Multiplexer-Majority Chains: Managing Correlation and Cost in Stochastic Number Generation

Timothy J. Baker, Owen Hoffend and John P. Hayes
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109
{bakertim, ohoffend, jhayes}@umich.edu

ABSTRACT

High-cost stochastic number generators (SNGs) are the main source of stochastic numbers (SNs) in stochastic computing. Interacting SNs must usually be uncorrelated for satisfactory results, but deliberate correlation can sometimes dramatically reduce area and/or improve accuracy. However, very little is known about the correlation behavior of SNGs. In this work, a core SNG component, its probability conversion circuit (PCC), is analyzed to reveal important tradeoffs between area, correlation, and accuracy. We show that PCCs of the weighted binary generator (WBG) type cannot consistently generate correlated bitstreams, which leads to inaccurate outputs for some designs. In contrast, comparator-based PCCs (CMPs) can generate highly correlated bitstreams but are about twice as large as WBGs. To overcome these area-correlation limitations, a novel class of PCCs called multiplexer majority chains (MMCs) is introduced. Some MMCs are area efficient like WBGs but can generate highly correlated SNs like CMPs and can reduce the area of a filtering circuit by 30% while sacrificing only 7% accuracy. The large influence of PCC design on circuit area and accuracy is explored and suggestions are made for selecting the best PCC based on a target system's correlation requirements.

CCS CONCEPTS

• Hardware ~ Emerging technologies ~ Analysis and design of emerging devices and systems; • Hardware ~ Very large-scale integration design ~ Application-specific VLSI designs

KEYWORDS

Probabilistic computing, correlation, accuracy, simulation.

ACM Reference format:

Timothy J. Baker, Owen Hoffend and John P. Hayes. 2022. Multiplexer Majority Chains: Managing Correlation and Cost in Stochastic Number Generation. In *Proceedings of ACM International Symposium on Nanoscale Architectures (NANOARCH'22)*. ACM. New York, NY, USA. 6 pages.

NANOARCH'22, December 7–9, 2022, Virtual, OR, USA
 978-1-4503-9938-8/22/12...$15.00
https://doi.org/10.1145/3565478.3572326

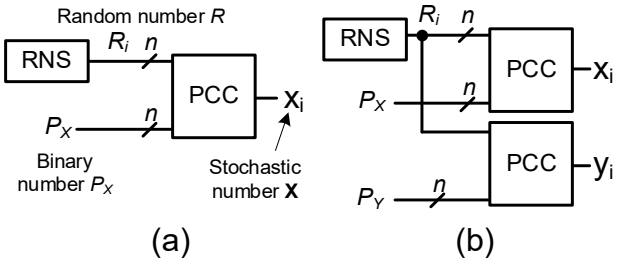

Figure 1: (a) SN generator using a random number source (RNS) and a probability conversion circuit (PCC) to create SN X. (b) Sharing an RNS between two SN generators to create correlated SNs X and Y.

1 Introduction

Modern edge devices can benefit greatly from implementing computationally demanding algorithms on-chip [1]. However, they often have strict energy and area budgets that make such implementations difficult or infeasible. A potential solution to this challenge is stochastic computing (SC) [2]. In SC, data are encoded into pseudo-random bitstreams called stochastic numbers (SNs) which are used to perform very low-cost computation. For instance, a single AND gate can multiply two SNs. Low multiplication cost and other features of SC like fault tolerance have made SC a promising technology for on-chip implementations of useful algorithms related to digital filtering [3][4], image processing [5][6][7], and neural networks [8][9][10].

Although SC is cheap, encoding non-stochastic data into SNs is very costly and subject to various subtle error types. A typical SN generator (Fig. 1a) consists of a random number source (RNS) and a probability conversion circuit (PCC). The RNS supplies a sequence of random numbers which the PCC converts to a probabilistic bit-stream forming SN **X**. SN generators are needed not only for primary input variables, but also to restore internal SNs whose randomness has been degraded by prior computation. SN generators often form the largest part of a stochastic circuit, and their high cost is a major obstacle to the deployment of SC [5].

Partly in response to SNG cost, there has been growing interest in stochastic circuits that deliberately take advantage of correlation [11]. Examples are found in image processing and neural networks [6][7][9][10]. Correlation-based circuits can lower SNG overhead by allowing two or more SNs to share one RNS, as in Fig. 1b. Such

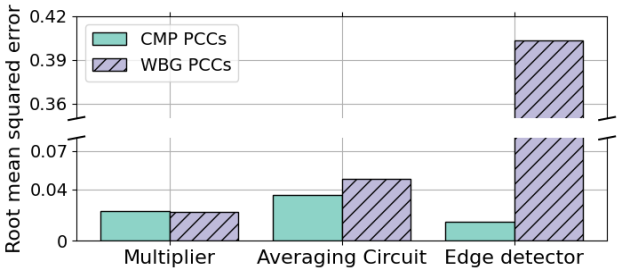

Figure 2: Comparison of output error for various circuit types whose PCCs are either CMPs (solid bars) or WBGs (striped bars). Examples include basic multiplication, weighted addition [3] used in digital filtering or neural networks, and edge detection used in image processing [5].

designs, however, still require one PCC per distinct SN, making PCCs the predominant factor influencing stochastic circuit area. Optimizing PCC design can therefore greatly improve cost. For example, replacing traditional comparator-based PCCs (which we call CMPs) with weighted binary generators (WBGs) can reduce overall circuit area by 50-60% in SC digital filter designs [3][4].

In addition to circuit size, PCC design can significantly affect circuit accuracy, a fact that has been noticed, but has received little attention [3][12]. Fig. 2 shows the expected error level for three diverse stochastic designs [3][5]. In each case, the average output error is measured when using either CMP or WBG PCCs. As can be seen, the PCC's impact on errors varies widely. A key motivation for this work is to explain behavior like that depicted in Fig. 2 and show when and how PCC choice influences accuracy.

The main contributions of this work are:

1. Demonstration that area-efficient WBG PCCs cannot generate maximally correlated SNs and so are unsuitable for correlation-reliant designs.
2. The novel multiplexer-majority chain (MMC) PCC design which enables flexible area-correlation trade-offs in SC.
3. Two case studies that illustrate important, but subtle, area-accuracy trade-offs in PCC design.

2 Background

Here, we review relevant concepts from stochastic computing.

2.1 Stochastic Computing Basics

SC uses SNs to represent data. An SN **X** is a sequence of random bits $x_1, x_2, \ldots, x_N$ where each bit has the same probability P_X of taking value 1, namely, $P_X = \mathbb{P}(x_i = 1)$. **X**'s value depends on the format used. Here, we focus on unipolar format where **X**'s value is simply P_X, but our results readily extend to the bipolar SN format which is used to represent negative numbers in SC.

A central advantage of SN encoding is that arithmetic is performed using simple logic elements. For example, the AND gate in Fig. 3 with SN inputs **X** and **Y** and output **Z** acts as a multiplier because $P_Z = P_X P_Y$ if **X** and **Y**'s bits are independent or uncorrelated. Addition also has very low cost in SC. Both multiplexer (mux) and majority (maj) gates can be used to add SNs **X** and **Y** by using a control SN **R** with value $P_R = 0.5$. For a maj

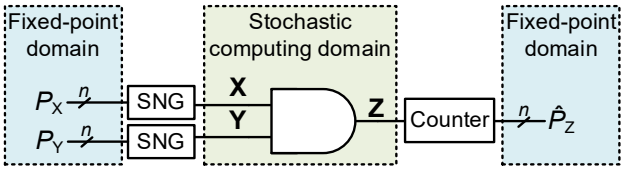

Figure 3: Generic SC system using SN generators (SNGs) to convert from fixed-point format to SNs, and a counter to convert output SN Z to a fixed-point estimate of Z's value.

adder, **R** serves as the third input along **X** and **Y** while for a mux adder, **R** serves as the select signal. In both cases, the mux or maj outputs **Z** with $P_Z = 0.5(P_X + P_Y)$ [13].

2.2 SN Generation and Estimation

Fig. 3 shows a how an SC circuit is embedded in a traditional computing system. SN generators are used to convert fixed-point values P_X and P_Y into SNs **X** and **Y**. Then **X** and **Y** are processed through an SN datapath such as a multiplier. The computation output **Z** is converted back to fixed-point by a counter. Often, the area of an SC system is dominated by the SN generators [5].

Fig. 1a shows the prototypical SNG consisting of an RNS and a PCC where an LFSR serves as the RNS [2]. The output of the RNS during the its clock cycle is $R_i \in [0,1)$, a uniform random variable. While LFSRs normally do not output $R_i = 0$, they can be easily modified to do so [14] and all LFSRs considered in this paper are assumed to be modified in this way. The PCC uses R_i along with its value input P_X to generate an SN bit x_i that satisfies

$$\mathbb{P}(x_i = 1) = P_X. \tag{1}$$

For example, a digital comparator that outputs $x_i = R_i < P_X$ is a common PCC type. Since R_i is uniformly distributed, $\mathbb{P}(x_i = 1) = \mathbb{P}(R_i < P_X) = P_X$ confirming that the comparator satisfies (1) and is therefore a valid PCC. Another PCC type is the weighted binary generator (WBG) [15] whose two-stage implementation is illustrated in Fig. 4.

After generation, SNs are processed through an SN arithmetic circuit (e.g., an AND gate multiplier or a mux adder) and produce an output SN **Z,** as illustrated in Fig. 3. If **Z**'s length L is a power of two, a digital counter can then be used to determine the frequency of 1s in **Z** as $\hat{P}_Z = \frac{1}{L}\sum_{i=1}^{L} z_i$, effectively converting from the SC domain back to the fixed-point domain.

The counter's output $\hat{P}_Z$ is an estimate of **Z**'s target value P_Z^*, which is the intended result of the computation. The difference between $\hat{P}_Z$ and P_Z^* is the circuit's error which fluctuates randomly due to the stochasticity of **Z**. Expected error can be quantified with the mean square error (MSE) metric

$$\text{MSE} = \mathbb{E}\left[P_Z^* - \hat{P}_Z\right] \tag{2}$$

Generally, SC has a fundamental accuracy-latency trade-off as using longer SNs leads to lower MSE because random errors tend to average out over many clock cycles [16].

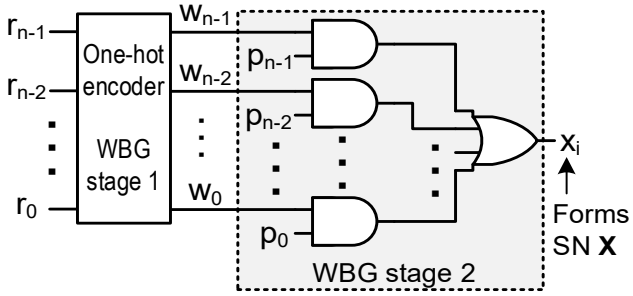

Figure 4: An *n*-bit weighted binary generator (WBG) used as a PCC to generate SN $\mathbf{X} = x_1 x_2 \ldots x_N$ with $P(x_i = 1) = P_X$.

2.3 Correlation

Correlations often arise between SN bits with major implications on circuit function and accuracy. In SC, the stochastic cross-correlation (SCC) metric [11] is typically used to quantify correlation between bits of two SNs **X** and **Y**.

$$\mathrm{SCC}(\mathbf{X},\mathbf{Y}) = \begin{cases} \dfrac{P_{X\wedge Y} - P_X P_Y}{\min(P_X, P_Y) - P_X P_Y} & \text{if } P_{X\wedge Y} \geq P_X P_Y \\ \dfrac{P_{X\wedge Y} - P_X P_Y}{P_X P_Y - \max(P_X + P_Y - 1, 0)} & \text{otherwise} \end{cases} \quad (3)$$

SCC's numerator is the difference between the actual overlap of 1s in two SNs $\left(P_{X\wedge Y} = \mathbb{P}(x_i \wedge y_i = 1)\right)$ and the expected overlap of 1s ($P_X P_Y$). SCC's denominator normalizes the value such that an SCC of $+1$ indicates maximum overlap of 1s based on P_X and P_Y. Likewise, $\mathrm{SCC} = -1$ indicates a minimum overlap of 1s, and $\mathrm{SCC} = 0$ indicates that the two SNs are uncorrelated. For example, $\mathbf{A} = 110101$ and $\mathbf{B} = 100101$ have an estimated SCC of $+1$ because their 1s overlap as much as possible, while **A** and $\mathbf{C} = 001010$ have an estimated SCC of -1 because their 1s never overlap. Two SNs **X** and **Y** have $\mathrm{SCC} = 0$ when $P_{X\wedge Y} = P_X P_Y$.

In SC design, correlation can be exploited to change circuit function [11]. For example, an AND gate multiplies uncorrelated inputs **X** and **Y** with $\mathrm{SCC} = 0$, but an AND gate outputs $P_Z = \min(P_X, P_Y)$ when **X** and **Y** are maximally correlated with $\mathrm{SCC} = 1$. Using correlated SNs to implement functions like min, max and absolute value has practical applications in image processing [6][7] sorting networks [17] and neural networks [9][10]. For example, a neural network's max pooling and ReLU activation functions rely on the maximum function which can be implemented with just an OR gate if the input SNs are correlated.

3 Multiplexer Majority Chains

This section introduces a novel and flexible PCC design style based on systematically combining mux and maj gates.

3.1 Design

Fig. 5 shows a 5-bit version of our proposed multiplexer-majority chain (MMC) framework. Like all 5-bit PCCs, the MMC has a random input $R_i = r_4 r_3 r_2 r_1 r_0$ and a value input $P_X = p_4 p_3 p_2 p_1 p_0$; it outputs a single SN bit x_i. Each *M* block in the MMC performs the same stochastic add operation [2]:

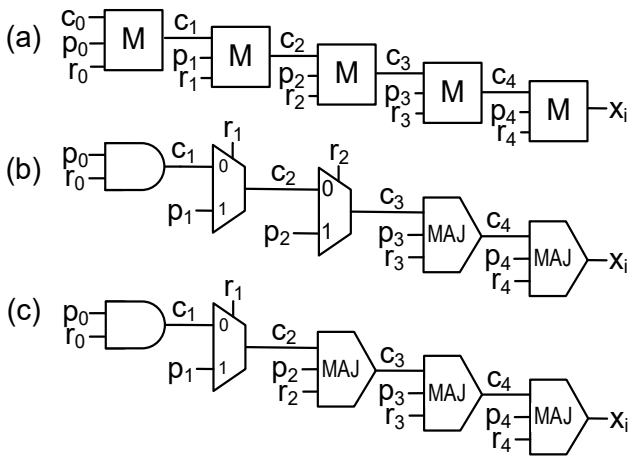

Figure 5: MMC framework. (a) 5-bit MMC. Each *M* block computes (4) and is implemented by either a mux or maj gate. Note the leftmost *M* block always simplifies to an AND gate because $c_0 = 0$. (b-c) Examples of 5-bit MMC PCCs.

$$\mathbb{P}(c_{j+1} = 1) = \frac{1}{2}\left(\mathbb{P}(p_j = 1) + \mathbb{P}(c_j = 1)\right) \quad (4)$$

The chain's output $x_i = c_5$ is seen to satisfy the PCC equation (1) by recursively applying (4) and noting that P_X's bits are nonrandom $\left(\mathbb{P}(p_j = 1) = p_j\right)$. In particular,

$$\mathbb{P}(x_i = 1) = \frac{1}{2}(p_4 + \frac{1}{2}(p_3 + \frac{1}{2}(p_2 + \frac{1}{2}(p_1 + \frac{1}{2}(p_0 + 0)))))$$

which, when expanded, yields

$$\mathbb{P}(x_i = 1) = \frac{1}{2}p_4 + \frac{1}{4}p_3 + \frac{1}{8}p_2 + \frac{1}{16}p_1 + \frac{1}{32}p_0 = P_X$$

implying that (1) is satisfied and that the MMC is a valid PCC.

The stochastic operation (4) of each *M*-block is equivalent to a basic SN scaled addition operation. In a similar vein as SN addition, each *M*-block can be implemented by either a mux or a maj gate [13]. For example, Fig. 5b is a mux-maj chain that uses two muxes and two maj gates while Fig. 5c uses one mux and three maj gates. Figs. 5b and 5c also explicitly show that the leftmost *M*-block always simplifies to an AND gate because $c_0 = 0$.

An important constraint we impose on an MMC's design is that the chain must consist of mux gates followed by maj gates, as illustrated in Fig. 5. In other words, maj gates always occupy the chain stages corresponding to the most significant bits (MSBs) of the PCC's R_i and P_X inputs. Although designs that do not follow this rule are still valid PCCs, the restriction has a significant effect on correlation, as explained in Sec. 3.3. Adhering to this restriction implies that there are *n* possible *n*-bit MMC designs.

A specific MMC can be identified by a parameter *k*, the number of maj gates used in the design. When $k = 0$, each *M*-block is implemented as a mux gate. The resulting PCC is a chain of mux gates which has appeared before [8] and is logically equivalent to a WBG. Thus, the WBG is special case of an MMC. Likewise, when $k = n - 1$, each *M*-block in an *n*-bit MMC is implemented as a maj gate. The resulting maj chain design has also appeared before [18] but has rarely been explicitly used.

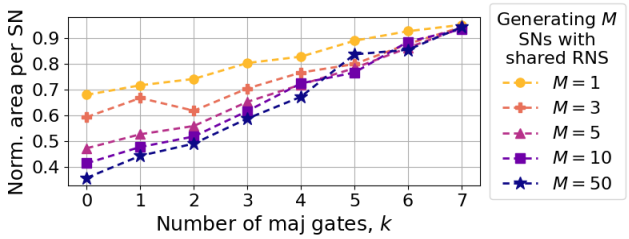

Figure 6: MMC correlation-area tradeoff. Synthesized 8-bit MMC area per SN. Area is normalized by dividing by the PCC area of one CMP-type PCC.

Interestingly, a chain of n maj gates forms the carry (or borrow) logic of an n-bit ripple subtracter and can be shown to compute $\bar{R}_i < P_X$ where $\bar{R}_i = \bar{r}_{n-1}\bar{r}_{n-2} \ldots \bar{r}_0$. Since $\mathbb{P}(r_j = 1) = P(r_j = 0) = 0.5$, the inversions on $\bar{R}_i$'s bits can be absorbed into the RNS allowing us to henceforth consider a chain of maj gates as implementing a standard CMP that computes $R_i < P_X$. Thus, in terms of Boolean logic, both WBGs and CMPs are special cases of MMCs. The MMC framework also generalizes to novel PCC designs which are implemented with a mix of mux and maj gates. Examples of new PCC designs include Figs. 5b and 5c. Next, we analyze the area and correlation properties of MMCs.

3.2 Area Analysis

SN generators often dominate the area of a stochastic circuit [5]. Thus, minimizing PCC cost is crucial to meeting SC's promise of low-cost, yet computationally powerful circuits. Here, we investigate how an MMC's area varies with k, the number of maj gates in the chain. To synthesize all designs, Synopsys Design Compiler (DC) with the Nangate 45nm cell library [19] is used. Synopsys DC is given a Verilog description of the MMC's logic and the final synthesized area is reported. The objective of this area analysis is to characterize the PCC based on its Boolean function rather than its physical chip area; the objective is not to compare chain implementations against non-chain implementations of PCCs. Note that the design synthesized by DC is not necessarily a chain of mux and maj gates.

The $M = 1$ curve in Fig. 6 shows that a single MMC area's increases linearly with k. When $k = 7$, the MMC resembles a CMP and area is at its highest. When $k = 0$, the MMC is a WBG, and has an area 28% lower than that of a CMP. We also synthesized 6-bit through 12-bit MMCs and found similar results (which are omitted here for brevity).

When generating two or more correlated SNs with a shared RNS as in Fig. 1b, considerable area can be saved by sharing part of the WBG alongside the shared RNS [4]. In contrast, no portion of a CMP can be shared in a similar manner, and so WBGs become even more area efficient than CMPs when generating many correlated SNs. Here, we extend [4] by asking: Do MMCs become more area efficient when generating correlated SNs?

Fig. 6 shows the per-SN area cost of using MMCs to generate M correlated SNs with $M = 3,\ 5,\ 10$ or 50. For each M, the MCCs' area increases linearly with k as in the single $M = 1$ SN

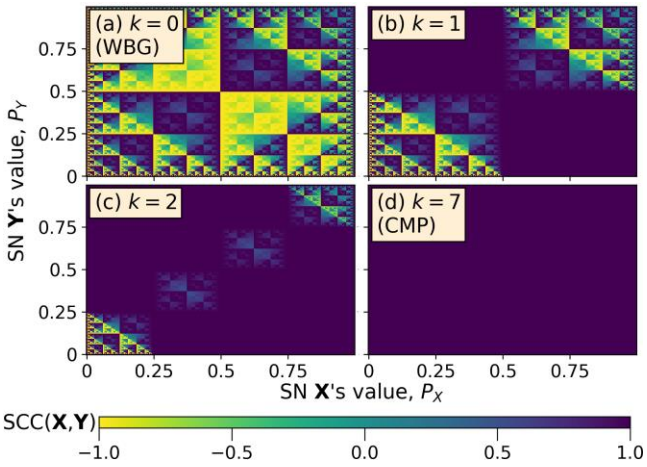

Figure 7: SCC(X,Y) as a function of P_X, P_Y for 8-bit MMCs with various numbers of maj gates, k. The desired SCC in all plots is SCC(X, Y) = 1 (dark purple) for all P_X, P_Y.

case, but the area efficiency improves. For example, when $M = 50$ the area per SN for WBGs is 62% lower than for CMPs. This 62% difference is much better than the 28% difference noted in the single SN case. When $1 \leq k \leq 6$, the per-SN area of the MMC is also reduced compared to the single SN case, implying that hybrid MMCs also become more area efficient when generating SNs with a shared RNS, particularly for small k. Note that M in Fig. 6 is determined by the application, specifically by how many input SNs that share an RNS are needed.

3.3 Correlation Analysis

Recent work has demonstrated that correlation amongst input SNs can sometimes drastically change a circuit's function [11] or improve its area or accuracy [3][4]. The most frequent correlation level required by such techniques is a pairwise SCC of +1 between all or most input SNs. The correlated SN generator in Fig. 1b is often used to generate SNs with SCC(**X**, **Y**) = 1, however, the actual SCC of the generated SNs can vary wildly depending on the PCC used, as we show next.

Fig. 7 plots SCC(**X**, **Y**) as a function of P_X, P_Y when using the correlated SN generator in Fig. 1b with various PCCs. Fig. 7d corresponds to using CMPs and shows SCC = 1 for all P_X, P_Y, as desired. In contrast, Fig. 7a uses WBGs and shows that the SCC varies greatly with P_X, P_Y and that the SCC often takes negative values, which is antithetical to the goal of SCC = 1. Thus, although WBGs are very area efficient at producing SNs with a shared RNS (Fig. 6), a WBG is unable to consistently generate maximally correlated SNs. This drawback of WBGs is an important conclusion of our work, and we examine its consequences in Sec. 4.

Next, Fig. 7b shows how SCC(**X**,**Y**) varies with P_X and P_Y, for an 8-bit MMC with $k = 1$. The overall correlation is much higher than in the WBG case (Fig. 7a) although the number of maj gates k has only increased from 0 to 1. When k is increased further to $k = 2$, Fig. 7c shows that SCC approaches +1 for many more values of P_X, P_Y. To illustrate this trend, Fig. 8 plots SCC(**X**,**Y**)

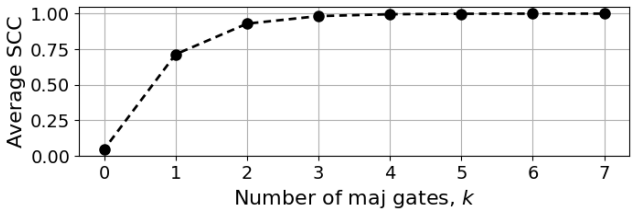

Figure 8: SCC averaged across all possible values of P_X, P_Y when using 8-bit MMCs in a correlated SN generator.

averaged over all P_X, P_Y against the MMC parameter k. When $k = 0$, the MMC is equivalent to a WBG, and the average SCC is nearly 0. As k increases, the average SCC quickly rises until it hits a maximum value 1 when $k = 7$ and the MMC is a CMP-type PCC. The reason for the rapid growth of correlation in Fig. 8 is that the maj gates always act on the MSBs of the MMCs' value inputs P_X and P_Y. Consequently, the maj gates have significant influence on the MMC output and cause the correlation to reflect that of a CMP more so than that of a WBG.

4 Digital Filtering Case Studies

The following two case studies use MMCs to show the large influence that PCCs have on circuit cost and accuracy. For both cases, the SNG precision is set to 8 bits and 256-bit SNs are used. The SNGs employ MMC PCCs where k varies from 0 to 7.

4.1 Finite Impulse Response Filtering

Digital finite impulse response (FIR) filters are widely used to denoise signals. CeMux is an SC mux-based addition method designed to implement low-cost FIR filtering [3]. CeMux computes

$$Z_t = \frac{1}{\sum_{k=0}^{M-1}|h_k|}\sum_{k=0}^{M-1} h_k X_{t-k} \quad (5)$$

where M is the number of filter taps, X_t is the noisy input signal, h_k are the constant filter coefficients, and Z_t is the filtered signal. To implement (5), CeMux encodes the input signal X_t into a set of SNs $\mathbf{X}_1$, $\mathbf{X}_2$ …, $\mathbf{X}_M$ that are processed through a tree of multiplexers whose final output is $\mathbf{Z}$ with value Z_t (5) [3]. CeMux is up to 12x more accurate than other mux-based adders and relies heavily on correlation to reduce error. For example, it uses a shared RNS with CMP PCCs to carefully correlate its input SNs $\mathbf{X}_1$, $\mathbf{X}_2$, ..., $\mathbf{X}_M$ in a way that improves accuracy.

Here, we investigate how CeMux's area and accuracy vary with its MMC parameter k when implementing an electrocardiogram (ECG) denoising task like that of [3]. CeMux is synthesized using Synopsys DC and signal-to-noise ratio (SNR) is used to quantify CeMux's accuracy:

$$\text{SNR} = 10\log_{10}\frac{\mathbb{E}[P_{Z_t}^2]}{\text{MSE}} \quad (6)$$

where $\mathbb{E}[P_Z^2]$ is the average signal power and MSE (2) quantifies the average noise power.

Fig. 9a illustrates the experimental results. Since PCCs account for about 80% of CeMux's area [3], its area is highly influenced by

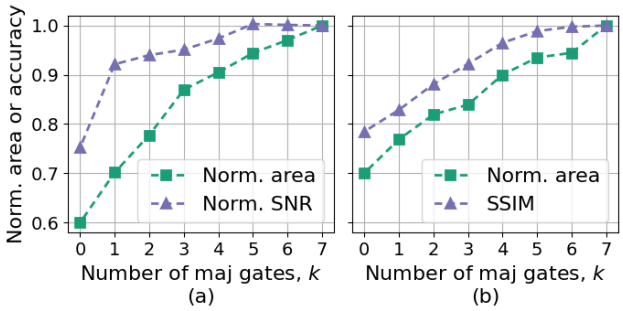

Figure 9: Influence of MMC design on circuit area and accuracy for (a) CeMux FIR filter; (b) SC median filter. Normalized values are computed by dividing by the area or SNR corresponding to $k = 7$.

the MMC design and CeMux's overall area increases linearly with k. Compared to CMPs, using WBGs leads to a significant area savings of 40% but degrades accuracy by about 25% because WBGs fail to maximally correlate input SNs. In contrast, using MMC PCCs with $k = 1$ in CeMux saves 30% area while only degrading accuracy by about 7% because the MMCs mostly generate highly correlated input SNs.

Overall, when accuracy must be maximized, CeMux should employ CMPs, i.e., MMCs with $k = 7$. However, area can be greatly reduced in exchange for a small amount of accuracy by using MMCs with fewer maj gates. Using MMCs with $k = 1$ yields the best trade-off where a significant 30% of area is saved in exchange for 7% lower SNR. MMCs with higher k values can also be used if slightly more accuracy is desired.

4.2 Median Filtering

Median filters are effective at filtering out impulse noise types. For example, the image in Fig. 10a corrupted by salt-and-pepper noise can be mostly recovered by convolving the noisy image with a 3x3 median filter as shown by the filtered image in Fig. 10d. A 3x3 median filter replaces each noisy image pixel with the median value of its surrounding 8 noisy pixels and itself. In SC, a median filter can be implemented very efficiently by using AND/OR gates to implement a series of MIN and MAX operations on correlated SNs [6][17]. For accurate computation, SC median filters require that their input SNs be maximally correlated with a pairwise SCC of +1.

To evaluate the tradeoff between area and performance for the 3x3 median filter circuit, we corrupted 10 grayscale test images from the MATLAB image processing toolbox with random salt-and-pepper noise. Each pixel was assigned a 5% chance of becoming corrupted to all-black or all-white. An SC median filter was then simulated with each corrupted image as input and the structural similarity index measure (SSIM) [20] between the circuit's output and target output was calculated.

In [20], it is shown that two distorted images having the same MSE relative to an unmodified image can vary substantially in terms of perceived visual quality. SSIM is a similarity measure designed to better capture perceptual differences between images. It is based on the hypothesis that the human visual system is better

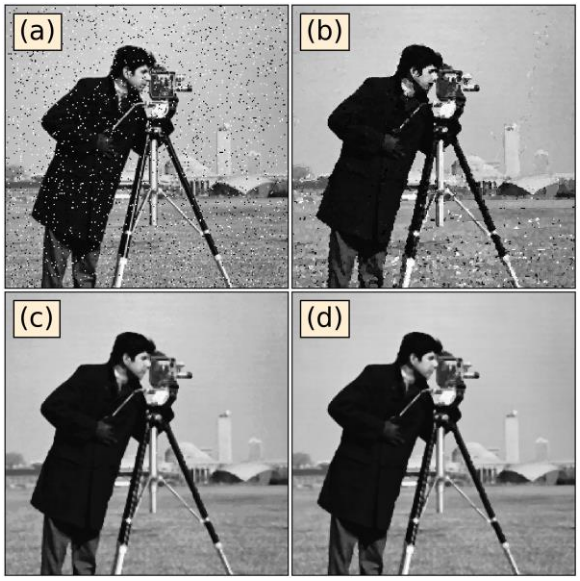

Figure 10: (a) Image corrupted by random salt-and-pepper noise. Image filtered by SC median filter with (b) WBG PCCs; (c) CMP PCCs; (d) MMC PCCs with $k = 4$ maj gates.

adapted to perceiving large-scale structural features of images compared to low-level details or differences in luminance and contrast. SSIM varies between 0 and 1 where a higher SSIM implies the two images are more visually similar.

Fig. 9b shows how the median filter's area and SSIM varies with the MMC parameter k. The median filter area, including the SNGs and median filter circuit, varies linearly with k. Using WBG PCCs in place of CMPs leads to an area savings of 30%, but also a low SSIM of 0.78. Fig. 9b indicates that SSIM grows with k, but its growth exhibits diminishing returns after $k > 4$, implying that $k = 4$ is a good choice for the median filter MMCs. In that case, the SSIM is 0.965 out of a maximum of 1 and area is 10% lower than using CMPs.

Fig. 10 shows the "cameraman" test image and the SC median filter's output when using MMCs with different k values. The filtering performed is poor when using WBG PCCs, which highlights the drawback of employing low-cost WBG PCCs in designs that require highly correlated SNs. In contrast, MMC PCCs with $k = 4$ maj gates lead to effective filtering and an SSIM of 0.97, with an area savings of about 10% compared to using CMPs.

5 Discussion

The influence of PCC design on circuit area and accuracy varies with the application as demonstrated by Fig. 2 and Sec. 4's case studies. Circuits like CeMux that benefit from, but do not require, correlated input SNs can use CMPs to maximize accuracy or can use MMCs with few maj gates (e.g., $k = 1$) to save area in exchange for some accuracy loss. Circuits like SC median filters that require highly correlated inputs can use CMPs to maximize accuracy or use MMCs with many maj gates (e.g., $k = 4$ for 8-bit precision) to save area at the cost of some accuracy. Finally, circuits that require independent inputs, like the SN multiplier should use separate RNSs to ensure SN independence and should always employ WBG PCCs to achieve the lowest area.

Overall, input correlation revolves around SN generator design. Here, we introduced MMCs to obtain a unified view of WBG and CMP PCCs, which enabled in-depth analysis of the area and correlation properties of PCC designs. As demonstrated in the two case studies, MMCs not only provide insight into correlation, but they also enable designers to trade off cost and accuracy in SN generation.

ACKNOWLEDGEMENT

This research was supported by the U.S. National Science Foundation under grant CCF-2006704.

REFERENCES

[1] A. Marchisio et al. 2019. Deep learning for edge computing: current trends, cross-layer optimizations, and open research challenges. *Proc. ISVLSI,* 553-559.

[2] B.R. Gaines. 1969. Stochastic computing systems. *Advances in Information Systems Science,* Springer New York, 37-172.

[3] T.J. Baker and J.P. Hayes. 2022. CeMux: maximizing the accuracy of stochastic mux adders and an application to filter design. *ACM Trans. on Design Auto. Elec. Sys*, **27**, 3, 1-26.

[4] K. Zhong, M. Yang and W. Qian. 2018. Optimizing stochastic computing-based FIR filters. *Proc. DSP*, 1-5.

[5] V.T. Lee et al. 2018. Architecture considerations for stochastic computing accelerators. *IEEE Trans. CAD,* **37**, 11, 2277-2289.

[6] R.K. Budhwani, R. Ragavan and O. Sentieys. 2017. Taking advantage of correlation in stochastic computing. *Proc. ISCAS,* 1-4.

[7] H. Abdellatef, M.K. Hani and N. Shaikh-Husin. 2019. Accurate and compact stochastic computations by exploiting correlation. *Turkish J. Elec. Engin. & Comp. Sci,*, **27**, 1.

[8] B.D. Brown and H.C. Card. 2001. Stochastic neural computation. I. Computational elements. *IEEE Trans. Computers*, **50**, 9, 891-905.

[9] C.F. Frasser, et al. 2021. Exploiting correlation in stochastic computing based deep neural networks. *Proc. DCIS,* 1-6.

[10] H. Abdellatef et al. 2018. Stochastic computing correlation utilization in convolutional neural network basic functions. *Telecomm. Comp. Electron. & Control*, **16**, 6, 2835-2843.

[11] A. Alaghi, and J.P. Hayes. 2013. Exploiting correlation in stochastic circuit design. *Proc. ICCD,* 39–46.

[12] S.A. Salehi. 2019. Low-correlation low-cost stochastic number generators for stochastic computing. *Proc GlobalSIP*, 1-5.

[13] T. Chen and J.P. Hayes. 2015. Equivalence among stochastic logic circuits and its application. *Proc. DAC,* 1-6.

[14] B. Arazi. 1981. On the synthesis of de Bruijn sequences. *Information and Control*, **49,** 2, 81-90.

[15] P.K. Gupta and R. Kumaresan. 1988. Binary multiplication with PN sequences. *IEEE Trans. ASSP*, **36**, 603-606.

[16] T.J. Baker and J.P. Hayes. 2020. "Bayesian accuracy analysis of stochastic circuits," *Proc. ICCAD*, 1-9.

[17] M.R. Alam, M.H. Najafi and N. TaheriNejad. 2022. "Sorting in memristive memory," J. *Emerg. Technol. Comput. Syst.*, to appear.

[18] M. van Daalen et al. 1993. Device for generating binary sequences for stochastic computing. *Electronic Letters*, **29**, 1, 80-81.

[19] E. Stine et al. 2007. FreePDK: An open-source variation-aware design kit. *Proc. Microelectronics Systems Education (MSE)*, 173-174.

[20] Z. Wang et al. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, **13**, 4, 600-612.