An Area and Power Efficient Stochastic Number Generator for Bayesian Sensor Fusion Circuits

Jérémy Belot, Abdelkarim Cherkaoui, Raphael Laurent, Laurent Fesquet

An Area and Power Efficient Stochastic Number Generator for Bayesian Sensor Fusion Circuits

Jérémy BELOT*†, Abdelkarim CHERKAOUI†, Raphaël LAURENT† and Laurent FESQUET*
Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, F-38000 Grenoble, France
†HawAI.tech, F-38000 Grenoble, France

***Abstract*—In recent years, stochastic computing became popular in Bayesian circuits implementation because it enables compact and low power architectures. These architectures use Stochastic Number Generators (SNGs) that encode data in random bit-streams. SNGs are composed of Random Number Generators (RNGs) which contribute significantly to the circuit area and power consumption: according to our measurements, up to 29% of the area and 85% of the circuit power consumption, excluding memories. In this paper, we compare SNG implementations in terms of accuracy, area and power consumption. Furthermore, we propose a new SNG architecture that uses a single RNG for the whole design in order to generate the required stochastic bit-streams. The proposed architecture allows to save up to 11% of area and 58% of power consumption compared to the state of the art, with no significant accuracy loss.**

I. INTRODUCTION

In the last few years, the advent of the IoT (Internet of Things) and the need for low-power integrated circuits sparked the interest in new computing paradigms. In many applications, such as autonomous systems powered by a battery, computing speed is a secondary requirement compared to energy efficiency. One of the approaches for designing such low-power circuits, called Stochastic Computing, is a non-standard computing method that encodes data using the probability of the bits at '1' occurring in a random bit-stream. This approach was proposed more than 50 years ago [1]. Although it was quickly abandoned in favor of a faster classical binary representation, this approach regained interest due to the increase of power and surface constraints for applications where speed is not of primary importance.

Its principle is to encode data in a stochastic bit-stream where the encoded value is related to the ratio r between the number of zeros and ones in the bit-stream. For the unipolar stochastic representation, the encoded value corresponds directly to this ratio r. This approach is particularly interesting in that it only requires a small amount of logic resources to process computations which are demanding in standard representation, such as multiplication. Indeed, if we consider two independent unipolar stochastic bit-streams, the multiplication of their encoded values is done by a simple 2-input AND gate. For example, let $s_0 = 0101100100100011$ and $s_1 = 0110110010111001$ be two unipolar stochastic bit-streams, respectively having the values $P(s_0) = 7/16$ and $P(s_1) = 9/16$. Their AND gated bit-stream is $s_0 \wedge s_1 = 0100100000100001$ and $P(s_0 \wedge s_1) = 4/16 \simeq 63/256$.

Apart from area reduction, other advantages of the stochastic representation are an increased robustness (bit flips have a lower impact on the encoded value than in base-2 representation) and progressive precision (the longer is the bit-stream, the higher is the precision). On the one hand, the increased robustness allows to reduce the circuit voltage, which reduces its power consumption. On the other hand, precision can be adapted dynamically without major hardware modification. This can be exploited for reducing the power consumption by doing a fast evaluation with low energy when the application allows it.

In recent years, architectures based on stochastic computing have been successfully used to implement Bayesian inference in sensor fusion models and temporal filters [2]. Such circuits use Stochastic Number Generators (SNGs) to encode probability values in stochastic bit-streams. These SNGs are critical in the implementation of stochastic circuits: their bit-streams must be pairwise independent as well as non biased in order to maximize the computation precision for a given bit-stream length. Moreover, their contribution to the circuit power consumption and area is significant and a number of previous works have focused on reducing their impact on the circuit with methods sharing common resources [3]–[7].

In this paper, we consider a Bayesian sensor fusion problem implemented in a dedicated stochastic circuit [8], for which we compare, in terms of area, power consumption, and computation accuracy different SNG implementations sharing hardware resources. We introduce a new solution of Random Number Generator (RNG) sharing over a large amount of stochastic bit-streams, which significantly enhances the circuit area and power efficiency.

II. RELATED WORKS

A. Stochastic circuits for Bayesian sensor fusion

Sensor fusion circuits combine data from several sensors to synthesize information on a state variable or to reduce sensor noise and enhance measurements accuracy. For many AI systems where most of the computation power is in the cloud, they provide computing capability at the edge to transmit only essential data to remote servers and save precious communication bandwidth.

In this context, Bayesian inference provides a theoretical framework that allows building models for sensor fusion (see for instance [8], [9]). In addition to fusing sensor data, such explainable models allow sensor diagnosis by inferring values

for one sensor knowing the output states and the other sensors data.

In a Bayesian sensor fusion model, a state variable S is usually inferred from n independent sensor readings K_i using sensor models $P(K_i|S)$ thanks to Bayes' theorem:

$$P(S|\wedge_{i=1}^{n} K_i) \propto P(S)\prod_{i=1}^{n} P(K_i|S) \quad (1)$$

The probability distribution $P(S)$, usually called *prior*, encodes *a priori* knowledge about the state S. The conditional probability distribution $P(K_i|S)$ on sensor reading K_i knowing the state S, called a *likelihood*, encodes knowledge related to a physical model of the sensor.

When S is a discrete variable with cardinal m, and given a set $\{k_1, \ldots, k_n\}$ of acquired sensor samples, for each $1 \leq j \leq m$, Eq. 1 becomes:

$$P(S = s_j|\wedge_{i=1}^{n} K_i = k_i) \propto P(S = s_j)\prod_{i=1}^{n} P(K_i = k_i|S = s_j) \quad (2)$$

As the inference of Eq. 2 only involves computing products, it may be efficiently implemented as a multiplication matrix of $1+n$ columns (for the *prior* and the *likelihoods*) computing in parallel on m rows the posterior distribution for all values of S [8]. Consequently, the stochastic circuit dedicated to Bayesian sensor fusion, shown in Fig. 1, is comprised of the following blocks:

- Memories that store sensor models $P(K_i|S = s_j)$,
- Stochastic Number Generators, which are comprised of RNGs and Probability Converter Circuits (PCC), usually a comparator, that convert *likelihood* values into stochastic bit-streams,
- AND gates that perform the multiplication of *likelihoods* and *priors*,
- and finally, counters that convert the resulting stochastic bit-steams into standard binary representations.

When a new sample k_i is acquired, the *likelihood* $P(K_i = k_i|S = s_j)$ is read from memory. Then, this value is compared to a random number in the PCC to generate a bit at '1' or at '0' with probability $P(K_i = k_i|S = s_j)$ according to a Bernoulli trial. Since rows are independent, it is possible to share one RNG per column. Thus, only $n + 1$ RNGs are needed to generate $m * (n + 1)$ stochastic bit-streams. These RNGs are usually implemented using Linear-Feedback Shift Registers (LFSR). Although LFSRs are pseudo-deterministic RNGs, they are popular in these implementations due to their small area and power consumption while having good bias properties.

B. Methods to reduce SNG area and power consumption

Previous works proposed solutions to make the SNGs more power and area efficient, mainly by sharing logical resources.

First, in [3], Chen and Hayes developed the concept of stochastic bit-stream *isolation*, aiming at reducing correlation between bit-streams while sharing a SNG over several of them. To do so, a so called *isolator* is implemented, which actually corresponds to a register, to shift the bit-streams in time and, thus, reduce their cross-correlation. However, this single shared RNG needs to have low auto-correlation to prevent a decrease in computation accuracy. LFSRs do not satisfy this requirement, and more sophisticated RNGs have higher area and power consumption.

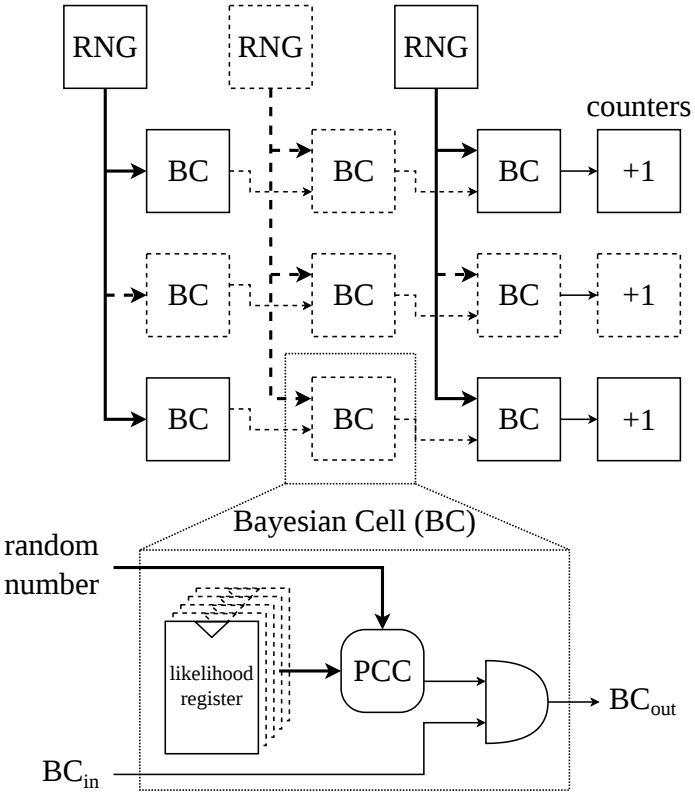


Figure 1: A stochastic circuit architecture for Bayesian sensor fusion: a matrix of Bayesian cells (BC).

In [4], Neugebauer *et. al.* propose to use an RNG based on a substitution box (S-Box), the SBoNG. These RNGs are larger than LFSRs, but they provide bit-streams with lower auto-correlation. They can therefore be shared over several SNGs using the *isolators* mentioned above. This method allows to save up to 20% of area compared to the LFSR implementation when sharing one RNG over 100 SNGs.

Other works focus on the PCC part of the SNG. [5] describes an efficient PCC, based on the Weighted Binary Generator (WBG). The WBG in itself is larger than a comparator-based PCC. However, a part of the WBG, the Weight Generator (WG), can be shared over one whole column since it does not depend on the *likelihood* to be encoded, as shown in Fig. 2a. Consequently, area and power consumption can be saved this way. On the contrary, a comparator cannot be split with a common part sharing resources, making mandatory the whole PCC implementation for each cell.

Finally, recent works proposed to share the same random numbers, even with a simple LFSR, to generate several

stochastic bit-streams by using bit permutations between the RNG and the PCC. While [6] proposes to use circular shifts on the same random numbers, in [7], Salehi defines an algorithm to determine the set of k permutations over a n-bit random number that minimize pairwise correlations between the stochastic bit-streams. This method achieves less correlated stochastic bit-streams than in [6], and saves up to 50% area and power consumption of the SNGs, while maintaining equivalent computation accuracy.

In conclusion, wire permutations of [7] allow to share one RNG over several SNGs. However, if the RNG is shared over more than two SNGs, accuracy starts dropping. When using *isolators* as in [3], one RNG can be shared over more than two SNGs without accuracy loss, but the hardware overhead can be significant. In the next section, we propose a new method for sharing one RNG over several SNGs, with minimal accuracy loss, and minimal hardware overhead.

III. A SNG BASED ON SHIFT REGISTER ISOLATOR

In this proposition, we introduce two improvements to the method proposed in [3]:

- Firstly, we insert *isolators* at the output of one RNG instead of at the SNGs outputs. We therefore shift random numbers instead of bit-streams. This RNG is shared over all the SNGs. This way, we limit the hardware overhead when adding new rows and columns to the design.
- Secondly, as in [5], we reduce the hardware cost of the PCCs by mutualizing a part of the WBG, the WG, and attaching it to the RNG, before the *isolators*. This amounts to factorising a part of the hardware resources which is used in every Bayesian cell.

Fig. 2a shows the architecture of the WBG. It can be separated into two parts. Firstly, the WG associates a weight to each output bit. Then, the Probability Encoder generates the stochastic bit-stream according to the associated *likelihood* or *prior*.

The architecture proposed in this paper, called Shift Register Isolator (SRI), is depicted in Fig. 2b. Only one RNG is used in this design. The WG is connected to the output of the RNG. Its output is then shifted using a register pipeline. Finally, the pipelined random numbers are connected to the encoders for each Bayesian cell.

Contrarily to [7], the SRI can share one RNG over all SNGs if its output has sufficiently low auto-correlation. Moreover, one WG is shared over all the columns, which is not possible when permutations are used. Indeed, permutations have to be applied before the WG.

Compared to the method proposed in [3], AND gates are no longer separated with registers. Therefore, they are optimized during synthesis into one larger AND gate per row. Moreover, when adding new columns to the design, the number of additional registers is no longer dependent on the number of lines, but only on the RNG output data width which is fixed (it is associated to the data coding precision). Consequently, this method saves area and power consumption in designs with a large number of rows and columns. For example, in a design

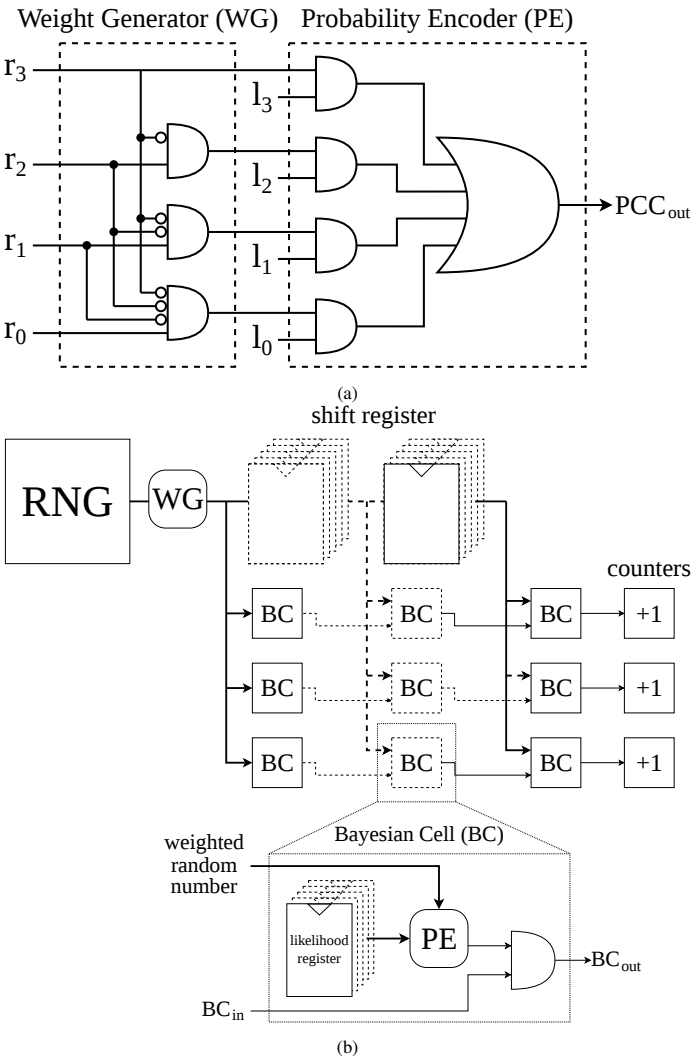


Figure 2: Proposed Shift Register Isolator implementation. (a) WBG decomposition adapted from [5] for 4-bits random number ($r_{i,i\in[0;3]}$) and 4-bits *likelihood* ($l_{i,i\in[0;3]}$). (b) Global architecture.

with 128 rows, if data is encoded with 8 bits, each added column involves adding 8 registers instead of 128 registers for the method of [3].

As in [3], using *isolators* introduces a latency corresponding to the number of columns. However, this latency is negligible compared to the computing time associated with the bit-stream length.

IV. ACCURACY, AREA AND POWER COMPARISONS OF SNG ARCHITECTURES

This section compares different SNG implementations, including the proposition of Sec. III, in terms of accuracy, area and power consumption.

A. SNG quality metrics

In this paper, we use two complementary metrics for computation accuracy: the Stochastic Computing Correlation

quantifies at the component level the RNG capacity to generate uncorrelated bit-streams, and the Kullback-Leibler Divergence measures at the application level the quality of the output probability distribution.

1) Stochastic Computing Correlation: [10] introduces a metric to measure the correlation between two stochastic bit-streams X and Y, the Stochastic Computing Correlation (SCC), defined by Eq. 3:

$$SCC(X,Y) = \begin{cases} 0 & \text{if } \delta(X,Y) = 0 \\ \frac{\delta(X,Y)}{\min(p_X,p_Y) - p_X.p_Y} & \text{if } \delta(X,Y) > 0 \\ \frac{\delta(X,Y)}{p_X.p_Y - \max(p_X+p_Y-1,0)} & \text{if } \delta(X,Y) < 0 \end{cases} \tag{3}$$

where p_X and p_Y are respectively the encoded values of the bit-streams X and Y, $p_{X\wedge Y}$ is the encoded value of the bit-stream at the output of the AND gate with X and Y as inputs, and $\delta(X,Y) = p_{X\wedge Y} - p_X.p_Y$. The SCC measure varies in the $[-1;1]$ segment. SCC $= 0$ means that the two bit-streams are uncorrelated, SCC $= 1$ means that the bit-streams are equal, and SCC $= -1$ means that they are opposite, that is, each bit of X at '0' corresponds to a '1' for Y and vice versa.

To compute the correlation between two SNGs S and S' as defined in [6], we perform the SCC average on all possible values of X and Y:

$$SCC_{\text{avg}}(S,S') = \sum_{i=0}^{2^s-1} \sum_{j=0}^{2^s-1} \frac{|SCC(S_i, S'_j)|}{(2^s \times 2^s)} \tag{4}$$

where s is the data precision, and S_i and S'_j are the generated bit-streams with the SNG S (respectively S') and with the encoded value $i/2^s$ (respectively $j/2^s$). Then, SCC_{avg} is always positive, and the lower it is, the less correlated are the SNGs.

To measure the stochastic quality of a set of n SNGs, we therefore perform the average over the $\binom{n}{2}$ pairwise SCC averages.

2) Kullback-Leibler Divergence: Another way to measure the quality of SNGs in this application is to compare the resulting distribution of the stochastic circuit P_{stoc} with a reference distribution P_{float} computed with floating point multiplications. For this purpose, we use the Kullback-Leibler Divergence (KLD), which provides a measure of distance between probability distributions which is grounded in information theory:

$$D_{\text{KL}}(P_{\text{stoc}} \| P_{\text{float}}) = \sum_{j=1}^{m} P_{\text{stoc}}(j) \log \frac{P_{\text{stoc}}(j)}{P_{\text{float}}(j)} \tag{5}$$

where $P_{stoc}(j)$ (respectively $P_{float}(j)$) corresponds to the result of inference $P(S = s_j | \wedge_{i=1}^{n} K_i)$ computed in stochastic (respectively floating point) representation.

The KLD is always positive. The lower it is, the closer are the output and reference distributions, so the better is the set of SNGs.

B. Protocol

This study is divided into two parts. Firstly, a comparison in terms of correlation and accuracy is realized at software level, by simulating RNGs and architecture behaviors in C language. Secondly, a comparison in terms of area and power is done at hardware level by implementing these architectures in a STMicroelectronics 65 nm CMOS technology and using retro annotated simulations.

The studied RNGs architectures are the following:

- 8-bit, 16-bit and 32-bit LFSR,
- 8-bit, 16-bit and 32-bit SBoNG of [4],
- 16-bit and 32-bit Xoroshiro,
- STRNG, a True Random Number Generator (TRNG) based on a Self-Timed Ring, introduced in [11], and simulated in software via a set of recorded output data.

Note that simple RNGs such as LFSRs have highly auto-correlated outputs, while the STRNG has low to nonexistant auto-correlation at its output.

These RNG implementations are combined with different PCCs (comparator or WBG), and with or without RNG sharing using the permutation method of [7], or our proposed SRI.

We set the problem dimensions as follows:

- *likelihoods* are encoded on 8 bits and so are the random numbers (which correspond to the 8 least significant bits of the RNG outputs),
- the counters have a size of 8 bits,
- the number of columns (number of sensors + 1) is fixed to 8 for the software simulation and ranges from 8 to 16 for the hardware comparison,
- the discretization of the state variable S (number of rows) varies in the set $\{8, 16, 32, 64\}$ for the hardware comparison, and is chosen as large as possible for the software comparison in order to get stable results in a reasonable computation time,
- the bit-stream lengths vary exponentially from 2^3 to 2^{23} cycles.

To put these numbers into perspective, let us note that the sound source localisation application presented in [9] implements Bayesian sensor fusion with 4096 lines and 104 columns (grouped in slices of 10 columns) and considers typical bit-stream lengths varying between 2^{12} and 2^{23}.

For a fair comparison, we perform beforehand a study on RNG seeds, similar to the one developed in [12], looking for those minimizing the pairwise SCC_{avg}. This study is carried out in an exhaustive manner for the 8-bit LFSR, but the problem quickly becomes intractable with the growing size of RNGs. Therefore, we use Monte Carlo samples to compute the pairwise SCC_{avg} with a set of randomly generated seeds N_{trials} times (with $N_{trials} \geq 1000$), and register the ones with the lowest pairwise SCC_{avg}. Our study shows that optimized seeding reduces KLD up to 30% compared to a fully random seeding, which is significant and has to be properly studied. Notice that it is not in the scope of this paper.

Likewise, computing SCC_{avg} is intractable over the 8 columns, so it is also done with randomly generated *likeli-*

hoods values over a large number of rows (around 10000). KLD measurements are done at the same time with this same set of random data.

C. Computation accuracy results

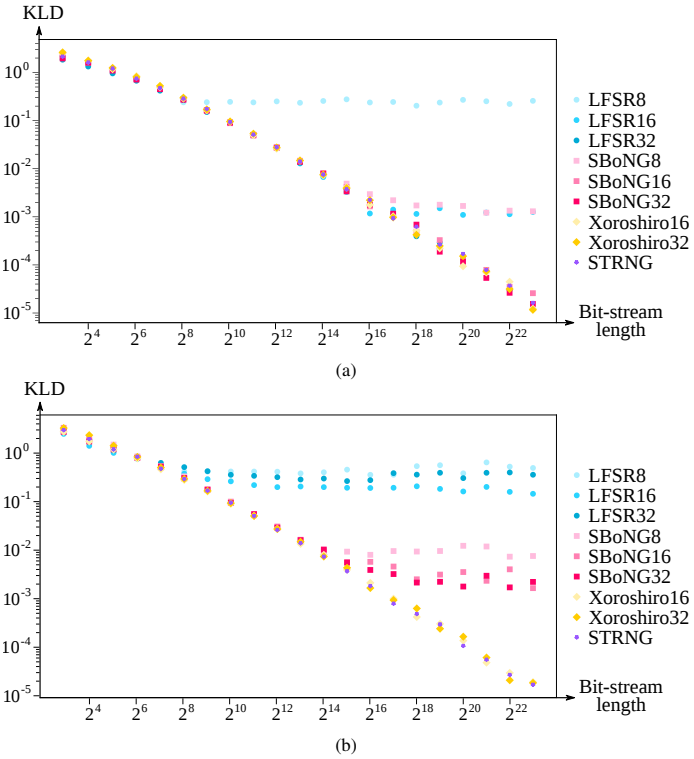


Figure 3: KLD measurement for different RNGs as a function of the bit-stream length. (a) Bare implementation. (b) With our proposed SRI.

Fig. 3a shows the impact of RNGs on the KLD in a design without the SNG optimizations of Sec. II-B and Sec. III. Independently of the chosen RNG, if the bit-stream length is shorter than the RNG period ($2^s - 1$ for LFSR and 2^{2s} for SBoNG and Xoroshiro, s being the RNG output precision), all studied RNGs lead to similar KLD (even the TRNG). In other words, in terms of SC multiplication accuracy, even RNGs with high auto-correlation such as LFSRs tend to behave like ideal RNGs as long as the stochastic bit-stream is shorter than the period. Beyond this period, the KLD curve reaches a plateau, and the SC precision can no longer be enhanced with the bit-stream length.

Fig. 3b shows the impact of the proposed SRI on the KLD. Simple, high auto-correlated RNGs such as LFSRs are not compatible with our SRI proposition and lead to significant accuracy loss due to the use of *isolation*. Accuracy is also impacted when using SBoNGs but if the stochastic bit-stream length does not exceed 2^{13} cycles, the KLD loss remains negligible. For RNGs with low auto-correlation, such as Xoroshiro or STRNG, *isolation* does not affect the KLD.

Table I provides more detailed KLD and SCC measurements for different SNG configurations, including SRI architecture

Sharing version	RNG	Error measurement after 2^{13} cycles ($\times 10^{-2}$)	
		SCC	KLD
Bare	LFSR-16	3.71	1.50
	LFSR-32	3.83	1.47
	SBoNG-8	3.52	1.35
	Xoroshiro-16	3.44	1.30
	STRNG	3.67	1.35
WBG	LFSR-16	3.08	1.28
	LFSR-32	3.37	1.33
	SBoNG-8	3.45	1.47
	Xoroshiro-16	3.34	1.47
	STRNG	3.47	1.41
Inverse Permutation [7] + *WBG*	*LFSR-16*	*4.26*	*1.42*
	LFSR-32	*4.35*	*1.47*
	STRNG	4.30	1.46
Isolator + WBG	SBoNG-8	4.72	1.65
	Xoroshiro-16	3.52	1.41
	STRNG	3.31	1.40
Shift Register Isolators	**SBoNG-8**	**4.67**	**1.64**
	Xoroshiro-16	**3.20**	**1.35**
	STRNG	3.26	1.40

Table I: Comparison of the accuracy of SNGs, measured at the component level by the SCC and at the application level by the KLD.

proposed in this paper for a bit-stream length of 2^{13}. We can see that the use of WBG does not decrease computation accuracy. Furthermore, the inverse permutation of [7], which, to the best of our knowledge, corresponds to the *state of the art*, shows almost the same ideal KLD and SCC as the bare implementation with the use of any studied RNG. Until this breaking point of 2^{13}, the proposed SRI architecture also achieves equivalent ideal accuracy for low auto-correlated RNG such as SBoNG and Xoroshiro.

In conclusion, when using pseudo-deterministic RNGs, computation accuracy increases with the bit-stream length as long as it is shorter than the RNG period. This result tends to support the exclusive use of simple RNGs such as LFSRs for SC since larger RNGs such as Xoroshiro or the STRNG do not significantly enhance accuracy. However, to allow the use of our proposed SRI, the RNG output must have a low auto-correlation. Thus, these more sophisticated RNGs can paradoxically allow a more compact design (cf Sec. IV-D) without accuracy loss. The proposed SRI architecture with both Xoroshiro-16 and SBoNG-8 satisfies these requirements and shows similar accuracy than state of the art implementations.

For a given application, the required bit-stream length L depends on the number of sensors and on the target accuracy. According to our analysis and to Fig. 3, for a given bit-stream, some architectures are more accurate than others. Consequently, L is set depending on the application, and the choice of an adequate architecture is set depending on L. In the following, we compare architectures with similar accuracy for L ranging from 2^{10} to 2^{20} cycles.

D. Power and area results

Since area and power measurements are independent of the computation time, the hardware results presented in this section do not depend on the bit-stream length. For the hardware simulations, this length is arbitrarily chosen at 10000 cycles, a number large enough to obtain representative results. These results are therefore consistent with any value of L, as long as the studied architecture achieves ideal accuracy, according to Fig. 3.

A first hardware simulation of the bare architecture of Fig. 1 using LFSR32, with 8 rows and 16 columns, shows that the the RNGs represent 29% of the circuit area (8135 out of 27775 μm^2), and 85% of its power consumption (2.451 out of 2.883 mW). This first result strongly motivates the use of schemes sharing RNGs to reduce their implementation cost.

In a second set of experiments, we compare the 4 more compact SNG architectures reaching ideal accuracy within the studied range of use ($2^{10} \leq L \leq 2^{20}$), and using or not our proposed SRI:

- Inverse permutation architectures of [7] sharing either LFSR16 or LFSR32 with WBG as PCC. These architectures can be used until 2^{16} and 2^{32} cycles without accuracy loss, and are respectively noted SOTA16 and SOTA32 (for State Of The Art);
- SRI architectures with either a sharing of a SBoNG-8 or a Xoroshiro-16, which can be used respectively until 2^{13} and 2^{32} cycles, and respectively noted SRI13 and SRI32.

We plot in Fig. 4a the power consumption comparison of these architectures, for circuits of 8 to 64 rows and 8 to 16 columns.

Since these implementations are not usable in the same range of use, we have to do 3 pairwise comparisons according to the bit-stream length L required in the application:

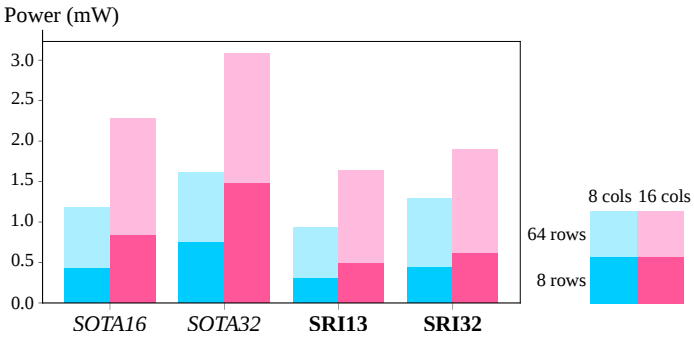
- between SOTA16 and SRI13 when $2^{10} < L < 2^{13}$,
- between SOTA16 and SRI32 when $2^{13} < L < 2^{16}$,
- between SOTA32 and SRI32 when $2^{16} < L < 2^{20}$.

For the sake of brevity, we do not present the full comparative study made on circuit areas for the different implementations (which show little variations) but only describe the main results.

Circuits using the SRI13 always have a smaller area and power consumption than their SOTA16 counterparts, as well as SRI32 compared to SOTA32.

Indeed, when comparing SOTA16 and SRI13 (for $L < 2^{13}$), for the worst case, (64 rows and 8 columns), the SRI13 architecture achieves a slight reduction of 0.2% in terms of area (from 70223 to 70075 μm^2) but is significantly better in power consumption compared to SOTA16 with more than 20% of reduction (from 1.175 to 0.931 mW). These results become, as expected, much better with the growing number of columns, with up to 4.5% of area (from 18227 to 17398 μm^2) and 41% of power consumption reduction (from 0.833 to 0.488 mW) for the best case (16 columns and 8 rows).

Besides, when comparing SOTA32 and SRI32, (for $L > 2^{16}$), the hardware comparison is also in favor of our proposition, with an area reduction of 11% in the best case (from 20155 to 17845 μm^2), and a power reduction ranging from 18% (from 1.611 to 1.307 mW) to almost 58% (from 1.48 to 0.623 mW) by using the SRI.



(a)

		nb_{col}	
		8	16
L	2^{10} - 2^{13}	**SRI13**	**SRI13**
	2^{13} - 2^{16}	*SOTA16*	**SRI32**
	2^{16} - 2^{20}	**SRI32**	**SRI32**

(b)

Figure 4: Comparison between *state of the art* and **proposed SRI** architectures. (a) Power consumption of the Bayesian computing core on different architectures and circuit dimensions. (b) Best architecture according to the required bit-stream length and number of columns.

However, when $2^{13} < L < 2^{16}$, we can see that the SRI32 implementation consumes from 6% (from 0.423 to 0.449 mW) to 11% (from 1.175 to 1.307 mW) more power than the SOTA16 implementation with 8 columns. But the trend is reversing with a 16 columns architecture, where our proposition becomes more power-efficient again, with 15 (from 2.278 to 1.924 mW) to 25% (from 0.833 to 0.623 mW) saved power.

In other words, and as shown in Fig. 4b, the proposed SRI architecture achieves better results for almost every scenario except when using a bit-stream of length between 2^{13} and 2^{16} with a circuit of 8 columns or less.

Finally, even though this SRI was introduced for a Bayesian circuit, it could be interest for other applications: it could typically provide gains when a) the number of parallel computations (analogous to our circuit rows) gets higher ; and b) the number of operands involved in these parallel computations (analogous to our circuit columns) gets higher too.

Note that for future works on different applications, other metrics than those used in this paper may be required, and the Root Mean Square Error could be used to replace the KLD for applications which do not compute an output probability distribution.

V. Conclusion

In this article, we introduced a new method for sharing a single RNG over a large number of stochastic bit-streams by shifting random numbers at its output. For this purpose, the RNG output needs to have lower auto-correlation than a standard LFSR and so has to be larger. However, the smaller is not always the better, since this method saves area and power by allowing to use one single RNG for the whole design. We also performed comparisons of different RNGs combined with different resource sharing methods, including the one proposed in this paper, on a Bayesian circuit for sensor fusion, in terms of area and power consumption in a 65 nm CMOS technology. The proposed architecture allows to save up to 11% of area and 58% of power consumption compared to the state of the art. Finally, we believe that our proposition can benefit to the stochastic computing community as it should generalize to other applications than the Bayesian sensor fusion circuits.

Acknowledgment

This work has been supported by the grant number 2019/0311 from ANRT.

References

[1] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), p. 149–156, Association for Computing Machinery, 1967.

[2] M. Faix, E. Mazer, R. Laurent, M. O. Abdallah, R. Le Hy, and J. Lobo, "Cognitive computation: a bayesian machine case study," in *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pp. 67–75, IEEE, 2015.

[3] T. Chen and J. P. Hayes, "Analyzing and controlling accuracy in stochastic circuits," in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pp. 367–373, 2014.

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

[5] M. Yang, B. Li, D. J. Lilja, B. Yuan, and W. Qian, "Towards theoretical cost limit of stochastic number generators for stochastic computing," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 154–159, IEEE, 2018.

[6] H. Ichihara, S. Ishii, D. Sunamori, T. Iwagaki, and T. Inoue, "Compact and accurate stochastic circuits with shared random number sources," in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pp. 361–366, 2014.

[7] S. A. Salehi, "Low-cost stochastic number generators for stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 992–1001, 2020.

[8] A. Coninx, P. Bessière, E. Mazer, J. Droulez, R. Laurent, M. A. Aslam, and J. Lobo, "Bayesian sensor fusion with fast and low power stochastic circuits," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, IEEE, 2016.

[9] R. Frisch, R. Laurent, M. Faix, L. Girin, L. Fesquet, A. Lux, J. Droulez, P. Bessière, and E. Mazer, "A Bayesian stochastic machine for sound source localization," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, IEEE, 2017.

[10] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 39–46, 2013.

[11] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "A self-timed ring based true random number generator," in *2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems*, pp. 99–106, 2013.

[12] J. H. Anderson, Y. Hara-Azumi, and S. Yamashita, "Effect of lfsr seeding, scrambling and feedback polynomial on stochastic computing accuracy," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1550–1555, 2016.