+1$th and all future approximants. As shown by the dashed red lines, these are dependent upon the conditioning of $\boldsymbol{A}$.

approximants with matrix conditioning to infer digit stability. In contrast to the E-method, this work is applicable to any stationary iterative method and, as also shown in Fig. 2, holds for both well and ill-conditioned $\boldsymbol{A}$. With particularly well conditioned matrices, we can predict the generation of more than one stable digit per iteration.

We make the following novel contributions in this article:

- Using interval and forward error analyses, a theorem for the rate of stable MSD growth within the approximants produced by any stationary iterative method.
- Theoretical comparison of our proposal versus existing methods allowing the skipping of MSD calculation.
- An exemplary hardware implementation of our proposal using the Jacobi method.
- Empirical performance comparisons against the state-of-the-art arbitrary-precision iterative solver. For the solution of a set of representative linear equations, we achieve speedups of 2.0–2.2× over this prior work.

2 Background

2.1 Redundant Number Representation

In a redundant number system, the representation associated with a value is not unique, i.e., the same value can be encoded in more than one way. The most widely used of the redundant systems is the class of *symmetric signed-digit* number representations [6], originally conceived for the purpose of performing carry-free addition [7]. Here, digits can take any value from within the set

$$S \coloneqq \{-\gamma, -\gamma+1, \cdots, \gamma-1, \gamma\},$$

where $r/2 \leq \gamma \leq r-1$ for some radix r. When $\gamma = r-1$, the digit set S is said to be *maximally redundant*. Henceforth, we assume the use of a symmetric, maximally redundant digit set, and our example implementation uses radix-2 number representation of this form. Our results could be extended to asymmetric ($\min S \neq -\max S$) and non-maximally redundant digit sets if required.

2.2 Hardware Applications of Redundancy

The performance of many custom hardware systems is predominantly dependent upon the speed of their underlying arithmetic operators [8]. When these employ conventional, nonredundant number representations, carry propagation is often the primary factor determining their latency. The introduction of redundancy, however, often allows execution times to be shortened due to the reduction—and sometimes complete elimination—of carry chains [9]. Interest in the acceleration of arithmetic circuits using redundant number systems is growing. Signed-digit representations have been used within high-radix adders [10] and dividers [11], and constant-vector multipliers [12], to improve performance and reduce power consumption versus their conventional equivalents. Fast multipliers using signed-digit representation during their partial product generation [13] and reduction [14] steps have also been proposed.

Similarly to the E-method [5], the work we describe herein uses redundancy in order to infer digit stability within iterative algorithms. In contrast to that technique, however, ours is less restrictive and more widely applicable.

2.3 Online Arithmetic Essentials

The *de facto* standard for MSD-first calculation is Ercegovac's *online arithmetic* [15]. An important characteristic of online operators is that of *online delay*, typically denoted δ. Classical digit-serial online operators produce output digits at the same rate as they consume them, but delayed by a fixed number of digits: δ. When operators are chained to form a datapath, its overall online delay is the summation of operators' delays through the longest path [16].

Online delay is typically considered to be a limitation in terms of throughput, thus effort has been made to reduce it through the use of composite online functions [17], [18], multioperand operators [19], [20], and high radices [21]. For ARCHITECT, we showed that online delay could also be used to infer the presence of identical MSDs within iterative computations [3], [4]. Since datapaths composed of online operators compute from MSD first and outputs begin to be generated δ digits after input digits are consumed, an output's first D digits are wholly dependent upon its inputs' first $D+\delta$ digits. Since iterative methods' inputs are its previously generated outputs, this allows us to guarantee that, if approximants k and $k+1$ are equal in their first D MSDs, approximant $k+2$ will have $D-\delta$ MSDs in common with both when computed using online arithmetic. In this article, we prove that MSDs can be declared identical not just for *one* approximant, but across *all* future approximants.

2.4 Stationary Iterative Methods

In numerical linear algebra, a straightforward way to solve a system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is to transform it into a linear fixed-point iteration of the form

$$\boldsymbol{M}\boldsymbol{x}^{(k+1)} = \boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{b} \tag{1}$$

with $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$ and $\boldsymbol{M}$ non-singular [22][1]. Defining iteration matrix $\boldsymbol{G} = \boldsymbol{M}^{-1}\boldsymbol{N}$, (1) can also be written as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{G}\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b}. \tag{2}$$

Achievement of convergence requires that $\boldsymbol{G}$'s spectral radius $\rho(\boldsymbol{G}) < 1$. Such *stationary iterative methods* are widely used in the approximate solution of nonlinear [24], differential [25], and integral equations [26]. They also play a significant role in multigrid theory; multigrid methods commonly serve as preconditioners for many other iterative algorithms [27]. For scenarios in which high-precision results are required, mixed-precision methods enabling performant and efficient implementation have been proposed [28]. In contrast to standard approaches, we adopt an MSD-first arbitrary-precision computation paradigm enabling iterative refinement limited only by memory capacity [2].

The work we present in this article applies to any method of the form in (1). While we use stationary iterative methods as accessible examples for our analysis, our proposal could be extended to nonlinear fixed-point iterations.

3 Preliminaries & Notation

For the remainder of this article, we assume the use of a fixed-point radix-r symmetric signed-digit number representation system with maximal redundancy.

A scalar is denoted by a normal symbol x. For convenience, we assume that all redundantly represented numbers have $|x| < 1$ and can be expressed as $x = \sum_{i=1}^{D} x_i r^{-i}$, where x_i is the ith MSD of D-digit x.

A vector is represented by a bold symbol $\boldsymbol{x}$, with its jth element denoted x_j. Where a vector is composed of signed-digit numbers, x_{ji} is the ith MSD of the jth element of $\boldsymbol{x}$.

An approximant of an iterative method at iteration $k \in \mathbb{N}_{>0}$ is denoted $\boldsymbol{x}^{(k)}$, while its exact result is $\boldsymbol{x}^*$. The residue of an iterative method at iteration k is $\boldsymbol{s}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^*$.

A matrix is represented by a bold capital symbol $\boldsymbol{X}$, and the p-norm of either a matrix or a vector is given by $\|\bullet\|_p$.

4 Digit Stability Inference

Assume that a stationary iterative method is used to solve a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Further assume that the inequality $\|\boldsymbol{G}\|_\infty < 1$ holds[2]. If approximants to $\boldsymbol{x}^*$ are vectors with digits selected from a symmetric maximally redundant signed-digit set, knowledge of the number of identical MSDs in any two successive approximants $\hat{k} - 1$ and $\hat{k}$ allows us to declare that subsets of MSDs in all approximants $k \geq \hat{k}$ will never change. The key steps in the derivation that follows are:

1) **Lemma 2**: If it is known that D MSDs of successive approximants' elements are identical, we can bound the magnitude of the algorithm residue based on D and $\boldsymbol{G}$.
2) **Lemma 3**: Given a particular residue bound, we prove that a quantity of the current and future approximants' MSDs can never change.
3) **Theorem 1**: Bringing Lemmas 2 and 3 together, we infer the minimum number of permanently identical MSDs per approximant based on D and $\boldsymbol{G}$.

Let us begin by formally defining the meaning of digit stability within the approximants of an iterative algorithm.

Definition 1 (Digit stability)**.** *The D MSDs of an approximant $\hat{k}$ are said to be stable iff*

$$x_i^{(k)} = x_i^{\left(\hat{k}\right)} \quad \forall k > \hat{k}\ \forall i \in \{1, 2, \cdots, D\}.$$

Our choice of number system means that we can append digits to a number x to form a new number, $\tilde{x}$, representing any value within a symmetric interval around x. We call such numbers *consistent* in the values they represent.

Definition 2 (Digit consistency)**.** *Let x be a number composed of D digits selected from a symmetric maximally redundant signed-digit set. Further let y be a second number, similarly constructed, comprising any finite number of digits. y is said to be consistent with x iff*

$$y \in \left(x - r^{-D},\ x + r^{-D}\right).$$

Lemma 1 (Representation interval)**.** *Let x be a D-digit number. If additional digits are appended to x to form a new number, $\tilde{x}$, then $\tilde{x}$ is consistent with x.*

Proof. By definition,

$$x = \sum_{i=1}^{D} x_i r^{-i}.$$

Since $\tilde{x}$ contains $\tilde{D} > D$ digits, with its D MSDs the same as those in x,

$$\begin{aligned}\tilde{x} &= \sum_{i=1}^{D} x_i r^{-i} + \sum_{i=D+1}^{\tilde{D}} \tilde{x}_i r^{-i} \\ &= x + \sum_{i=D+1}^{\tilde{D}} \tilde{x}_i r^{-i}.\end{aligned}$$

The digit extrema in our number system are $-(r-1)$ and $r-1$. We can thus deduce that

$$\begin{aligned}\tilde{x} &\in \left[x - \sum_{i=D+1}^{\tilde{D}} (r-1)\, r^{-i},\ x + \sum_{i=D+1}^{\tilde{D}} (r-1)\, r^{-i}\right] \\ &= \left[x - r^{-D} + r^{-\tilde{D}},\ x + r^{-D} - r^{-\tilde{D}}\right] \\ &\subset \left(x - r^{-D},\ x + r^{-D}\right),\end{aligned}$$

and so, per Definition 2, $\tilde{x}$ is consistent with x. □

Suppose now that we know—via runtime digit-by-digit comparison—that some D MSDs within successive approximants k and $k+1$ are identical. Given particular iteration matrix conditioning, we can bound the algorithm residue for approximant $k+1$.

1. If desired, explicit preconditioning can be applied by substituting a preconditioning matrix $\boldsymbol{R}$ for $\boldsymbol{M}$, resulting in an alternative stationary iterative method [23]. Implicit preconditioning is beyond the scope of this article, but could be incorporated if it can be expressed in terms of standard online arithmetic operators [5].

2. We adopt the infinity-norm in the analysis that follows since digit stability is ensured through bounds on worst-case perturbations of $\boldsymbol{x}^{(k)}$. In the common case of Hermitian matrices, $\rho(\boldsymbol{G}) = \|\boldsymbol{G}\|_2$ and $\|\boldsymbol{G}\|_2 \leq \|\boldsymbol{G}\|_\infty$, thus a bound on $\|\boldsymbol{G}\|_\infty$ corresponds to a bound on $\rho(\boldsymbol{G})$ [29]. Finally note that, although we present our analysis in a general setting, its application is intended for methods where $\|\boldsymbol{G}\|_\infty$ is readily computable, such as Jacobi.

Lemma 2 (Residue bound)**.** *If the elements of $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(k+1)}$ share a minimum of D identical MSDs, then*

$$\left\|\boldsymbol{s}^{(k+1)}\right\|_\infty < \frac{2\left\|\boldsymbol{G}\right\|_\infty}{1-\left\|\boldsymbol{G}\right\|_\infty} r^{-D}.$$

Proof. Manipulation of (1) allows us to deduce that

$$\begin{aligned}\boldsymbol{M}\left(\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\right) &= (\boldsymbol{N}-\boldsymbol{M})\,\boldsymbol{x}^{(k)}+\boldsymbol{A}\boldsymbol{x}^{*}\\ \boldsymbol{A}\boldsymbol{s}^{(k)} &= \boldsymbol{M}\left(\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k+1)}\right).\end{aligned}$$

Given that $\boldsymbol{A}^{-1}=\sum_{i=0}^{\infty}\boldsymbol{G}^i\boldsymbol{M}^{-1}$ [3], we therefore have

$$\boldsymbol{s}^{(k)}=\sum_{i=0}^{\infty}\boldsymbol{G}^i\left(\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k+1)}\right).$$

Taking norms and recalling that $\left\|\boldsymbol{G}\right\|_\infty<1$,

$$\begin{aligned}\left\|\boldsymbol{s}^{(k)}\right\|_\infty &\le \left\|\sum_{i=0}^{\infty}\boldsymbol{G}^i\right\|_\infty\left\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k+1)}\right\|_\infty\\ &\le \frac{1}{1-\left\|\boldsymbol{G}\right\|_\infty}\left\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k+1)}\right\|_\infty. \qquad (3)\end{aligned}$$

Let j be the index for which $x_j^{(k)}$ and $x_j^{(k+1)}$ are the successive elements sharing the fewest identical MSDs. We define the number of contiguous MSDs shared by the jth elements as D. From Lemma 1 we know that

$$x_j^{(k)}\in\left(\sum_{i=1}^{D}x_{ji}^{(k)}r^{-i}-r^{-D},\ \sum_{i=1}^{D}x_{ji}^{(k)}r^{-i}+r^{-D}\right)$$

and

$$x_j^{(k+1)}\in\left(\sum_{i=1}^{D}x_{ji}^{(k+1)}r^{-i}-r^{-D},\ \sum_{i=1}^{D}x_{ji}^{(k+1)}r^{-i}+r^{-D}\right).$$

Since $x_{ji}^{(k)}=x_{ji}^{(k+1)}\ \forall i\in\{1,2,\cdots,D\}$, we find that

$$\left|x_j^{(k)}-x_j^{(k+1)}\right|<2r^{-D},$$

giving a bound on the vector norm of

$$\left\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k+1)}\right\|_\infty<2r^{-D}. \qquad (4)$$

Transformation of (2) reveals that

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \boldsymbol{G}\boldsymbol{x}^{(k)}+\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x}^{*}\\ &= \boldsymbol{G}\boldsymbol{x}^{(k)}+(\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{x}^{*}\\ \boldsymbol{s}^{(k+1)} &= \boldsymbol{G}\boldsymbol{s}^{(k)}.\end{aligned}$$

Taking norms,

$$\left\|\boldsymbol{s}^{(k+1)}\right\|_\infty\le\left\|\boldsymbol{G}\right\|_\infty\left\|\boldsymbol{s}^{(k)}\right\|_\infty, \qquad (5)$$

which, when combined with (3), results in

$$\left\|\boldsymbol{s}^{(k+1)}\right\|_\infty\le\frac{\left\|\boldsymbol{G}\right\|_\infty}{1-\left\|\boldsymbol{G}\right\|_\infty}\left\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k+1)}\right\|_\infty.$$

Substitution of (4) then gives

$$\left\|\boldsymbol{s}^{(k+1)}\right\|_\infty<\frac{2\left\|\boldsymbol{G}\right\|_\infty}{1-\left\|\boldsymbol{G}\right\|_\infty}r^{-D}. \qquad (6)$$

□

Given a particular residue bound, our next task is to show that we can guarantee MSD stability within the current and future approximants.

Lemma 3 (Existence of digit stability)**.** *If the condition*

$$\left\|\boldsymbol{s}^{(\hat{k})}\right\|_\infty<r^{-D} \qquad (7)$$

holds, then x_j^* is consistent with the $D-1$ MSDs of $x_j^{(k)}\ \forall k\ge\hat{k}\ \forall j$, and these MSDs are stable.

Proof. Convergence results on the algorithm ensure that there must exist an approximant $\hat{k}$ for which

$$x_j^*\in\left(x_j^{(\hat{k})}-r^{-D},\ x_j^{(\hat{k})}+r^{-D}\right)\quad\forall j.$$

From Lemma 1, we know that x_j^* is consistent with $x_j^{(\hat{k})}$.

Through repeated self-substitution of (5),

$$\left\|\boldsymbol{s}^{(k)}\right\|_\infty\le\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}\left\|\boldsymbol{s}^{(\hat{k})}\right\|_\infty \qquad (8)$$

which, given (7), means that

$$\left\|\boldsymbol{s}^{(k)}\right\|_\infty<\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D}$$

and thus

$$\left|s_j^{(k)}\right|<\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D}\quad\forall j.$$

For approximant k, therefore,

$$x_j^*\in I\coloneqq\left(x_j^{(k)}-\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D},\ x_j^{(k)}+\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D}\right)\quad\forall j.$$

Let us consider how the perturbation of one or more of the $D-1$ MSDs in any approximant $k\ge\hat{k}$ would affect algorithmic convergence. Such a perturbation would produce a new interval, I'. If $x_j^*\notin I'$, such a new representation of x_j would be inconsistent with the proof of convergence, thus the $D-1$ MSDs of $x_j^{(k)}$ must be identical for all $k\ge\hat{k}$.

Consider an increase of the $D-1$th MSD by one unit, leading to a representation consistent with any value in

$$\begin{aligned}I'\coloneqq\Big(&x_j^{(\hat{k})}-\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D}+r^{-(D-1)},\\ &x_j^{(\hat{k})}+\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D}+r^{-(D-1)}\Big)\quad\forall j.\end{aligned}$$

Comparing the upper bound of I and the lower bound of I', we have

$$\begin{aligned}\min I'-\max I &= -2\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}r^{-D}+r^{-(D-1)}\\ &= \left(r-2\left\|\boldsymbol{G}\right\|_\infty^{k-\hat{k}}\right)r^{-D}. \qquad (9)\end{aligned}$$

Since $r\ge2$ and $\left\|\boldsymbol{G}\right\|_\infty<1$, (9) is strictly positive. This means that $I\cap I'=\emptyset$, and thus $x_j^*\notin I'$.

Clearly, a unit increase of *any* digit in $x_{ji}^{(k)}\ \forall i\in\{1,2,\cdots,D-1\}$ would lead to $I\cap I'=\emptyset$, violating the algorithm's convergence. A similar argument can be made for a unit decrease of $x_{ji}^{(k)}\ \forall i\in\{1,2,\cdots,D-1\}$. Thus, x_j^* is consistent with the $D-1$ MSDs of $x_j^{(k)}\ \forall k\ge\hat{k}\ \forall j$, and these MSDs are stable.

□

We are now able to bound the current and future iterations' residues and ensure that stable MSDs exist, but the

relationship between these two features is currently missing. Combining Lemmas 2 and 3 will allow us to establish this, thereby providing a guaranteed minimum number of stable digits for the current and all future approximants.

Theorem 1 (Inference of digit stability). *If $\boldsymbol{x}^{(\hat{k}-1)}$ and $\boldsymbol{x}^{(\hat{k})}$ share a minimum of D identical MSDs, then x_j^* is consistent with the $D + \left\lfloor \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}} \right\rfloor - 1$ MSDs of $x_j^{(k)}$ $\forall k \geq \hat{k}$ $\forall j$, and these MSDs are stable.*

Proof. Since the D MSDs of each element of approximants $\hat{k}-1$ and $\hat{k}$ are identical, we can apply Lemma 2 to approximants $\hat{k}-1$ and $\hat{k}$ to find that (6) holds for $\hat{k}$, i.e.,

$$\left\|\boldsymbol{s}^{(\hat{k})}\right\|_\infty < \frac{2\|\boldsymbol{G}\|_\infty}{1-\|\boldsymbol{G}\|_\infty} r^{-D}.$$

Substituting this inequality into (8), we can deduce that

$$\begin{aligned}\left\|\boldsymbol{s}^{(k)}\right\|_\infty &< \|\boldsymbol{G}\|_\infty^{k-\hat{k}} \frac{2\|\boldsymbol{G}\|_\infty}{1-\|\boldsymbol{G}\|_\infty} r^{-D} \\ &= r^{-\left(D+\log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}}\right)} \\ &\leq r^{-\left(D+\left\lfloor \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}} \right\rfloor\right)}.\end{aligned}$$

We can therefore apply Lemma 3 with this bound on $\left\|\boldsymbol{s}^{(\hat{k})}\right\|_\infty$, from which we are finally able to infer that x_j^* is consistent with the $D + \left\lfloor \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}} \right\rfloor - 1$ MSDs of $x_j^{(k)}$ $\forall k \geq \hat{k}$ $\forall j$, and that those MSDs are stable. □

Examination of Theorem 1 allows us to understand the shapes of the stability regions seen in Fig. 2 for different $\|\boldsymbol{G}\|_\infty$. The relationship between the number of identical MSDs within approximants $\hat{k}-1$ and $\hat{k}$ and the quantity that stabilize by approximant $\hat{k}$ is controlled by $D + \left\lfloor \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}} \right\rfloor - 1$ with $k = \hat{k}$, i.e., $D + \left\lfloor \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty} \right\rfloor - 1$. For the most well conditioned systems, i.e., those with low $\|\boldsymbol{G}\|_\infty$, $\log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty}$ is more positive, while for particularly ill-conditioned systems it is more negative. This explains the leftward and rightward shifts present in Fig. 2 for high and low values of $\|\boldsymbol{G}\|_\infty$, respectively. The point at which D identical MSDs infer the presence of D stable digits within approximant $\hat{k}$ occurs when $\|\boldsymbol{G}\|_\infty = \frac{1}{2r+1}$. Beyond $\hat{k}$, we see a linear increase in $\log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}}$, and therefore in the number of stable digits, with k. This applies even for the most ill-conditioned systems; an increasing number of digits will therefore always stabilize over time.

5 Prototype Implementation

In order to evaluate the effectiveness of our proposal, we built a hardware implementation based on our previous work, ARCHITECT [3], modified to allow the runtime inference, and subsequent avoidance of recalculation, of digits known to have stabilized. As the digits of approximant k are generated, their values are compared on-the-fly with those of previously generated approximant $k-1$, fetched from on-chip memory. Once some $D > 0$ successive MSDs are found

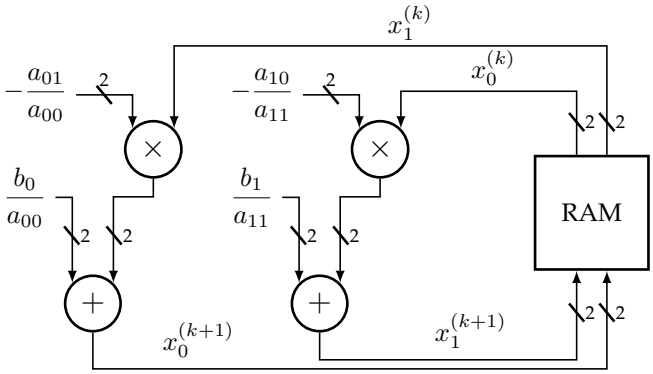

Fig. 3. Arbitrary-precision two-dimensional Jacobi method benchmark datapath [3]. Adders and multipliers are radix-2 signed-digit online operators with online delay $\delta_\times = 3$ and $\delta_+ = 2$.

to be identical across all pairs of elements $\boldsymbol{x}_j^{(k-1)}$ and $\boldsymbol{x}_j^{(k)}$, we designate $\hat{k} \leftarrow k$ and, for all subsequent approximants, the generation of each approximant's first

$$\psi^{(k)} = D + \left\lfloor \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2\|\boldsymbol{G}\|_\infty^{k-\hat{k}+1}} \right\rfloor - 1$$

digits is skipped. Note that we do not need to calculate logarithms or perform exponentiation in hardware. Instead, we can use the more computationally efficient form

$$\psi^{(k)} = D + \left\lfloor \alpha - \left(k - \hat{k} + 1\right)\beta \right\rfloor - 1, \tag{10}$$

where $\alpha = \log_r \frac{1-\|\boldsymbol{G}\|_\infty}{2}$ and $\beta = \log_r \|\boldsymbol{G}\|_\infty$ are constants that we precompute and feed in along with $\boldsymbol{A}$, $\boldsymbol{b}$, and $\boldsymbol{x}^{(0)}$.

Our prototype was a Jacobi method implementation. Jacobi iterates in the form of (1) with $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{M} = \text{diag}(\boldsymbol{A})$. As a toy example, our implementation solved linear systems with matrix size $N = 2$. Its datapath is shown in Fig. 3, and is identical in structure to that used in our previous work [3], facilitating direct comparison. Like its predecessor, this hardware is capable of arbitrary-accuracy result generation but, by virtue of the novel proposal in this article, it can do so more efficiently by skipping the calculation of MSDs known to have stabilized.

6 Evaluation

There are three obvious comparison points for our implementation: ARCHITECT with online delay-based MSD elision [3], the E-method [5], and the broad class of conventional, LSD-first iterative solvers. For the MSD-first methods, we conducted theoretical analysis (Section 6.1) to uncover the shortcomings of the prior art. We also performed experiments (Section 6.2) to quantify the gains realized through the employment of our proposal in hardware. For comparison against LSD-first arithmetic, we implemented datapaths composed of parallel-in, serial-out (PISO) operators of the same form we previously used to evaluate ARCHITECT. These operate in a similar digit-serial fashion, but require the compile-time determination of precision.

Our hardware implementations all targeted a Xilinx Virtex UltraScale field-programmable gate array (part number XCVU190-FLGB2104-3-E) and were compiled using Vi-

vado 2016.4. We verified all results obtained in hardware against golden software models written in MATLAB.

6.1 Theoretical Analysis

As was mentioned in Section 2, ARCHITECT's former MSD elision strategy is unable to infer the existence of stable digits [3]. In the worst case, as shown in Table 1, we are forced to compute the values of δ *more* MSDs for *every* approximant when using that method, potentially wasting significant time and energy in doing so. The hardware realization of the proposal in this article is actually simpler than its online delay-based predecessor, leading to the multiple performance boosts we elaborate upon in Section 6.2.1. A benefit of our previous proposal is its applicability to any iterative method. We leave the generalization of the technique we propose in this article to future work.

The E-method, designed for the efficient evaluation of polynomial and rational functions, is the only existing work allowing the declaration of MSDs as stable across the approximants of an iterative algorithm [5]. Its MSD-first Jacobi solver produces one new less-significant digit for each of the elements of its solution vector per iteration. To achieve this, the target linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ must fulfill a list of strict conditions. In particular: (i) $\|\boldsymbol{G}\|_\infty \leq {}^1\!/_{2r}$, i.e., a more restrictive requirement than strict diagonal dominance of $\boldsymbol{A}$, and (ii) $\|\boldsymbol{b}\|_\infty < 1$. (ii) is required since $\boldsymbol{b}$ forms the algorithm's initial internal residue, which must begin and remain bounded within $(-1, 1)^N$ in order to produce valid digits at each iteration.

As reflected in Table 1, our proposal is far less restrictive than the E-method. Our work holds for any stationary iterative method, while the E-method is a particular Jacobi implementation. Furthermore, we impose no restrictions upon the target system beyond $\|\boldsymbol{G}\|_\infty < 1$, meaning that users can realize the benefits of digit stability even for very poorly conditioned matrices. In order to achieve the same rate of stable MSD growth, solving (10) for $\beta = 1$ shows that our proposal requires $\|\boldsymbol{G}\|_\infty = {}^1\!/_r$: double that for the E-method. This technique is thus able to achieve the E-method's growth rate for a wider range of differently conditioned matrices. With $\|\boldsymbol{G}\|_\infty < {}^1\!/_r$, we achieve a growth rate faster than the E-method's, while the opposite is true when $\|\boldsymbol{G}\|_\infty \in ({}^1\!/_r, 1)$. An advantage of the E-method over our proposal is that the former does not require knowledge of MSDs shared between approximants; the conditions enumerated above guarantee that digits will begin to stabilize immediately. However, as we showed in our previous work, it is trivial to implement logic to detect the existence of identical MSDs in successive approximants [3].

6.2 Empirical Analysis

In order to compare the performance of our new hardware implementation (Section 5) against that of our previous work [3], we experimented with linear systems of the form

$$\boldsymbol{A}_m = \begin{pmatrix} 1 & 1-2^{-m} \\ 1-2^{-m} & 1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}, \quad \boldsymbol{x}^{(0)} = \boldsymbol{0}, \tag{11}$$

with b_0 and b_1 randomly selected from a uniform distribution in the range $[0, 1)$. We used the termination criterion $\|\boldsymbol{A}_m\boldsymbol{x} - \boldsymbol{b}\|_2 < \eta$, with $\eta \in (0, 1]$. The conditioning of $\boldsymbol{A}_m$ was controlled via $m \geq 0$, and convergence was always guaranteed since $\boldsymbol{A}_m$ is strictly diagonally dominant $\forall m$. This setup mirrored that employed in our previous work [3], enabling direct comparison.

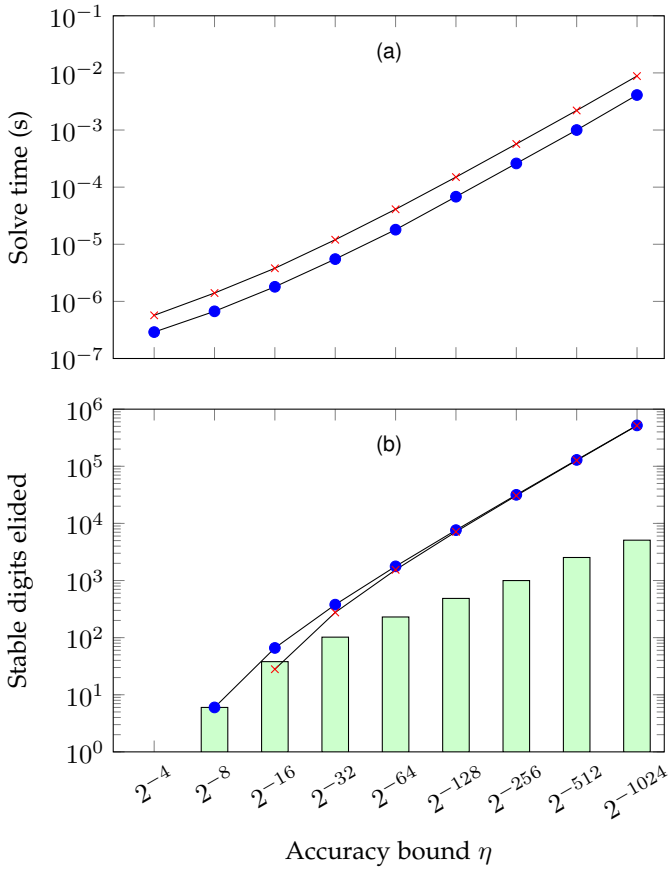

Fig. 4. How the requested accuracy bound η affects the (a) solve time and (b) number of stable digits that do not need to be calculated by ARCHITECT implementations using the MSD recalculation avoidance strategies introduced in this article (—●—) and our previous work [3] (—×—) for the solution of systems of the form in (11) with $m = 1$. The bars in (b) denote the absolute differences between the competing implementations. Points with zero elided stable digits are not visible due to (b)'s logarithmic y-axis.

6.2.1 Scalablity Comparison

In Fig. 4, we consider the scalability of arbitrary-precision two-dimensional Jacobi solvers featuring the techniques enabling the avoidance of MSD recomputation detailed in this article and our previous work [3]. For these experiments, we fixed $m = 1$ in (11) and varied accuracy bound η.

Fig. 4a shows that we achieve approximately constant solve time speedups over our previous work. Speedups ranged from 2.0× (for $\eta = 2^{-4}$) to 2.2× (2^{-1024}). The saturation is due to properties of the arbitrary-precision arithmetic operators shared by both implementations, which require an increasing number of clock cycles to generate each digit as the significance of those digits decreases [2], [4]. As η falls, the increasing time per digit generation begins to dominate the gains realized through our new proposal's MSD elision.

Fig. 4b shows that our new analysis allows us to avoid the recomputation of a mean 1.3× more MSDs than when

TABLE 1
Properties of Approaches for the Inference of Identical and Stable MSDs in Current and Future Approximants

Approach	Iterative method	Runtime detection	$\|\boldsymbol{G}\|_\infty$	$\|\boldsymbol{b}\|_\infty$	Guaranteed-stable digits in approximant $k \geq \hat{k}$	Guaranteed-identical digits between approximants k and $k+1$ $\forall k \geq \hat{k}$
Our previous work [3]	Any	✔	–	$[0, \infty)$	0	$D - \delta\left(k - \hat{k} + 1\right)$
E-method [5]	Jacobi	✘	$[0, 1/2r]$	$[0, 1)$	$D + k - \hat{k} + 1$	$D + k - \hat{k} + 1$
This work	Stationary	✔	$[0, 1)$	$[0, \infty)$	$D + \left\lfloor \alpha - \left(k - \hat{k} + 1\right)\beta \right\rfloor - 1$	$D + \left\lfloor \alpha - \left(k - \hat{k} + 1\right)\beta \right\rfloor - 1$

To enable comparison, we assume that D MSDs of all elements of the most recently computed two approximants, $\hat{k} - 1$ and $\hat{k}$, are known to be the same. For compactness, we abbreviate $\alpha = \log_r \frac{1 - \|\boldsymbol{G}\|_\infty}{2}$ and $\beta = \log_r \|\boldsymbol{G}\|_\infty$ in the final row of the table.

TABLE 2
Comparison of Iterative Solvers with MSD Elision Capability

Approach	Lookup tables	Flip-flops	Memory blocks	Max. operating frequency (MHz)
Our previous work [3]	1191	992	24	150
This work	1047	849	24	190

using the online delay-based proposal introduced in our previous work. With a very low accuracy requirement, $\eta = 2^{-4}$, neither implementation computes for long enough to allow for any MSD elision. Our new proposal becomes effective sooner than its predecessor, at $\eta = 2^{-8}$ rather than 2^{-16}, due to the former's lack of dependence on online delay δ. For our highest tested accuracy, that with $\eta = 2^{-1024}$, the difference in uncomputed MSDs was 5085 in favor of our new technique.

Along with the approximately linear increase in newly elided MSDs shown in Fig. 4b, the speedups shown in Fig. 4a were the result of logic simplifications—and consequently maximum operating frequency increases—over our former implementation. The digit generation-scheduling logic for our new implementation is more straightforward than that of its predecessor due to the latter's aforementioned dependence on δ. As shown in Table 2, the implementation we propose in this article is smaller and faster than that using our formerly proposed MSD elision approach.

6.2.2 Performance Comparison

We now show how the conditioning of $\boldsymbol{A}_m$ affects the performance of our arbitrary-precision iterative solvers compared to implementations relying on traditional LSD-first arithmetic. For the experiments reported in Fig. 5, we relaxed the constraint on m in (11) but fixed $\eta = 2^{-6}$.

In Fig. 5a, we compare our implementations against a Jacobi solver featuring LSD-first PISO arithmetic operators with a precision of 32 bits (LSD-32), a commonly encountered data width. For the solution of well conditioned linear systems, i.e., those with low m, LSD-32 is said to have *over-budgeted* precision: results take longer to compute than had a lower precision been chosen instead. As a result, both ARCHITECT-based implementations compute more quickly than LSD-32 when $m \leq 0.27$. The benefits of our new MSD elision strategy come to the fore with higher m. For

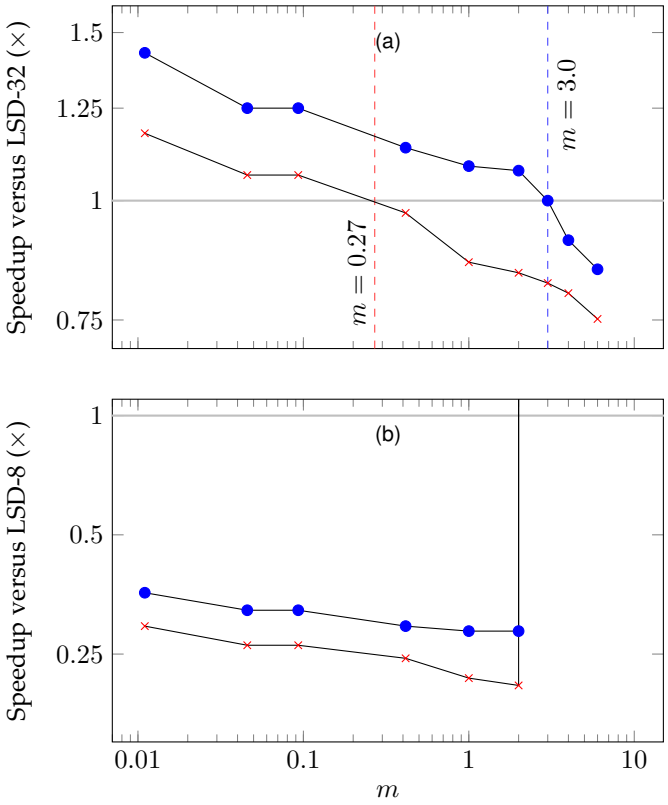

Fig. 5. How the conditioning of $\boldsymbol{A}_m$ affects the solve time of ARCHITECT implementations with MSD elision implemented per the proposal in this article (—●—) and our previous work [3] (—×—) versus LSD-first arithmetic with a fixed precision of (a) 32 and (b) 8 bits. As a result of the analysis presented herein, our new implementation computes more quickly than LSD-32 when $m \leq 3.0$, whereas our previous implementation can only beat LSD-32 when $m \leq 0.27$. (b) shows that both arbitrary-precision iterative solvers lead to an effectively infinite speedup when $m > 2$ since LSD-8 cannot ever converge to accurate-enough results. While performance slowdowns were observed for $m \leq 2$, our new proposal outperformed its predecessor in all cases, as for LSD-32.

$0.27 < m \leq 3.0$, our new implementation beats its LSD-first competitor in terms of solve time, while that presented in our previous work does not.

Fig. 5b shows the results of the same experiments as performed for Fig. 5a, but compared against an 8-bit LSD-first arithmetic implementation (LSD-8) instead. Here, high m results in ill-conditioned systems, for which LSD-8 is said to have *under-budgeted* precision. When $m > 2$, only our arbitrary-precision solvers can converge to results of great-

enough accuracy. In these cases, their performance speedups are effectively infinite. For $m \leq 2$, while both our new and prior implementations experience slowdowns versus LSD-8, the former is faster than the latter in all cases.

7 Conclusion & Future Work

In this article, we presented a theorem allowing us to predict the rate of stable MSD growth across the approximants of any stationary iterative method using maximally redundant number representation. With knowledge that some number of MSDs are common to two successive approximants, our analysis allows us to declare when, and which, MSDs in all future approximants will stabilize. The recomputation of these digits can thus be avoided, facilitating performance speedups. Unlike the E-method, this proposal holds, and is of benefit for, linear systems of any conditioning.

We demonstrated efficiency over our previous work [3] and conventional (LSD-first) arithmetic implementations using a hardware implementation of our proposal for the Jacobi method. Against the former, we achieved speedups of 2.0–2.2$\times$ for the solution of a range of representative two-dimensional linear systems. Versus the latter, we demonstrated gains in cases where LSD-first solvers have precisions either too low or too high to suit the problems at hand.

In the future, we will extend our analysis to more iterative methods, including gradient descent and Krylov subspace methods. We foresee that MSD-first stochastic gradient descent with digit stability declaration would be of particular interest to the deep learning community. We are also keen to adapt our proposal to Newton's method, for which we expect to achieve substantial performance gains due to its quadratic convergence.

Acknowledgments

The authors are grateful for the support of the United Kingdom EPSRC (grants EP/P010040/1 and EP/L016796/1), Imagination Technologies, the Royal Academy of Engineering, and the China Scholarship Council. They also wish to thank Milos D. Ercegovac for his helpful suggestions.

Supporting data for this article are available online at https://doi.org/10.5281/zenodo.3564471.

References

[1] M. A. Olshanskii and E. E. Tyrtyshnikov, *Iterative Methods for Linear Systems: Theory and Applications*. SIAM, 2014.
[2] H. Li, J. J. Davis, J. Wickerson, and G. A. Constantinides, "ARCHITECT: Arbitrary-precision constant-hardware iterative compute," in *International Conference on Field Programmable Technology*, 2017.
[3] ——, "Digit elision for arbitrary-accuracy iterative computation," in *IEEE Symposium on Computer Arithmetic*, 2018.
[4] ——, "ARCHITECT: Arbitrary-precision hardware with digit elision for efficient iterative compute," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, 2020.
[5] M. D. Ercegovac, "A general hardware-oriented method for evaluation of functions and computations in a digital computer," *IEEE Transactions on Computers*, vol. C-26, no. 7, 1977.
[6] L. Mi, *Arithmetic and Logic in Computer Systems*. Wiley, 2004.
[7] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, 1961.
[8] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y. K. Cheung, and G. A. Constantinides, "Deep neural network approximation for custom hardware: Where we've been, where we're going," *ACM Computing Surveys*, vol. 52, no. 2, 2019.
[9] P. K. Meher and T. Stouraitis, *Arithmetic Circuits for DSP Applications*. Wiley, 2017.
[10] S. Timarchi, N. Akbarzadeh, and A. A. Hamidi, "Maximally redundant high-radix signed-digit residue number system," in *CSI International Symposium on Computer Architecture and Digital Systems*, 2015.
[11] S. Amanollahi and G. Jaberipur, "Energy-efficient VLSI realization of Binary64 division with redundant number systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, 2016.
[12] C. Fan, Y. Niu, G. Shi, F. Li, F. Qi, X. Xie, and D. Jiao, "An improved signed digit representation approach for constant vector multiplication," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 10, 2016.
[13] X. Cui, W. Liu, X. Chen, E. E. Swartzlander, and F. Lombardi, "A modified partial product generator for redundant binary multipliers," *IEEE Transactions on Computers*, vol. 65, no. 4, 2015.
[14] A. Kaivani and S. Ko, "Floating-point butterfly architecture based on binary signed-digit representation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, 2015.
[15] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Elsevier, 2004.
[16] Y. Zhao, J. Wickerson, and G. A. Constantinides, "An efficient implementation of online arithmetic," in *International Conference on Field Programmable Technology*, 2016.
[17] P. Adharapurapu and M. D. Ercegovac, "A composite arithmetic scheme for evaluation of multinomials," in *Asilomar Conference on Signals, Systems, and Computers*, 2004.
[18] M. D. Ercegovac, "On digit-by-digit methods for computing certain functions," in *Asilomar Conference on Signals, Systems, and Computers*, 2007.
[19] G. B. Joseph and R. Devanathan, "Algorithms for multiplierless multiple constant multiplication in online arithmetic," *Circuits, Systems, and Signal Processing*, vol. 37, no. 11, 2018.
[20] J. Villalba, T. Lang, and J. Hormigo, "Radix-2 multioperand and multiformat streaming online addition," *IEEE Transactions on Computers*, vol. 61, no. 6, 2011.
[21] G. B. Joseph and R. Devanathan, "Design and analysis of online arithmetic operators for streaming data in FPGAs," *International Journal of Applied Engineering Research*, vol. 11, no. 3, 2016.
[22] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.
[23] D. Evans and C. Okeke, "The modified preconditioned Jacobi method for iterative solution of linear systems of equations," *International Journal of Computer Mathematics*, vol. 44, no. 1-4, 1992.
[24] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviychuk, "Non-smooth Newton methods for deformable multi-body dynamics," *ACM Transactions on Graphics*, vol. 38, no. 5, 2019.
[25] W. Koh, R. Ahmad, S. Jaaman, and J. Sulaiman, "Pricing Asian option by solving Black-Scholes PDE using Gauss-Seidel method," in *International Conference on Computing, Mathematics and Statistics*, 2019.
[26] D. Yuan and X. Zhang, "An overview of numerical methods for the first kind Fredholm integral equation," *SN Applied Sciences*, vol. 1, no. 10, 2019.
[27] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.
[28] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, and S. Tomov, "Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy," *ACM Transactions on Mathematical Software*, vol. 34, no. 4, 2008.
[29] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.