Margin Propagation based Analog Soft-Gates for Probabilistic Computing

Ankita Nandi[†], Pratik Kumar[†], Shantanu Chakrabartty[‡], and Chetan Singh Thakur[†]
{ankitanandi, pratikkumar, csthakur}@iisc.ac.in, shantanu@wustl.edu
[†] NeuRonICS Lab, Department of Electronic Systems Engineering, Indian Institute of Science, Bengaluru 560012, India
[‡]Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA

***Abstract*—Soft computing gates offer a promising approach for efficient and parallel processing of probabilistic signals. These gates are widely used in Bayesian networks and various machine learning models. However, unlike digital logic gates, the design and scaling of analog Soft-Gates is challenging due to analog artifacts, i.e., sensitivity to biasing, mismatch, and temperature variations. In this paper, we present a systematic framework for designing analog Soft-Gates that leverage the bias and temperature scalability of the Margin Propagation principle. Specifically, the paper proposes an adaptive design strategy to alleviate mismatch artifacts and to trade-off probabilistic computational accuracy, area efficiency, and power consumption. We demonstrate the design synthesis of a Soft-Gate and apply it to error correction decoding and filtering tasks. The reported Mean Square Error of the Soft-Gate is less than 10^{-2}, indicating its accuracy in probabilistic computations. For edge filtering applications, the proposed Soft-Gates can achieve an average Structural Similarity Index of 0.95. The estimated energy consumption in 180nm CMOS technology is in the order of pico-Joules, validating the gate's energy efficiency.**

***Index Terms*—Analog Soft-Gates, Generalized Margin Propagation, Shape-based Analog Computing, Probabilistic Computation**

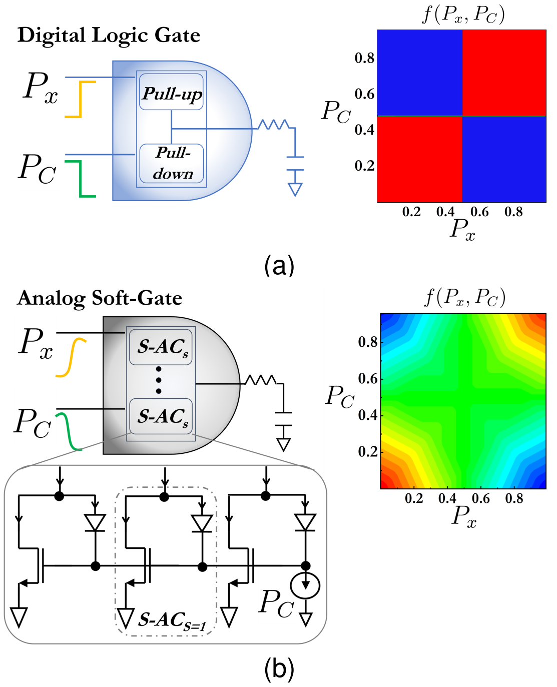


Fig. 1. A visual representation showcasing: (a) Traditional CMOS digital logic gate operating at *logic high* and *logic low*; (b) Analog Soft-Gate design based on the Generalized Margin Propagation [3], illustrating analog probabilistic inputs (P_x & P_c) and output, $f(P_x, P_C)$ where the logic block is implemented using modular shape-based analog compute (S-AC) unit.

I. Introduction

Probabilistic information processing is crucial for many applications, including stochastic logical operations [1], error correction algorithms [2], filtering [3], and machine learning [4]. To design such applications efficiently, Soft-Gates can be designed using analog techniques [5] for better energy efficiency, reduced area, and optimized resource utilization as it can leverage the continuous nature of analog circuitry. Fig. 1 represents an illustrative comparison between conventional digital gates (Fig. 1a) and the proposed analog Soft-Gate (Fig. 1b).

Recent research has focused on exploiting their potential in synthetic biology, aiming to predict subject behavior in advance for reducing the cost of wet lab experiments [6], [7]. However, inherent inaccuracies stemming from CMOS process variations, temperature dependencies, and noise compromise the reliability of probabilistic signals within the implemented analog Soft-Gate. Additionally, the non-modularity and non-scalability of analog designs make their adoption challenging.

In this work, we propose analog soft computing gates based on a generalization of the Margin Propagation (MP) principle [8] called Shape-based Analog Computing (S-AC) [3], [9]. Similar to digital logic circuits, MP-based analog circuits have been shown to be robust to variations in transistor biasing [8] and variations in temperature [3], hence they are ideal candidates for implementing analog Soft-Gates. Furthermore, in [3], it was shown that MP-based circuits are scalable across different process nodes, which could make the designs scalable and portable. In this paper, we show that by adjusting the design parameters, the proposed analog Soft-Gates can be tailored to balance computational accuracy and resource utilization. We also demonstrate the design synthesis of a Soft-Gate and its application in decoding Low-Density Parity Check (LDPC) codes [10] and edge filtering [11]. The experimental results showcase the efficacy of our modular framework in achieving efficient probabilistic computations while maintaining satisfactory levels of accuracy. In comparison to traditional analog implementations, our soft computing gate exhibits notable improvements in area and power efficiency, underscoring the advantages of our proposed design methodology. The key

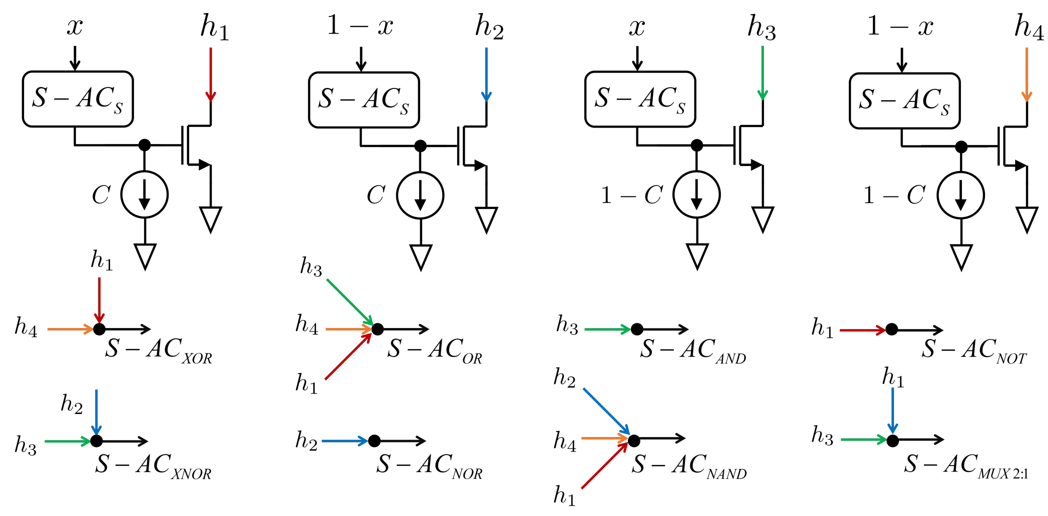


Fig. 2. Architecture for 2-input S-AC adaptive Soft-Gates and S-AC multiplexer. The input currents are denoted as x and C, while the desired output currents are represented by $S-AC_{XOR}$, $S-AC_{OR}$, $S-AC_{AND}$, $S-AC_{NOT}$, $S-AC_{XNOR}$, $S-AC_{NOR}$, $S-AC_{NAND}$, and $S-AC_{MUX}$.

contributions of this work are as follows:

- Design formulation of analog Soft-Gates based on the Generalized Margin Propagation (GMP) principle.
- Systematic synthesis for Soft-Gates, covering the step-by-step process from specification to netlist generation.
- Application of the proposed gates in error correction algorithms and image processing.

The paper is organized as follows: Section II describes the background of the GMP framework. Section III presents the mathematics, design implementation, and results of the proposed adaptive Soft-Gates. Section IV discusses the Design Flow Synthesis of the proposed gates. The applications are presented in Section V and the conclusion in Section VI.

II. Overview of Generalized Margin Propagation

In this section, we briefly describe the Generalized Margin Propagation (GMP) framework presented in [3]. This mathematical framework aimed to create a robust, non-linear monotonic shape using linear splines. The function creating this shape is given by

$$\sum\nolimits_{i=1}^{N}\sum\nolimits_{j=1}^{S}[x_{i,j}-h]_{+} \quad = C \tag{1}$$

Here, $h(.)$ is a function of a matrix whose input elements are $x_{i,j}, i = 1, .., N; j = 1, .., S$ and C is a design hyperparameter. N denotes the number of input elements, and S denotes the number of splines used for approximation. The notation $[\cdot]_{+}$ denotes the rectification function. For the special case of $N = 1$ and $S = 1$, (1) reduces to Margin Propagation (MP) function [8] and can be given as

$$[x-h]_{+} = C \tag{2}$$

The mapping of (1) can also be written in Shape-based Analog Computing (S-AC) form as

$$\sum\nolimits_{i=1}^{N}\sum\nolimits_{j=1}^{S} f(x_{i,j}, h) = C, \forall i = 1, .., N, \forall j = 1, .., S \tag{3}$$

TABLE I
Normalized Response of S-AC Units Shown in Fig. 2

x	C	h_1	h_2	h_3	h_4
0	0	0	1	0	0
0	1	0	0	0	1
1	0	1	0	0	0
1	1	0	0	1	0

*Boundary conditions where 1 shows the maximum applied current I_{max} and 0 shows the minimum applied current I_{min}.

where the function $f(\cdot,\cdot)$ models the forward and reverse currents of the MOSFETS [3]. We have utilized this function $f(\cdot,\cdot)$ from (3) for various S values to implement Soft-Gates in Section III.

III. Adaptive Soft-Computing Gates

Here, we show the mathematical framework for designing Soft-Gates along with its circuit implementation and results. The design methodology is presented for 2-input gates but can be generalized for multiple inputs and cascaded gates as well. Table I shows the responses (h_1, h_2, h_3 and h_4) of different S-AC circuit combinations shown in Fig. 2 (where CMOS implementation of each S-AC$_S$ cell is shown in Fig. 1b) for the four boundary (logic) conditions. Here the inputs to the circuit (currents) are denoted by $P_x(1) \cong x$ and $P_C(1) \cong C$. It may be noted that one or more combinations of h_1, h_2, h_3, and h_4 can effectively represent any logic gate. Using this observation, the detailed construction of each Soft-Gate is explained below.

A. Mathematical Formulation

1) Soft XOR and XNOR Gates: If $P_x(1) \cong x$ and $P_C(1) \cong C$ are the two inputs, then the sum-of-product (SOP) [12] form of expression for XOR gate can be used to represent its probabilistic outcome P_{XOR} as

$$P_{XOR}(1) = P_x(1)P_C(0) + P_x(0)P_C(1) \tag{4}$$

It can be observed from Table I that the outcome of the XOR gate can be realized by combining the response of column h_1

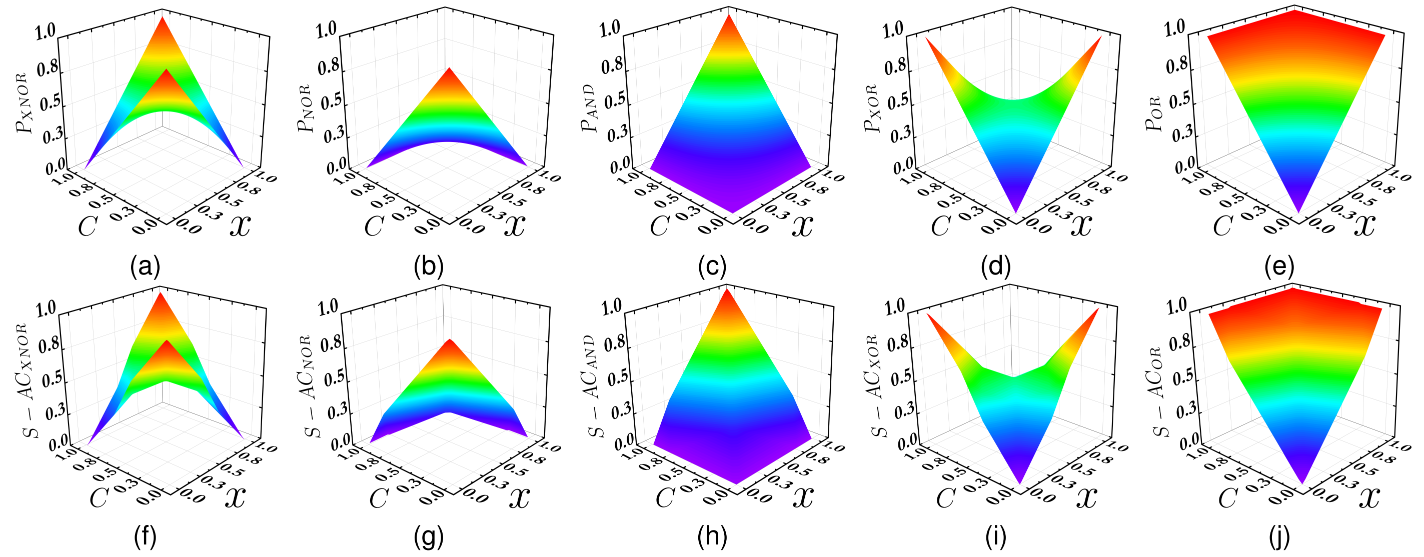


Fig. 3. Surface plot for ideal and Soft-Gates implemented with $S = 3$ illustrated for: (a) XNOR gate implementing (6); (b) NOR gate implementing (8); (c) AND gate implementing (12); (d) XOR gate implementing (4); (e) OR gate implementing (10); (f) $S-AC_{XNOR}$ gate implementing (7); (g) $S-AC_{NOR}$ gate implementing (9); (h) $S-AC_{AND}$ gate implementing (13); (i) $S-AC_{XOR}$ gate implementing (5) and, (j) $S-AC_{OR}$ gate implementing (11).

and h_4, where h_1 and h_4 are the responses of the functions $f(x, C) = h_1$ and $f(1-x, 1-C) = h_4$ respectively. It can hence be concluded that $h_1 + h_4$ can represent the equivalent $S-AC_{XOR}$ gate and can be written as

$$S-AC_{XOR} \cong f(x, C) + f(1-x, 1-C) \tag{5}$$

Similarly, the probabilistic outcome of the XNOR gate is given by the sum of its min-terms in (6), and its S-AC equivalent is given by (7).

$$P_{XNOR}(1) = P_x(0)P_C(0) + P_x(1)P_C(1) \tag{6}$$

$$S-AC_{XNOR} \cong f(x, 1-C) + f(1-x, C) \tag{7}$$

2) Soft NOR and OR Gates: The concept used in XOR can be extended to the OR and the NOR gates. The probabilistic NOR gate is true when both the inputs are *low* and can be computed by (8).

$$P_{NOR}(1) = P_x(0)P_C(0) \tag{8}$$

The equivalent S-AC based NOR gate from Table I and Fig. 2 can thus be written as in (9).

$$S-AC_{NOR} = f(1-x, C) \tag{9}$$

The OR gate is given by (10) as the complement of the NOR gate and Thus can be constructed using the complement function explained in a later section.

$$P_{OR}(1) = 1 - P_{NOR}(1) \tag{10}$$

However, Table I and Fig. 2 can be used to construct S-AC$_{OR}$ without the complement function as expressed below.

$$S-AC_{OR} = f(1-x, 1-C) + f(x, C) + f(x, 1-C) \tag{11}$$

3) Soft AND and NAND Gates: The formulation of probabilistic AND is given by (12), and its S-AC equivalent by (13).

$$P_{AND}(1) = P_x(1)P_C(1) \tag{12}$$

$$S-AC_{AND} = f(x, 1-C) \tag{13}$$

We can similarly construct the NAND gate as under:

$$P_{NAND}(1) = 1 - P_{AND}(1) \tag{14}$$

$$S-AC_{NAND} = f(1-x, 1-C) + f(x, C) + f(1-x, C) \tag{15}$$

4) Soft NOT Gate: The soft inverter can be constructed by passing the maximum allowable current as x i.e., $x = I_{max}$, and C as the respective input that needs to be inverted.

5) Soft Multiplexer: $2:1$ multiplexer based on S-AC can also be realized as a combination of two S-AC$_s$ blocks (Fig. 2) implementing $f(x_1, C)$ and $f(x_2, 1-C)$ where x_1 and x_2 being two different inputs. If C is utilized as a select line and C is very small, then x_1 is selected. However, if C is large, then x_2 is selected. Thus they are combined to make the multiplexer.

B. CMOS Circuit Implementation

We use S-AC$_S$ (Fig. 1b) blocks for $N = 1$ and $S = 3$ in the design of each gate. We use four different input configurations as explained in Table I. To represent each pair of input combinations, we require one S-AC block per spline. These blocks can then be combined to represent the output of the gate. For instance, as seen in Fig. 2, $h_1 + h_4 = f(x, C) + f(1-x, 1-C)$ is the output of the S-AC$_{XOR}$ gate while its complementary output is represented using $h_2 + h_3$. Similarly, it is observed from Table I that $f(x, 1-C)$ is the S-AC operation of AND gate, which is represented by h_3 in Fig. 2. The same applies

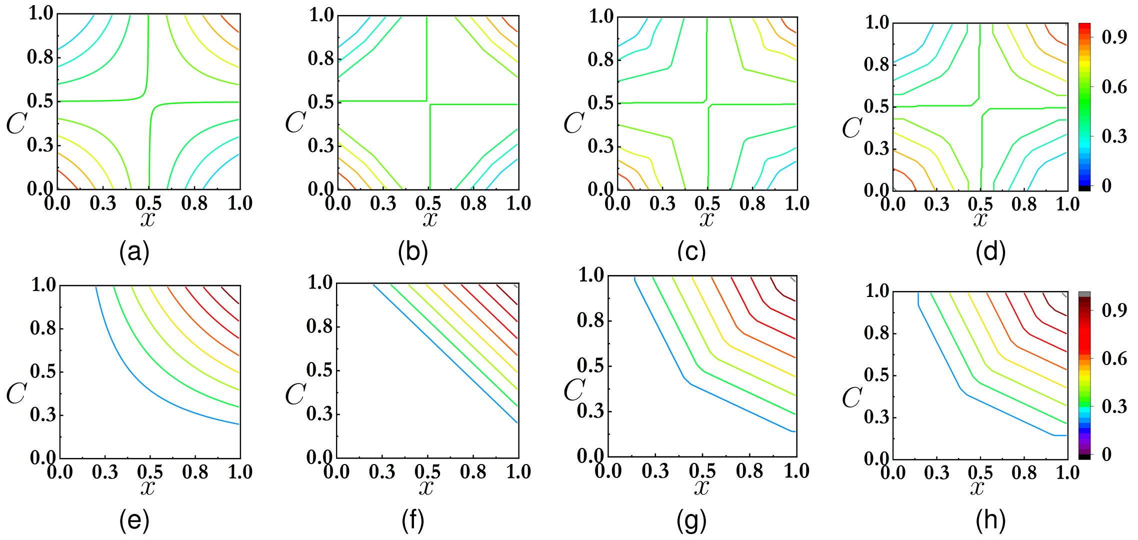


Fig. 4. Effect of varying number of splines S in the implemented design and comparison with ideal, where the characteristics are shown for (a) Ideal probabilistic XNOR Gate; (b) S-AC$_{XNOR}$ gate for S = 1; (c) S-AC$_{XNOR}$ gate for S = 2, and (d) S-AC$_{XNOR}$ gate for S = 3; (e) Ideal probabilistic AND Gate; (f) S-AC$_{AND}$ gate for S = 1; (g) S-AC$_{AND}$ gate for S = 2, and (h) S-AC$_{AND}$ gate for S = 3. The figures demonstrate the improvement in accuracy with the increase in the number of splines (S).

to the other gates. The soft inverter, however, requires the x input to S-AC as the maximum allowable current, that is $h_1 = f(I_{max}, C)$ where $h_1 = \bar{C}$. The multiplexer is represented by $h_1 + h_3$ but for different x (viz. x_1 and x_2) as they represent different inputs of the $2:1$ multiplexer.

C. Result

Characteristic surface plot and their comparison with ideal gates of various implemented Soft-Gate for design parameters $N = 1$ and $S = 3$ are shown in Fig. 3, while their error, area, and energy metrics are presented in Table II. It must be noted that although the results have been reported for $S = 3$, the value of S can be reduced to fit into power and area budgets at the cost of accuracy and vice versa. Fig. 3 reveals a close match with ideal except for minor artifacts arising from spline tuning points. The spline-induced artifacts have been summarized in Table II. We report the Maximum Absolute Error, Mean Square Error, and Mean Absolute Error of the S-AC$_{XNOR}$, S-AC$_{AND}$ and S-AC$_{NOR}$ gates for $S = 3$ indicating that they can be further tuned for more accuracy as explained in the next section. It can be noted that the S-AC$_{NOT}$ plot has negligible deviation as the S-AC operation essentially performs a subtraction, as is the case for a probabilistic complement; thus, the errors of the complement gates can be estimated to be of a similar order. The energies of $S = 3$ probabilistic gates can be found to be in the order of pico-Joules for the given accuracy, as seen in Table II.

D. Adaptive Nature of the S-AC Gates

S-AC Gates can be adapted to cater to the accuracy requirements of specific applications. Fig. 4 illustrates the effect of the number of splines S in the form of multiple line segments in the contour. Fig. 4a shows the contours of an ideal probabilistic XNOR gate. Fig. 4b, which is an estimation using only single segments, is the $S = 1$ representation of the S-AC$_{XNOR}$ gate. Fig. 4c-4d, which is an estimation using two and three segments, are the $S = 2$ and $S = 3$ representations of the S-AC$_{XNOR}$ gate respectively. Similarly, the $S = 1, 2, 3$ for S-AC$_{AND}$ gate can be visualized by the contours in Fig. 4f-4h. It can be clearly observed that as the splines increase, the plots become more accurate.

TABLE II
S-AC Soft-Gate Error Estimation @$S = 3$

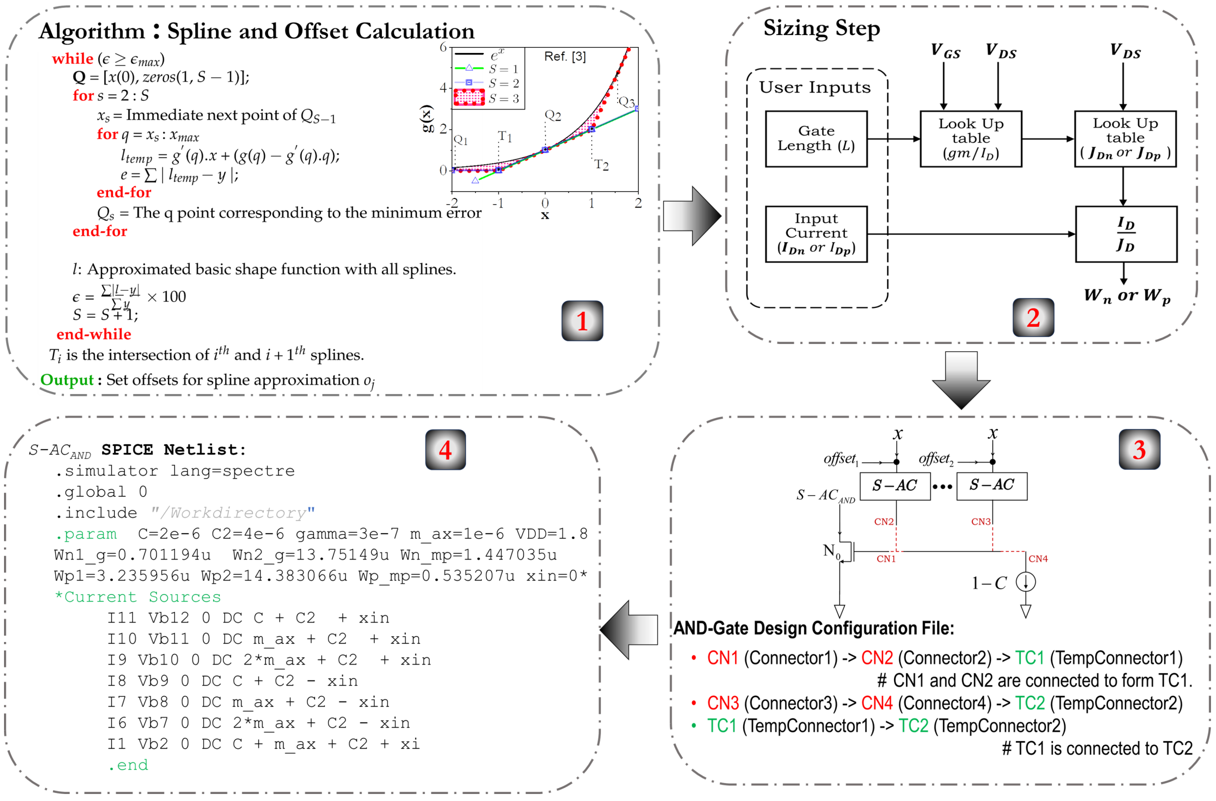
Parameters	S-AC$_{XNOR}$	S-AC$_{AND}$	S-AC$_{NOR}$
Maximum Absolute Error	0.0539	0.1038	0.1054
Mean Square Error	5.73E-04	0.0022	0.0024
Mean Absolute Error	0.0197	0.0383	0.0402
Energy* (pJ)	15.9	4.47	4.5
Area† (μm^2)	2657.22	116.13	116.13

*Reported at Strong Inversion for $V_{supply} = 1.1V$.
†Values at $180nm$ technology node including peripheral circuits.

IV. Design Flow Synthesis of Analog Soft-Gates

This section presents the synthesis flow for the analog Soft-Gates in brief, covering the entire process from specification to netlist. All the Soft-Gates illustrated in Fig. 2 rely on the S-AC compute unit as their fundamental computational block. These modular analog blocks can be arranged in different patterns along with any current mirrors to implement the aforementioned Soft-Gates. The following steps elaborate on the design steps:

1) **Approximation Step:** This crucial step in the synthesis flow involves approximating a given monotonic, non-linear shape by fitting one or multiple linear splines to represent the shape. This multi-spline-based approximation of a non-linear function employs various mathematical techniques to construct a smooth curve that passes through a set of given points known as tuning and tangent points and has been explained in [3]. By optimizing the number of splines and



Algorithm : Spline and Offset Calculation

```
while ($\epsilon \geq \epsilon_{max}$)
  $\mathbf{Q} = [x(0), zeros(1, S-1)]$;
  for $s = 2 : S$
    $x_s$ = Immediate next point of $Q_{S-1}$
    for $q = x_s : x_{max}$
      $l_{temp} = g'(q).x + (g(q) - g'(q).q)$;
      $e = \sum | l_{temp} - y |$;
    end-for
    $Q_s$ = The q point corresponding to the minimum error
  end-for

  $l$: Approximated basic shape function with all splines.
  $\epsilon = \frac{\sum|l-y|}{\sum y} \times 100$
  $S = S + 1$;
end-while
$T_i$ is the intersection of $i^{th}$ and $i+1^{th}$ splines.
Output : Set offsets for spline approximation $o_j$
```

S-AC_{AND} **SPICE Netlist:**

```
.simulator lang=spectre
.global 0
.include "/Workdirectory"
.param  C=2e-6 C2=4e-6 gamma=3e-7 m_ax=1e-6 VDD=1.8
Wn1_g=0.701194u  Wn2_g=13.75149u Wn_mp=1.447035u
Wp1=3.235956u Wp2=14.383066u Wp_mp=0.535207u xin=0*
*Current Sources
     I11 Vb12 0 DC C + C2  + xin
     I10 Vb11 0 DC m_ax + C2  + xin
     I9 Vb10 0 DC 2*m_ax + C2  + xin
     I8 Vb9 0 DC C + C2 - xin
     I7 Vb8 0 DC m_ax + C2 - xin
     I6 Vb7 0 DC 2*m_ax + C2 - xin
     I1 Vb2 0 DC C + m_ax + C2 + xi
     .end
```

AND-Gate Design Configuration File:

- CN1 (Connector1) -> CN2 (Connector2) -> TC1 (TempConnector1)
  # CN1 and CN2 are connected to form TC1.
- CN3 (Connector3) -> CN4 (Connector4) -> TC2 (TempConnector2)
- TC1 (TempConnector1) -> TC2 (TempConnector2)
  # TC1 is connected to TC2

Fig. 5. Illustration of the step-by-step process for designing and synthesizing Analog Soft-Gates: (1) **Approximation Step:** Calculates Design Offsets (O_j) and Splines using [3]; (2) **Design Sizing Step:** Involves designing N-Type and P-Type S-AC compute units using the $\frac{g_m}{I_d}$ based Look-up Table design methodology [13] and MATLAB; (3) **Interconnection Step:** Extracts information from the S-AC Design Configuration File (.dcf) to establish interconnections between various S-AC based AND Soft-Gate units; (4) **Netlist Generation:** Generates the SPICE netlist for the computational module of the analog Soft-Gates.

their tuning and tangent points calculated by the Spline-Offset calculation algorithm, the approximation error can be minimized, as shown in Step 1 of Fig. 4.

2) **S-AC Design Sizing Step:** This step employs the MATLAB flow mentioned in [13] for automated sizing of S-AC units and current mirrors according to the specifications. The information on splines, offset, and S-AC cells obtained from the previous step are utilized in the $\frac{g_m}{I_D}$ based look-up table data for the pre-defined technology node to create the S-AC cells (Step 2 of Fig. 4) for subsequent steps.
3) **Interconnection Step:** This is followed by interconnecting S-AC blocks to create a Soft-Gate using the ”S-AC Design Configuration file (.dcf)”. This file contains information about how the S-AC blocks interconnection with itself and current mirrors to implement a particular functional gate.
4) **Netlist Generation:** The final step is to generate a SPICE netlist that includes information about the S-AC cells, their interconnections, and the values of various parameters, such as transistor sizes, resistance, capacitance values, and operating frequencies. To generate the netlist, the parameters of the cells are taken as variables, and the netlist is constructed by replacing these variables with appropriate values based on the user-specified requirements. The netlist can then be used for further simulations and verification to ensure that the circuit meets the desired specifications.

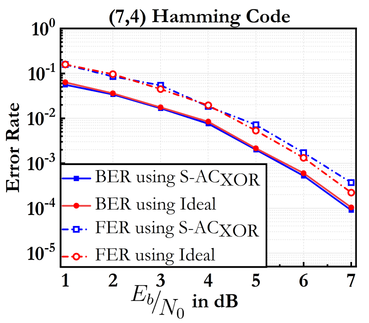
V. Application Case-Study

A. *Error Correction Decoding of LDPC using S-AC$_{XOR}$*

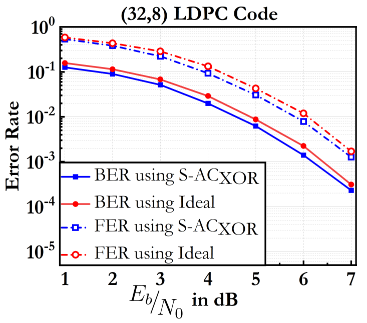
To analyze the utility of the Soft-Gates, we employ the S-AC$_{XOR}$ design for decoding Low-Density Parity Check (LDPC) Codes [10] using the Sum-Product Algorithm [14]. This traditional Algorithm uses complex functions like $log \prod tanh(\cdot)$ to analyze the extrinsic reliability (probability) of the bit. We reformulate the evaluation of the parity check constraints in the Sum-Product decoding and replace $log \prod tanh(\cdot)$ with the S-AC$_{XOR}$ gate. Fig. 6a and Fig. 6b represent close compliance of error rates (bits and frame) for an Additive White Gaussian Noise (AWGN) channel, using the conventional computation and the proposed gates.

B. *Edge Detection using S-AC$_{AND}$*

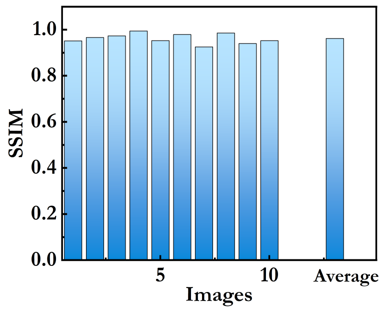
Edge Detection employs AND operation for the convolution of the pixel intensities with the kernel. If the intensities are normalized representations, then a soft AND operation is suitable. We perform the convolution using the Sobel 3×3 kernel and compare the results of S-AC$_{AND}$ convolution with the ideal. In order to quantify our results, we use Structural Similarity (SSIM) as a metric of the accuracy of the image [15], [16] on ten randomly chosen images from the MATLAB Image Processing Toolbox. The Structural Similarity Index (SSIM) is generally $\geq$ 0.93 (Fig. 6c). One of the images (Fig. 6d)



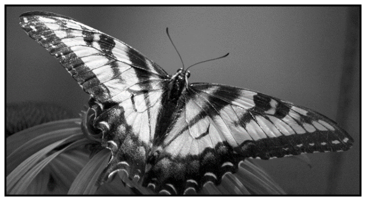
(a)



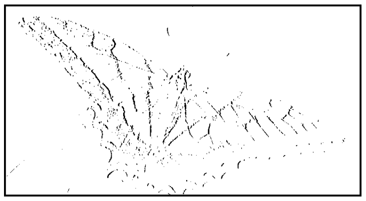
(b)



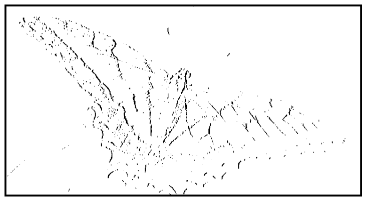
(c)



(d)



(e)



(f)

Fig. 6. Applications of S-AC Soft-Gate. Fig. (a)-(b) shows the Bit Error Rate (BER) and the Frame Error rate (FER) using S-AC$_{XOR}$ for (a) $(7,4)$ Hamming Code; (b) $(32,8)$ Regular LDPC Code, computed using the Sum-Product Decoding algorithm through ideal and S-AC$_{XOR}$ computations. Fig. (c)-(f) show edge Detection using the Sobel horizontal filter using S-AC$_{AND}$; (c) Structural Similarity (SSIM) of ten randomly chosen images from the MATLAB repository along with their average SSIM using the S-AC$_{AND}$ gate; Edge Detection for a single image where (d) is the Original Image; (e) Resultant image after ideal software computation; (f) Resultant image after S-AC$_{AND}$ Computation.

with the ideal and S-AC outcome is presented in Fig. 6e and Fig. 6f, respectively.

VI. Conclusion & Future Work

In this work, we presented a modular and systematic framework for designing analog Soft-Gates based on the GMP principle [3]. The proposed designs are modular, cascadable, process, and bias scalable, thereby addressing traditional Soft-Gate limitations. The proposed Soft-Gates can be tailored for various applications for varying accuracy and resource utilization. We validated the approach through synthesis and applied it in error correction decoding and image processing. Future research will explore the applicability in biology and machine learning.

Acknowledgment

The authors would like to acknowledge the joint IISc-WashU MoU to facilitate the collaboration between the two institutions. This work is also supported by the Department of Science and Technology of India (SERB CRG/2021/005478, DST/IMP/2018/000550), Indian National Academy of Engineering (SP/INAE-22-2106) and the Prime Minister's Research Fellowship (PMRF ID: 0200994).

References

[1] A. Alaghi and J. P. Hayes, "On the Functions Realized by Stochastic Computing Circuits," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, (New York, NY, USA), p. 331–336, Association for Computing Machinery, 2015.

[2] S. Sharifi Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of ldpc codes," *IEEE Communications Letters*, vol. 10, no. 10, pp. 716–718, 2006.

[3] P. Kumar, A. Nandi, S. Chakrabartty, and C. S. Thakur, "Process, Bias, and Temperature Scalable CMOS Analog Computing Circuits for Machine Learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 1, pp. 128–141, 2023.

[4] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, pp. 131–163, 1997.

[5] B. Vigoda, *Analog Logic: Continuous-Time Analog Circuits for Statistical Signal Processing*. PhD thesis, Program in Media Arts and Sciences, School of Architecture and Planning, Massachusetts Institute of Technology, 2003.

[6] E. Agliari, M. Altavilla, A. Barra, L. Dello Schiavo, and E. Katz, "Notes on stochastic (bio)-logic gates: computing with allosteric cooperativity," *Scientific reports*, vol. 5, no. 1, p. 9415, 2015.

[7] D. Sanassy, H. Fellermann, and e. a. Krasnogor, "Modelling and Stochastic Simulation of Synthetic Biological Boolean Gates," in *2014 IEEE Intl Conf on HPCC,CSS,ICESS*, pp. 404–408, 2014.

[8] M. Gu and S. Chakrabartty, "Synthesis of Bias-Scalable CMOS Analog Computational Circuits Using Margin Propagation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 2, pp. 243–254, 2012.

[9] P. Kumar, A. Nandi, S. Chakrabartty, and C. S. Thakur, "Bias-Scalable Near-Memory CMOS Analog Processor for Machine Learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 312–322, 2023.

[10] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[11] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.

[12] M. M. Mano, *Digital logic and computer design*. Pearson Education India, 2017.

[13] P. G. Jespers and B. Murmann, *Systematic design of analog CMOS circuits*. Cambridge University Press, 2017.

[14] S. J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*. Cambridge University Press, 2009.

[15] V. Mishra, N. Hassan, A. Mehta, and U. Chatterjee, "DARK-Adders: Digital Hardware Trojan Attack on Block-based Approximate Adders," in *2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID)*, pp. 371–376, 2023.

[16] A. Hore and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, Aug 2010.