Article

Implementation of the Hindmarsh–Rose Model Using Stochastic Computing

Oscar Camps [1], Stavros G. Stavrinides [2], Carol de Benito [1,3] and Rodrigo Picos [1,3,*]

[1] Industrial Engineering and Construction Department, University of Balearic Islands, 07122, Palma, Spain; oscar.camps.pascual@gmail.com
[2] Physics Department, International Hellenic University, 65404, Kavala, Greece; s.stavrinides@ihu.edu.gr
[3] Balearic Islands Health Institute (IdISBa) 07120, Palma, Spain
[*] Correspondence: rodrigo.picos@uib.es

Abstract: In this paper we present a successful implementation of the Hindmarsh–Rose model within a SC environment. The merits of the proposed approach are design simplicity, due to stochastic computing, and the ease of implementation. Simulation results showed that the approximation achieved is equivalent to introducing a noise source into the original model. A study for the level of noise introduced, according to the number of bits in the stochastic sequence, has been performed. Additionally, we demonstrate that such an approach, even though it is noisy, it reproduces the behaviour of biological systems, which are intrinsically noisy. It is also demonstrated that a speedup of x2 compared to biological systems is easily achievable, with a very small number of gates, thus paving the road for the *in silico* implementation of large neuron networks.

Keywords: stochastic logic; chaotic systems; approximate computing; Hindmarsh Rose system

1. Introduction

Simulation of the brain is one of the big issues for this century. The brain being the most complex structure in the known universe, problems inherent to this goal are many and different [1–4]. The current proposals cover many different approaches from quantum computing [5] to neural networks [6], including simulation at the level of tissues [7], interactions with different parts [8], or descriptions of the communications at a higher level [9].

The human brain has around 86 billion neurons, with thousands of inputs each. Thus, one of the biggest challenges is the required computing infrastructure. There are many different activities related to this [4,10–12], usually based on high performance computing resources. For instance, [13] proposes using memristors to implement biologically inspired neural networks, while [14] proposed a framework for mult-scale hardware simulation of the brains, or [15] implemented an accelerator for neural network simulation using analog elements.

This way, brain simulation is leaking results into neuromorphic computing, both providing computing capabilities or even new simulation and emulation approaches; for instance, the IoT computing environment can take great profit . Within this framework, approximate computing emerges as an important technological pathway, since it enables high power savings [16]. This framework offers energy savings, trading accuracy for energy, though. To this direction some methods for successfully implementing approximate computing, ranging from software-based approaches (programming methods and algorithms) to hardware implementations or ubiquitous solutions, have been proposed.

Stochastic Computing (SC), proposed by Von Neumann in 1956 [17], is one such approach that makes a trade off between calculation time and accuracy. It offers significant advantages, the most important of them being the utilization of simple logic operations and the significant reduction of circuit components, when we refer to hardware implementation, apparently, reducing the power consumption. The main drawback of this approach is the increased logic-operation time demanded, which in its turn increases the total power consumption when the number of bits exceeds 16–17 [18,19]; but there have already appeared techniques that allow this problem to be alleviated, making SC competitive even for higher bit-numbers [19].It has to be mentioned that the property of stochasticity demanded for number generation could be considered as an equivalent of noisy system, its existence being many times beneficial for the designed system.

The neuromorphic computing approach characteristics lay very close to stochastic computing ones, demonstrating a noteworthy implementation compatibility, i.e., the sets of consecutive spikes in neuromorphic, as these are compared to the pulse-trains in stochastic computing.

In this paper we present a successful implementation of the Hindmarsh–Rose (HR) model within a SC environment. The merits of the proposed approach are design simplicity (due to SC) and the ease of implementation. Simulation results in Matlab showed that the SC achieved approximation is equivalent to introducing a noise source into the original model.

A study for the level of noise introduced, according to the number of bits in the stochastic sequence, has been performed. Additionally, we demonstrate that such an approach, even though it is noisy, it is

Although this appears to be a certain drawback, the proposed approach is beneficial for both edge computing of parallel processing systems; especially when we are having in mind neuromorphic-inspired designed systems.

2. Stochastic Computing Implementation of Analog Systems

2.1. Introduction to Stochastic Computing

In the approach proposed by Stochastic Computing (SC), numbers are ideally represented by probabilities. Thus, operations between numbers can be represented as compound probabilities. However, and due to the fact that we cannot represent an actual probability p, we estimate this probability as the average value of a set of samples P, thus assuming $p \approx < P >$ and also obtaining an error for the estimation ϵ_p, that can be related to the standard deviation σ_p of the average value p.

When the SC approach is used, the real numbers are represented by a binary string of random zeros and ones. The probability of finding a one is related to the value of the represented real number [20]. Since the string length is finite, the probability is calculated as the average value of this string (between zero and one). These strings are usually referred to as Stochastic Computing numbers (SCN) or Stochastic Encoded Numbers (SEN). As for the rest of this paper, we will use the second calling convention, as well as also using Binary Encoded Numbers (BEN) as the name for those numbers that are encoded as classical binary numbers. It has to be noted that two main mappings from real to SEN are typically used, either from the real domain [0,1], or from the real domain $[-1\,1]$.

Some basic arithmetic operations can be performed using basic gates, depending on the chosen mapping. For instance, considering the [0,1] domain, multiplicating two Stochastic Computing numbers is performed using an AND gate. On the other hand, if the $[-1,1]$ domain is considered, it requires the multiplication to be done using XNOR gates, as shown in Table 1.

Adding two numbers is slightly more difficult, since the result could be higher than 1 and, thus, out of the considered range. This is solved by implementing $(x+y)/2$, which is always in range. This function is usually constructed with a multiplexer, as in Table 1. In this Table, p represents a signal where a "1" is found with a probability of 0.5. It is interesting to notice that the same gate is used in both the [0,1] and the $[-1,1]$ domains.

In the case where other operations that are more complex are needed, many different implementations are found in the literature (division [21], square roots [21], reversible gates [22], *etc.*), though they are not to be presented here to focus on the implementation procedure of the studied system, which only needs additions (substraction) and multiplications.

Table 1. Implementation of basic operations in Stochastic Computing in the [−1,1] range. The value of p is such that it is 0 or 1 with a 0.5 probability, $\bar{p} = 1 - p$, and q is a random variable with a normal distribution between -1 and 1. Notice that these functions can be trivially expanded to work for arbitrary vector length.

Operation	Implementation	Function Name
(x+y)/2	OR(AND(p,x), AND($\bar{p}$,y))	add(x,y)
x*y	XNOR(x,y)	mult(x,y)
-x	NOT(x)	neg(x)
Real number to SEN	0 if q>x; 1 otherwise	get_sn(x)

Related to the conversion between BEN and SEN, it is usually implemented using a N-bit random number generator (RNG), whose output is compared to the value of the N-bit BEN. If the RNG number is below the BEN, the binary output is 1, or 0 otherwise, as described in Table 1. The complementary operation, SEN converted back to its BEN representation, a counter is used to determine the number of 1 in the string, which is a estimation of the probability. This obviously presents some error, as discussed below.

2.2. Error Estimation

Stochastic numbers are equivalent to bimodal processes. Thus, the error made when considering the approximation of a SEN to the value it represents, the error can be calculated using a random walk process of length n. Thus, it is proportional to $\sqrt{n}$ [23]. This way, for a number representing N bits, the noise caused by the random walk process can be considered to be included in the lowest $N/2$ bits. That is, the relation between the power S_p of the signal and the power N_p of the noise (i.e., the noise figure NF) is:

$$NF = 10log_{10}\left(\frac{S_p}{N_p}\right) = 10log_{10}\left(\frac{2^N}{2^{N/2}}\right) \approx 3.01N/2\,dB \tag{1}$$

This NF is a key parameter to determine the required number of bits. It is also closely related to the sensitivity of the equations system to noise. That is, we have empirically seen by simulation that linear equations can still behave correctly using a low N. However, nonlinear systems need higher values in order to reproduce a correct behavior [24].

On the other hand, it is also worth noting that the noise can also be interpreted as a random noise of amplitude $2^{1-N/2}$, assuming that we are using the $[-1,1]$ range. For instance, a 16 bit implementation would lead to an equivalent noise amplitude of $2^-7 \approx 0.008$ in the mentioned range.

2.3. Implementation of Basic Differential Equations

The process of implementing differential equations using SC requires rewriting them in a specific way [24]. Specifically, the three different transformations below are needed:

1. The equation terms must be organised in a form suited to SC. As an example, the additions must be replaced by half additions: $a + b \rightarrow (2a + 2b)/2$, while multiplications remain the same. In the case where more complex operations are needed, an expansive reworking of the equations may be needed to ensure all the operations can be implemented in SC in the $[-1,1]$ or [0,1] range.
2. All the variables have to fall inside the chosen domain ($[-1..1]$ or [0..1]).
3. Any remaining coefficient must be below 1 (in absolute value). This is done using a time scaling.

Once these three transformations are performed, the equations may be processed as a SC system. As an example, multiplications and additions found in the original system are to be implemented with the functions in Table 1.

Another of the elements needed to implement ODE is an integrator. It can be realized easily using the description in Figure 1. The circuit there performs a continuous integration, which is implemented by using a counter that increases or deacreses one unit depending on whether the input is 1 or 0. The value of this counter is a BEN, which is then converted to SEN. Notice that the design of the counter includes deciding on the number of bits it has, this being equivalent to set the precision of the integrator, with a noise figure provided by Equation (1). In relation with the number of RNG that are used for this scheme, [25] shows that the use of the same RNG in both inputs leads to an actual improvement of the accuracy.

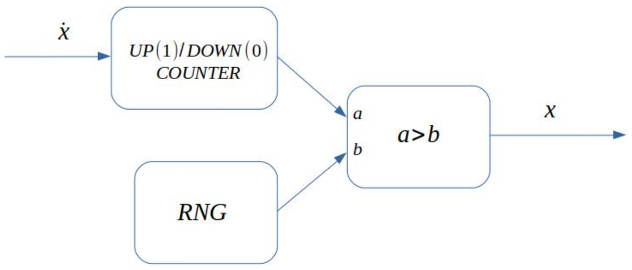

Figure 1. Basic implementation scheme of a SC integrator. Notice that both the input $\dot{x}(t)$ and the output $x(t)$ are SEN numbers.

A closely related topic is how the time step Δt in the simulation is related to the numer of bits. Obviously, this is dependent on which integration method is used. Just to illustrate the point, let's assume that a simple first order rectangular integration with no explicit time dependence is utilized. In this case, we can write in a first approximation of order $O(\Delta t^2)$:

$$\begin{aligned} \dot{x} &= f(x) \\ \dot{x} &\approx \frac{\Delta x}{\Delta t} = f(x) \rightarrow \Delta x = f(x)\Delta t \end{aligned} \tag{2}$$

On the other hand, applying the integrator in Figure 1 for a large number N_{acc} of iterations of a stochastic number, it is clear that the output at the counter $Int(x(t))$ has to be:

$$Int(x) = \frac{N_{acc} \cdot p(1)}{2^N} \tag{3}$$

where $p(1)$ is the probability of having a 1 at the input. As per the definition of the SEN, $p(1)$ corresponds to the value represented in the SCN, while N_{acc} can be identified as the number of 1 counted by the accumulator, corresponding to the estimation of the value of $p(1)$ with a noise, as discussed above. Using Eq. (2) with Eq. (3), the effective time step is determined as:

$$\Delta t = \frac{N_{acc}}{2^N} \tag{4}$$

Notice that the integrator depicted in Figure 1 also includes a binary to stochastic (B2S) converter, which is simply a random number generator (RNG), plus a comparator. This way, the output of the integrator is already also a SC number that can be used further in the circuit.

3. The Hindmarsh–Rose Model

3.1. Original Model

The Hindmarsh-Rose model [26,27] was proposed in the first half of the 1980's to describe the membrane potential $x(t)$ of a neuron, coupled to the rate of transport of sodium and potassium ions $y(t)$, as well as to a so-called adaptation current $z(t)$. The equations proposed by this model are as follows, when expressed in the dimensionless form:

$$\frac{dx}{dt} = y - ax^3 + bx^2 - z + I \tag{5}$$

$$\frac{dy}{dt} = c - dx^2 - y \tag{6}$$

$$\frac{dz}{dt} = r(s(x - x_R) - z) \tag{7}$$

The usual values of the dimensionless parameters are a=1, b=3, c=1, d=5, r=10^{-3}, s=4, x_R=-1.6. The first four (a,b,c,d) are used to model the behaviour of the fast ion channels, while r models the slow ion channels. The time scale of the neuron is defined by r, and x_R is related to the membrane potential. The value of the forcing current into the neuron I is around -10 to 10, and we have used I=3. Some procedures to fit these parameters to actual measured neuronal behaviour can be found in the literature [28–30].

3.2. Stochastic Computing Implementation

In order to implement the model, equations 5-7 must be rewritten in a suitable form, as discussed above. That is, we have to ensure the following conditions [24]:

1. All the additions are expressed in a suitable form for stochastic computing.
2. All the values of x,y and z are inside the [-1,1] interval.
3. All the values of the parameters are inside the [-1,1] interval.

Thus, first we change all the additions according to:

$$x + y \longrightarrow \frac{1}{2}(2x + 2y) \tag{8}$$

Second, we perform a change of variables to get all the values of the variables in the desired range [-1, 1]:

$$x' \longrightarrow \frac{x - x_a}{x_\sigma} \tag{9}$$

$$y' \longrightarrow \frac{y - y_a}{y_\sigma} \tag{10}$$

$$z' \longrightarrow \frac{z - z_a}{z_\sigma} \tag{11}$$

Finally, we absorb the maximum (absolute) value τ of all the multiplicative factors after the above changes inside the time variable:

$$t' \longrightarrow \frac{t}{\tau} \tag{12}$$

In order to perform the implementation, we show the code required to do so in Fig. 3.2. The auxiliary functions add(), mult(), neg() and get_sn() are defined in Table 1. Notice that, to eliminate the problems associated to correlation when using a squared variable, we use different stochastic values each time we need to use x (X0, X1, X2, X3, X4, X5), y (Y0, Y1), and z (Z0, Z1). All these values obviously represent the same values but, due to the probabilistic nature of the stochastic representation, are different chains of 0 and 1.

4. Results

All the results in this section have been obtained using Matlab simulation. The corresponding code is freely available at https://github.com/rpicos-uib/stochastic_nonlinear_chaos, in the HR_model folder.

The first batch of results concern the number of bits (i.e. the length of the chain) needed to obtain the expected behaviour. That is, we expect to be able to reproduce spiking. To do so, we first integrated the HR model using conventional arithmetic, for the set of

```
    f10 = mult(get_sn(x1),Y0)), neg(mult(get_sn(x2),X0));
    f11 = add(add(get_sn(x0), f10);
    f12 = add(mult(get_sn(x3),X1), neg(mult(get_sn(x4),X2)));
    f13 = add(neg(mult(get_sn(x5),Z0)), get_sn(u) );
    f14 = add(f12,f13);
    x = add(f11,f14);
    f20 = neg(add(get_sn(y0), mult(get_sn(y1),X3)));
    f21 = add(mult(get_sn(y2),X4), neg(mult(get_sn(y3),Y1)));
    y = add(f20,f21);
    f30 = add(mult(get_sn(z1),X5), neg(mult(get_sn(z2),Z1)));
    z = add(neg(get_sn(z0)), f30);
```

Listing 1. Example of code used to implement Eq. 5-7 in a form suited to Stochastic Computing. The auxiliary functions are specified in Table 1. The values x0..x5, y0..y3, z0..z2 are the values of the parameters in the transformed equations.

parameters a=1, b=3, c=1, d=5, r=10^{-3}, s=4, x_R=-1.6. For the sake of convenience, we will call this simulation the *exact* solution. We integrated up to a final time of 100s with dt=0.01s, using the transformed equations discussed in the previous section, and initial conditions x=0.351, y=0.862, z=0.666, which are equivalent to the non-scaled x=0.1, y=0.1, z=3. Notice that, since the scaling is different for the different variables, the initial conditions do not scale equally. The results for the x variable are shown in Fig. 2. We can see that the spiking behaviour is present, with amplitudes close to 0.6 and increasing distance between them. The phase space representation of this system (the x-y variables) is shown in Fig. 3.

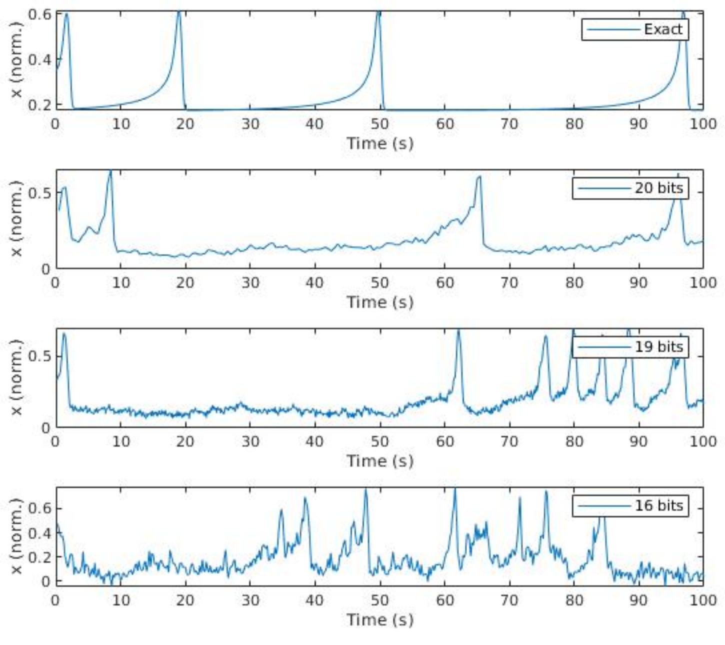

Figure 2. Temporal evolution of the x variable in the HR model, using conventional arithmetic (top, "exact"), and stochastic computing (20, 19, 16 bits).

After the exact simulation, we have tested three different lengths l_B of the SEN. Specifically, we have tested for N_b=20, 19, 16, where $l_B = 2^{N_b}$. The temporal evolution of these solutions are depicted in Fig. 2, and the space states are plotted in Fig. 3. All of these simulations were obtained using the same initial conditions and parameter values than for the exact solution.

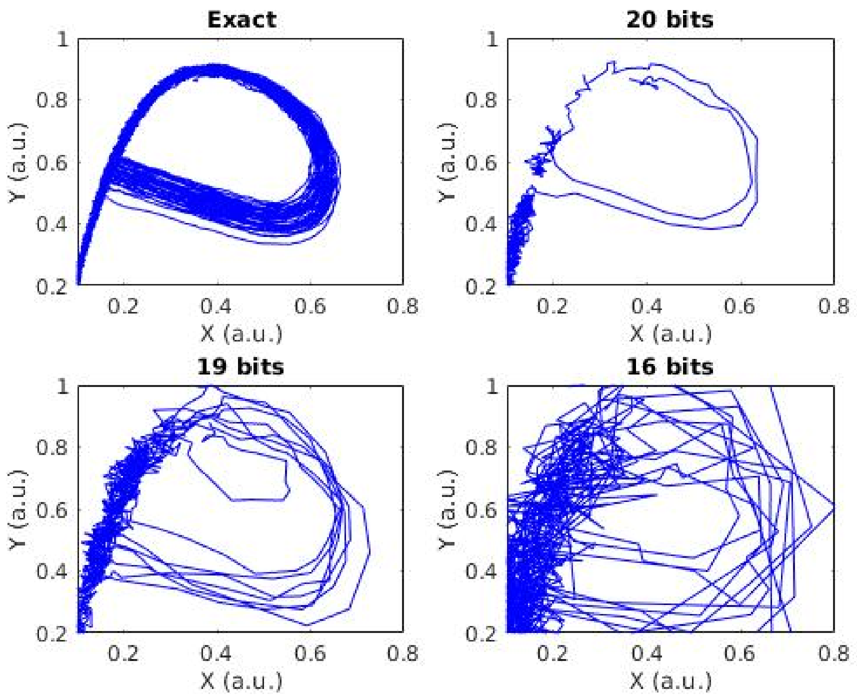

Figure 3. Space state of the x and y variables in the HR model, using conventional arithmetic (top, "exact"), and stochastic computing (20, 19, 16 bits).

As is clear from the Figures, the spiking behaviour is present in all three SC simulations. The results corresponding to 16 bits seem too much noisy, and the space state portrait is hard to distinguish. However, at 19 and 20 bits, the signal is much clearer, and the portrait is obviously present. In addition, the temporal evolution of the signal is less noisy.

It is worth noticing that the temporal evolution of the signals is quite different from the exact solution. This could be expected, since one of the main characteristics of nonlinear systems is that small differences at the input cause enormous differences in the dynamical evolution. As discussed above, we have noise in all of the parameters of the system: the variables and the constants. That is, we can assume that we have a noise signal overlapping the integrator, as mentioned when discussing the noise in SC systems. We have checked the effect of noise in the exact solution by introducing three noise sources (ϵ_x, ϵ_y, ϵ_z) in the HR equations:

$$\frac{dx}{dt} = y - ax^3 + bx^2 - z + I + \epsilon_x \quad (13)$$

$$\frac{dy}{dt} = c - dx^2 - y + \epsilon_y \quad (14)$$

$$\frac{dz}{dt} = r(s(x - x_R) - z) + \epsilon_z \quad (15)$$

The noise sources are white noise generators of constant amplitude a_ϵ. Figure 4 compares three different simulation using the exact solution (top) with $a_\epsilon = 0.02$ noise, and three different simulations using SC and 19 bits (bottom). As can be seen in the exact solution, even such a small noise affects enormously the dynamics.

To further study this equivalence, we have performed several simulations for different numbers of bits, between 11 and 24. The equivalent noise level has been calculated as the rms noise when the x variable is not spiking, since in the exact solution this would correspond to a zero value. The results are shown in Fig. 5, where the noise equivalent level for each simulation is shown as a cross, and an average for all the simulations

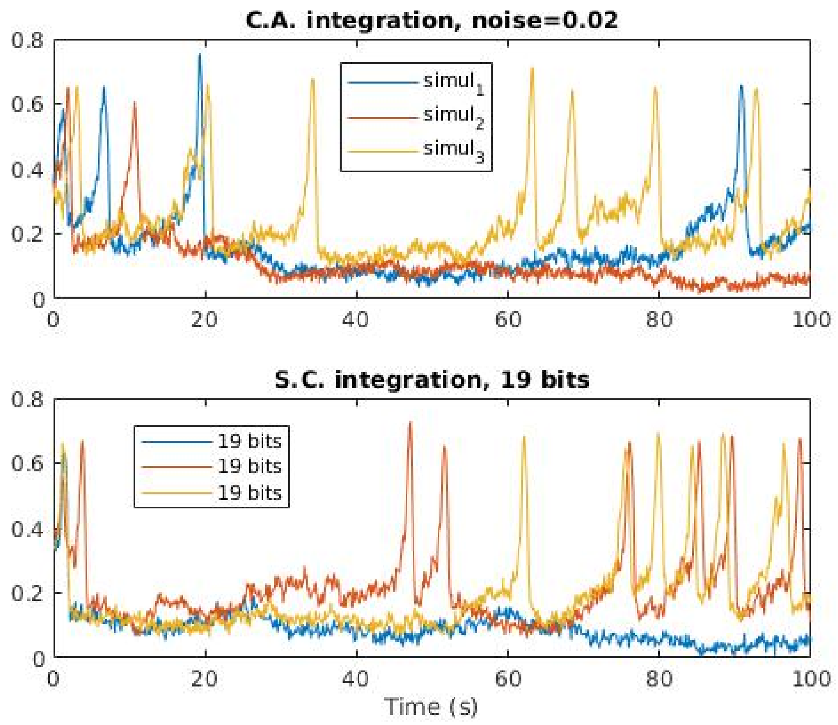

Figure 4. Comparison between three different simulation using the exact solution (top) with $a_\epsilon = 0.02$ noise, and three different simulations using SC and 19 bits (bottom).

corresponding to a given number of bits are shown as a green circle. It can be seen that the tendency is downwards, as expected, with a noise following a power law as in Eq. 16:

$$\epsilon_x = 2^{-\eta N} \tag{16}$$

where η is a parameter that accounts for the effects on nonlinearity and feedback on the system. In our case, η has been found to be $\eta = 1/3.5$.

5. Conclusion

In this paper we have presented a discussion on the implementation of the HR neuron model using Stochastic Computing. This approach has benefits considering the ease of implementation, requiring a very low number of gates, since arithmetic operations can be carried out using simple logic operations, as can be XNOR for multiplication or a multiplexer for addition. In order to implement the equations, we have rewritten them, as stated in [24], to be normalized in the range [-1,1] for all the possible mathematical operations, including values of the constants and the time scale.

These modified equations have been implemented in Matlab and simulated for different number of SCN lengths. It has been found that SC implementation is equivalent to introducing a noise source in the original equations. We have empirically found that the relation between the amplitude of the introduced noise and the length of the SCN follows an inverse power low as this is expressed in Eq. 16.

Related to the speed of a possible ASIC or FPGA realization, we can state that we need 2^N clock cycles to perform a full operation equivalent to a conventional arithmetic implementation. That is, we can do the following number of steps per unit of time:

$$N_O = \frac{2^N}{T_{clk}} \tag{17}$$

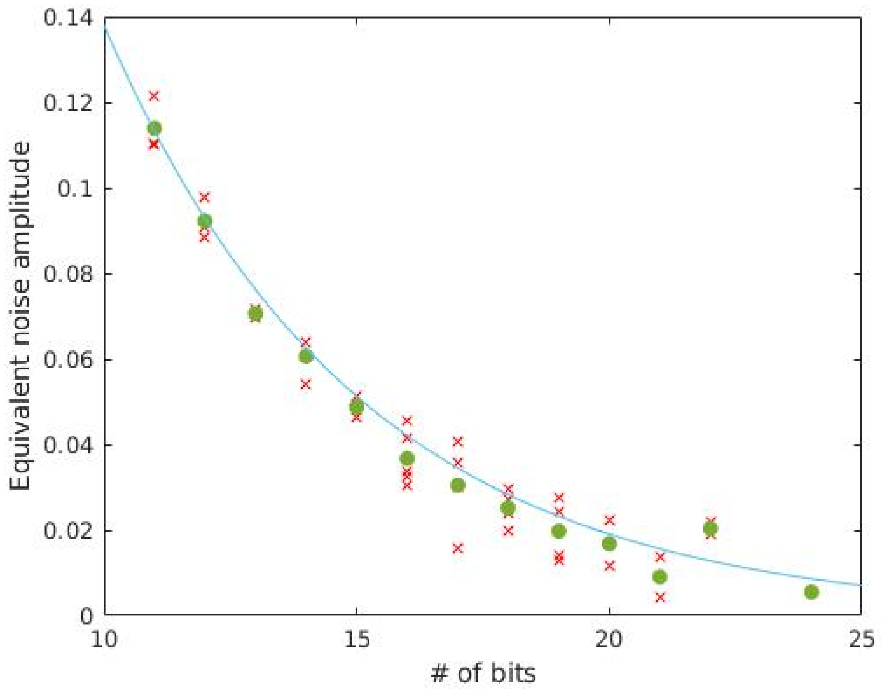

Figure 5. Equivalent noise amplitude for different number of bits. The red x correspond to the noise level of a single simulation; the green o are the average of all the simulations with a given number of bits; the blue line is the fitting of the averages according to Eq. 16, with $\eta = 1/3.5$.

In our case, we have seen that using 19 bits provided a good enough approximation. Thus, assuming a FPGA running at 100MHz and with the same $dt = 0.01$s as used in the simulations, we would need t_O real time seconds to get one second in simulation time:

$$t_O = 2^N T_{clk} \frac{1s}{dt} \tag{18}$$

This results in $t_O \approx 0.52$s. This way, a very simple implementation can provide a x2 speedup with reference to a real time system, including noise. This would pave the way to *in silico* simulation of simple biological systems, since the low number of required gates per neuron allows for complex systems.

However, it has to be noted that the speed of the simulated systems would be closer to the biological originals, which take advantage of parallelism instead of just speed. Although this appears to be a certain drawback, the proposed approach is beneficial for both edge computing of parallel processing systems; especially when we are having in mind neuromorphic-inspired designed systems.

Author Contributions: All the authors participated in all stages of this paper. All authors have read and agreed to the published version of the manuscript.

Funding: Part of this work was funded under the Spanish Ministerio de Economía y Competitividad DPI2017-86610-P and TEC2017-84877-R (also supported by the FEDER program).

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
B2S	Binary to Stochastic
BEN	Binary Encoded Number
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
IoT	Internet of Things
NF	Noise Figure
ODE	Ordinary Differential Equation
RNG	Random Number Generator
SC	Stochastic Computing
SCN	Stochastic Computing Number (see also SEN)
SEN	Stochastic Encoded Number(s)

References

1. Cattell, R.; Parker, A. Challenges for brain emulation: why is building a brain so difficult. *Natural intelligence* **2012**, *1*, 17–31.
2. Makin, S. The four biggest challenges in brain simulation. *Nature* **2019**, p. 7766. doi:10.1038/d41586-019-02209-z.
3. D'Angelo, E.; Jirsa, V. The quest for multiscale brain modeling. *Trends in Neurosciences* **2022**.
4. Open source brain.
5. Blue, C. Could Quantum Computing Revolutionize Our Study of Human Cognition? *APS Observer* **2021**, *34*.
6. Yamaura, H.; Igarashi, J.; Yamazaki, T. Simulation of a Human-Scale Cerebellar Network Model on the K Computer. *Frontiers in Neuroinformatics* **2020**, *14*. doi:10.3389/fninf.2020.00016.
7. Ellingsrud, A.J.; Boullé, N.; Farrell, P.E.; Rognes, M.E. Accurate numerical simulation of electrodiffusion and water movement in brain tissue. *Mathematical Medicine and Biology: A Journal of the IMA* **2021**, *38*, 516–551.
8. Schirner, M.; Kong, X.; Yeo, B.T.; Deco, G.; Ritter, P. Dynamic primitives of brain network interaction. *NeuroImage* **2022**, *250*, 118928. doi:https://doi.org/10.1016/j.neuroimage.2022.118928.
9. Bunruangses, M.; Youplao, P.; Amiri, I.S.; Pornsuwancharoen, N.; Yupapin, P. Brain sensor and communication model using plasmonic microring antenna network. *Optical and Quantum Electronics* **2019**, *51*, 1–10.
10. Amunts, K.; Ebell, C.; Muller, J.; Telefont, M.; Knoll, A.; Lippert, T. The human brain project: creating a European research infrastructure to decode the human brain. *Neuron* **2016**, *92*, 574–581.
11. Schirner, M.; Domide, L.; Perdikis, D.; Triebkorn, P.; Stefanovski, L.; Pai, R.; Popa, P.; Valean, B.; Palmer, J.; Langford, C.; Blickensdörfer, A.; van der Vlag, M.; Diaz-Pier, S.; Peyser, A.; Klijn, W.; Pleiter, D.; Nahm, A.; Schmid, O.; Woodman, M.; Zehl, L.; Fousek, J.; Petkoski, S.; Kusch, L.; Hashemi, M.; Marinazzo, D.; Mangin, J.F.; Flöel, A.; Akintoye, S.; Stahl, B.C.; Cepic, M.; Johnson, E.; Deco, G.; McIntosh, A.R.; Hilgetag, C.C.; Morgan, M.; Schuller, B.; Upton, A.; McMurtrie, C.; Dickscheid, T.; Bjaalie, J.G.; Amunts, K.; Mersmann, J.; Jirsa, V.; Ritter, P. Brain Modelling as a Service: The Virtual Brain on EBRAINS, 2021. doi:10.48550/ARXIV.2102.05888.
12. Sy, M.F.; Roman, B.; Kerrien, S.; Mendez, D.M.; Genet, H.; Wajerowicz, W.; Dupont, M.; Lavriushev, I.; Machon, J.; Pirman, K.; others. Blue Brain Nexus: An open, secure, scalable system for knowledge graph management and data-driven science. *Semantic Web*, pp. 1–31.
13. Camuñas-Mesa, L.A.; Linares-Barranco, B.; Serrano-Gotarredona, T. Neuromorphic spiking neural networks and their memristor-CMOS hardware implementations. *Materials* **2019**, *12*, 2745.
14. Höppner, S.; Mayr, C. Spinnaker2-towards extremely efficient digital neuromorphics and multi-scale brain emulation. *Proc. NICE* **2018**.
15. Schemmel, J.; Kriener, L.; Müller, P.; Meier, K. An accelerated analog neuromorphic hardware system emulating NMDA- and calcium-based non-linear dendrites. 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017, pp. 2217–2226.
16. Gao, M.; Wang, Q.; Arafin, M.T.; Lyu, Y.; Qu, G. Approximate computing for low power and security in the internet of things. *Computer* **2017**, *50*, 27–34.
17. Von Neumann, J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies* **1956**, *34*, 43–98.
18. Moons, B.; Verhelst, M. Energy-Efficiency and Accuracy of Stochastic Computing Circuits in Emerging Technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **2014**, *4*, 475–486. doi:10.1109/JETCAS.2014.2361070.
19. Li, S.; Glova, A.O.; Hu, X.; Gu, P.; Niu, D.; Malladi, K.T.; Zheng, H.; Brennan, B.; Xie, Y. SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator. MICRO, 2018, pp. 696–709.
20. Toral, S.; Quero, J.; Franquelo, L. Stochastic pulse coded arithmetic. Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on. IEEE, 2000, Vol. 1, pp. 599–602.

21. Marin, S.T.; Reboul, J.Q.; Franquelo, L.G. Digital stochastic realization of complex analog controllers. *IEEE Transactions on Industrial Electronics* **2002**, *49*, 1101–1109.
22. Khanday, F.A.; Akhtar, R. Reversible stochastic computing. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* **2020**, *n/a*, e2711, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/jnm.2711]. doi:10.1002/jnm.2711.
23. Camps, O.; Picos, R.; de Benito, C.; Al Chawa, M.M.; Stavrinides, S.G. Effective accuracy estimation and representation error reduction for stochastic logic operations. 2018 7th Int. Conf. on Modern Circuits and Systems Technologies (MOCAST). IEEE, 2018, pp. 1–4.
24. Camps, O.; Stavrinides, S.G.; Picos, R. Stochastic computing implementation of chaotic systems. *Mathematics* **2021**, *9*, 375.
25. Liu, S.; Gross, W.J.; Han, J. Introduction to Dynamic Stochastic Computing. *IEEE Circuits and Systems Magazine* **2020**, *20*, 19–33.
26. Hindmarsh, J.; Rose, R. A model of the nerve impulse using two first-order differential equations. *Nature* **1982**, *296*, 162–164.
27. Hindmarsh, J.L.; Rose, R. A model of neuronal bursting using three coupled first order differential equations. *Proceedings of the Royal society of London. Series B. Biological sciences* **1984**, *221*, 87–102.
28. Mukae, J.; Totoki, Y.; Suemitsu, H.; Matsuo, T. Parameter and input estimation in Hindmarsh-Rose neuron by adaptive observer. 2011 IEEE/SICE International Symposium on System Integration (SII), 2011, pp. 1090–1095. doi:10.1109/SII.2011.6147601.
29. Fradkov, A.L.; Kovalchukov, A.; Andrievsky, B. Parameter Estimation for Hindmarsh–Rose Neurons. *Electronics* **2022**, *11*, 885.
30. Beyhan, S. Affine TS fuzzy model-based estimation and control of Hindmarsh–Rose neuronal model. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2017**, *47*, 2342–2350.