An Energy Efficient Multi-Rail Architecture for Stochastic Computing: A Bayesian Sensor Fusion Case Study

Jérémy Belot, Abdelkarim Cherkaoui, Raphael Laurent, Laurent Fesquet

An Energy Efficient Multi-Rail Architecture for Stochastic Computing: A Bayesian Sensor Fusion Case Study

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*—Recently, Stochastic Computing has sparked interest in Bayesian inference resolution for its promising efficiency in area and power consumption. This representation encodes values by the rate of bits at '1' in a bit-stream. Still, in a sequential architecture, most of the energy cost is due to the long computation time required for achieving a satisfying accuracy. In this paper, we propose a multi-rail architecture for Bayesian sensor fusion problems based on a Shift Register Isolator and permutations in order to reduce the computation time and thus, the energy consumption, without a significant increase in area. Indeed, with this resource sharing strategy, we are able to reduce the energy consumption by up to 73% in return for an area overhead of 24%, while maintaining the computation accuracy.**

I. INTRODUCTION

In recent years, the advent of the Internet of Things and the energy constraints of these systems have brought to the fore sensor fusion solutions. These circuits reduce, at the closest to the sensor, a raw data flow into a few relevant information, saving unnecessary data transmission and thus power consumption. Bayesian inference appears to be a good way to build fault-tolerant and explainable models for sensor fusion. Recently, architectures based on Stochastic Computing (SC), a non-standard representation introduced in 1956 by Von Neumann [1] and later developed by Gaines [2], have been successfully used to implement sensor fusion models and temporal filters [3]–[6].

In unipolar representation, such circuits encode values by the rate of bits at '1' in a bit-stream. This approach is particularly interesting to perform, at a low logical cost, demanding computations in standard representation, such as multiplication. Indeed, the multiplication between two independent stochastic bit-streams can be performed with a simple AND-gate. For example, let $s_0 = 1011101000101011$ and $s_1 = 1000100101110010$ be two unipolar stochastic bit-streams, representing the values $P(s_0) = \frac{9}{16}$ and $P(s_1) = \frac{7}{16}$. Their AND gated bit-stream is $s_0 \wedge s_1 = 1000100000100010$ and $P(s_0 \wedge s_1) = \frac{4}{16} \simeq \frac{9}{16} \times \frac{7}{16} = \frac{63}{256}$. Of course, the longer the bit-streams, the better the accuracy.

In order to make these bit-streams independent, they are generated using Stochastic Number Generators (SNGs) that are generally composed by:

- a Random Number Generator (RNG), usually a Linear-Feedback Shift Register (LFSR) for its low area and power consumption, that provides the bit-stream independence if the seeds are chosen wisely;
- a Binary to Stochastic Converter (BSC), usually a comparator, that encodes a binary value into a bit-stream thanks to the generated random numbers.

These SNGs are responsible of the major part of SC circuit power consumption and area. Several works have successfully found ways to reduce this impact [6]–[8], but this is not the main purpose of this paper.

The conversion from the stochastic to the standard representation is done with a counter reckoning the number of '1' in the output bit-stream, and another one reckoning the Stochastic Bit-stream Length (SBL). The results correspond therefore to the ratio between the number of '1' and the SBL.

Such SC circuits can be implemented following two ways:

- Sequentially and single-rail: each bit-stream is single-wired, and each bit that composes it is generated at a different clock cycle. This is the most efficient strategy in terms of area and instantaneous power. However, this implies long computation time, especially when the wanted accuracy is high, which increases the energy consumption.
- Fully parallel: as in [9], each bit-stream is a bus, and each bit that makes it up is generated in the same clock cycle. This implementation allows the calculation to be performed in a single clock cycle, which can be interesting in terms of energy efficiency, if the required accuracy is not too high, but at the price of a larger area and instantaneous power consumption. Moreover, in the case of dedicated circuits to Bayesian sensor fusion, the required accuracy strongly depends on the application and is sometimes wanted configurable. Therefore, the fully parallel implementation could not be suitable.

In this paper, we propose a hybrid solution, sequential and multi-rails taking advantage of both techniques, reducing computation time and energy while keeping comparable area and accuracy thanks to an implementation sharing resources. Firstly, we present the different works useful to introduce our solution. Then, we describe the proposed architecture and its specificities. Finally, we proceed to measurements and comparisons in terms of accuracy, computing time, area and energy consumption.

II. Related Works

A. Stochastic circuits for Bayesian sensor fusion

In a Bayesian sensor fusion model, a state variable S is usually inferred from n independent sensor readings K_j using sensor models $P(K_j|S)$ thanks to Bayes' theorem [10]:

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

The probability distribution $P(S)$, usually called *prior*, encodes *a priori* knowledge about the state S. The conditional probability distribution $P(K_j|S)$ on sensor reading K_j 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 $1 \leq i \leq m$, Eq. 1 becomes:

$$P(S = s_i|\wedge_{j=1}^{n} K_j = k_j) \propto P(S = s_i)\prod_{j=1}^{n} P(K_j = k_j|S = s_i) \quad (2)$$

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 [4].

When a new sample k_j is acquired, the *likelihood* $P(K_j = k_j|S = s_i)$ is read from memory. Then, this value is converted into a bit-stream with the corresponding probability thanks to a BSC and a RNG. Since rows are independent, it is possible to share one RNG per column. Thus, only $n + 1$ independent numbers are needed to generate $m * (n + 1)$ stochastic bit-streams.

Fig. 2 shows the architecture of our Bayesian sensor fusion circuit. The independent random numbers are given by the shift register, as proposed in [6] and developed in the next subsection.

B. Architecture for sharing stochastic resources

In [7], Chen *et al.* introduce the notion of *stochastic isolation* in order to reuse one RNG over several bit-streams and thus save area and power consumption. To do so, they use registers (*isolators*) to create new bit-streams from others. These new bit-streams are shifted by one clock cycle in time and can therefore be considered as independent of the original ones. In counterpart, this single RNG has to show low auto-correlation properties to maintain original accuracy, and must be more sophisticated than a simple LFSR.

In [8], Yang *et al.* propose a power and area efficient way to use the Weighted Binary Generator (WBG), a BSC introduced by Gupta *et al.* in [11]. As shown in Fig. 1, the WBG has a part, the Weight Generator (WG), which does not depend on the binary value to be encoded. Thus, just like RNGs, these WGs can be shared over each computationally independent bit-stream (rows in our case), as shown in Fig. 2. The second part,

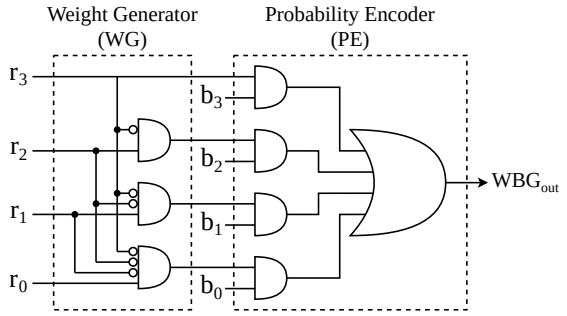

Figure 1: Decomposition of a WBG with 4-bits random number ($r_{i,i\in[0;3]}$) and 4-bits binary value ($b_{i,i\in[0;3]}$).

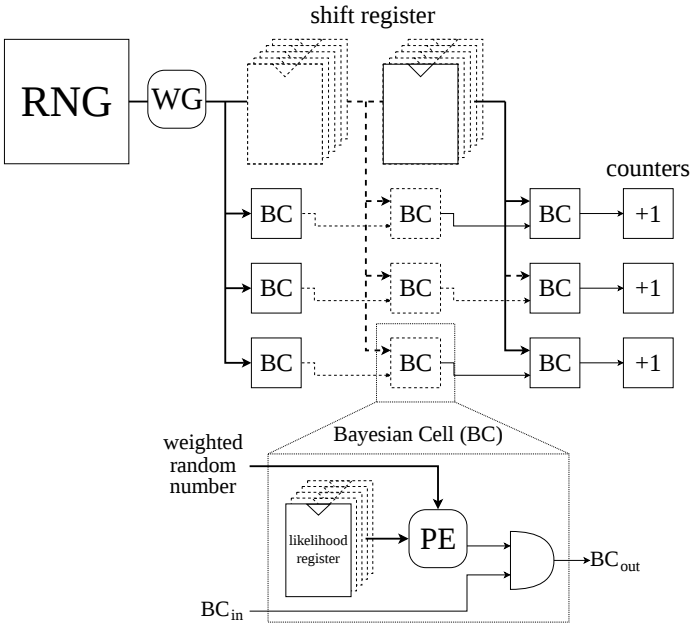

Figure 2: Bayesian sensor fusion architecture with SRI solution.

the Probability Encoder (PE), replaces the standard comparator in each cell of the matrix and, since it is smaller, this solution saves area and power consumption.

Taking advantage of these two works, in [6], Belot *et al.* propose a Shifted Register Isolator (SRI), which does not place the *isolation* at the bit-stream level anymore, but at the random number level. This becomes more interesting than Chen *et al.*'s *isolators* [7] when the size of the random numbers is smaller than the number of independent computations (number of rows here). It also implements a single WG for the whole circuit as shown in Fig. 2.

III. A multi-rail sequential SC architecture

In this paper, we propose a sequential and multi-rail architecture that further reduces the energy consumption compared to a single-rail one, without significantly increasing the SC circuit area. Moreover, this partially parallel architecture allows a compromise between area and energy consumption, which

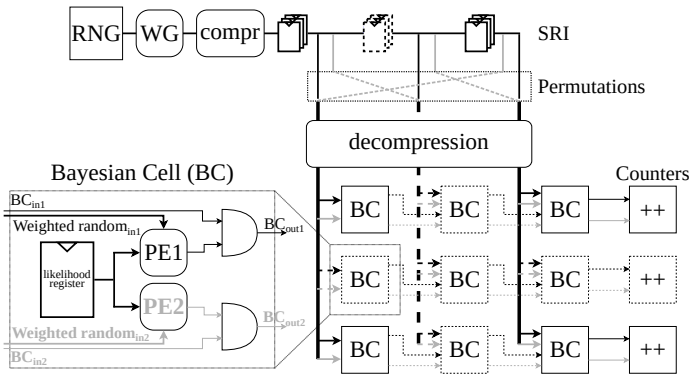

Figure 3: Proposed sequential multi-rails (here 2 rails, in black and in gray) SC architecture for Bayesian sensor fusion.

is not possible in the fully parallel architecture presented in [9]. The proposed architecture is described in Fig. 3.

In this proposition, each row is split in n rails, $n \in \mathbb{N}^*$, which multiplies the number of AND-gates and PEs. The counters are modified to add more than one bit per clock cycle, and the likelihood registers are shared between the rails.

The different random numbers required for each rail are generated without any additional cost by simply permuting the outputs of the SRI between the columns. Thus, each rail in a row produces an independent bit-stream since they get different random number as input. This is done with a negligible accuracy loss until a certain number of rails. This matter is discussed in Sec. IV-A.

Moreover, one can notice that the architecture is also slightly different from the one presented in [6]. We indeed optimize it with additional registers just after the WG in order to remove glitches occurring in PEs and thus reduce the power consumption. We also compressed the data in the shift-register in order to decrease the number of used registers. It is possible to do so since the output of the WG is redundant. Indeed, at each clock cycle, its output has always only one bit at '1' and the others are '0'. So we can encode the position of the bit at '1' instead of the whole output, and then implement only $log_2(n_{bit})$ registers for each stage of the shift-register, instead of n_{bit} (3 instead of 8 in our case).

In conclusion, the implementation of additional rails leads to an increase in area and power consumption. However, since many resources are shared between the rails (RNG, WG, SRI, likelihood registers), this is offset by the reduction of the computation time, allowing energy savings, as shown in Sec. IV-B.

IV. Accuracy, area and energy comparisons

A. Accuracy measurement and columns permutations

In order to determine the best sets of columns permutations minimizing the accuracy losses, a software simulation of the circuit has been performed in C language. The software simulation is carried out with randomized RNG seeds and likelihoods data. We use a variable SBL, 8 columns, and as many rows as possible to get a good accuracy measurement in an acceptable simulation time (around 30000 rows).

We measure the accuracy by comparing the resulting distribution of our SC circuit P_{stoc} with the one computed with floating point (reference) P_{float} using the Kullback-Leibler Divergence (KLD), defined in Eq. 3.

$$D_{\text{KL}}(P_{\text{stoc}} \| P_{\text{float}}) = \sum_{j=1}^{m} P_{\text{stoc}}(j) \log \frac{P_{\text{stoc}}(j)}{P_{\text{float}}(j)} \qquad (3)$$

The KLD is always positive and the lower it is, the closer the two distributions are, so the better is the given permutation.

We first measure the precision (KLD) with 2 rails in an exhaustive way for all the $8! = 40320$ possible permutations. We therefore have a set of measurements that reflect the pairwise impact of column permutations on accuracy, independently of the data. Using these measurements, we built an algorithm that is able to select the permutations minimizing their pairwise correlation for a given number of rails. Then, the accuracy (KLD) is measured to ensure that it remains approximately the same than before permutation. After that, the selected set of permutations is directly hard-coded in the circuit. This method represents a huge saving in simulation time, the exhaustive way becoming quickly intractable with a running time proportional to $(8!)^{n_{rails}}$.

For the sake of fair comparison, for the following, we wish to achieve the same KLD as the single-rail architecture, so that we can compare energy consumption at equal precision. We therefore have to slightly increase the multi-rail SBLs instead of simply dividing the single-rail SBL by the number of rails. In consequence, this accuracy correction slightly increases the energy consumption.

Table I shows the different SBL necessary to reach certain values of precision depending on the number of rails.

	Nb_rails							
	1	2	3	4	5	6	7	8
KLD	SBL (in cycles)							
0.020	6000	3000	2050	1580	1330	1100	930	880
0.010	12000	6050	4150	3050	2500	2100	1800	1700
0.005	24000	12900	8900	6900	5900	5400	4600	4500

Table I: Stochastic Bit-stream Length required to reach different values of KLD according to the number of rails.

We see that the greater the number of rails, the more cycles must be added to the ideal SBL $\frac{SBL_{1rail}}{nb_{rails}}$ to reach the reference KLD. Furthermore, the increase of the number of permutations seems limiting the circuit to reach high precision, as shown in the last row of Tab. I. Indeed, the number of additional required cycles becomes very large, more than 1000 cycles when using more than 4 rails. These points show, despite our efforts to limit them, the negative impact of permutations on the accuracy, which must be compensated by a higher SBL and therefore a higher energy consumption.

B. Area and energy consumption measurement

The following hardware results in terms of area and energy consumption are gathered with a STMicroelectronics 65 nm

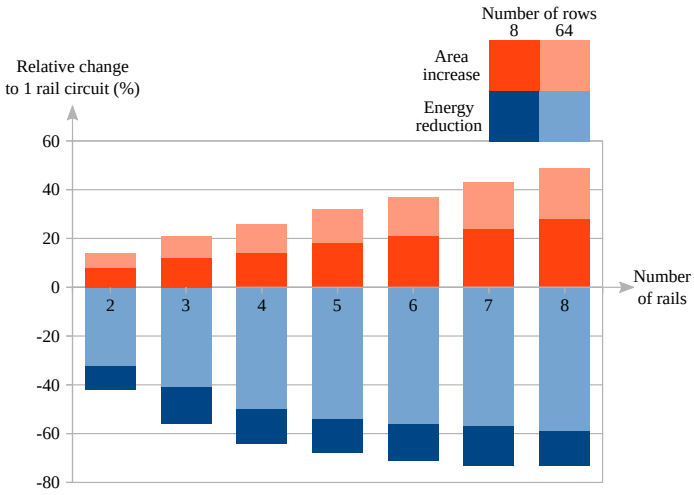

Figure 4: Relative change of area and energy consumption compared to a 1-rail architecture with a KLD=0.01.

CMOS technology and using retro annotated simulations. The circuit dimensions are as it follows:

- The likelihoods are encoded with 8 bits, as well as the random inputs of the PEs, and thanks to the compression, the SRI is coded on 3 bits;
- The RNG is a 16-bits Xoroshiro, which has a lower auto-correlation than the LFSR as it is required for using the SRI;
- The number of columns is fixed to 8 and the number of rows varies from 8 to 64;
- The counters have 8-bit outputs;
- The reference accuracy varies in the range {0.005, 0.010, 0.020};
- The clock frequency is set to 100 MHz;
- The number of rails varies from 1 to 8.

The results are measured on the whole Bayesian sensor fusion circuit, including the SC core described in Fig. 3, but also a control block (Finite State Machine) and models memories.

Table II shows the absolute comparison in energy and area consumption with different accuracy and number of rails.

			Number of rails							
	KLD	Rows	1	2	3	4	5	6	7	8
Energy consumption (nJ)	0.020	8	21.5	12.6	9.70	8.36	7.70	6.95	6.36	6.34
		64	69.2	47.9	41.9	37.0	35.4	33.4	32.3	31.5
	0.010	8	41.7	24.1	18.4	15.0	13.4	12.2	11.2	11.1
		64	132	89.8	77.8	65.1	60.6	57.6	56.1	54.5
	0.005	8	82.2	50.1	38.1	32.4	30.0	29.4	26.7	27.4
		64	256	184	159	139	134	137	133	133
Area (10^3µm²)		8	31.9	34.4	35.7	36.4	37.6	38.6	39.5	40.7
		64	149	169	180	187	197	204	213	221

Table II: Energy and area comparison according to the number of rails.

Fig. 4 shows the relative changes of these measurements compared to the single-rail architecture for a KLD of 0.01. Note that the relative comparisons are not linear since, for example, doubling the area would result in a +100% increase, while halving the energy would lead to a -50% reduction.

We can see that the more rails are implemented, the more energy is saved, until reaching a plateau with 8 rails. With a KLD of 0.01 and 8 rails, it is therefore possible to save up to 73% of energy with 8 rows and up to 59% with 64 rows.

On the other hand, the circuit area is linearly increasing with the rising number of rails. Taking 8 rails, the total circuit area overhead is up to 28% with 8 rows and up to 49% with 64 rows. Note that for the specific block of the SC core described in Fig. 3, these overheads reach 58% with 8 rows (from 15000 to 23800µm²) and up to 94% with 64 rows (from 77000 to 149400µm²).

Finally, notice that 8 rails does not seem to be a judicious choice since we obtain the same energy performances with 7 rails (-73%) while reducing the area overhead (+24% instead of +28%). Therefore, we did not study solutions beyond 8 rails, the plateau being reached.

V. Conclusion

In this article, we introduced the SC multi-rails architecture, a new way to parallelize SC circuits in order to reduce their computation time and thus their energy consumption. This proposition is based on an optimized version of the SRI introduced in [6], and the permutations of its output random numbers to generate independence between the different rails without hardware cost and significant accuracy losses. Comparisons in terms of area and energy have been carried out and our proposal is able to save up to 73% of energy with 7 rails and a KLD of 0.01, in return for a 24% increase in surface area on the total circuit.

Acknowledgment

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

References

[1] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata studies*, vol. 34, pp. 43–98, 1956.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] J. Belot, A. Cherkaoui, R. Laurent, and L. Fesquet, "An area and power efficient stochastic number generator for bayesian sensor fusion circuits," *IEEE Design Test*, pp. 1–1, 2021.

[7] 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.

[8] 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.

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

[10] P. Bessiere, E. Mazer, J. M. Ahuactzin, and K. Mekhnacha, *Bayesian programming*. CRC press, 2013.

[11] P. K. Gupta and R. Kumaresan, "Binary multiplication with pn sequences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 603–606, 1988.