Zero Correlation Error: A Metric for Finite-Length Bitstream Independence in Stochastic Computing

Hsuan Hsiao
University of Toronto
julie.hsiao@mail.utoronto.ca

Joshua San Miguel
University of Wisconsin-Madison
jsanmiguel@wisc.edu

Yuko Hara-Azumi
Tokyo Institute of Technology
hara@cad.ict.e.titech.ac.jp

Jason Anderson
University of Toronto
janders@ece.utoronto.ca

ABSTRACT

Stochastic computing (SC), with its probabilistic data representation format, has sparked renewed interest due to its ability to use very simple circuits to implement complex operations. Though unlike traditional binary computing, SC needs to carefully handle correlations that exist across data values to avoid the risk of unacceptably inaccurate results. With many SC circuits designed to operate under the assumption that input values are independent, it is important to provide the ability to accurately measure and characterize independence of SC bitstreams. We propose *zero correlation error (ZCE)*, a metric that quantifies how independent two finite-length bitstreams are, and show that it addresses fundamental limitations in metrics currently used by the SC community. Through evaluation at both the functional unit level and application level, we demonstrate how ZCE can be an effective tool for analyzing SC bitstreams, simulating circuits and design space exploration.

1 INTRODUCTION

Stochastic computing (SC) is a reemerging computing paradigm—with applications in image processing [2, 7, 12], error correction codes [6, 17] and neural networks [3, 15, 16]—that performs computation on bit-serial unary bitstreams as opposed to bit-parallel binary-encoded registers [8, 18]. Values in SC are represented by the probability that a bit is set in a bitstream. Because of its value encoding format and its serial nature, SC is capable of performing computation with extremely small functional units (e.g. multiplication is reduced to a single AND gate).

Unlike traditional binary computing, SC needs to handle data values with care to avoid the risk of unacceptably inaccurate computation results. Because SC is probabilistic, data values (bitstreams) need to be statistically independent for key operations (e.g. multiply [9] and scaled add [9]). SC circuit designers need a way to accurately measure and characterize independence to ensure a correctly functioning system. Applying the definition of independence from classical probability theory to SC bitstreams is non-trivial since in practical implementations of SC, bitstreams are not truly random and are finite in length. Currently, the metric used for this purpose is stochastic cross correlation (SCC) [1].

ASPDAC '21, January 18–21, 2021, Tokyo, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431552

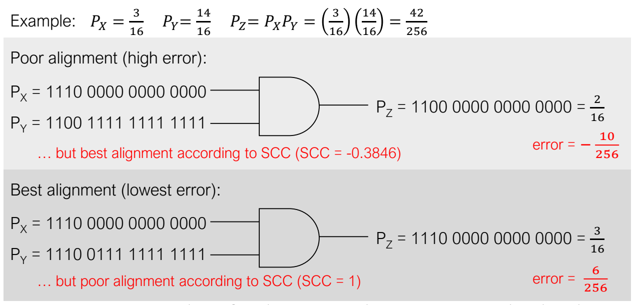

Example: $P_X = \frac{3}{16}$ $P_Y = \frac{14}{16}$ $P_Z = P_X P_Y = \left(\frac{3}{16}\right)\left(\frac{14}{16}\right) = \frac{42}{256}$
Poor alignment (high error):
P_X = 1110 0000 0000 0000
P_Y = 1100 1111 1111 1111
P_Z = 1100 0000 0000 0000 = $\frac{2}{16}$
error = $-\frac{10}{256}$
… but best alignment according to SCC (SCC = -0.3846)
Best alignment (lowest error):
P_X = 1110 0000 0000 0000
P_Y = 1110 0111 1111 1111
P_Z = 1110 0000 0000 0000 = $\frac{3}{16}$
error = $\frac{6}{256}$
… but poor alignment according to SCC (SCC = 1)

Figure 1: Example of when an alignment with the lowest $|SCC|$ does not correspond to the one with highest accuracy.

SCC reasons about the similarity between two finite-length bitstreams by looking at the alignment of 0s and 1s in the two bitstreams. For a functional unit that prefers independent inputs for higher accuracy, the highest accuracy is attained when the input bitstreams are uncorrelated, i.e. $SCC = 0$. However, depending on the values represented by the input bitstreams, sometimes $SCC = 0$ is not achievable. To find the alignment that results in the highest accuracy, one might assume that finding the alignment where $|SCC|$ is closest to 0 would result in the highest accuracy. This is however not always the case, as illustrated in Figure 1. When multiplying input values $P_X = \frac{3}{16}$ and $P_Y = \frac{14}{16}$, there are three possible alignment variants: one matched 1s ($SCC = -1$), two matched 1s ($SCC = -0.3846$), and three matched 1s ($SCC = +1$). If we use the best alignment according to the SCC metric ($SCC = -0.3846$), the final result has an error of $-\frac{10}{256}$, compared to the actual best alignment, which results in an error of $\frac{6}{256}$. From this counterexample, we see that while SCC can be applied to reason about correlation between two bitstreams, it has fundamental limitations in evaluating the independence of bitstreams when certain conditions are not met (i.e. when SCC cannot reach 0).

Other than SCC, the majority of prior works that analyze error either focus on aspects unrelated to correlation or account for multiple sources of error [5, 14]; thus they are not suited for evaluating independence. To better understand and isolate the errors stemming from a lack of input independence, it is desirable to have a metric that quantifies this. We make the following contributions:

- We demonstrate the disparity between bitstream independence and SCC. We perform a detailed analysis of SCC and show that SCC is not always able to identify when two bitstreams are independent (on average 20% of the time for 64-bit bitstreams).
- We propose a new metric called *zero correlation error* (ZCE), which allows for measuring the independence of two finite-length bitstreams, and present a step by step walk through of its derivation based on probability theory.

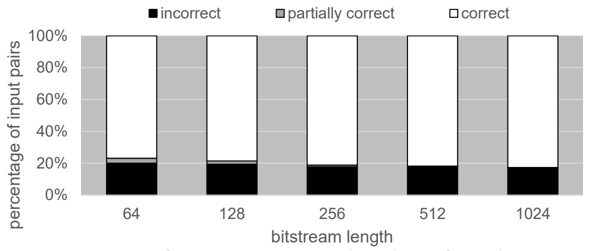


Figure 2: How often SCC correctly identifies the most independent input pairs. Correct refers to when SCC identifies the same alignment as the maximally independent alignment; partially correctly refers to when SCC identifies the same alignment as *one of* the maximally independent alignments (i.e. when two alignments have the same $|P_{X\wedge Y} - P_X P_Y|$); and incorrect refers to when SCC identifies a different alignment from the maximally independent one.

- We evaluate the practical application of ZCE, in terms of design space exploration, profiling, and determining optimal bitstreams.

2 BACKGROUND AND MOTIVATION

This section provides a primer on SC, independence and cross-correlation, and motivates the need for a new metric.

2.1 Stochastic Computing

Stochastic computing is a computing paradigm where data is encoded as unary bitstreams and their values are represented as the probability of a bit being set (e.g. a bitstream representing 0.5 will have half of its bits set and half unset in the unipolar representation) [9, 18]. For example, in Figure 1, the value $P_Y = \frac{14}{16}$ is represented by a bitstream of length 16, where 14 of the bits are set to 1, regardless of where in the bitstreams the 1s are located. Computation in SC leverages the transformation of probability values through various basic circuit gates. One major advantage of stochastic computing is that several arithmetic computations can be performed with very simple and small numbers of gates (e.g. multiplication using a single AND gate).

2.2 Independence of Bitstreams

In probability theory, if two events have their joint probability equal to the product of the two probabilities (i.e. $P_{X\cap Y} = P_X P_Y$), the two events are said to be *independent*. However, with finite-length bitstreams, $P_{X\cap Y}$ is not always attainable. Thus instead, SC uses the AND-gate result $P_{X\wedge Y}$, and we refer to this as the finite-length joint probability. Two finite-length bitstreams are said to be *as independent as they can be* if $P_{X\wedge Y} = P_X P_Y$. Because the result of the AND gate depends on the alignment of 0s and 1s, measuring independence of two bitstreams requires analyzing their alignments, also referred to as their *cross-correlation*.

2.3 SCC and Limitations

Stochastic cross correlation (SCC) is the de facto metric to assess the cross-correlation between two SC bitstreams [1]. The more aligned 0s and 1s there are between two bitstreams, the more *positively correlated* the two bitstreams are. Similarly, the more misaligned 0s and 1s there are between two bitstreams, the more *negatively*

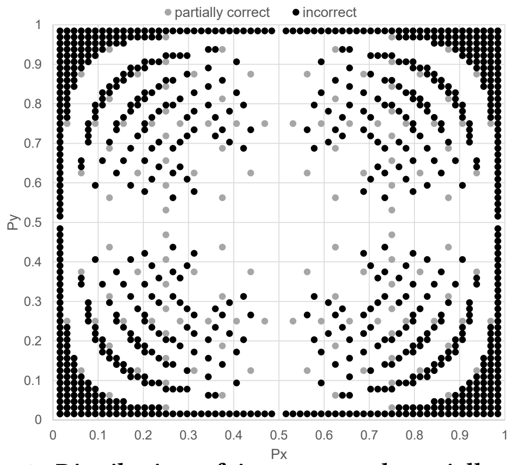


Figure 3: Distribution of incorrect and partially correct alignments for independence identified by SCC, for a bitstream length of 64. The horizontal and vertical axes show the values represented by two SC bitstreams P_X and P_Y, respectively. Black dots indicate combinations of P_X, P_Y values where SCC incorrectly identifies the maximally independent alignment. Grey dots indicate combinations of P_X, P_Y values where SCC only identifies one of the maximally independent alignments.

correlated the two bitstreams are. While SCC and independence are closely related, our insight in this work is recognizing that they are not equivalent when dealing with finite-length bitstreams.

The SCC of two input bitstreams X and Y is defined as:

$$SCC(X,Y) = \begin{cases} \frac{P_{X\wedge Y} - P_X P_Y}{\min(P_X, P_Y) - P_X P_Y} & \text{if } P_{X\wedge Y} > P_X P_Y \\ \frac{P_{X\wedge Y} - P_X P_Y}{P_X P_Y - \max(P_X + P_Y - 1, 0)} & \text{otherwise} \end{cases} \quad (1)$$

or alternatively:

$$SCC(X,Y) = \begin{cases} \frac{ad-bc}{L\cdot\min(a+b, a+c) - (a+b)(a+c)} & \text{if } ad > bc \\ \frac{ad-bc}{(a+b)(a+c) - L\cdot\max(a-d, 0)} & \text{otherwise} \end{cases} \quad (2)$$

where a, b, c, d are the number of bits where the alignment of (X, Y) are $(1,1), (1,0), (0,1), (0,0)$, respectively, and L is the length of the bitstream. For example, the inputs at the top of Figure 1 yield $a = 2, b = 1, c = 12, d = 1$. The SCC metric is designed to be value-independent and always outputs +1 if X and Y are maximally positively correlated and -1 if they are maximally negatively correlated. SCC outputs 0 if the degree of alignment (ab) equals the degree of misalignment (bc). All other points in between -1 and 1 are linearly interpolated.

In analyzing SC circuits, it can be tempting to use SCC as a proxy to assess the accuracy of computations if the correlation affinity of the functional unit is known. For example, if a functional unit is designed to yield the highest accuracy when the input bitstreams are most positively correlated (e.g. absolute subtract [2]), it makes sense to evaluate the SCC of the input bitstreams and make the conclusion that if a given pair of input bitstreams has SCC closer to 1, then the result of the computation is more accurate. *This line of reasoning works when the functional unit is designed with affinity towards positive or negative correlation but breaks down when it is designed to have affinity for independent inputs.*

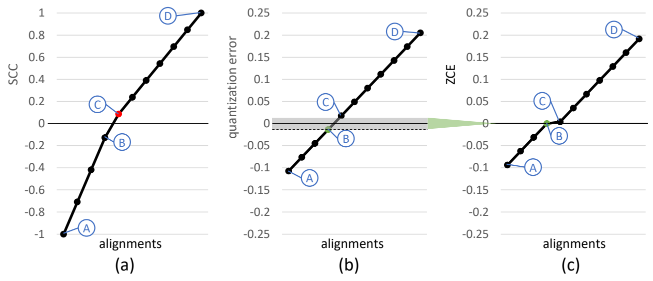


Figure 4: Plots of (a) SCC, (b) true error and (c) ZCE for all possible alignments of $P_X \times P_Y = \frac{21}{32} \times \frac{22}{32}$. Point B is the most independent alignment, whereas SCC incorrectly identifies point C as the most independent alignment.

In this work, we recognize that having close-to-zero SCC between two bitstreams does not necessarily imply that they are maximally independent, even though they can sometimes yield the same alignment. If we look at all pairs of values X and Y representable in a finite bitstream length and compare the most uncorrelated alignment (suggested by SCC) with the most independent alignment (suggested by minimizing $|P_{X\wedge Y} - P_X P_Y|$), we can get a sense of how often they correspond to each other. Figure 2 shows this across different bitstream lengths. We see that SCC is not able to identify the maximally independent alignment 18% of the time, on average. To further try to understand when this happens, Figure 3 shows the distribution of input value pairs where SCC incorrectly identifies the maximally independent alignment, or is only able to partially identify some maximally independent alignments. We can see that this phenomenon is not uniformly distributed across all input values, and there are particular input value pairs that pose difficulties (e.g. combinations of very small and very large values). Since the fraction of cases where SCC is not able to identify the maximally independent alignment is non-negligible, this further motivates the need for a better metric to quantify independence.

3 ZERO CORRELATION ERROR

We propose *zero correlation error (ZCE)* with two goals in mind:

(1) The metric should preserve the property where the more independent an alignment is, the closer to 0 it should yield;
(2) The metric should output 0 for all alignments that correspond to the most independent alignment possible given a finite bitstream length, allowing for quick identification of whether independence is achieved.

We want these properties so that it not only becomes possible to use ZCE to evaluate independence of bitstreams, but that it can also be used as a proxy to evaluate accuracy, and allows for efficient analysis and design exploration of SC circuits.

3.1 SCC vs. ZCE

This section illustrates why SCC cannot always identify the most independent alignment and describes what we want to accomplish in deriving ZCE. We plot the value of SCC, ZCE, and quantization error for all possible alignments of $P_X \times P_Y = \frac{21}{32} \times \frac{22}{32}$ in Figure 4. Comparing the SCC plot in Figure 4a and the quantization error plot in Figure 4b, we see that the SCC metric takes the two end points (A and D) and normalizes these points to to -1 and 1, respectively. While this was done in order to allow for identification of maximally positive and negative correlation, it results in a difference in slope when approaching 0 from the positive and negative side. This difference in slope is the reason why the $|SCC|$ closest to 0 is not always the most independent alignment. In this example, if we look at the quantization error, the point with the lowest absolute error is B, yet the point with the lowest absolute SCC is C (since there are more points on the positive side than the negative side). Looking at this plot, we can observe that SCC is perfectly capable of evaluating how positively or negatively correlated two bitstreams are, and is able to indicate whether a certain alignment has reached the most positive correlation possible ($SCC = +1$) or the most negative correlation possible ($SCC = -1$).

Visually, the intent of ZCE is twofold. First, we want to retain the slope of the line shown in the quantization error (to satisfy goal (1) above). Second, we want to "thicken" the crossing at 0 such that if the minimum positive error and the minimum negative error have the same magnitude, they both appear as $ZCE = 0$ (to satisfy goal (2)). Doing this gives us Figure 4c, which now enables us to evaluate how independent two bitstreams are and validate that a certain alignment has reached the most independence possible. In this example, we see that the points with the lowest absolute error and the lowest ZCE are both B. ZCE further provides a direction by maintaining the direction of error — ZCE is a positive value if two bitstreams are positively correlated, and vice versa.

Takeaway: SCC should still serve as the metric for determining maximally positive and maximally negative correlation. Mirroring this, our goal with ZCE is to instead determine how maximally independent two bitstreams are.

3.2 Derivation of ZCE

With the high-level goal of ZCE in mind, we now walk through the derivation of the ZCE equation.

Quantization Error: As mentioned in Section 2.2, two finite-length SC bitstreams P_X and P_Y are as independent as they can be when $P_{X\wedge Y} - P_X P_Y = 0$. Assume L is the length of the bitstream. While $P_X P_Y$ requires L^2 precision, $P_{X\wedge Y}$ only has L precision, which results in a certain amount of quantization error introduced. Given P_X and P_Y as inputs, the amount of quantization error for any arbitrary pair of bitstreams can be expressed as

$$\Delta_0 = 1/L \cdot \left\lfloor \frac{P_X P_Y}{1/L} + \frac{1}{2} \right\rfloor - P_X P_Y \tag{3}$$

The first term is the product of inputs quantized to $\frac{1}{L}$, and the second term is the desired product. Δ_0 therefore represents the amount of error when we have the most independent alignment (i.e. distance from 0 of point B in Figure 4(b)). Looking at the bit values in the two input bitstreams, if a, b, c, d are the number of bits where $(X, Y) = (1,1), (1,0), (0,1), (0,0)$ respectively, Equation 3 can also be equivalently expressed as

$$\Delta_0 = \frac{\left\lfloor \frac{(a+b)(a+c)}{L} + \frac{1}{2} \right\rfloor}{L} - \frac{(a+b)(a+c)}{L^2} \tag{4}$$

Note that in this equation, since $a + b$ equals the number of 1s in X and $a + c$ equals the number of 1s in Y, the value of Δ_0 is the same regardless of the actual alignment between X and Y.

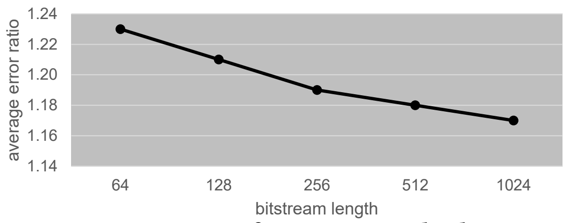


Figure 5: Average error of AND-gate multiplication when following the alignment recommended by SCC, normalized to quantization error.

Actual Error: For the two bitstreams ZCE is evaluating, they may have a different alignment from Δ_0 (i.e. can be any point on Figure 4(b)). We can compute the error (distance to the x-axis) as:

$$\Delta = P_{X \wedge Y} - P_X P_Y \tag{5}$$

where $P_{X \wedge Y}$ is the AND-gate result of the two input bitstreams. Again, Equation 5 can equivalently be expressed as

$$\Delta = \frac{a}{L} - \frac{(a+b)(a+c)}{L^2} \tag{6}$$

Zero Correlation Error: With Δ and Δ_0 defined, we can now use them to evaluate how close you are to the most independent alignment (displacement from the lowest error – i.e. removing the error due to limited precision) with the following metric:

$$\text{ZCE} = \frac{\Delta}{|\Delta|}\left(|\Delta| - |\Delta_0|\right) = \Delta\left(1 - \left|\frac{\Delta_0}{\Delta}\right|\right) \tag{7}$$

4 EVALUATION

In this section, we demonstrate how ZCE can be valuable in the design of SC systems via use cases in determining optimal alignments, profiling bitstreams and design space exploration.

4.1 Use Case: Finding Optimal Alignment

How do we align the input bitstreams such that our circuit yields the lowest error? This is an important question for SC circuit designers since the cross-correlation of input bitstreams can affect the accuracy of the computation. For functional units that have affinity towards independent input bitstreams (e.g. AND-gate multiplication, MUX-based scaled addition), ZCE is a good metric to help answer this question. In Section 2.3, we motivated the need for a better metric to quantify independence of bitstreams than SCC, and here we compare how well ZCE and SCC perform in this task.

Multiplication: In the first experiment, we evaluate the input bitstreams of an AND-gate multiply. For all possible values that the two input bitstreams can take, we use SCC and ZCE to recommend an optimal alignment. With each of the recommended alignments, we evaluate the accuracy of the multiplication output and compare it to the desired product of the two input values when quantized to a finite bitstream length. Because of the way ZCE is designed, the AND-gate products of ZCE's recommended alignments are all equivalent to the quantization results and are the best that can be achieved. In comparison, Figure 5 shows the average error across all possible input value pairs, normalized to quantization error, when the alignment recommended by SCC is used. We can see that the error decreases as the bitstream length increases, which is due to the diminishing contribution of a bit in the overall value as the bitstream length increases.

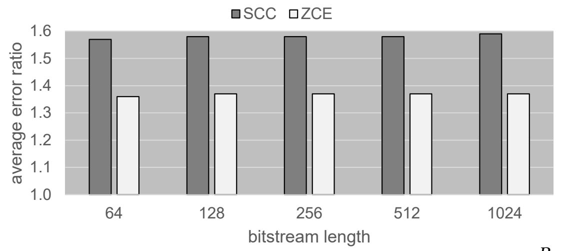


Figure 6: Average error of scale-by-4 addition ($P_Z = \frac{P_X+P_Y}{4}$) when following the alignment recommended by SCC and ZCE, normalized to quantization error.

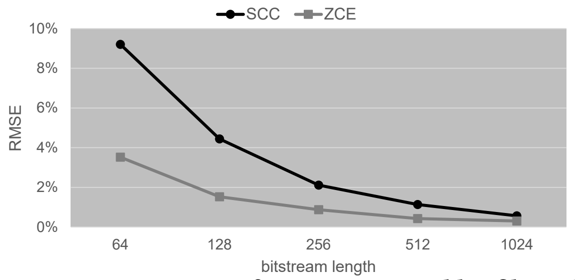


Figure 7: Average RMSE of a 3×3 Gaussian blur filter when following the alignment recommended by SCC and ZCE.

Scaled Addition: We perform the same experiment with a multiplexer that implements scaled addition in SC. For a scaled addition, the highest accuracy can be achieved when each of the input bitstreams are independent from the bitstream of the mux select signal. To evaluate how well SCC and ZCE can serve as a proxy for accuracy for a scaled addition, we pick an example computation: $P_Z = \frac{P_X+P_Y}{4}$. In the experiment, the select bitstream is kept the same ($\frac{1}{4}$), and SCC and ZCE are used to find the best alignment for the input bitstreams with respect to the select bitstream. Figure 6 shows the ratio of average error across all possible pairs if the recommended alignment of SCC and ZCE are used, normalized to quantization error. Here, we see that the error across different bitstream lengths stays mostly constant, and ZCE produces lower error than SCC. One thing to note is that unlike the AND-gate multiply experiment, ZCE does not necessarily achieve the lowest possible error (i.e. only quantization error). To understand this, we can look at the logic expression of a 2-to-1 mux, which is $Z = \neg SX + SY$. As shown previously, ZCE is able to find the most independent alignment for each of the two components ($\neg SX$ and SY) since they are both AND gates. However, because the error of a mux is now from more than one component, it is possible that aggregating a mix of positive and negative errors can cancel out and result in lower error. In ZCE's case, because it outputs 0 for the most independent alignment, the cancelling of positive and negative does not occur, leading to non-zero amounts of error compared to quantization error.

2D Convolution: We perform application-level analysis on a 2D convolution benchmark. In this experiment, we take a 256×256 grayscale image and convolve it with a 3×3 weight filter that represents a Gaussian blur. For each multiplication in the benchmark, SCC and ZCE are used to determine the best alignment for each pixel and weight value. The output of the multiplication is then accumulated in a parallel adder to sum up the result of nine products for each output pixel. The root-mean-square error (RMSE)

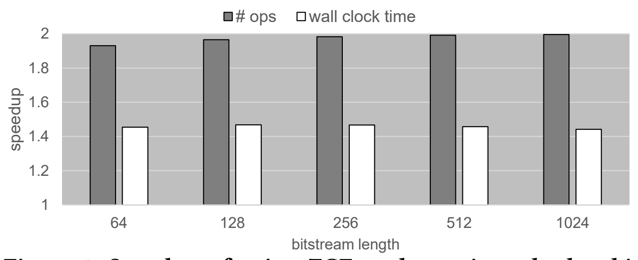


Figure 8: Speedup of using ZCE to determine whether bitstreams are optimally aligned compared to using SCC.

with respect to the floating-point version is computed for six different images, and the geomean across all six images for each of the different bitstream lengths are shown in Figure 7. We can see that ZCE consistently produces alignment recommendations that lead to higher overall accuracy than SCC, across all bitstream lengths. Again, the gap between the errors that result from a difference in alignment recommendation narrows as bitstream length increases, due to each bit having a lower contribution to the overall value as bitstream length increases.

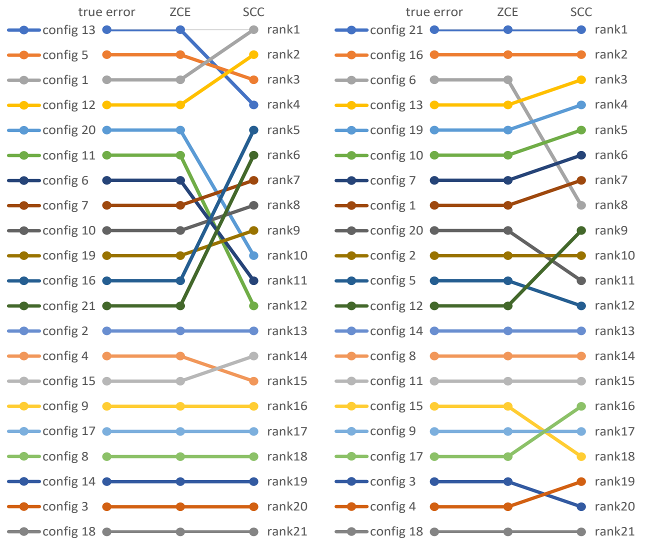
Takeaway: ZCE consistently identifies alignments that yield lower error than SCC, both at the functional unit level and at the application level. Thus, it can serve as a valuable tool for measuring the independence of bitstreams in arbitrary SC systems.

4.2 Use Case: Profiling Bitstreams

How quickly can we determine if our bitstreams are aligned as independently as they can be? This is an important question for designers who are simulating and debugging their SC circuits. We evaluate the amount of time it takes to answer whether bitstreams are optimally aligned using SCC and ZCE. Using ZCE to figure out whether a pair of bitstreams is optimally aligned simply involves checking whether ZCE equals zero. On the other hand, since there are input value pairs where it is not possible for SCC to output 0, using SCC to figure out whether two bitstreams are optimally aligned often involves evaluating the SCC of a second alignment in order to see which one yields $|SCC|$ closer to 0. Recall though that even when $|SCC|$ is closest to 0, it may not be the most independent alignment; for simplicity, since this experiment evaluates simulation time, we assume that identifying $|SCC|$ closest to 0 is sufficient.

In this experiment, we generate random bitstream alignments for all possible input value pairs and measure both the number of operations (i.e. number of times each metric is invoked) and the amount of time it takes to compute. We average the results over six different runs and show the speedup of using ZCE compared to using SCC in Figure 8. In terms of the number of operations, we observe that SCC needs to be computed twice most of the time, and the speedup of ZCE increases as the bitstream length increases. This is because the number of input value pairs where $SCC = 0$ is impossible (i.e. no alignment exists with $SCC = 0$) increases as bitstream length increases, from 91% for a length-64 bitstream to 99% for a length-1024 bitstream. If we look at the wall clock speedup, we see that using ZCE is faster by roughly 1.45×, even though ZCE's computation is more involved than SCC's.

Takeaway: ZCE can determine whether two bitstreams are optimally aligned faster than SCC since the optimal alignment is defined to always yield $ZCE = 0$. This simplifies and speeds up the profiling and debugging of SC circuits during design iterations.



(a) 6-bit LFSRs. **(b) 8-bit LFSRs.**

Figure 9: Ranking of 6-bit and 8-bit LFSR configurations based on true error (via AND-gate multiplication), ZCE and SCC, averaged over all possible input value pairs. ZCE's and SCC's rankings yield a Kendall's τ coefficient of 1 and 0.82 respectively for 6-bit LFSRs, and 1 and 0.83 for 8-bit LFSRs.

4.3 Use Case: Design Space Exploration

How can we compare different number generator designs and select the best one(s) for our circuit? When designing an SC application, the choice of random number generator can affect the cross-correlation between different values and thus the accuracy of the computation. In this exploration, we evaluate the independence of bitstreams generated by different pairs of linear feedback shift registers (LFSRs). For a 6-bit LFSR, there are six possible designs with different feedback taps, leading to a total of $\left(\!\binom{6}{2}\!\right) = 21$ different LFSR pairs, which we refer to as a configuration. For each configuration, we evaluate all possible value pairs and compute the average SCC and ZCE, as well as the average computation error.

Multiplication: The first experiment evaluates the ability of SCC and ZCE to rank LFSR configurations when bitstreams are used as inputs to an AND-gate multiply. Figure 9a shows the overall ranking of the configurations using the true error, ZCE and SCC. We see that ZCE is able to produce the same rank ordering as the true error, while SCC produces a slightly different ranking. Using Kendall's τ coefficient [10] to measure the rank correlation between SCC and true error, we get 0.82 out of a range of -1 (most dissimilar) to 1 (most similar). Figure 9b shows similar results when evaluating six possible designs (i.e. 21 configurations) for 8-bit LFSRs; SCC yields a Kendall's τ coefficient of 0.83.

2D Convolution: To evaluate ZCE's ability to rank number generator designs at an application level, we compare the true error, ZCE and SCC for the same 3×3 Gaussian blur benchmark with 6 different grayscale input images used in Section 4.1. All the input pixels share

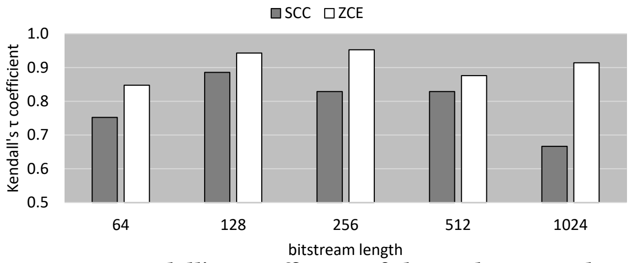


Figure 10: Kendall's τ coefficient of the rankings produced by ZCE and SCC for a 3×3 Gaussian blur benchmark.

a random number generator, and all the weight values share a random number generator. For each of the different bitstream lengths, six different LFSR designs are considered, resulting in 21 LFSR-pair configurations. For this experiment, the ranking produced by ZCE does not 100% agree with the ranking based on true error. This follows from our observation in Section 4.1, whereby using the ZCE-recommended alignment may not necessarily produce the lowest application error due to the effect of positive and negative errors cancelling out in the actual computation. Figure 10 shows the Kendall's τ coefficient for ZCE and SCC when compared to the ranking based on true error. Across different bitstream lengths, ZCE consistently produces a ranking that is more similar to the reference ranking than SCC.

Takeaway: Compared to SCC, evaluation of design choices using ZCE yields higher similarity to evaluation based on functional unit and application error. ZCE thus serves as a valuable metric for pruning design spaces and comparing different implementations / parameters of SC number generators.

5 RELATED WORK

As described in Section 2.3, SCC [1] is a metric that evaluates the cross-correlation between two SC bitstreams. SCC can be used to identify if and to what degree two bitstreams are positively or negatively correlated, whereas ZCE can be used to identify if and to what degree two bitstreams are independent. Other than SCC, there have been several tools in literature that address correlation. Problablistic transfer matrices (PTM) is an algebraic framework designed to help analyze correlation-induced errors in an SC circuit [1, 5]. With PTM, the user provides the probability of each input combination (e.g. P_{00}, P_{01}, P_{10}, P_{11}) in a row vector and performs matrix multiplication with a matrix that describes the logic function of the circuit. The resultant vector yields the expected value of the circuit with the given input correlation. Automated synthesis of number sequences used for random number generators has also been proposed to produce optimal accuracy for SC circuits [11]. Here a mixed integer programming formulation is used to generate number sequences to obtain maximum accuracy for a circuit. ZCE is orthogonal to these works, serving as an evaluation metric as opposed to a synthesis tool.

Another form of correlation for SC bitstreams is autocorrelation, which refers to the correlation of a bitstream with a delayed version of itself. SBoNG [13] is a random number generator designed to have low autocorrelation and has been proposed to enable higher potential for sharing among different bitstreams. SANG [4] is an algorithm that generates bitstreams with prescribed autocorrelation properties and has been proposed to help designers understand and manage the effects of autocorrelation. Though these works focus on number generation, ZCE instead provides a methodology for measuring bitstream independence.

6 CONCLUSION

In this work, we introduce ZCE, an independence metric for finite-length SC bitstreams. Through careful analysis of the existing SCC metric, we highlight limitations in its ability to evaluate bitstream independence and show how ZCE can overcome them. We then present several use cases where ZCE can be a valuable tool in the design cycle of SC systems. Compared to SCC, ZCE is able to provide better alignment recommendations for lower application error, faster identification of bitstream independence, and more accurate comparison between variants of random number generators when choosing what to include in an SC system.

7 ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful feedback, as well as Dr. Vincent T. Lee and Dr. Armin Alaghi for their valuable discussions.

REFERENCES

[1] A. Alaghi and J. P. Hayes. 2013. Exploiting correlation in stochastic circuit design. In *IEEE 31st International Conference on Computer Design*. 39–46.

[2] A. Alaghi and J. P. Hayes. 2013. Stochastic circuits for real-time image-processing applications. In *50th ACM/EDAC/IEEE Design Automation Conference*. 1–6.

[3] A. Ardakani et al. 2017. VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing. *IEEE Transactions on Very Large Scale Integration Systems* 25, 10 (2017), 2688–2699.

[4] T. Baker and J. Hayes. 2019. Impact of Autocorrelation on Stochastic Circuit Accuracy. In *IEEE Computer Society Annual Symposium on VLSI*. 271–277.

[5] T. Chen and J. P. Hayes. 2014. Analyzing and controlling accuracy in stochastic circuits. In *IEEE 32nd International Conference on Computer Design*. 367–373.

[6] Q. T. Dong et al. 2010. Stochastic Decoding of Turbo Codes. *IEEE Transactions on Signal Processing* 58, 12 (2010), 6421–6425.

[7] D. Fick et al. 2014. Mixed-signal stochastic computation demonstrated in an image sensor with integrated 2D edge detection and noise filtering. In *Proceedings of the IEEE Custom Integrated Circuits Conference*. 1–4.

[8] B. R. Gaines. 1967. Stochastic Computing. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. ACM, New York, NY, USA, 149–156.

[9] B. R. Gaines. 1969. *Stochastic Computing Systems*. Springer US, Boston, MA, 37–172.

[10] M. G. Kendall. 1938. A New Measure of Rank Correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[11] V. T. Lee et al. 2019. Synthesizing Number Generators for Stochastic Computing using Mixed Integer Programming. *CoRR* abs/1902.05971 (2019). arXiv:1902.05971

[12] P. Li and D. J. Lilja. 2011. Using stochastic computing to implement digital image processing algorithms. In *IEEE 29th International Conference on Computer Design*. 154–161.

[13] F. Neugebauer, I. Polian, and J. P. Hayes. 2017. Building a Better Random Number Generator for Stochastic Computing. In *Euromicro Conference on Digital System Design*. 1–8.

[14] F. Neugebauer, I. Polian, and J. P. Hayes. 2017. Framework for quantifying and managing accuracy in stochastic circuit design. In *Design, Automation & Test in Europe Conference & Exhibition*. 1–6.

[15] A. Ren et al. 2017. SC-DCNN: Highly-Scalable Deep Convolutional Neural Network Using Stochastic Computing. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. Association for Computing Machinery, New York, NY, USA, 405–418.

[16] H. Sim and J. Lee. 2019. Log-Quantized Stochastic Computing for Memory and Computation Efficient DNNs. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. Association for Computing Machinery, New York, NY, USA, 280–285.

[17] S. S. Tehrani, W. J. Gross, and S. Mannor. 2006. Stochastic decoding of LDPC codes. *IEEE Communications Letters* 10, 10 (2006), 716–718.

[18] J. von Neumann. 1956. Probabilistic logics and synthesis of reliable organisms from unreliable components. In *Automata Studies*, C. Shannon and J. McCarthy (Eds.). Princeton University Press, 43–98.