Design of Unsigned Approximate Hybrid Dividers based on Restoring Array and Logarithmic Dividers

Weiqiang Liu, *Senior Member, IEEE,* Tao Xu, Jing Li, Chenghua Wang, Paolo Montuschi, *Fellow, IEEE,* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract—Approximate computer arithmetic has been extensively studied due to its advantages to further reduce power consumption and increase performance at reduced accuracy. Although a number of approximate adders and multipliers have been studied, only a few approximate dividers have been proposed. A logarithmic divider (LD) has low complexity and accuracy, while an exact array divider (EXD) has a high complexity. Therefore, in this paper, an approximate hybrid divider (AXHD) is proposed. It takes advantage of both LD and EXD to achieve a tradeoff between hardware performance and accuracy. Exact restoring divider cells are used to generate the most significant bits (MSBs) of the quotient for attaining a high accuracy while the other quotient digits are generated by using a LD as an approximate scheme to improve figures of merit such as power consumption, area and delay. To further save hardware resources, a so-called eliminated approximate hybrid divider (E-AXHD) based on AXHD is also proposed. In this improved design, a reduced width divider is used to replace the EXD in AXHD. Specifically, for a 16-by-8 design, $n/(n+1)$ array division is used to replace the $n/8$ array division $(n<8)$. The proposed AXHD and E-AXHD are evaluated and analyzed using error and hardware metrics. The proposed designs are also compared with EXD, LD and previous approximate dividers. The results show that the proposed designs outperform previous approximate dividers by considering both energy and error. The proposed hybrid dividers are of particular interest for error tolerant applications such as image processing and machine learning.

Index Terms—Approximate computing, logarithmic divider, restoring array divider, low Power.

1 Introduction

It is difficult to further reduce power dissipation and improve performance of integrated circuits (ICs) under the requirement of 100% accuracy. New computing paradigms have been investigated to deal with these challenges in IC design. The technical literature shows that over the years many efforts have been targeting the design of fast computing systems while generating an output with the highest possible precision. However, this also results in a high power consumption and a large hardware complexity. In many error tolerant or error resilient applications (such as digital signal processing, machine learning and computer vision), human perception can mitigate the effects of some computational errors. Therefore, approximate (or inexact) computing has been proposed as an innovative technique for the design of low power and high performance systems [1]- [5].

Approximate computer arithmetic has been studied due to its advantages to further reduce power consumption and increase performance at the cost of an acceptable accuracy. As fundamental operations of an arithmetic processor, addition and multiplication are very important for achieving high performance. Therefore, they have been extensively studied for approximate computing. Error metrics including the error rate (ER), error distance (ED), mean error distance (MED), mean relative error distance (MRED) [6] have been proposed for evaluating the designs of approximate arithmetic circuits. Approximate adders have been extensively studied in the technical literature; among the many proposed schemes, designs include both speculative [7]- [8] and non-speculative transistor-level full adders [9]- [10]. The operation of multiplication is more complex than addition. Approximate design techniques can be applied to different parts of a conventional multiplier, such as operands [13]- [14], partial product (PP) generation [15], PP tree [16]- [17] and compressors [18].

Compared to approximate adders and multipliers, approximate dividers have received less attention in the technical literature. However, an approximate divider has been proposed back to early 1960s [13]. The logarithmic divider (LD) proposed by Mitchell is an approximate divider because it introduces errors during the conversion from binary to logarithmic operands. Therefore, the division operation is transformed into a subtraction. As a result, the design complexity can be significantly reduced with improved performance at the expense of accuracy. However, LD introduces large errors which makes it less attractive for many applications that require high accuracy. Recently, several approximate dividers have been proposed by either

- *W. Liu, T. Xu, and C. Wang are with College of Electronic Information and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 211106. E-mail: {liuweiqiang, xutao666, chwang}@nuaa.edu.cn*
- *J. Li is with School of Information Science and Technology, ShanghaiTech University, Shanghai, China, 201210. E-mail: lijing1@shanghaitech.edu.cn*
- *P. Montuschi is with Control and Computer Engineering Department. Politecnico di Torino, Torino, Italy, 10129. E-mail: paolo.montuschi@polito.it*
- *F. Lombardi is with Department of Electrical and Computer Engineering, Northeastern University, Boston, USA, MA 02115. E-mail: lombardi@ece.neu.edu*

Manuscript received XX XX, 2020; revised XX XX, 2020.

simplifying exact designs [19]- [20] or using approximate operands [22]- [24].

The design of an approximate unsigned non-restoring divider (AXDnr) has been proposed in [19]. Different AXDnrs have been proposed by replacing the logic primitives with approximate subtractors. Three types of approximate subtractor cells (AXSCs) are then designed at transistor level. Exact subtractor cells, which are critical in the operation of the array divider, are truncated or replaced by AXSCs using various schemes. Both restoring and non-restoring array dividers have been analyzed for approximate computing. The main difference between the restoring and the non-restoring divisions is that the non-restoring type does not correct the partial remainder if a subtraction has a negative result. In hardware, the non-restoring divider has a remainder correction circuit to ensure that the sign of the remainder is consistent with the dividend. It has been shown that an approximate unsigned restoring divider (AXDr) has better performances than AXDnr with respect to power consumption, while also introducing a small degradation in accuracy [20]. The same replacement and truncation schemes of [19] have been applied to a restoring divider. A high-radix division scheme without look-up tables has been proposed in [21]. The replacement and truncation schemes from [19] have also been used to this high-radix array divider. Generally, approximate array dividers have high accuracy but also high complexity and lower performances due to its array structure. Furthermore, the delay is not significantly improved because only the less significant part is approximated.

A dynamic approximate divider (DAXD) has been proposed in [22]. Only those selected bits that start from the most significant "1" are used as the operands. For different lengths of input operands, leading one detectors and a barrel shifter are utilized to improve the accuracy. The lengths of the bits processed by the accurate divider are adjustable to reduce inaccuracy and power dissipation. Therefore, the divider size is reduced for low power and high performance. However, its accuracy is rather low due to the truncation of the operands; moreover, overflow may occur.

A round-based high speed yet energy-efficient approximate divider (SEERAD) has been proposed in [23]. This divider is based on the rounding of the divisor, so that the division operation can be performed by using only Shift and Add operations. This simplification highly reduces the delay at a very small loss in precision. However, as SEERAD uses the look-up tables, it requires large area and power consumption.

A fast and energy efficient truncation-based approximate divider (TruncApp) has been proposed in [24]. The division operation is performed by multiplying the truncated dividend by the approximate inverse of the divisor. TruncApp generally improves power consumption over SEERAD.

As discussed previously, by transforming the operands into the logarithmic domain, approximate division can significantly save power and area. However, its accuracy is rather low. Conventional array dividers are more accurate, but they also dissipate more power. Therefore, it is possible to take advantage of both schemes to design approximate hybrid divider (AXHD). This is accomplished by combining the exact restoring array divider and the LD as proposed in this paper. Exact restoring divider cells are used to correct the errors generated by a conventional logarithmic divider. The proposed approximate hybrid divider is evaluated comprehensively with error and hardware metrics. They are compared with existing designs. The results show that the proposed design outperforms previous designs at an appropriate replacement depth. Applications to image processing and the softmax function in neural networks are also provided to show the validity of the proposed designs.

This paper is an extension of our previous work that was published in [25]. The following new and important technical contributions have been added to this journal version:

- The proposed AXHDs are mathematically generalized and analyzed in Section 3.
- A new type of truncated approximate hybrid dividers (E-AXHDs) is proposed to further reduce the hardware complexity under the same error level as the AXHDs, in which a reduced width array divider is used.
- The proposed AXHDs are further compared with the latest approximate dividers including AXDrs [20], approximate operands based SEERADs [23] and TruncApps [24]. The results show the proposed designs with replacement depth of 12 outperforms all these previously proposed designs by considering both energy and error.
- The proposed approximate dividers are applied to compute the softmax function which is widely used in the deep neural networks, to show the validity of the proposed designs.

The paper is organized as follows. Conventional restoring array divider and the logarithmic divider are reviewed in Section 2. Section 3 presents the proposed designs of AXHD and E-AXHD in details, inclusive of analysis, architectures and examples. Error analysis and hardware evaluation of the proposed designs and comparison with AXDrs, LD and previous approximate dividers are provided in Section 4. The applications of the proposed design to image processing and the softmax function is presented in Section 5. Related works on approximate dividers and the differences between the proposed designs are illustrated in Section 6. The conclusions are provided in Section 7.

2 Review and Preliminaries

This section presents a brief review of conventional restoring array divider, and the LD, as basis for this work.

2.1 Exact Restoring Array Divider (EXDr)

Division is more complex than multiplication, which requires quotient digit selection and the detection of overflow. Considering an integer divider, assuming that X is a dividend and Y is a divisor (non-zero). The results of the division are Q and R. The expression of the divider can be written as:

$$X = YQ + R \tag{1}$$

where, the dividend X and the remainder R have the same sign and $|R| < |Y|$. The restoring divider [11] is a widely used type of array divider. The trial subtraction is performed in each row. The corresponding quotient digit is 1 (0) if the trial difference is positive (negative). If the quotient digit is 1, the trial difference is moved as partial remainder to the next row; otherwise, the partial remainder is immediately dropped. The term "restoring" indicates that the remainder is restored to the correct value if the trial subtraction was incorrect for the intermediate quotient digit. The restoring divider cell consists of an exact subtractor cell (EXSC) and two transistors as shown in Fig. 1; the expressions of EXDCr are given in Eqs. (2) and (3).

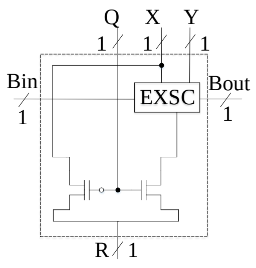


Fig. 1: An exact restoring divider cell (EXDCr) [20].

$$R = Q(X \oplus Y \oplus Bin) + \overline{Q}X \tag{2}$$

$$Bout = \overline{X \oplus Y}Bin + \overline{X}Y \tag{3}$$

An 8-by-4 exact restoring divider is shown in Fig. 2 [20]. Generally, a n-by-$n/2$ (n-bit dividend and $n/2$-bit divisor) restoring array divider consists of $n^2/2$ subtractors and its critical path delay is $O(n^2)$. Therefore, the array divider is rather complex and slow. The red lines in Fig. 2 show the critical paths.

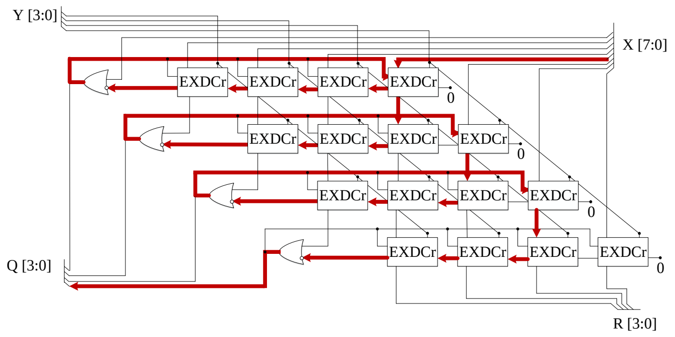


Fig. 2: A conventional unsigned 8-by-4 restoring array divider [20].

Fig. 3 shows the operation of an 8-by-4 exact restoring array divider. For example, consider $X = 157$ and $Y = 12$ as dividend and divisor, respectively. The expression of the quotient is as follows:

$$Q_3 = X_7 + \overline{Bout_3} \tag{4}$$

$$Q_n = R_{n+1} + \overline{Bout_n}, 0 \leq n \leq 2 \tag{5}$$

The quotient and the remainder obtained from the 8-by-4 exact restoring divider are 13 and 1, respectively.

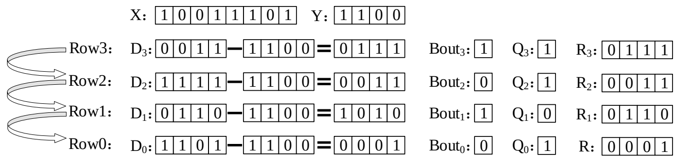

X: 1 0 0 1 1 1 0 1 Y: 1 1 0 0
Row3: D3: 0 0 1 1 − 1 1 0 0 = 0 1 1 1 Bout3: 1 Q3: 1 R3: 0 1 1 1
Row2: D2: 1 1 1 1 − 1 1 0 0 = 0 0 1 1 Bout2: 0 Q2: 1 R2: 0 0 1 1
Row1: D1: 0 1 1 0 − 1 1 0 0 = 1 0 1 0 Bout1: 1 Q1: 0 R1: 0 1 1 0
Row0: D0: 1 1 0 1 − 1 1 0 0 = 0 0 0 1 Bout0: 0 Q0: 1 R: 0 0 0 1

Fig. 3: Example of 8-by-4 exact restoring divider with X=157 and Y=12.

2.2 Logarithmic Divider (LD)

The operands of a LD are given by the dividend X and the divisor Y. X and Y are both positive integers, different from zero, because the cases when either X or Y, or both are zero are usually handled separately. k is the exponent, and m is the fractional part of the mantissa. They are expressed as follows:

$$X = 2^{k_1}(1 + m_1) \tag{6}$$

$$Y = 2^{k_2}(1 + m_2) \tag{7}$$

where, k_1 and k_2 are the so-called characteristics of X and Y, respectively and represent the position of the leading one bits. m_1 and m_2 are in the range of [0,1). The logarithm of a quotient, $i.e.$ Q, is equal to the difference of the logarithms of the dividend and divisor.

$$Q = \frac{X}{Y} = 2^{k_1 - k_2}\frac{1 + m_1}{1 + m_2} \tag{8}$$

$$\log_2 Q = k_1 - k_2 + \log_2(1 + m_1) - \log_2(1 + m_2) \tag{9}$$

As shown in [13], $\log_2(1 + m) \approx m$ when $0 \leq m < 1$, the approximate quotient can be expressed as follows:

$$\log_2 Q \approx k_1 - k_2 + m_1 - m_2 \tag{10}$$

Therefore, Eq. (10) can be calculated by subtractions. There are four steps in the conventional LD proposed by Mitchell [13]: leading one detection, binary-to-logarithm conversion, mantissa subtraction and logarithm-to-binary conversion. The leftmost one bit is detected by the leading one detector (LOD). The binary-to-logarithm converter (BLC) converts the input operands into logarithms. Only the most significant bit is converted to logarithm, the logarithm of the mantissa is aromatically truncated by default. The division is performed by a mantissa subtraction with the Subtractor in the logarithmic domain (Fig. 4), while the EXSC cells in Section 2.1 are employed to implement the subtractor. The logarithm-to-binary converter (LBC) converts the subtraction result back to a binary number as the final quotient. The LD is shown in Fig. 4 [13]. As the LD is inherently approximate, the remainder is not provided. Furthermore, conventional restoring array dividers overflow when X is large and Y is small, while the LD does not overflow.

3 Proposed Approximate Hybrid Dividers

In this section, the proposed approximate hybrid dividers (AXHD and E-AXHD) are presented in details.

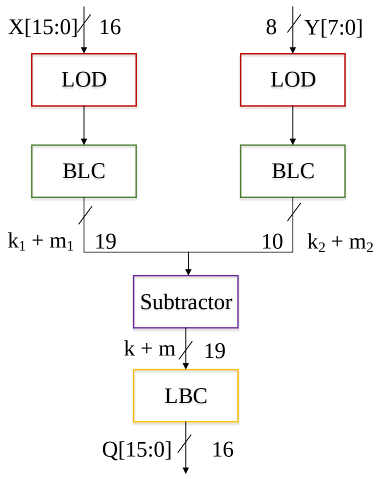


Fig. 4: A 16-by-8 unsigned logarithmic divider.

3.1 Derivation of the Proposed AXHD

A LD consumes significantly less power and requires less area than conventional array dividers [13], in which the rows for the subtraction operations require more hardware and area compared with the simple subtraction of a LD. However, when the operand width of the LD is large, the accuracy decreases due to the approximation in the logarithmic conversion. The conventional array dividers include restoring and non-restoring designs. As shown in [19]- [20], the hardware performance of the restoring array divider is better compared with the non-restoring array divider. Therefore, a new AXHD combining the exact restoring array divider and the LD is proposed in this section. Its design principle is to use the exact division for the most significant bits and an approximate logarithmic division for the remaining bits. The trade-off is therefore between accuracy and hardware performance (power, delay and area).

Let n be the total bit-width of the dividend ($i.e.$ X) and h be the least significant bits that will be sent to the LD (so defined as the replacement depth). For unsigned integer division, the dividend is firstly segmented into two parts: X_1 which is the $(n-h)$ most significant bits and X_2 which is the h least significant bits; the divisor is Y ($n/2$ bits). X_1 and Y are processed by the exact restoring array divider to generate the exact partial quotient (Q_1) and the $n/2$ bit partial remainder R_1. Then, R_1 and X_2 are concatenated as the dividend for the LD to generate the approximate quotient Q_2, where Y is still the divisor. Therefore, the $(n-h)$ most significant bits of the quotient ($i.e.$, Q_1) are generated by the restoring array divider and the h least significant bits of the quotient ($i.e.$, Q_2) are generated by the LD. The final Q is the concatenation of Q_1 and Q_2.

The expression of the proposed AXHD can be derived as follows. X is on n bits and Y is on $n/2$ bits. X is split into two parts: X_1 and X_2 are on $(n-h)$ and h bits, respectively. By dividing X by Y we get Q on n bits.

The expression of the exact quotient from the restoring array divider ($i.e.$ Q_1) is given by:

$$Q_1 = \frac{X_1 - R_1}{Y} \times 2^h \tag{11}$$

By dividing X_1 by Y, we get Q_1 on $(n-h)$ bits. The partial remainder R_1 is in a range of [0, Y), by definition of division. Therefore, R_1 is on $n/2$ bits. The expression of X is given by:

$$X = Q_1 \times Y \times 2^h + R_1 \times 2^h + X_2 \tag{12}$$

This implies that the expression of Q is given by:

$$Q = \frac{X}{Y} = Q_1 \times 2^h + \frac{R_1 \times 2^h + X_2}{Y} \tag{13}$$

Q_1 has on the most significant $(n-h)$ bits that are not zero, and hence $Q_1 \times 2^h$ is on n bits and occupies the most significant part with non-zero bits and the least significant part (h bits) equals to zero.

When $h < n/2$, R_1 has $n/2$ bits and therefore $T = R_1 \times 2^h + X_2$ is on $(n/2+h)$ bits, where the $n/2$ most significant bits are covered by R_1 and the least h significant bits by X_2, $i.e.$ by concatenation. When $h > n/2$, the upper $h - n/2$ bits of R_1 must be 0 and the lower $n-h$ bits of R_1 are selected. Therefore $T = R_1 \times 2^h + X_2$ is on n bits, where the $n-h$ most significant bits are covered by R_1 and the least h significant bits by X_2, $i.e.$ by concatenation. As per (6) and (7), the following expressions are then obtained.

$$T = 2^{k_1}(1 + m_1) \tag{14}$$

$$Y = 2^{k_2}(1 + m_2) \tag{15}$$

where, k_1 and k_2 represent the position of the leading most significant bits. m_1 and m_2 are the mantissa parts and in the range of [0,1). According to the approximation method in Eqs. (9)-(10), the expression of the approximate quotient ($i.e.$ Q_2) is given by:

$$Q_2 = \frac{T}{Y} \approx 2^{k_1-k_2} + (m_1 - m_2) \times 2^{k_1-k_2} \tag{16}$$

When $h < n/2$, This implies that T/Y is a $(n/2+h)$ by $n/2$ division. So, Q_2 is $n/2+h$ bits. The lower h bits of Q_2 are selected, because the higher $n/2$ bits must be 0. When $h > n/2$, This implies that T/Y is a (n) by $n/2$ division. So, Q_2 is n bits. The lower h bits of Q_2 are selected, because the higher $n-h$ bits must be 0.

Therefore, the final Q is obtained as $Q = Q_1 \times 2^h + Q_2$ by a simple concatenation. Although the value of $m_1 - m_2$ is different, the value of $1 + m_1 - m_2$ is always larger than 0.

$$Q = \frac{X_1 - R_1}{Y} \times 2^h + 2^{k_1-k_2} \times (1 + m_1 - m_2) \tag{17}$$

The errors in the final quotient are from the LD because the array divider is exact. Therefore, the accuracy of the proposed AXHD decreases by increasing h; at the same time, performance is improved by the use of a simpler LD. The accuracy can be controlled by the replacement depth and the appropriate value of h should be selected according to the accuracy requirement of a specific application.

Table 1 lists all symbols, the definitions and bit-widths used in AXHD.

TABLE 1: Symbols, the Definitions and Bit-Widths Used in AXHD.

Symbols	Definition	Bit-width
X	Total Dividend	n
Y	Divisor	$n/2$
X_1	Dividend of EXD	$n-h$
X_2	Partial Dividend of LD	h
R_1	Remainder of EXD	$n/2$
T	Dividend of LD	$n/2+h$
Q_1	Quotient from EXD	$n-h$
Q_2	Quotient from LD	h
Q	Total Quotient	n

3.2 Hardware Architectures of the AXHD

The proposed AXHD is shown in Fig. 5. As mentioned previously, conventional restoring array divider may overflow when X is large and Y is small, while the logarithmic divider does not. Therefore, extra rows are added to the array to avoid overflowing by computing more trial subtractions. For a 16-by-8 exact restoring divider, 8 extra rows are added, in which both the quotient and the remainder are 16-bit wide.

In the proposed AXHD, the partial remainder generated by the restoring divider cells is concatenated with X_2 and computed by the LD for the remaining quotient digits. Therefore, (n-h) rows of divider cells make up an exact divider whose input dividend is (n-h)-bit wide. In particular, h=n corresponds to a full LD and h=0 corresponds to an exact array divider. If h is sufficiently large, then a very simple hardware can be expected for implementing the exact division part.

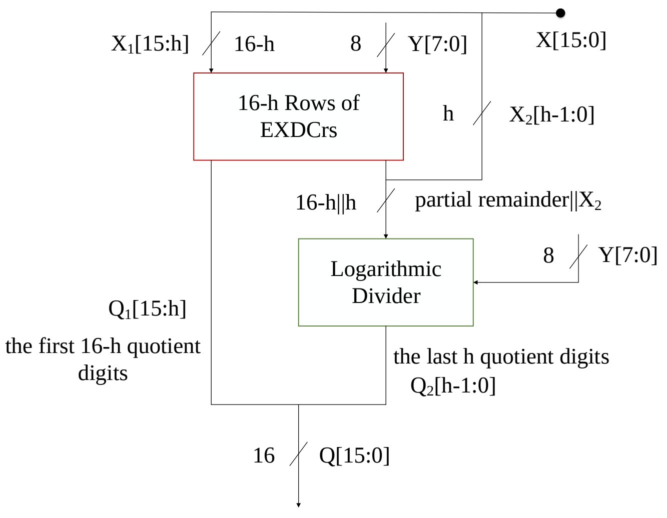


Fig. 5: 16-by-8 AXHD based on restoring array and logarithmic dividers.

Fig. 6 depicts the detailed block diagram of the proposed AXHD which combines the restoring array and logarithmic divider for h=14. The first two rows of the exact restoring divider cells correspond to (16-h)=2 and compute X[15] and X[14] with Y[7:0]. The partial remainder R[15:14] is concatenated with X[13:0] to form the dividend for the LD computing the division with Y[7:0] as the divisor.

The length of the quotient digits required by the array divider cells can be adjusted by h. A small h results in more

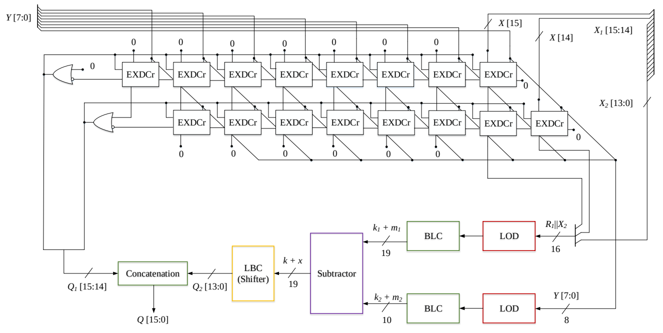


Fig. 6: A detailed block diagram of the 16-by-8 AXHD for h=14.

accurate results at the cost of both higher power consumption and hardware complexity. As mentioned previously, a logarithmic divider is used for the least quotient bits; thus, the remainder is not available. However, the computation of the remainder is naturally not required in terms of an approximate division.

3.3 The Proposed Eliminated AXHD (E-AXHD)

In this section, an improved eliminated approximate hybird divider (E-AXHD) is proposed. For a conventional 16-by-8 array divider, each row is fixed with eight subtractor cells. However, for the proposed AXHD design, the number of subtractor cells per row can be reduced. The reason is that when the dividend is smaller than the divisor, the lower significant bits of the divisor have no effect on the result. After selecting the replacement depth h, the $(n-h)$ most significant bits of the dividend are processed by the array divider. Since the bit-width of the divisor is 8 bits, when $h>8$, $(n-h)$-by-8 array divider can be replaced by a $(n-h)$-by-$(n-h+1)$ array divider with no loss of precision. For example, consider $X_1 = 1101$ and $Y = 10010010$ as the dividend and divisor, respectively; so and the quotient result is 0. However, using $X_1 = 1101$ divided by $Y = 10010$ as alternative, the result of the quotient is still 0. Note that this improvement can be only applied when $h>8$. Table 2 lists the number of subtractor cells reduced by E-AXHD compared with AXHD at the same replacement depth.

TABLE 2: Reduced Number of Subtractor Cells in E-AXHDs.

h	...9	10	11	12	13	14	15
Saved Cells (#)	0	6	10	12	12	10	6

Fig. 7 is the diagram of the proposed E-AXHD. The $(n-h)$ most significant bits of the dividend is selected. For selecting the divisor, the position of the leftmost one is obtained by the leading one detector (LOD); then the following $n-h+1$ bits are selected. For example, when $h=12$, the 4-by-8 array divider is replaced by a 4-by-5 array divider in E-AXHD. Suppose that $Y = 10010010$ is the divisor; so as $k>5$, 5 bits are selected from left to right as the divisor. Suppose $Y = 00000110$ is the divisor, then as $k<5$, 5 bits are selected from the right to the left as the divisor. All other processing steps are the same as AXHD. Without affecting the accuracy, E-AXHD can further reduce the hardware by eliminating the redundant subtractor cells compared with AXHD.

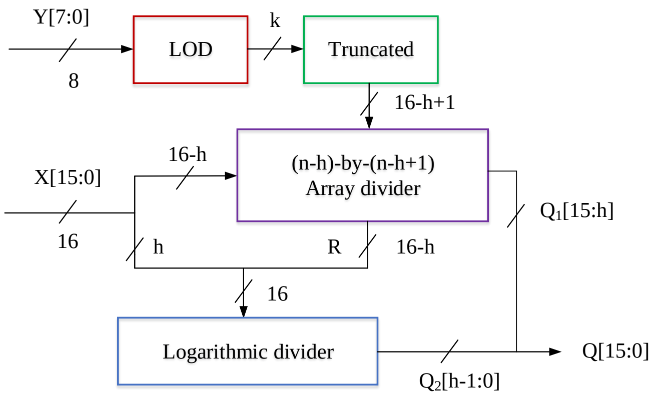


Fig. 7: 16-by-8 E-AXHD based on restoring array and logarithmic dividers.

Fig. 8 depicts the detailed block diagram of the proposed E-AXHD which combines the restoring array and logarithmic divider for h=14. Compared with Fig. 6, it can be seen that E-AXHD significantly reduces hardware resource consumption than AXHD at h=14.

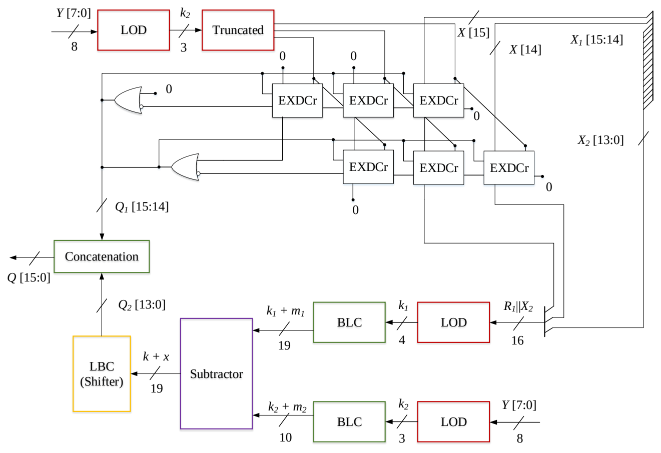


Fig. 8: A detailed block diagram of the 16-by-8 E-AXHD for h=14.

3.4 Division Examples Using AXHDs

A detailed example that is compatible with Fig. 6 is shown in Fig. 9. In this example, the dividend is given by 43382 and the divisor is 84. The steps to calculate the approximate quotient using AXHD are given as follows:

Step 1: Segment X =1010100101110110 into X_1=10 and X_2=10100101110110.

Step 2: The restoring array divider receives X_1 as dividend and Y=01010100 as divisor and generates the partial quotient Q_1=00 and the partial remainder R=10.

Step 3: The partial remainder R from the exact array divider is concatenated with X_2=10100101110110 to form the dividend $R||X_2$=1010100101110110 of the LD. Q_2=00001000000101 is generated by the LD with divisor Y.

Step 4: The final approximate quotient Q=0000100000101 is generated by concatenating Q_1 and Q_2.

The final result shows that the difference between the exact quotient ($i.e.$, Q=516) and the approximate quotient ($i.e.$, Q=517) is 1. In this work, the designs of both 16-by-8 and 8-by-4 AXHDs will be further studied. The error and hardware evaluation are presented in the next section.

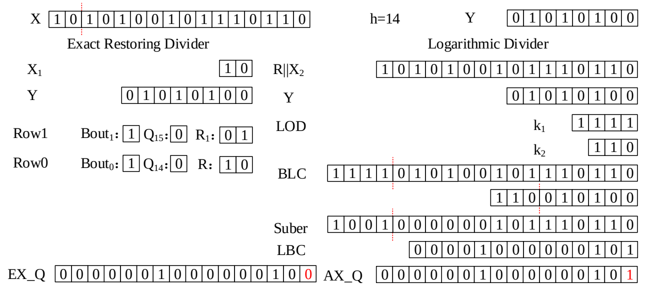

X 1 0 1 0 1 0 0 1 0 1 1 1 0 1 1 0
h=14
Y 0 1 0 1 0 1 0 0
Exact Restoring Divider
Logarithmic Divider
X1 1 0
R||X2 1 0 1 0 1 0 0 1 0 1 1 1 0 1 1 0
Y 0 1 0 1 0 1 0 0
Y 0 1 0 1 0 1 0 0
Row1 Bout1: 1 Q15: 0 R1: 0 1
LOD
k1 1 1 1 1
k2 1 1 0
Row0 Bout0: 1 Q14: 0 R: 1 0
BLC
1 1 1 1 0 1 0 1 0 0 1 0 1 1 1 0 1 1 0
1 1 0 0 1 0 1 0 0
Suber 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1 0 1 1 0
LBC 0 0 0 0 1 0 0 0 0 0 0 1 0 1
EX_Q 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
AX_Q 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1

Fig. 9: Example of the proposed 16-by-8 AXHD (h=14).

4 Error and Hardware Evaluation

For approximate arithmetic circuits, several metrics have been used to measure the error of the approximate designs. The Normalized Mean Error Distance (NMED), the Mean Relative Error Distance (MRED) and the maximum absolute error (MAE) [6] are widely used to evaluate the error characteristics. The NMED is defined as the mean error distance normalized by the maximum output of the accurate design. The MRED is defined as the mean value of the relative error distance that is the error distance over the absolute accurate output. The MAE is defined as the maximum absolute value. Therefore, these metrics are used for the error evaluation of the proposed designs. The error results are generated using exhaustive simulation (2^{16} for 16-bits and 2^8 for 8-bits).

For the hardware evaluation, the delay, area, power and energy are provided. The proposed designs are described at gate-level in Verilog HDL and verified by Synopsys VCS. All designs are then synthesized by the Synopsys Design Compiler using the NanGate 45 nm Open Cell Library [28]. In the simulation of each design, a supply voltage of 1.25 V and room temperature are assumed. The constraints for max area and delay are set to zero μm^2 and 1 ns, respectively. The average power consumption is found using the Synopsys Power Compiler with a back annotated switching activity file generated from the random input vectors. After synthesis, the tool generates a gate-level netlist in Verilog. The netlist is used to perform a static timing analysis (STA) to find the timing information. The actual test stimuli is given to the simulator along with the testbench (.v). After simulation, the simulator generates a value change dump (.vcd) file which records all actual transitions at the circuit nodes. All designs are synthesized and optimized using the default compiler options. The components, including EXDCr and LOD cells are synthesized using the same process.

Abbreviations for all approximate dividers in this paper are shown in Table 3.

TABLE 3: Abbreviations for Approximate Dividers.

AXHD	Approximate Hybrid Divider Combined with Restoring Array and Logarithmic Dividers.
E-AXHD	Eliminated Approximate Hybrid Divider based on AXHD.
AXDr(i) [20]	Approximate Restoring Array Divider with Replacement Schemes of i.
SEERAD(i) [23]	The High Speed yet Energy-efficient Rounding-based Approximate Divider with Security Level of i.
TruncApp(i) [24]	The Truncation-based Approximate Divider with Truncation Length of i.

4.1 Error Evaluation

16-by-8 and 8-by-4 are the most widely used designs. For example, in many image processing applications, the 8-bit divider is usually used; in machine learning applications, the 16-bit divider is the most common size for low power design. Therefore, in this section, we have provided the error and hardware results for both 16-by-8 and 8-by-4 designs. AXHD can only generate the quotient. The NMED, MRED and MAE of the integer quotient are provided in Table 4 for both 16-by-8 and 8-by-4 designs, respectively. The exact array divider is accurate divider, so it is not shown in the table. As h increases, more errors are introduced due to the increased bits allocated to the LD in the design. In both 16-bit and 8-bit designs, the error of AXHD increases logarithmically with a linear increase of h. Both the NEMDs and MREDs grow with an increase of h. However, when $h > n/2$, the error grows slower. Therefore, a better tradeoff between error and hardware performance can be achieved by using a large value of h, as further studied in the following subsection.

TABLE 4: NMED (10^{-6}), MRED (%) and MAE of 8-by-4 and 16-by-8 AXHDs.

Design	8-by-4				16-by-8			
	h	NMED (10^{-6})	MRED (%)	MAE	h	NMED (10^{-6})	MRED (%)	MAE
AXHD	1	0	0	0	2	0.32	0.03	0
	2	0.29	0.26	4	4	2.17	0.15	13
	3	0.83	0.63	7	6	8.99	0.45	21
	4	1.71	1.08	8	8	33.5	1.15	47
	5	3.19	1.52	11	10	89.2	1.83	92
	6	4.32	1.7	14	12	140.9	2.00	115
	7	4.91	1.76	17	14	179.7	2.03	121
	8	4.91	1.76	21	16	188.6	2.04	134

E-AXHD is an improved design of the 16-by-8 AXHD, and its NMED, MRED and MAE are the same as the 16-by-8 AXHD at the same replacement depth. However, compared with AXHD, performance has been further improved, as described in detail in the next subsection.

4.2 Hardware Evaluation

Previous approximate array dividers [20] have no significant improvement in terms of delay and area. For a divider with an approximate configuration, the approximation takes place at the subtractor array; so the delay and area of the inexact dividers is almost the same as the exact counterpart. In the proposed designs, the delay and area have been improved significantly when the replacement depth h is large because the delay and area of the LD are small.

One of the goals of an approximate design is to reduce the power consumption by tolerating computational errors. And energy is used to evaluate the performance of the designs, which is derived from the product of power consumption and delay. The power consumption, delay, area, energy of the E-AXHD and its improvement (%) over AXHD are reported in Table 7 at different values of replacement depth for 16-by-8 design. As shown in Table 7, E-AXHD has reduced power, delay, area and energy by up to 14.0%, 12.8%, 15.4%, and 22.2%, respectively, compared with AXHD.

TABLE 5: Hardware Results for the 16-by-8 E-AXHDs.

Depth (h)	Power (μw)	Delay (ns)	Area (μm^2)	Energy (pJ)
9	1024(0%)	4.99(0%)	1010(0%)	5.1(0%)
10	831(6.1%)	4.08(8.1%)	870(6.3%)	3.4(12.8%)
11	685(11.6%)	3.29(11.8%)	740(11.8%)	2.3(20.7%)
12	613(14.0%)	2.98(8.0%)	668(13.8%)	1.8(21.7%)
13	571(13.9%)	2.46(9.6%)	598(15.4%)	1.4(22.2%)
14	534(13.3%)	2.02(8.2%)	540(14.4%)	1.08(20.0%)
15	506(7.7%)	1.5(12.8%)	501(9.4%)	0.76(19.0%)

4.3 Comparison with Approximate Dividers

As per their performance, AXHDs are selected with replacement depths of 4, 8 and 12 for further comparison with other approximate dividers such as AXDr3 [20] with a TR scheme, SEERAD [23] and TruncApp [24]. As per the results in Section 4.2, the E-AXHD designs with replacement depths of 10, 12 and 14 are selected to further compare with other approximate dividers. SEERAD and TruncApp are recently proposed approximate dividers. SEERAD is designed based on rounding the divisor so that the division operation can be performed using only Shift and Add operations. SEERADs with all levels (from 1 to 4) are selected. In TruncApp, the division operation is performed by multiplying the truncated dividend by the approximate inverse of the divisor. TruncApp (3), TruncApp (4) and TruncApp_AM (5) are selected in this work, because they are the best choices among other truncation based approximate dividers as per the MRED and NMED values. In TruncApp_AM, the approximate multiplier is used. Moreover, AXDr3, SEERAD and TruncApp show better performance compared with other approximate dividers. The parameters of the proposed 16-by-8 designs as well as those of AXDr3, SEERAD and TruncApp dividers are reported in Table 8. As AXHDs and AXDr3_TRs have different replacement depths, h=4, 8 and 12 are chosen for comparison due to the good tradeoff between accuracy and hardware performance.

As shown in Fig. 10, the proposed designs are more accurate compared with SEERADs and TruncApps. However, in SEERAD and TruncApp, the delay is better compared with AXHDs and E-AXHDs. SEERAD (1) has the smallest delay and energy but also the largest NMED. TruncApp designs have low energy dissipation and relatively small NMED value. AXDr3 using the TR scheme generally has a small NMED but the energy dissipation is large due to the complex array circuitry. Different applications have different requirements for accuracy and energy consumption. Generally, the proposed AXHD and E-AXHD designs

TABLE 6: Comprehensive Comparison of Approximate Dividers.

Architecture	Area (μm^2)	Delay (ns)	Power (μw)	Energy (pJ)	MRED (%)	NMED (10^{-4})
SEERAD(1)	**1187**	1.21	572	0.69	15.58	764
SEERAD(2)	1732	1.5	981	1.47	8.52	411
SEERAD(3)	2006	1.85	1492	2.76	4.97	223
SEERAD(4)	3481	2.45	2993	7.33	2.71	109
TruncApp(3)	1401	1.35	520	0.7	9.8	523
TruncApp(4)	1628	1.6	683	1.09	4.22	188
TruncApp_AM(5)	1612	1.63	652	1.06	3.76	147
AXDr3_TR(4)	1316	8.84	1876	16.58	0.01	0.01
AXDr3_TR(8)	**1224**	8.75	1595	13.95	0.29	0.10
AXDr3_TR(12)	**1126**	8.57	1288	11.04	2.86	1.55
AXHD(4)	1523	7.83	2130	16.68	0.15	0.02
AXHD(8)	**1184**	5.69	1279	7.28	1.15	0.34
AXHD(12)	**775**	3.24	712	2.31	2.0	1.41
E-AXHD(10)	**870**	4.08	831	3.4	1.83	0.89
E-AXHD(12)	668	2.98	613	1.8	2.0	1.41
E-AXHD(14)	540	2.02	534	1.08	2.03	1.80

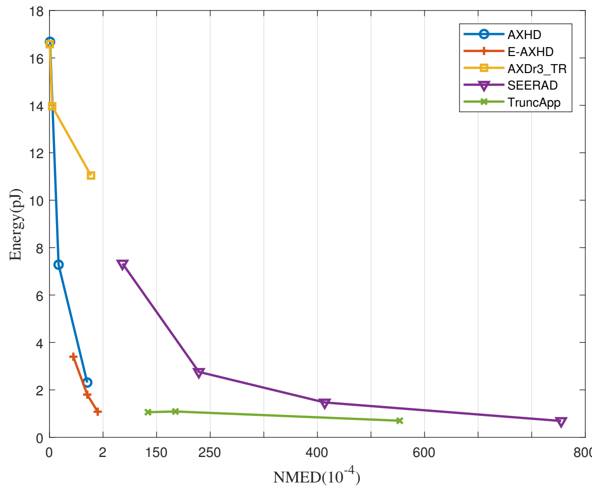


Fig. 10: Trade-off between Energy and NMED of the proposed AXHDs, E-AXHD, AXDr3_TRs, SEERADs and TruncApps.

have a higher accuracy and moderate energy consumption compared with SEERAD and TruncApp.

5 Applications

Among the error-tolerant applications, image processing and visualization are some of the most frequently found in the literature so appropriate for testing and benchmarking approximate computing systems [26]. In this section, image processing involving pixel division (only for quotient calculation) are studied using the proposed AXHDs and E-AXHDs. The proposed AXHDs and E-AXHDs are applied to change detection and background removal [27]. Moreover, the proposed designs is also applied to the softmax layer in deep neural networks (DNN).

5.1 Pixel Division

16-by-8 AXHDs and E-AXHDs and EXDr are used to compute the inputs of 8-bit grayscale images for pixel division. To perform a 16-by-8 division for this application using two 8-bit grayscale images, the pixel values of the first image are multiplied by 64 as dividends. The peak signal-noise ratio (PSNR) (based on the mean squared error) at different replacement depths is provided to show the difference between exact and approximate results. The power is measured with the specific benchmark data.

5.1.1 *Change Detection*

The output image from the change detection only shows the difference between two input images. If there is a change, the pixel of the intensity-change region in the image shows a significant difference between the two inputs images. Otherwise, the output image has pixels with a single value. Fig. 11 shows the 2 input images and 4 output images processed by EXDr, E-AXHD, SEERAD and TruncApp. E-AXHD(14), SEERAD(2) and TruncApp(4) have been selected for fair comparison, because they have similar energy consumption. Table 9 provide the PSNR results of the proposed AXHDs and E-AXHDs with various replacement depths.

TABLE 7: PSNR Results of AXHDs and E-AXHDs with Different Replacement Depths for Change Detection.

Depth (AXHD)	1	2	3	4	5	6	7	8
PSNR /dB	Inf	Inf	84.69	64.99	61.32	54.17	39.87	39.71
Depth (E-AXHD)	9	10	11	12	13	14	15	16
PSNR /dB	39.71	39.71	39.71	39.71	39.71	39.71	39.71	39.71

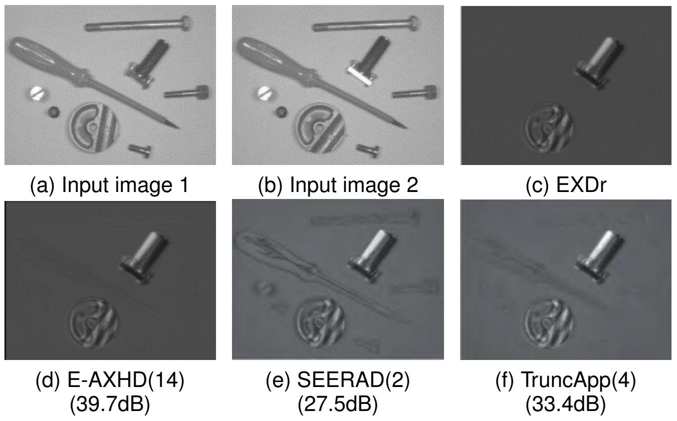


Fig. 11: Change detection using EXDr, E-AXHD, SEERAD and TruncApp.

As can be seen in Table 9, when h is smaller than 8, the proposed AXHDs generate results with high PSNRs. When h is larger than 8, the PSNR (39.71) does not further decrease, because the input dividend is fully computed by the logarithmic divider, however the power can be further reduced. The PSNR of the output image processed by E-AXHD(14) is higher than those generated by SEERAD(2) and TruncApp(4) in Fig. 11.

5.1.2 *Background Removal*

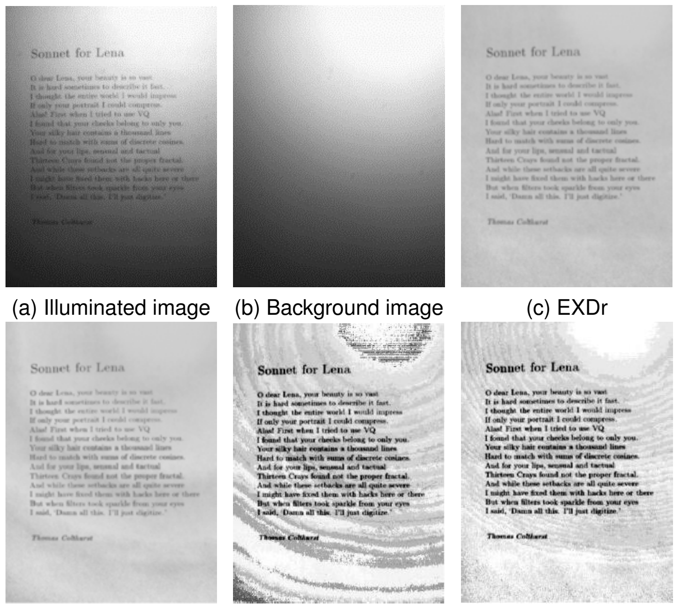
Background removal consists of removing background illumination from a image so that the foreground objects can be separated for a more clear visualization and/or further processing. Similar to change detection, there is no significant loss of accuracy when the replacement depth h is

1 and 2 for background removal. The PSNR decreases when the replacement depth h increases. When the replacement depth h is larger than 7, the PSNR converges to 27.12. The results for the proposed E-AXHD with h=14, SEERAD(2) and TruncApp(4) are shown in Fig. 12. The proposed E-AXHD(14) show similar results as the exact dividers.

The PSNR decreases with larger replacement depth. The h value can therefore be selected according to the required PSNR threshold value, and determine the required hardware resources.

TABLE 8: PSNR Results of AXHDs and E-AXHDs with Different Replacement Depths for Background Removal.

Depth (AXHD)	1	2	3	4	5	6	7	8
PSNR /dB	Inf	Inf	74.17	62.95	56.07	50.32	43.04	27.12
Depth (E-AXHD)	9	10	11	12	13	14	15	16
PSNR /dB	27.12	27.12	27.12	27.12	27.12	27.12	27.12	27.12


(a) Illuminated image (b) Background image (c) EXDr

(d) E-AXHD(14) (27.1dB) (e) SEERAD(2) (15.8dB) (f) TruncApp(4) (21.4dB)

Fig. 12: Background removal using EXDr, E-AXHD, SEERAD and TruncApp.

5.2 Softmax Layer

The softmax function is widely used in the final layer of the DNN to deal with multi-classification problems. Softmax function can produce a probability distribution for predicted output classes. Different from other functional layers in DNN, the softmax layer contains exponentiation and division operation. To further evaluate the effectiveness of the proposed approximate dividers, we have applied them to the softmax function. The softmax function can be expressed as follows [32]:

$$P = \frac{e^{x^T W_j}}{\sum_{k=1}^{K} e^{x^T W_k}} \tag{18}$$

where, P is the predicted probability for the jth class given a sample vector x and a weighting vector w. K is the number of classes.

In the Softmax application, the division is performed by 16-by-32. Therefore, the bit width of the divisor is set to 32 bits in the proposed design, and the range of the approximate replacement depth h is still [0,16]. In this work, the MINST database for the handwriting digits is used [33]. Initially, the number of labels is set to 10. Then training data samples are used for training. Finally, testing data is applied to the approximate designs to find the probability of a correct classification.

The results are reported in Table 9. The probability obtained by using the exact division is 92.5%. The results show that the proposed AXHDs and E-AXHDs have a probability of more than 90% for $h < 6$ and around 86.5% when $h > 10$. The approximate designs can be selected according to the accuracy requirements of different neural networks.

TABLE 9: Probability of AXHDs and E-AXHDs with Different Replacement Depths for Softmax Layer.

Depth (AXHD)	1	2	3	4	5	6	7	8
Probability(%)	92.5	92.4	92	91.4	90.7	89.8	88.6	87.4
Depth (E-AXHD)	9	10	11	12	13	14	15	16
Probability(%)	86.8	86.6	86.5	86.5	86.5	86.5	86.5	86.5

In order to further validate the proposed approximate dividers, it is applied into the softmax function based multi-classification problem (the proposed approximate divider replaces the exact dividers). The case study is to classify the objects in the urban environments from processed remote sensing images. The data file is an attribute image derived from hyperspectral airborne and elevation images provided by the National Ecological Observatory Network (NEON) [34]. The image is processed to form a data set consisting of six layers to be labelled. There are five types of labels, which are asphalt, concrete, grass, trees and buildings. Based on the softmax function, the softmax regression algorithm is added. The labeled data training set is then used. For each label attribute, a random gradient descent function is used to calculate the weight. After the training, the activation function is used to determine whether the attribute belongs to the output Class, and then use iterative methods to minimize classification errors and try to adjust the internal parameters of the classifier until the error (also known as loss) converges to a minimum value. Finally, the test set is used to evaluate the trained classifier. The classification results using both exact dividers and the proposed approximate dividers are shown in Table 12. It can be seen, the classification accuracy from the approximate design can still reach above 90% when $h < 12$, which proves that the trained classifier using the proposed approximate divider is acceptable for similar data sets.

TABLE 10: Classification Results from the Softmax Regression Classifier.

Classes	Asphalt	Concrete	Grass	Tree	Building
EXDr	0.996	0.988	0.990	0.995	0.993
AXHD (h=4)	0.955	0.948	0.949	0.954	0.950
AXHD (h=8)	0.932	0.922	0.925	0.930	0.926
E-AXHD (h=12)	0.908	0.893	0.896	0.904	0.899

6 Related Works

In this section, the differences between the proposed designs and state-of-the-art approximate dividers are discussed and presented.

Both the AXDr [20] and DAXD [22] are based on the array structure. AXDr replace the exact subtractor cells with approximate versions while DAXD reduce the bit-width of the array. However, both have large area and long delay due to the array structure. Compared with AXDr and DAXD, the proposed hybrid designs take advantage of logarithmic operation, which further reduces the hardware consumption and the delay.

SEERAD [23] and TruncApp [24] convert division into multiplication for simplification. Although this structure can greatly improve the speed performance, hardware is further increased due to the use of a multiplier and look-up tables. Furthermore, their errors are significantly larger than the array based designs. Compared with SEERAD and TruncApp, the proposed designs use exact array divider for most significant bits to achieve a higher accuracy. At the same time, the logarithmic operation can further reduce the hardware complexity.

7 Conclusion

This paper has presented a detailed design, analysis and evaluation of novel hybrid approximate dividers based on both restoring array and logarithmic schemes. The LD has been combined with a conventional array divider, because it has the advantage of low power consumption. Different design sizes (16-by-8 and 8-by-4) and the replacement depth h have been considered to evaluate performance and the error characteristics of these inexact hybrid dividers. By replacing the last rows of exact cells by the logarithmic divider, a good tradeoff between accuracy and hardware performance (*i.e.* power, delay and energy) has been achieved. The proposed dividers have been analyzed using error and hardware metrics; the results have shown that the proposed AXHDs and E-AXHDs perform better than other approximate dividers, especially when the replacement depth h was large. The proposed hybrid dividers could be of interest for image processing applications.

Acknowledgment

This work has been supported by the National Natural Science Foundation of China under Grant 61871216, the Fundamental Research Funds for the Central Universities China under Grant NE2019102 and the Six Talent Peaks Project in Jiangsu Province (2018-XYDXX-009). The work of F. Lombardi is supported by the National Science Foundation (USA).

References

[1] J. Miao, A. Gerstlauer and M. Orshansky, *"Multi-level approximate logic synthesis under general error constraints"*, in Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2014, pp. 504-510.

[2] J. Han and M. Orshansky, *"Approximate computing: an emerging paradigm for energy-efficient design"*, in Proc. 18th IEEE European Test Symposium (ETS), 2013, pp. 1-6.

[3] S. Venkataramani, S. Chakradhar, K. Roy and A. Raghunathan, *"Approximate computing and the quest for computing efficiency"*, in Proc. 52nd Annual Design Automation Conference (DAC), 2015, Article 120, 6 pages.

[4] Q. Xu, N.-S. Kim and T. Mytkowicz, *"Approximate Computing: A Survey"*, IEEE Design & Test, vol. 33, no.1, pp. 8-22, 2016.

[5] L. Leem, H. Cho, J. Bau, Q.A. Jacobson and S. Mitra. *"ERSA: Error resilient system architecture for probabilistic applications"*, in Pro. Design, Automation and Test in Europe (DATE), 2010, pp. 1560-1565.

[6] J. Liang, J. Han, and F. Lombardi, *"New metrics for the reliability of approximate and probabilistic adders"*, IEEE Trans. Computers, vol. 63, pp. 1760-1771, 2013.

[7] S.-L. Lu, *"Speeding up processing with approximation circuits"*, Computer, vol. 37, pp. 67-73, 2004.

[8] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo and Z. H. Kong, *"Design of low-power high-speed truncation error tolerant adder and its application in digital signal processing"*, IEEE Trans. VLSI Syst., vol. 18, pp. 1225-1229, 2010.

[9] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie and C. Lucas, *"Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications"*, IEEE Trans. Circuits Syst.: Part I Regular Papers, vol. 57, pp. 850-862, 2010.

[10] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, *"Low power digital signal processing using approximate adders"*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 32, pp. 124-137, 2013.

[11] J. E. Robertson, *"A new class of digital division methods"*, IRE Transactions on Electronic Computers, vol. EC-7, no. 3, pp. 218-222, Sept. 1958.

[12] H. H. Guild, *"Some cellular logic arrays for non-restoring binary division"*, Radio and Electronic Engineer, vol. 39, no. 6, pp. 345-348, June 1970.

[13] J. Mitchell, *"Computer multiplication and division using binary logarithms"*, IRE Trans. Electron. Comput., vol. 11, pp. 512-517, 1962.

[14] S. Hashemi, R. Bahar and S. Reda, *"DRUM: A dynamic range unbiased multiplier for approximate applications"*, in Proc. IEEE/ACM Int. Conf. Computer Design (ICCD), 2015, pp. 418 - 425.

[15] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han and F. Lombardi, *"Design of approximate radix-4 Booth multipliers for error-tolerant computing"*, IEEE Trans. Computers, vol. 66, pp. 1435-1441, Aug. 2017.

[16] Y.-H. Chen and T.-Y. Chang, *"A high-accuracy adaptive conditional probability estimator for fixed-width Booth multipliers"*, IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 59, pp. 594-603, 2012.

[17] G. Zervakis, K. Tsoumanis, S. Xydis, N. Axelos and K. Pekmestzi, *"Approximate multiplier architectures through partial product perforation: power-area tradeoffs analysis"*, in Proc. ACM Great Lake Symp. VLSI (GLSVLSI), 2015, pp. 229-232.

[18] A. Momeni, J. Han, P. Montuschi and F. Lombardi, *"Design and Analysis of Approximate Compressors for Multiplication"*, IEEE Trans. Computers, vol. 64, no. 4, pp. 984-994, 2015.

[19] L. Chen, Jie Han, W. Liu, and F. Lombardi, *"Design of approximate unsigned integer non-restoring divider for inexact computing"*, In Proc. Great Lakes Symp. VLSI (GLSVLSI), pp. 51-56, 2015.

[20] L. Chen, Jie Han, W. Liu, and F. Lombardi, *"On the design of approximate restoring dividers for error-tolerant applications"*, IEEE Trans. Computers, vol. 65, pp. 2522-2533, 2016.

[21] T. Aoki, K. Nakazawa, and T. Higuchi. *"High-radix parallel VLSI dividers without using quotient digit selection tables"*, In Proc. 30th IEEE Int. Symp. Multiple-Valued Logic (ISMVL), pp. 345-352, 2000.

[22] S. Hashemi, R. Bahar, and S. Reda. *"A low-power dynamic divider for approximate applications"*, In Proc. 53rd Annual Design Automation Conference (DAC), p. 2-6, 2016.

[23] 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 Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1481-1484, 2016.

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

[25] W. Liu, J. Li, T. Xu, C. Wang, P. Montuschi and F. Lombardi, *"Combining Restoring Array and Logarithmic Dividers into an Approximate Hybrid Design"*, in Proc. 25rd IEEE Symp. Computer Arithmetic (ARITH), pp. 80-86, 2018.

[26] A. Yazdanbakhsh, D. Mahajan, P. Lotfi-Kamran, H. Esmaeilzadeh, *"AXBENCH: A Multi-Platform Benchmark Suite for Approximate Computing"*, IEEE Design and Test, vol. 34, no. 2, pp. 60-68, 2017.

[27] R. Fisher, S. Perkins, A. Walker and E. Wolfart. Pixel division. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixdiv.htm, 2003.

[28] Nangate 45nm Open Cell Library. http://www.nangate.com/.
[29] U. Qidwai and C.H. Chen, *"Digital Image Processing: An Algorithmic Approach with MATLAB"*, CRC Press, 2009.
[30] The USC-SIPI Image Database [Online]. Available: http://sipi.usc.edu/database
[31] B. Parhami, *"Computer Arithmetic: Algorithms and Hardware Designs: Oxford University Press"*, 2000.
[32] B. Yuan, *"Efficient hardware architecture of softmax layer in deep neural network"*, in Proc. 29th IEEE International System-on-Chip Conference (SOCC), 2016.
[33] Y. LeCun, C. Cortes, C. Burges, The MNIST Database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/, 2019.
[34] Wolfe J, Jin X, Bahr T, et al. *"Application of softmax regression and its validation for spectral-based land cover mapping"*. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2017, 42: 455.

Weiqiang Liu (S'10-M'12-SM'15) is currently a Professor and the Vice Dean of College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. He received the B.S. degree in Information Engineering from NUAA and the Ph.D. degree in Electronic Engineering from Queens University Belfast (QUB), Belfast, United Kingdom, in 2006 and 2012, respectively. In Dec. 2013, he joined the College of Electronic and Information Engineering, NUAA. He has served as a Guest Editor of Proceedings of the IEEE and Associate Editors of IEEE Transactions on Circuits and Systems I: Regular Paper, IEEE Transactions on Emerging Topic in Computing and Computers, IEEE Transactions on Computers, and a Steering Committee Member of IEEE Transactions on Multi-Scale Computing Systems. He is the Program Co-Chair of IEEE Symposium on Computer Arithmetic (ARITH), and program members for a number of international conferences. His research interests include emerging technologies in computing systems, computer arithmetic, hardware security and VLSI design for digital signal processing and cryptography. He has published one research book by Artech House and over 100 leading journal and conference papers. He is a Senior Member of the IEEE and the Chinese Institute of Electronics.

Tao Xu received the B.Sc. degree in Electronic Information Engineering from the Kunming University of Science and Technology, Kunming, China, in 2015. He is pursuing his Master degree at the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. His research interests include computer arithmetic and approximate computing.

Jing Li received the B.Sc. degree in Electronic Information Science and Technology from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2018. He is currently pursuing the Master degree with ShanghaiTech University, Shanghai, China. His current research interests are computer vision and machine learning.

Chenghua Wang received the B.Sc. and M.Sc. degrees from Southeast University, Nanjing, China, in 1984 and 1987, respectively. In 1987, he joined the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, where he became a full Professor in 2001. He has published 6 books and over 100 technical papers in journals and conference proceedings. He is the recipient of more than ten teaching and research awards at the provincial and ministerial level. His current research interests include design and test of integrated circuits, and circuits & systems for communications.

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 (BoG) at Politecnico di Torino. Previously, he served as Chair of the Department from 2003 to 2011. His research interests include computer arithmetic and architectures, computer graphics, electronic publications, semantics and education. He is an IEEE Fellow, an IEEE Computer Society (CS) Golden Core member, a recipient of the "Distinguished Service" and the "Spirit of the Computer Society" awards. He is currently serving as the acting Editor-in-Chief of IEEE Transaction on Emerging Topics in Computing. He served as Editor-in-Chief of IEEE Transactions on Computers (2015-18), as the Chair of the CS Awards Committee (2017-18), and as a member of the IEEE Publications Services and Products Board (2018-20). Previously, he served in a number of Boards and Committees, including the IEEE Products and Services Committee and the BoG of the CS. 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. In the past Dr. Lombardi was the Editor-In-Chief of the IEEE Transactions on Computers and the inaugural Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computing. Currently, he is the Editor-in-Chief of the IEEE Transactions on Nanotechnology. Since 2019, he serves as the Vice President for Publications of the IEEE Computer Society. 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. He is a Fellow of IEEE.