Adaptive Approximation in Arithmetic Circuits: A Low-Power Unsigned Divider Design

Honglan Jiang*, Leibo Liu†, Fabrizio Lombardi‡ and Jie Han*
*Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada
†Institute of Microelectronics, Tsinghua University, Beijing, China
‡ Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA
Email: {honglan, jhan8}@ualberta.ca, liulb@tsinghua.edu.cn, lombardi@ece.neu.edu.

***Abstract*—Many approximate arithmetic circuits have been proposed for high-performance and low-power applications. However, most designs are either hardware-efficient with a low accuracy or very accurate with a limited hardware saving, mostly due to the use of a static approximation. In this paper, an adaptive approximation approach is proposed for the design of a divider. In this design, division is computed by using a reduced-width divider and a shifter by adaptively pruning the input bits. Specifically, for a $2n/n$ division $2k/k$ bits are selected starting from the most significant '1' in the dividend/divisor. At the same time, redundant least significant bits (LSBs) are truncated or if the number of remaining LSBs is smaller than $2k$ for the dividend or k for the divisor, '0's are appended to the LSBs of the input. To avoid overflow, a $2(k+1)/(k+1)$ divider is used to compute the division of the $2k$-bit dividend and the k-bit divisor, both with the most significant bits being '0'. Thus, $k < n$ is a key variable that determines the size of the divider and the accuracy of the approximate design. Finally, an error correction circuit is proposed to recover the error caused by the shifter by using OR gates. The synthesis results in an industrial 28nm CMOS process show that the proposed 16/8 approximate divider using an 8/4 accurate divider is $2.5\times$ as fast and consumes 34.42% of the power of the accurate 16/8 design. Compared with the other approximate dividers, the proposed design is significantly more accurate at a similar power-delay product. Moreover, simulation results show that the proposed approximate divider outperforms the other designs in two image processing applications.**

***Index Terms*—adaptive approximation, approximate divider, overflow, accuracy, low-power**

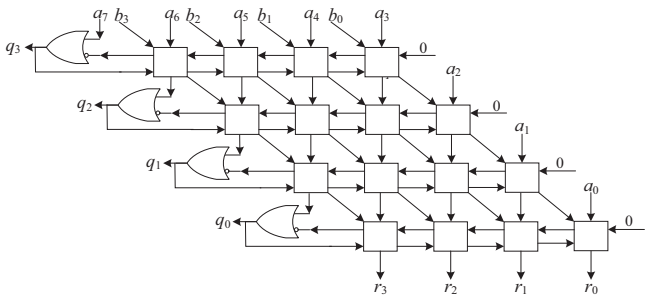
I. Introduction

Compared with multiplication and addition, division is a less frequently used arithmetic operation [1]; however, its long latency determines the speed of an application once it is used. Several schemes have been proposed to improve the performance of division, such as using a high-radix [2] or a carry/borrow lookahead circuit in an array divider [3]. However, the improvement in performance is usually obtained at the expense of a high power dissipation and a large area due to the complexity of its intrinsic structure.

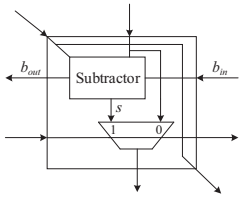
On the other hand, precise computing is not required for many applications such as image processing, clustering and recognition due to the inherent error tolerance. It has been the main reason for the wide interest and investigation into the emerging paradigm of approximate or inexact computing [4]. A large number of approximate designs have been proposed for multiplication and addition for pursuing improvements in speed, power dissipation and/or circuit complexity [5]. Compared with multiplication and addition, less attention has been paid to the approximation of division.

Division can be implemented in sequential and combinational circuits. Sequential division is usually implemented by using the digit recurrent algorithm [6] or the functional iterative algorithm [7]. Its latency is significantly longer than a combinational divider due to the recurrent/iterative nature of its operation. A combinational division is implemented by shift and subtraction/addition operations. An $8/4$ unsigned restoring array divider is shown in Fig. 1. In general, n^2 subtractor cells are required in a $2n/n$ unsigned restoring array divider. Due to the borrow ripples in each row, the critical path is in $O(n^2)$, while it is in $O(n)$ for an $n \times n$ array multiplier. Thus, an array divider incurs a higher hardware overhead and a lower speed than multipliers.

In [8], [9], three approximate subtractors have been proposed for an array divider by simplifying the transistor-level circuit of an accurate subtractor. Four replacement schemes, namely vertical, horizontal, square and triangle replacements, are then considered for nonrestoring [8] and restoring array dividers [9]. To ensure a high accuracy, only the less significant part of the divider is approximated. Therefore, the critical path delays of these designs, denoted as AXDnr and AXDr, are not significantly reduced. The savings in power dissipation and area are also small compared with the accurate array divider.



(a) An $8/4$ unsigned restoring array divider



(b) Subtractor cell

Fig. 1. An $8/4$ unsigned restoring array divider with constituent subtractor cells [1].

978-3-9819263-0-9/DATE18/©2018 EDAA

To save additional hardware, a dynamic approximate divider (DAXD) has been designed [10]. In a $2n/n$ DAXD, $2k/k$ bits in the inputs are dynamically selected starting from the most significant '1', whereas the least significant bits (LSBs) are truncated. The quotient is finally obtained approximately by using a $2k/k$ exact divider and a shifter. As the circuit complexity and critical path of a $2n/n$ array divider are in $O(n^2)$ [1], DAXD shows a substantial improvement in speed, area and power consumption compared with AXDnr and AXDr. However, the accuracy of DAXD is very low due to the overflow problem caused by the truncation and the $2k/k$ divider.

In addition to array dividers, a rounding-based approximate divider, referred to as SEERAD, has been proposed [11]. To compute A/B, the divisor B is rounded to a form of $2^{K+L}/D$, where $K = \lfloor log_2(B) \rfloor$, and L and D are constant integers. Then, the value of A/B is approximated by $A \times D/2^{K+L}$. By arranging D and L, four accuracy levels are devised for SEERAD. Thus, approximate division is implemented by a rounding block, a look-up table, a small multiplier, adders and a shifter. Without using a traditional division structure, SEERAD is fast, but it incurs a substantial power dissipation and a large area due to the use of the look-up table.

In this paper, a novel approximate unsigned divider using adaptive approximation is proposed for low-power and high-performance operation. In this design, input pruning, division strategy, and error correction work synergistically to ensure a high accuracy with a very low maximum error distance.

This paper presents the following novel contributions. Adaptive pruning schemes are analyzed in detail for four different scenarios of the dividend and divisor. Based on this analysis, new division strategies are proposed to avoid the possible occurrence of overflow found in the approximate divider in [10]. Finally, an error correction circuit using OR gates is utilized for achieving a high accuracy at a very small hardware overhead.

Compared with the exact $16/8$ array divider, the proposed adaptive approximation-based divider (denoted as AAXD) using an $8/4$ divider achieves a speedup by 60.51%, a reduction in power dissipation by 65.88% and in area by 38.63%. For a more accurate configuration using a $12/6$ divider, the AAXD is 26.54% faster and 34.13% more power efficient than the accurate design. Two image processing applications, change detection and foreground extraction, show that a higher image quality is obtained by using the proposed design than using other approximate dividers.

II. Proposed Approximate Divider

A. Motivation

Two widely used division algorithms are the nonrestoring division and restoring division. In the restoring division, the partial remainder is corrected when a subtraction yields a negative result, whereas it is not corrected in the nonrestoring division. In this paper, the more commonly used restoring division is considered. Different from multiplication and addition, the inputs of division have a strict range requirement. In a $2n/n$ divider, the n most significant bits (MSBs) of the dividend A must be smaller than the divisor B to guarantee that no overflow occurs [1].

For approximations in an adder or a multiplier, truncation is an efficient approach to reducing hardware consumption [5], [12]. Improvements in power dissipation and critical path delay can also be obtained for an approximate divider design; however, the static approximation using traditional truncation on the LSBs of the input operands results in large relative errors, especially for small input operands. Thus, the proposed approximate divider uses adaptive approximation by selectively discarding some LSBs of the input operands; then, a smaller divider is used to process the remaining bits.

B. Design

The basic structure of the proposed approximate unsigned divider is shown in Fig. 2. In this design, $2k$ (or k) MSBs of the dividend (or divisor) are adaptively chosen from the $2n$ (or n) inputs using leading one position detectors (LOPDs) and multiplexers (here, $k < n$), according to the pruning scheme. An exact $2(k+1)/(k+1)$ divider is then used to compute the division of the selected bits. The $(k+1)$-bit quotient is shifted by a shifter for a number of bits calculated by a subtractor, which results in a $(n+1)$-bit intermediate result. Finally, the n-bit approximate quotient is obtained by correcting the $(n+1)$-bit intermediate result using an error correction circuit. The detailed structure of each circuit in Fig. 2 is discussed next.

C. Input Pruning

Fig. 3 shows a straightforward pruning scheme for a $2n$-bit unsigned dividend $A = \sum_{i=0}^{2n-1} a_i 2^i = (a_{2n-1}a_{2n-2}\cdots a_1 a_0)_2$. To obtain a $2k$-bit dividend, '0's at the bit positions higher than the most significant '1' are truncated; the redundant LSBs are pruned if the number of remaining LSBs is larger than $2k$. Similarly, a k-bit number is determined from the n-bit divisor $B = \sum_{i=0}^{n-1} b_i 2^i = (b_{n-1}b_{n-2}\cdots b_1 b_0)_2$.

Let the bit positions of the most significant '1', known as the leading '1' positions, for A and B be l_A and l_B, respectively. Input operands of a division can be determined as in Fig. 3 when l_A and l_B are larger than or equal to $2k-1$ and $k-1$, respectively. A different pruning scheme is required for the input operands when $l_A < 2k-1$ or $l_B < k-1$. Therefore,

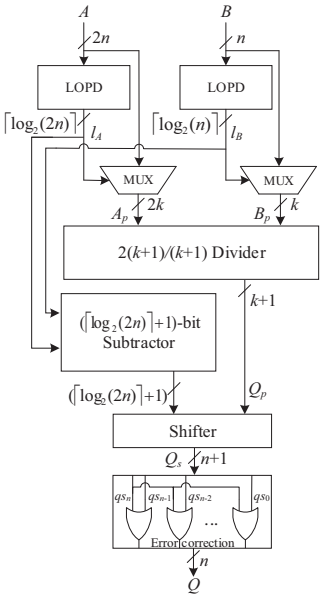


Fig. 2. Proposed adaptive approximation-based divider (AAXD). LOPD: leading one position detector.

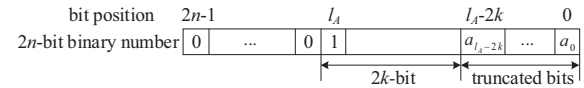


Fig. 3. Pruning scheme for a $2n$-bit unsigned number A when $l_A \geq 2k-1$ [10].

four scenarios are discussed here for different values of l_A and l_B to find the most appropriate pruning scheme.

(i) $l_A \geq 2k-1$ and $l_B \geq k-1$

In this case, the pruned dividend $A_p = 2^{l_A} + \sum_{i=l_A-2k+1}^{l_A-1} a_i 2^i = (1a_{l_A-1}\cdots a_{l_A-2k+1})_2$, and the pruned divisor $B_p = 2^{l_B} + \sum_{i=l_B-k+1}^{l_B-1} b_i 2^i = (1b_{l_B-1}\cdots b_{l_B-k+1})_2$, then the division becomes

$$\frac{A_p}{B_p} = \frac{(1a_{l_A-1}\cdots a_{l_A-2k+1})_2}{(1b_{l_B-1}\cdots b_{l_B-k+1})_2}. \quad (1)$$

A proper sized divider should be used to eliminate overflow for the largest possible quotient of A_p/B_p. The largest quotient is obtained when $A_p = (11\cdots1)_2 = 2^{2k}-1$ and $B_p = (10\cdots0)_2 = 2^{k-1}$, which is given by

$$\lfloor \frac{2^{2k}-1}{2^{k-1}} \rfloor = 2^{k+1}-1. \quad (2)$$

As the bit-width of the output for a $2k/k$ divider is k, overflow occurs when the quotient is larger than 2^k-1. This indicates that overflow is possible when using a $2k/k$ divider to compute A_p/B_p even when there is no overflow for a $2n/n$ divider computing A/B. As per (2), the output of the reduced-width divider should use at least $k+1$ bits to avoid overflow. Therefore, the $2k$-bit pruned dividend is expanded to $(2k+2)$-bit by adding two '0's at the $(2k+2)^{th}$ and $(2k+1)^{th}$ bit positions; a '0' is added to the $(k+1)^{th}$ bit position of the pruned divisor. Then, a $2(k+1)/(k+1)$ divider is used to compute the division. No overflow occurs because $(001a_{l_A-1}\cdots a_{l_A-k+2})_2$ is always smaller than $(01b_{l_B-1}\cdots b_{l_B-k+1})_2$.

As $A \approx A_p \cdot 2^{l_A-2k+1}$, and $B \approx B_p \cdot 2^{l_B-k+1}$, the approximate A/B is given by

$$\frac{A}{B} \approx \frac{2^{l_A-2k+1}A_p}{2^{l_B-k+1}B_p} = \lfloor \frac{A_p}{B_p} \rfloor 2^{l_A-l_B-k}. \quad (3)$$

This is the quotient of the $2(k+1)/(k+1)$ divider multiplied by $2^{l_A-l_B-k}$. In this case, the multiplication is implemented by left shifting $\lfloor \frac{A_p}{B_p} \rfloor$ for l_A-l_B-k bits.

The largest possible value of l_A-l_B-k is $n-k$ because $l_A-l_B \leq n$, in which case the approximate quotient is $(n+1)$-bit. It is generated by left shifting the $(k+1)$-bit quotient of the reduced-width divider for $(n-k)$ bits. To ensure an n-bit output for a $2n/n$ divider, the quotient is approximated by $2^n-1 = (11\cdots1)_2$ using an error correction circuit when the n^{th} bit of the shifted result is '1'.

The smallest possible value of l_A-l_B-k is $-n-k+1$, in which case the output quotient is a fractional value. As only integer numbers are considered for a $2n/n$ unsigned divider, the quotient is approximated by 0 when l_A-l_B-k is smaller than or equal to $-(k+1)$.

(ii) $l_A \geq 2k-1$ and $l_B < k-1$

When $l_B < k-1$, the most significant '1' of B is located in one of its k LSBs. Thus, the pruning scheme in Fig. 3 is not applicable. In [10], k LSBs of B are selected as the divisor for a $2k/k$ divider, which indicates $B_p = (b_{k-1}\cdots b_0)_2$. Then, the quotient is given by

$$\frac{A_p}{B_p} = \frac{(1a_{l_A-1}\cdots a_{l_A-2k+1})_2}{(b_{k-1}\cdots b_0)_2}. \quad (4)$$

As $l_B < k-1$, $b_{k-1} = 0$ and hence, $(1a_{l_A-1}\cdots a_{l_A-k+1})_2$ is always larger than $(b_{k-1}\cdots b_0)_2$. Overflow is possible even when a $2(k+1)/(k+1)$ divider is used.

To solve this problem, another pruning scheme is designed for $l_B < k-1$, as shown in Fig. 4. The k-bit B_p is composed of l_B+1 LSBs in B as the higher bits and $k-l_B-1$ '0's as the lower bits, i.e., $B_p = (1b_{l_B-1}\cdots b_0 0\cdots0)_2$. Then, the division becomes

$$\frac{A_p}{B_p} = \frac{(1a_{l_A-1}\cdots a_{l_A-2k+1})_2}{(1b_{l_B-1}\cdots b_0 0\cdots0)_2}. \quad (5)$$

This is similar to (1) in scenario (i). Thus, 2-bit and 1-bit '0's are appended to the most significant positions of A_p and B_p, and a $2(k+1)/(k+1)$ divider is used to compute A_p/B_p to avoid overflow. The approximate result of A/B is also given by (3).

(iii) $l_A < 2k-1$ and $l_B \geq k-1$

Note that the dividend A can be zero, in which case l_A is set to zero (i.e., with the same leading one position as number $(00\cdots01)_2$). Because $A_p = (a_0 0\cdots0)_2$ when $l_A = 0$, the quotient is obtained as 0 no matter a_0 is '0' or '1'. As discussed above, A_p and B_p are pruned using the schemes shown in Fig. 4 and Fig. 3, respectively. Thus, the same approximate division is obtained by using a $2(k+1)/(k+1)$ divider as in (3).

(iv) $l_A < 2k-1$ and $l_B < k-1$

Both the input operands of the division are pruned using the scheme in Fig. 4. In this scenario, an accurate $2n/n$ division is performed by using a $2(k+1)/(k+1)$ divider.

D. Leading One Position Detection (LOPD)

As shown in Fig. 2, a leading one position detector (LOPD) is used to detect the bit position of the most significant '1' in each input. It is implemented by a priority encoder. Table I is the truth table for the function of an 8-to-3 priority encoder, i.e.,

$$O_0 = I_7 \vee \overline{I}_6(I_5 \vee \overline{I}_4 I_3 \vee \overline{I}_4\overline{I}_2 I_1), \quad (6)$$

$$O_1 = I_7 \vee I_6 \vee \overline{I}_5\overline{I}_4(I_3 \vee I_2), \quad (7)$$

$$O_2 = I_7 \vee I_6 \vee I_5 \vee I_4, \quad (8)$$

where the disjunction "$\vee$" is implemented by an OR operation.

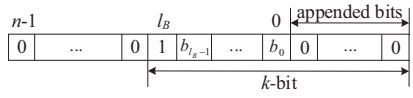


Fig. 4. Pruning scheme for an n-bit unsigned number B when $l_B < k-1$.

TABLE I. Truth table of an 8-to-3 priority encoder.

Inputs								Outputs		
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

The leading one positions (l_A and l_B) are then used to determine the $2k$-bit A_p and the k-bit B_p from the $2n$-bit dividend and the n-bit divisor, respectively. Multiplexers are used to implement the pruning schemes in Figs. 3 and 4. The pruned inputs A_p and B_p are then processed by using a $2(k+1)/(k+1)$ divider. Note that the structure of the $2(k+1)/(k+1)$ divider can be different according to specific application requirements, e.g., an array divider, a sequential divider or a high-radix divider. Meanwhile, the shifting direction and number of bits are computed by subtracting the two leading one positions using a $(\lceil log_2(2n) \rceil + 1)$-bit subtractor. $(n+1)$-bit intermediate result Q_s is generated after left shifting the $(k+1)$-bit output of the reduced-width divider for $l_A - l_B - k$ bits. Finally, the error correction circuit uses n OR gates to perform $q_i = qs_i \vee qs_n$, $i = 0, 1, \cdots, n-1$, where q_i and qs_i are the i^{th} LSBs of Q and Q_s, respectively. This circuit corrects the erroneous results that are larger than $2^n - 1$ ($qs_n = 1$) to $2^n - 1 (qs_i = 1$ for $i = 0, 1, \cdots, n-1)$, which ensures that an n-bit approximate quotient is obtained.

The most significant circuit of the proposed approximate divider is the $2(k+1)/(k+1)$ divider, whereas other components (LOPD, multiplexer, subtractor and shifter) are relatively small. Moreover, the subtractor works in parallel with the $2(k+1)/(k+1)$ divider. Thus, the circuit complexity and critical path of the approximate divider are close to $O((k+1)^2)$ when a $2(k+1)/(k+1)$ array divider is used. This is significantly smaller compared with that of the exact array divider ($O(n^2)$), especially for a small k.

III. Simulation Results

To assess the accuracy and circuit characteristics, the proposed design is implemented in MATLAB and VHDL. The other approximate dividers, AXDr, DAXD and SEERAD, are considered for comparison.

A. Error Characteristics

The error rate (ER), normalized mean error distance ($NMED$), mean relative error distance ($MRED$) and the maximum error distance (ED_{max}) are considered to evaluate the accuracy of $16/8$ approximate dividers. The $NMED$ is defined as the mean value of the error distances normalized by the maximum possible accurate output. The $MRED$ is the mean value of the relative error distance that is the absolute ratio between the error distance and the accurate output. All valid combinations in the range of $[0, 65535]$ and $(0, 255]$ are used as the input dividends and divisors. They are carefully selected to meet the no overflow condition of an accurate 16/8 divider. The simulation results are shown in Table II, in which AXDr1, AXDr2, and AXDr3 are the approximate restoring array dividers with triangle replacement using approximate subtractor 1, 2, and 3, respectively [9]. The parameter value is the replacement depth for AXDrs, while it is the accuracy level for SEERAD. For DAXD and the proposed adaptive approximate divider (AAXD), the parameter value is the bit-width of the pruned dividend A_p.

TABLE II. Error characteristics of approximate 16/8 dividers.

Divider	Parameter	ER (%)	$NMED$ (%)	$MRED$ (%)	ED_{max}
AXDr1	10	80.94	1.32	4.32	116
AXDr1	9	71.12	0.67	2.45	102
AXDr1	8	50.93	0.29	1.21	51
AXDr2	10	93.51	2.45	11.88	245
AXDr2	9	88.19	1.33	6.20	227
AXDr2	8	78.38	0.72	3.38	160
AXDr3	10	78.64	0.97	3.25	119
AXDr3	9	66.63	0.51	1.84	109
AXDr3	8	48.39	0.26	0.96	85
SEERAD	1	99.99	7.64	15.58	96
SEERAD	2	99.99	4.11	8.52	64
SEERAD	3	99.99	2.23	4.97	81
SEERAD	4	99.99	1.09	2.71	165
DAXD	8	91.43	7.44	16.39	240
DAXD	10	85.57	6.65	14.74	224
DAXD	12	75.77	6.39	13.41	205
AAXD	6	91.06	2.97	6.61	49
AAXD	8	84.49	1.46	3.12	27
AAXD	10	73.74	0.72	1.52	14

Note: The parameter value is for the replacement depth and the accuracy level for AXDrs and SEERAD, respectively. It is the bit-width of the pruned dividend in DAXD and AAXD.

Table II shows that the proposed AAXD has the smallest ED_{max}, whereas DAXD has the largest ED_{max} due to the overflow caused by approximation. The ED_{max} of AXDr2 is also very large. Among all designs, AXDr1 and AXDr3 have relatively small ERs, whereas SEERAD has the largest ER that is close to 100%. AAXD shows similar moderate ERs as AXDr2 and DAXD. In terms of $NMED$, AXDrs show the best performance, and AAXD has slightly larger values. DAXD and SEERAD of accuracy levels 1 and 2 result in very large values of $NMED$. The $MRED$ shows a similar trend with the $NMED$ except that AXDr2 with a depth of 10 results in a very large $MRED$.

In summary, the proposed AAXD is very accurate in terms of ED_{max}, $NMED$ and $MRED$ compared with the other approximate designs. AXDr1 and AXDr3 are also very accurate because only some less significant subtractors are approximated; however, their hardware improvements are very limited, as shown next. The accuracy of DAXD is lower than other designs due to the possible overflow.

B. Circuit Measurements

To obtain the circuit measurements, the approximate dividers and the exact unsigned restoring array divider (EXDr) are implemented in VHDL and synthesized in ST's 28nm CMOS process using the Synopsys Design Compiler under the same voltage, temperature and frequency. The supply voltage is 1 V, the simulation temperature is 25°C, and the frequency used for power estimation is 200 MHz. The critical path delay and area are reported by the Synopsys Design Compiler. The power dissipation is estimated by using the PrimeTime-PX tool with 5 million random input combinations. For ease of comparison, the same array structure and subtractor cells are used in the accurate part of AXDrs, DAXD and AAXD. As more complex figures

TABLE III. Circuit measurements of the considered dividers.

Divider	Parameter	Delay (ns)	Area (μm^2)	Power (μW)	PDP (fJ)	ADP ($ns \cdot \mu m^2$)
EXDr	**–**	**4.71**	**285.8**	**128.00**	**602.88**	**1,345.9**
AXDr1	10	4.38	280.2	113.90	498.88	1,227.3
AXDr1	9	4.40	281.2	116.70	513.48	1,237.3
AXDr1	8	4.44	282.3	119.50	530.58	1,253.6
AXDr2	10	4.65	252.6	94.68	440.26	1,174.7
AXDr2	9	4.67	259.5	100.80	470.74	1,211.8
AXDr2	8	4.69	267.5	108.00	506.52	1,254.5
AXDr3	10	4.38	216.6	59.98	262.71	948.6
AXDr3	9	4.39	227.3	70.38	308.97	998.0
AXDr3	8	4.42	239.6	82.29	363.72	1,058.9
SEERAD	1	1.15	204.3	56.04	64.45	235.0
SEERAD	2	2.02	253.1	80.61	162.83	511.3
SEERAD	3	1.81	333.4	107.80	195.12	603.5
SEERAD	4	2.23	480.1	169.80	378.65	1,070.7
DAXD	8	2.06	206.9	53.49	110.19	426.3
DAXD	10	2.73	245.5	63.83	174.26	670.1
DAXD	12	3.69	286.9	86.99	320.99	1,058.7
AAXD	6	1.86	175.4	43.67	81.23	326.3
AAXD	8	2.59	231.4	61.82	160.11	598.5
AAXD	10	3.46	294.9	84.31	291.71	1,020.4

of merit, the power-delay product (PDP) and area-delay product (ADP) are calculated from the measurements. The results are reported in Table III.

Compared with the accurate design, the proposed 16/8 AAXD with a 6-bit pruned dividend is 60.51% faster and achieves 38.63% and 65.88% reductions in area and power dissipation, respectively. Using a $12/6$ accurate divider (for a 10-bit pruned dividend), the AAXD incurs a 26.54% shorter delay and consumes a smaller power by 34.13% than the accurate design, albeit with a 3.18% increase in area due to the additional circuits of LOPDs, multiplexers, subtractor and shifter. Overall, the PDP and ADP of the proposed design are reduced by 51.61% to 86.53%, and 24.18% to 75.76%, respectively.

Among all considered designs, SEERAD shows the shortest delay because its critical path is significantly reduced due to the use of a multiplier instead of a divider structure. However, SEERAD incurs a large area and high power consumption when its accuracy level is 3 or 4 due to the lookup table used for storing the constants. Although SEERAD-1 (for accuracy level 1) and SEERAD-2 (for accuracy level 2) are more power and area efficient with a very short delay, their accuracy is significantly lower than the other approximate dividers, as shown in Table II.

The hardware improvements for AXDr1 and AXDr2 are very minor compared with the accurate counterpart, although the power and area reductions are larger for AXDr3. Moreover, AXDrs are the slowest because replacing the exact subtractors with approximate ones does not significantly reduce the carry/borrow chain or the critical path. DAXD shows a rather small delay and power dissipation, but its area is slightly larger than the accurate design when a $12/6$ accurate divider is used.

Compared with the other approximate dividers, the proposed AAXD outperforms AXDr1 and AXDr2 in delay, area and power dissipation. Also, it shows a shorter delay and a similar power dissipation and therefore, smaller values of PDP and ADP (except for AAXD-10) compared with AXDr3. Using a same sized accurate divider, AAXD is faster and more power efficient than DAXD. Compared with the two SEERADs with higher accuracy, SEERAD-3 and SEERAD-4, AAXD-8 and AAXD-10 show smaller PDP and ADP.

C. Discussion

For a further comparison of approximate dividers, the error and circuit measures are jointly considered. The metrics $MRED$ and PDP are selected as representatives to show the error and circuit characteristics. As shown in Fig. 5, the proposed AAXD has a much smaller value of $MRED$ than the other approximate designs when a similar PDP is considered. AXDr3 also shows a good tradeoff in $MRED$ and PDP with a higher accuracy, however its delay is very long. Although some configurations of AXDr1 and AXDr2 show small $MRED$s, their PDPs are generally high. On the contrary, DAXD has a very low PDP but a significantly large $MRED$. The $MRED$ and PDP are moderate for SEERAD, and they vary with the accuracy level. Overall, the proposed AAXD shows the best tradeoff among the considered approximate dividers.

IV. Image Processing Applications

In addition to human perceptual limitations, some image processing algorithms are inherently error tolerant. Therefore, approximate circuits have widely been used in image processing to improve hardware efficiency. As two common applications of dividers, change detection and foreground extraction [13] are considered to further assess the accuracy of approximate dividers.

Change Detection: Change detection in image processing can be implemented by computing the ratio of two pixel values using a divider. For the two 8-bit gray-level images in Fig. 6(a) and (b), the pixel values of the first image are multiplied by 64 as the dividends at a higher precision. Thus, $16/8$ dividers are sufficient for this application. The designs with similar values of PDP (about 300 fJ) are selected, including AXDr3-9, DAXD-12 and AAXD-10 (see Fig. 5). For the other approximate designs, configurations with PDPs close to 300 fJ are selected, including AXDr1-10, AXDr2-10 and SEERAD-4.

Fig. 6 shows the output images for change detection; the peak signal-to-noise ratios (PSNRs) are shown in the parentheses. The gray-level images are obtained by scaling the division results to 8-bit pixel values using

$$P_{out} = 255 \times \frac{P_{ratio} - P_{min}}{P_{max} - P_{min}}, \tag{9}$$

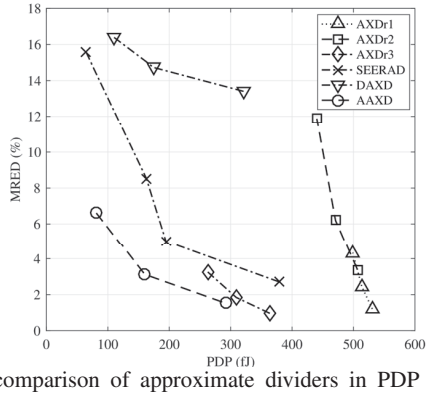
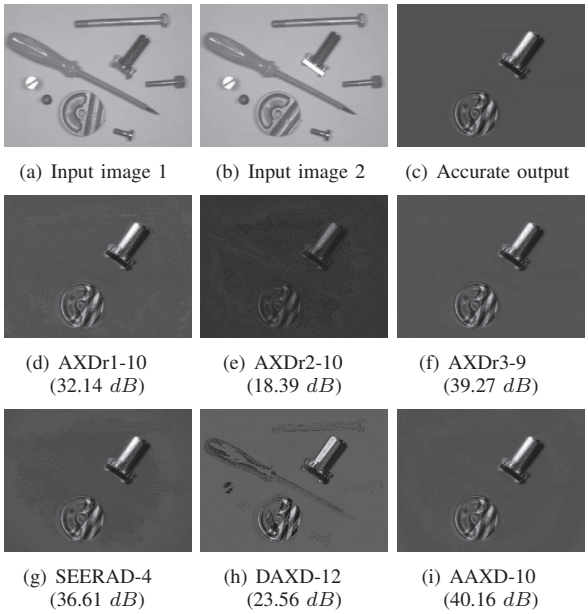


Fig. 5. A comparison of approximate dividers in PDP and MRED. The replacement depths of AXDr1, AXDr2 and AXDr3 are from 8 to 10 from right to left. The accuracy levels of SEERAD are from 1 to 4 from left to right. The pruned dividend width is from 8 to 12 for DAXD, and it is from 6 to 10 for AAXD from left to right.



(a) Input image 1 (b) Input image 2 (c) Accurate output

(d) AXDr1-10 (32.14 dB) (e) AXDr2-10 (18.39 dB) (f) AXDr3-9 (39.27 dB)

(g) SEERAD-4 (36.61 dB) (h) DAXD-12 (23.56 dB) (i) AAXD-10 (40.16 dB)

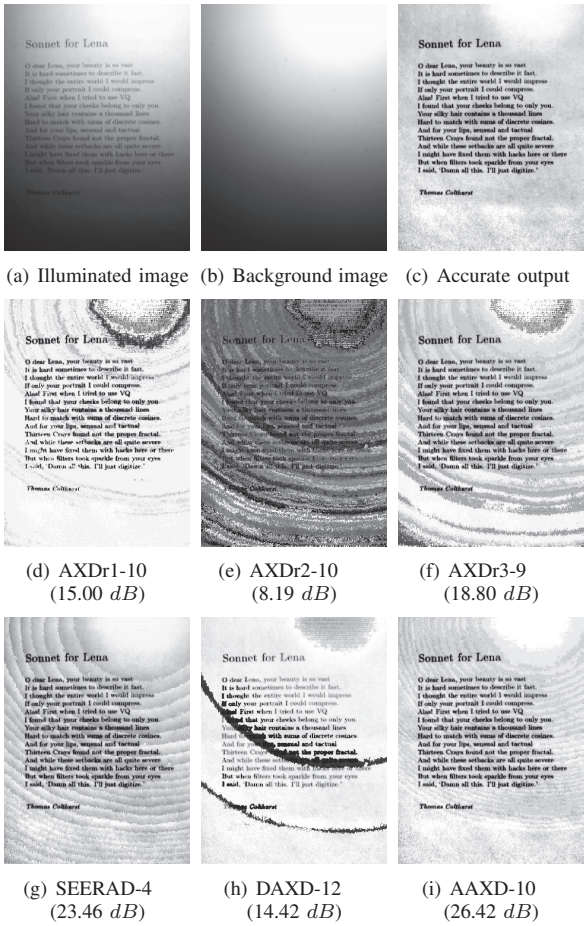
Fig. 6. Change detection quality using different dividers (PSNR).

where P_{ratio} is a computed pixel ratio, P_{max} and P_{min} are the maximum and minimum values of the division results, respectively. As can be seen, AAXD-10 and AXDr3-9 perform similarly well as an accurate divider, whereas AXDr-2 and DAXD-12 produce results with a low quality. AXDr1-10 and SEERAD-4 produce images that are acceptable for a visual inspection. The quantitative evaluation in PSNR produces consistent results as the visual inspection.

Foreground Extraction: By using pixel division, the unwanted illumination can be removed from an illuminated image to show the objects in the foreground more clearly. Similar to the change detection, the 8-bit gray-level pixel values of the illuminated image (Fig. 7(a)) are multiplied by 64 to obtain a higher precision. Fig. 7(b) shows the background that needs to be removed. The same design configurations of the $16/8$ approximate dividers are selected for the foreground extraction as for the change detection. As shown in Fig. 7, the foreground extraction quality (after scaling to 8-bit gray-level images using (9)) by AXDr2-10 is rather poor, and the defects in the image obtained by DAXD-12 are also significant. AAXD-10 produces the most accurate extracted image compared to the one by accurate dividers. The PSNR values in the parentheses indicate that AAXD-10 performs the best, followed by SEERAD-4, and the other designs have poorer performance for foreground extraction.

V. Conclusion

This paper proposes an approximate unsigned divider using adaptive approximation. A novel pruning scheme and error correction circuits are utilized for the divider to attain a high accuracy. The use of a reduced-width divider and a shifter leads to a high-performance and low-power operation. As per the synthesis results in ST's 28nm CMOS process, the proposed design achieves improvements by more than 60% in speed and power dissipation compared with an accurate design. The proposed divider is more accurate than the other approximate dividers when a similar PDP is considered. Two image processing applications illustrate the accuracy and hardware efficiency of the proposed design.



(a) Illuminated image (b) Background image (c) Accurate output

(d) AXDr1-10 (15.00 dB) (e) AXDr2-10 (8.19 dB) (f) AXDr3-9 (18.80 dB)

(g) SEERAD-4 (23.46 dB) (h) DAXD-12 (14.42 dB) (i) AAXD-10 (26.42 dB)

Fig. 7. Foreground extraction quality using different dividers (PSNR).

References

[1] B. Parhami, *Computer arithmetic*. Oxford university press, 2000.

[2] J. E. Robertson, "A new class of digital division methods," *IRE Transactions on Electronic Computers*, no. 3, pp. 218–222, 1958.

[3] M. Cappa and V. C. Hamacher, "An augmented iterative array for high-speed binary division," *IEEE Transactions on Computers*, vol. 100, no. 2, pp. 172–175, 1973.

[4] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm For Energy-Efficient Design," in *ETS*, 2013.

[5] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM JETC*, vol. 13, no. 4, p. 60, 2017.

[6] W. Liu and A. Nannarelli, "Power efficient division and square root unit," *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1059–1070, 2012.

[7] M. J. Flynn, "On division by functional iteration," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 702–706, 1970.

[8] L. Chen, J. Han, W. Liu, and F. Lombardi, "Design of approximate unsigned integer non-restoring divider for inexact computing," in *GLSVLSI*, pp. 51–56, 2015.

[9] ——, "On the design of approximate restoring dividers for error-tolerant applications," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2522–2533, 2016.

[10] S. Hashemi, R. Bahar, and S. Reda, "A low-power dynamic divider for approximate applications," in *DAC*, 2016.

[11] 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 *DATE*, pp. 1481–1484, 2016.

[12] B. Barrois, O. Sentieys, and D. Menard, "The hidden cost of functional approximation against careful data sizing–A case study," in *DATE*, pp. 181–186, 2017.

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