Approximate Restoring Dividers Using Inexact Cells and Estimation From Partial Remainders

Elizabeth Adams, *Member, IEEE*, Suganthi Venkatachalam, and Seok-Bum Ko, *Senior Member, IEEE*

Abstract—Approximate computing can be used in error-resilient applications to reduce power consumption and increase overall circuit performance. This article introduces two approximate dividers with restoring array-based architecture that achieve substantial hardware savings while maintaining high accuracy when compared to existing approximate designs. The first design replaces exact restoring divider cells with a proposed approximate cell in a column-wise fashion. The second design uses several rows of exact architecture to compute a partial remainder and then rounds and encodes the divisor and this partial remainder so that they may be used to express approximate outputs. A comprehensive accuracy and performance evaluation are performed for the proposed dividers as well as other state-of-the-art designs. When compared to an exact design, the proposed dividers have a reduced area and power consumption of 46 and 57 percent respectively while introducing minimal error. Furthermore, the trade-off between accuracy and improved performance is explored for various approximate dividers in order to determine which designs achieve the best compromise. The accuracy of the proposed dividers is then demonstrated using two image processing applications.

Index Terms—Approximate computing, restoring division, error analysis, image processing applications, low-power

1 Introduction

Traditional computer arithmetic architectures typically involve the use of digital circuitry to express logical functions with high accuracy. Some applications, such as multimedia, data mining, and neural networks, exhibit a property known as error-resilience, meaning that high accuracy is not a strict requirement and some error may be tolerated in exchange for improved processing efficiency [2]. Approximate computing is the field concerned with sacrificing accuracy in exchange for improved efficiency. Approximation in hardware typically involves altering circuitry with the goal of introducing minimal error in exchange for improvements in delay, circuit area, and power consumption [3].

Approximate addition has been widely explored in the literature, where designs typically either involve transistor-level modifications to the half-adder and full-adder architectures [4], [5], [6] or approximation applied at a higher level to multiple-bit adders [7], [8], [9], [10], [11]. Similarily, inexact multiplication has also been extensively explored, where approximation has been introduced to both *AND*-gate based multipliers [12] and to Booth multipliers [13], [14], [15], [16], [17]. In comparison, less attention has been given to the development of approximate division algorithms.

While multiplication can be implemented as completely parallel additions, division is unique in that it requires that subtractions be performed sequentially and thus approximation techniques that are popular in multiplication may not be well suited for division algorithms. [18] introduces several approximate subtractor designs which are used in approximate divider cells. In [18], some exact divider cells in a non-restoring divider are replaced with inexact cells that make use of these approximate subtractors, resulting in lowered power consumption. These inexact subtractors are also used to form approximate cells in a low-power restoring divider design in [19]. In [20], inexact operands are inputted to an exact divider, where the operands are approximated by detecting the leading-ones and then truncating all bits beyond a specified width, resulting in substantial power savings. An approximate divider is introduced in [21] where a truncated dividend is multiplied by an approximate inverse divisor, resulting in improved area and power consumption. Delay and energy consumption are reduced in [22], where division is transformed to a multiplication operation by rounding the divisor to specific form. [23] proposes a low-power design that prunes the input bits before inputting them to a reduced-width divider. In [24], a traditional restoring array-based divider is combined with the approximate logarithmic divider introduced in [25] to produce a hybrid approximate divider design that achieves impressive power and delay savings.

This paper proposes two approximate restoring divider models 1 and 2—AXRD-M1 and AXRD-M2. A novel approximate restoring divider cell is introduced and the AXRD-M1 divider design replaces some exact cells with these approximate cells in a method similar to the triangle replacement technique in [19]. The proposed AXRD-M2 design eliminates entire horizontal array rows and generates inexact outputs based on the value of exact partial remainders in a high-level structure that is similar to that proposed in [24]. While [24] replaces rows of exact cells with an approximate logarithmic divider, the AXRD-M2 design instead recodes the partial remainder outputted by the last row of exact cells in order to approximate the remaining quotient bits.

This paper is an extension of our conference work [1]. The main improvements and novel contributions of this paper include:

1) AXRD-M2 dividers are introduced in which entire rows of the divider array are replaced with reduced approximate logic and further approximation is applied via the use of Karnaugh maps to reduce the circuitry for the outputted quotient.
2) Error analysis is expanded to include the calculation of error rate (percentage of outputs with non-zero error) and maximum error distance (largest possible distance between actual value and approximate value) for both quotient and remainder.
3) An analysis of accuracy-performance tradeoff for all designs is performed by comparing quotient error metrics (i.e., MRED and NMED) with power-delay product (PDP) and area-delay product (ADP). To further evaluate this tradeoff, several combined accuracy-performance metrics are introduced, namely MRED-PDP product (MPDPP), MRED-ADP product (MADPP), NMED-PDP product (NPDPP), and NMED-ADP product (NADPP). The use of such tradeoff metrics allows for a better insight into the ability of the implemented designs to provide performance savings while maintaining strong accuracy metrics.
4) Dividers are used to perform foreground extraction in addition to change detection, where peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) are used to measure the performance of each model.

2 Exact Restoring Dividers

Restoring division algorithms generally involve the division of a $2k$-bit dividend z by a k-bit divisor d to produce a k-bit quotient q

• *The authors are with the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK S7N 5A2, Canada. E-mail: {lizzie.adams, suganthi.venkat, seokbum.ko}@usask.ca.*

Manuscript received 11 Feb. 2019; revised 6 Sept. 2019; accepted 11 Oct. 2019. Date of publication 15 Nov. 2019; date of current version 10 Mar. 2020. (Corresponding author: Seok-Bum Ko.) Recommended for acceptance by L. A. Sousa. Digital Object Identifier no. 10.1109/TC.2019.2953751

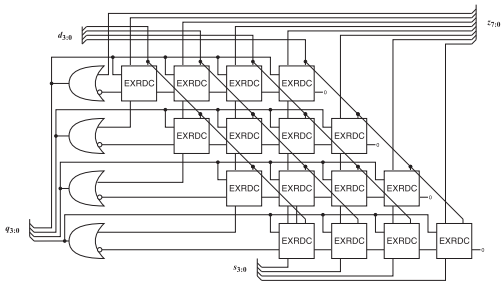


Fig. 1. Architecture of $8/4$ exact restoring divider.

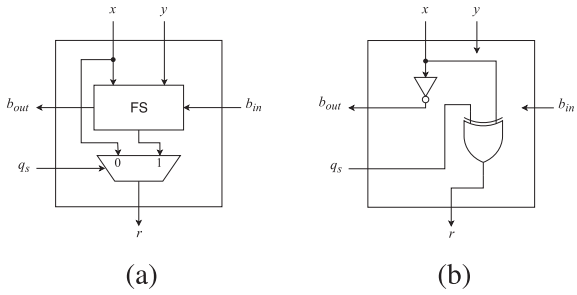


Fig. 2. Circuit diagrams for (a) exact restoring divider cell (EXRDC) and (b) proposed approximate restoring divider cell (AXRDC).

and k-bit remainder s, where $s < d$. Restoring division involves repeatedly attempting to subtract the divisor from the shifted partial remainder where a negative partial remainder results in the restoring of its prior value.

In restoring division, each iteration j involves the subtraction of the shifted divisor $2^k d$ from the shifted partial remainder $2s^{(j-1)}$ to produce a trial difference. If the result of the subtraction is non-negative, the quotient bit q_{k-j} is set to 1 and the next partial remainder $s^{(j)}$ is loaded with the value of the trial difference. If the subtraction produces a negative result, the quotient bit q_{k-j} is set to 0 and the partial remainder is restored to its prior shifted value $s^{(j)} = 2s^{(j-1)}$. After $j = k$ iterations, all quotient bits have been selected and the remainder s becomes the most-significant k bits of the final partial remainder $s^{(k)}$.

Restoring division is commonly implemented using an array divider structure. A $2k/k$ division makes use of an array of k rows, where each row contains k restoring divider cells and a single OR-gate. Each exact restoring divider cell (EXRDC) consists of a full subtractor (FS) and multiplexer, as shown in Fig. 2a. The multiplexer is used to select the difference in the case of a non-negative partial remainder and otherwise selects the prior value in the case of a restored partial remainder. The architecture of an $8/4$ array-based restoring divider (i.e., 8-bit dividend, 4-bit divisor) is illustrated in Fig. 1.

3 Proposed Approximate Restoring Dividers

An exact $2k/k$ division requires $k \times k$ restoring divider cells as well as k OR-gates. As k grows, the hardware required to compute the division quickly becomes expensive. This section describes two approximate division architectures that compute an approximate quotient and remainder through the use of reduced hardware. The first approximate architecture AXRD-M1 replaces some of the exact restoring divider cells with approximate restoring divider cells whose logic expressions are simplified. The second approximate architecture AXRD-M2 involves the elimination of some rows of the divider array.

As the ratio of approximate cells to exact cells in AXRD-M1 increases, the accuracy decreases and the hardware savings improve. A similar tradeoff occurs in AXRD-M2 as the number of eliminated rows increases. The term approximation factor p is used to refer to the magnitude of approximation for a given design. In AXRD-M1, p expresses the number of columns whose exact cells have been replaced with approximate cells. In AXRD-M2, p indicates the number of rows that are eliminated from the array.

It should be noted that, while AXRD-M1 computes an approximate remainder, AXRD-M2 does not. This is related to the fact that the quotient computation is the operation of importance with regards to the use of approximate dividers for most applications [19]. The architecture of AXRD-M1 is structured such that the computing of an approximate remainder in addition to approximate quotient bits requires no additional circuity; that is, the gates contained within the divider cells of AXRD-M1 generate both approximate quotient bits and an approximate remainder. The architecture of AXRD-M2 differs in this regard in that additional gates would be needed to compute an approximate remainder. For this reason, it was decided that no extra circuitry was to be added to AXRD-M2 for the purpose of computing a remainder in order to achieve maximal hardware savings.

3.1 Approximate Restoring Divider Model 1 (AXRD-M1)

This section describes the architecture of the proposed approximate restoring divider model 1 (AXRD-M1). As described above, this design involves replacing all exact cells in the p least-significant columns with approximate cells. The circuitry of the exact restoring divider cell (EXRDC) is illustrated in Fig. 2a. The outputs of EXRDC r and b_{out} can be expressed as

$$r = q_s d + \overline{q_s} x\,,$$
$$b_{out} = \overline{x \oplus y} \cdot b_{in} + \overline{x} y\,.$$

The approximate restoring divider cell (AXRDC) makes use of simplified circuitry to reduce hardware cost. The circuit diagram for the AXRDC is shown in Fig. 2b and illustrates that the outputs r and b_{out} are independent of y and b_{in}. The corresponding logic expressions are shown below in which r and b_{out} are expressed only in terms of x and q_s:

$$r = q_s \oplus x\,, b_{out} = \overline{x}\,.$$

The AXRD-M1 design for approximation factor p involves replacing EXRDCs with AXRDCs in the p least-significant columns of the array. For a $2k/k$ divison and approximation factor p, the number of approximated cells is given by

$$\begin{cases} \dfrac{p(p+1)}{2}\,, & \text{for } p <= k \\ k(2p-k+1) - \dfrac{p(p+1)}{2}\,, & \text{for } p > k\,. \end{cases}$$

3.2 Approximate Restoring Divider Model 2 (AXRD-M2)

This section describes the architecture of the proposed approximate restoring divider model 2 (AXRD-M2). The design makes use of exact architecture to perform the first $k - p$ trial subtractions to produce the first p quotient bits and the partial remainder $s^{(k-p)}$. The position of the leading-ones in the divisor d and the partial remainder $s^{(k-p)}$ are used to produce encoded values n and m which are then used to approximate the final value of the remaining quotient bits. The overall architecture of AXRD-M2 is shown in Fig. 3.

Independent of p, the first three bits of the divisor d are examined by d-LOD for leading-ones. The location of the leading-one or its absence is used to round the divisor as shown in Equation (1). More concisely, the location of the leading-one is determined and the value of the rounded divisor d_{rnd} is computed by keeping the leading-one and setting all other bits to zero. In the case that the

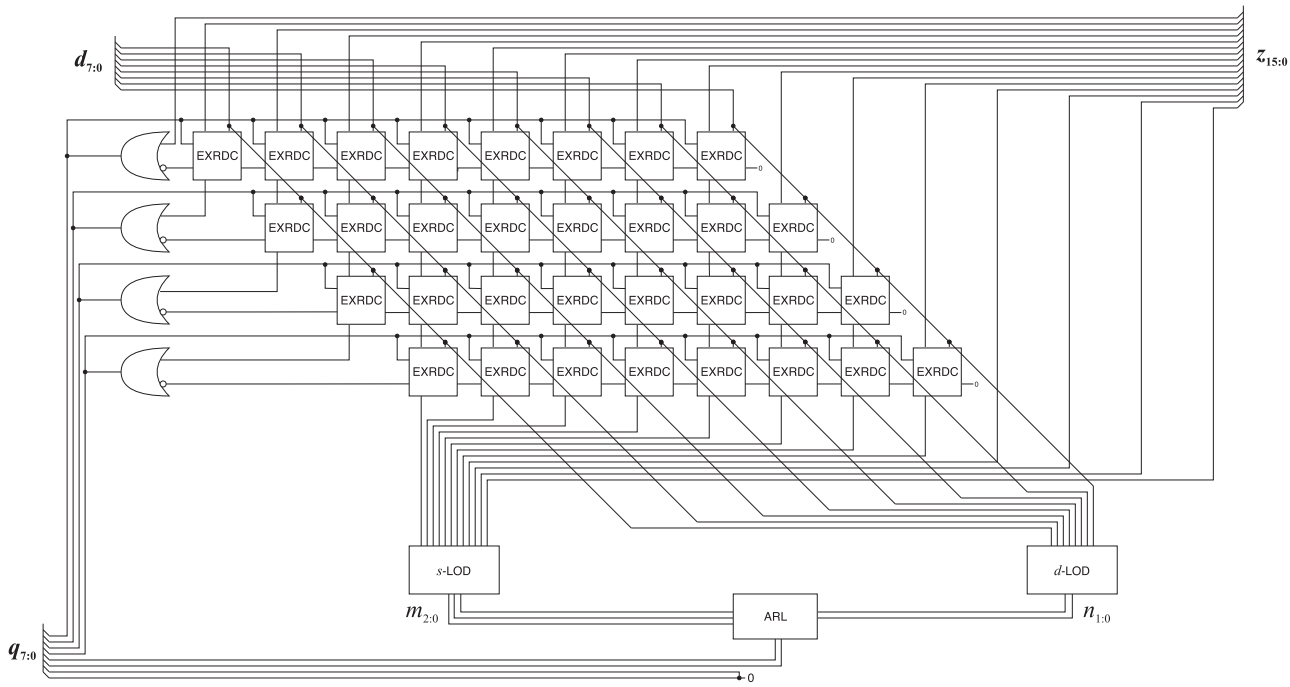


Fig. 3. Architecture of $16/8$ proposed design AXRD-M2 for $p = 4$.

leading-one is absent, the leading-one is assumed to be at bit-position $k-4$ and the value of d_{rnd} consequently becomes 16.

$$d_{rnd} = \begin{cases} 2^7 = 128, & d_{k-1:k-3} = 1\text{xx} \\ 2^6 = 64, & d_{k-1:k-3} = 01\text{x} \\ 2^5 = 32, & d_{k-1:k-3} = 001 \\ 2^4 = 16, & d_{k-1:k-3} = 000 \end{cases}. \tag{1}$$

The first seven bits of partial remainder $s^{(k-p)}$ are similarly examined by s-LOD to produce a rounded partial remainder $s_{rnd}^{(k-p)}$ as shown in Equation (2). It should be noted that the width of $s^{(k-p)}$ is $k+p$ bits. In the case that the leading-one is absent, the partial remainder is rounded to zero.

$$s_{rnd}^{(k-p)} = \begin{cases} 2^{k+p-1}, & s_{k+p-1:k+p-7}^{(k-p)} = 1\text{xxxxxx} \\ 2^{k+p-2}, & s_{k+p-1:k+p-7}^{(k-p)} = 01\text{xxxxx} \\ 2^{k+p-3}, & s_{k+p-1:k+p-7}^{(k-p)} = 001\text{xxxx} \\ 2^{k+p-4}, & s_{k+p-1:k+p-7}^{(k-p)} = 0001\text{xxx} \\ 2^{k+p-5}, & s_{k+p-1:k+p-7}^{(k-p)} = 00001\text{xx} \\ 2^{k+p-6}, & s_{k+p-1:k+p-7}^{(k-p)} = 000001\text{x} \\ 2^{k+p-7}, & s_{k+p-1:k+p-7}^{(k-p)} = 0000001 \\ 0, & s_{k+p-1:k+p-7}^{(k-p)} = 0000000 \end{cases}. \tag{2}$$

To provide a more tangible example, Equation (2) for $p = 4$ can be used to generate

$$s_{rnd}^{(4)} = \begin{cases} 2^{11} = 2048, & s_{11:5}^{(4)} = 1\text{xxxxxx} \\ 2^{10} = 1024, & s_{11:5}^{(4)} = 01\text{xxxxx} \\ 2^9 = 512, & s_{11:5}^{(4)} = 001\text{xxxx} \\ 2^8 = 256, & s_{11:5}^{(4)} = 0001\text{xxx} \\ 2^7 = 128, & s_{11:5}^{(4)} = 00001\text{xx} \\ 2^6 = 64, & s_{11:5}^{(4)} = 000001\text{x} \\ 2^5 = 32, & s_{11:5}^{(4)} = 0000001 \\ 0, & s_{11:5}^{(4)} = 0000000 \end{cases},$$

where the 7 most-significant bits of $s^{(4)}$ are examined to produce $s_{rnd}^{(4)}$.

The result of rounding the divisor d_{rnd} and partial remainder $s_{rnd}^{(k-p)}$ are then used to generate encoded values n and m respectively, where n is a 2-bit number and m is a 3-bit number. The entire process of rounding the divisor and partial remainder as well as generating n and m is illustrated in Tables 1 and 2.

The approximate remaining quotient bits resulting from dividing the rounded divisor and rounded partial remainder can be expressed in terms of n and m. However, these equations are expressed in terms of 5 input bits and therefore can be quite complicated. This problem is remedied by applying further approximation to the logic expressions for the remaining quotient bits $q_{p-1:0}$ via the use of Karnaugh maps. For $p = 4$, the Karnaugh maps for q_3 and q_2 are shown in Figs. 4a and 4b respectively. The corresponding logic expressions are described in Equations (3) and (4). Table 3 demonstrates the entire process of generating m and n from the rounded partial remainder and divisor and using these encoded values to generate the approximate remaining quotient bits for $p = 4$. Table 3 also illustrates the exact quotient value q_{exact} resulting from dividing the rounded partial remainder $s_{rnd}^{(k-p)}$ by the rounded divisor d_{rnd} and the approximate quotient value q_{approx} that is derived from the Karnaugh maps in Figs. 4a and 4b, as well as the difference between them. Error is exhibited only by the inexact quotient and specifically in cases where the exact quotient value is larger than 15 due to the fact that the approximation only expresses the inexact quotient using four bits. It should be noted that q_{exact} in this context is not truly exact because approximation has already been applied when rounding the partial remainder and the divisor. The expressing of these outputs in terms of m and n is performed by the approximate recoded logic (ARL) block as shown in Fig. 3.

$$q_3 = m_2 m_1 + m_2 \overline{n}_1 \tag{3}$$

$$q_2 = m_2 m_0 + m_1 \overline{n}_1. \tag{4}$$

For any approximate arithmetic circuit, the magnitude of approximation incorporated into the design is primarily limited by the ceiling on acceptable error metrics defined by the intended application.

TABLE 1
Divisor Rounding and Encoding to Produce n

$d_{k-1:k-3}$	d_{rnd}	n_1	n_0
000	16	0	0
001	32	0	1
01x	64	1	0
1xx	128	1	1

TABLE 2
Partial Remainder Rounding and Encoding to Produce m

$s^{(k-p)}_{k+p-1:k+p-7}$	$s^{(k-p)}_{rnd}$	m_2	m_1	m_0
0000000	0	0	0	0
0000001	2^{k+p-7}	0	0	1
000001x	2^{k+p-6}	0	1	0
00001xx	2^{k+p-5}	0	1	1
0001xxx	2^{k+p-4}	1	0	0
001xxxx	2^{k+p-3}	1	0	1
01xxxxx	2^{k+p-2}	1	1	0
1xxxxxx	2^{k+p-1}	1	1	1

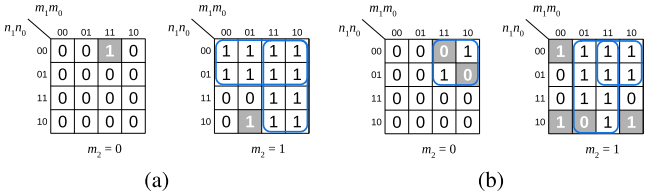


Fig. 4. Karnaugh maps used to apply further approximation to quotient bits (a) q_3 and (b) q_2. Approximated bits are shaded.

During the design of AXRD-M2, competitive error metrics were only achieved when eliminating no more than $n/2$ array rows and thus approximation factors greater than 4 were not explored. More specifically, approximation factors of $p = 2, 3, 4$ exhibited a desirable trade-off between reduction in accuracy and hardware savings and therefore those designs are included in this paper. Only the architecture for $p = 4$ is illustrated in full (Fig. 3), but the architectures for $p = 2, 3$ share logic with that of $p = 4$. More specifically, the logic expression of q_3 for $p = 4$ is equivalent to that of q_2 for $p = 3$ and q_1 for $p = 2$. Similarly, the expression of q_2 for $p = 4$ is equal to that of q_1 for $p = 3$ and q_0 for $p = 2$. In the $p = 4$ design, the two least-significant quotient bits $q_1 q_0$ were truncated as was the least-significant quotient bit q_0 for $p = 3$.

4 Experimental Results

This section provides an evaluation of the proposed approximate dividers in terms of accuracy and hardware efficiency. The proposed designs are compared with AXDr1, AXDr2, and AXDr3 designs using triangle replacement introduced in [19], the AAXD design proposed in [23], and the AXHD design proposed in [24]. For AXDr dividers, approximation factor p refers to the number of columns whose exact cells are replaced with approximate cells, whereas p refers to the bitwidth of the pruned divisor in the context of AAXD designs. For AXHD designs, p refers to the approximation depth as defined in their work. The approximation factors used in comparison designs were chosen such that resulting error metrics are similar to those of the proposed designs, allowing designs of similar accuracy to be compared in terms of hardware efficiency.

4.1 Accuracy Evaluation

Accuracy evaluation was performed for $16/8$ dividers using exhaustive analysis. All dividers were implemented in Verilog and simulated using Mentor Graphics ModelSim. Python scripts were used to generate inputs as well as to calculate error metrics. The accuracy of approximate dividers was evaluated using mean relative error distance (MRED) and normalized mean error distance (NMED) as defined in [26]. Error rate (ER) and maximum error distance (ED_{max}) are also given. Simulation results are summarized in Table 4 where accuracy measurements are provided for all dividers for quotient and remainder, where applicable.

TABLE 3
Use of Encoded Values n and m to Produce Approximate Quotient Bits for $p = 4$

$s^{(k-p)}_{rnd}$	d_{rnd}	Encoding		Quotient Values		
		$m_{2:0}$	$n_{1:0}$	q_{exact}	q_{approx}	$difference$
0	16	000	00	0	0	0
0	32	000	01	0	0	0
0	64	000	10	0	0	0
0	128	000	11	0	0	0
32	16	001	00	2	2	0
32	32	001	01	1	1	0
32	64	001	10	0	0	0
32	128	001	11	0	0	0
64	16	010	00	4	4	0
64	32	010	01	2	2	0
64	64	010	10	1	1	0
64	128	010	11	0	0	0
128	16	011	00	8	8	0
128	32	011	01	4	4	0
128	64	011	10	2	2	0
128	128	011	11	1	1	0
256	16	100	00	16	15	1
256	32	100	01	8	8	0
256	64	100	10	4	4	0
256	128	100	11	2	2	0
512	16	101	00	32	15	17
512	32	101	01	16	15	1
512	64	101	10	8	8	0
512	128	101	11	4	4	0
1024	16	110	00	64	15	49
1024	32	110	01	32	15	17
1024	64	110	10	16	15	1
1024	128	110	11	8	8	0
2048	16	111	00	128	15	113
2048	32	111	01	64	15	49
2048	64	111	10	32	15	17
2048	128	111	11	16	15	1

Table 4 shows that AXRD-M1, AXDr1, and AXDr3 perform best in terms of quotient MRED, with the lowest MRED being achieved by AXDr1 for $p = 6$. Considering quotient NMED, the highest-performing designs are AXRD-M1, AXDr1, AXDr3, and AXHD, where AXHD for $p = 12$ has the lowest NMED value. It is important to note that the low NMED values achieved by AXHD are largely due to the fact that a $2k/k$ division in this design produces a $2k$-bit quotient rather than a k-bit quotient, meaning that NMED for this design is generated by dividing MED by $2^{2k} - 1$ rather than $2^k - 1$, resulting in a substantially smaller value. The lowest quotient ER is demonstrated by AXRD-M1 for $p = 4$ and AXDr1 for $p = 6$, where both designs achieve similar values. AXRD-M1, AXRD-M2, and AAXD have small ED_{max}, where the AXRD-M2 obtains notably small values.

The sequential nature of division means that errors introduced in one stage will be propagated to the next stage. Because the remainder is computed at the very final stage, it tends to have a lower accuracy than that of the quotient. Approximate divider designs often prioritize the accuracy of the quotient over that of the remainder, and some approximate dividers do not compute a remainder value at all. Only a subset of the implemented designs compute a remainder, namely AXRD-M1 and the AXDr designs. As reflected in Table 4, the remainder accuracy metrics achieved

TABLE 4
Accuracy and Hardware Metrics for Implemented Approximate $16/8$ Dividers

Design	p	Quotient Accuracy				Remainder Accuracy				Hardware Metrics				
		ER (%)	MRED (10^{-2})	NMED (10^{-2})	ED_{max}	ER (%)	MRED (10^{0})	NMED (10^{0})	ED_{max}	Delay (ns)	Area (μm^2)	Power (mW)	PDP (pJ)	ADP ($\mu m^2 \cdot$ ns)
Exact	–	–	–	–	–	–	–	–	–	5.05	827.6	0.3703	1.870	4180
AXRD-M1	4	9.26	0.226	0.037	7	84.99	1.662	0.088	253	4.51	735.8	0.3325	1.500	3319
	6	23.47	0.494	0.101	31	94.25	2.269	0.175	255	4.05	635.0	0.2809	1.138	2572
	8	66.29	1.909	0.449	127	98.40	2.893	0.261	255	3.15	472.0	0.1856	0.585	1487
AXRD-M2	2	40.95	1.066	0.163	3	–	–	–	–	3.79	625.7	0.2554	0.968	2371
	3	64.73	2.214	0.351	7	–	–	–	–	3.15	523.8	0.2032	0.640	1650
	4	81.35	4.248	0.759	15	–	–	–	–	2.53	420.8	0.1525	0.386	1055
AXDr1 [19]	6	9.12	0.182	0.038	12	67.10	0.873	0.111	252	4.69	802.8	0.3472	1.628	3765
	8	50.93	1.207	0.292	51	86.97	3.103	0.292	255	4.53	778.7	0.3195	1.447	3527
	10	80.94	4.323	1.320	116	92.48	3.786	0.320	255	4.36	755.3	0.2854	1.244	3293
AXDr2 [19]	6	50.47	1.272	0.251	63	88.61	2.351	0.258	255	3.59	700.9	0.2957	1.062	2516
	8	96.43	6.543	1.395	255	96.98	4.407	0.329	255	2.66	567.7	0.2213	0.589	1510
	10	99.41	25.640	5.177	255	98.22	4.424	0.333	255	1.56	336.6	0.1039	0.162	525
AXDr3 [19]	6	11.95	0.211	0.049	21	84.01	1.748	0.120	254	5.02	721.8	0.3135	1.574	3623
	8	48.39	0.961	0.257	85	93.76	2.542	0.278	255	5.37	630.0	0.2538	1.363	3383
	10	78.64	3.251	0.967	119	95.98	2.265	0.278	255	5.75	568.8	0.2079	1.195	3271
AAXD [23]	5	73.74	1.521	0.716	14	–	–	–	–	3.23	870.1	0.2518	0.813	2811
	4	84.49	3.121	1.462	27	–	–	–	–	2.48	693.0	0.1666	0.413	1717
	3	91.06	6.343	2.966	49	–	–	–	–	1.81	580.3	0.1143	0.207	1050
AXHD [24]	12	15.86	3.946	0.019 [1]	4080	–	–	–	–	2.15	821.2	0.1473	0.317	1766
	14	15.86	3.946	0.058	16320	–	–	–	–	2.16	775.4	0.1435	0.310	1675
	16	15.86	3.946	0.208	65280	–	–	–	–	2.22	760.3	0.1421	0.315	1688

by those designs are generally quite poor, although this is not of great concern considering that applications for approximate remainders are rather limited. Regardless, it is worth mentioning that the smallest remainder MRED and NMED are achieved by AXDr1 and AXRD-M1, respectively. All dividers that compute a remainder are capable of producing maximal or almost-maximal remainder error distances.

4.2 Hardware Evaluation

All inexact $16/8$ dividers as well as an exact $16/8$ restoring divider were implemented in Verilog and synthesized in TSMC 65 nm process technology using Synopsys Design Compiler. For all simulations, an operating voltage of 1V and an operating temperature of 25° C were used. Synopsys Design compiler was used to report critical path delay, circuit area, and power dissipation. Power-delay product (PDP) and area-delay product (ADP) are used to provide a more complete measurement of hardware efficiency. Simulation results are summarized in Table 4.

The proposed AXRD-M1 and AXRD-M2 provide reductions in critical path delay of up to 38 and 49 percent respectively when compared to the exact design. The AXDr [19] designs have relatively high delay, especially AXDr3 whose delay value actually surpasses that of the exact divider for $p > 6$. The most impressive delay metrics are achieved by the AAXD [23] and AXHD [24] designs, where AAXD for $p = 3$ demonstrates a reduction in critical path delay of 64 percent. In terms of circuit area, AXRD-M1, AXRD-M2, and AXDr2 exhibit the most substantial reduction. AXRD-M1 and AXRD-M2 achieve area savings of up to 43 and 46 percent respectively when compared to the exact divider, while AXDr2 for $p = 10$ has the lowest circuit area of all implemented designs with a reduction of 59 percent. It should be noted that AAXD has area larger than that of the exact divider for $p = 5$. The greatest reduction in power dissipation is achieved by AXRD-M1, AXRD-M2, AAXD, and AXHD. When compared to the exact divider, the proposed designs AXRD-M1 and AXRD-M2 reduce power consumption by up to 50 and 57 percent respectively, while the greatest reduction of 69 percent is obtained by AAXD for $p = 3$.

A greater understanding of the hardware performance of a design can be obtained by considering the combined hardware metrics shown in Table 4, namely power-delay product (PDP) and area-delay product (ADP). The most impressive PDP values are those for AXRD-M2, AAXD, and AXHD. The proposed design AXRD-M2 provides a reduction in PDP over the exact divider of up to 79 percent, and the smallest PDP of all implemented designs is achieved by AAXD for $p = 3$ with a reduction of 89 percent. The designs exhibiting the lowest ADP values are AXRD-M2, AXDr2, and AAXD, where the proposed AXRD-M2 has a reduction of up to 75 percent, and the lowest ADP is obtained by AXDr2 for $p = 10$ with a reduction of 87 percent.

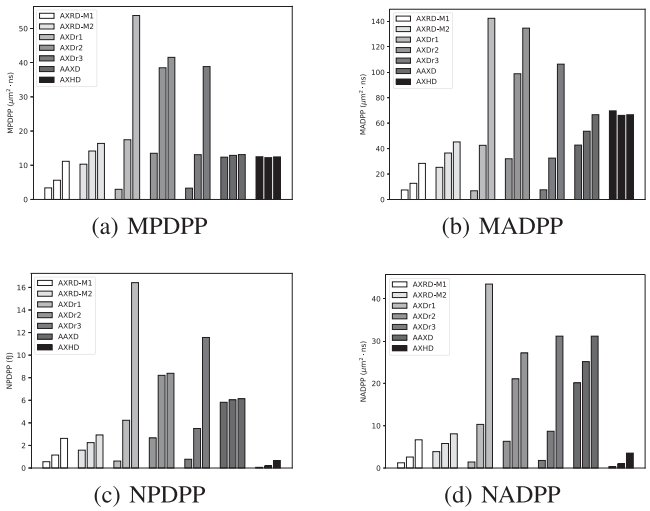
4.3 Accuracy-Performance Tradeoff

As shown in Table 4, the best MRED is achieved by AXRD-M1, AXDr1, and AXDr3, which exhibit similar values for $p = 4, 6, 6$, respectively. Among these designs, AXRD-M1 provides a small advantage over the others in terms of delay, AXRD-M1 and AXDr3 achieve the smallest area metrics, and AXDr3 has the most impressive power savings. The best NMED is achieved by AXRD-M1, AXDr1, AXDr3, and AXHD, where similar values are exihibited for $p = 4, 6, 12$, respectively. Among these designs, AXRD-M1 and AXDr3 provide a modest advantage in area savings, while AXHD achieves substantial improvements over the others in terms of both delay and power consumption.

There is a significant variance in the accuracy and hardware performance of the implemented dividers as shown in Table 4. In order to achieve a better understanding of the general ability of the divider to provide improved performance while maintaining accuracy, an anlysis of the accuracy-performance tradeoff of each design is needed. Several papers have made use of combined accuracy-hardware metrics to provide greater insight into the ability of an approximate design to achieve substantial performance improvements while maintaining high accuracy [18], [19], [24], [27], [28], [29]. To provide a thorough and complete evaluation of all implemented designs, four accuracy-hardware tradeoff metrics are proposed, namely MRED-PDP product (MPDPP) [29], MRED-ADP product (MADPP), NMED-PDP product (NPDPP), and NMED-ADP product (NADPP). Table 5 summarizes the tradeoff metrics calculated for all implemented designs. Figs. 5a, 5b, 5c, and 5d illustrate the relationship between approximation factor and each tradeoff metric. The lowest MPDPP and MADPP are achieved by AXDr1 and the lowest NPDPP and NADPP are achieved by AXHD. As per the discussion in Section 4.1, it is unsurprising that AXHD has such low values for these metrics,

TABLE 5
Approximate $16/8$ Divider Accuracy-Hardware Tradeoff Metrics

Design	p	Tradeoff Metrics			
		MPDPP (fJ)	MADPP (μm$^2\cdot$ ns)	NPDPP (fJ)	NADPP (μm$^2\cdot$ ns)
AXRD-M1	4	3.388	7.497	0.558	1.234
	6	5.624	12.715	1.149	2.598
	8	11.158	28.375	2.623	6.669
AXRD-M2	2	10.314	25.266	1.581	3.874
	3	14.169	36.524	2.248	5.794
	4	16.390	45.231	2.927	8.077
AXDr1 [19]	6	2.968	6.862	0.616	1.425
	8	17.468	42.572	4.229	10.308
	10	53.794	142.361	16.422	43.458
AXDr2 [19]	6	13.506	32.015	2.663	6.312
	8	38.514	98.804	8.210	21.062
	10	41.558	134.635	8.390	27.182
AXDr3 [19]	6	3.315	7.633	0.776	1.788
	8	13.099	32.515	3.499	8.686
	10	38.864	106.330	11.565	31.641
AAXD [23]	5	12.371	42.750	5.823	20.123
	4	12.895	53.637	6.041	25.130
	3	13.123	66.630	6.137	31.156
AXHD [24]	12	12.497	69.670	0.060	0.337
	14	12.232	66.097	0.181	0.978
	16	12.449	66.608	0.657	3.517



(a) MPDPP (b) MADPP (c) NPDPP (d) NADPP

Fig. 5. Graphs illustrating accuracy-performance tradeoff metrics for all implemented approximate dividers.

considering that they are based off of NMED measurements. While AXDr1 has very low MPDPP and MADPP for $p = 6$, it can be seen in Figs. 5a and 5b that these values sharply increase with p, and a similar trend is exhibited by AXDr3. The proposed designs AXRD-M1 and AXRD-M2 not only have accuracy-performance tradeoff metrics on par with the highest-achieving of the implemented designs, but it is also apparent from Figs. 5a, 5b, 5c, and 5d that they provide an advantage over other state-of-the-art designs in that their ability to achieve competitive tradeoff metrics is substantially less dependent on p.

5 Applications

The proposed designs were testing using two pixel subtraction applications, namely change detection and foreground extraction.

5.1 Change Detection

The change between two images can be detected by dividing the pixel values of two images, resulting in quotient values that express that ratio of the pixel input values. Approximate $16/8$ dividers were used to detect the change between the 8-bit grayscale

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

Fig. 6. Change detection results for (a) input image 1 and (b) input image 2. Results using (c) exact divider, (d) AXRD-M1 $p = 4$, (e) AXRD-M2 $p = 2$.

TABLE 6
Approximate $16/8$ Divider Application Metrics

Design	p	Change Detection		Background Removal	
		PSNR (dB)	SSIM	PSNR (dB)	SSIM
AXRD-M1	4	60.2	0.9996	51.8	0.9957
	6	57.7	0.9993	48.9	0.9920
	8	50.7	0.9959	42.1	0.9765
AXRD-M2	2	53.7	0.9983	48.5	0.9927
	3	51.3	0.9972	44.4	0.9887
	4	45.3	0.9893	37.6	0.9739
AXDr1 [19]	6	57.4	0.9993	49.1	0.9936
	8	52.2	0.9982	42.6	0.9868
	10	40.8	0.9614	36.1	0.9440
AXDr2 [19]	6	51.9	0.9981	42.9	0.9901
	8	41.6	0.9912	35.3	0.9593
	10	29.8	0.9262	29.6	0.8309
AXDr3 [19]	6	60.9	0.9997	57.1	0.9988
	8	54.8	0.9986	46.4	0.9859
	10	45.2	0.9841	36.7	0.9219
AAXD [23]	5	46.9	0.9949	45.0	0.9857
	4	39.5	0.9829	40.7	0.9683
	3	36.8	0.9474	35.1	0.9257
AXHD [24]	12	46.3	0.9959	42.7	0.9874
	14	46.3	0.9959	42.7	0.9874
	16	46.3	0.9959	42.7	0.9874

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

Fig. 7. Background removal results for (a) input image 1 and (b) input image 2. Results using (c) exact divider, (d) AXRD-M1 $p = 4$,(e) AXRD-M2 $p = 2$.

images shown in (a) and (b) of Fig. 6. The pixel values of the image shown in Fig. 6a served as the dividend in computations and therefore was multiplied by a factor of 64 to improve precision. Peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) index were used to express the efficacy of the inexact dividers in detecting change. The change detection calculated using an exact $16/8$ divider is shown in Fig. 6c and is used as reference when calculating PSNR and SSIM. The results of computing change detection using the proposed AXRD-M1 and AXRD-M2 are shown in Figs. 6d and 6e. The PSNR and SSIM values computed for all implemented approximate dividers are summarized in Table 6.

As visible in Figs. 6d and 6e, the proposed approximate dividers produced images that are visually similar to that generated by the exact divider. The images most closely resembling that of the exact divider were generated by AXRD-M1 AXDR-M2 and AXDr designs. AXDr3 produces the best result with a PSNR and SSIM of 60.9 dB and 0.9997 respectively, closely followed by AXRD-M1 with a PSNR and SSIM of 60.2 dB and 0.9996 respectively.

5.2 Foreground Extraction

Given an image with visually disruptive background variation, the division of the image by its background allows for the foreground to be extracted. Approximate $16/8$ dividers were used to extract the foreground of the image shown in Fig. 7a by dividing its pixel

values by that of the background image shown in Fig. 7b. Both images are in 8-bit grayscale and the image in Fig. 7a was multiplied by a factor of 64 prior to division in order to improve precision. The effectiveness of various inexact dividers in extracting the image foreground is expressed using PSNR and SSIM. The reference image produced using an exact $16/8$ divider is shown in Fig. 7c and the images produced using AXRD-M1 and AXRD-M2 are shown in Figs. 7d and 7e. PSNR and SSIM values computed for all implemented approximate dividers are summarized in Table 6.

Most approximate dividers produced outputs similar to that of the exact divider. The proposed dividers AXRD-M1 and AXRD-M2 produce good results for $p = 4, 6$ and $p = 2, 3$ respectively. The output images of highest quality were produced by AXDr designs with the highest PSNR and SSIM of 57.1 dB and 0.9988 being produced by AXDr3, closely followed by AXRD-M1 with a PSNR and SSIM of 51.8 dB and 0.9957 respectively.

6 Conclusion

This paper introduced two novel approximate divider designs. The first design involves replacing exact restoring divider cells of lower-significant with a novel approximate cell, allowing for improved hardware efficiency with little impact on accuracy. The second design involves the rounding and encoding of the divisor and partial remainder, resulting in substantially increased performance. The proposed designs occupy up to 46 percent less area and consume up to 57 percent less power than the exact restoring divider. When compared to other approximate dividers, the proposed designs were shown to provide a good compromise between improving performance and maintaining accuracy. Change detection and foreground extraction were implemented and used to demonstrate the accuracy of the proposed designs in image processing applications.

Acknowledgments

The authors would like to thank Natural Sciences and Engineering Research Council of Canada (NSERC) and the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada. This work was partially supported by the R&D program of MOTIE/KEIT [10077609, Developing Processor-Memory-Storage Integrated Architecture for Low Power, High Performance Big Data Servers]. This article is an extension of our work originally presented in 2019 ISCAS [1].

References

[1] S. Venkatachalam, E. Adams, and S. Ko, "Design of approximate restoring dividers," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2019, pp. 1–5.
[2] J. Han, "Introduction to approximate computing," in *Proc. IEEE 34th VLSI Test Symp.*, Apr. 2016, Art. no. 1.
[3] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE Eur. Test Symp.*, May 2013, pp. 1–6.
[4] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.
[5] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in *Proc. 13th IEEE Int. Conf. Nanotechnol.*, Aug. 2013, pp. 690–693.
[6] 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. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.
[7] 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. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.
[8] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proc. 12th Int. Symp. Integr. Circuits*, Dec. 2009, pp. 69–72.
[9] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *Proc. Int. SoC Design Conf.*, Nov. 2010, pp. 323–327.
[10] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. DAC Design Autom. Conf.*, Jun. 2012, pp. 820–825.
[11] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2012, pp. 728–735.
[12] S. Venkatachalam and S. Ko, "Design of power and area efficient approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1782–1786, May 2017.
[13] 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. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.
[14] L. Qian, C. Wang, W. Liu, F. Lombardi, and J. Han, "Design and evaluation of an approximate wallace-booth multiplier," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2016, pp. 1974–1977.
[15] S. Venkatachalam, H. J. Lee, and S. Ko, "Power efficient approximate booth multiplier," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2018, pp. 1–4.
[16] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.
[17] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 421–430, Mar. 2018.
[18] L. Chen, J. Han, W. Liu, and F. Lombardi, "Design of approximate unsigned integer non-restoring divider for inexact computing," in *Proc. 25th Edition Great Lakes Symp. VLSI*, 2015, pp. 51–56. [Online]. Available: http://doi.acm.org/10.1145/2742060.2742063
[19] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the design of approximate restoring dividers for error-tolerant applications," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2522–2533, Aug. 2016.
[20] S. Hashemi, R. I. Bahar, and S. Reda, "A low-power dynamic divider for approximate applications," in *Proc. 53nd ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2016, pp. 1–6.
[21] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, and Z. Navabi, "Truncapp: A truncation-based approximate divider for energy efficient DSP applications," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, Mar. 2017, pp. 1635–1638.
[22] 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 Autom. Test Eur. Conf. Exhibit.*, Mar. 2016, pp. 1481–1484.
[23] H. Jiang, L. Liu, F. Lombardi, and J. Han, "Adaptive approximation in arithmetic circuits: A low-power unsigned divider design," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, Mar. 2018, pp. 1411–1416.
[24] 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. IEEE 25th Symp. Comput. Arithmetic*, Jun. 2018, pp. 92–98.
[25] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 4, pp. 512–517, Aug. 1962.
[26] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.
[27] B. Shao and P. Li, "A model for array-based approximate arithmetic computing with application to multiplier and squarer design," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Aug. 2014, pp. 9–14.
[28] W. Liu *et al.*, "Design and analysis of approximate redundant binary multipliers," *IEEE Trans. Comput.*, vol. 68, no. 6, pp. 804–819, Jun. 2019.
[29] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "Low-power approximate multipliers using encoded partial products and approximate compressors," *IEEE J. Emerging Selected Topics Circuits Syst.*, vol. 8, no. 3, pp. 404–416, Sep. 2018.