# Fast and Scaled Counting-Based Stochastic Computing Divider Design

Yibo Liu, *Graduate Student Member, IEEE*, Shuyuan Yu, *Member, IEEE*, Maliha Tasnim, *Graduate Student Member, IEEE*, and Sheldon X.-D. Tan, *Senior Member, IEEE*

***Abstract*—This article presents novel designs for stochastic computing (SC)-based dividers, which promise low latency, high energy efficiency, as well as high accuracy for error-tolerant arithmetic operations. We first introduce CBDIV, which is based on the recently proposed counter-based SC concept and correlation-based SC to perform division. Then, we introduce FSCDIV, which further improves the accuracy of CBDIV by applying a scaling strategy and mitigating the latency by optimizing the counting scheme. The FSCDIV will equally scale up the divider and dividend before the division process, and thereby avoid large relative error when both input values of the divider and dividend are small. The proposed fast counting method accelerates FSCDIV by counting new bit pair (0–1 pair) among only half of the stochastic number bitstream instead of the entire bitstream, resulting in almost half of the counting latency and one-fourth of the overall division operation latency. The experimental results demonstrate that the proposed CBDIV, implemented in a 32-nm technology node, outperforms state-of-the-art works by 77.8% in accuracy, 37.1% in delay, 21.5% in area, 50.6% in area delay product (ADP), and 25.9% in power consumption. Compared to the fixed-point division baseline, CBDIV also achieves a 31.9% reduction in energy consumption and is more energy efficient than existing SC-based dividers for binary inputs and outputs required in efficient image processing implementations. Moreover, we demonstrate that FSCDIV improves delay by 56.4%, ADP by 16.0%, energy consumption by 45.0%, and accuracy by 61.2%. We also evaluate CBDIV and FSCDIV designs in a contrast stretch image processing workload, and the results show that the proposed designs can improve the image quality by up to 18.3 dB on average when compared to state-of-the-art works.**

***Index Terms*—Approximate computing, arithmetic circuit design, division, stochastic circuit design, stochastic computing (SC).**

## I. Introduction

STOCHASTIC computing (SC) has emerged as a potential alternative to traditional binary computing, offering an inexpensive and error-tolerant solution. SC's superior error resilience, its more balanced performance tradeoffs, including accuracy, energy, and its economical implementation of intricate arithmetic tasks, provide it a clear edge over binary designs. SC's simplicity allows numerous arithmetic tasks, like multiplication, to be implemented using merely an AND operation, or an XNOR gate for bipolar cases. These characteristics have paved the way for its use in various error-resistant workloads, such as error-correcting codes [1], image processing applications [2], and deep neural networks (DNNs) [3], [4], [5], [6], [7], [8], [9]. Furthermore, some recent research extended SC applications. For instance, [10] applied SC to numerical simulation, [11] integrated SC to sound source localization system, and [12] applied SC to build energy-efficient Bayesian neural networks.

Despite these advantages, the ease of SC's hardware implementation is not without challenges: for starters, conventional SC requires $2^N$ cycles to manage $N$-bit precision binary inputs, a potentially lengthy process particularly for large $N$, making SC more fitting for applications that can tolerate lower precision. Second, the precision of SC tasks hinges on the randomness of two bitstreams (ideally uncorrelated) in conjunction with the length of the bitstream. Consequently, a substantial body of research has been initiated to devise top-tier random number generators (RNGs) showcasing minimal or zero correlation, such as low-discrepancy (LD) sequences [13], [14] and bit scrambling methods [15], [16].

More recently, an advanced and precision-oriented SC multiplier has been introduced to address the two issues mentioned above in the traditional SC [5]. This new multiplier, instead of resorting to an AND gate for multiplying two bitstreams, essentially keeps track of the "1"s in one multiplicand bitstream based on the value of the other multiplicand bitstream. The bitstream that is counted can be deterministically created based on a finite state machine (FSM). Thus, the overall design boils down to two counters coupled with a simple bitstream generator. We have named this design the *counting-based SC multiplier* (CBSC-Multiplier). The CBSC-Multiplier comes with two vital advantages: first, it eliminates the need for bitstream randomness without any accuracy compromise, proving highly compatible with the correlation demands of existing SC-based methods. Second, it potentially outperforms the traditional SC due to its ability to execute early termination (partial counting), a feature not readily achievable in the conventional SC methods. Although Wu et al. [17] proposed normalized stability, a universe measurement metric to evaluate the circuit's potential for early termination, it is not optimized for specific SC operation design.

Manuscript received 1 June 2023; revised 26 August 2023, 19 November 2023, and 25 January 2024; accepted 2 February 2024. Date of publication 14 February 2024; date of current version 19 July 2024. This work was supported in part by the National Science Foundation under Grant CCF-2113928 and Grant CCF-2305437. This article was recommended by Associate Editor X. Jiao. *(Corresponding author: Yibo Liu.)*

The authors are with the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA 92521 USA (e-mail: yliu401@ucr.edu).

Digital Object Identifier 10.1109/TCAD.2024.3365426

In the realm of division operations underpinned by stochastic computing (SC), current research suggests that the tasks can be conducted considerably more effectively when the two input bitstreams exhibit a high degree of correlation, as division can be approximated by calculating the conditional probability of the two streams [18], [19]. To reconcile the distinct correlation prerequisites between traditional SC divisions, a translation module, referred to as a skewed synchronizer, was presented, albeit at the expense of increased area overheads [19]. Chu et al. [20] applied a JK flip flop-based SC divider, which has some hardware improvement in certain applications. Asadi et al. [21] proposed an LD SC method to reduce the hardware cost and improve the accuracy. Shaowei et al. [22] proposed to exploit the maximally correlated input bitstreams to improve the accuracy with almost no extra hardware cost.

Inspired by the aforementioned correlated division and FSM-based deterministic bitstream generation, in this article, we propose novel SC-based division designs, our main key contributions are as follows.

1) We first introduce CBDIV design, which takes advantage of both the correlation stipulation inherent to established SC-based division approaches and the more efficient counting-based SC (CBSC) framework, resulting in a much more efficient partial counting process.
2) Then, we introduce FSCDIV design, which further extends CBDIV by introducing three new ideas—a new scaling method for both the divisor and dividend to improve accuracy, reducing half of the counting latency by counting bit pairs (0–1 pair), and a new selective counting strategy.
3) The experimental results demonstrate that the proposed CBDIV design, implemented in a 32-nm technology node, outperforms state-of-the-art SC dividers in terms of delay (by 37.1%), area (by 21.5%), area delay product (ADP) (by 50.6%), and power (by 25.9%). CBDIV also reduces energy consumption by 31.9% when compared to the fixed-point division design. To evaluate the error behavior of CBDIV, we use root-mean-square error (RMSE) and compare it with state-of-the-art works and the fixed-point divider. Our numerical results show that CBDIV outperforms state-of-the-art works by at least 77.8% on average. Furthermore, CBDIV with 5-bit precision can outperform state-of-the-art works with 7-bit precision in accuracy by 15.4%. Upon improving CBDIV, FSCDIV further enhances delay (by 56.4%), ADP (by 16.0%), energy consumption (by 45.0%), and accuracy (by 61.2%) over the CBDIV design. We also evaluate the CBDIV and the FSCDIV in a contrast stretch image processing workload, demonstrating that the proposed designs improve image quality by an average of 20.6 and 23.5 dB, respectively, when compared to state-of-the-art works baseline.

The proposed CBSC-based division design can be extrapolated to additional nonlinear arithmetic tasks, paving the way for more extensive CBSC. The CBDIV and FSCDIV can integrate effortlessly with CBSC multiplication, rendering them suitable for hybrid computing in scenarios where both binary and SC bitstreams are essential to optimize different arithmetic operations for efficiency.

This article is organized as follows. Section II reviews some recently proposed SC division methods and a state-of-the-art work as preliminary knowledge. Sections III and IV present the proposed CBDIV and FSCDIV design. Experimental results for the hardware performance, error metrics of FSCDIV design and the comparison with other SC dividers, fixed-point dividers, and a state-of-the-art work are summarized in Section V. Section VI concludes this article.

## II. Review of Related Work

This section provides an overview of several pertinent studies that are relevant to the newly proposed division designs.

### *A. Traditional SC-Based Division Design*

Conventional SC division adheres to Gaines's model [23], which employs an up–down counter to achieve a state of equilibrium when the values of two stochastic numbers (SNs), $x_{1,SN}$ and $x_{2,SN} \cdot p_{z,SN}$, are equal, where $p_{z,SN}$ equals to $x_{1,SN}$ divided by $x_{2,SN}$. Therefore, it necessitates SC multiplication for $x_{2,SN} \cdot p_{z,SN}$. Gaines's design was found to require an extended period to converge [18], and it mandated that both the dividend and the divisor be uncorrelated. A recent development by Temenos and Sotiriadis [24] addressed these shortcomings through the implementation of a deterministic FSM-based stochastic division design (DFSMDIV), which dispelled the need for independence between the input divisor and dividend SN bitstreams. Nevertheless, DFSMDIV demands an average of at least 2 orders of magnitude ($10^2$) clock cycles to converge and fails to yield any notable improvements in accuracy.

Another strategy involves establishing the correlation between the divisor and the dividend in the SC bitstream. Chen and Hayes [18] introduced CORDIV, uncovering that division could be conceptualized as the conditional probability of two SN numbers: $P_{X_1|X_2}$. The corresponding formula can be simplified as follows:

$$P_{X_1|X_2} = p_{x_1,x_2}/p_{x_2} = x_{1,SN}/x_{2,SN}. \tag{1}$$

When $x_1$ and $x_2$ share a strong correlation, $p_{x_2}$ represents the event of $x_2$, and $p_{x_1,x_2}$ stands for the joint event of $x_1$ and $x_2$. Consequently, the output probability $P_{\text{Quotient}}$ can be articulated by the ratio of the number of "1" bits in the dividend and divisor SN bitstreams: $N^1_{\text{Dividend}}/N^1_{\text{Divisor}}$. Therefore, CORDIV necessitates that the divisor always surpasses the dividend to prevent the quotient from exceeding one. This research substantiated that, when fed with highly correlated input SN bitstreams, CORDIV could yield significantly more precise results in comparison with Gaines' design [23].

However, the precision of the CORDIV method is contingent upon the quality of the input SN bitstreams. The framework of the CORDIV design is shown in Fig. 1. Furthermore, the SN generator (SNG) shown in Fig. 1 [18] was proposed to mitigate this problem. However, the accuracy still depends on the distribution of bit "1" in input SN bitstreams, shown in Fig. 2, which illustrates that the position

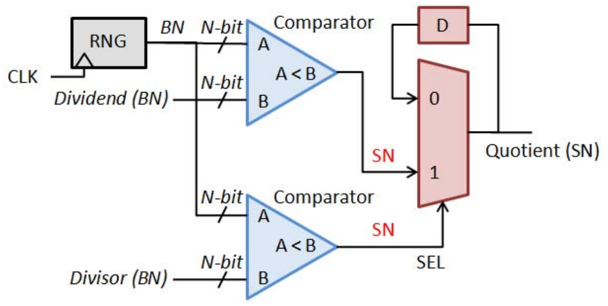


Fig. 1. CORDIV design diagram.

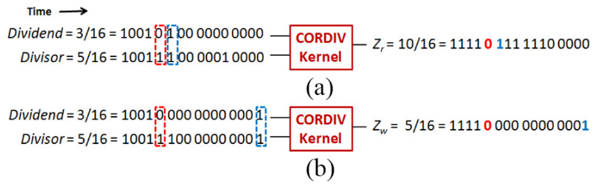

Time
Dividend = 3/16 = 1001 0 1 00 0000 0000
Divisor = 5/16 = 1001 1 1 00 0001 0000
CORDIV Kernel
Z_r = 10/16 = 1111 0 111 1110 0000
(a)
Dividend = 3/16 = 1001 0 000 0000 000 1
Divisor = 5/16 = 1001 1 100 0000 000 1
CORDIV Kernel
Z_w = 5/16 = 1111 0 000 0000 0001
(b)

Fig. 2. CORDIV method input SN bitstreams with different bit distribution. (a) Quotient result with appropriate bit distribution. (b) Quotient result with extreme bit distribution.

of bit "1" in the divisor can significantly influence the accuracy of the quotient.

Specifically, the two input probabilities are 3/16 and 5/16, respectively, with an exact quotient result of 0.6. Based on Fig. 1, when the comparator outputs of the dividend and divisor pair are 0–0 or 1–0, the quotient will output the previous value. When the pair is 1–1, the quotient output will be "1" and when the pair is 0–1, the output will be "0." Based on this observation, given the two different bitstreams of dividends and divisors as shown in Fig. 2, the final results can be quite different. Specifically for the Fig. 2(b) case, with an extreme bit "1" distribution, where too many 0–0 pairs appear between a 1–0 pair and the following 1–1 pair, CORDIV computes $Z_w = 5/16 = 0.3125$, which is a significant deviation from the exact result.

Adding a skewed synchronizer could have improve the accuracy, but still not enough, especially in certain output value ranges [19]. Moreover, all these SC-based division designs encounter issues of accuracy variation across different output value ranges and latency issues, with an $N$-bit binary input requiring $2^N$ cycles to complete the computation process, a characteristic drawback of SC implementation. Additionally, energy consumption escalates when the inputs and outputs are in binary form, necessitating an extra interface with other computational components typically found in image processing applications [25].

## B. Counter-Based SC Multiplication

Counter-based SC was basically proposed to mitigate several shortcomings of traditional SC methods by avoiding the random bitstream.

Assuming the precision is $n$-bit for the two provided binary numbers (BNs) $x$ and $w$, which both lie in the range of [0, 1], the traditional SC multiplier using an AND gate (for unipolar encoding) will take $2^n$, equivalent to the length of the

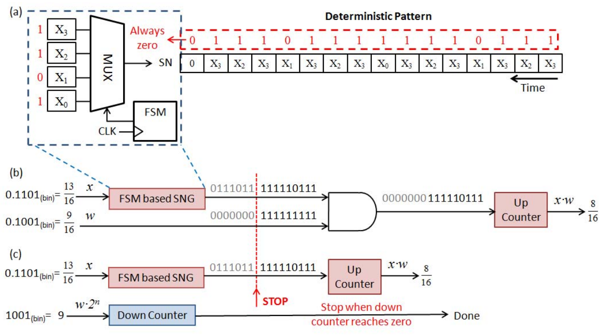


Fig. 3. (a) CBSC-based SNG which generates SN bitstream in a deterministic way. (b) CBSC concept. (c) CBSC multiplication method.

SN bitstream, cycles to complete the task. To enhance this, Sim and Lee [5] presented a CBSC multiplier design with remarkable accuracy. The multiplier encompasses two counters and an FSM-based SNG, as demonstrated in Fig. 3(c). The SNG is utilized to generate an LD SN bitstream of $x$. The authors proposed a deterministic method for this. The technique evenly distributes bit $x_{i-1}$, which is the $i$th bit of $x$, based on its binary weight $2^{i-1}$. For instance, if $i = 3$, then $x_2$ will appear four times in the resultant SN, as depicted in Fig. 3(a). This type of SN generation can be simplified and implemented by an FSM and a MUX as shown in Fig. 3(a). The FSM generates a specific bitstream for the given binary values of the data, which is 1101 in the figure. The output SN bitstream is generated by copying the bit values of the input BN with the order of X3 X2 X3 X1 - X3 X2 X3 X0 - X3 X2 X3 X1 - X3 X2 X3 0, and the FSM will generate the value sequence of 3 2 3 1 - 3 2 3 0 - 3 2 3 1 - 3 2 3.

Since SC multiplication is merely an AND operation, if the SN bitstream for $w$ is arranged to be a sequence of "1"s followed by several "0"s, as illustrated in Fig. 3(b), the multiplication merely requires counting the first $w \cdot 2^n$ bits of the bitstream, corresponding to those bits "1," and does not need to count the other half of the output bitstream, corresponding to those bits "0."

Therefore, the entire counting operation demands only the same latency to complete, significantly reducing computing time. The authors employed a down counter to materialize this concept. Using $w \cdot 2^n$ as the starting value, the down counter decreases by one in each clock cycle. Once it hits "zero," the process concludes. Consequently, this design is more straightforward than a conventional SNG (typically utilizing Linear Feedback Shift Registers) and replaces AND gates with a down counter, which proves to be much more cost-effective than an SNG.

## C. Scaled CBSC-Based Multiplication

One long-standing problem for SC multiplication is that when two numbers are small, the relative error of the multiplication result can be very significant. To mitigate this issue, a scaled CBSC-Multiplier has been proposed recently in [7]. In the scaled-CBSC multiplication method, an $N$-bit BN is represented by a "scaled-binary" 2-tuple $\{M, N\}$. Here, the scaling term has $M$ bits, and the original data is then divided into $2^M$ partitions. To make the binary data larger, we then left shift it by $N/2^M$ bits or one partition at a time and we try to make the scaled number to be larger than 0.5 for both numbers so that the accuracy can be significantly improved compared to no-scaled SC multiplication.

## III. Counting-Based SC Division Design: CBDIV

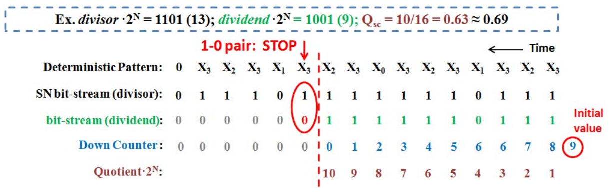
In this section, we try to improve the current SC division method from two main issues. The first issue is that the division result varies when SN bitstreams are generated differently, which is shown in Fig. 2. The second issue is that the current SC division has a large operation latency.

To address these issues, we first present a novel division method named CBDIV, standing for *Counting-Based Division*. The central concept of CBDIV is to generate the SN bitstream in a deterministic pattern to prevent the quotient value from fluctuating across different SN bitstreams. Contrary to CORDIV [18] or ISCBDIV [19], we do not need a synchronizer or an expensive RNG-comparator structure to ensure a strong correlation between two input SN bitstreams. Instead, SN generation in CBDIV adopts a deterministic pattern depicted in Fig. 3(a), which is immune to random fluctuations. Similar to CORDIV [18], we assume the input dividend is smaller than the divisor.

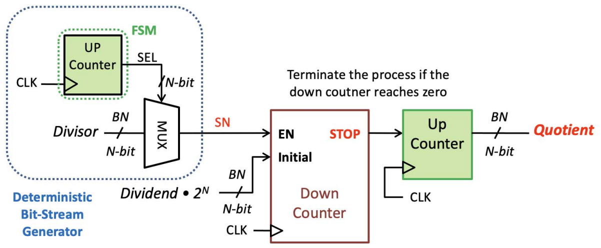
With the SN bitstream generated under a deterministic pattern, the location of bit "1" is fixed. We can efficiently arrange the dividend SN bitstream to promote as many and quick 1–1, 0–0 pairs as possible, as the location of bit "1" in the dividend SN bitstream is entirely determined according to the divisor SN. This implies if a bit in divisor SN is "1," the bit in dividend SN at the corresponding location will also be set to "1" to ensure all the 1–1 pairs are continuously found. Hence, before $S_{\text{Dividend}}$ exhausts all bit "1"s in the SN bitstream, $S_{\text{Dividend}}$ entirely mirrors $S_{\text{Divisor}}$. In simpler terms, the appearance of the first 1–0 pair indicates that the dividend (which is less than the divisor by design) has already exhausted all the bit "1" and no more 1–1 pair remains. This can be depicted in an example shown in the first two rows in Fig. 4(a) Here, the value of the dividend SN $S_{\text{Dividend}}$ and the divisor SN $S_{\text{Divisor}}$ are 6/16 and 9/16, respectively.

As observable, the right parts of $S_{\text{Divisor}}$ and $S_{\text{Dividend}}$ in the blue dash line block in Fig. 4(a) are identical after the arrangement. Then, according to the CORDIV method [11], the quotient SN bitstream $S_{\text{Quotient}}$ simply gains all its bit "1"s in the right half at the third row in Fig. 4(a). A counter that increases by 1 in every clock cycle suffices to procure the binary form value, as shown in the 4th row in Fig. 4(a). It is worth noting that the remaining bits outside the blue frame in $S_{\text{Quotient}}$ are all bit "0." As a result, we do not need to continue counting and can terminate the process, thereby reducing the computation time. The result now counts $10/16 = 0.625$, which is a reasonable approximation to the exact result $9/13{=}0.692$.

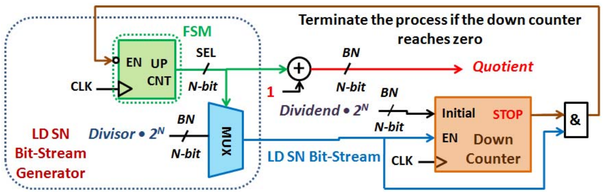
The CBDIV method that we have developed essentially requires three counters to complete the division process: two up counters and one down counter, as depicted in Fig. 4(b). One up counter is utilized as an FSM to generate the divisor SN bitstream. This bitstream follows a similar low discrepancy


Ex. divisor ·2^N = 1101 (13); dividend ·2^N = 1001 (9); Q_sc = 10/16 = 0.63 ≈ 0.69
1-0 pair: STOP
Time
Deterministic Pattern: 0 X3 X2 X3 X1 X3 X2 X3 X0 X3 X2 X3 X1 X3 X2 X3
SN bit-stream (divisor): 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1
bit-stream (dividend): 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1
Initial value
Down Counter: 0 0 0 0 0 0 0 1 2 3 4 5 6 6 7 8 9
Quotient·2^N: 10 9 8 7 6 5 4 3 2 1

(a)



(b)



(c)

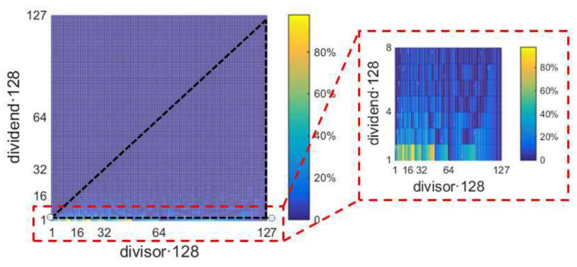
Fig. 4. (a) New CBDIV concept. (b) New CBDIV hardware design. (c) Optimized CBDIV design.

distribution as proposed in [5]. Since the two input bitstreams, $S_{\text{Dividend}}$ and $S_{\text{Divisor}}$, are identical before the division process concludes, we only need to generate one SN bitstream, $S_{\text{Divisor}}$. The other SN bitstream, $S_{\text{Dividend}}$, is copied from $S_{\text{Divisor}}$. The second up counter is employed to convert the quotient SN bitstream, $S_{\text{Quotient}}$, to binary form. It simply increments by one in each clock cycle as the bits in $S_{\text{Quotient}}$ are always "1" before the division process concludes.

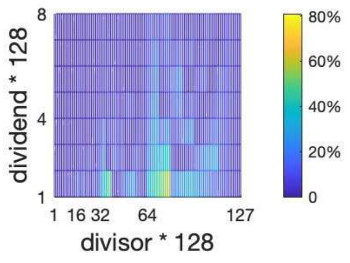
The down counter, with an initial value of $S_{\text{Dividend}} \cdot 2^N$, determines when to end the process. The enable signal "EN" of the down counter is connected to the $S_{\text{Divisor}}$signal, meaning the down counter will decrement by one whenever a bit "1" appears in $S_{\text{Divisor}}$.

Upon observing the fourth and fifth rows in Fig. 4(a), we notice that the FSM output is always one less than the output quotient value before the process ends. This implies that we can deduce the quotient from the FSM output value even without a counter. Therefore, we combine the two up counters. In light of the $S_{\text{Divisor}}$ generation, we retain the up counter used as the FSM. When the division process finishes, we simply increment the output of the FSM by one to obtain the quotient. Now, we only require two counters to form our CBDIV design. The optimized CBDIV design is shown in Fig. 4(c).

Note that in this work we use unipolar encoded SN. The proposed divider can handle negative numbers by using an extra sign bit comparison part and keeping the division operation part unchanged. In future work, we would like to explore the divider architecture for bipolar encoded SN to further improve the robustness and error-tolerance.



(a)



(b)

Fig. 5. (a) Error statistics for the original CBDIV method. (b) Error statistics for FSCDIV on the same zoomed area.

## IV. Further Efficiency Enhancement Design: FSCDIV

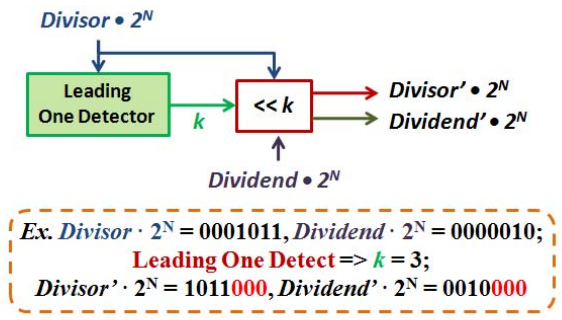
In this section, we present three innovative schemes of the FSCDIV that further improve both the accuracy and the latency of the CBDIV method. The method consists of a new coordinated scaling scheme, a simple half bitstream reduction scheme, and a novel selective counting method.

### A. New Coordinated Scaling for Accuracy Improvement

As we mentioned, due to the natural intrinsic of SN bitstream-based computing, the conventional SC computing has a very large relative error when both input numbers are small. Inspired by the scaling method that improves the accuracy of SC multiplication in [7], we propose to scale up the original small dividend and divisor, ahead of the original SC division.

Different than the scaling process in SC multiplication, which sacrifices latency for higher accuracy, the scaling SC division will not only significantly improve accuracy, but also benefit from latency reduction. This is because the scaling scheme in SC division is different than that in SC multiplication. This will be illustrated by two new schemes introduced in Sections IV-B and IV-C, which leverages the new scaling SC division to further reduce the latency without a negative impact on accuracy.

For SC division, similar to the SC multiplication, when two small numbers are involved, the resultant relative error can be significant. Fig. 5(a) displays the error statistics of state-of-the-art SC-based division, CBDIV method, with 7-bit inputs. Assuming the dividend is always smaller than the divisor, only the down right triangle area surrounded by the black dash line (margin included) is meaningful. To show the area with a large relative error more clearly, we zoom in on the marginal area marked with a red block and show it on the right side in Fig. 5(a). As we can see, CBDIV method suffers a large relative error with a small divisor, especially when the divisor is smaller than 0.5. Such error patterns also exist for existing SC-based divisions. Because when two input values are small, the number of "1" bits among the SN is rare, one bit counting error can therefore cause a larger relative error.



*Ex.* Divisor · $2^N$ = 0001011, Dividend · $2^N$ = 0000010;
Leading One Detect => $k = 3$;
Divisor' · $2^N$ = 1011000, Dividend' · $2^N$ = 0010000

Fig. 6. Scaling block design to promise the divisor to be large enough.

To mitigate this issue, we introduce a scaling scheme to scale the divisor and the dividend with the same ratio to avoid a small divisor, which is illustrated in Fig. 6. We call this *coordinated scaling*. Before introducing the detail of the new scheme, we first show the relative error for FSCDIV on the same zoomed area in Fig. 5(b). As we can see errors have been substantially reduced across those areas.

The coordinated scaling process is carried out before the SC division kernel. To determine whether the divisor is "small" and how many times the divisor is required to be scaled up, a leading one detector (LOD) is introduced to detect the location of the first bit "1" (also called leading bit "1") in the divisor binary bitstream. Then depending on the location of the leading bit "1," we equally left shift both the divisor and the dividend binary bitstream until the divisor's leading bit "1" is shifted to the first bit of the bitstream. It is obvious that after scaling, the divisor is promised to be $\geq 0.5$. Then, the LD bitstream of the divisor will start with a bit "1," which means even with the same counting strategy, the improved CBDIV method now avoids the extra relative error no matter whether the divisor is "small" or "large."

Notice that by scaling the divisor and the dividend with a same ratio $\alpha$, as expressed in

$$Q = \text{dividend}/\text{divisor} = (\text{dividend} \cdot \alpha)/(\text{divisor} \cdot \alpha) \quad (2)$$

the scaling related work is only an extra step conducted ahead of SC division kernel. Such scaling has no influence on the division quotient $Q$, which means no extra "scaling factor" is needed, making the SC divisor easy to interface with other binary computations. This is fundamentally different from the scaling SC multiplication, in which the scaling scheme complexes the SN number to a two-tuple of *scalingfactor*, *SN*. The extra scaling factor in scaling multiplication has to be stored and computed both before and after the SN multiplication kernel.

We note that the proposed scaling method is different from the scaling scheme proposed for the multiplication in [7] as we do not have a special scaling item first. Specifically, the scaled SC multiplication will scale up the input before the multiplication and then scale back/down the result. So there are two scaling operations in the scaled SC multiplication, and the data are stored in "scaled-binary" 2-tuple $\{M, N\}$, where $M$ is the scaling term. In contrast, the proposed scaled SC division scales both dividend and divisor at the same time (so called coordinated scaling) before the division operation. As the scaling ratio is cancelled in the division operation, there is no need for a second scaling to the division output.

### B. New Half Bitstream Reduction Counting Strategy

The second idea is based on the observation that the SN bitstream of the scaled divisor and dividend have fixed patterns of bit "1" distribution, which can be leveraged by only counting half the length of the bitstream instead of the entire bitstream when computing the quotient.

Specifically, as mentioned in Section III, the quotient counting value equals to $Q \cdot 2^N$ for $N$-bit inputs. This value can be also expressed as

$$Q \cdot 2^N = 2^N - (1 - Q) \cdot 2^N. \tag{3}$$

By observing Fig. 4(a), as the value $Q \cdot 2^N$ actually equals to the number of cycles in the right half of the bitstreams; the value $(1 - Q) \cdot 2^N$ equals to the number of cycles in the left half. So calculating the value $2^N - (1-Q) \cdot 2^N$ is equivalent to doing down counting of the left half bitstreams with an initial value of $2^N$.

Since the dividend SN stream is generated by copying the right half of the divisor SN stream, the divisor and dividend SN streams are completely the same in the right half. As proved in Section III, all the 1–1 pairs will appear in the right half, meaning that all the 1–0 pairs will occur in the left half of the bit-steams. The termination conditions of the CBDIV method can be written as: 1) start counting from the right side; 2) all the 1–1 pairs have been used up; and 3) meet a 1–0 pair when doing counting. The first 1–0 pair [when counting from the right side, marked with a red circle in Fig. 4(a)] can be treated as the boundary of two halves of the bitstreams. Alternatively, we count from the left side and finish the quotient counting when using up all the 1–0 pairs in the left half bitstreams. The termination condition of the CBDIV method now can be simplified to: doing down counting with an initial value of $2^N$ until the last 1–0 pair is counted.

We show the new counting scheme of the CBSC division in Fig. 7. Note that the FSCDIV is counting 1–0 pairs in a left-to-right order, thus the time flow direction in Fig. 7 is opposite to the previous conventional right-to-left 1–1 pair counting methods. Since the new counting scheme actually just counts the bitstream from an opposite direction. The average latency keeps the same ($2^{N-1}$), which can be expressed as

$$\begin{aligned} \text{Lat}_{\text{Orig}} &= Q \cdot 2^N \\ \text{Lat}_{\text{New}} &= (1 - Q) \cdot 2^N \end{aligned} \tag{4}$$

where $\text{Lat}_{\text{Orig}}$ and $\text{Lat}_{\text{New}}$ represent the latency of the original CBDIV method and the new counting scheme, respectively. From (4), it is easy to derive that when $Q \geq 0.5$, $\text{Lat}_{\text{New}}$ will be smaller than the average, outperforming $\text{Lat}_{\text{Orig}}$ (shown in case (i) in Fig. 7); while when $Q < 0.5$, $\text{Lat}_{\text{New}}$ will be worse

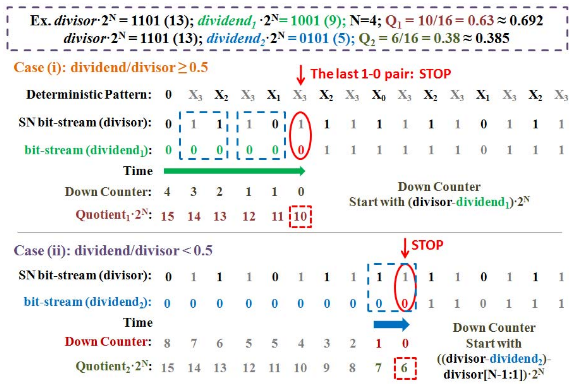

Ex. divisor·2^N = 1101 (13); dividend_1·2^N= 1001 (9); N=4; Q_1 = 10/16 = 0.63 ≈ 0.692
divisor·2^N = 1101 (13); dividend_2·2^N = 0101 (5); Q_2 = 6/16 = 0.38 ≈ 0.385
Case (i): dividend/divisor ≥ 0.5
The last 1-0 pair: STOP
Deterministic Pattern:
SN bit-stream (divisor):
bit-stream (dividend_1):
Time
Down Counter:
Quotient_1·2^N:
Down Counter Start with (divisor-dividend_1)·2^N
Case (ii): dividend/divisor < 0.5
STOP
SN bit-stream (divisor):
bit-stream (dividend_2):
Time
Down Counter:
Quotient_2·2^N:
Down Counter Start with ((divisor-dividend_2)-divisor[N-1:1])·2^N

Fig. 7. Counting strategy for the fast and scaled CBSC division.

than $\text{Lat}_{\text{Orig}}$, shown in case (ii) in Fig. 7. Since when $Q < 0.5$, the first $2^{N-1}$ cycles which marked with gray color in case (ii) in Fig. 7 are always promised to be counted. To accelerate the counting, we can directly start from the middle point, which will save $2^{N-1}$ cycles. Now, the latency of the new counting strategy can be expressed as

$$\text{Lat}'_{\text{New}} = \begin{cases} (1 - Q) \cdot 2^N (Q \geq 0.5) \\ (0.5 - Q) \cdot 2^N (Q < 0.5) \end{cases} \tag{5}$$

which will further be reduced to half of $\text{Lat}_{\text{Orig}}$.

To quickly determine whether the $Q$ value is greater or less than 0.5, we compute the difference between the divisor and dividend via a binary subtraction, which is cheap. If the first (most significant) bit of the difference is "1" then its equivalent to the $Q$ will be greater than 0.5. In this way, the hardware cost for such a comparison should be marginal.

### C. New Selective Counting Strategy

It turns out that the counting process can be further improved. As mentioned in Section IV-A, by introducing the scaling scheme, the divisor now is promised to be $\geq$ 0.5. According to the fixed bitstream pattern in the counter-based SC design, the most significant binary bit $x_{N-1}$ ($N$-bit data) will appear once every two cycles, so the 1–0 pair will occur once every two cycles as shown in Fig. 7. As a result, we do not need to count the $x_{N-1}$ bit in the bitstream. What we need is just to count the other bit ($x_i$, $i \neq N-1$) in the adjacent two cycles (marked with a blue square frame in each case in Fig. 7). The reduced value for the down counting (this is a special counter) in each cycle now is based on the value of a 2-bit data $d$ (d[1:0])

$$d = x_{N-1} + x_i (i \neq N-1) = 1'b1 + x_i (i \neq N-1). \tag{6}$$

Also, since $x_{N-1}$ is promised to be "1," it is no longer needed to be generated in the divisor bitstream (marked with gray color in Fig. 7). The FSM now only requires the remaining $N$-$1$-bit ($x$[N-2:0]) data as the input, which will be demonstrated in Fig. 8. We note that the stop condition for the new counting process also requires to be adjusted. Since the reduced value $d$ can either be one or two in each cycle now, after the last counting cycle, the remaining number of 1–0 pairs can either be one or zero. We show examples for each of these two possibilities in Fig. 7. When the remaining number

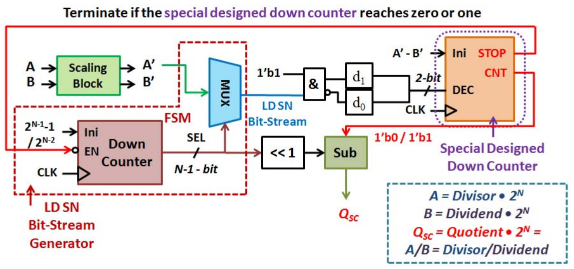


Fig. 8. Fast and scaled CBSC division design.

of 1–0 pairs is zero, shown in case (ii), we do not change the quotient value and simply terminate the process. While when the remaining number of 1–0 pairs is one [shown in case (i)] as $x_{N-1}$ is promised to be "1" and no longer generated due to the aforementioned selective counting strategy, the 1–0 pair in the red circle in case (i) will not be counted. In these cases, we terminate the process when the down counting value equals to one, and further subtract one from the quotient value for the compensation.

### D. Hardware Design for the Proposed FSCDIV

We show the architecture of the fast and scaled CBSC division design in Fig. 8. By comparing Fig. 8 with Fig. 4(b), the down counter used for counting the remaining 1–1 pairs in the original CBDIV design is now replaced with a specially designed down counter to count the remaining 1–0 pairs with an initial value of divisor $\cdot\, 2^N$ − dividend $\cdot\, 2^N$, since we can easily prove that

$$\begin{aligned} N_{11} &= \text{dividend} \cdot 2^N \\ N_{10} &= \text{divisor} \cdot 2^N - N_{11} \\ &= \text{divisor} \cdot 2^N - \text{dividend} \cdot 2^N \\ &= A - B \end{aligned} \tag{7}$$

where $N_{11}/N_{10}$ is the total number of the 1–1/1–0 pairs in the bitstreams. For convenience, we set $A$ = divisor $\cdot\, 2^N$, $B$ = dividend $\cdot\, 2^N$, $A'$ and $B'$ are the scaled inputs of the division process. The specially designed down counter decreases one or two depending on the input data of the "DEC" port, which is represented by $d_1 d_0$. With the scaling block, $A'$ (scaled divisor) is promised to be $\geq 0.5 \cdot 2^N$. So the MSB (most significant bit) of $A'$ (represented by $a'_{N-1}$) is promised to be "1," which is no need to generate with the LD bitstream generator (SNG) as mentioned before in Section IV-C. The input of the SNG then is set to be *N-1*-bit. According to the simplified counting scheme illustrated in Section IV-B, the select signal ["SEL" in Fig. 4(b)] for the MUX, which is the output of the FSM block (a down counter decrease one in each cycle with an initial value of $2^{N-1} - 1/2^{N-2}$, depending on whether $Q$ is larger or smaller than 0.5). Since there are only two possible cases for the binary form of $d_1 d_0$: 2'b10/2'b01, we use an AND gate and a NAND gate with the inputs of $a'_{N-1}$ and the LD SN bit to obtain $d_1$ and $d_0$, respectively. And based on the improved counting scheme illustrated in Section IV-C, the specially designed down counter will stop counting until the counting value equals to one or zero. To save the resource, similar to the original CBDIV design, we share the down counter which is the FSM block in the SNG to obtain the quotient value $Q_{SC}$. As the output value of the FSM block actually is only half of the quotient value, we do 1-bit left shifting operation. Also as mentioned in Section IV-C, when we do selective counting, the remaining 1–0 pairs can either be zero or one. Thus, we need further do a subtraction operation of the shifted FSM output value by the counting value ("CNT" port) of the specially designed down counter after the termination of the SC division counting process to obtain the quotient value.

## V. Experimental Results and Discussion

In this section, we evaluate the proposed Counting-based SC Divider, CBDIV as well as the Scaled Counting-based SC Divider, FSCDIV in three aspects. We also compare CBDIV against the binary logic baseline and state-of-the-art works.

### A. Experimental Setup

In order to asses the performance of the proposed CBDIV design, our work compared its error behavior and hardware performance to a fixed-point divider that uses a long division algorithm. The hardware performance of CBDIV was evaluated by measuring area, critical path, average latency (in clock cycles), average delay, the ADP, power, and total energy consumption. Additionally, CBDIV was compared to several state-of-the-art works, such as the DFSMDIV (*SC division with deterministic FSMs*) [24], CORDIV [18], JKSSDIV [20], and ISCBDIV [19], all utilizing BN inputs with 7-bit precision (corresponding to a 128-bit length bitstream).

The next step we evaluate the further improved proposed FSCDIV design. We show the improvements of the error behavior and the hardware performance of FSCDIV compared to the CBDIV. We also compare with a fixed-point divider using long division algorithm.

The dividers were implemented in Verilog and synthesized with Synopsys Design Compiler using EDK 32-nm standard cell library [26]. To ensure a reasonable comparison, all dividers have binary inputs and outputs. For the in-stream SC-based division designs, SNGs were added to generate SN bitstreams, and a counter was included to count the value of the resultant SN bitstream. The hardware evaluation is demonstrated in Table I. To ensure a justified power and energy comparison, all dividers share a fixed 100-MHz frequency input clock. Energy consumption was evaluated according to the overall execution time and the power consumed by the divider during the entire division process.

We use RMSE to evaluate the error behavior. The behavioral emulation result for all the SC-based dividers is listed in Table I. In MATLAB, we use the uniform random generator to obtain the dividend in the range of [0, 0.5], and the divisor in the range of [0.5, 1] for over 100 000 iterations and compute the RMSE.

### B. Hardware Performance

*1) Hardware Performance Analysis for CBDIV:* Assessing the second column from Table I, it is noticeable that our

TABLE I
HARDWARE PERFORMANCE FOR SC AND FIXED-POINT DIVIDERS

| **Dividers** | **Area** ($\mu m^2$) | **Critical Path (ns)** | **Avg. Latency (cycles)** | **Avg. Delay (ns)** | **ADP** | **Power** ($\mu$**W**) | **Avg. Energy (pJ)** |
|---|---|---|---|---|---|---|---|
| Fixed-Point | 1090 | 3.16 | 17 | 53.72 | 58555 | 41.4 | 7.038 |
| DFSMDIV [24] | 520 | 1.24 | 128 | 158.72 | 82534 | 22.3 | 28.544 |
| CORDIV [18] | 381 | 0.94 | 128 | 120.32 | 45842 | 14.7 | 18.765 |
| JKSSDIV [20] | 285 | 1.22 | 128 | 156.16 | 44506 | 11.8 | 15.040 |
| ISCBDIV [19] | 394 | 0.98 | 128 | 125.44 | 49423 | 15.8 | 20.211 |
| **CBDIV (proposed)** | **299** | **1.72** | **64** | **110.08** | **32914** | **10.9** | **6.976** |
| **FSCDIV (proposed)** | **576** | **3.00** | **16** | **48.00** | **27648** | **24.0** | **3.837** |

TABLE II
HARDWARE PERFORMANCE OF FSCDIV AND FIXED-POINT DIVIDERS OVER DIFFERENT BITS

| **Dividers** | **Area** ($\mu m^2$) | **Critical Path (ns)** | **Avg. Latency (cycles)** | **Avg. Delay (ns)** | **ADP** | **Power** ($\mu$**W**) | **Avg. Energy (pJ)** |
|---|---|---|---|---|---|---|---|
| Fixed-Point 6bits | 990 | 3.14 | 14 | 43.96 | 43520 | 34.77 | 4.87 |
| Fixed-Point 7bits | 1090 | 3.16 | 17 | 53.72 | 58555 | 41.40 | 7.038 |
| Fixed-Point 8bits | 1249 | 3.66 | 19 | 69.54 | 86855 | 46.73 | 8.87 |
| **FSCDIV (proposed)** 6bits | **465** | **2.88** | **8** | **23.04** | **10713** | **17.03** | **1.36** |
| **FSCDIV (proposed)** 7bits | **576** | **3.00** | **16** | **48.00** | **27648** | **24.0** | **3.837** |
| **FSCDIV (proposed)** 8bits | **683** | **3.30** | **32** | **105.6** | **72124** | **25.42** | **8.134** |

introduced stochastic divider, CBDIV [27], excels in area efficiency among all present stochastic dividers. It achieves a reduction in area by 72.6% when juxtaposed with the fixed-point divider (binary logic) baseline and surpasses the performance of current leading stochastic dividers by a minimum of 21.5%. Since the CBDIV design's latency is influenced by the dividend value, as illustrated in Section III, it exhibits variable latency based on input data. This is a departure from the fixed latency observed in prior SC division designs at a given precision level. Therefore, in Table I, we utilize *Avg.Latency* and *Avg.Delay* to quantify the average computation time for CBDIV, and these metrics are compared against preceding work. Column 4 shows that CBDIV reduces average latency by 65.6%, and column 5 reveals that the *Avg.Delay* of CBDIV stands at 75.68 ns, exceeding the performance of other contemporary works by at least 37.1%. While the latency of CBDIV is 40.9% greater than that of the fixed-point divider, the ADP is significantly lower (61.4% less), and shows a minimum improvement of 50.6% compared to past studies, as demonstrated in column 6.

Reviewing the final two columns of Table I, it is apparent that CBDIV surpasses all other listed division designs in terms of power and energy consumption. When it comes to power consumption, CBDIV is superior to current leading designs by 25.9%.

For the energy consumption aspect, we note that in-stream SC-based division designs will cost even more energy than the fixed-point divider when SNGs and the counter are taken into consideration. CBDIV design has solved this problem by improving the computing latency, achieving a 31.9% energy reduction compared to the fixed-point divider baseline.

In Table II, we evaluate the hardware performance of the proposed FSCDIV over fixed-point divider on various bit-widths. As we can see that FSCDIV is superior to the fixed-point divider in area, delay, ADP, power, and energy through 6–8 bit, at a very small cost of accuracy. But we also observe that these advantage or benefits will become smaller as the bit-width increases, and will eventually vanish at a further higher bit-width, which is typical for SC.

TABLE III
DIFFERENT OPTIMIZATION EFFECTS ON LATENCY

| Applied Strategies | Avg.Latency(cycles) |
|---|---|
| None (CBDIV) (baseline) | 63.8 |
| Coordinated scaling scheme | 49.8 |
| Coordinated scaling scheme; Selective counting | 32.17 |
| Coordinated scaling scheme; Half bitstream reduction | 25.35 |
| All (FSCDIV) | 16.14 |

*2) Hardware Performance Analysis for FSCDIV:* As the further improved FSCDIV design ameliorates the latency and the error behavior from CBDIV approach by adding and replacing some components, it will inevitably introduce hardware resource overheads. However, by observing the fourth and fifth columns in Table I, we notice that FSCDIV can significantly reduce 75.0% latency and 56.4% delay when compared with the CBDIV. The most important point is that for the ADP aspect, FSCDIV outperforms the CBDIV baseline by 16.0%, which means the proposed FSCDIV design performs much better on the overall hardware performance evaluation. The FSCDIV even shows a smaller computing latency and delay when compared to the fixed-point divider, and improves the area and the ADP by 47.2% and 52.8%, respectively.

From the last two columns in Table I, we observe that FSCDIV though does not perform the best in power consumption among all the listed dividers, it has the lowest total energy consumption among all the listed dividers in Table I, improving 45.0% energy consumption when compared to the CBDIV baseline. Furthermore, we separately evaluate the impact of the three proposed FSCDIV optimization strategies on latency in Table III. The baseline will be the original CBDIV without optimizations. As we add the three optimization strategies one by one, we can see the average latency is mitigated from 63.8 cycles to 16.14 cycles.

## *C. Error Behavior Analysis and Comparison*

Figs. 10 and 11(a) and (b) show the accuracy comparison among the proposed CBDIV and the further improved

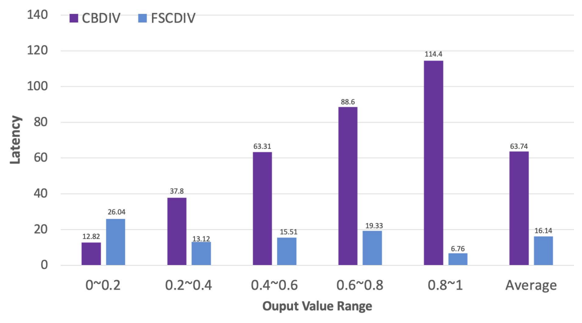


Fig. 9. Latency of CBDIV and FSCDIV regarding to quotient values.

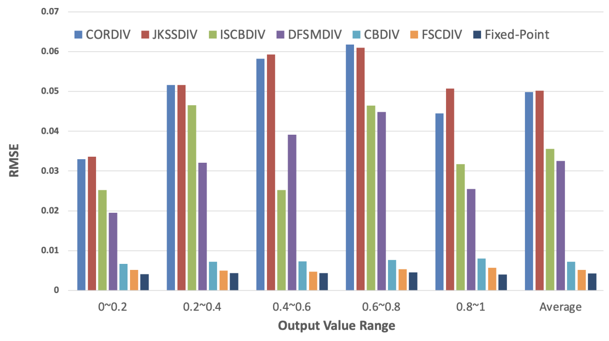


Fig. 10. Error behavior of CBDIV, FSCDIV, and other dividers with different output value ranges (7-bit precision).

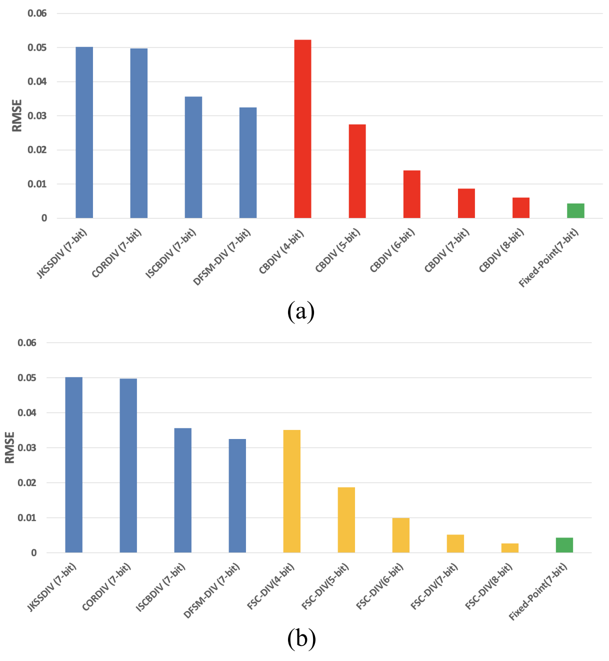


Fig. 11. Error behavior of the proposed SC dividers compare with other SOTA SC dividers as well as the fixed-point divider under 7-bit. (a) Error behavior of the proposed CBDIV under 4–8 bit. (b) Error behavior of the proposed FSCDIV under 4–8 bit.

FSCDIV methods with the baseline fixed-point divider and existing approaches: CORDIV [18], JKSSDIV [20], ISCBDIV [19], and DFSMDIV [24]. Linear feedback shift register (LFSR) is utilized to generate SN bitstreams for binary inputs in state-of-the-art approaches for comparison. The floating-point output quotient is used as the "golden" result. We show the results over five nonoverlapped output value ranges as well as the average RMSE value obtained from 100 000 computations for all the algorithms, all possible input operand values are equally taken into account.

*1) Error Behavior of CBDIV:* An examination of Fig. 10 reveals CBDIV surpasses other contemporary works across all five nonoverlapping output value domains. When contrasted with DFSM-DIV, which possesses the smallest average RMSE value, CBDIV demonstrates an improvement of 77.8%. To add further context, it is worth mentioning that in Fig. 10, CBDIV exhibits comparable RMSE values across varied output value domains, a characteristic that sets it apart from the results achieved by existing work. This discrepancy can be attributed to the inherent attributes of distinct SC division methodologies. Moreover, Fig. 11(a)provides a comparison of CBDIV's error behavior against the fixed-point baseline and other leading works, particularly with respect to varying bitstream lengths (precision). From Fig. 11(a), it can be observed that even at a 5-bit precision, CBDIV's average RMSE value outperforms DFSMDIV's 7-bit precision by 15.4%.

*2) Error Behavior of FSCDIV:* From Fig. 11(b), the further improved FSCDIV method ameliorating 61.2% accuracy when compared to the CBDIV. The FSCDIV method even demonstrates a very small gap between the fixed-point design on average RMSE.

## D. Comparison in Digital Image Processing Application

We implement the proposed CBDIV and further enhanced FSCDIV method for a digital image process application, *Contrast Stretch* to compare the performance against the baseline design and state of art works in a real application scenario.

The input binary precision of the proposed FSCDIV, CBDIV, as well as state-of-the-art ISCBDIV [19] and DFSMDIV [24] division methods are all 8-bit, as the pixels of the JPEG format are in 8-bit binary precision. Thus, the SC bitstream length in this application will be 256-bit.

The contrast-stretched pixel $G(x, y)$ of an arbitrary input image pixel $I(x, y)$ can be calculated as

$$G(x, y) = \frac{I(x, y) - I_{\min}}{I_{\max} - I_{\min}} \cdot 255. \tag{8}$$

$I_{\max}$ and $I_{\min}$ represent the largest and smallest pixel values in image $I(x, y)$. It is important to notice that we initially carry out the division of $([I(x, y) - I_{\min}]/[I_{\max} - I_{\min}])$, as SC dividers invariably necessitate the dividend to be smaller than the divisor. Additionally, we employ the CBSC method for multiplication as detailed in Section II-B. The number 255 is conceptualized as $255/256 \cdot 2^8$, leading to a $w$ value of 255/256 in the CBSC multiplier and the operation will require 255 clock cycles for completion. Owing to the deterministic pattern utilized in the CBSC method, all bit "1"s in the divisor $([I(x, y) - I_{\min}]/[I_{\max} - I_{\min}])$ bitstream materialize within these 255 clock cycles, which implies we only need to tally the number of bit "1"s in the divisor bitstream. For CBDIV and FSCDIV, the counting procedure concludes synchronously with the termination of the corresponding divider.

TABLE IV
PSNR FOR CONTRAST STRETCH APPLICATION USING SC DIVIDERS

| | PSNR (dB) | | | | |
|---|---|---|---|---|---|
| | ISCBDIV | DFSMDIV | **CBDIV (proposed)** | **FSCDIV (proposed)** | Fixed-Point |
| Lena | 26.39 | 24.18 | **41.78** | **46.46** | 59.08 |
| Portrait | 24.85 | 27.33 | **41.73** | **42.81** | 58.56 |
| Woman | 27.64 | 25.79 | **37.22** | **43.85** | 58.80 |
| Map | 26.84 | 36.35 | **49.67** | **49.81** | 58.73 |

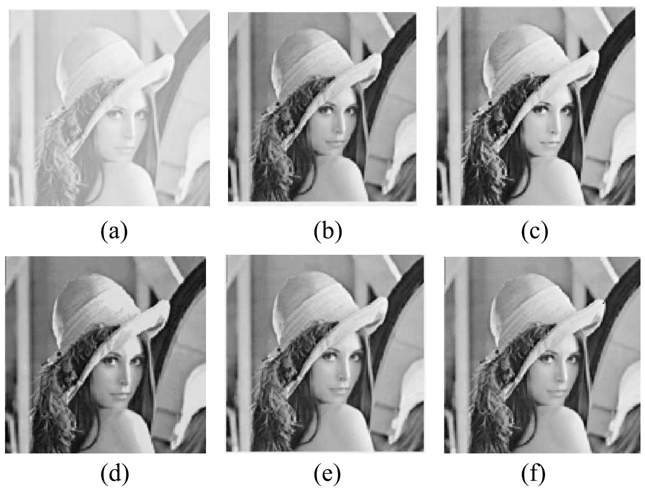


Fig. 12. Example of the original image and contrast stretch results operated by various dividers. (a) Original image. (b) Fixed-point divider. (c) ISCBDIV. (d) DFSMDIV. (e) CBDIV. (f) FSCDIV.

No further action is mandated and we directly output the CBDIV or FSCDIV result as the $G(x, y)$ value. However, as ISCBDIV and DFSMDIV produce SN bitstreams, we simply append a counter to procure $G(x, y)$.

We apply the contrast stretch operation to a number of input images and use peak signal-to-noise ratio (PSNR) defined in (9) to assess the output image quality. Image processing results subjected to the contrast stretching with a floating-point divider are used as the baseline. Table IV presents the PSNR values for all tested images using different SC dividers. It is observable that the proposed CBDIV method surpasses the two leading SC dividers in all tested images, and it enhances the image quality by an average of 15.18 dB

$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{I_{\max}^2}{\mathrm{MSE}\big(G_{\text{baseline}}, G_{\text{target}}\big)}. \tag{9}$$

The further enhanced FSCDIV ameliorates the output image quality with extra 3.13 dB from the proposed CBDIV method.

## VI. Conclusion

In this article, we proposed two novel SC divider designs, CBDIV and FSCDIV, improving SC division in both speed and accuracy. CBDIV generated SNs in a deterministic pattern, resulting in higher accuracy and utilizing CBSC's efficiency to perform division. FSCDIV further improved CBDIV by reducing the overall latency to one-fourth and improved accuracy through a new scaling method and selective counting strategy. Our experimental results demonstrated that the proposed CBDIV, implemented in 32-nm technology, outperformed state-of-the-art works by 77.8% in accuracy, 37.1% in delay, 21.5% in area, 50.6% in ADP, and 25.9% in power. Additionally, CBDIV reduced energy consumption by 31.9% when compared to the fixed-point division baseline and was significantly more energy-efficient than existing SC-based dividers for binary inputs and outputs required in efficient image processing implementations. (Note: this information may require updating based on more recent technology nodes or advances in the field.) Moreover, we demonstrated that FSCDIV could improve delay by 56.4%, ADP by 16.0%, energy consumption by 45.0%, and accuracy by 61.2%. We also evaluated the proposed CBDIV and FSCDIV designs in a contrast stretch image processing workload and showed that it improved image quality by 18.3 dB on average when compared to state-of-the-art works.

## References

[1] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5617–5626, Nov. 2011.

[2] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 449–462, Mar. 2014.

[3] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," in *Proc. 53nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2016, pp. 1–6.

[4] H. Sim, D. Nguyen, J. Lee, and K. Choi, "Scalable stochastic-computing accelerator for convolutional neural networks," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2017, pp. 696–701.

[5] H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[6] R. Hojabr et al., "SkippyNN: An embedded stochastic-computing accelerator for convolutional neural networks," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, p. 132.

[7] S. Yu and S. X.-D. Tan, "Scaled-CBSC: Scaled counting-based stochastic computing scheme for improved accuracy," in *Proc. Design Autom. Conf. (DAC)*, 2022, pp. 1–6.

[8] Z. Chen, Y. Ma, and Z. Wang, "Optimizing stochastic computing for low latency inference of convolutional neural networks," in *Proc. 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–7.

[9] Y. Liu, S. Yu, S. Peng, and S. X.-D. Tan, "Runtime long-term reliability management using stochastic computing in deep neural networks," in *Proc. 22nd Int. Symp. Qual. Electron. Design (ISQED)*, 2021, pp. 553–558.

[10] S. Latif et al., "IoT technology enabled stochastic computing paradigm for numerical simulation of heterogeneous mosquito model," *Multimedia Tools Appl.*, vol. 82, no. 1, pp. 18851–18866, 2023.

[11] P. Schober, S. N. Estiri, S. Aygun, A. H. Jalilvand, M. H. Najafi, and N. TaheriNejad, "Stochastic computing design and implementation of a sound source localization system," *IEEE J. Emerg. Select. Topics Circuits Syst.*, vol. 13, no. 1, pp. 295–311, Mar. 2023.

[12] X. Jia et al., "An energy-efficient bayesian neural network implementation using stochastic computing method," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 3, 2023, doi: 10.1109/TNNLS.2023.3265533.

[13] S. Liu and J. Han, "Energy efficient stochastic computing with sobol sequences," in *Proc. Design, Autom. Test Europe Conf. Exhib.*, 2017, pp. 650–653.

[14] S. Liu and J. Han, "Toward energy-efficient stochastic circuits using parallel sobol sequences," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 7, pp. 1326–1339, Jul. 2018.

[15] F. Neugebauer, I. Polian, and J. P. Hayes, "Building a better random number generator for stochastic computing," in *Proc. Euromicro Conf. Digit. System Design (DSD)*, 2017, pp. 1–8.

[16] K. Kim, J. Lee, and K. Choi, "An energy-efficient random number generator for stochastic circuits," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2016, pp. 256–261.

[17] D. Wu, R. Yin, and J. S. Miguel, “Normalized stability: A cross-level design metric for early termination in stochastic computing,” in *Proc. 26th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2021, pp. 254–259.
[18] T.-H. Chen and J. P. Hayes, “Design of division circuits for stochastic computing,” in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2016, pp. 116–121.
[19] D. Wu and J. S. Miguel, “In-stream stochastic division and square root via correlation,” in *Proc. 56th Annu. Design Autom. Conf.*, 2019, p. 162.
[20] S.-I. Chu, “New divider design for stochastic computing,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 1, pp. 147–151, Jan. 2020.
[21] S. Asadi, M. H. Najafi, and M. Imani, “CORLD: In-stream correlation manipulation for low-discrepancy stochastic computing,” in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2021, pp. 1–9.
[22] W. Shaowei, X. Guangjun, H. Jie, and Z. Yongqiang, “Highly accurate division and square root circuits by exploiting signal correlation in stochastic computing,” *Int. J. Circuit Theory Appl.*, vol. 50, pp. 1375–1385, Apr. 2022. [Online]. Available: https://doi.org/10.1002/cta.3219
[23] B. R. Gaines, “Stochastic computing systems,” in *Advances in Information Systems Science*. Boston, MA, USA: Springer, 1969, pp. 37–172.
[24] N. Temenos and P. P. Sotiriadis, “Deterministic finite state machines for stochastic division in unipolar format,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–5.
[25] S.-I. Chu, Y.-M. Lee, C.-E. Hsieh, J.-H. Yen, and Y.-J. Huang, “Stochastic circuit design of image contrast stretching,” in *Proc. Int. SoC Design Conf. (ISOCC)*, 2019, pp. 81–82.
[26] R. Goldman, K. Bartleson, T. Wood, K. Kranen, V. Melikyan, and E. Babayan, “32/28nm educational design kit: Capabilities, deployment and future,” in *Proc. IEEE Asia Pacific Conf. Postgrad. Res. Microelectron. Electron. (PrimeAsia)*, 2013, pp. 284–288.
[27] S. Yu, Y. Liu, and S. X.-D. Tan, “Approximate divider design based on counting-based stochastic computing division,” in *Proc. ACM/IEEE 3rd Workshop Mach. Learn. CAD (MLCAD)*, 2021, pp. 1–6.

**Yibo Liu** (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2017, and the M.S. degree in computer engineering in 2019. He is currently pursuing the Ph.D. degree with the University of California at Riverside, Riverside, CA, USA.

**Shuyuan Yu** (Member, IEEE) received the B.S. and M.S. degrees in communication engineering from Fudan University, Shanghai, China, in 2015 and 2018, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA, USA, in 2023.

**Maliha Tasnim** (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronics engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2016. She is currently pursuing the Ph.D. degree in electrical engineering with the University of California at Riverside, Riverside, CA, USA.

Her current research includes efficient design of computing circuits for fast and parallel processing of data intensive application.

**Sheldon X.-D. Tan** (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1992 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Iowa, Iowa City, IA, USA, in 1999.

He is a Professor with the Department of Electrical Engineering, University of California at Riverside, Riverside, CA, USA, where he is also a Cooperative Faculty Member with the Department of Computer Science and Engineering. He was a Visiting Professor with Kyoto University, Kyoto, Japan, as a JSPS Fellow from December 2017 to January 2018. His research interests include machine and deep learning for VLSI reliability modeling and optimization at circuit and system levels, machine learning for circuit and thermal simulation, thermal modeling, optimization and dynamic thermal management for many-core processors, efficient hardware for machine learning and AI, and parallel computing and simulation based on GPU and multicore systems. He has published more than 330 technical papers and has coauthored six books on those areas.

Dr. Tan received the NSF CAREER Award in 2004. He received the Best Paper Awards from ICSICT’18, ASICON’17, ICCD’07, and DAC’09. He also received the Honorable Mention Best Paper Award from SMACD’18. He is serving as the TPC Chair for ASPDAC 2021 and the TPC Vice Chair for ASPDAC 2020. He is serving or served as the Editor-in-Chief for *Integration, the VLSI Journal* (Elsevier) and an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, *Microelectronics Reliability* (Elsevier), and *Electronics, Microelectronics and Optoelectronics Section* (MDPI).