Design, Evaluation and Application of Approximate High-Radix Dividers

Linbin Chen, *Student Member IEEE,* Jie Han, *Member IEEE*, Weiqiang Liu, Senior *Member IEEE IEEE,* Paolo Montuschi *Fellow IEEE* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract— Approximate high radix dividers (HR-AXDs) are proposed and investigated in this paper. High-radix division is reviewed and inexact computing is introduced at different levels. Design parameters such as number of bits (N) and radix (r) are considered in the analysis; the replacement of exact cells with inexact cells in a binary signed-digit adder is introduced by utilizing different replacement schemes. Cell truncation and error compensation are also proposed to further extend inexact computation. Circuit-level performance and the error characteristics of the inexact high radix dividers are analyzed for the proposed designs. The combined assessment of the normal error distance, power dissipation and delay is investigated and applications of approximate high-radix dividers are treated in detail. The simulation results show that the proposed approximate dividers offer extensive saving in terms of power dissipation, circuit complexity and delay, while only incurring in a small degradation in accuracy thus making them possibly suitable and interesting to some applications and domains such as low power/mobile computing.

Index Terms— Approximate Divider, High-radix, Inexact computing, Normalized Error Distance, Power Dissipation

1 Introduction

Modern computer arithmetic applications are implemented using digital logic circuits, thus operating with a high degree of precision. However, many applications such as machine learning, multimedia and image processing can tolerate errors and imprecision in computation and still produce results that can be useful in many applications in which human senses (such as vision) are involved. The paradigm of inexact computation relies on relaxing fully precise and completely deterministic modules when for example, designing energy efficient systems; this allows imprecise computation to redirect the existing design process by taking advantage of a decrease in complexity and cost with possibly a potential increase in performance and power efficiency. Approximate (or inexact) computing relies on using this property to design simplified, yet approximate circuits operating at higher performance and/or lower power consumption compared with precise (exact) logic circuits.

Approximate computing is well suited to arithmetic circuits such as approximate adders (AXA), multipliers (AXM) and dividers (AXD). These designs fall into two categories: circuit and algorithm. For a circuit, the exact deign is modified at either transistor or gate level by assessing the benefits obtained from the approximate designs with respect to a reduced hardware complexity. Five approximate MAs (AMAs) have been obtained using logic reductions at transistor level, i.e., by removing some transistors to attain reductions in power dissipation and circuit complexity [1]. The three AXAs of[2] show attractive operational profiles in performance, hardware efficiency and power-delay product (PDP) at a good accuracy. When designing multiple-bit approximate adders, it is advisable to start from modifying the algorithm. For example, carry propagation has received considerable attention. N. Zhu etc. have proposed error-tolerant adders in which he carry propagation chain is truncated by partitioning the adder into several sub-adders [3-5]. [6] has proposed a variable latency speculative adder (VLSA) with error detection and recovery. The speculative adder of [7] reduces hardware overhead while keeping a low error rate when the operands are in 2's complement format under a Gaussian distribution; the accurate configurable adder of [8] allows adaptive operations, so either approximate, or accurate by runtime configuration. Approximate multipliers (AXM) have been also proposed; since multiplication is usually implemented by utilizing an array of adders, some of the LSBs in the partial products can be omitted [9] (while adding error compensation mechanisms), thus fewer adders can be removed in the array for faster operation. In [10], a simplified 2×2 multiplier is used as module for designing a larger multiplier; a novel design using input pre-processing and error compensation is proposed for reducing the critical path delay of a multiplier.

Approximate arithmetic division has not yet been extensively analyzed; there are increasing demands for high-speed dividers in today's floating point units (FPU) and digital signal processors for image and three-dimensional graphics applications, such as 2D image

- *Linbin Chen and Fabrizio Lombardi are with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115. E-mail: chen.lin@husky.neu.edu; lombardi@ece.neu.edu*
- *Jie Han is with the Electrical and Computer Engineering Department, University of Alberta Edmonton, AB, Canada. E-mail: jhan8@ualberta.ca.*
- *Weiqiang Liu is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: liuweiqiang@nuaa.edu.cn.*
- *Paolo Montuschi is with the Department of Control and Computer Engineering, Politecnico di Torino, 10129, Italy. E-mail: paolo.montuschi@polito.it.*

This manuscript is an extended version of the conference article listed in the bibliography as [18].

background removal, change detection [11] and graphic rendering. These applications ultimately target human vision, so they can tolerate errors and imprecision in computing and still produce meaningful results. AXD designs have been proposed by modifying the division algorithm; [12] has proposed a dynamic AXD by truncating the input operands to reduce the size of the arithmetic divider circuit, while [13, 14] has proposed a rounding-and-truncation based approximate divider by converting division into a series of shift and add operations. However, it is also possible to exploit the design space in which the original division algorithm is not fundamentally modified, but the transistor-level hardware is changed to reduce delay and power consumption.

In the technical literatures, many exact algorithms have been proposed for division [15]. Digit recurrence methods are widely used for hardware design; implementations can be either sequential, combinational, or a combination of both. The implementation of a combinational divider, is amenable to pipeline, thus would give higher throughput. Based on these findings, the combinational implementations of a divider based on restoring, non-restoring [16] and SRT algorithms [17]are treated in more detail as according to the following features:

1. In these division algorithms, computation is carried out by addition/subtraction and shifting, so only basic and simple operations are employed. These operations are also highly modularized and enable an inexact design at module (or cell) level.
2. A combinational array divider provides flexibility and granularity for an inexact design at all levels, i.e. from a cell through each computational stage up to the entire array.
3. In a fully combinational array divider, the sequential division unit consists of several computational stages, in which different design parameters can be changed and assessed while meeting often conflicting figures of merit for approximate computing.

Other methods for computing division exist in the technical literature, for example the Newton-Raphson method based on the intensive use of multiplications. However, in this paper we focus on digit-by-digit division only. Hereafter unless explicitly stated, division is intended to be digit-by-digit. Different from multiplication, a digit-by-digit division method is mostly a sequential process, while multiplication can be executed as multi-operand parallel additions (i.e. partial product generation can be parallelized). Thus, for approximate division, the impact of the sequential processing nature on accuracy must be carefully assessed; for example, when calculating the quotient, iterations introduce a cumulative error. Therefore, error propagation must be mitigated in such approximate design.

In the previous work by the same authors [18], designs of approximate radix-2 unsigned non-restoring and restoring dividers (denoted as AXDnr and AXDr) have been proposed. New approximate AXDr cells (denoted as AXDCr) have been investigated and approximate computing has been applied to division by considering different schemes by which exact cells are replaced/truncated in the array divider circuit. In [19], a high-radix division algorithm [20] and its results have been analyzed, generalized and a novel architecture has been proposed for approximate high-radix division in a similar fashion as in [18]. The original exact design in [20] has presented an algorithm based on a SRT digit-by-digit scheme for high-radix division; it employs a prescaling technique for division by digit recurrence. In the inexact design of [19], the signed-digit adder cell is simplified, while still utilizing replacement or truncation; its contribution extends also to an algorithmic domain as different numbers of bits *(N)*, radix number (*r*) and replacement/truncation depth *(d)* are also considered. While presenting a more detailed circuit schematics of the design, this paper proposes a new error compensation scheme to address the significant decrease of accuracy in [19] when a truncation scheme is utilized. Comparison to the radix-2 AXDr design is also provided to address the difference between high-radix and traditional (radix-2) dividers. Circuit-level performance and the error characteristics of the inexact high radix dividers are also analyzed and assessed both at algorithmic and circuit levels, inclusive of performance for an image processing application.

The paper is organized as follows. Section 2 gives a brief review of the high radix division algorithm with pre-scaling and the simple quotient selection logic. Section 3 introduces the proposed approximate adder cells and error compensation cell designs and application to divider design as an array-level approximation. In Section 4, the Normalized Error Distance (NED), the error distribution, power and delay are simulated to assess the performance of exact and approximate dividers. Possible trade-offs between the two metrics of NED and power are also discussed. Simulation results for different image processing applications are presented in Section 5. Section 6 concludes the paper.

2 Review

This section presents a brief review of the high radix division algorithm and the hardware implementation with prescaling by [20] as relevant to the proposed scheme .

2.1 High-radix division algorithm

In the current literature, the mostly widely used approaches for division are based either on selection tables, or on prescaling or a combination of the two [17]. In the selection table based approach approach, the digits of the quotient are obtained by inspecting the dividend and the partial remainder; according to the number of value combinations (as stored in digit selection tables), the quotient digit is obtained at each iteration [15]. However, for high radix the size of these tables increases, yielding large implementations. Among the techniques for addressing this negative design feature, prescaling (as proposed in [20-22]) is widely employed.

An efficient and unified implementation of high-radix array dividers with no lookup table for quotient digit selection using prescaling has been presented in [20]. In this algorithm, additional prescaling and conversion of the number representation are utilized; at each iteration, the quotient digit is directly obtained from the most significant (integer) part of the partial remainder. This makes possible the construction of fully combinational high-radix dividers exhibiting a latency similar to a radix-2 divider.

The high-radix division algorithm of [20] is based on the standard SRT digit-by-digit division; let the dividend and divisor be denoted by X and D, respectively (normalized as $1/2 \leq X<D< 1$). The SRT division algorithm with n-bit precision is given by the following recursive equations:

$$R^{(0)} = X,$$

$$R^{(j+1)} = rR^{(j)} - q_{j+1}D. \tag{1}$$

where $(j = 0,1,2,\dots,n-1)$, r is the radix value (usually $r = 2^m (m = 1,2,3,\dots)$), j is the iteration step, $R^{(j)}$is the partial remainder at step j, and q_j is the jth quotient digit selected from {-(r-1), … -1, 0, 1, …, r-1}. The quotient Q is given by

$$Q = \sum_{j=1}^{n} r^{-j} \cdot q_j\,, \tag{2}$$

In the design of [20], Eq. (1) is calculated using Signed-Digit (carry-free) adders based on the redundant BSD representation; so, the shifted partial remainder $rR^{(j)}$ is represented by using a BSD numbering system (denoted by the subscript SD2):

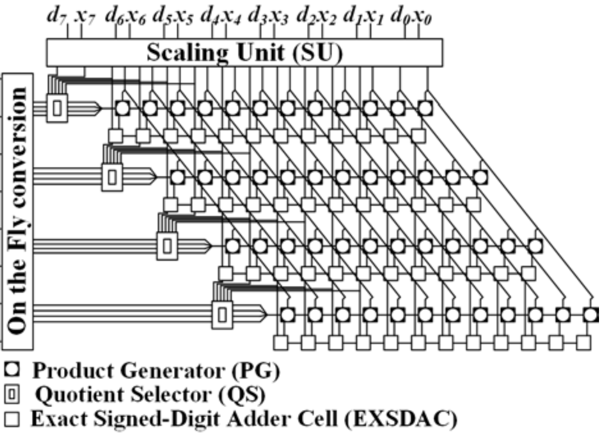

$d_7\,x_7\;d_6x_6\;d_5x_5\;d_4x_4\;d_3x_3\;d_2x_2\;d_1x_1\;d_0x_0$
Scaling Unit (SU)
On the Fly conversion
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8
Product Generator (PG)
Quotient Selector (QS)
Exact Signed-Digit Adder Cell (EXSDAC)

Fig. 2 Example of HR-EXD for 8-bit radix-4 [20]

$$rR^{(j)} = [a_{m-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-k}]_{SD2}$$
$$= \sum_{i=-k}^{m-1} 2^i a_i\,, \tag{3}$$

where $a_i \in \{-1,0,1\}$. Then, the $(j+1)^{th}$ quotient digit q_{j+1} is obtained directly from the integer part of $rR^{(j)}$ by converting the t most significant digits $a_{-1}a_{-2} \dots a_{-t}$ of the fractional part of $rR^{(j)}$ to a non-redundant form, containing only digits from the set {-1, 0} or {0, 1}, i.e.

$$rR^{(j)} = [c_{m-1} \dots c_1 c_0 . b_{-1} \dots b_{-t} a_{-t-1} \dots a_{-k}]_{SD2} \tag{4}$$

where $b_i \in \{-1,0\}$ or $\{0,1\}$, a_i, $c_i \in \{-1,0,1\}$, and

$$q_{j+1} = [c_{m-1} \dots c_1 c_0]_{SD2} = \sum_{i=-k}^{m-1} 2^i c_i, \tag{5}$$

where $c_i \in \{-1,0,1\}$ and $q_{j+1} \in \{-(r-1),\dots,0,\dots,r-1\}$.

To achieve this quotient selection method, the divisor D and the dividend X must be prescaled, so changing the divisor range from [1/2,1) to [D_{min}, 1). D_{min} is the lower bound of the scaled D and is determined by the following equation:

$$D_{min} = \frac{r-2+2^{-t}}{r-1} \leq D \leq 1, \tag{6}$$

The proof of equation (6) has been provided by T. Aoki et al in [20]. The number of iterations in the division process can be reduced by increasing the radix r, i.e. by selecting $r = 2^m$; this allows the generation of m quotient bits at each step, such that the number of steps is reduced to $\lceil n/m \rceil$.

2.2 Exact High-radix Divider (HR-EXD) Hardware Design

The implementation of a HR-EXD for the algorithm reviewed previously has been presented in [20]. The structure of an 8-bit radix-4 divider with r=4, t=2 is shown in Fig. 2; it consists of the following basic blocks.

HR-EXD Row: The hardware implementation of the HR-EXD iteration row is shown in Fig. 1; it contains a row of Product Generation (PG) modules, a row of BSD Adders (BSDA) and a Quotient Selection (QS) module. As redundant BSD representation (RBR) allows addition without propagating a carry. When compared to a non-

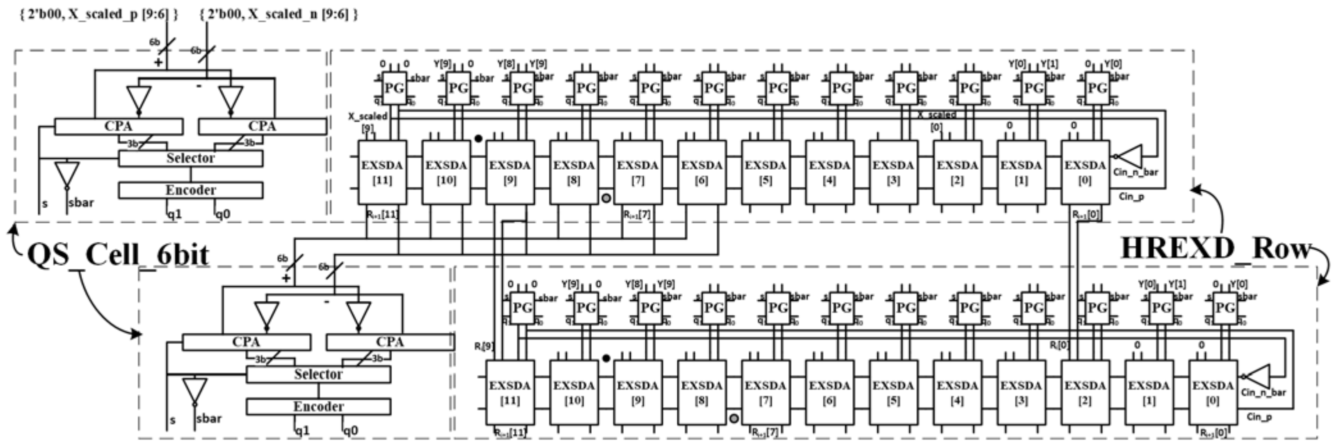


Fig. 1 HR-EXD Iteration Row Schematics

redundant representation, RBR makes digit-wise logic operations slower, however arithmetic operations are faster when a large bit width is used [17]. In a high-radix division circuit design, a Binary Signed-Digit (BSD) numbering system (as a subcategory of a RBR system [23]) with a digit set of {-1,0,1} is used to represent the partial remainder at each stage. The intermediate quotient $q_j \in [-3,+3]$ is represented in signed-magnitude form; Positive/negative flag encoding (Table 1) is used for each BSD digit.

TABLE 1 ENCODING OF BSD DIGITS

BSD Digits *d*	***d*⁺**	***d*⁻**
-1	0	1
0	1 or 1	0 or 0
1	1	0

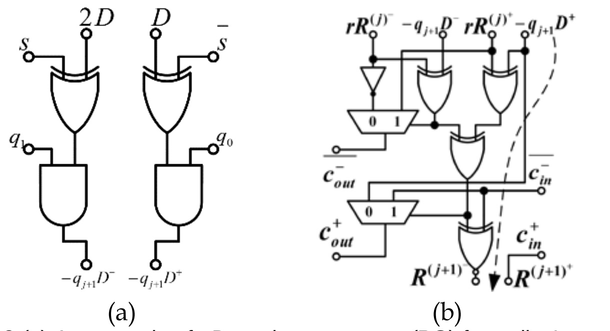


Fig. 3 (a) An example of qD product generator (PG) for radix-4 case [20] (b) Exact Signed Digit Adder Cell (EXSDAC)

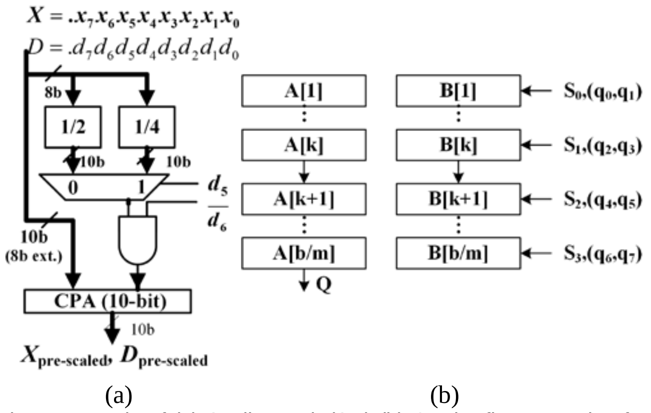


Fig. 4 Example of (a) Scaling Unit (SU) (b) On-the-fly conversion for *r*=4, *t*=2 [20]

Quotient Digit Selector (QS): As per the algorithm in [20], the real binary value of the integer part of $rR^{(j)}$ is selected as the $(j+1)^{th}$ quotient digit q_{j+1}. The implementation of this quotient selection rule does not require a digit selection table; it requires only a high-speed signed-digit carry propagation adder (SDCPA) for a limited number of bits (Fig. 1). For a higher radix, the design of the QS can be extended based on the radix-4 implementation.

Product Generator (PG): PG requires all multiples of D $(-3D,\ldots,3D)$ as pairs generated by shift operations. The circuit diagram of a PG is shown in Fig. 3(a); the PG module for radix-8 and higher can be extended from the radix-4 case.

Exact Signed Digit Adder Cell (EXSDAC): Fig. 1 (b) shows the Exact Signed Digit Adder Cell (EXSDAC); The EXSDAC remains the same for all HR-EXD irrespective of bit width and radix. The approximate designs shown next are based on the approximation of EXSDAC, i.e., the proposed approximate scheme is suitable for HR-EXD with different bit width and radix.

Scaling Unit (SU): as input operands, the divisor D and the dividend X must be prescaled prior to starting the iterations in the division. Fig. 4(a) shows the operand scaling unit using shifters and carry propagation adders (CPA). For radix-8 and higher radix values, the scaling unit does not change because it can be extended based on the radix-4 implementation.

On-the-Fly Conversion: The quotient must be converted from a signed-digit representation to a two's complement representation. This is accomplished by an addition after the quotient is fully computed; however, this addition increases the overall execution time. So to avoid this step, the on-the-fly algorithm of [24] is usually used to perform the conversion in a digit-serial fashion to generate the digits of the quotient; this technique is also utilized in this paper. The recursive equation is shown below, the on-the-fly conversion scheme is shown in Fig. 4(b).

$$A[k+1] = \begin{cases} A[k] + p_{k+1}r^{-(k+1)}, & if\ p_{k+1} \geq 0 \\ B[k] + (r - |p_{k+1}|)r^{-(k+1)}, & if\ p_{k+1} < 0 \end{cases}$$

$$B[k+1] = \begin{cases} A[k] + (p_{k+1} - 1)r^{-(k+1)}, & if\ p_{k+1} \geq 0 \\ B[k] + ((r-1) - |p_{k+1}|)r^{-(k+1)}, & if\ p_{k+1} < 0 \end{cases}.$$

3 PROPOSED APPROXIMATE DESIGNS

As shown previously in Fig. 2, the binary signed-digit adder (made of exact cells, EXSDACs) is one of the most common used computational modules of a high radix divider; hence, approximation for inexact computing must be directed to this module.

3.1 Approximate Signed-Digit Adder Cell (AXSDAC)

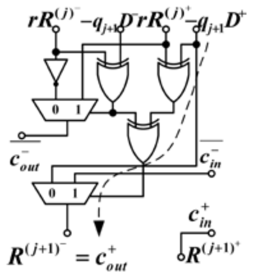


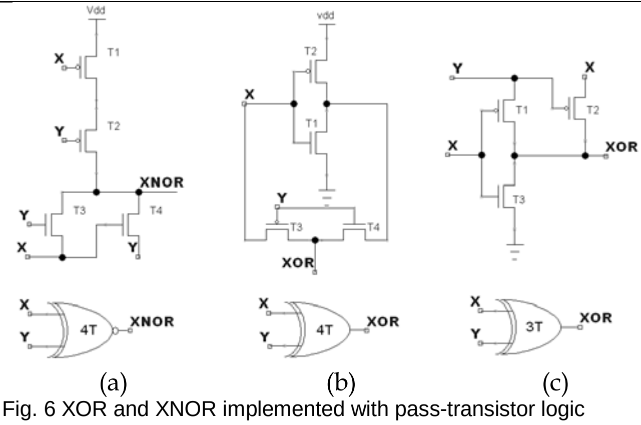
Fig. 5 Approximate design of EXSDAC (AXSDAC).

The EXSDAC computes the subtraction or addition per the quotient selection output. As $R^{(j+1)^+} = C^+_{in}$, so the input and output functions of these two signals can be ignored. The critical path of this design has a delay of 3Δ, where Δ is the unitary delay through XOR gate (arrow in Fig. 3b). In [19], approximate design of EXSDAC (denoted as AXSDAC)is proposed(Fig. 5). $R^{(j+1)^-}$ is simplified to C^+_{out} by changing the value of the other 8 outputs.

Table 2 shows the truth table of the AXSDAC; this design has therefore 8 incorrect outputs out of 32 outputs (denotes as bold in the table), so in theory its error rate is 25% (if all combinations are equally probable). A pass-transistor logic design is utilized to further decrease the circuit complexity, the delay and the power consumption; for example, the XOR and XNOR gates are shown in Fig. 6.

TABLE 2 TRUTH TABLE OF PROPOSED AXSDAC

$rR^{(j)+}$	$rR^{(j)-}$	$-q_{j+1}D^-$	$-q_{j+1}D^+$	$\overline{C_{in}^-}$	$\overline{C_{out}^-}$	C_{out}^+	$R^{(j+1)-}$	ED
0	0	0	0	1	1	0	0	0
1	0	0	0	1	1	1	1	0
0	1	0	0	1	0	1	1	0
1	1	0	0	1	1	0	0	0
0	0	1	0	1	1	1	1	0
1	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**-1**
0	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**-1**
1	1	1	0	1	1	1	1	0
0	0	0	1	1	0	1	1	0
1	0	0	1	1	1	0	0	0
0	1	0	1	1	0	0	0	0
1	1	0	1	1	0	1	1	0
0	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**-1**
1	0	1	1	1	1	1	1	0
0	1	1	1	1	0	1	1	0
1	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**-1**
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**+1**
1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0
1	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**+1**
0	0	1	0	0	1	0	0	0
1	0	1	0	0	1	1	1	0
0	1	1	0	0	0	1	1	0
1	1	1	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0
1	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**+1**
0	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**+1**
1	1	0	1	0	0	0	0	0
0	0	1	1	0	0	1	1	0
1	0	1	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0
1	1	1	1	0	0	1	1	0



(a) (b) (c)

Fig. 6 XOR and XNOR implemented with pass-transistor logic

3.2 Approximate High Radix Array Divider (HR-AXD)

1) Replacement Scheme and Truncation Scheme

When designing the approximate divider, EXSDACs are selectively replaced by AXSDACs; hence in this case, approximation is the process by which an exact cell is replaced by an approximate cell or simply truncated. Replacement and Truncation scheme are proposed in [19]. The extent by which this replacement process is performed in a divider, is quantified by the depth d, *i.e.* the number of rows (and/or columns) in the divider with approximate cells. For an N bit width of a radix 2^m(m=1, 2, 3…) divider, the number of EXSDACs is given as (N+2m)×(N/m); m EXSDACs are combined into a replacement element (RE). For example, in the 8-bit radix-4 divider, two EXSDACs are treated together as a single RE (shown in Fig. 7 by the dotted rectangles of EXSDACs). The replacement configurations and corresponding depth d for an 8-bit radix-4 array divider are shown in Fig. 7.

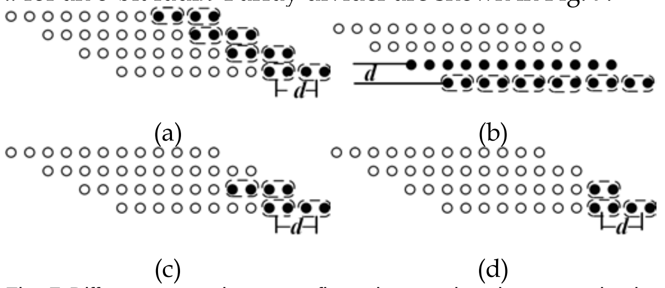


Fig. 7 Different approximate configurations and replacement depths for 8-bit radix-4 divider (a) VR d=2 (b) HR d=2 (c) SR d=2 (d) TR d=2

Four types of replacement are used for approximation in the divider design:

- *Vertical Replacement (VR):* The least significant REs in each row of the divider are replaced. So, both the remainder and the quotient show a small error distance, while taking advantage of the power-saving characteristics of the AXSDACs. The depth of the vertical replacement can be increased to further decrease the power, while tolerating more errors in the output. $m(N/m)d = Nd$ EXSDACs are replaced with AXSDACs. An example for m=2, d=2 is shown in Fig. 7(a).
- *Horizontal Replacement (HR):* In a divider, the value of the quotient is mostly related to the carry signal of each cell in a single row. For example, consider the last row corresponding to the LSB of Q; if the final value of reminder R is not of significant concern, then all EXSDACs in the last row can be replaced with AXSDACs at no significant loss of accuracy in Q. If an error can be tolerated in Q, then an increase in the depth of the horizontal replacement up to the dth LSB of Q is possible. An example of a horizontal replacement divider of depth d=2 is shown in Fig. 7(b).$(N + 2m)d$ EXSDACs are replaced with AXSDACs in an approximate divider with a horizontal replacement of depth d.
- *Square Replacement (SR):* the so-called square configuration is generated by combining the vertical and horizontal replacements. So, md^2 EXSDACs are replaced with AXSDACs; an example of a square replacement of depth d=2 is shown in Fig. 7(c).
- *Triangle Replacement (TR):* Consider the integer pair (x,y) as coordinates of individual RE in a divider. For the replacement of an exact RE(i,j) (i<d or j<d) with an

inexact RE in a triangle approximation divider with depth d (d≥1), $md(d+1)/2$ EXSDACs are replaced with AXSDACs. An example of a triangle replacement divider with d=2 is shown in Fig. 7(d).

Truncation is different from replacement, because the EXSDACs are not changed to AXSDACs, instead they are eliminated. As for replacement, four types of truncation are used as approximation in the divider design: Vertical Truncation (VT), Horizontal Truncation (HT), Squared Truncation (ST) and Triangle Truncation (TT).

Consider next circuit complexity as given by the number of transistors in an implementation; based on the different replacements, this metric can be derived as follows. Consider a radix $r = 2^m (m = 1,2,3\ldots)$ for a N-bit divider with replacement depth d; the number of adder cells is given by $(N+2m)\cdot N/m$. At a depth d, the replaced cells are given by $N\cdot d$, $(N+2m)\cdot d$, $m\cdot d^2$, and $m\cdot d\cdot(d+1)/2$ for the vertical, horizontal, square and triangle replacements. For each replaced cell, the number of transistors is reduced by 25%. The number of transistors of a PG is $3\cdot 2^m$, the number transistors of a QS is $m^2+33m+4$, the number of transistors of a SU is $24m(m+N)$, while the number of transistors for the on-the-fly conversion circuit is $4N(N+1)/m+20$; so, the total number of transistors (Z) for a HR-AXD replacement configuration is given by:

$$Z = 22(N+2m)\frac{N}{m} - \begin{Bmatrix} N\cdot d & (VR) \\ (N+2m)\cdot d & (HR) \\ m\cdot d^2 & (SR) \\ m\cdot(d+1)\cdot\frac{d}{2} & (TR) \end{Bmatrix}\cdot\frac{4Nd}{m}$$
$$+[3\cdot 2^m\cdot(N+2m)+m^2+33m+4]\frac{N}{m}$$
$$+24m(m+N)+\frac{4N(N+1)}{m}+20$$

2) *Error Compensation Scheme*

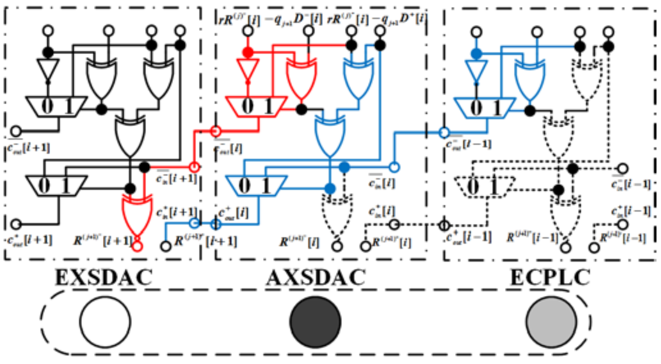


Fig. 8 Error compensation for truncation

The reduction of cells by truncation is amenable to error compensation. For a radix $r = 2^m$ divider, the most significant $(3m)$th bits of the residual $rR^{(j)}$of each adder stage are used to generate the quotient for the next stage; so, to attain a reduced inaccuracy in the quotient Q, the computation at the$(3m)$th bits of each adder stage must be done as nearly exact as possible.

Consider the three cascading EXSDACs (shown in Fig. 8)to form a cluster; assume that the left EXSDAC is at the$(3m)$th bit position of an adder stage. Using truncation, if the middle and right EXSDACs are eliminated by truncation, the $(3m)$th cell will not generate a correct output for $R^{(j+1)}$, so making the quotient inaccurate. For the $(3m)$th bit cells to compute correctly, two paths (shown in red and blue) must be preserved; these paths are dependent on the middle and right cells. As shown in Fig. 8, only some of the EXSDACs are needed for preserving correctness in these two paths; so, the circuit drawn in dotted lines can be eliminated with no impact on the computation performed by the left cell. The middle cell corresponds to the AXSDAC for replacement; the right cell is modified and it consists of only 8 transistors to compute $\overline{C_{out}}$. This circuit is hereafter referred to as the *Error Compensation Logic Cell* (ECPLC).

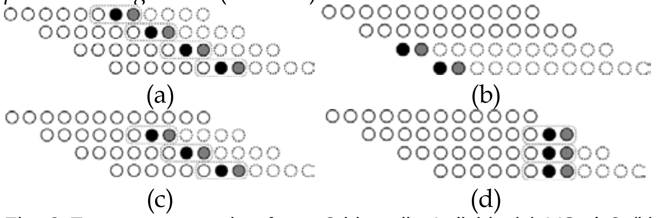


Fig. 9 Error compensation for an8-bit radix-4 divider(a) VC d=3 (b) HC d=2 (b) SC d=3 (c) TC d=3. The three clustered cells are the same as in Fig. 8.

The truncation schemes VT, ST, TT and HT can then be modified for error compensation by utilizing the AXSDAC and ECPLC; the error compensation schemes for VT, HT, ST and TT are denoted as VC, HC, SC and TC respectively. Fig. 9 shows examples of the error compensation schemes. The simultaneous utilization of a cluster consisting of these three cells (EXSDAC, AXSDAC and ECPLC) effectively results in an *error buffer* between the right located truncated cells (dotted circles) and the left located exact cells (solid circles). Therefore, the error compensation scheme is not only applied to the truncation depth that reaches the $(3m)$th bit cell at each adder stage, but it can be applied to schemes of any truncation depth. The effect of the error buffer can be clearly observed from the simulation results presented next.

4 Simulation Results

In this section, the designs of AXSDAC and approximate high-radix dividers are synthesis and simulated using Synopsis Design Compiler targeting 1GHz; predictive technology models at 45nm feature size are utilized in the simulation. For comparison purpose, the metrics of previous approximate restoring divider in [18] are also simulated and presented for HR-AXD.

4.1 NED

The Normalized Error distance (NED) is defined as the Mean Error Distance (MED) normalized by the maximum ED[25]. The maximum value of the ED is 1, so in this case the NED is equal to the MED. For HR-AXD, only the 8-bit radix-4, 12-bit radix-4 and 12-bit radix-8 dividers are presented (the trend and conclusions for these dividers are applicable also to higher radix dividers). The NED results are plotted in log scale in Fig. 11, Fig. 13 and Fig. 14 for different bit width, radix, and approximate configurations; as expected, the divider with a higher depth d has a larger NED. The horizontal configurations have the worst NED among all different bit width, radix and schemes, while the triangle configurations are the best.

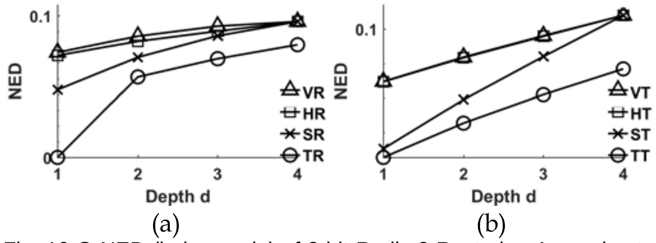


Fig. 10 Q NED (in log scale) of 8-bit Radix-2 Restoring Approximate Divider in [18] (a) Replacement (b) Truncation

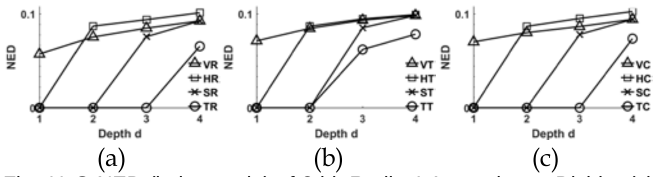


Fig. 11 Q NED (in log scale) of 8-bit Radix-4 Approximate Divider (a) Replacement (b) Truncation (c) Error-compensation

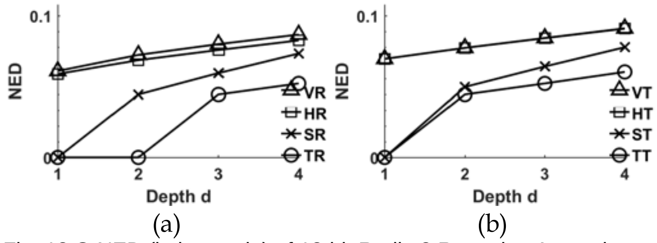


Fig. 12 Q NED (in log scale) of 12-bit Radix-2 Restoring Approximate Divider in [18] (a) Replacement (b) Truncation

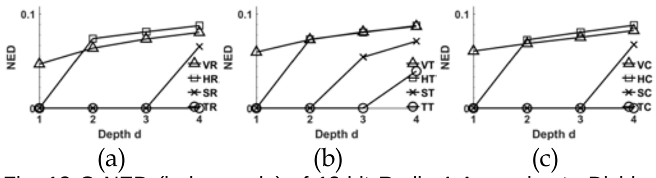


Fig. 13 Q NED (in log scale) of 12-bit Radix-4 Approximate Divider (a) Replacement (b) Truncation (c) Error-compensation

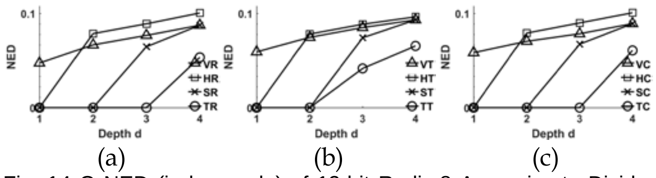


Fig. 14 Q NED (in log scale) of 12-bit Radix-8 Approximate Divider (a) Replacement (b) Truncation (c) Error-compensation

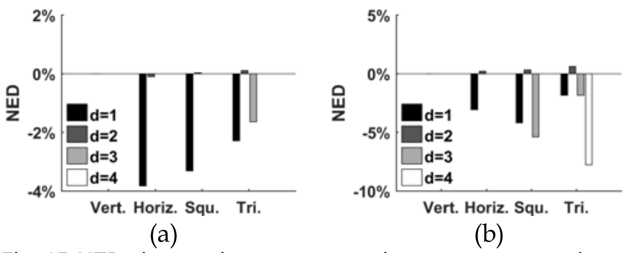


Fig. 15 NED changes in percentage using error compensation with respect to original truncation scheme a) Radix-4 (b) Radix-8

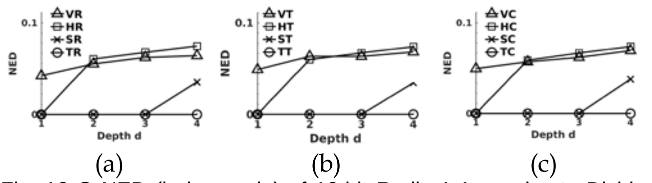


Fig. 16 Q NED (in log scale) of 16-bit Radix-4 Approximate Divider (a) Replacement (b) Truncation (c) Error-compensation

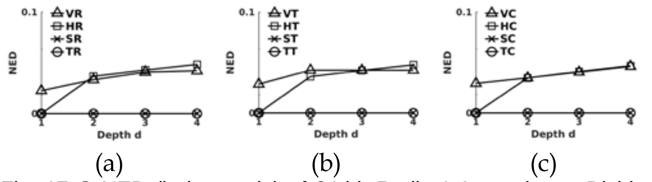


Fig. 17 Q NED (in log scale) of 24-bit Radix-4 Approximate Divider (a) Replacement (b) Truncation (c) Error-compensation

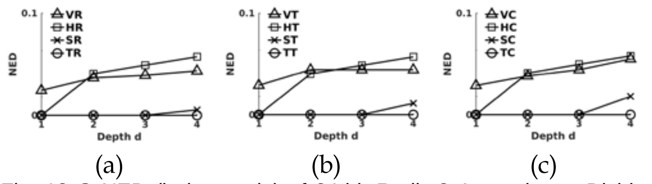


Fig. 18 Q NED (in log scale) of 24-bit Radix-8 Approximate Divider (a) Replacement (b) Truncation (c) Error-compensation

All dividers employing truncation have the worst NED compared to the other two inexact schemes. So, a truncation scheme has a higher NED than a replacement scheme; the difference in NED between them is compensated up to 10% by using the proposed error compensation scheme. The proposed error compensation scheme accomplishes a better NED when truncation is utilized, as it nearly reduces the NED to the same level as a replacement scheme. The compensated NEDs are show in Fig. 15; the NED of truncation is reduced when error compensation is applied. Error compensation is more pronounced for ST and TT compared to the original truncation scheme. When considering the different radix schemes of a 12-bit divider, it is observed that a higher radix results in a higher NED; this occurs because a higher radix makes the quotient digit q_j(as generated at each iteration of the divider) to a larger weight for generating the final Q result. Therefore, the error introduced at each iteration by these approximated configurations has a larger weight and is reflected in the NED of the output Q. The radix-2 AXDr proposed in [18] is also simulated for comparison (Fig. 10 and Fig. 12 both in log scale). Compared to the former AXDr divider, HR-AXD has better NED metrics than

AXDr.

As for high order architectures, 16-bit and 24-bit HR-AXD have been simulated and assessed (Fig. 16 to Fig. 18). As expected, a larger bit width provides a higher precision for the divider; hence, the NED decreases as the bit width increases. Also, when increasing the bit width, the NED differences between replacement, truncation and error compensation are significantly reduced.

4.2 Distribution of Error Occurrence

The error occurrence of different approximate configurations is simulated to assess the effect of parameters (such as the radix, approximation scheme, and depth) on the output error of the proposed dividers.

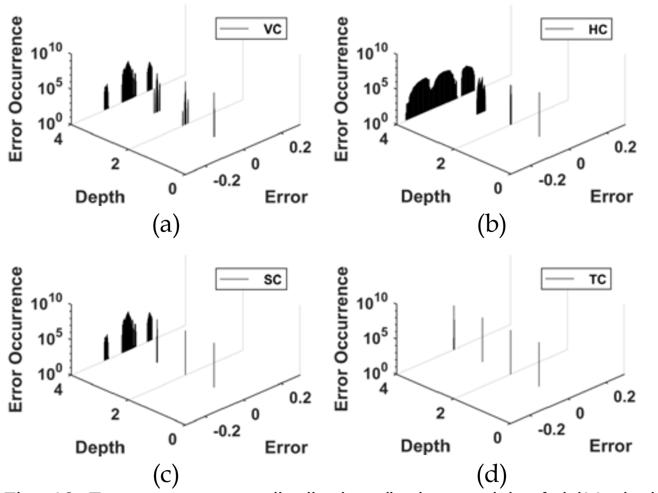


Fig. 19 Error occurrence distribution (in log scale) of (a)Vertical (b)Horizontal (c) Square (d)Triangle Error compensation scheme of 12bit Radix-8 approximate divider

Fig. 19, Fig. 20 and Fig. 21 show the distribution of error occurrence with respect to different truncation/approximation schemes and radix. The distribution is 0-biased and the absolute maximum error is very small in all cases, except for the horizontal scheme. Fig. 19 shows that a horizontal (triangle) scheme has the widest (narrowest) distribution; Fig. 20 shows that the truncation scheme has the largest distribution width and higher average error occurrence. Fig. 21 shows that the distribution is narrowed by increasing the radix, while increasing the error occurrence. It also can be clearly shown that the distribution is widening by increasing the depth in all cases as corresponding to the larger amount of truncated values.

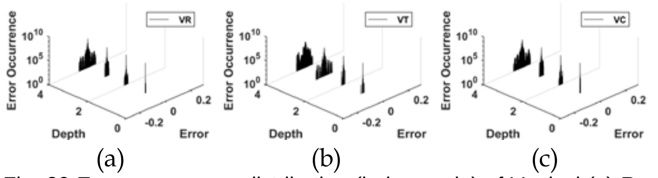


Fig. 20 Error occurrence distribution (in log scale) of Vertical (a) Replacement (b) Truncation (c) Error Compensation approximate for 12bit radix-4 dividers

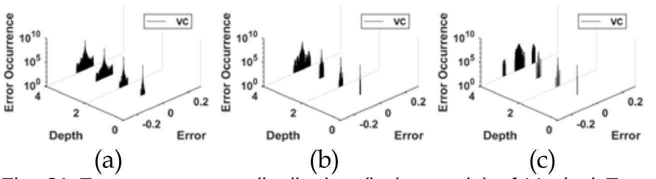


Fig. 21 Error occurrence distribution (in log scale) of Vertical Error Compensation Scheme of 12-bit (a) Radix-2 [18] (b) Radix-4 (c) Radix-8 dividers

4.3 Distribution of Accumulated Error Value

The accumulated error value distribution map for different input combinations of the approximate configurations is simulated to assess the effect of parameters (such as the radix, approximation scheme) on the range of values for which errors may occur.

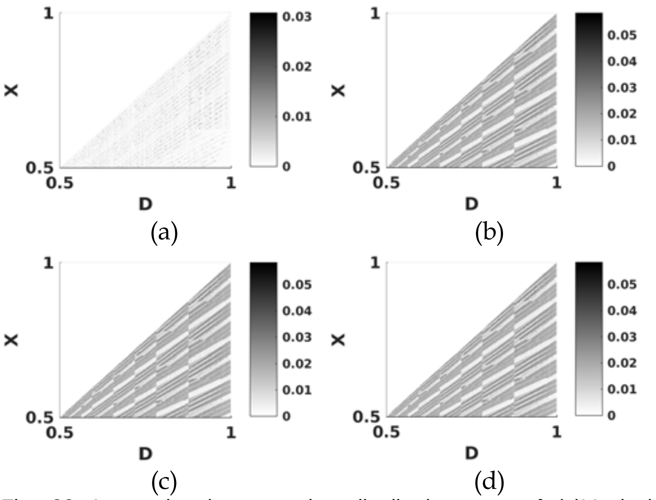


Fig. 22 Accumulated error value distribution map of (a)Vertical (b)Horizontal (c)Square (d)Triangle Error compensation scheme for 12bit Radix-8 approximate divider

Fig. 22 and Fig. 23 show the distribution of error value with respect to the truncation/approximation scheme and radix. The input value pair forms a triangle from 0.5 to 1; so, the error spreads all over this area; the darker color represents the larger accumulated error. The error occurs mostly along bands that are nearly parallel to the X=D 45° line and the largest error is always located right on the edges of each band. Fig. 22 shows that a vertical scheme has the smallest average error; in Fig. 23 the truncation scheme has the largest error and wider average error occurrence. The error compensation scheme effectively recovers the error generated by the truncation scheme.

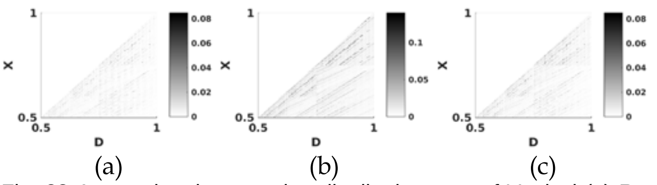


Fig. 23 Accumulated error value distribution map of Vertical (a) Replacement (b)Truncation (c) Error Compensation for approximate 12bit radix-4 dividers

4.4 Power

One of the primary goals of an approximate design is to decrease the power consumption by tolerating a computational error. The power simulation results are shown as Fig. 24 through Fig. 26. As expected, the divider with a higher depth *d* has the smaller power consumption. By increasing *d*, the power consumption for a horizontal configuration decreases faster than for the other types of configuration, so the square and triangle configurations decrease their power consumption at a lower rate than the horizontal and vertical configurations.

All truncated dividers save more power than the other two types of scheme considered in this manuscript. The error compensation scheme consumes a slightly more power than truncation, but its value is still well lower than a replacement scheme. The difference between truncation and error compensation for the power is shown as Fig. 27; the power increase due to the error compensation circuit is marginal. The largest power penalty is not larger than 0.1% (Fig. 27); the delay remains nearly unchanged (more detail in next section), because each stage of the divider array performs a constant time operation in the signed-digit adder.

When two 12-bit HR-AXD (at different radix) are compared, a similar amount of power is consumed, because although a high radix divider can reduce the number of iterations in the division process, in each iteration, it requires more bit cells. Thus, the total power consumption of HR-AXD does not substantially change regardless of the radix. Moreover, the radix-2 AXDr (Fig. 24(a) and Fig. 25(a)) consumes less power than HR-AXD, because the complexity of AXDr is significantly less than HR-AXD. HR-AXD has a better delay than AXDr but at a higher cost in hardware.

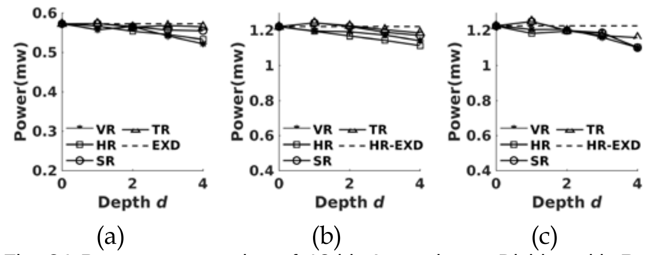


Fig. 24 Power consumption of 12-bit Approximate Divider with Replacement scheme (a) Radix-2 [18] (b) Radix-4 (c) Radix-8

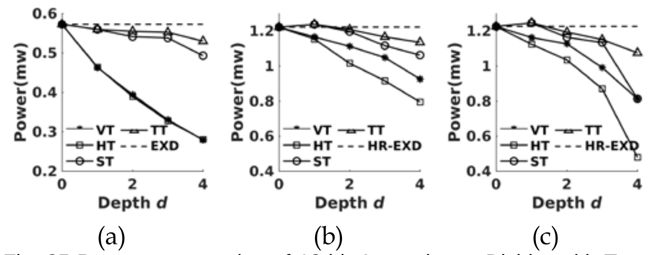


Fig. 25 Power consumption of 12-bit Approximate Divider with Truncation scheme (a) Radix-2[18] (b) Radix-4 (c) Radix-8

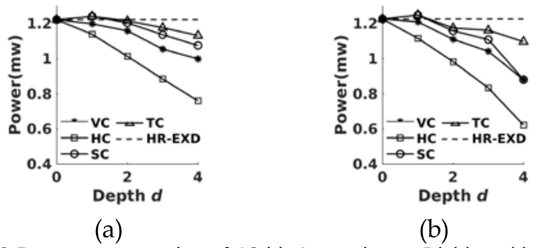


Fig. 26 Power consumption of 12-bit Approximate Divider with Error-Compensation scheme (a) Radix-4 (b) Radix-8

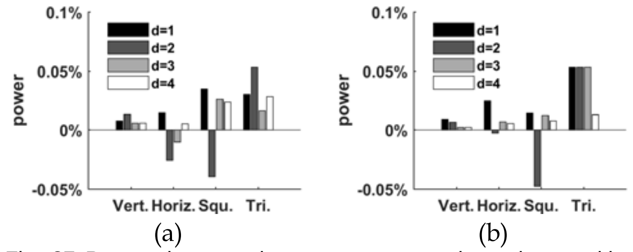


Fig. 27 Power change using error compensation scheme with respect to original truncation scheme (a) Radix-4 (b) Radix-8

4.5 Delay

The delay of each module in the divider is analyzed so that the total delay is then established.

- The delays of the SU and QS are proportional to the bit width and the radix.
- The delay of the PG for each stage is not related to the bit width, but it is proportional to the radix.
- Each binary signed digit adder row has a constant delay.

The critical path of the entire divider is close to the MSB of the array; it starts from the SU through each stage of the PG and the row of the signed digit adder; it finally passes through the QS and On-the-fly conversion module. For a divider with an approximate configuration, (either replacement or truncation), the approximation takes place at the LSB of the adder array, so the delay of the inexact divider is almost the same as the exact counterpart; the only exception is the horizontal configuration. For the horizontal configuration, the replacement scheme has a smaller delay because several stages (equal to the value of the replacement depth) are designed using AXSDACs, that have a smaller delay than EXSDACs. A truncation scheme has even a lower delay than the replacement scheme, because cells are removed. The delays of the different approximate schemes are plotted as Fig. 28 through Fig. 30; as the radix increases from radix-2 to higher radix, the delay decreases, because a high radix divider requires a fewer number of add-subtract iterations, hence the critical path delay is also shorter. Compared to the AXDr (Fig. 28(a) and Fig. 29(a)), the delay of HR-AXD is reduced by nearly 60%. This is the largest benefit when using the carry free HR-AXD compared to the ripple carry radix-2 divider (which has a smaller delay at a high degree of computation parallelism).

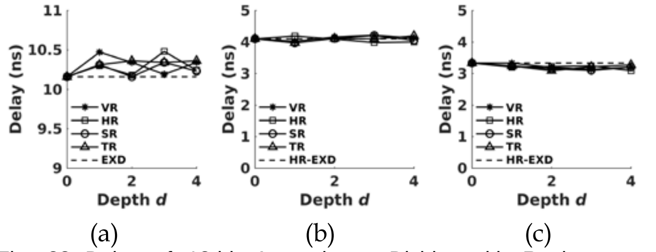


Fig. 28 Delay of 12-bit Approximate Divider with Replacement scheme (a) Radix-2 [18] (b) Radix-4 (c) Radix-8

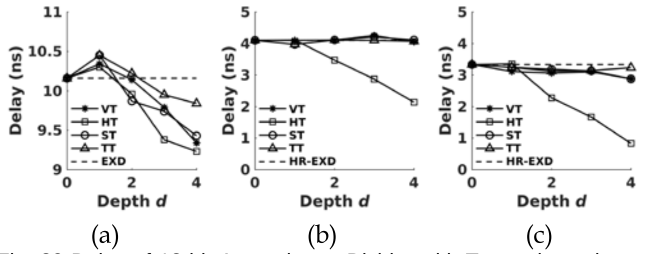


Fig. 29 Delay of 12-bit Approximate Divider with Truncation scheme (a) Radix-2 [18] (b) Radix-4 (c) Radix-8

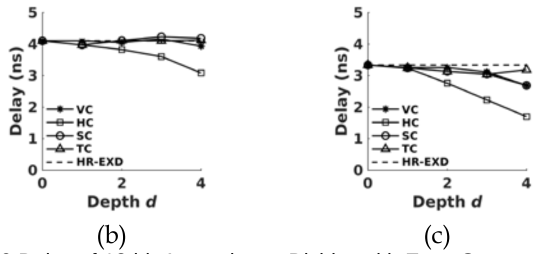


Fig. 30 Delay of 12-bit Approximate Divider with Error-Compensation scheme (a) Radix-4 (b) Radix-8

4.6 Area

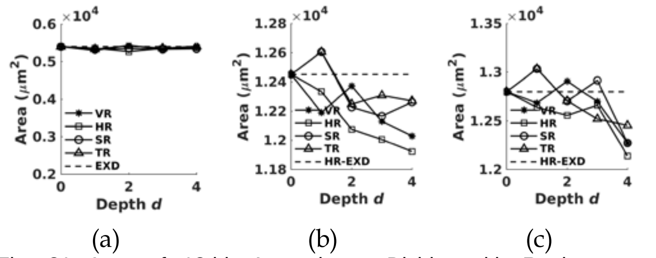


Fig. 31 Area of 12-bit Approximate Divider with Replacement scheme (a) Radix-2 [18] (b) Radix-4 (c) Radix-8

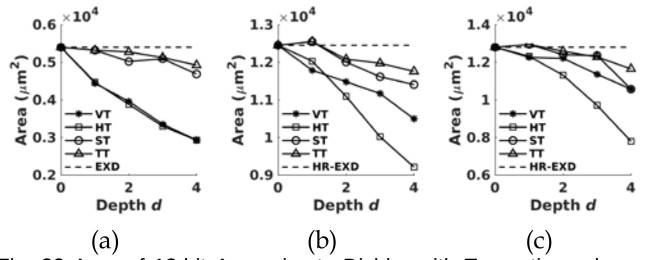


Fig. 32 Area of 12-bit Approximate Divider with Truncation scheme (a) Radix-2[18] (b) Radix-4 (c) Radix-8

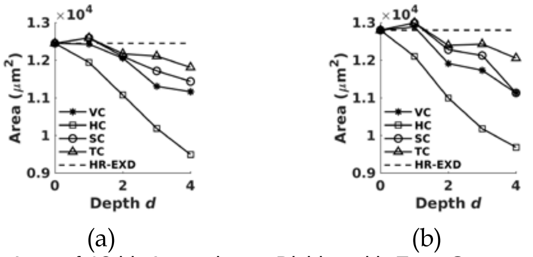


Fig. 33 Area of 12-bit Approximate Divider with Error-Compensation scheme (a) Radix-4 (b) Radix-8

The synthesis results for the area are shown in Fig. 31 through Fig. 33. As expected, the area results of HR-AXD follow the same trends as the power results. The Radix-2 AXDr occupies a smaller area than HR-AXD, thus confirming again that HR-AXD has a higher complexity than AXDr.

4.7 Trade-off between NED and Power

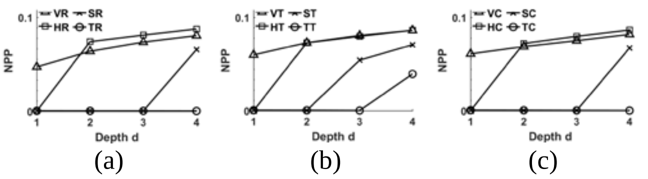


Fig. 34 NPP (in log scale) of 12-bit Radix-4 Approximate Divider with (a) Replacement Scheme (b) Truncation Scheme (c) Error Compensation Scheme

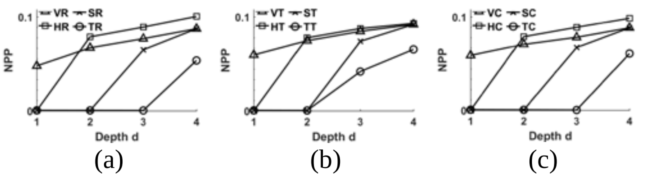


Fig. 35 NPP (in log scale) of 12-bit Radix-8 Approximate Divider with (a) Replacement Scheme (b) Truncation Scheme (c) Error Compensation Scheme

An approximate arithmetic design always must balance accuracy and energy dissipation. As shown previously as the depth changes, the power dissipation increases while the NED decreases. To further evaluate this trade-off, the MED Power Product (MPP) has been introduced in [26]. In this paper, the NPP (NED power product) is used as more relevant than the MPP. Fig. 34 and Fig. 35 show the NPP in log scale of the 12-bit radix-4 and radix-8 dividers using different approximation schemes; the error compensation scheme has the lowest NPP compared to the other two schemes. Compensation recovers part of the error introduced by a truncation scheme to the same level as a replacement scheme, while still preserving the low power advantage of truncation.

5 Applications

In this section, the proposed approximate schemes for high radix division are evaluated for image analysis (on a pixel basis) using input grayscale images normalized in the range [1/2, 1). 12-bit approximate dividers with different configurations are utilized; the approximations for

HR-AXD used in these applications are shown in Table 3; these configurations are chosen such that the power dissipation of these HR-AXD is nearly the same.

TABLE 3 APPROXIMATION DEPTH D CONFIGURATIONS OF 12-BIT HR-AXD USED FOR APPLICATION ANALYSIS

	VR/VT	HR/HT	SR/ST	TR/TT
Radix-4	1	2	3	4
Radix-8	1	2	3	4
Radix-2[18]	2	4	6	8

The Peak Signal-to-Noise Ratio (PSNR) and the SSIM [27] are used as metrics to evaluate the image quality of inexact schemes against an exact scheme. The Mean Structural SIMilarity Index (MSSIM) is a metric to assess image quality; it is based on the finding that human vision is highly adapted to extract structural information from the viewing field. Hence, such measure of structural information change can provide a good approximation to the perceived image distortion. These metrics are given by

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

$$MSSIM(X,Y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j)$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the reference and the distorted images, respectively; $l(x,y)$, $c(x,y)$ and $s(x,y)$ are the luminance, contrast and structure comparison function; α, β, and γ are parameters used to adjust the relative importance of the three components; x_j and y_j are the image contents at the j-th local window; and M is the number of local windows in the image.

5.1 Image Change Detection

Change detection consists of finding the fractional change (or ratio) between two frames of a sequence, i.e.,

$$Q(i,j) = F_{t1}(i,j) \div F_{t2}(i,j)$$

where $F_{t1}(i,j)$ and $F_{t2}(i,j)$ are individual pixel values of two frames of a video or two pictures taken at t_1 and t_2. Q is the resulting output image.

If there is no movement, then the output image Q mostly consists of 1-valued pixels. However, when there is a movement, then the pixels in the regions of the image in which the intensity spatially changes, exhibit significant differences between the two frames. Change detection has been considered also in [19]. Fig. 36 shows the simulation results of the same image (IMG1 in Table 4) in [19] using the Peak Signal-to-Noise Ratio (PSNR) as metric. The square approximation has the best PSNR on average, especially for the radix-4 divider; moreover, the radix-8 divider PSNR is lower than radix-4 dividers on average. A truncated scheme has a PSNR lower than the corresponding replacement scheme, while an error compensation scheme regains most of the PSNR value lost due to truncation. Its value is now at nearly the same level as the corresponding replacement scheme.

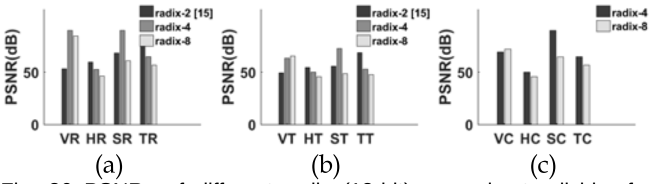
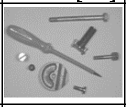
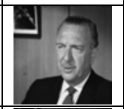
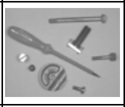
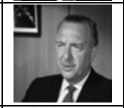
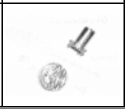
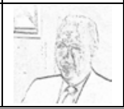
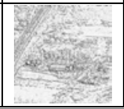
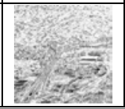
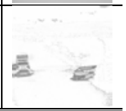


Fig. 36 PSNRs of different radix (12-bit) approximate divider for change detection: (a) replacement (b) truncation (c) with error compensation

TABLE 4 IMAGES FOR CHANGE DETECTION

	IMG1[19]	IMG2	IMG3	IMG4	IMG5
F_{t1}					
F_{t2}					
Q					

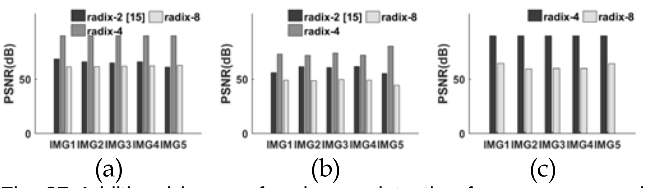
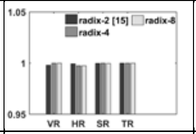
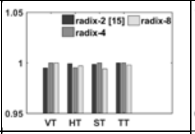
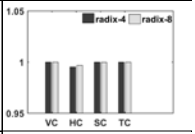
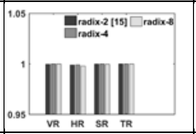
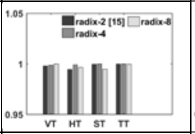
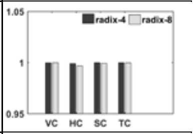
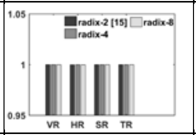
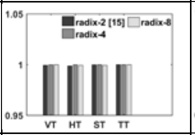
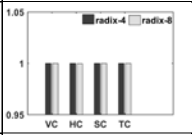
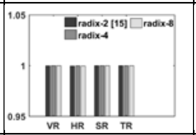
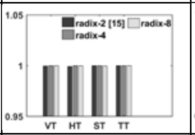
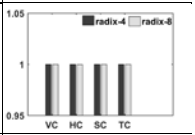
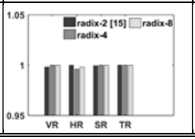
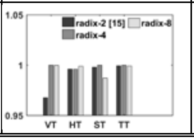
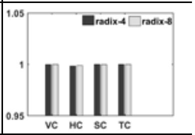


Fig. 37 Additional images for change detection for square approximation (a) Replacement (b) Truncation (c) Error compensation

TABLE 5 MSSIM VALUE FOR ALL IMAGES

	Replacement	Truncation	Error compensation
IMG1 [19]			
IMG2			
IMG3			
IMG4			
IMG5			

Following the initial results of [19], more images (IMG2~5) from [28] are evaluated in this manuscript to assess the error for additional image pairs (Table 4). The square approximation results are plotted in Fig. 37; the proposed approximate divider works well for all considered images, especially for radix-4 dividers. The truncated schemes have the lowest PSNR, while the error compensated schemes efficiently recover the error due to truncation to nearly the same level as a replacement scheme. The MSSIM results of all images are shown as bar charts in Table 5; although the truncation scheme has the lowest MSSIM, the average MSSIMs in all images are above 0.95, so a very high value and confirming that the the proposed approximate dividers will not incur in a significant distortion as observed by human vision.

5.2 Image Scaling

Another common application of image division is to scale down the image values, so that they can be within two saturation limits (minimum and maximum) for appropriate viewing. This is can be performed using division by a constant, i.e.,

$$Q(i,j) = P(i,j) \div C$$

in which P is the image to be scaled, C is the scaling factor and Q is the resulting image.

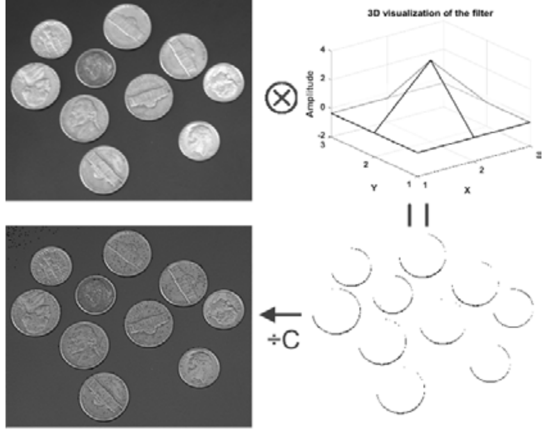
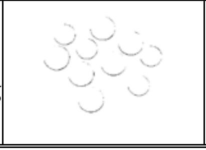
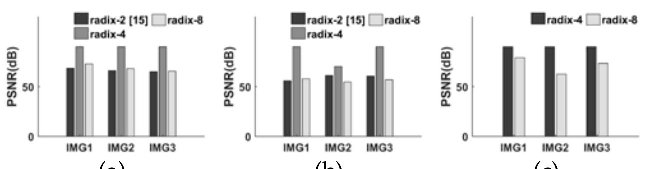


Fig. 38 Example of sharpening image with convolution and use of image scaling[29]

The need of this operation arises on two scenarios: if the source image is too dark or too brigh; as result of a processing step, the resulting matrix has entries with very high values. An example (Fig. 38) of the second scenario is image sharpening this involves first convolution with a 2D filter (top right in Fig. 38). As the products of sums can result in very high values, the convolved image (bottom right in Fig. 38) is divided by a constant, so restoring the appropriate viewing image (bottom left in Fig. 38).

TABLE 6 IMAGES FOR SCALING APPLICATION

	IMG1	IMG2	IMG3
Before scaling			
After scaling			



(a) (b) (c)

Fig. 39 PSNR of Images scaling application for square approximation (a) Replacement (b) Truncation (c) Error compensation

TABLE 7 MSSIM VALUE FOR ALL IMAGES

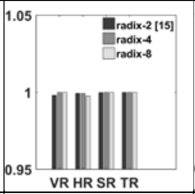
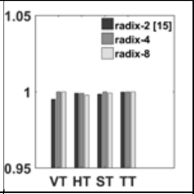
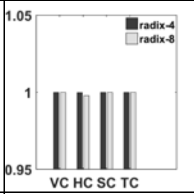
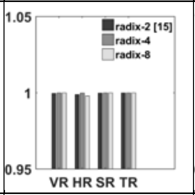
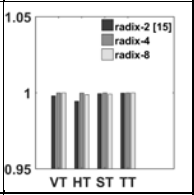
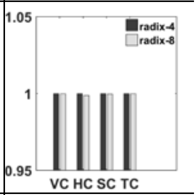
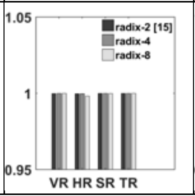
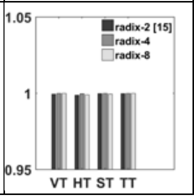
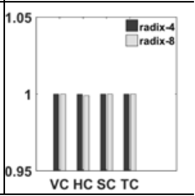
	Replacement	Truncation	Error compensation
IMG1			
IMG2			
IMG3			

Table 6 shows the images used in the scaling application. The results for the triangle approximation are plotted in Fig. 39; the proposed approximate divider works well among all considered images, especially for radix-4 divider. In general, the truncation schemes have the lowest PSNR, while the error compensated schemes efficiently recover the error caused by truncation to nearly the same level as a replacement scheme. The scaling application requires an image divided by a constant value, so the overall PSNR is higher than the PSNR for the change detection application. The MSSIM results of all images are shown in Table 7. Although a truncation scheme has the lowest MSSIM, the average MSSIMs in all images are above 0.95, so indicating that the proposed approximate dividers do not cause a significant distortion in the image scaling application.

6 CONCLUSION

This paper has presented a detailed analysis, design and evaluation of high radix parallel dividers that utilize approximate criteria in their operation. The basic computational modules for high-radix division have been reviewed and the signed-digit adder has been simplified to accomplish inexact computing. Replacement, truncation and error compensation schemes have been proposed; different design parameters including the number of bits *N*, the radix number *r*, and the replacement depth *d* have been varied to evaluate the performance and error characteristics of these inexact high radix dividers. Compared to the original exact design counterpart in [20], simulation has shown that the power dissipation and delay of the inexact designs are better than the original exact design(for all cases). Moreover, the following conclusions can be drawn.

- The error characteristics can be clearly seen from the NED and error distribution. A larger value for *d* provides a larger NED and a wider error distribution; among all schemes, the triangle replacement divider has the best NED and the narrowest error distribution among the placement schemes considered in this manuscript. A truncated scheme introduces more error, but if error compensation is introduced, the error introduced by truncation can be mitigated by utilizing a compensation scheme; this proposed scheme utilizes a 3-cell cluster that results in a significant improvement in error characteristics at a marginal increase in power dissipation.
- The power consumption and area reduce rapidly as the depth increases, i.e. the higher the depth is, more pronounced is the power and area reduction. A truncation scheme provides a significant power and area reduction compared to a replacement scheme; the error compensation scheme slightly increases the power consumption and area compared to truncation scheme, but its value is still significantly lower than the replacement schemes.
- The delay of a radix-4 divider is significantly less (nearly 60%) than for a radix-2 divider. Compared to radix-4, the delay is reduced about 25% for radix-8 dividers. As only N/m stages are required to complete the division, larger the value of m, smaller is the delay. The delay of all approximate divider schemes is however not significantly affected by the different approximate configurations.

Compared to the radix-2 AXDr divider design of [18], a HR-AXD has a higher complexity and power consumption, but the error characteristics and computation speed are significantly improved by using HR-AXD; The combined metric of the NPP is reduced when the proposed error compensation scheme is utilized; so the HR-AXD with error compensation scheme is better suited for latency critical applications requiring a small NPP. Compared to other state-of-the-art approximate dividers, e.g., TruncApp[14] and SEERAD[13], the proposed design employes a high-radix array division algorithm by focusing on a transistor-level approximate implementation, so better suited to meet specific application constraints. For example, when considering image division, the error metrics of the proposed design (at an average of 50dB for the PSNR and 0.98 for the MSSIM) are significantly better than for SEERAD and TruncApp (at an average of 28dB for the PSNR and 0.77 for the MSSIM); therefore, improvements in performance must also met satisfactory application requirements and limiting errors in computation.

In conclusion, when designing an approximate array divider, metrics (and related design parameters) must be considered and met as per the specific application. For power critical application, the former AXD design [18] could be still a better choice; for speed critical applications, HR-AXD is overall the best candidate. When designing a divider, the power penalty incurred by choosing a high radix can be reduced using approximation; most of the errors introduced by approximation can be compensated using the proposed scheme

REFERENCES

[1] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise Adders for Low-Power Approximate Computing," in *International Symposium on Low Power Electronics and Design*, 2011, pp. 409-414.

[2] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-Based Adders for Inexact Computing," in *13th IEEE Conference on Nanotechnology (IEEE-NANO)*, 2013, pp. 690-693.

[3] N. Zhu, W.-L. Goh, and K.-S. Yeo, "An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application," in *12th International Symposium on Integrated Circuits*, 2009, pp. 69-72.

[4] N. Zhu, W.-L. Goh, G. Wang, and K.-S. Yeo, "Enhanced Low-Power High-Speed Adder for Error-Tolerant Application," in *International SoC Design Conference (ISOCC)*, 2010, pp. 323-327.

[5] N. Zhu, W.-L. Goh, and K.-S. Yeo, "Ultra Low-Power Hig-Speed Flexible Probabilistic Adder for Error-Tolerant Applications," in *International SoC Design Conference (ISOCC)*, 2011, pp. 393-396.

[6] A. K. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in *Design, Automation and Test in Europe (DATE)*, 2008, pp. 1250-1255.

[7] D. Esposito, D. D. Caro, and A. G. M. Strollo, "Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 63, no. 8, pp. 1200-1209, 2016.

[8] A. B. Kahng and S. Kang, "Accuracy-Configurable Adder for Approximate Arithmetic Designs," in *49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 820-825.

[9] K. Y. Kyaw, W.-L. Goh, and K.-S. Yeo, "Low-Power High-Speed Multiplier for Error-Tolerant Application," in *International Conference of Electron Devices and Solid-State Circuits*, 2010, pp. 1-4.

[10] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," in *24th International Conference on VLSI Design*, 2011, pp. 346-351.

[11] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2003). *Hypermedia Image Processing Reference (HIPR2)*. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2

[12] S. Hashemi, R. I. Bahar, and S. Reda, "A low-power dynamic divider for approximate applications," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1-6.

[13] R. Zendegani, M. Kamal, A. Fayyazi, A. Afzali-Kusha, S. Safari, and M. Pedram, "SEERAD: A high speed yet energy-efficient rounding-based approximate divider," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*,

2016, pp. 1481-1484.

[14] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, and Z. Navabi, "TruncApp: A truncation-based approximate divider for energy efficient DSP applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 1635-1638.

[15] S. F. Oberman and M. Flynn, "Division algorithms and implementations," *IEEE Transactions on Computers*, vol. 46, no. 8, pp. 833-854, 1997.

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

[17] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2004.

[18] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the Design of Approximate Restoring Dividers for Error-Tolerant Applications," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2522-2533, 2016.

[19] L. Chen, J. Han, W. Liu, P. Montuschi, and F. Lombardi, "Design of Approximate High-Radix Dividers by Inexact Binary Signed-Digit Addition," in *27th Great Lakes Symposium on VLSI*, Banff, Alberta, USA, Canada, 2017.

[20] T. Aoki, K. Nakazawa, and T. Higuchi, "High-radix parallel VLSI dividers without using quotient digit selection tables," in *30th IEEE International Symposium on Multiple-Valued Logic*, 2000, pp. 345-352.

[21] M. D. Ercegovac, T. Lang, and P. Montuschi, "Very-high radix division with prescaling and selection by rounding," *IEEE Transactions on Computers*, vol. 43, no. 8, pp. 909-918, 1994.

[22] P. Montuschi and T. Lang, "Boosting very-high radix division with prescaling and selection by rounding," *IEEE Transactions on Computers*, vol. 50, no. 1, pp. 13-27, 2001.

[23] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 389-400, 1961.

[24] M. D. Ercegovac and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations," *IEEE Transactions on Computers*, vol. C-36, no. 7, pp. 895-897, 1987.

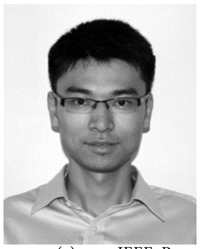
[25] L. Jinghang, H. Jie, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760-1771, 2013.

[26] L. Chen, J. Han, W. Liu, and F. Lombardi, "Design of Approximate Unsigned Integer Non-restoring Divider for Inexact Computing," in *25th Great Lakes Symposium on VLSI*, Pittsburgh, Pennsylvania, USA, 2015, pp. 51-56, 2742063: ACM.

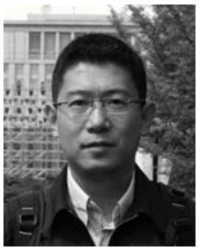
[27] W. Zhou, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.

[28] "The USC-SIPI Image Database," A. G. Weber, Ed., 5 ed. Signal and Image Processing Institute, University of Southern California, Department of Electrical Engineering.

[29] U. Qidwai and C. H. Chen, *Digital Image Processing: An Algorithmic Approach with MATLAB*. CRC Press, 2009.



Linbin Chen received the B.Sc. degree in information engineering from Beijing Institute of Technology, China, in 2009, and the M.S. degree from Northeastern University, Boston, USA, in 2012, where he is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering. His research interests include low power and high performance VLSI design, emerging logic and memory devices and circuits, inexact and fault tolerant computing.



Jie Han (S'02-M'05-SM'16) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from Delft University of Technology, The Netherlands, in 2004.
He is currently an associate professor in the Department of Electrical and Computer Engineering at the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computation, reliability and fault tolerance, nanoelectronic circuits and systems, and novel computational models for nanoscale and biological applications.

Weiqiang Liu (S'10-M'12-SM'15) received the B.Sc. degree in Information Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China and the Ph.D. degree in Electronic Engineering from the Queen's University Belfast, Belfast, UK, in 2006 and 2012, respectively. He is currently an associate professor in the College of Electronic and Information Engineering at Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include approximate computing, computer arithmetic and cryptographic hardware.

Paolo Montuschi (M'90-SM'07-F'14) is a Full Professor in the Department of Control and Computer Engineering and a Member of the Board of Governors at Politecnico di Torino, His research interests include computer arithmetic and architectures, computer graphics, semantics and education. He is an IEEE Fellow, an IEEE Computer Society Golden Core member, a recipient of the "Distinguished Service" and the "Spirit of the Computer Society" awards. Currently, he is serving as Editor-in-Chief of IEEE Transactions on Computers, as the Chair of the Ccomputer Society Awards Committee, and as a member of the IEEE Publications Services and Products Board, He is a life member of the International Academy of Sciences of Turin and of Eta Kappa Nu (the Honor Society of IEEE). In March 2017, he co-founded the first HKN Student Chapter in Italy.

Fabrizio Lombardi (M'81-SM'02-F'09) graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. from the University of London (1982).
He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books.