Design of Approximate Unsigned Integer Non-restoring Divider for Inexact Computing

Linbin Chen
ECE Department
Northeastern University
Boston, MA USA
chen.lin@husky.neu.edu

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

Fabrizio Lombardi
ECE Department.
Northeastern University
Boston, MA USA
lombardi@ece.neu.edu

ABSTRACT

This paper proposes several approximate divider designs; two different levels of approximation (cell and array levels) are investigated for non-restoring division. Three approximate subtractor cells are proposed and designed for the basic subtraction; these cells mitigate accuracy in subtraction with other metrics, such as circuit complexity and power dissipation. At array level, by considering the exact cells, both replacement and truncation schemes are introduced for approximate array divider design. A comprehensive evaluation of approximation at both cell and divider level is pursued. Different circuit metrics including complexity and power dissipation are evaluated by HSPICE simulation. Mean error distance (MED), normalized error distance (NED) and MED-power product (MPP) are provided to substantiate the accuracy and power trade-off of inexact computing. Different applications in image processing are investigated by utilizing the proposed approximate arithmetic circuits.

Categories and Subject Descriptors

B.7.1 [**INTEGRATED CIRCUITS**]: Types and Design Styles – *Algorithms implemented in hardware*.

General Terms

Design, Performance.

Keywords

Inexact computing, Division, Error Distance, Power Dissipation

1. INTRODUCTION

Most computer arithmetic applications are implemented using digital logic circuits, thus operating with a high degree of reliability and accuracy. However, many applications such as multimedia and image processing can tolerate errors and imprecision in computation and still produce meaningful and useful results. The paradigm of inexact computation relies on relaxing fully precise and completely deterministic building modules when for example, designing energy efficient systems. This allows imprecise computation to redirect the existing design process of digital circuits and systems by taking advantage of a decrease in complexity and cost with possibly a potential increase in performance and power efficiency.

GLSVLSI'15, May 20–22, 2015, Pittsburgh, PA, USA.

http://dx.doi.org/10.1145/2742060.2742063

Inexact computing is well suited for arithmetic circuits such as adders and multipliers. Recently, five approximate mirror adders (AMAs) have been investigated by logic reduction at transistor level [1]. The three approximate XOR-based adders (AXAs) of [2] show attractive operational profiles for performance, hardware efficiency and power-delay product (PDP), while retaining a good accuracy. Approximate multipliers based on approximate adders have also been proposed, since a multiplier is usually implemented by cascading multiple adders. Some of the less significant bits in the partial products can be truncated [3] while providing error compensation mechanisms; this type of scheme removes some of the adders for a faster execution of this operation. In [4], a simplified 2×2 bit multiplier is used as modular block of a multiplier with larger operand size. An efficient multiplier design using input pre-processing and additional error compensation is proposed for reducing the critical path delay in [5]. To the authors' best knowledge, there have not been any technical literature on an approximate divider (AXD) design. In this paper, the approximate subtractor (AXS) is initially considered as a first level of approximation for a non-restoring AXD (AXDnr). New AXDnr cells (AXDCnr) are proposed and the approximate operation is carried further at divider level by considering different schemes by which exact cells can be replaced by approximate cells or truncated (removed). A comprehensive evaluation of approximation at both cell- and divider-levels is pursued using HSPICE simulation at PTM 32nm technology; various circuit metrics (such as complexity, and power dissipation) are evaluated. MED, NED and MED Power Product are provided to substantiate the accuracy and power trade-off of inexact computing. Different applications in image processing are investigated by utilizing the proposed approximate arithmetic circuits. As for image processing, approximate division often results in a very marginal decrease in quality (as measured by metrics such as PSNR).

2. REVIEW

2.1 Exact Subtractor (EXS)

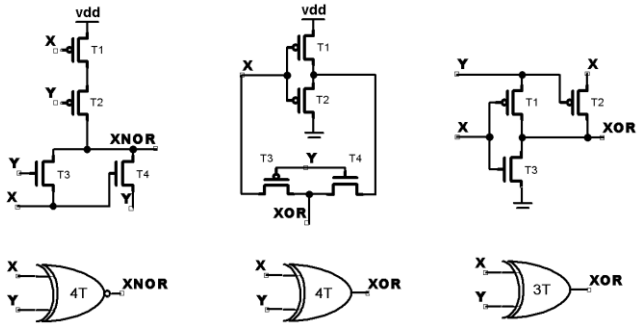
The functions of an exact subtractor cell (EXSC) and exact full adder cell (EXAC) are quite similar (Table 1). D denotes the difference between X and Y, B_{in} is the borrow bit to the lower position and B_{out} is the borrow bit from the higher position. So, the discussion of EXSC is also applicable to EXAC. An unsigned N-bit ripple EXS can be implemented using N cascading EXSCs.

There are three basic Boolean operations in EXSC and EXAC: XOR/XNOR, AND and OR. Two XOR/XNOR cascaded gates generate the outputs S or D; two AND and one OR gates generate C_{out} or B_{out}. At circuit-level the XOR can be implemented using 3 to 4 transistors (Figure 1(b) or (c)). The XNOR can be implemented using 4 transistors (Figure 1(a)). XOR and XNOR can be used interchangeably to implement the sum output S and

the subtraction output D. The AND and OR functions can be implemented using a 2-transistor MUX (Figure 2).

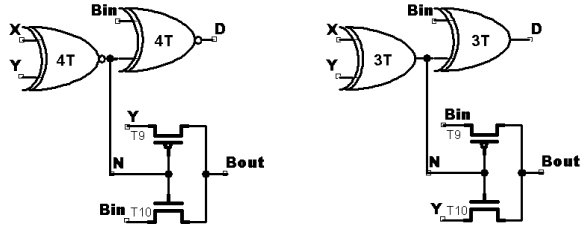
Table 1 EXAC and EXSC Functions

	S or D	C_{out} or B_{out}
EXAC	$S = X \oplus Y \oplus C_{in}$	$C_{out} = X \oplus Y \cdot C_{in} + XY$
EXSC	$D = X \oplus Y \oplus B_{in}$	$B_{out} = \overline{X \oplus Y} \cdot B_{in} + \bar{X}Y$



(a) (b) (c)

Figure 1 Three implementations of XOR and XNOR gates and related symbols. (a) 4T XNOR from [6] (b) 4T XOR from [7] and (c) 3T XOR from [8]



(a) (b)

Figure 2 Implementations of an EXSC (a) based on a 4T XNOR and (b) based on a 3T XOR

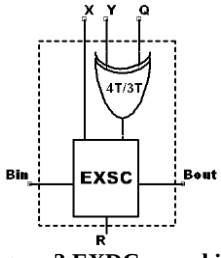
Similar to the EXACs in [2], the implementations of the EXSC are shown in Figure 2; when simulating by HSPICE, the signal integrity may be degraded due to the pass transistor logic; thus, a buffer is inserted to preserve an acceptable signal integrity.

2.2 Exact Non-Restoring Divider (EXDnr)

For integer division, the operands are the dividend X and the non-zero divisor Y, and the results of the operation are the quotient Q and the remainder R, i.e.

$$X = YQ + R$$

where the sign of the remainder R is the same as the dividend X and $|R| < |Y|$. A common division algorithm is **non-restoring division** [9].



$$R = X \oplus (Y \oplus Q) \oplus B_{in}$$
$$B_{out} = \overline{X \oplus Y \oplus Q} \cdot B_{in} + \bar{X}(Y \oplus Q)$$

Figure 3 EXDCnr and its logic functions using EXSC

Exact Non-Restoring Divider Cell (EXDCnr): An EXDCnr is made of an EXSC and a XOR gate (Figure 3). Q is a control signal such that when $Q = 0$, EXDCnr is configured as an exact full subtractor (X-Y); when $Q = 1$, EXDCnr is configured as an exact full adder (X+Y). This configurable divider cell is the basic building block for an EXDnr.

Exact Non-restoring Divider (EXDnr): A 8-by-4 bits unsigned EXDnr is shown in Figure 4; it computes the unsigned integer division for X[7:0], Y[3:0], Q[3:0] and R[3:0].

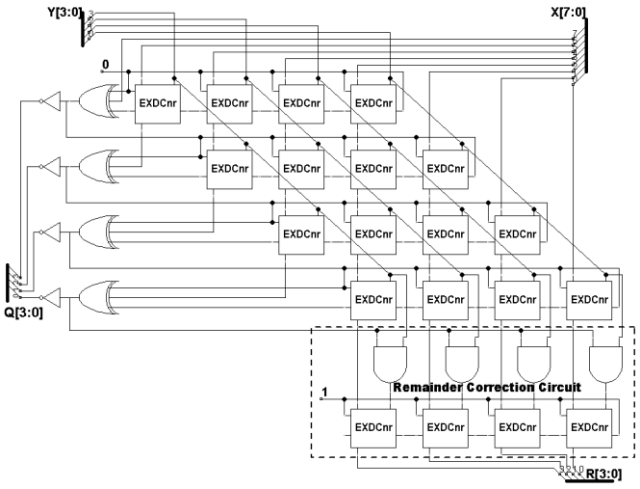
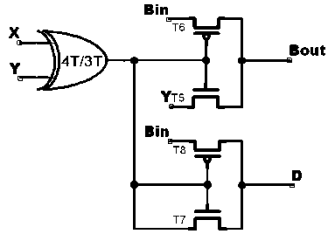


Figure 4 8-to-4 unsigned EXDnr

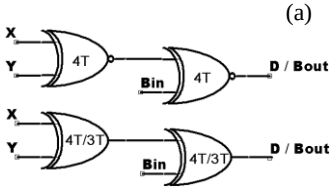
3. APPROXIMATE SUBTRACTOR (AXS)

3.1 Approximate Subtractor Cell (AXSC)



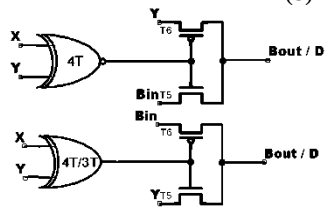
$$D = X \oplus Y + B_{in}$$
$$B_{out} = \overline{X \oplus Y} \cdot B_{in} + \bar{X}Y$$

(a)



$$D = X \oplus Y \oplus B_{in}$$
$$B_{out} = D \text{ or } B_{out} = B_{in}$$

(b)



$$D = B_{out}$$
$$B_{out} = \overline{X \oplus Y} \cdot B_{in} + \bar{X}Y$$

(c)

Figure 5 (a) AXSC1 (b) AXSC2 and (c) AXSC3

Three types of AXSC (AXSC1-AXSC3) are introduced next; their diagrams and functions are shown as Figure 5; two versions (the XOR and XNOR based designs) are proposed for some cells (AXSC2, AXSC3). There are two XOR/XNORs in an EXSC; each of them consists of 4 or 3 transistors. Therefore, the elimination of one of them in an AXS design is an obvious choice for reducing the number of transistors (AXSC1 and AXSC3 abide by this consideration). In the operation of a subtractor, the accuracy of B_{out} is in general more important than D; so, in an AXSC design, B_{out} is unchanged (as in AXSC1 and AXSC3). To

reduce the delay, the output signal D and B_{out} can be combined (as occurring in AXSC2 and AXSC3).The truth table for each input combination of these approximate cells is shown in Table 2. The number of transistors and the error distance are shown in Table 3.

Table 2 Truth Table of 3 Proposed AXS Cells

X	Y	B_{in}	EXSC		AXSC1		AXSC2		AXSC3	
			B_{out}	D	B_{out}	D	B_{out}	D	B_{out}	D
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1
0	1	1	1	0	1	**1**	**0**	0	1	**1**
1	0	0	0	1	0	1	**1**	1	0	**0**
1	0	1	0	0	0	**1**	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1

Table 3 Transistor Count and Error Distance of AXSCs

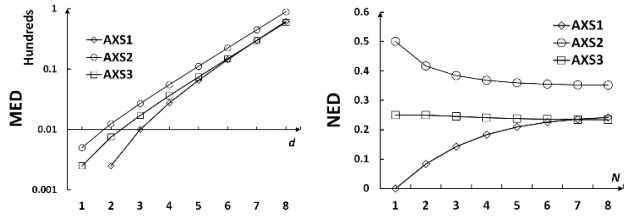
Subtractor Cell	Transistor Count	Error Distance
EXSC	10	0
AXSC1	8 or 7	2
AXSC2	8, 7 or 6	4
AXSC3	6 or 5	2

3.2 Approximate Subtractor (AXS)

In this section, an N-bit AXS is evaluated. The three types of AXSC are used to replace some of the LSBs of an EXS. The replacement depth *d* is used to represent the number of the EXSCs replaced by the AXSCs.

3.2.1 MED and NED

In addition to the error distance (ED), the mean error distance (MED) and the normalized error distance (NED) have been proposed [10] by considering the averaging effect of multiple inputs and the normalization of multiple-bit adders. The MED is defined as the mean value of the EDs of all possible outputs for each input. The NED is defined as the MED normalized by the maximum ED that a design incurs. The MED is effective in evaluating multiple-bit approximate arithmetic circuit, while the NED is nearly invariant with the size of an implementation and therefore, is useful in the assessment of a specific type of design.



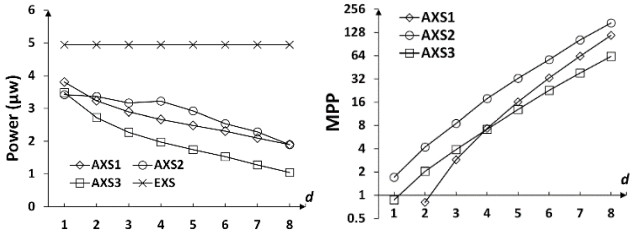
(a) (b)

Figure 6 (a) MED of 8-bits AXS with depth *d*, and (b) NED of N-bit AXS with depth *d*=N

The MEDs for 8-bit AXS with replacement depth *d* are shown in Figure 6(a) on a logarithmic scale; AXS1 and AXS3 have the smallest MED, because they nearly preserves the exact computation for subtraction. The NEDs of the proposed AXS with different AXSCs and replacement depth *d=N* are plotted in Figure 6(b); the value of the NED for different AXS reduces to a constant as the width of the subtractor increases; AXS1 and AXS3 have relative smaller NEDs compared to AXS2.

3.2.2 MED Power Product (MPP)

MPP is used to assess the trade-off between computational accuracy and power consumption for approximate computing circuits. Figure 7(a) shows the dynamic power by exhaustive simulation using AXSCs in AXS. As the replacement depth *d* increases, the power consumption decreases. The switching activity of AXS2 is higher (different from AXS1 and AXS3), hence the power of AXS2 is not as good as the other two schemes. MPP is shown in Figure 7 (b). AXS1 has the best MPP when *d* is low, but it deteriorates at higher values of *d*. When *d* increases, AXS3 is the best scheme. On average, the MPP shows that AXS3 is the best choice for an AXS.



(a) (b)

Figure 7 (a) Power and (b) MPP of 8-bit AXS

4. APPROXIMATE NON-RESTORING DIVIDER (AXDnr)

4.1 AXDCnr

The design of an AXDCnr takes advantage of the corresponding AXSC, because subtraction is the basic operation of division. The corresponding AXDCnr1-3 can be derived by substituting the EXSC in Figure 3 with AXSC1-3 presented in previous section.

4.2 AXDnr

4.2.1 Replacement Scheme

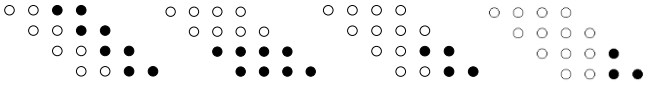
A simple divider-level approximation is to use cell replacement, *i.e.* to replace some EXDCnrs in the EXDnr with the AXDCnrs developed in Section 4.1. The approximate cells have a smaller circuit complexity, so making the entire divider less complex and consuming less power. These advantageous features are accomplished at the expenses of introducing errors at the outputs for the quotient Q and/or the remainder R. The errors in Q and R are correlated. So an approximate divider can be designed to have either a more accurate Q with a less accurate R (division), or a more accurate R with a less accurate Q (modulo). Approximation is therefore the process by which 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 array with approximate cells. Four types of approximation are used (Figure 8) for the division operation for the accuracies of Q and R.

Vertical Replacement (VR): The least significant (LSB) EXDCnrs in each row of the divider are replaced by AXDCnrs. The depth of the vertical replacement can be increased to further decrease the power while tolerating more errors in the output. Hence, $4\times d$ EXDCnr cells are replaced with AXDCnr cells in a divider with a vertical approximation of depth *d*. One example of depth *d*=2 is shown in Figure 8(a).

Horizontal Replacement (HR): The value of the quotient is mostly related to the borrow signal of each cell in a single row. For example, consider the last row corresponding to the LSB of Q; if the value of R is not of significant concern, then all EXDCnrs in the last row can be replaced with AXDCnrs without losing the accuracy of Q. If some error can be tolerated at 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 8(b).

Square Replacement (SR): By combining the vertical and horizontal replacement; a new configuration is referred to as the square replacement. Hence, in a d^{th} AXDnr, d^2 EXDCnrs are replaced with AXDCnrs. An example of a square replacement AXDnr of depth *d=2* is shown as Figure 8(c).

Triangle Replacement (TR): Consider the integer pair (*i,j*) as coordinates of each cell in a divider (Figure 9). An EXDCnr *(i,j)* (*i<d* or *j<d*) is replaced by a AXDCnr. So, in a triangle approximation divider with depth *d* (*d*≥1), *d(d+1)/2* EXDCnrs are replaced with AXDCs. An example of a triangle approximation divider with *d=2* is shown in Figure 8(d).



(a) (b) (c) (d)

Figure 8 Four replacement types of 8-to-4 bit AXDnr: (a) VR, (b) HR, (c) SR, and (d) TR. The replaced cells are shaded.

4.2.2 Truncation Scheme

Truncation consists of fully removing at least a cell (so no replacement with AXDCnr(s)); this process is shown by the shaded cells in Figure 8. The input X of the removed cell is left unchanged and moved downwards along the remainder output direction, while the input Y of the removed cell is discarded. Similar to replacement scheme, a truncation scheme has also four configurations: Vertical Truncation (VT), Horizontal Truncation (HT), Square Truncation (ST) and Triangle Truncation (TT).

4.3 Evaluation of AXDnrs

4.3.1 Error Estimation of AXDnrs

Error generation and propagation can be qualitatively analyzed by the location of the replaced or truncated cells in the divider. A so-called *error impact coefficient* (denoted by *S*) is proposed to simplify the analysis for finding the impact of the inexact or truncated cells in the array divider. An approximate scheme consists of replacing or truncating only a portion of the divider; this leads to different approximate configurations with different accuracy for the output values (Q and R). In a divider, the dividend X and the divisor Y are provided as inputs at the north side, while the quotient Q and the reminder R are generated at the west and the south sides respectively (Figure 9). So, each cell (located at a unique position in the divider) plays a different role in generating an error; intuitively, cell replacement or truncation must not preferably occur at the MSBs of Q and R. Therefore, the coefficients $S_{Q,(ij)}$ and $S_{R,(i,j)}$ are given by

$$\begin{cases} S_{Q,ij} = \frac{1}{\sqrt{(i-N)^2+(j-2N+1)^2}} \\ S_{R,ij} = \frac{1}{\sqrt{(i+1)^2+(j-N)^2}} \end{cases} \quad (i \in [1,N), j \in [1,2N-1))$$

in which i, j are the coordinates of the cell in the divider ((0,0) is the origin, as shown in (Figure 9). $S_{Q,(ij)}$ is effectively the distance from the MSB cell of Q to a cell position in the divider; $S_{R,(i,j)}$ is the distance from the MSB cell of R to a cell position in the array. Larger $S_{Q,(i,j)}$ or $S_{R,(i,j)}$ is, a greater error is contributed by the (i,j) cell to the final outputs Q_{error} and R_{error}. So, the impacts of errors at the outputs Q and R are given by

$$S_Q = \sum S_{Q,ji} \quad S_R = \sum S_{R,ij}$$

where, $S_{Q,ij}$ and $S_{R,ij}$ are due to cell (i,j) when using an approximate implementation. Figure 10 illustrates the coefficients S_Q and S_R of the replacement or truncation configurations at different depths. The trends for S_Q and S_R are consistent with the NEDs plotted in Figure 11; so, S_Q and S_R are good qualitative metrics for analyzing different replacement or truncation schemes.

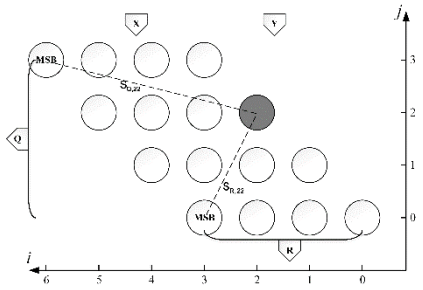
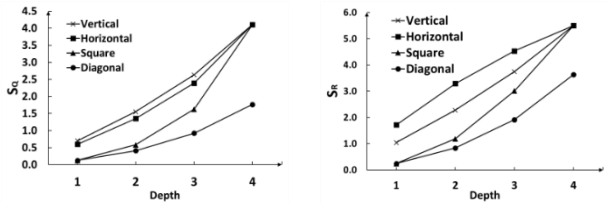


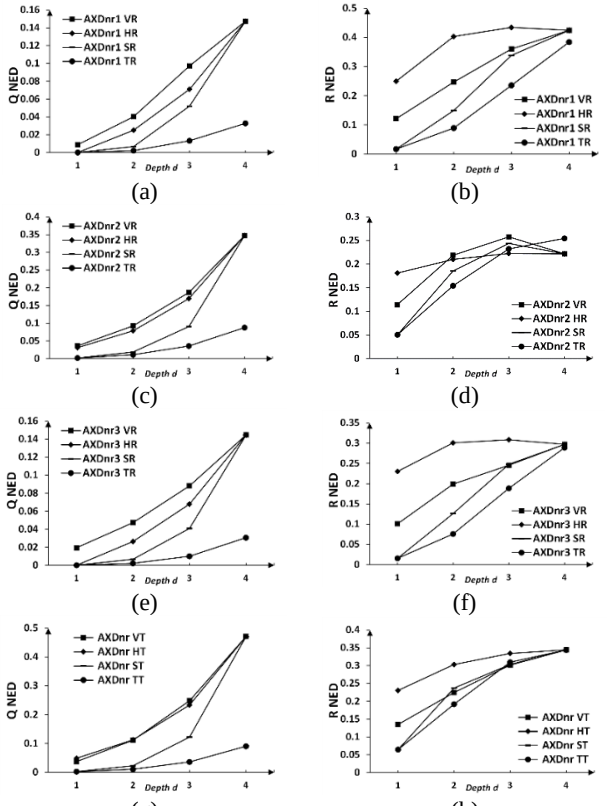
Figure 9 Error impact coefficients S_Q and S_R



(a) (b)

Figure 10 Error impact coefficients of different replacement or truncation configurations vs depth: (a) S_Q and (b) S_R

4.3.2 Normalized Error Distance (NED)



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 11 NED of AXDnrs for Quotient Q and R (a)(b) AXDnr1, (b)(c) AXDnr2, (d)(e) AXDnr3, (g)(h) Truncation

For a more accurate evaluation of the different approximate designs, the NED has been simulated at different depths (Figure 11). Among the replacement schemes, the error of AXDnr2 is the largest in all cases for Q, but not for R; this is due to the worst (best) accuracy for B_{out} (D) in AXSC2. AXDnr1 and AXDnr3

have on average the smallest NED for both Q and R. Among truncation schemes, the NED of Q is larger than any of its corresponding replacement schemes, because more error is introduced by simply removing a cell compared with replacing it with the proposed AXDC. For both replacement and truncation schemes, the NED of Q for the vertical and horizontal approximations increases rapidly with depth, more than for the square and triangle approximations; the triangle approximation shows the smallest NED for Q compared to the other replacement schemes. The NED of R also increases with replacement depth; moreover, it is larger than the NED of Q. This occurs because the error in R is strongly dependent on the error in Q; however when using a horizontal replacement, the NED of R is less dependent on the replacement depth.

4.3.3 Power consumption

Figure 12 shows the power consumption of AXDnrs versus replacement depth. All AXDnrs consume less power as the replacement depth increases; this is significantly lower than the power consumed by EXDnrs. The power dissipation of the truncation schemes may be slightly larger than the replacement schemes when the depth is small, however the power of a truncation scheme decreases at the highest rate.

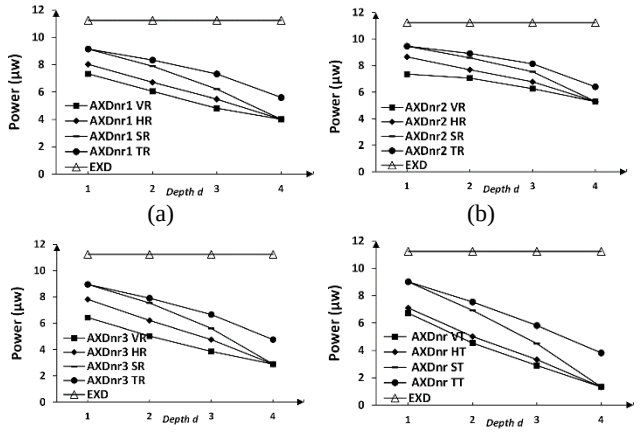


Figure 12 Power consumption of AXDnrs for Quotient Q (a) AXDnr1 (b) AXDnr2 (c) AXDnr3 and (d) Truncation

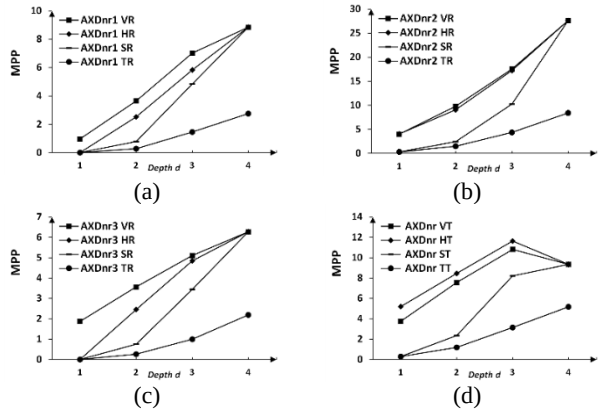


Figure 13 MED Power Product of AXDnrs for Quotient Q (a) AXDnr1 (b) AXDnr2 (c) AXDnr3 and (d) Truncation

4.3.4 MED Power Product (MPP)

To evaluate the tradeoff between computation accuracy and power consumption of the AXDnrs, the MPP of the AXDnrs is calculated and plotted in Figure 13. Although the power saving of a truncation scheme is larger than for a replacement scheme, this is at the expense of the error; hence, the triangle replacement scheme with AXDnr3 has the smallest MPP. AXDnr3 is very promising for an approximate divider design requiring both high accuracy and low power consumption. AXDnr2 is again shown to be the worst design regardless of the type of replacement. Compared to a replacement scheme, a truncation scheme is not suitable for both high accuracy and low power AXDnr designs.

5. APPLICATIONS

5.1 Pixel Subtraction

Pixel subtraction takes two images as input and produces as output a third image whose values are the subtraction of the second image from the first image on a pixel basis. 8-bit ripple borrow EXS and AXS with replacement depth d=4 are used.

Background Removal: In this case, background variations in illumination are subtracted from a scene, such that the foreground objects can be better viewed. For example, the image X in Figure 14 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 subtracted from the poorly illuminated image X, the output has a relatively constant illumination. Figure 14 shows the simulation results. The results of AXS1 and AXS3 are better.

Figure 14 Output images for background removal

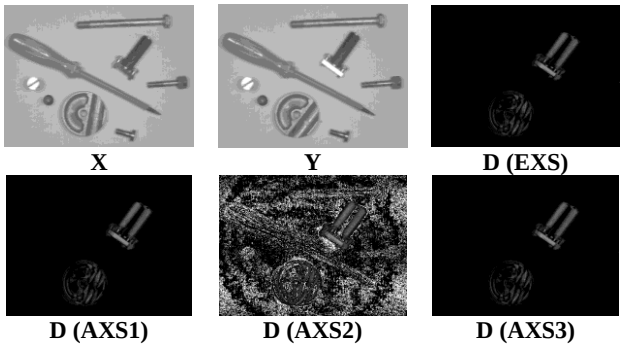


Figure 15 Output images for change detection

Change Detection: The image difference is also used for change detection. If the difference between two frames of a sequence of images is calculated and there is no movement in the scene, then the output image mostly consists of zero value pixels. If there is movement, then the pixels in those regions of the image in which the intensity spatially changes, exhibit significant differences between the two frames. Figure 15 shows the change detection results of a sequence of frames X and Y. AXS1 and AXS3 are capable of detecting the moving part of the image.

The peak signal to noise ratio (PSNR) is commonly used to measure the quality of lossy compression codecs of images. In this case, it is used to assess the quality of the images processed by an inexact arithmetic circuit compared to the ones by an exact arithmetic circuit. For the two image subtraction applications considered in this manuscript, the PSNR results are shown in Figure 16. AXS1 and AXS3 based subtractors have the larger PSNR, so generating the best approximation and the least image errors when replacing EXSCs with AXSCs.

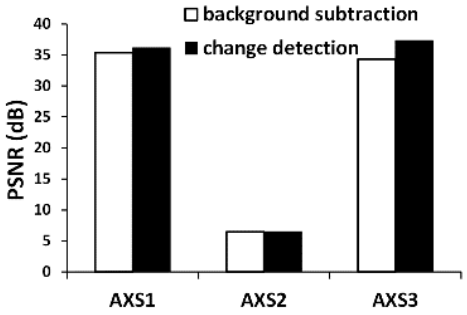


Figure 16 PSNR of AXSs for pixel subtraction

5.2 Pixel Division

The change detection and background removal application can be also realized by division operation. In image analysis, if only integer division is performed, then the results are typically rounded at the output to the next lowest integer. 16-to-8 EXDnr and AXDnr are used to compute the same 8-bit grayscale images as in the previous section; the approximations used in these applications are with depth 3 for VR and HR, depth 2 for VT and HT, depth 5 for SR, depth 4 for ST, depth 7 for TR and depth 5 for TT. These configurations are selected to ensure that the power consumptions are nearly equal.

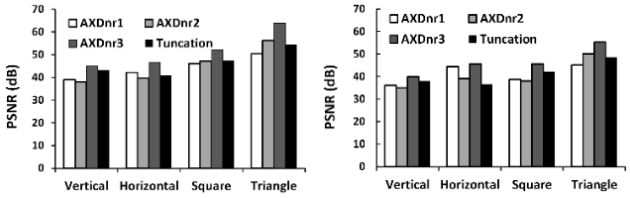


Figure 17 Simulated PSNR of pixel division (a) change detection (b) background removal

It can be seen from both change detection and background removal results of Figure 17 that the triangle replacement with AXDnr3 has the best PSNRs while the vertical replacement has overall the worst PSNR.

6. CONCLUSION

This paper has presented a detailed analysis, design and evaluation of dividers that utilize approximate criteria in their operation. As basic operation for division, subtraction has been initially considered. Subsequently, AXDnr have been proposed by introducing approximate computing at cell as well as at divider levels. Division presents unique challenges for approximate computing; it generates two output values (the remainder R and the quotient Q), so changes in cells as well as the dividers may affect one of the outputs more than the other. Hence, different arrangements in the approximate cells have been investigated; these arrangement replaces exact with cells or truncates (eliminates) them according to the direction of computation flow. The following conclusions can be drawn. (1) The error in Q for different approximation configurations increases with the replacement or truncation depth; the triangle approximation increases at a smaller rate. (2) AXDnr1 and AXDnr3 have on average the smallest NED for Q. However, the error introduced by a truncation scheme is significantly more than a replacement scheme. (3) The error in R due to replacement is significantly larger than the error in Q because R is dependent on Q of previous stage. (4) Power consumption is reduced considerably when the approximation depth increases. Higher the depth, more pronounced is the power reduction. (5) For all depths, AXDnr1 and AXDnr2 consume more power than AXDnr3; the truncation schemes require considerably less power than the replacement schemes. (6) The trade-off between accuracy and power consumption makes an approximate divider design an application specific process. For an application requiring both high accuracy and low power, the MED power product metric shows that AXDnr3 is a good approximate scheme. For Q-oriented applications (image processing), AXDnr3 with a triangle replacement shows the best results.

7. ACKNOWLEDGMENT

This research is supported in part by an NSERC Discovery Grant and by a grant from NSFC (No. 61401197).

8. REFERENCES

[1] Gupta, V., Mohapatra, D., Park, S.P., Raghunathan, A. and Roy, K. 2011. IMPACT: IMPrecise Adders for Low-Power Approximate Computing. In *Proceedings of the International Symposium on Low Power Electronics and Design* (1-3 Aug. 2011), 409-414.

[2] Yang, Z., Jain, A., Liang, J., Han, J. and Lombardi, F. 2013. Approximate XOR/XNOR-based Adders for Inexact Computing. In *Proceedings of the 13th IEEE Conference on Nanotechnology* (5-8 Aug. 2013), 690-693.

[3] Kyaw, K.Y., Goh, W.-L. and Yeo, K.-S. 2010. Low-Power High-Speed Multiplier for Error-Tolerant Application. In *Proceedings of the International Conference of Electron Devices and Solid-State Circuits* (15-17 Dec. 2010), 1-4.

[4] Kulkarni, P., Gupta, P. and Ercegovac, M. 2011. Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In *Proceedings of the 24th International Conference on VLSI Design* (2-7 Jan. 2011), 346-351.

[5] Momeni, A., Han, J., Montuschi, P. and Lombardi, F. 2014. Design and Analysis of Approximate Compressors for Multiplication. *IEEE Trans. Comput.* (Accepted to appear).

[6] Lin, J., Hwang, Y.-T., Sheu, M.-H. and Ho, C.-C. 2007. A Novel High-Speed and Energy Efficient 10-Transistor Full Adder Design. *IEEE Trans. Circuits Syst. I, Reg. Papers 54*, 5, 1050-1059.

[7] Mahmoud, H.A. and Bayoumi, M.A. 1999. A 10-Transistor Low-Power High-Speed Full Adder Cell. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (Jul 1999), 43-46.

[8] Chowdhury, S.R., Banerjee, A., Roy, A. and Saha, H. 2008. A High Speed 8 Transistor Full Adder Design Using Novel 3 Transistor XOR Gates. *International Journal of Electronics, Circuits and Systems 2*, 4, 217-223.

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

[10] Liang, J., Han, J. and Lombardi, F. 2013. New Metrics for the Reliability of Approximate and Probabilistic Adders. *IEEE Trans. Comput. 62*, 9, 1760-1771.