# Design of Approximate High-Radix Dividers by Inexact Binary Signed-Digit Addition

Linbin Chen
Fabrizio Lombardi
ECE Department
Northeastern University
lombardi@ece.neu.edu

Paolo Montuschi
ECE Department
Polytechnic University of Turin, Italy
paolo.montuschi@polito.it

Jie Han
ECE Department
University of Alberta
Edmonton, AB, Canada
jhan8@ualberta.ca

Weiqiang Liu
College of EIE.
Nanjing University of Aero & Astro., China
liuweiqiang@nuaa.edu.cn

## 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 schemes with inexact cells and truncation schemes of exact cells in the binary signed-digit adder array is introduced. 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.

## CCS Concepts

**• Hardware~Integrated circuits • Hardware~Arithmetic and datapath circuits • Hardware~Combinational circuits**

## Keywords

Approximate Divider; High-radix; Normalized Error Distance; Power Dissipation

## 1. INTRODUCTION

Most computer arithmetic applications are implemented using digital logic circuits, thus operating with a high degree of precision. However, many applications such as multimedia and image processing can tolerate errors and imprecision in computation and still produce results that can be useful in which human senses (such as vision) are involved. 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.

There are increasing demands for high-speed dividers in today's floating point units (FPU) and digital signal processors. Reductions in delay and power consumption have been studied for the division operation; different algorithms for division can be found in the technical literature [1]. Digit-by-digit (digit recurrence) methods are widely used in hardware design; implementations can be either sequential, combinational, or a combination of both. Based on these findings, the combinational implementations of a divider based on restoring, non-restoring [2] and SRT algorithms [3] are treated in more detail as they employ highly modularized structure for inexact designs.

In the previous work by the same authors [4], the designs of approximate unsigned non-restoring and restoring dividers (denoted as AXDnr and AXDr) have been proposed. New approximate AXDnr cells (denoted as AXDCnr) 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.

[5] has presented an algorithm for high-radix exact division; this algorithm is based on a prescaling technique of division by digit recurrence. In this manuscript we revisit, generalize and extend the results of [5] and present a novel architecture for approximate division. The approximate design concept of [4] is explored in more detail for a high-radix array divider (such as proposed in [5]). At circuit level, the signed-digit adder cell is simplified; replacement or truncation is utilized in the divider. The contribution of this manuscript extends also to an algorithmic domain as different numbers of bits *(N),* radix number (*r*) and the replacement/truncation depth *(d)* are also considered. Circuit-level performance and the error characteristics of these inexact high radix dividers are analyzed, inclusive of error analysis for image processing applications.

## 2. REVIEW OF DESIGN BASIS

In a high-radix division circuit design, Redundant Binary Representation (RBR) is used to represent the quotient digits when they are selected at each stage. For example, in the radix-4 divider discussed later, the intermediate quotient $q_j \in [-3, +3]$ is represented in signed-magnitude form; a Binary Signed-Digit (BSD) numbering system (as a subcategory of a RBR system [6]) with a digit set of {-1,0,1} is used to represent the partial remainder at each stage. RBR allows addition without propagating a carry. When compared to a non-redundant representation, RBR makes digit-wise logic operations slower, however arithmetic operations are faster when a large bit width is used [3].

### 2.1 High-radix Division Algorithm

There are two approaches for division by digit recurrence [3]. The first approach is based on selection tables; the second approach is based on prescaling. In the first approach, the digits of the quotient are obtained by inspecting the dividend and the partial remainder; as per the number of value combinations (as stored in digit selection tables), the quotient digit is obtained at each iteration [7]. However, for higher radix the size of these tables increases, yielding large implementations. Among the techniques for addressing this negative design feature, prescaling technique


*GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada*

DOI: http://dx.doi.org/10.1145/3060403.3060404

(as proposed in [8, 9]) 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 [5]. In this algorithm, by doing additional prescaling and converting the number representation, the quotient digit is directly obtained from the most significant (integer) part of the partial remainder at every iteration; this makes possible the construction of fully combinational high-radix dividers exhibiting a lower latency compared to a radix-2 divider.

Let the dividend and divisor be denoted by $X$ and $D$ respectively (normalized as $1/2 \le X < D < 1$); A standard SRT division algorithm with $n$-bit precision is given by the recursive equations:

$$R^{(0)} = X$$
$$R^{(j+1)} = rR^{(j)} - q_{j+1}D, \quad (1)$$

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 $j$th quotient digit selected from {-($r$-1), … -1, 0, 1, …, $r$-1}. The quotient $Q$ and the final remainder $R$ are given by

$$Q = \sum_{j=1}^{n} r^{-j} \cdot q_j, \quad (2)$$
$$R = r^{-n} \cdot R^{(n)}. \quad (3)$$

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 $\lceil n/m \rceil$.

Normally, (1) as a recursive equation is calculated using Signed-Digit (carry-free) adders based on a redundant BSD representation. The shifted partial remainder $rR^{(j)}$ is bounded as

$$0 \le \left|rR^{(j)}\right| < rD < r = 2^m. \quad (4)$$

So, $rR^{(j)}$ can be represented by using a BSD numbering system (denoted as SD2) as

$$rR^{(j)} = [a_{m-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-k}]_{SD2} = \sum_{i=-k}^{m-1} 2^i a_i, \quad (5)$$

where $a_i \in \{-1,0,1\}$.

The high-radix division algorithm of [5] is based on a SRT digit-by-digit division; additionally, it requires the following two steps (or conditions) to be performed (or satisfied): (A) The divisor $D$ and the dividend $X$ must be prescaled transforming the divisor range from [1/2,1) to [$D_{min}$, 1) as described later (operand scaling does not affect the result of division, because $X/D = MX/MD$, where $M$ denotes the scale factor. Note that it is well known to affect the value of the final partial remainder, if necessary, because its value is multiplied by $M$. (B) Convert 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}. After the second step, (5) is given by

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

where $b_i \in \{-1,0\}$ or $\{0,1\}$, $a_i$, $c_i \in \{-1,0,1\}$. Then, the $(j+1)^{th}$ quotient digit $q_{j+1}$ can be obtained directly from the integer part of $rR^{(j)}$,i.e.

$$q_{j+1} = [c_{m-1} \dots c_1 c_0]_{SD2} = \sum_{i=-k}^{m-1} 2^i c_i, \quad (6)$$

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

The proof of the algorithm in [5] as well as the derivation of the scaling range [$D_{min}$,1) are as follows. Since, $rR^{(j)}$ can be represented as

$$rR^{(j)} = q_{j+1} + \epsilon \quad (7)$$

where $\epsilon$ denotes explicit value of the fractional part digits and it is given by

$$\epsilon = [0 . b_{-1} \dots b_{-t} a_{-t-1} \dots a_{-k}]_{SD2} = \Sigma_{i=1}^{t} 2^{-i} b_{-i} + \Sigma_{i=t+1}^{k} 2^{-i} a_{-i}$$

The fractional part $\epsilon$ is a function of $t$ and is bounded in the following rages:

$$\text{If } b_i \in \{-1,0\}, \text{ then } -1 < \epsilon < 2^{-t} \quad (8)$$
$$\text{If } b_i \in \{0,1\}, \text{ then } -2^{-t} < \epsilon < 1 \quad (9)$$

In SRT division using the maximally redundant digit set $\{-(r-1), \dots, r-1\}$, $q_{i+1} (\in \{-(r-1), \dots, r-1\})$ is obtained if $q_{i+1}$ satisfies the following conditions:

$$rR^{(j)} \le (q_{i+1} + 1)D \ \ (q_{i+1} \in \{-(r-1), \dots, r-2\}) \quad (10)$$
$$(q_{i+1} + 1)D \le rR^{(j)} \ \ (q_{i+1} \in \{-(r-2), \dots, r-1\}) \quad (11)$$

By substituting (7) in (10) and (11), then

$$q_{i+1} + \epsilon \le (q_{i+1} + 1)D \ \ (q_{i+1} \in \{-(r-1), \dots, r-2\}) \quad (12)$$
$$(q_{i+1} - 1)D \le q_{i+1} + \epsilon \ (q_{i+1} \in \{-(r-1), \dots, r-2\}) \quad (13)$$

Using (12) and (13) and considering the bound of $\epsilon$ given by (8) or (9), the possible scaling range of $D$ must be satisfied for selecting $q_{j+1}$ (as the explicit value of the integer part of $rR^{(j)}$) is derived as in (14). Finally, the process to select $q_{j+1}$ is summarized as following:

- Prescale the operands $X$ and $D$ to satisfy

$$D_{min} = \frac{r-2+2^{-t}}{r-1} \le D \le 1, \quad (14)$$

where $D_{min}$ is the lower bound of the scaled $D$.

- Convert the $t$ most significant digits $a_{-1}a_{-2} \dots a_{-t}$ ($a_i \in \{-1,0,1\}$) of the fractional part of every partial remainder $rR^{(j)}$ into its non-redundant representation $b_{-1}b_{-2} \dots b_{-t}$ as

$$b_{-1}b_{-2} \dots b_{-t} \in \{-1,0\} \text{ for } rR^{(j)} > 0 \quad (15)$$
$$b_{-1}b_{-2} \dots b_{-t} \in \{0,1\} \text{ for } rR^{(j)} < 0 \quad (16)$$

## 2.2 Exact High-radix Divider (HR-EXD)

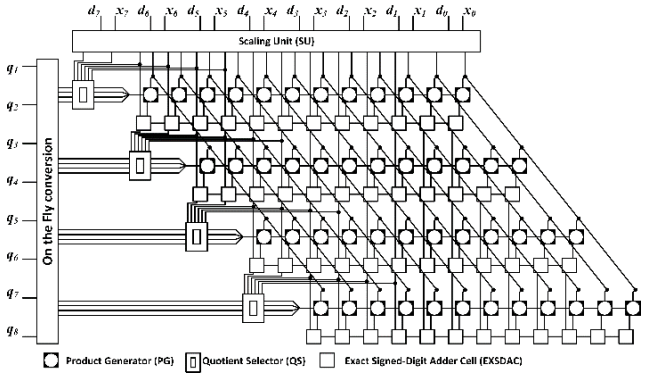
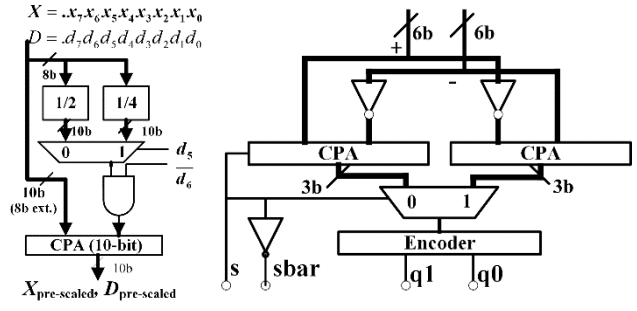



**Figure 1. Example of HR-EXD for 8-bit radix-4 [5]**

The implementation of the HR-EXD for the algorithm reviewed previously is presented in [5]. The structure of an 8-bit radix-4 divider with $r$=4, $t$=2 is shown in Figure 1; it consists of the following modules as basic blocks.

*Scaling Unit (SU):* as input operands, the divisor $D$ and the dividend $X$ must be prescaled prior to starting the division iterations. For example, in a divider with $r$=4 and $t$=2, the divisor and the dividend must be prescaled to transform the divisor range from $[\frac{1}{2}, 1)$ to $[\frac{3}{4}, 1)$; prescaling can be done using different scale factors for 3 different ranges of $D$ ($D$>=3/4, 5/8<=$D$<3/4, 1/2<$D$<=5/8). All these constant-scale multiplications can be achieved by shift-and-add operations; Figure 2(a) shows the

operand scaling unit using shifters and carry propagation adders (CPA). In general, as the radix $r$ increases and the number of digit $t$ increases, a higher complexity is required for operand scaling and the multiplication at each iteration. Therefore, the values of $r$ and $t$ must be selected to meet the requirements of performance and complexity. In this paper, the parameter $t=m$ *(where* $r=2^m$) has been selected so the same configuration as in [5], as it does not lead to a loss of generality of the proposed method. For radix-8 and a higher radix, the structure of the scaling unit is also the same and it can be extended based on the radix-4 case (Figure 2(a)).



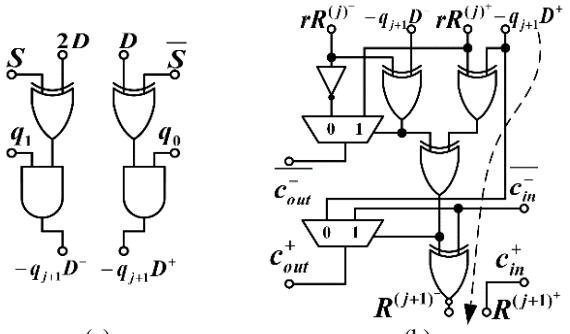

(a) (b)

**Figure 2. (a) SU module for r=4, t=2 (b) QS module for radix 4 ($s$: sign, $q_1q_0$: magnitude) [5]**

*Quotient Digit Selector (QS):* According to the algorithm in [5], the real binary value of the integer part of $rR^{(j)}$ is selected as the $(j+1)^{th}$ quotient digit $q_{j+1}$. Implementation of this quotient selection rule does not need a digit selection table; it needs only a high-speed signed-digit carry propagation adder (SDCPA) for a limited number of bits (Figure 2(b)). The word length of SDCPA is $2m+t$ bits, because the integer part of $rR^{(j)}$ must also be converted into a non-redundant form to obtain the explicit value of $q_{i+1}$. This conversion is performed by adding the positive digits and the two's complement of the negative digits of $rR^{(j)}$ for $rR^{(j)} >= 0$, and by adding the negative digits and the two's complement of the positive digits of $rR^{(j)}$ for $rR^{(j)} <= 0$. The obtained quotient digit $q_{j+1}$ is in a sign-magnitude redundant form; it is then finally transformed into a non-redundant 2's complementary representation using the on-the-fly scheme of [10]. For a higher radix, the design of the QS can be extended based on the radix-4 case (Figure 2(b)).

*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 completely computed; however, this addition increases the overall execution time. So to avoid this step, the on-the-fly algorithm of [10] is used to perform the conversion in a digit-serial fashion to generate the digits of the quotient.

*Product Generator (PG):* The high-radix division algorithm employs a BSD number representation for partial remainders and quotient digits, because this representation exploits the hardware simplicity of a radix-2 scheme; a drawback of high-radix algorithms is that they require the generation of multiples of the divisor $qD$ ($q \in \{-(r-1), \dots, r+1\}$). In the radix-4 divider, this problem is addressed by designing a $qD$ product generator (PG). PG represents all multiples of $D$ $(-3D, \dots, 3D)$ as pair of multiples that are generated by shift operations. The circuit diagram of a PG is shown in Figure 3(a); to transfer the output directly to the binary signed digit adder, PG must have the additional function of complementing the outputs. The PG module for radix-8 and higher, can be extended from the radix-4 case.




(a) (b)

**Figure 3. (a) PG module for radix-4 case (b) EXSDAC module [5]**

*Exact Signed Digit Adder Cell (EXSDAC):* Figure 3(b) shows the Exact Signed Digit Adder Cell (EXSDAC); it consists of XOR/XNOR gates and 2-1 MUXs (each designed by modifying 4-2 compressors). The EXSDACs implement (1); the adder inputs $rR^{(j)}$, $-\mathrm{q_{j+1}D}$ and output $R^{(j+1)}$ are represented in binary signed digit form $(rR^{(j)^+}, rR^{(j)^-})$, $(-q_{\mathrm{j+1}}\mathrm{D}^+, -q_{\mathrm{j+1}}\mathrm{D}^-)$ and $(R^{(j+1)^+}, R^{(j+1)^-})$ respectively. The EXSDAC remains the same for all HR-EXD at different bit width and radix. The approximate designs shown next are based on the approximation of EXSDAC; thus, the proposed approximate design methodology is suitable for HR-EXD with different bit width and radix.

# 3. PROPOSED APPROXIMATE DESIGNS

## 3.1 Approximate Signed-Digit Adder Cell (AXSDAC)

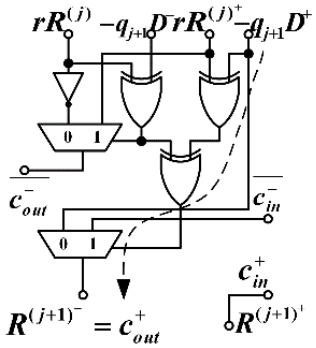



**Figure 4. Approximate design of EXSDAC (AXSDAC)**

The EXSDAC computes the subtraction or addition according to the quotient selection output. As $R^{(j+1)^+} = \mathrm{C_{in}^+}$, 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 any gate (Figure 4(b)). By observing the truth table of EXSDAC, the $C_{out}^+$ has the same value as the input $\mathrm{R^{(j+1)^-}}$ in 24 out of 32 state combinations; therefore, an approximate design must consider this feature. The approximate design of EXSDAC (denoted as AXSDAC) is shown in Figure 4. $\mathrm{R^{(j+1)^-}}$ is simplified to be equal to $C_{out}^+$ by changing the 8 outputs value of $C_{out}^+$. This design has therefore 8 incorrect outputs out of 32 outputs, so in theory its error rate is 25% (assuming that all combinations are equally probable). In terms of circuit implementation, pass-transistor logic design of [4] is utilized to further decrease the circuit complexity, the delay and the power consumption.

## 3.2 Approximate High Radix Divider (HR-AXD)

### *3.2.1 Replacement Scheme*

When designing the approximate divider, EXSDACs are selectively replaced by AXSDACs; hence, approximation is the process by which an exact cell is replaced by an approximate cell. 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 $2\hat{}m(m = 1, 2, 3 \dots)$ divider, the number of EXSDACs is given as $(N + 2m) * (N/m)$; $m$ EXSDACs are combined together 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 Figure 5 by the dotted rectangles of EXSDACs). The replacement configurations and corresponding depth $d$ for an 8-bit radix-4 array divider are shown in Figure 5. Four types of replacement are used for approximation:

*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 at output. Hence, $m\left(\frac{N}{m}\right)d = Nd$. An example for $m$=2, $d$=2 is shown in Figure 5(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 $d$th LSB of Q is possible. An example of a horizontal replacement divider of depth $d$=2 is shown in Figure 5(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 Figure 5(c).

*Triangle Replacement (TR):* Consider the integer pair (x,y) as coordinates of each 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 $d$ ($d\geq1$), $md(d + 1)/2$ EXSDACs are replaced with AXSDACs. An example of a triangle replacement divider with $d$=2 is shown in Figure 5(d).

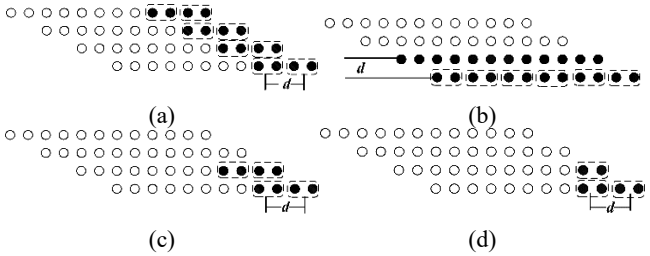



**Figure 5. Different approximate configurations and replacement depths for 8-bit radix-4 divider (a) VR $d$=2 (b) HR $d$=2 (c) SR $d$=2 (d) TR $d$=2**

### *3.2.2 Truncation Scheme*

Truncation is different from replacement, because the EXSDACs are not changed to AXSDACs, instead, are completely eliminated. Same as 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).

# 4. SIMULATION RESULTS

## 4.1 AXSDAC

Predictive technology models at 45nm feature size are utilized in the HSPICE simulation. AXSDAC and EXSDAC are simulated at a 1GHz frequency; a fan-out of 4 is utilized in all simulations. The simulation results for the delay, power consumption and the power-delay product (PDP) are given in Table 1. As expected, the proposed inexact design shows significant improvements in delay, power consumption and PDP. As a measure of circuit complexity; the approximate designs incur in a reduction of 18% in the number of transistors.

**Table 1. Simulation results of EXSDAC and AXSDAC**

| Design | Num. of Transistors | Delay (ps) | Power (µW) | PDP (aJ) |
|---|---|---|---|---|
| EXSDAC | 22 | 6.36 | 2.98 | 18 |
| AXSDAC | 18 | 4.35 | 1.14 | 5 |

## 4.2 HR-AXD

### *4.2.1 NED*

The Normalized Error distance (NED) is defined as the Mean Error Distance (MED) normalized by the maximum ED [11]. The maximum value of the ED is 1, so in this case the NED is equal to the MED. Only the 8-bit radix-2 and radix-4, 12-bit radix-2, radix -4 and radix 8 divider are evaluated (the trend and conclusions for these dividers are applicable also to a higher radix divider). The NED simulation results are shown in Figure 6 and Figure 7 for different bit width, radix, and approximation configurations.

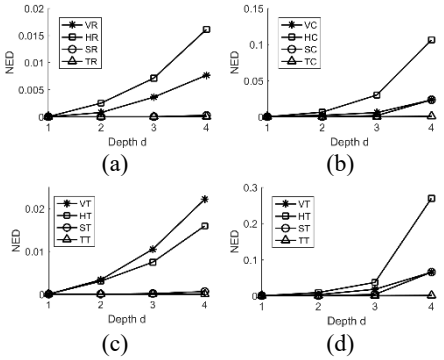



**Figure 6 Q NED of 8-bit HR-AXD (a) (c) Radix-2 (b) (d) Radix-4**

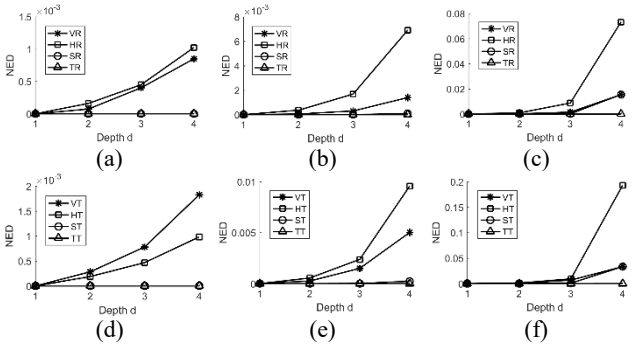



**Figure 7. Q NED of 12-bit HR-AXD with (a)(d) Radix-2 (b)(e) Radix-4 (c)(f) Radix-8**

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. 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; Compare the 8-bit Radix-4 divider and the 12-bit Radix-4 divider. A larger bit width provides a higher precision for the divider; hence, the NED decreases as the bit width increases. When considering the three 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 configuration has a larger weight and is reflected in the NED of the output Q.

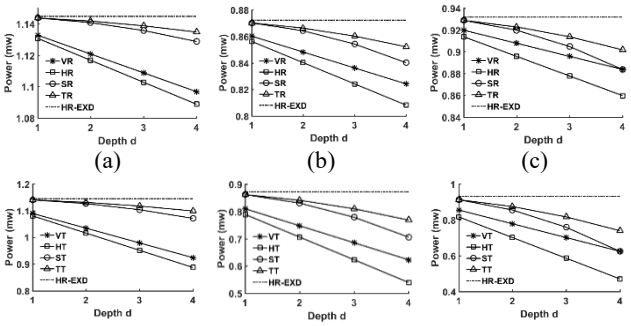



**Figure 8. Power Consumption of 12-bit HR-AXD with (a)(d) Radix-2 (b)(e) Radix-4 (c)(f) Radix-8**

### *4.2.2 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 Figure 8.

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 replacement scheme counterpart. When comparing the 8-bit radix-4 and the 12-bit radix-4 divider, a larger bit width divider consumes more power as expected; when the three 12-bit dividers (at different radix) are compared, the radix-4 and radix-8 dividers consume a lower power than the radix-2 divider, because a high radix divider can reduce the number of iterations in the division process, thus significantly decreasing the circuit complexity of the sign-digit adder array. Moreover, the radix-8 divider consumes more power than radix-4 divider, because, as the radix increases, the implementation of the SU module, the PG module and the On-the-Fly conversion module are more complex, so becoming dominated in the whole divider.

### *4.2.3 Delay*

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 whole divider 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 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 a number of 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 average delays of the different approximate schemes versus bit width and radix are plotted in Figure 9; as the radix increases, the delay decreases, because a high radix divider requires a fewer number of add-subtract iterations, hence the critical path delay is also shorter. However, the delay of the radix-8 divider is higher than for radix-4; as the radix increases, the SU, QS and PG are more complex and the delay is dominated by QS and PG at each iteration stage. The advantage of fewer number of iteration stages is therefore not of a significant impact. A pipelined version of the high-radix divider can be realized by inserting flip-flops at the output of each adder stage to increase the overall computation throughput.

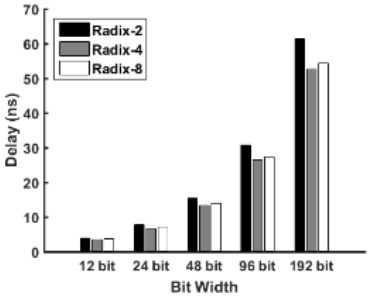



**Figure 9. Average Delay with different radix**

### *4.2.4 Trade-off between NED and Power*

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 [4]. In this paper the NPP (NED power product) is used as more relevant than the MPP. Figure 10 show the NPP of the 12-bit radix-2, radix-4 and radix-8 dividers using different approximation schemes; the replacement scheme has the lowest NPP compared to the other two schemes.

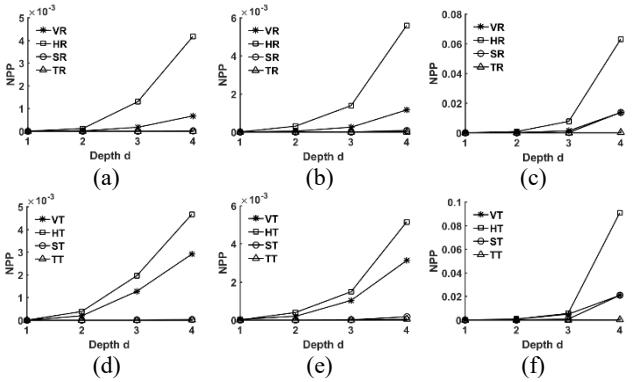



**Figure 10. NPP of 12-bit HR-AXD with (a)(d) Radix-2 (b)(e) Radix-4 (c)(f) Radix-8**

## 5. APPLICATIONS

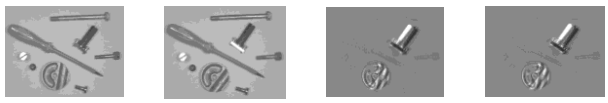
In this section, the approximate schemes for high radix division are evaluated using different applications involving image analysis (on a pixel basis). The proposed high-radix dividers are assessed for pixel division applications. For image analysis, the input gray scale images are normalized in the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

the approximations used in these applications are shown in Table 2; these configurations are chosen such that the power dissipation of these dividers is nearly the same.

**Table 2. Approximation depth *d* configurations of 12-bit HR-AXD Used for Application Analysis**

| | VR/VT | HR/HT | SR/ST | TR/TT |
|---|---|---|---|---|
| Radix-2 | 2 | 2 | 5 | 7 |
| Radix-4 | 3 | 2 | 4 | 5 |
| Radix-8 | 4 | 2 | 4 | 3 |

*Change detection:* The fractional change or ratio between two frames of a sequence of images is used for change detection. If there is no movement in the scene, then the output image mostly consists of one value 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. After calculating the pixel division, the resulting pixels are scaled up and rounded to the integer range [0,255] to display the resulting images. Figure 11 shows an example of the results for change detection of a sequence of two frames X and Y.



**X** **Y** **HR-EXD** **TR**

**Figure 11. An example of TR HR-AXD for change detection**

*Background removal:* In this case, background variations in illumination are divided from a scene, such that the foreground objects can be better viewed. For example, the image X in Figure 12 shows some text that has been badly illuminated during capture (i.e. there is a strong illumination gradient across the image). If a blank page Y is divided from the poorly illuminated image X, the output has a relatively constant illumination. Following this operation, a simple step for the threshold can be used to produce a high-contrast text image. Figure 12 shows an example of the results.

**X** **Y** **HR-EXD** **TR**

**Figure 12. An example of TR HR-AXD for background removal**

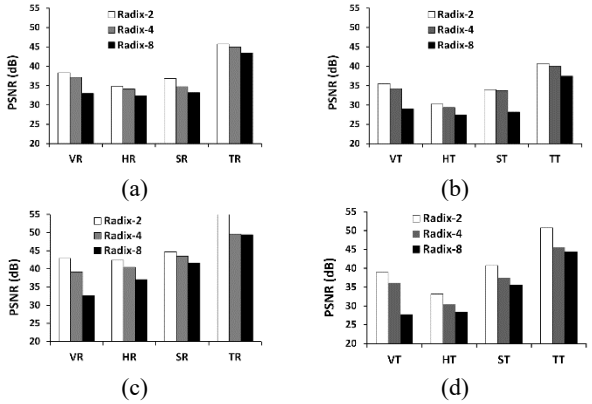



**Figure 13. PSNR of 12-bit HR-AXD for image application Change detection with replacement (a) and truncation (b); Background removal with replacement (c) and truncation (d)**

Figure 13 shows the Peak Signal-to-Noise Ratio (PSNR) for change detection and background removal. In both cases, the triangle approximations have the best PSNRs; moreover, the PSNR is lower for higher radix dividers. A truncated scheme has a lower PSNR than the corresponding replacement scheme.

# 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 following conclusions can be drawn: A larger value for $d$ provides a larger NED; among all schemes, the triangle replacement divider has the best NED among the replacement schemes. A truncated scheme introduces more error. The power consumption reduces rapidly as the depth increases, i.e. the higher the depth is, more pronounced is the power reduction. A truncation scheme provides a significant power reduction compared to a replacement scheme. The approximation schemes have only a small impact on the delay of the divider; the delay is reduced for higher radix dividers, because it takes only $N/m$ stages to complete the division. This advantage is reduced when the radix is higher than 8. Compared to radix-2 division, a high radix approximate divider is faster; its power dissipation is lower for radix-4 and radix-8, although it increases for radix values greater than 8. In conclusion, when designing an approximate array divider, metrics (and related design parameters) have to be considered and met as per the application. This paper has shown that a triangle-based replacement scheme at a moderate radix value (not higher than 8) is the best approximate divider scheme to achieve low power consumption, high speed and a small error for an application such as image division.

# REFERENCES


[1] S. F. Oberman and M. Flynn, "Division algorithms and implementations," *Computers, IEEE Transactions on,* vol. 46, pp. 833-854, 1997.

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

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

[4] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the Design of Approximate Restoring Dividers for Error-Tolerant Applications," *Computers, IEEE Transactions on,* vol. PP, pp. 1-1, 2015.

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

[6] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Transactions on Electronic Computers,* vol. EC-10, pp. 389-400, 1961.

[7] S. F. Oberman and M. Flynn, "Division algorithms and implementations," *IEEE Trans. Comput.,* vol. 46, pp. 833-854, 1997.

[8] M. D. Ercegovac, T. Lang, and P. Montuschi, "Very-high radix division with prescaling and selection by rounding," *IEEE Trans. Comput.,* vol. 43, pp. 909-918, 1994.

[9] P. Montuschi and T. Lang, "Boosting very-high radix division with prescaling and selection by rounding," *IEEE Trans. Comput.,* vol. 50, pp. 13-27, 2001.

[10] M. D. Ercegovac and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations," *IEEE Trans. Comput.,* vol. C-36, pp. 895-897, 1987.

[11] L. Jinghang, H. Jie, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Trans. Comput.,* vol. 62, pp. 1760-1771, 2013.