Theory and Implementation of Process and Temperature Scalable Shape-based CMOS Analog Circuits

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

Abstract—Analog computing is attractive to its digital counterparts due to its potential for achieving high compute density and energy efficiency. However, the device-to-device variability and challenges in porting existing designs to advance process nodes have posed a major hindrance in harnessing the full potential of analog computations for Machine Learning (ML) applications. This work proposes a novel analog computing framework for designing an analog ML processor similar to that of a digital design where the designs can be scaled and ported to advanced process nodes without architectural changes. At the core of our work lies shape-based analog computing (S-AC). It utilizes device primitives to yield a robust proto-function through which other non-linear shapes can be derived. S-AC paradigm also allows the user to trade off computational precision with silicon circuit area and power. Thus allowing users to build a truly power-efficient and scalable analog architecture where the same synthesized analog circuit can operate across different biasing regimes of transistors and simultaneously scale across process nodes. As a proof of concept, we show the implementation of commonly used mathematical functions for carrying standard ML tasks in both planar CMOS 180nm and FinFET 7nm process nodes. The synthesized Shape-based ML architecture has been demonstrated for its classification accuracy on standard data sets at different process nodes.

Index Terms—Machine Learning, Process Scalability, Analog-Standard Cells, Analog Approximate Computing.

I. INTRODUCTION

MACHINE Learning (ML) algorithms are characterized by extensive use of high-performance computational resources, with memory footprints, compute loads and energy costs that are all quite large. However, their implementation from data centers to edge devices are subject to significant resource and energy constraints. To overcome this challenge, numerous works have reported the advantages of approximate computing to acquire a boost in performance parameters at the cost of minimal loss of accuracy [1], [2]. Most of the works done on these lines are digital in nature and have shown little significance in terms of power density and performance benefits. In this regard, the use of analog ML could offer a drastic improvement in performance parameters. Such analog designs can exploit computational primitives inherent to the MOS device for delivering state-of-the-art power, performance, and area benefits.

Today, with continuous down-scaling of IC technology, where each advanced node offers a boost in performance, the trend is towards implementing ML designs in sub-10nm. However, this is specifically challenging in analog for several reasons: (a) At lower technology, the voltage headroom available is significantly less, which makes the effective range of operation limited. Hence the design becomes more susceptible to noise. It can also be analyzed from Fig. 1 that as we move to a more advanced node, Moderate Inversion (MI) becomes more prominent and occupies a significant operating range of the total voltage supply range for that technology. MI is also an optimal region that offers the best trade-off between energy efficiency and speed as the $g_m/I_d \cdot f_T$ curve peaks in this region as seen in Fig. 1. So for a truly scalable analog design, one would need the circuit to operate efficiently in all regions, including MI, (b) Analog circuit techniques that used to work at higher technology nodes can no longer be applied to sub-10nm nodes. This is because the lower process technologies such as FinFETs were mostly designed for digital circuits [3], (c) Analog designs do not operate on...
the principle of thresholding, unlike digital designs. Thus the output can therefore be easily affected by non-idealities. In contrast, the robustness of digital design also comes from noise and timing margins. Digital designs work on thresholding and hence are immune to noise. Moreover, such designs are synchronous to a clock whose frequency can be varied, hence immune to timing and delay variation. To achieve the same in analog, the designs need to at least be robust enough within a user-defined boundary. (d) Analog designs are stable only in their defined biases. They tend to lose performance and functionality if operated beyond their defined specifications or operating regimes. (e) Analog designs lack the power of digital synthesis because they do not have the robustness of standard cell libraries similar to digital designs.

To overcome some of the above challenges in analog, various works have been presented in the literature on the scalability of analog circuits. Depending on the goals of scaling, some focus on the optimization of area, SNR, dynamic range, bandwidth, and power for a fixed building block [4], [5], [6] while others have utilized principles such as translinear to design their circuits [7], [8], [9]. However, many of these designs are limited to their defined biases and process technology node. Our work aims to address this major shortcoming of process technology scalability, bias scalability, and modularity in analog designs. We here propose a synthesizable and scalable analog computing framework that can leverage the benefits of both analog and digital designs.

At the core of our work lies shape-based analog computing. Shape-based analog computations are a class of mathematical computations that aid in approximating a solution [10] to the desired function. The paradigm of the proposed shape-based analog computing (S-AC) is dedicated to the idea of a fundamental bias- and process- robust shape function through which other shapes can be obtained using simple translation, inversion, addition, and subtraction, as shown in [11]. S-AC focuses only on the shape of the desired computations where the hardware is designed solely using universal conservation principles. This allows the framework to be scalable across process nodes. Furthermore, the bias independent feature of the shape function allows the S-AC units to operate at different speeds and power budgets, as happens in digital circuits. S-AC also offers a common module to carry out all major computations required in ML, thus making it easy to scale and synthesize the logic. We show that the operation of S-AC units varies with acceptable margins from the weak inversion (WI) through moderate inversion (MI) to strong inversion (SI) and across temperatures (−45°C to 125°C).

We briefly describe the contributions of this work as under:

• Design of process technology scalable analog computing framework. The results are shown and compared for planar CMOS 180nm and finFET 7nm [12] process nodes.
• Demonstration of temperature scalability of proposed process scalable framework.
• Detailed mathematical framework for implementing basic proto-function and derived shape function based on shape-based analog computing framework.
• Demonstration of the shape-based system on planar CMOS and finFET MOS for ML tasks.

The rest of the sections are organized as follows. Section II presents the mathematical framework for shape-based analog computing along with its different MOS circuit implementations. Section III shows the design and implementation of various shape-based analog computing modules along with performance analysis. Section IV shows the implementation of a standard ML system using shape-based analog blocks followed by results and the paper is concluded in Section V.

II. THEORY OF SHAPE-BASED ANALOG COMPUTING

In this section, we present a detailed description of the shape-based analog computational framework along with its mathematical formulation. This section extends our previous work in the area of bias-scalable analog computing circuits [13] in generating more complex proto-functions that are matched to the physical operating principles of MOSFETs and diodes. It also extends our understanding of the process technology invariant feature of S-AC circuits using a mathematical approach. We start by approximating a non-linear monotonic function that forms the basic proto-shape. This proto-shape is later used to construct other non-linear functions.

A. Multi-Spline Approximation of log-sum-exp function

Let us consider a log-sum-exp function given by
\[ h(\mathbf{x}) = C \cdot \log \left( \sum_{i=1}^{N} e^{x_i} \right) \] (1)

where \( h(\cdot) \) is an estimate or proto-shape, \( C \) is a hyper-parameter and \( \mathbf{x} \) is a vector with elements \( x_i \).

Without going into a detailed mathematical exposition, we can show that \( h(\cdot) \) satisfies

\[ 1 \geq \frac{\partial h}{\partial x_i} \geq 0, \forall i \] (2)

and

\[ \lim_{x_i \to \infty} \frac{\partial h}{\partial x_i} = 1 \]
\[ \lim_{x_i \to -\infty} \frac{\partial h}{\partial x_i} = 0 \] (3)

The property in (2) ensures that the proto-shape \( h \) is monotonic with respect to its variable. Equation (1) can be written as

\[ \sum_{i=1}^{N} e^{x_i - h(x)} = 1 \] (4)

Let us approximate this non-linear function \textit{exponential} using linear splines. It can also be noted that the same methodology can also be extended to other non-linear functions. As a proof-of-concept, we showed here for log-sum-exponential.

Fig. 2a shows the plot of the \textit{exponential} function and its approximation using one-spline (\( S = 1 \)), two-splines (\( S = 2 \)) and three-splines (\( S = 3 \)). Here, \( Q_1, Q_2, \ldots, Q_S \) are the tuning points and \( T_1, T_2, \ldots, T_S \) are the tuning points.

The generic line equation for the \( j^{\text{th}} \) spline where \( j \in (1, \ldots, S) \) when approximated using piece-wise-linear lines can be written using point-slope form as

\[ \theta_j(x) = e^{Q_j} \cdot x + e^{Q_j} \cdot (1 - Q_j); \quad \forall x \geq 0 \] (5)

where \( e^{Q_j} \) is the slope and \( e^{Q_j} \cdot (1 - Q_j) \) is its intercept on the line on the vertical axis. The tuning points \( T_j \) (intercept between \( j^{\text{th}} \) and \( (j+1)^{\text{th}} \) spline) can be obtained by equating the line equations of \( j^{\text{th}} \) and \( (j+1)^{\text{th}} \) spline at \( x = T_j \) and can be written as,

\[ e^{Q_j} \cdot T_j + e^{Q_j} \cdot (1 - Q_j) = e^{Q_{j+1}} \cdot T_j + e^{Q_{j+1}} \cdot (1 - Q_{j+1}) \] (6)

Equation (6) can be re-written as

\[ T_j = \frac{Q_{j+1} \cdot e^{Q_{j+1}} - Q_j \cdot e^{Q_j}}{e^{Q_{j+1}} - e^{Q_j}} - 1 \] (7)

Then, the approximation of the function \( e^x \) using \( S \)-splines (where a special case of 3-spline approximation is shown in Fig. 2a) can be written using point-slope form and using (5) and (7) as

\[ e^x \cong \sum_{j=1}^{S} e^{Q_j} \cdot \left[ x - T_j \right]_+ + \left( e^{Q_2} - e^{Q_1} \right) \cdot \left[ x - T_2 \right]_+ + \cdots + \left( e^{Q_S} - \cdots - e^{Q_2} - e^{Q_1} \right) \cdot \left[ x - T_S \right]_+ \] (8)

Equation (8) can then be generalized as

\[ e^x \cong \sum_{j=1}^{S} \left( e^{Q_j} - \sum_{k=1}^{j-1} e^{Q_k} \right) \cdot \left[ x - T_j \right]_+ \] (9)

The above equation (9) shows the approximation of \( e^x \) using \( S \)-splines. For ease of understanding let us choose a specific case of 3-splines, viz. \( S = 3 \). Let the tangential points for this case be \( Q_1 = \log_e \left( \frac{1}{2} \right), \; Q_2 = \log_e(1), \; Q_3 = \log_e(2) \). Then by using (7) we get,

\[ T_1 = \log_e \left( \frac{1}{2} \right) - 1 = -\log_e 2 - 1 \] (10)
\[ T_2 = \frac{-\log_e \left( \frac{1}{2} \right)}{1/2} - 1 = \log_e 2 - 1 \] (11)
\[ T_3 = \frac{2 \log_e 2}{2 - 1} - 1 = 2 \log_e 2 - 1 \] (12)

Using equation (10) - (12) in (9), we have

\[ e^x \cong \frac{1}{2} [x + \log_e 2 + 1]_+ + \frac{1}{2} [x - \log_e 2 + 1]_+ + \cdots + \frac{1}{2} [x - 2 \cdot \log_e 2 + 1]_+ \] (13)

Substituting equation (13) in (4) we get

\[ \sum_{i=1}^{N} [x_i + O_1 - h]_+ = [x_i + O_2 - h]_+ = [x_i + O_3 - h]_+ = C' \] (14)

Here, \( O_1, O_2 \) and \( O_3 \) are the offsets and \( C' = 2C \) is a tunable parameter. Equation (14) shows the approximation of exponential function with 3-splines. The same can then be extended to a generic case of \( S \)-splines, \( N \)-inputs as

\[ \sum_{i=1}^{N} \sum_{j=1}^{S} [x_i + O_j - h]_+ = C \] (15)

where \( O_j \) is the offset due to \( j^{\text{th}} \) spline, \( x_{i,j} \) is the \( i^{\text{th}} \) input corresponding to \( j^{\text{th}} \) spline, \( C \) is a hyperparameter and \( S \) is a design parameter also called splines count. It can be noted from Fig. 2a that with the increase in the number of splines \( S \), the computational precision increases, while the input dimension increases along \( N \). Equation (15) can be referred as Generalized Margin Propagation function (GMP) which maps to shape function \( h(\cdot) \).

\[ \text{B. MOS implementation of Shape-function } h(\cdot) \]

In its most general form, the drain-to-source current (: \( I_{ds} \)) flowing through an n-type MOSFET can be expressed as the difference between the forward and reverse currents (14) as

\[ I_{ds} = I_s[f(V_g, V_s) - f(V_d, V_d)] \] (16)

where \( I_s \) is the specific current and \( f : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \) is a function that models the forward and reverse currents with respect to the gate (\( V_g \)), drain (\( V_d \)) and source (\( V_s \)) voltages.
be computed as a solution to the equation

\[ V_{\text{h}} \text{ shown at different process technology for input } x \text{ and spline-count } S = 1, \]

\[ V_{\text{h}} \text{ shown at different process technology for input } x \text{ and spline-count } S = 3, \]

\[ V_{\text{h}} \text{ shown at various operating regimes for 180nm process node}, \]

\[ V_{\text{h}} \text{ shown at various operating regimes for 7nm process node}. \]

Fig. 3. (a) The proto-function shape \( h(x) \) shown at various operating regimes for 7nm process node.

Fig. 4. Plot showing effect of temperature variation on proto-function shape \( h(x) \) for temperature varying from \(-45^\circ\text{C}\) to \(-125^\circ\text{C}\) in 180nm.

respectively. \( V_{\text{h}} \) can be implemented using CMOS circuits as shown in Fig. 25. Here, \( x_{i,j} \) is the input current for the \( i^{\text{th}} \) input and the \( j^{\text{th}} \) spline and \( h(X) \) is the output current. \( V_{i,j}, V_B \) are the voltages across the \( N_{i,j}^{th} \) transistor, \( C \) is a constant current and \( D_{i,j} \) denotes diode elements (Schottky, MOS diode or any other). Applying KCL at node \( V_B, \) (17) can be obtained while the current across diode \( D_{i,j} \) gives (18). Similar operation can be obtained in other quadrant using PMOS variant shown in Fig. 2c. For the specific case of approximating the non-linear function using single spline such that \( S = 1, \) Equation (17) and (18) changes to

\[
\sum_{i=1}^{N} f(V_i, V_B) = C, \quad \forall i = 1, ..., N \quad (19)
\]

\[
f(V_B, 0) - f(V_B, V_i) + f(V_i, V_B) = x_i \quad (20)
\]

The above four equations (17) - (20) governs the working principle of S-AC circuits.

C. Process and Temperature Invariant Proto-function

The rationale behind shape-based computing is to create proto-functions that remain invariant to biasing and operating conditions. Here, we choose \( f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \) as a function that models the forward and reverses currents of MOSFET. It can be noted that (16) holds irrespective of the choice of transistor models such as EKV (Enz, Krummenacher, and Vittoz) [15], ACM (Advanced Compact MOSFET) [16], etc. or operating regimes, i.e. weak-inversion, moderate-inversion or strong-inversion, or process nodes viz. MOSFET, finFET, etc. The function \( f(\cdot, \cdot) \) always satisfies the following properties:

- \( f(0, 0) = 0 \) and \( f(\cdot, \cdot) \) is always positive or \( f(\cdot, \cdot) \geq 0, \)

by construction.

- \( f(\cdot, \cdot) \) is monotonic. For \( V_{g1} > V_{g2}, \) \( f(V_{g1}, V_s) > f(V_{g2}, V_s) \) and for \( V_{s1} > V_{s2}, \) \( f(V_g, V_{s1}) < f(V_g, V_{s2}) \).

Given an input matrix \( X \in \mathbb{R}^{N \times S} \) where \( x_i \in \mathbb{R}^N, \forall i = 1, ..., N \) is the input vector and \( x_j \in \mathbb{R}^S, \forall j = 1, ..., S \) is the number of splines, the proto-function \( h : \mathbb{R}^{N \times S} \rightarrow \mathbb{R} \) can be computed as a solution to the equation \( h(X) = f(V_B, 0) \) where the variable \( V_B \) is the solution to:

\[
\sum_{i=1}^{N} \sum_{j=1}^{S} f(V_{i,j}, V_B) = C, \quad \forall i = 1, ..., N \quad (17)
\]

\[
f(V_B, 0) - f(V_B, V_{i,j}) + f(V_{i,j}, V_B) = x_{i,j} \quad (18)
\]

Here, \( C \) is a hyper-parameter and \( V_{i,j} \) is an internal variable. Equations (17) - (18) can be implemented using CMOS circuits as shown in Fig. 25. Here, \( x_{i,j} \) is the input current for the \( i^{th} \) input and the \( j^{th} \) spline and \( h(X) \) is the output current. \( V_{i,j}, V_B \) are the voltages across the \( N_{i,j}^{th} \) transistor, \( C \) is a constant current and \( D_{i,j} \) denotes diode elements (Schottky, MOS diode or any other). Applying KCL at node \( V_B, \) (17) can be obtained while the current across diode \( D_{i,j} \) gives (18). Similar operation can be obtained in other quadrant using PMOS variant shown in Fig. 2c. For the specific case of approximating the non-linear function using single spline such that \( S = 1, \) Equation (17) and (18) changes to

\[
\sum_{i=1}^{N} f(V_i, V_B) = C, \quad \forall i = 1, ..., N \quad (19)
\]

\[
f(V_B, 0) - f(V_B, V_i) + f(V_i, V_B) = x_i \quad (20)
\]

The above four equations (17) - (20) governs the working principle of S-AC circuits.
regimes which correspond to different functions \( f \) in (17)-(18). The plots show that the shape of the proto-function remains invariant to the biasing condition and is constrained within a well-defined "margin". This margin is determined by the design parameter \( S \) and the inherent feedback from the hyper-parameter \( C \). Fig. 3d shows similar response plots for 7nm process node. Fig. 4 shows the effect of temperature on the proto-function. It can be observed that even with the change in temperature, the proto-function shape remains unaffected.

**D. Deep-threshold Operation of S-AC Circuit**

To exploit the complete available current range (down to the diffusion diodes’ reverse leakage currents) of transistors in planar CMOS, two approaches can be followed. First, the \( V_{GS} \) must be made negative; second, the threshold voltage, \( V_{TH} \), must be increased in a way that the channel inversion occurs at higher \( V_{GS} \) voltages [17, 18, 19, 20]. Therefore, to conduct smaller currents, \( V_{GS} \) should be biased to the reverse direction (\( V_{GS} < 0V \) for NMOS and \( V_{GS} > 0V \) for PMOS), which is defined as the deep sub-threshold region, and MOSFET working in this region is thus nominated as DSMOS (Deep Sub-threshold MOS). For this approach, the source voltage shifting technique can be used. By shifting the source voltage slightly higher than the lowest potential (VSS), the gate voltage can reach down to the lowest potential (VSS). Fig. 5a shows the \( I_D \) vs. \( V_{GS} \) characteristic plot in log scale for source shifted MOS transistors. The lowest value of current found with the source shifting technique was 45 fA for NMOS and 44 fA for PMOS in CMOS 180nm process node. On analyzing Fig. 5a one can see that the source voltage shifting can be used to utilize the complete device physics of the transistor, i.e., exploiting the complete available current range (down to the diffusion diodes reverse leakage currents).

For the second approach (so-called channel conduction manipulation technique), the bodies of transistors are connected to the highest potential (VDD). This will prevent the channel inversion from taking place at low \( V_{GS} \) voltages which, along with source shifting, further lowers the bottom level of the operating current of the circuit. We used a technique that combines a fixed source shifted voltage with the channel conduction manipulation technique. Fig. 5b shows the circuit implementation of equations (17)-(18) using the above two approaches combined, where source voltage modulation along with channel conduction manipulation techniques were used to shift the operation in femto-ampere (fA). Fig. 5c shows the proto-function obtained in the deep-threshold operation of an NMOS for femto-ampere (fA) current. It shows that if biased properly, S-AC can also operate in the diode-leakage regime.

**III. SYNTHESIS OF ANALOG COMPUTING MODULES**

In this section, we focus on building various functions using the S-AC standard cell and analyze it in various aspects.

**A. Examples of different mathematical functions**

S-AC circuits shown in Fig. 2b and Fig. 2c can be used as analog standard cells to implement various mathematical computations. These P- and N-type S-AC cells can be used to synthesize basic mathematical functions, which can further be utilized to perform complex machine learning tasks. In this section, we show the implementation of a few widely used mathematical functions. However, it may be noted that the enlisted ones do not limit the framework. As a proof of concept, we show the S-AC-based implementation of various activation functions, energy-efficient multiplication, and computation of partial derivatives, alongside other mathematical functions such as Winner-Takes-all (WTA) for max, n-of-m selector, and n-to-m encoder. It can be noted that the proto-function \( h(x) \) can be mapped to exponential for \( e^x \) ranging from \([-\infty, 1)\). Within this defined range \( \frac{\partial h}{\partial x} < 1 \) and equation (2) and (3) are satisfied and hence even hyperbolic functions such as Cosine, Sine and others derived functions can be constructed. The following text has a detailed description and formulation of each of these circuits. We also demonstrate their performance in both 180nm and 7nm process technology nodes.

1) Cosine-Hyperbolic: The \( \cosh(\cdot) \) function can be constructed using N-type S-AC standard cells for \( S = 3 \) as shown in Fig. 6a. The \( \cosh(\cdot) \) function is given by:

\[
\cosh(x) = \frac{e^x + e^{-x}}{2}
\]  

(21)
Now, if in Fig. 2a, the response of one S-AC unit is $h(x) \approx \frac{e^x}{2}$, then by tuning the offsets $O_1, \cdots, O_3$, we can get $cosh()$. In terms of S-AC computation, (21) can be written as

$$cosh(x) = h(x_1) + h(-x_1)$$

Thus the $cosh()$ function is the addition (KCL) of the response of S-AC ($y_1$ in Fig. 6a) and its vertically flipped response (shown by $y_2$ in Fig. 6a). Fig. 7a shows characteristics obtained using Fig. 6a across different process nodes for the same values of offset. The $cosh()$ function is crucial for constructing $tanh()$ function which is used as activation function in a number of machine learning applications.

2) Sine-Hyperbolic: The $sinh()$ function can be constructed using both P- and N-type S-AC standard cells for $S = 3$ as shown in Fig. 6b. The $sinh()$ function is given by:

$$sinh(x) = \frac{e^x - e^{-x}}{2}$$

Similarly if we can get some $h(x) \approx \frac{e^x}{2}$. Thus, in terms of S-AC computation, (23) can be written as

$$sinh(x) = h(x_1) - h(-x_1)$$

Fig. 7b shows characteristics obtained using Fig. 6b across different process nodes. The $sinh()$ function is crucial for constructing $tanh()$ function which is used as activation function instead of sigmoid.

3) ReLU: ReLU function is a widely used activation function and can be implemented using a single $S = 1$ S-AC standard cell as shown in [11]. Fig. 6c implements $max\{0, x_1\}$. In Fig. 7c we demonstrate the shape of ReLU implemented using the circuit in Fig. 6c to be invariant across different process nodes.

4) Tangent-Hyperbolic: The $tanh()$ function is another widely used activation function. It is used sometimes in lieu of sigmoid function. The $tanh()$ can be expressed mathematically as:

$$tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$$

In other words, when taken in the logarithmic domain and expanded, it can also be expressed using S-AC as:

$$tanh(x) = h(0, x_1 + x_2) - h(x_1, x_2)$$

The circuit for $tanh()$ is shown in Fig. 6d. We show that the $tanh()$ function constructed using S-AC blocks is also invariant to process nodes in Fig. 7d.

5) Sigmoid: The sigmoid curve is a shifted version of the $tanh()$ function. Thus the same circuit can be utilized for sigmoid, as is shown in Fig. 6d. We show that the sigmoid function constructed using S-AC blocks is also invariant to process nodes in Fig. 7e.

6) Soft-Plus: The soft plus activation function can be obtained by applying two inputs to $S = 3$ S-AC standard cells. We show in Fig. 7f that the soft-plus function constructed using the circuit in Fig. 6e is invariant to process nodes.

7) S-AC based Four Quadrant Multiplier: The four-quadrant multiplier is proposed in [11]. Owing to the Lipchitz behavior of the S-AC function, the design is formulated such that the multiplication satisfies the Lipchitz condition. Fig. 8a
Fig. 7. Non-linear characteristic plots and its temperature variation obtained from corresponding S-AC standard cells shown in Fig. 6 for (a) \( \cosh(x) \), (b) \( \sinh(x) \), (c) ReLU, (d) \( \tanh(x) \), (e) Sigmoid, (f) Soft-Plus at FinFET (7nm) and CMOS (180nm) process nodes.

shows the S-AC cell-based implementation of a multiplier circuit. Consider the following equation, where \( y \) is given by

\[
y = h(C + w + C + x) - h(C + w + C - x) + \ldots
\]

\[
h(C - w + C - x) - h(C - w + C + x)
\] (27)

The goal is to implement scalar multiplication between two variables \( x \) and \( w \). Here \( x, w, y \in \mathbb{R} \), \( h \) is a non-linear monotonic function and \( C \) is a hyperparameter. If we write the Taylor expansion of \( h(x) \) around \( w \) and ignore the higher-order terms, we will get

\[
h(C + w + C + x) = h(C + w) + (C + x) \Delta h(C + w) + \ldots
\]

\[
\frac{C + x^2}{2} \Delta^2 h(C + w) \quad (28)
\]

\[
h(C + w + C - x) = h(C + w) + (C - x) \Delta h(C + w) + \ldots
\]

\[
\frac{C - x^2}{2} \Delta^2 h(C + w) \quad (29)
\]
Fig. 9. Multiplier characteristic curve obtained for design parameter $S = 3$ (a) at different operating regimes for 7nm process node, (b) at different operating regimes for 180nm process node. (c) Implementation of $N$ inputs winner takes all (WTA) standard cell using S-AC units. The same circuit can be tuned to function as a soft-WTA and Current-Max unit.

It can be analyzed from Fig. 9a and Fig. 9b that the shape of the multiplier characteristic curve remains preserved when the circuit operation moves from WI to SI.

8) S-AC based Addition, Subtraction, and Multiply-Accumulate: In analog circuits, addition and subtraction come almost for free owing to the basic current conservation principles. The multiplication can be carried out in parallel and added together using Kirchoff’s current law to perform the accumulation.

9) S-AC based Winner-Take-All: A winner-take-all (WTA) circuit is designed to emulate the $\max(\cdot)$ function. The proposed S-AC-based WTA circuit is shown in Fig. 9c. It is modular, i.e., it can be extended for $N$ inputs like the original circuit proposed by Lazzaro et al. (1989) [21]. Fig. 10a and Fig. 10b show the current characteristics of $180\,nm$ and $7\,nm$ for an input differential current. When the differential current is $0$, the currents are equal. The corresponding voltage outputs for both the process nodes are seen in Fig. 10c and Fig. 10d.

10) S-AC based N-of-M Encoder: N-of-M encoding is of particular importance in machine learning. The N-of-M encoder circuit extends the computational capabilities of the standard WTA circuit and has found profound importance in sparsely distributed memory [22]. N-of-M encoder allows the user to obtain the max current that takes into account the influence due to the contribution of top $M$ winners out of $N$ inputs ($M/N$). For the specific case of $M = 1$, the N-
Fig. 11. Response of five-input S-AC WTA showing $M$ selected winners as a function of hyperparameter $C$ at (a) 180nm and (b) 7nm process nodes. Here, $I_{out}$ shows the contribution $M$ selected winners in the output. Response of five-input S-AC WTA showing individual outputs [$I_{out1}, \ldots, I_{out5}$] as a function of hyperparameter $C$ at (c) 180nm and (d) 7nm process node. Response of Fig. 11 is obtained for the inputs [$x_1, x_2, x_3, x_4, x_5$] = [$\alpha, 2\alpha, 3\alpha, 4\alpha, 5\alpha$] where $\alpha = 1\mu A$ for 180nm and $10nA$ for 7nm.

of-M encoder behaves like a simple WTA circuit. Fig. 11a shows the response of the five-input S-AC-based WTA circuit shown in Fig. 9c as a function of hyperparameter $C$. It can be noted that with the increase in hyper-parameter value, the output current $I_{out}$ decreases and is the result of more than one winner. Using (19), $I_{out}$ is given by

$$I_{out} = \frac{\sum_{i=1}^{M} x_i - C}{M}$$  \hspace{1cm} (35)$$

where $M$ is the number of winners. We implemented a similar encoder circuit in 7nm, and the results are shown in Fig. 11b. It can easily be analyzed that depending on the hyper-parameter $C$, the circuit can select the top $M$ winners.

11) S-AC based SoftArgmax: In machine learning, SoftArgmax offers an improvement over Argmax to support back-propagation and gradient operation. The S-AC-based WTA circuit can be configured to implement SoftArgmax. Fig 11c shows the response curve of outputs $I_{out1}, \ldots, I_{out5}$ for the variation in hyper-parameter $C$. It can be observed that with the increase in hyper-parameter, outputs corresponding to the maximum input along with other inputs are activated and can be given by

$$I_{out_i} = x_i - C, \quad \forall x_i > C$$  \hspace{1cm} (36)$$

12) S-AC based Max Circuit: The S-AC winner-take-all circuit can be configured to select the maximum input among the given set of $N$ inputs. For hyper-parameter, $C \to 0$, the circuit starts behaving as a max input selector.

13) S-AC based Partial Derivative: Partial derivatives are used very often in machine learning algorithms for finding the minimum or maximum of a function. They are often used in back-propagation technique to find gradients. Without going into explicit details, we here show how can we calculate partial derivative using S-AC functions. Taking partial derivative of (15) with respect to $x_i$,

$$\frac{\partial h}{\partial x_i} = \frac{1}{|A|} \mathbb{1}(x_{i,j} > h)$$  \hspace{1cm} (37)$$

where $|A|$ indicates the number of $x_{i,j}$ such that $x_{i,j} > h$ and $\mathbb{1}$ is the indicator function. It can be observed that the partial derivative in the S-AC domain is mapped to a simple fractional count $A$.

B. Performance Analysis

This section shows the performance analysis of the S-AC computational blocks at different process technology nodes, viz. 180nm and 7nm and at different operating conditions. The inter-dependence of settling time, bandwidth, maximum operating frequency, throughput, power, and energy parameters trade-off for designing a high-performance, the scalable system has been explained in detail.
In this parallel current-mode configuration, because we are summing two uncorrelated noise sources, the overall noise increases as $\sqrt{2}$, while the correlated input signal amplitude increases by 2. Mathematically, for a single S-AC block having input signal $x_1$ and gain $G_1$, the total combined input signal is given by $x_1 + n_{in_1}$. The output includes the output signal $Z$ plus the total output noise $n_{out_1}$.

$$n_{out_1} = n_{in_1} \times G_1 + n_{ckt_1}$$

$$Z = x_{in_1} \times G_1$$

The SNR is calculated by dividing the output RMS signal power by the output RMS noise power. This comes out to be

$$SNR_1 = \frac{Z}{n_{out_1}} = \frac{(x_{in_1} \times G_1)^2}{(n_{in_1} \times G_1)^2 + n_{ckt_1}^2}$$

Assuming the external noise input power is minimal (for simplicity), then (39) reduces to

$$SNR_1 = \frac{(x_{in_1} \times G_1)^2}{n_{ckt_1}^2}$$

Now, adding a second S-AC block in parallel increases the RMS signal power by 2x, but only increases the RMS circuit noise by $\sqrt{2}$ because the circuit adds uncorrelated noise. So instead of the noise doubling, we obtain a noise of $\sqrt{2} \times n_{ckt}$. The SNR equation for two interconnected S-AC blocks (assuming the composite gain remains almost the same) becomes

$$SNR_2 = \frac{2(x_{in_2} \times G_2)^2}{(2n_{in} \times G)^2 + \sqrt{2} \left((n_{ckt_2})^2 + (n_{ckt_1})^2\right)}$$

For $n_{ckt_1} = n_{ckt_2} = n_{ckt}$ and assuming the external input noise power is minimal (81) changes to

$$SNR_2 = \frac{(x_{in} \times G)^2}{0.5(n_{ckt})^2}$$

On comparing (41) and (43) one can analyze that for each increase in the number of connected S-AC blocks in parallel, the circuit SNR increases by twice.

### 5) Operational Performance, Energy and Computational Analysis:

Table I gives shows the operational performance parameters of the S-AC analog cells at different operating regimes and at different process nodes. We here computed the peak capabilities of the S-AC architectures for some of the widely used performance metrics such as Computational efficiency, Power efficiency, and System efficiency. It can be observed that the Computational efficiency is highest in

| Parameter | Technology and Operating Regimes | CMOS 180nm | FinFET 7nm |
|-----------|----------------------------------|-------------|------------|
| SI | MI | WI | SI | MI | WI |
| Computational Efficiency (TOPS/mm²) | 0.043 | 0.036 | 2.39×10⁻⁴ | 165 | 148 | 13 |
| Power Efficiency (TOPS/W) | 0.223 | 2.69 | 31.6 | 22.68 | 165.49 | 1073.88 |
| System Efficiency (µJ/MAC) | 4.47 | 0.211 | 0.037 | 0.187 | 0.0036 | 7.32×10⁻³ |
TABLE II
ENERGY/OPTERATION, COMPUTATIONAL COMPLEXITY (CC) AND MEAN ABSOLUTE DEVIATION

| Operation       | MaxErr* | CC  | Process Node | Energy per Operation (fJ) |
|-----------------|---------|-----|--------------|--------------------------|
|                 |         |     |              | WI           | MI          | SI          |
| Cosh            | 0.05994 | O(1)| 180nm        | 3.2          | 5.184       | 8.437       |
|                 |         |     | 7 nm         | 0.0505       | 2.16        | 2.61        |
| Sinh            | 0.00982 | O(1)| 180nm        | 2.56         | 10.368      | 1.956       |
|                 |         |     | 7 nm         | 0.0735       | 4.22        | 8.924       |
| Tanh            | 0.00542 | O(1)| 180nm        | 1.12         | 4.536       | 598.5       |
|                 |         |     | 7 nm         | 0.0322       | 1.39        | 54.11       |
| ReLU            | 0.03374 | O(1)| 180nm        | 0.32         | 1.296       | 171         |
|                 |         |     | 7 nm         | 0.0097       | 0.34        | 5.59        |
| Sigmoid         | 0.00542 | O(1)| 180nm        | 1.12         | 4.536       | 598.5       |
|                 |         |     | 7 nm         | 0.0322       | 1.39        | 54.11       |
| Soft-plus       | 0.03212 | O(1)| 180nm        | 1.12         | 4.536       | 598.5       |
|                 |         |     | 7 nm         | 0.0322       | 1.39        | 54.11       |
| WTA (N-input)   | 0.176   | O(1)| 180nm        | N × 0.16     | N × 0.648   | N × 85.5    |
|                 |         |     | 7 nm         | N × 0.0496   | N × 0.24    | N × 85.5    |
| Multiply        | 0.0139  | N(1)| 180nm        | 0.16         | 0.648       | 85.5        |
|                 |         |     | 7 nm         | 0.0046       | 0.27        | 7.73        |
| Divide          | 0.0139  | N(1)| 180nm        | 0.16         | 0.648       | 85.5        |
|                 |         |     | 7 nm         | 0.0046       | 0.27        | 7.73        |
| Integrate       | -       | O(N)| 180nm        | N × 0.16     | N × 0.648   | N × 85.5    |
|                 |         |     | 7 nm         | N × 0.0496   | N × 0.24    | N × 85.5    |
| Differentiate   | -       | O(N)| 180nm        | N × 0.16     | N × 0.648   | N × 85.5    |
|                 |         |     | 7 nm         | N × 0.0496   | N × 0.24    | N × 85.5    |

*MaxErr = MAX_vg | Mean Absolute Deviation| between 180nm and 7nm process node at room temperature.

SI for planar CMOS while it is highest in MI for Finfet node. It also shows that the highest power efficiency and lowest pJ/MAC operation can be obtained in WI.

Table II summarizes the energy requirement, computational complexity, and Mean-Derivation for the S-AC block for carrying out basic computational operations at different operating regimes and at different process nodes. We reported the maximum mean absolute deviation obtained from the resultant functional shapes at 180nm and 7nm when similar architecture was used to obtain the same operation in both process nodes. Table II also shows the computation complexity for different operations. As expected, most of the computations on the circuit are single-shot computations based on the voltage/current settlements, the have an O(1) complexity. Finally, Table II also summarizes the Energy/Operation obtained at different operating regimes and at different process nodes for different S-AC computations. The least energy consumption is seen in the weak inversion regime and the worst in the strong inversion regime.

IV. EXAMPLE OF SHAPE-BASED ANALOG COMPUTING SYSTEMS

This section shows the mapping procedure for a generic ML application. Here, we show the design flow optimization of a shape-based neural network synthesized using S-AC standard cells but the same can be generalized for other machine learning algorithms. We present the software-hardware co-design procedure by implementing the standard decision function using a process-independent flow.

A. Algorithm Mapping

Consider a vector \( \mathbf{x} \in \mathbb{R}^N \) where the output \( y \) for a standard MLP [31] is given as,

\[
y = g(f(x))
\]

(44)

where \( g(\cdot) \) is any non-linear function be it \( \text{tanh}, \text{sigmoid}, \text{ReLU}, \text{etc} \) and \( f(\mathbf{x}) \) be the decision function given by

\[
f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b
\]

(45)

where \( \mathbf{w} \in \mathbb{R}^N \) is the trained weight vector, \( \mathbf{x} \in \mathbb{R}^N \) is the input vector, \( b \in \mathbb{R} \) is the bias and the function \( f: \mathbb{R}^N \rightarrow \mathbb{R} \) is the decision function. For \( \{x, y\} \in \mathbb{R} \), the decision function \( f(x) \) can then be rewritten as

\[
f(x) = w \cdot x + b
\]

(46)

Using equation (28) and (34) in (46) the decision function gets mapped to shape-based form as

\[
f(x) = h(2C + w + x) - h(2C + w - x) + \cdots + \cdots h(2C - w - x) - h(2C - w + x) + b
\]

(47)

Equation (47) can be viewed as generic decision function \( f \) mapped into shape domain. This shape equation can then be easily synthesized using only S-AC analog cells. It can be noted that variable \( b \) can be assumed as a constant current added to the dot-product \( w \cdot x \) and implemented using KCL without additional circuits. Furthermore, to add non-linearity to this output, function \( g \) can also be mapped to the shape domain using the formulation demonstrated in Section III.

B. Shape-Based Classification System

Fig. 14a shows the system-level implementation of a shape-based neural network using S-AC analog cells. The network was mapped using the algorithm mapping approach mentioned above and designed using the cells mentioned in the previous section. Fig. 14b shows the layout of the synthesized S-AC neural network used for simulation. Fig. 14c shows the linear and non-linear classification results obtained from a part of the synthesized network shown in Fig. 14a and Fig. 14b for XOR data-set. It can be observed that even for the minimal bias current of 80pA, the obtained accuracy is around 95%.

Table III summarizes the classification accuracy of synthesized neural networks using shaped-based analog standard cells at different process technology nodes. Results are also presented for circuits operating at different operating regimes in that process node. We verified our system on the standard Activity Recognition dataset (AREm) [32] dataset. We chose two of the activities as positive cases, i.e., bending and lying activities, to verify the classification capability of our system. One versus all approach was used on the AReM dataset for binary classification. We also verified the functionality of the standard XOR dataset [33]. It can be analyzed that the classification accuracy of implemented hardware is similar to that of software at both the process nodes, be it 180nm or 7nm implementation. The functionality also remains unaffected in different operating regimes. This clearly signifies that the design is both process technology scalable and bias scalable.

V. CONCLUSION

In this work, we proposed a modular analog computing framework that is both bias- and process-technology scalable.
We presented the detailed mathematical framework for designing a shape-based analog computing system using S-AC standard cells similar to the standard cells in digital design. Shape-based analog computing framework provides flexibility to the end-user to select computational precision at the cost of increased hardware accuracy. Process scalability and design space trade-offs analyses for designing a high-performance ML system using shape-based analog standard cell was carried out in detail at different process nodes and at different operating conditions. As a prototype application, the classification results at different process nodes and different conditions have been presented in Table III. In general, shape-based analog standard cells utilize a fundamental bias- and process- robust shape function and focus on the desired shape rather than computational precision to obtain scalability. The designed system is simple yet scalable and robust to non-ideal effects, which also simultaneously offers low-energy and high-performance footprints. S-AC framework can also be utilized to carry out complex machine learning tasks. Future works will include the demonstration of large-scale deep neural networks using the shape-based framework.

### A. Acknowledgments

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).

### REFERENCES

[1] S. Reda and M. Shafique, Eds., *Approximate Circuits, Methodologies and CAD*. Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-319-99322-5

[2] S. Mittal, “A Survey of Techniques for Approximate Computing,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–33, 2016.

[3] K. Berggren and et. al., “Roadmap on emerging hardware and technology for machine learning,” 2021 [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6528/aba70f/pdf

[4] H. Johan, S. Michiel, and H. Arthur, “Analog Circuit Design: Scalable Analog Circuit Design, High-Speed D/A Converters, RF Power Amplifiers,” 2002.

[5] C. S. Thakur, R. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, “A low power trainable neuromorphic integrated circuit that is tolerant to device mismatch,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 2, pp. 211–221, 2016.

[6] C. S. Thakur, R. Wang, T. J. Hamilton, R. Etienne-Cummings, J. Tapson, and A. van Schaik, “An analogue neuromorphic co-processor that utilizes device mismatch for learning applications,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1174–1184, 2017.

[7] A. G. Andreou and K. A. Boahen, “Translinear circuits in subthreshold MOS,” *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 141–166, 1996.

[8] E. Seevinck and R. J. Wiegerink, “Generalized translinear circuit principle,” *IEEE journal of solid-state circuits*, vol. 26, no. 8, pp. 1098–1102, 1991.

[9] E. Vittoz and J. Fellrath, “CMOS Analog Integrated Circuits Based on Weak Inversion Operations,” *IEEE journal of solid-state circuits*, vol. 12, no. 3, pp. 224–231, 1977.

[10] M. Chari and S. Salon, “5 - SHAPE FUNCTIONS,” in *Numerical Methods in Electromagnetism*, ser. Electromagnetism, M. Chari and S. Salon, Eds. San Diego: Academic Press, 2000, pp. 189–281.
[11] P. Kumar, A. Nandi, S. Chakrabarty, and C. S. Thakur, “CMOS Circuits for Shape-Based Analog Machine Learning,” 2022. [Online]. Available: https://arxiv.org/abs/2202.05022

[12] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, “ASAP?: A 7-nm finFET predictive process design kit,” Microelectronics Journal, vol. 53, pp. 105–112, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002626921630026X

[13] M. Gu and S. Chakrabarty, “Synthesis of Bias-Scaleable CMOS Analog Computational Circuits Using Margin Propagation,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, no. 2, pp. 243–254, 2012.

[14] Y. Tsividis, The MOS Transistor. New York: Oxford University Press, 2013.

[15] C. C. Enz, F. Krummenacher, and E. A. Vittoz, “An Analytical MOS Transistor Model Valid in All Regions of Operation and Dedicated to Low-Voltage and Low-Current Applications,” Analog Integr. Circuits Signal Process., vol. 8, no. 1, p. 83–114, jul 1995. [Online]. Available: https://doi.org/10.1007/BF01239381

[16] C. Galup-Montoro, M. C. Schneider, A. I. A. Cunha, F. R. de Sousa, H. Klimach, and O. F. Siebel, “The Advanced Compact MOSFET (ACM) Model for Circuit Analysis and Design,” in 2007 IEEE Custom Integrated Circuits Conference, 2007, pp. 519–526.

[17] K. Monfaredi, H. F. Baghtash, and S. J. Azhari, “A Novel Ultra-Low-Power Low-Voltage Femto-Ampere Current Mirror,” Circuits, Systems, and Signal Processing, vol. 31, pp. 833–847, 2012.

[18] H. F. Baghtash, K. Monfaredi, and S. J. Azhari, “A Novel High Performance Atto-Ampere Current Mirror,” in 2010 International Conference on Signal Acquisition and Processing, 2010, pp. 27–30.

[19] S. J. Azhari and G. Nickhah, “A Novel Ultra-Low-Power, Low-Voltage, Ultra-High Output Resistance and Uniquely High Bandwidth Femto-Ampere Current Mirror,” Circuits, Systems, and Signal Processing, vol. 36, pp. 3527–3548, 2017.

[20] B. Linares-Barranco and T. Serrano-Gotarredona, “On the design and characterization of femtoampere current-mode circuits,” IEEE Journal of Solid-State Circuits, vol. 38, no. 8, pp. 1353–1363, 2003.

[21] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, “Winner-take-all networks of O (n) complexity,” Advances in neural information processing systems, vol. 1, 1988.

[22] S. B. Furber, W. John Bainbridge, J. Mike Cumpstey, and S. Temple, “Sparse distributed memory using n-of-m codes,” Neural Networks, vol. 17, no. 10, pp. 1437–1451, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608004001443

[23] W. Maass, “On the computational power of winner-take-all,” Neural Comput., vol. 12, no. 11, pp. 2519–2535, nov 2000. [Online]. Available: https://doi.org/10.1162/089976600300014827

[24] P. Kinget and M. Steyaert, Implications of Transistor Mismatch on Analog Circuit Design and System Performance”, bookTitle="Analog VLSI Integration of Massive Parallel Signal Processing Systems. Boston, MA: Springer US, 1997, pp. 21–81. [Online]. Available: https://doi.org/10.1007/978-1-4757-2580-3_2

[25] M. Pelgrom, A. van Nimmergen, and A. Welbers, “Matching properties of MOS transistors,” IEEE Journal of Solid-State Circuits, vol. 24, no. 5, pp. 1433–1439, 1989.

[26] T. Mizuno, J. Okumura, and A. Toriumi, “Experimental study of threshold voltage fluctuation due to statistical variation of channel dopant number in MOSFET’s,” IEEE Transactions on Electron Devices, vol. 41, no. 11, pp. 2216–2221, 1994.

[27] U. Grunebaum, J. Oehm, and K. Schumacher, “Mismatch modelling for large area mos devices,” in Proceedings of the 23rd European Solid-State Circuits Conference, 1997, pp. 268–271.

[28] K. R. Lakshminarayan, R. A. Hadaway, and M. A. Copeland, “Characterisation and modeling of mismatch in MOS transistors for precision analog design,” IEEE Journal of solid-state circuits, vol. 21, no. 6, pp. 1057–1066, 1986.

[29] S. Narayanan, A. Shafee, and R. Balasubramonian, “INXS: Bridging the throughput and energy gap for spiking neural networks,” in 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017, pp. 2451–2459.

[30] A. Shafee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, “Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” ACM SIGARCH Computer Architecture News, vol. 44, no. 3, pp. 14–26, 2016.

[31] N. Cristianini, J. Shawe-Taylor et al., An introduction to support vector machines and other kernel-based learning methods. Cambridge university press, 2000.