ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN2321-2152

www.ijmece .com

Vol 12, Issue.2 April 2024

The Analysis Of Stochastic Number Generator Using HDL Implementation

G.Krishnaveni 1, P.Priyanka 2, P.Sai Lakshmi 3, N.Ankitha 4
#1M. Tech, (Ph.D)
#2,#3,#4 Students, Dept of Electronics and Communication Engineering,BWEC, Bapatla,Andhra Pradesh,India

ABSTRACT_ For stochastic computing (SC) circuits to be accurate and area-efficient, stochastic number generators (SNGs) must be designed efficiently. SNGs based on linear feedback shift registers (LFSRs) are widely used in SC. On the other hand, we suggest a novel design strategy to minimize the size of SNGs: distributing among several SNGs a mix of negations and permutations of the output of a single LFSR. This method produces SC circuits with improved accuracy by providing SNGs with least average SC correlation (SCC) without requiring any extra hardware overhead... When a 10-bit LFSR is shared between two SNGs, our method produces stochastic bit streams with an average SCC that is 50% lower than the prior state-of-the-art work. The proposed design space for a $\boldsymbol{n}$-bit LFSR includes $\boldsymbol{n}! \times 2\ \boldsymbol{n}$ designs. Nevertheless, searching the entire area for the design with the least SCC becomes unfeasible when $\boldsymbol{n} > 7$. We offer an optimized search algorithm to tackle this problem. When $\boldsymbol{m} < \boldsymbol{n}$, our optimized search technique can locate a set of $\boldsymbol{m}$ distinct designs with least SCC values. This solves the issue of looking through the whole design area and makes it possible to explore design possibilities quickly.

Keywords—Stochastic number generator, Stochastic computing, Linear feedback shift

1.INTRODUCTION

Stochastic registering (SC) circuits perform calculations on bitstreams with diminished equipment intricacy [1]. Contrasted with customary double registering (BC), SC gives different benefits including diminished equipment intricacy and faulttolerant figuring. In light of these benefits, SC has been thought of as an option in contrast to BC in various applications, for example, picture handling, brain organizations, and computerized channels. A SC circuit utilizes stochastic number generators (SNGs) to change parallel numbers over completely to their comparing arbitrary bitstreams. In SC circuits, SNGs consume altogether more region contrasted with processing parts. Specifically, for applications, for example, picture handling and advanced channels, SC circuits require a few SNGs which consume around 80% of the complete region [4]. Moreover, the connection between's bitstreams produced by various SNGs in a SC circuit can amazingly lessen its computational precision. Subsequently, the plan of low-region SNGs that can produce low-associated bitstreams is fundamental for having region proficient and precise SC circuits. In this paper, we propose another plan space for low-region SNGs in view of straight criticism shift register (LFSR)- sharing procedure [4] with no equipment above. The plan space utilizes stages of a LFSR's results and their supplements to produce different bitstreams with least SC connection. For n = 10, the proposed SNGs create bitstreams with half less SC connection contrasted with earlier work with a similar equipment intricacy. We exhibit that applying the proposed strategy to plan SC circuits for computerized channels, edge recognition, and picture division applications works on their computational precision. In the following segment, we make sense of the inward circuit of SNG 7s, an action to assess SC connection for their bitstreams, and related earlier work. The proposed plan space for effective LFSR-sharing-based SNGs and a streamlined calculation for tracking down the plan for two SNGs with least SC connection. We sum up the calculation for multiple SNGs. We assess our methodology for a few applications lastly finish up the paper.

STOCHASTIC figuring (SC) has arisen as an offbeat strategy for performing calculations by rationale circuits [1]. Instead of performing calculation on deterministic double numbers, SC circuits are intended to handle irregular piece streams. The information and result are addressed by bit streams and their qualities are encoded as the probabilities of seeing 1's in the piece streams. Obviously, the qualities are restricted in the unit span [0, 1], since probabilities can't be past the unit stretch. Contrasted with deterministic twofold

processing, SC gives a few benefits, including decreased equipment intricacy and issue lenient registering. In light of these benefits, SC has been viewed as a fitting option in contrast to parallel registering in various applications, for example, low-thickness equality check (LDPC) disentangling [2], picture handling [3], brain networks [4], [5], and advanced channels [6], [7]. One principal benefit of SC is exceptionally low equipment intricacy could bring about cost-proficient processing circuits. The most well-known method for exhibiting the low equipment cost of SC is its execution of fundamental tasks, or at least, augmentation and expansion. Fig. 1(a) shows a straightforward AND door executing duplication in SC. For the AND entryway, the result is 1 just when the two data sources An and B are 1. Thusly, the likelihood of having 1 in the result bit stream is the duplication of the probabilities of having 1 in every one of the information bit streams, that is to say, P(C = 1) = P(A = 1)× P(B = 1), that is, c = a × b. Essentially, Fig. 1(b) shows a 2-input multiplexer figuring scaled expansion. For the multiplexer, yield C is 1 when S is 0 and An is 1 or when S is 1 and B is 1. Hence, P(C = 1) = (1 − P(S = 1)) × P(A = 1) + P(S = 1) × P(B = 1), that is, c = (1 − s) × a + s × b.

A stochastic number generator (SNG) is a fundamental piece of any SC circuit. A SC circuit utilizes SNGs to change over double numbers into their comparing irregular piece streams. They create arbitrary piece streams with probabilities of delivering 1's equivalent to their relating paired numbers. SNGs assume a focal part in the proficiency of a SC circuit for two reasons. In the first place, for SC circuits, the size of a SNG part is wonderful concerning the figuring part. This issue turns out to be more basic for applications with SC circuits that require numerous SNGs, for example, serious level advanced sifting and picture handling calculations. As a matter of fact, for a few SC plans, SNG circuits consume around 80% or even 90% of the complete region [8], [9]. Second, the nature of the irregular numbers created by SNGs can fundamentally influence the computational precision of the SC plans, and relationship among arbitrary piece streams is a wellspring of error in SC circuits. In this way, getting region effective and low-corresponded SNGs is a significant plan challenge for SC circuits.

Because of this test, the commitments of this article are the accompanying.

1) Presenting another stage based plan space for sharing an irregular number source (RNS) among a few SNGs. The plan space yields minimal expense and low-corresponded SNGs. Contrasted with SNGs with a similar equipment intricacy, the proposed SNGs produce irregular piece streams with lower cross connection.

2) Displaying the variety of SC connection for the proposed plan space and introducing a scanning calculation for tracking down the changes with least relationship.

2.LITERATURE SURVEY

2.1 B. R. Gaines, "Stochastic computing" AFIPS spring joint computer conference. ACM, pages149–156, 1967.

We give a short overview of stochastic computing (SC) and its uses. SC computes with randomized bit-streams that loosely resemble the neural spike trains of the brain. Its key feature is the use of low-cost and low-power logic elements to implement complex numerical operations in a highly error-tolerant fashion. These advantages must be weighed against SC's inherently slow computing speed and low precision. Although studied sporadically since its invention in the 1960s, SC has regained interest recently as potentially suited to some emerging nanotechnologies, and to applications such as ECC decoding and biomedical image processing. However, a number of major challenges must be overcome if this potential is to be fully realized.

2.2 P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, “Computation on Stochastic Bitstreams Digital Image Processing Case Studies,” IEEE Trans. VLSI Syst., vol. 22, no. 3, pp. 449–462, Mar. 2014.

Maintaining the reliability of integrated circuits as transistor sizes continue to shrink to nanoscale dimensions is a significant looming challenge for the industry. Computation on stochastic bit streams, which could replace conventional deterministic computation based on a binary radix, allows similar computation to be performed more reliably and often with less hardware area. Prior work discussed a variety of specific stochastic computational elements (SCEs) for applications such as artificial neural networks and control systems. Recently, very promising new SCEs have been developed based on finite-state machines (FSMs). In this paper, we introduce new SCEs based on FSMs for the task of digital image processing. We present five digital image processing algorithms as case studies of practical applications of the technique. We compare the error tolerance, hardware area, and latency of stochastic implementations to those of

conventional deterministic implementations using binary radix encoding. We also provide a rigorous analysis of a particular function, namely the stochastic linear gain function, which had only been validated experimentally in prior work.

2.3 H. Ichihara, and T. Sugino, "Compact and accurate digital filters based on stochastic computing," IEEE Transactions on Emerging Topics in Computing, 2019. Stochastic computing (SC), which is an approximate computation with probabilities, has attracted attention as an alternative to deterministic computing. In this paper, we discuss a design method for compact and accurate digital filters based on SC. Such filter designs are widely used for various purposes, such as image and signal processing and machine learning. Our design method involves two techniques. One is sharing random number sources with several stochastic number generators to reduce the areas required by these generators. Clarifying the influence of the correlation around multiplexers (MUXs) on computation accuracy and utilizing circular shifts of the output of random number sources, we can reduce the number of random number sources for a digital filter without losing accuracy. The other technique is to construct a MUX tree, which is the principal part of an SC-based filter. We formulate the correlation-induced errors produced by the MUX tree, and then propose an algorithm for constructing an optimum MUX tree to minimize the error. Experimental results show that the proposed design method can derive compact (approximately 70 percent area reduction) SC-based filters that retain high accuracy.

3.PROPOSED SYSTEM

We explain the proposed method for n = 4. We share the LFSR in Fig 1(a) between two comparators, CMP1 and CMP2, to build two SNGs, SNG1 and SNG2. Notice that, with no additional hardware, for each FF of the LFSR the output and its complement are both available to be used. In our approach, we connect the outputs of $L1$, $L2$, $L3$, and $L4$ to the $r1$, $r2$, $r3$, and $r4$ inputs of CMP1, respectively, to form SNG1. We then choose four signals from the eight available signals (i.e., $L1$, $L2$, $L3$, $L4$, and their complements), and connect a permutation of the chosen four signals to the four inputs of CMP2 for SNG2. To find the connection pattern that provides minimum $SCCavg$, we need to search all eligible combinations of four signals coming from an LFSR and their permutations. Only combinations that can generate correct bitstreams (i.e. bitstreams with correct number of 1's at the output of an SNG) are eligible. For n = 4, such combinations should produce 15 (i.e., 2 n − 1) different output numbers in each period. This requirement leads to the restriction that only one signal from every FF can be included in each combination: either an FF's output or its complement (not both). For example, when we consider the combination of $L1$ to $L4$, and not any of their complements), a permutation of numbers between 1 and 15 is fed to the connected CMP circuit and number 0 is excluded because it is not generated by the LFSR. Similarly, when we consider the combination of $L2$ to $L4$ and the complement of $L1$, (i.e., "$L4L3L2\bar{L1}$"), a permutation of numbers 0 and 2 to 15 is fed to the connected CMP component. That is, from numbers between 0 and 15, number 1 is excluded. It is easy to show that a different number is excluded for every other combination.

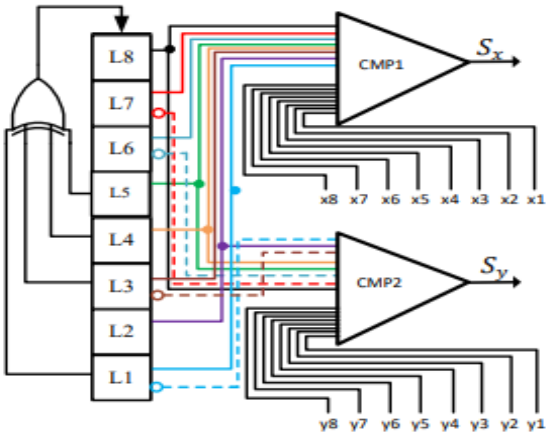
In general, for an n-bit LFSR, there are 2 n different combinations and for each combination, a number between 0 and 2 n − 1 is excluded while a permutation of other numbers is fed to the CMP component. Since there are 2 n different combinations for an n-bit LFSR and n! permutations for each combination, the proposed design space has 2 n × n! eligible designs, that is 2 n times larger than the design space proposed in [6]. The new design space can achieve $SCCavg$ values lower than the minimum $SCCavg$ from [6]. B. Exploring the Design Space To find the design with minimum $SCCavg$, a straightforward approach is to calculate $SCCavg$ values for all possible designs and choose the minimum value. However, for this approach computational complexity and required memory increase exponentially as n becomes larger. In fact, for n > 7, an exhaustive search for scanning the whole design space for finding the minimum $SCCavg$ is intractable. To overcome this problem, we propose a new approach. We evaluate all the possible designs for n = 3 to n = 6 and extract the pattern of $SCCavg$. Fig. 2 shows the behavior of $SCCavg$ for n = 3 to n = 6. The x axis shows all permutations for all combinations. From left to right for this axis, the combinations are sorted such that their excluded numbers increase from 0 to 2 n − 1. For each combination, permutations are in the reverse lexicographic order. The red dots in Fig. 2 mark the $SCCavg$ values for the first permutation of each combination. As we can see, for all combinations, the first permutation (in the reverse lexicographic order) provides a lower $SCCavg$ value compared to the other permutations. Thus, the design with minimum $SCCavg$ is the first permutation of one of the combinations for an LFSR's outputs. In fact, this observation is supported by [7], where it is shown that for a sequence, the reverse permutation

ISSN2321-2152

www.ijmece .com

Vol 12, Issue.2 April 2024

(i.e., the first in reverse lexicographic order) has the greatest distance (lowest cross correlation). We optimize our search algorithm based on this observation by calculating $SCCavg$ only for the first permutations and finding their minimum. This optimization can significantly reduce the computational complexity of our search algorithm; instead of 2 $n \times n!$ designs, the algorithm searches 2 n designs. Combination k means that the inverted outputs of flip-flops are used for bits which are '1' in the binary representation of k. In other words, we can imagine the output of LFSR is XORed with the binary value of k before it is used. For example, if k = 2 and n = 4, combination k means "$L4L3L2\bar{L}1$ " is used. For an n-bit LFSR, the output for combination k and output for combination (2 n − 1)− k are complement of each other and their sum is 2 n −1. For example, if n = 4, outputs for combination 2 and 13 are complement and their sum is 15. Therefore, for an n-bit LFSR the pattern of generated numbers for combination (2 n − 1)− k is the same as combination k, but flipped. This means that the generated bitstreams from two SNGs using these combinations are inverted of each other and the $SCCavg$ value for combination k and combination (2 n − 1)− k is the same. This is because the average of absolute values of (, s ′ j), for $i,j \in$ {0,1, . . . , (2 n − 1)}, are used for the calculation of $SCCavg$ in Equation (2). Fig. 2 shows that the $SCCavg$ values of the first permutations for combination k and combination (2 n − 1)− k are the same. Therefore, there are two combinations with minimum $SCCavg$ and the algorithm can find minimum $SCCavg$ values by searching only one half of the design space. This optimization speeds up the searching algorithm twice. The pseudocode in Algorithm 1 represents our optimized searching algorithm. In this algorithm, the direct connection of an LFSR's output, i.e., "Ln … $L2L1$ " is considered as SNG1. In a for-loop, different combinations for the first permutation of the LFSR are considered as SNG2. The algorithm uses "$L1L2$ … Ln" as the first permutation. In order to search all 2 n possible combinations, the algorithm performs bitwise XOR between the first permutation and the n-bit binary representation of k, where k varies from 0 to 2 n −1 inside the for-loop. Notice that due to the symmetry of $SCCavg$ (our second observation), the optimized algorithm searches only the first half of the design space and k varies from 0 to (2 n−1 −1). In every iteration of the for-loop, if the $SCCavg$ between SNG1 and SNG2 is less than $SCCmin$, the algorithm updates $SCCmin$by the $SCCavg$. The algorithm finds the



combination index (i.e., $kmin$) for the minimum $SCCavg$ (i.e., $SCCmin$) in the first half of the design space. This index is returned as 1st_kmin. Due to the existent symmetry, the combination index for the minimum in the second half (i.e., 2nd_kmin) is obtained by bitwise negation of the binary representation of $kmin$. Algorithm 1 returns the minimum of $SCCavg$ and its combination indices as $SCCmin$, 1st_kmin, and 2nd_kmin, respectively. Table II lists $SCCmin$, 1st_kmin, and 2nd_kmin for n = 4 to n = 8. Using the combination indices obtained from Algorithm 1, we design the connection circuit for the LFSRsharing SNGs. Notice that having two different possible combinations for $SCCmin$ means that there are two different designs that achieve minimum $SCCavg$. Fig. 3 shows both possible proposed circuits for n = 4, and one of the two possible circuits for n = 8.

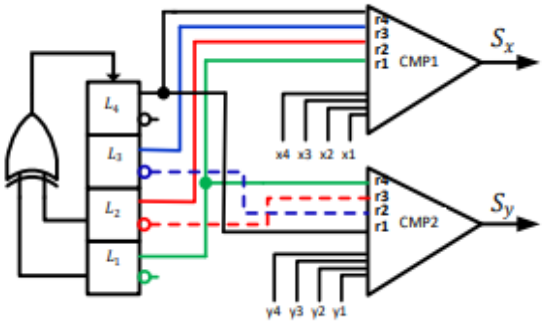


FIG. Proposed structure for sharing an LFSR with two SNGs based on output permutation(n=4 & n=8)

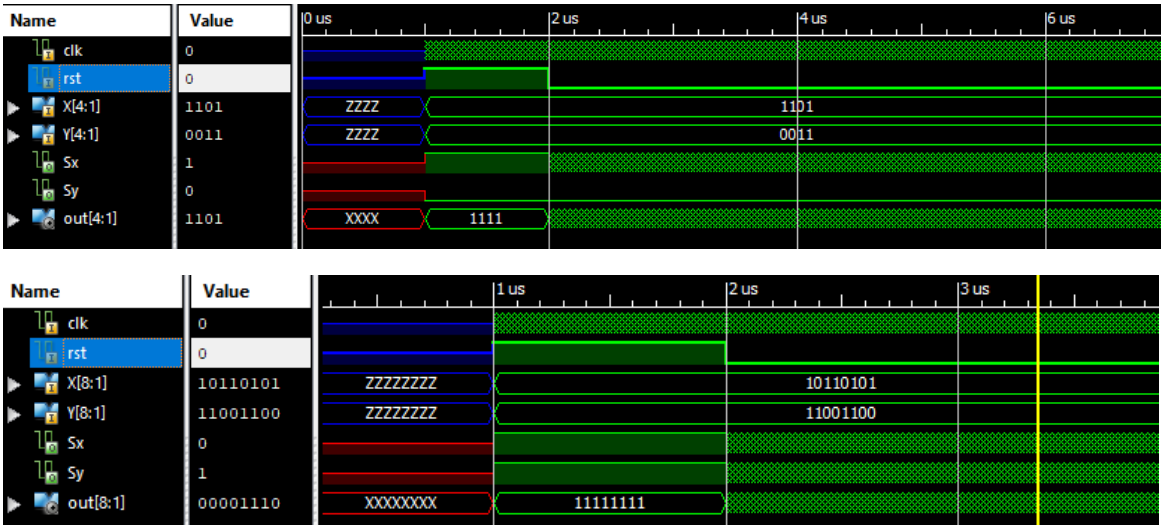
4.RESULTS AND DISCUSSION

ISSN2321-2152

www.ijmece .com

Vol 12, Issue.2 April 2024

4.1 Simulation Results of LFSR_PUF:



Sim.Result. N=4 & N=8

Area:

```
Slice Logic Utilization:
 Number of Slice Registers:                 4  out of  126800     0%
 Number of Slice LUTs:                      5  out of  63400      0%
    Number used as Logic:                   5  out of  63400      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:        9
   Number with an unused Flip Flop:         5  out of       9     55%
   Number with an unused LUT:               4  out of       9     44%
   Number of fully used LUT-FF pairs:       0  out of       9      0%
   Number of unique control sets:           1

IO Utilization:
 Number of IOs:                            12
 Number of bonded IOBs:                    12  out of     210      5%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                  1  out of      32      3%
```

N=4

```
Slice Logic Utilization:
 Number of Slice Registers:                 8  out of  126800     0%
 Number of Slice LUTs:                      9  out of  63400      0%
    Number used as Logic:                   9  out of  63400      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:       11
   Number with an unused Flip Flop:         3  out of      11     27%
   Number with an unused LUT:               2  out of      11     18%
   Number of fully used LUT-FF pairs:       6  out of      11     54%
   Number of unique control sets:           1

IO Utilization:
 Number of IOs:                            20
 Number of bonded IOBs:                    20  out of     210      9%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                  1  out of      32      3%
```

N=8

Delay:

ISSN2321-2152

www.ijmece .com

Vol 12, Issue.2 April 2024

```
Delay:                  1.650ns (Levels of Logic = 4)
  Source:               X<2> (PAD)
  Destination:          Sx (PAD)

  Data Path: X<2> to Sx
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             1    0.001   0.649  X_2_IBUF (X_2_IBUF)
     LUT6:I2->O            1    0.105   0.451  D2/S<4>1 (D2/S<4>)
     LUT3:I1->O            1    0.105   0.339  D2/S<4>2 (Sx_OBUF)
     OBUF:I->O                  0.000          Sx_OBUF (Sx)
    ----------------------------------------
    Total                       1.650ns (0.211ns logic, 1.439ns route)
                                        (12.8% logic, 87.2% route)
```

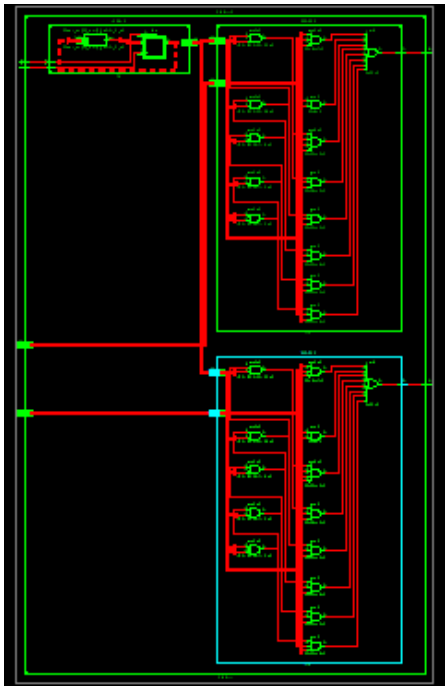
N=4

```
Delay:                  2.355ns (Levels of Logic = 5)
  Source:               X<2> (PAD)
  Destination:          Sx (PAD)

  Data Path: X<2> to Sx
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             1    0.001   0.649  X_2_IBUF (X_2_IBUF)
     LUT6:I2->O            1    0.105   0.451  D2/S<10>2 (D2/S<10>1)
     LUT6:I4->O            1    0.105   0.599  D2/S<10>3 (D2/S<10>2)
     LUT6:I3->O            1    0.105   0.339  D2/S<10>4 (Sx_OBUF)
     OBUF:I->O                  0.000          Sx_OBUF (Sx)
    ----------------------------------------
    Total                       2.355ns (0.316ns logic, 2.039ns route)
                                        (13.4% logic, 86.6% route)
```

N=8

RTL Schematic:

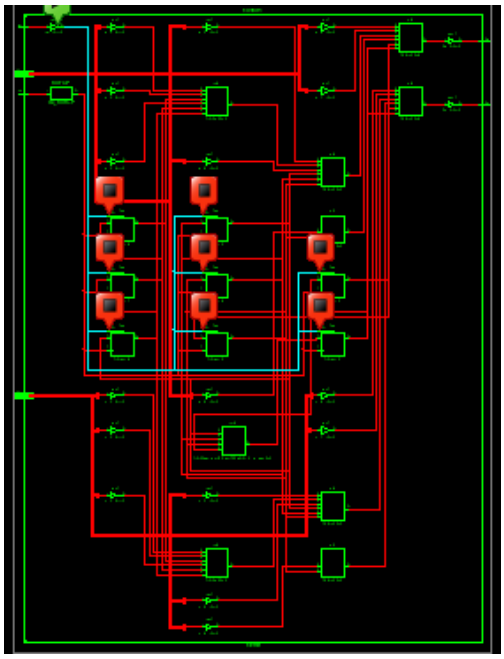


Technology Schematic:

ISSN2321-2152

www.ijmece .com

Vol 12, Issue.2 April 2024



5.CONCLUSION

We suggest an LFSR-sharing design space for low correlated and area-efficient SNG circuits. The permutation of every combination of the output bits of an LFSR and their logical complements makes up the design space. We examine how $SCCag$ behaves in this design space. Our findings demonstrate that the least amount of cross correlation for an LFSR's output can be obtained by combining two of its initial permutation in the opposite lexicographic order. Our approach produces SNGs with lower cross correlation values when compared to previous work with the same hardware complexity, i.e., the circular shifting and permutation-only methods.

We provide an efficient search strategy for locating LFSR-sharing-based SNG designs with the least amount of cross correlation, given the large expansion in the design space throughout previous work. In comparison to previous methods, we gain greater computational accuracy with reduced hardware complexity when we use the suggested SNGs in the SC-based implementation of many applications, as demonstrated by the results. Even though we were able to increase the search algorithm's speed inside the suggested design space, there is still room for improvement.

REFERENCES:

[1] B. R. Gaines, "Stochastic computing" AFIPS spring joint computer conference. ACM, pages149–156, 1967.

[2] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, “Computation on Stochastic Bitstreams Digital Image Processing Case Studies,” IEEE Trans. VLSI Syst., vol. 22, no. 3, pp. 449–462, Mar. 2014.

[3] H. Ichihara, and T. Sugino, “Compact and accurate digital filters based on stochastic computing,” IEEE Transactions on Emerging Topics in Computing, 2019.

[4] H. Ichihara, "Compact and accurate stochastic circuits with shared random number sources." IEEE 32nd International Conference on Computer Design (ICCD), pp. 361-366, 2014.

[5] A. Alaghi, and J.P. Hayes, “Exploiting Correlation in Stochastic Circuit Design,” Proc. Intl Conf. on Computer Design (ICCD), pp. 39–46, Oct. 2013.

[6] S. A. Salehi, "Low-Cost Stochastic Number Generators for Stochastic Computing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 4, pp. 992-1001, April 2020, doi: 10.1109/TVLSI.2019.2963678.

[7] Siegel, S. and Castellan, N. J. Nonparametric Statistics for the Behavioral Sciences. McGraw-Hill, 1988

[8] P. Eades, M. Hickey and R. Read, Some Hamilton Paths and a Minimal Change Algorithm, Journal of the ACM, 31 (1984) 19–29.

[9] T. Hough and F. Ruskey, An Efficient Implementation of the Eades, Hickey, Read Adjacent Interchange Combination Generation

Algorithm, Journal of Combinatorial Mathematics and Combinatorial Computing, 4 (1988), 79–86.

[10] Sevaux, M. and Sorensen, K. “Permutation distance measures for memetic algorithms with population management.” Metaheuristics International Conference, 2005.

[11] Ronald, S. “More distance functions for order-based encodings.” Proc IEEE Conference on Evolutionary Computation, pages 558–563, 1998.

[12] P. Li, and D. J. Lilja, “Using Stochastic Computing to Implement Digital Image Processing Algorithms,” Proc. ICCD, pp. 154–161, 2011.