CMOS Circuits for Shape-Based Analog Machine Learning

Pratik Kumar†, Ankita Nandi†, Shantanu Chakrabartty*, Chetan Singh Thakur†

{pratikkumar, ankitanandi, csthakur}@iisc.ac.in, {shantanu}@wustl.edu

†Department of Electronics Systems Engineering, Indian Institute of Science, Bangalore, India, 560012

*Department of Electrical and Systems Engineering, Washington University in St. Louis, USA, 63130

Abstract—While analog computing is attractive for implementing machine learning (ML) processors, the paradigm requires chip-in-the-loop training for every processor to alleviate artifacts due to device mismatch and device non-linearity. Speeding up chip-in-the-loop training requires re-biasing the circuits in a manner that the analog functions remain invariant across training and inference. In this paper, we present an analog computational paradigm and circuits using "shape" functions that remain invariant to transistor biasing (weak, moderate, and strong inversion) and ambient temperature variation. We show that a core *Shape-based Analog Compute (S-AC)* circuit could be re-biased and reused to implement: (a) non-linear functions; (b) inner-product building blocks; and (c) a mixed-signal logarithmic memory, all of which are integral towards designing an ML inference processor. Measured results using a prototype fabricated in a 180nm standard CMOS process demonstrate bias invariance and hence the resulting analog designs can be scaled for power and speed like digital logic circuits. We also demonstrate a regression task using these CMOS building blocks.

Index Terms—Analog Approximate Computing, Machine Learning, Logarithmic DAC, Analog Multiplier, ReLU, Bfloat16.
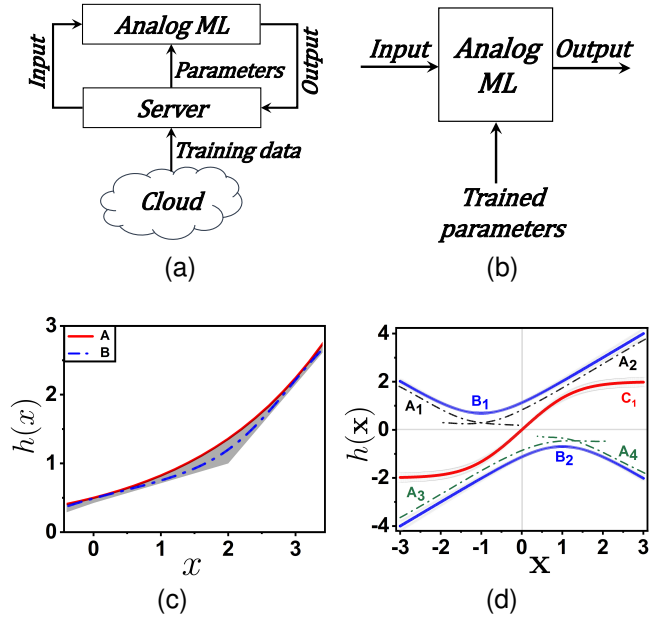
Fig. 1. (a) Chip-in-the-loop training of an analog ML processor; (b) Inference using analog ML processor programmed with trained parameters, (c) An example proto-function $h(x)$ whose shape remains invariant within a certain margin (shaded region) under different operating and biasing conditions, (d) Different non-linear functions $(A_1 - A_4), (B_1 - B_2)$ that can be implemented by translation, rotation, addition and subtraction of the proto-function.

I. INTRODUCTION

ANALOG computing techniques are attractive for implementing machine learning (ML) processors [1] because the paradigm can exploit computational primitives inherent in device physics and conservation principles to achieve very high computational density and energy efficiency. For instance, the compute-in-memory architectures proposed for ML processors could use translinear principles [2] or Ohm's law in conjunction with Kirchoff's current conservation law to implement energy-efficient matrix-vector-multipliers [3] and pattern classifiers [4]. Similarly, analog techniques could be used to synthesize different non-linear functions using very few transistors compared to their digital counterparts [5]. However, compared to digital implementations, analog computing by its nature, is approximate, relying on the accuracy of the physical models that govern the operation of the devices used in the computation. This requires proper biasing of the devices (to ensure sufficient dynamic range) and ensuring compliance with environmental factors like temperature. Therefore, analog ML processors have to use active temperature compensation

techniques [6], [7] and have to employ a chip-in-the-loop training procedure to compensate and calibrate for these devices and environmental artifacts [8].

This is illustrated in Fig. 1a, where during training the analog processor implements an ML model (for example a neural network) which is controlled by an external server. The server is assumed to have sufficient resources (memory, bandwidth and access to the data cloud) to store the training data and to perform a search over the ML parameters. In a typical chip-in-the-loop training procedure [9], the server programs the model parameters on the analog processor and then evaluates the output of the processor to determine the next set of parameters to be programmed (based on some optimization criterion). This procedure is iterated till some convergence criterion is reached, after which the programmed parameters are fixed over the duration of inference and deployment (as shown in Fig. 1b). While the chip-in-the-loop training can
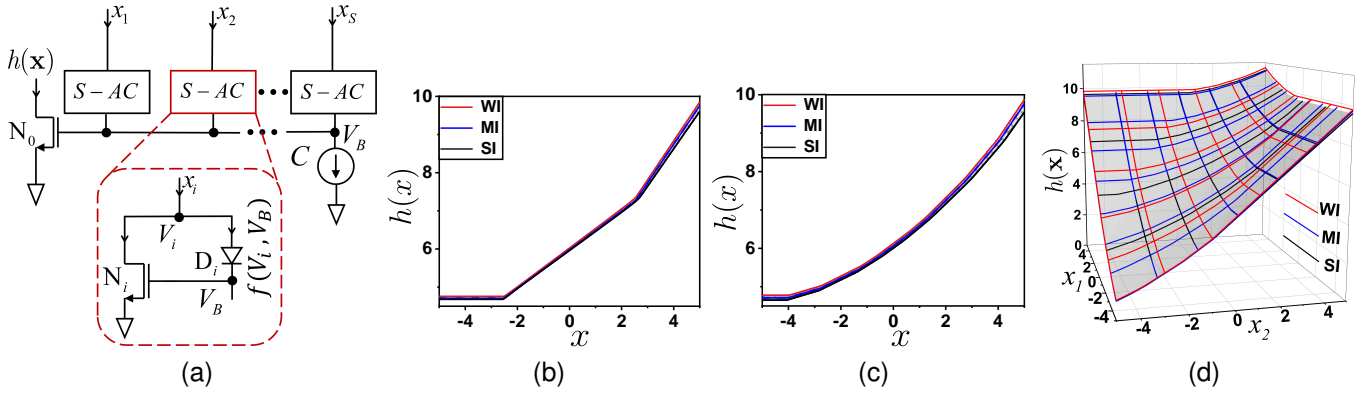
Fig. 2. (a) MOS S-AC unit implementation for Eq. (2)-(3) and shape function (as in Fig. 1c) shown at various operating regimes for (b) Input x and hyper-parameter $S = 1$, (c) Input x and hyper-parameter $S = 4$, (d) Inputs x_1, x_2 and hyper-parameter $S = 4$.

potentially compensate for any analog artifacts (mismatch and non-linearity), the procedure is time-consuming and has to be repeated for every analog processor. Proper initialization of the parameters and using a reduced training set (adaptation set) could potentially address this bottleneck [10]. However, it is still desired that the chip-in-the-loop training procedure be sped up significantly. Enhancing the speed requires the circuits to operate with higher currents which requires re-biasing the devices used in the computation. For example, analog computing circuits that operate using the MOSFET translinear principle require the transistors to be biased in one single regime i.e. weak-inversion [11] or strong-inversion [12]. Any deviation from the operating regime changes the function itself which introduces a mismatch between the training and inference operating conditions. Furthermore, operating the devices at a higher current leads to higher power dissipation and an increase in on-chip temperature and change in device characteristics.

In this paper, we present a *Shape-based Analog Computing (S-AC)* paradigm where the implemented functions remain robust to changes in biasing conditions and the operating temperature. Therefore, similar to digital circuits, S-AC circuits can be operated at different speeds and at different levels of power dissipation without changing the nature of the output function. The approach for synthesizing S-AC circuits is illustrated in Fig. 1c where we will first design a basic proto-function whose "shape" will remain invariant (within a prescribed error margin) to MOSFET biasing or the operating temperature. In this regard, we extend our previous work in the area of bias-scalable analog computing circuits [4] in generating more complex proto-functions that are matched to the physical operating principles of MOSFETs and diodes. As shown in Fig. 1d, the basic proto-function can then be translated, inverted, added and subtracted to obtain other non-linear and linear approximations.

In Section II we describe the shape-based analog synthesis approach and its relation to the most generalized form of the MOSFET model that is valid in all regions of operation. Then, in Section III, we present basic S-AC CMOS circuits that will be the building blocks for any analog ML inference processor. In Section IV we present measurement results using prototypes fabricated in a 180nm standard CMOS process and

in Section V, we demonstrate the functionality of a simple regression task combining the basic S-AC circuits. Finally, in Section VI we conclude the paper with brief discussions and a comparison of the results with other related work.

II. SHAPE-BASED ANALOG COMPUTATION

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 [8] as

$$I_{ds} = I_s[f(V_g, V_s) - f(V_g, V_d)] \qquad (1)$$

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 respectively. A similar expression as (1) also holds for a p-type MOSFET, except that the signs of the respective variables are reversed. Without any lack of generality, our analysis in this section will be based on the n-type MOSFET model; however, the formulation is applicable to p-type MOSFETs as well. It should also be noted that, as long as the source and the drain terminals are symmetric to each other, the expression in (1) holds irrespective of the choice of transistor models such as EKV (Enz, Krummenacher, and Vittoz) [13], ACM (Advanced Compact MOSFET) [14], 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 $V_{s1} > V_{s2}$, $f(V_g, V_{s1}) < f(V_g, V_{s2})$.

The rationale behind shape-based computing is to create proto-functions that are only dependent on the generic properties of $f(\cdot, \cdot)$, listed above, and which remain invariant to biasing and operating conditions. Here we specify one method to create such a proto-function:

Given an input vector $\mathbf{x} \in \mathbb{R}^S$ with elements $x_i \in \mathbb{R}, i = 1, .., S$, the proto-function $h : \mathbb{R}^S \to \mathbb{R}$ is computed as a

solution to the equation $h(\mathbf{x}) = f(V_B, 0)$ where the variable V_B is the solution to the following equations :

$$\sum_{i=1}^{S} f(V_i, V_B) \;=\; C \tag{2}$$

$$f(V_B, 0) - f(V_B, V_i) + f(V_i, V_B) = x_i, \forall i = 1, .., S \tag{3}$$

Here, C is a hyper-parameter, and V_i are internal variables. Without going into a detailed mathematical exposition, we can show that $h(\cdot)$ satisfies

$$1 \geq \frac{\partial h}{\partial x_i} \geq 0, \forall i \tag{4}$$

and

$$\lim_{x_i \to \infty} \frac{\partial h}{\partial x_i} \;=\; 1 \tag{5}$$

$$\lim_{x_i \to -\infty} \frac{\partial h}{\partial x_i} \;=\; 0 \tag{6}$$

The property in (4) ensures that the proto-function h is monotonic with respect to its variable (similar to the shape shown in Fig. 1c). Note that the properties described by equations (5) and (6) determines the two asymptotes of the proto-function, irrespective of the specific form of f. The hyper-parameter S and the vector $\mathbf{x}$ control the transition between the two asymptotes and hence can be used to adjust the non-linearity to the desired precision.

The equations (2)-(3) can be easily implemented using CMOS circuits as shown in Fig. 2a. Here, V_i and V_B are the voltages across the ith transistor, C is a constant current and D_i, $i \in (1, \cdots, S)$, denotes diode elements (Schottky, MOS diode or any other). Fig. 2b shows the example of the proto-function obtained using the circuit in Fig. 2a for $S = 1$. Similar results are plotted in Fig. 2c for $S = 4$. The results are also shown for different MOSFET biasing regimes, i.e., the Weak Inversion (WI), Moderate Inversion (MI), and Strong Inversion (SI) biasing regimes which correspond to different functions f in (2)-(3). The plots show that the shape of proto-function remains invariant to the biasing condition and is constrained within a well-defined "margin" that is determined by S. Note that the smoothness of the shape and the computational accuracy can be increased by increasing S, as observed in Fig. 2b and Fig. 2c. The effect of multiple inputs and the hyper-parameter S on the shape of the proto-function can be visualized in Fig. 2d in different operating regimes.

III. BASIC S-AC CIRCUITS FOR ML INFERENCE

The basic building blocks for designing an ML inference processor are: (a) non-linear computing circuits; (b) multiply-accumulate circuits; and (c) memory for storing the inference parameters and for supporting a digital interface for inputs and chip-in-the-loop training. Here we show that the basic S-AC circuit shown in Fig. 2a can be modified/extended to implement all the building blocks. Specifically, we implement a combination of a compressive mixed-signal memory and
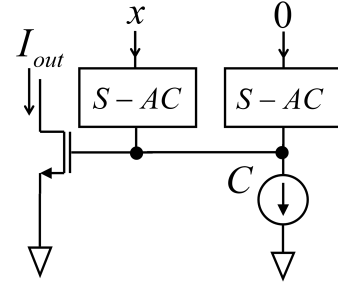

Fig. 3. Soft ReLU implementation using S-AC.

a non-linear multiplier circuit that results in a multiply-accumulate (MAC) operation which emulates computing using Bfloat16 number representation [15]. Note that any approximation error introduced in this mapping can be compensated during training itself, which is one of the main motivations for this work.

A. ReLU Implementation with S-AC

A soft ReLU function can be implemented using a one-dimensional proto-function shown in Fig. 2b and Fig. 2c. A circuit implementation of soft ReLU function is shown in Fig. 3. The basic circuit uses two S-AC units, one of which receives an input x and the other is driven by a zero current (or floating). The resulting function is similar to Fig. 2b where the constant current C determines the shape of the ReLU. Note that when the limit $C \to 0$, the proto-function converges to an ideal ReLU function. As described before in Section II, the shape of the proto-function and soft ReLU can be modified by adding more S-AC units in Fig. 3 which will result in an output similar to Fig. 2c. Also, note that other non-linear functions can be implemented by shift, translation, addition and subtraction of the basic proto-function like the $\tanh(\cdot)$ function illustrated by C_1 in Fig. 1d.

B. S-AC based Analog Multiplier

The S-AC proto-function h can be used to implement analog multipliers based on the following Taylor series approximations

$$
\begin{aligned}
&h\left(C + w + C + x\right) - h\left(C + w + C - x\right) \ldots \\
&+ h\left(C - w + C - x\right) - h\left(C - w + C + x\right) \\
&\approx 2x \times \left(\frac{dh(C+w)}{dw} - \frac{dh(C-w)}{dw} \right) \\
&\approx 2x \times (w^+ - w^-)
\end{aligned}
\tag{7}
$$

The constant C ensures that the input to the proto-function is always positive. The differential combination effectively cancels the zero-th order and second-order terms in the Taylor series [16] and the property of h in (4), leads to (7). Note that one of the differential arguments to the multiplier $(w^+ - w^-)$ is a non-linear map $\frac{dh}{dw}$, which based on property (4), is a compressive map. Thus, the stored parameters need to be pre-processed before and is presented as an input to the multiplier. This is the basis for our compressive memory design described in Section III-C.
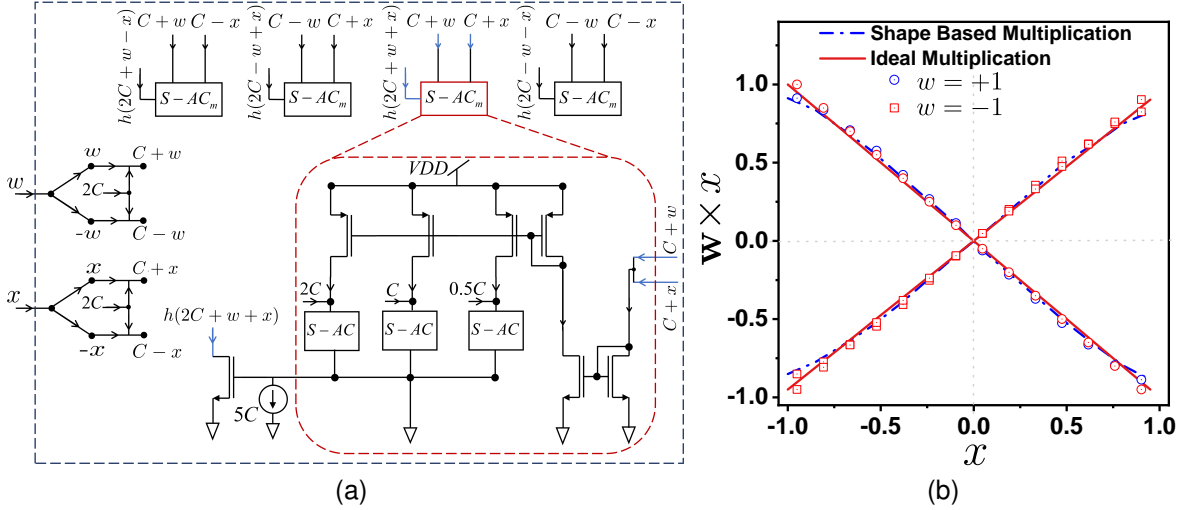
Fig. 4.   (a) MOS implementation of S-AC multiplier for hyper-parameter $S = 3$, (b) Comparison of four-quadrant S-AC multiplication with ideal multiplication.
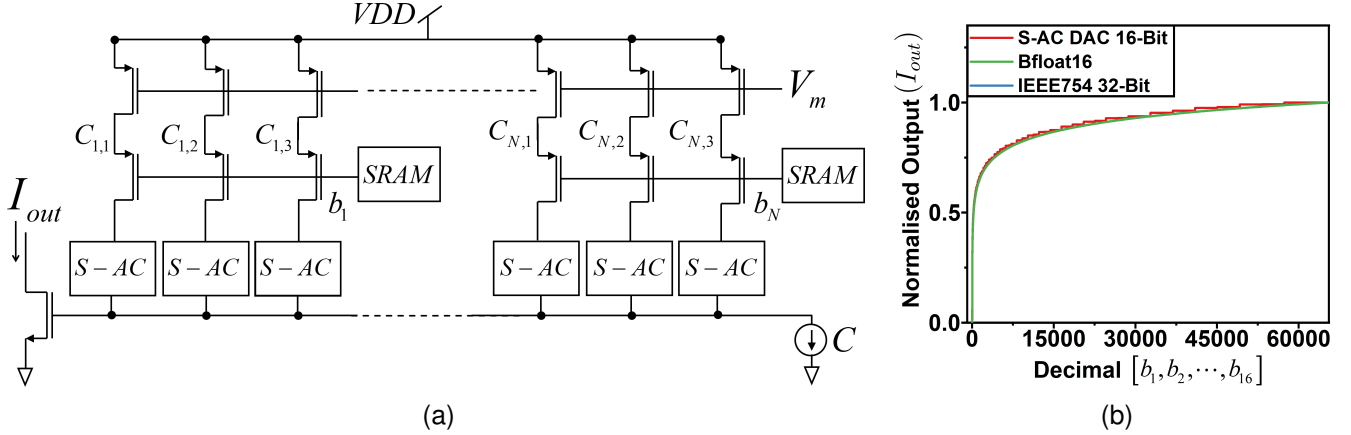

Fig. 5.   (a) Compressive log-binary DAC implementation using S-AC, (b) Comparison plot between S-AC log-binary DAC, Bfloat16 & IEEE32 number systems demonstrating close compliance of the corresponding log_2 curves with each other.

The circuit in Fig. 4a implements the scalar multiplication given in (7) where $w \in \mathbb{R}$, $x \in \mathbb{R}$ and the product $y \in \mathbb{R}$. Fig. 4a shows S-AC$_m$ (subscript m for multiplier) unit utilized to implement each component in (7). The inputs are first converted into their differential forms and constant (C) is added to the negative term to shift the operation in the first quadrant. The output from all S-AC$_m$ units are added and subtracted (differentially) as per (7) to obtain the desired multiplication. Fig. 4b shows a close approximation between the simulated output of the four-quadrant multiplier and the output obtained from an ideal multiplier. Based on this basic operation, multiply-accumulate operations and inner-products can now be implemented by combining element-wise S-AC multipliers with summing circuits based on Kirchhoff's current law. Other parallel analog matrix-vector-multiplier architectures have been reported in literature [17], [18].

C. Compressive Memory with S-AC

One of the major challenges in implementing an analog ML processor is the storage and updating of trained parameters. While analog memories based on memristors, floating-gates, and other nano-scale devices have been proposed for analog ML processors [19]–[21], their functional response and speed do not scale across training and inference. Therefore, in this paper, we propose to use a DAC based memory that uses an S-AC based analog frontend to implement a compressive function, as required by the multiplier in (7). Here we show that this compressive-expansive operation is equivalent to analog computing using Bfloat16 and the IEEE-754 single-precision (32-bit) number systems. Note that the Bfloat16 number system developed by Google Brain delivers more accurate results at lesser hardware as compared to IEEE 754 single-precision numbers for some neural network and is extensively used by Google cloud TPUs [15]. Consider a function $g(\mathbf{x})$ given by

$$g(\mathbf{x}) = \log_2 \left(\sum_i 2^{x_i} \right). \tag{8}$$

Then, it is easy to verify that $g(\mathbf{x})$ satisfies the properties

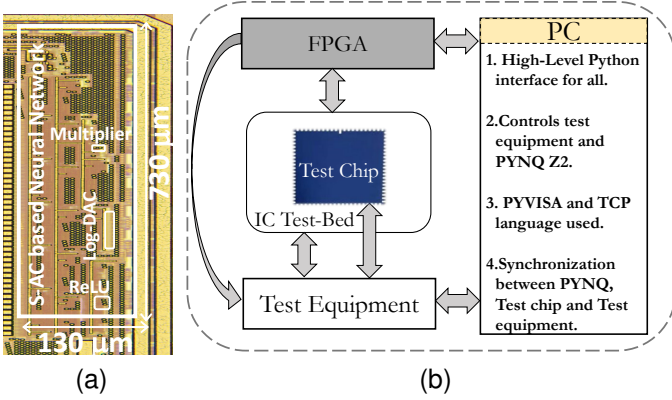$$1 \geq \left| \frac{\partial g}{\partial x_i} \right| \geq 0 \tag{9}$$

Fig. 6. (a) Die micro-photograph of the chip, (b) Test measurement setup.

Fig. 7. Measurement result of S-AC ReLU implementation shown in Fig. 3 for $C = 0.5$ demonstrating shape invariance across operating regions.

$$\lim_{x_i \to \infty} \frac{\partial g}{\partial x_i} = 1 \qquad (10)$$

similar to that of the proto-function $h(\mathbf{x})$ in (5) and (6). If $\mathbf{x}$ is denoted by its binary representation as $\mathbf{x} \cong \sum_{i=1}^{N} 2^i b_i$, then incrementing the hyper-parameter S per bit, we have

$$g(\mathbf{x}) = \log_2 \sum_{i=1}^{N} \sum_{j=1}^{S} 2^{C_{ij}} b_i = \log_2 \sum_{i,j:b_i=1}^{NS} 2^{C_{ij}} = g(\mathbf{B}) \qquad (11)$$

where $\mathbf{B} \in \{0,1\}^S \times \{0,1\}^N$ is a binary input matrix and N is the number of inputs.

It can be seen that (11) (logarithmic DAC) is a special case of (8) and hence can be approximated using the proto-function $h(\mathbf{x})$. Fig. 5a shows the circuit implementation of N-bit S-AC based compressive memory for $S = 3$. Switches connected at $b_1, b_2, ..., b_N$ are implemented using transmission gate (TG) switches. Here, $[b_1, ..., b_{N-1}, b_N]$ represent an N-bit binary number to be converted into its analog equivalent and $[C_{1,1}, C_{1,2}, C_{1,3}, ..., C_{N,3}]$ are the offsets when $S = 3$. The proposed S-AC based DAC converts the digital input into a compressive analog output. Note that this compressive output is implicitly expanded in (7) for multiplication. Fig. 5b compares the log_2 characteristics of the *Brain float* (Bfloat16) and the IEEE-754 single-precision (32-bit) number systems for 16-bit numbers normalized between 0 to 1 and the response obtained using the S-AC DAC for hyper-parameter $S = 4$. The results show compliance between the different logarithmic number representations.

IV. MEASUREMENT RESULTS

The S-AC building blocks have been prototyped in a standard *CMOS* 180nm process technology and Fig. 6a shows the die microphotograph of the chip. The functionality of the circuit modules has been verified using the test measurement setup shown in Fig. 6b. The test chip was mounted on a custom IC test board and the test vectors were generated using PYNQ-Z2 FPGA board which used a python-based interface to control the digital inputs and outputs. High precision analog test equipment were directly interfaced with the test chip and were controlled by PYNQ-Z2 FPGA board.
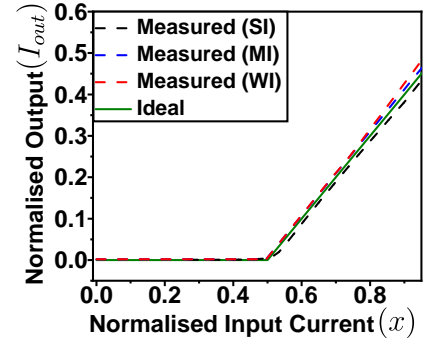
A. S-AC ReLU Measured Results

Fig. 7 shows the measured results of S-AC based ReLU implementation (Fig. 3) and its comparison with the ideal. It can be observed that the obtained normalized output current curve follows the desired non-linear shape and matches with the ideal. Furthermore, the non-linear shape remains invariant in weak, moderate and strong inversion regimes as desired.

B. S-AC Multiplier Measured Results

Fig. 8 shows the measured result of the implemented S-AC multiplier circuit for different values of hyper-parameter S and at different operating conditions. Fig. 8a shows the comparison plot of a four-quadrant multiplier for hyper-parameter $S = 1$ and $S = 3$. It can be noted that with the increase in hyper-parameter S, the multiplier accuracy increases and becomes much closer to the ideal. Fig. 8b is computed for $S = 3$, and shows the four-quadrant multiplication at different operating regimes in close compliance with each other. The results of Fig. 8a and Fig. 8b have been computed for $x \in (-1, 1)$ and for $\mathbf{w} = [-0.5, 0.5]$. Fig. 8c shows the multiplication curve for $x \in (-2, 2)$ and for different values of w.

C. S-AC Compressive Memory Measured Result

Fig. 9 shows the measured result of 8-bit S-AC based DAC as a function of equivalent decimal input varying from 0 to 255 at different operating regimes. It can be seen that the result closely approximates the desired ideal shape and the output shape is invariant across operating regimes. With increase in constant current C, along with the offsets $[C_{1,1}, C_{1,2}, C_{1,3}, ..., C_{N,3}]$ for $S = 3$, the S-AC DAC operation moves from WI to SI resulting in increasing power consumption but simultaneously reducing settling time and in turn improving throughput and speed. However, the optimum trade-off between energy and throughput can be obtained in the MI region of operation.

D. Energy and Error Analysis

Table I shows the energy per operation of basic operations mapped in the shape-based analog computation. It can be seen that as the circuit operating regimes move from WI to SI,
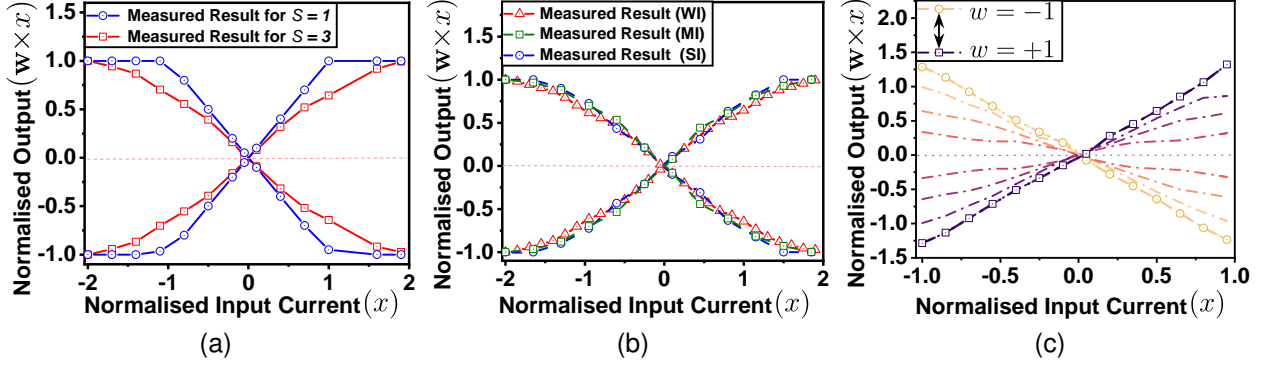
Fig. 8. Measurement result (normalized) of four-quadrant S-AC multiplication shown in Fig. 4a for (a) varying accuracies at different hyper-parameter values i.e. $S = 1$ and $S = 3$ for $\mathbf{w} = [-0.5, 0.5]$, (b) close compliance between multiplier curves at different operating regimes for $S = 3$ and, (c) multiplier curves for $\mathbf{w} = [-1, -0.75, -0.5, -0.25, 0.25, 0.5, 0.75, 1]$ at $S = 3$.
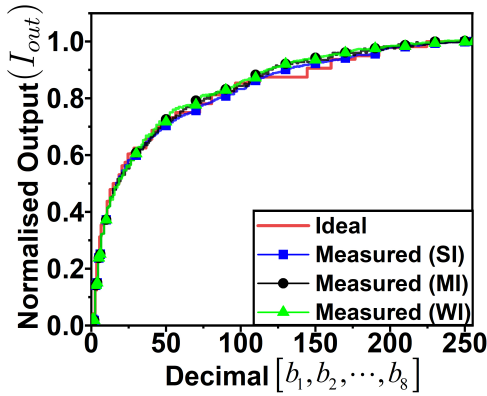

Fig. 9. Measurement result of 8-bit compressive DAC shown in Fig. 5a.

the energy per operation increases while an optimal balance between power and speed is always obtained in MI region.

The most significant errors introduced in the operation of S-AC circuits are represented by mismatches, noise and power-supply variations. As a result of these undesired effects, the functionality of the circuits can be severely affected by additive errors. In S-AC circuits, the margin between the shapes obtained in the SI and WI regimes takes into account all the variations due to second-order effects. This crucial feature allows the S-AC circuits to preserve the inherent shape of the implemented function.

E. Performance analysis

Temperature variation: We compare the effect of nominal temperature variation on S-AC units. Fig. 10 shows the mea-

TABLE I
ENERGY/OPERATION @VDD=1.1 V

Operation	Energy/Operation (pJ)		
	(SI)	(MI)	(WI)
Multiplication	5.23	4.01	0.57
Division	5.23	4.01	0.57
Dynamic ReLU	10.46	8.02	1.13

sured characteristic curves of S-AC based ReLU, Multiplier and DAC at different temperature points respectively. One can observe that even though there is a slight variation that can be attributed to the current mirrors in the desired curves but the overall characteristic shape is preserved.

Power & Task-Energy Efficiency: Fig. 11a shows a comparison plot between measured and simulated power of S-AC based unit when the operating current is varied such that circuit operations moves from WI to SI regime. It can be observed that the power consumption increases when circuit operation shifts from WI to SI regime.

Slew Rate: With the increase in the number of S-AC blocks, the corresponding slew rate and bandwidth increases as the number of inputs and the overall current available to charge the node capacitance increases. This results in an overall reduction in settling time and can be solely attributed to the constraints imposed by the hyper-parameter C in (2). It can also be noted that as value of this hyper-parameter C decreases i.e. when the circuit operation shifts from SI to the WI regime, the settling time increases because it takes more time for the capacitor at the gate of the output transistor (node V_B in Fig. 2a) to charge with the limited available current.

Settling Time: This settling time (including dead time, slew time, and recovery time) decides the maximum input frequency at which the system can operate (assuming all the operations to be performed are done parallel) and can be given by (12)

$$f_{max} = \frac{1}{\max\left(t_{settling,rise}, t_{settling,fall}\right) + \Delta t} \qquad (12)$$

Here, Δt is the margin for the unexpected error that can arise due to circuit variations [33]. It can safely be assumed to be between 5% of $t_{settling}$. Fig. 11b shows the measured settling time of a S-AC based unit when the operating current is varied such that the circuit moves from WI to SI region of operation. It can be observed that as the operating regime moves from WI to SI, the time required to charge the capacitance node improves. Hence the circuit can operate at a higher speed. Fig. 11c shows the variational performance efficiency (PE) and system efficiency (SE) when the circuit operating regime shifts from WI to SI. Note that PE increases with increase in operating current while SE deteriorates.
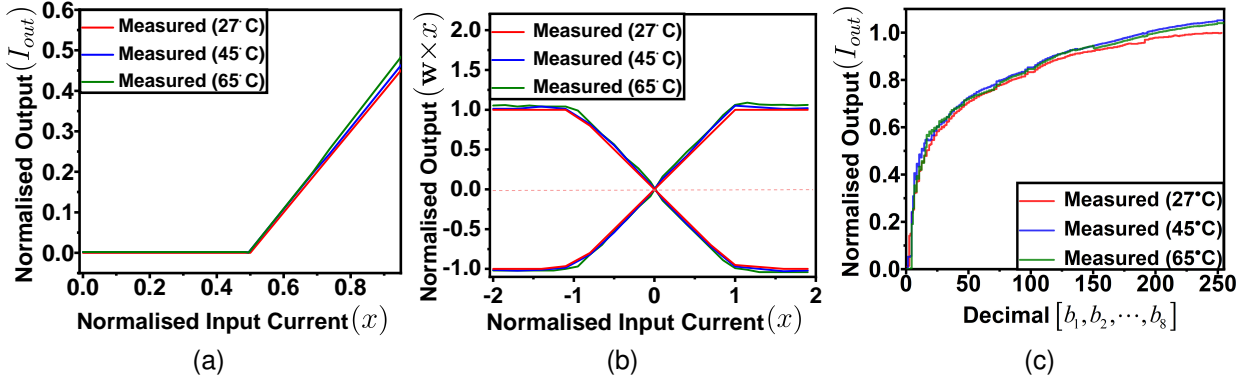
Fig. 10. Temperature measurement result for (a) S-AC based soft ReLU, (b) S-AC based four-quadrant multiplier and (c) S-AC based 8-bit log_2 DAC.
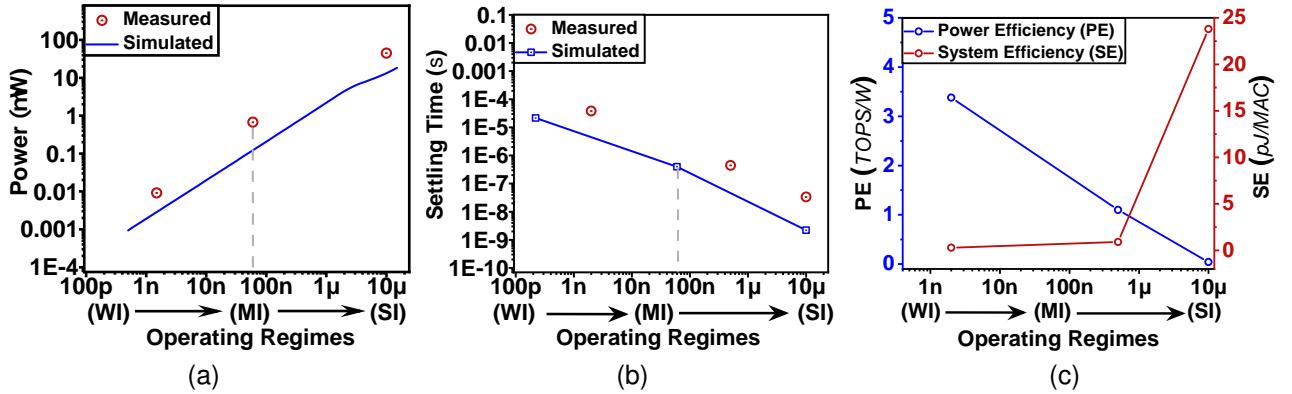

Fig. 11. (a) Power consumption, (b) Settling time, (c) Performance and System Efficiency of a single S-AC unit biased in different operating regimes.

V. REGRESSION RESULTS

In this section we demonstrate the functionality of the S-AC building blocks for a simple neural network regression task. Fig. 12a shows the neural architecture of a S-AC based 3-layer neural network for 6 hidden nodes and its circuit implementation. Inputs are converted into differential compressive form and passed to the hidden nodes. Here, S-AC compressive memory units are used to store weights in the compressed log domain. This in-memory computing architecture also reduces the energy wasted in moving data to and from the memory. For demonstration we use the S-AC architecture to learn a two-dimensional non-linear function given by

$$Y = \sin(2\pi x_1)\sin(2\pi x_2) \tag{13}$$

Fig. 12b shows that the output of the S-AC based regression task matches closely with a software-based expected outcome. Also, since S-AC blocks are regime-independent, the S-AC neural network architecture was also verified to be invariant to the biasing regime of the transistors.

VI. CONCLUSION

In this work, we proposed a novel shape-based analog approximate computing framework for designing analog machine learning processors. Like digital designs, the S-AC framework allows the user to trade-off precision of computation and speed of computation with energy and area. Also, the S-AC

based analog functions have been shown to remain invariant to biasing conditions and operating temperature. At a system level, the overall efficiency (power and speed) can be adjusted by adjusting a global bias current which in turn will bias the transistors in different operating regimes. As a result, the architecture is well suited for scalable chip-in-the-loop training of each analog processor to compensate for fabrication artifacts. During training, the S-AC based design can be biased in SI without changing the overall function for faster learning while the same system can be operated in WI for energy-efficient inference. The system parameters stored on a digital memory can be updated by an external digital processor. We reported the basic building blocks (ReLU and multiply-accumulate) of an ML processor using S-AC circuits and we also showed the implementation of S-AC compressive memory which mimics the computation using Bfloat16 and IEEE 754 single-precision number systems. Table II compares the measured performance of these basic building blocks with similar designs reported in the literature. As a proof of concept, we demonstrated the functionality of a 3-layer neural network regression task using S-AC basic building blocks. Our future works will include the demonstration of generic programmable architecture for deep neural networks.
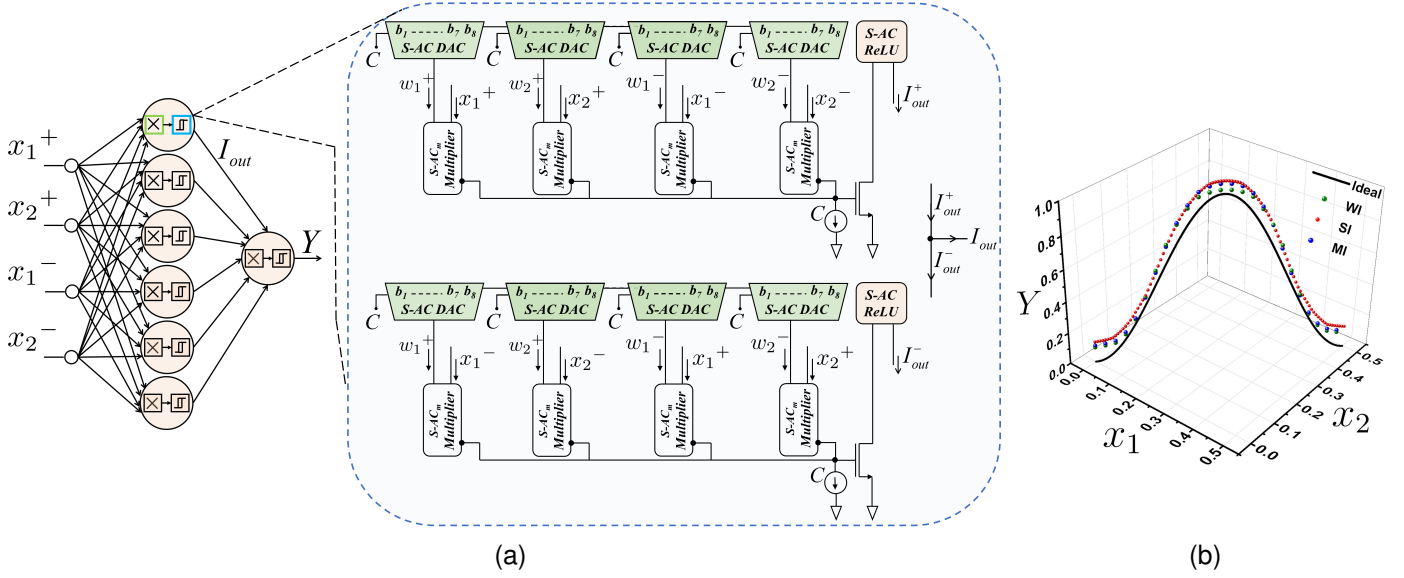
Fig. 12. (a) 3-layer neural network architecture and unit node implementation using S-AC DAC (Fig. 5a), S-AC ReLU (Fig. 3) and S-AC multiplier (Fig. 4a), (b) Output of Sine Regression through the S-AC architecture presented in Fig. 12a.

TABLE II
MODULE-WISE COMPARISON OF ANALOG DESIGNS

			Non-linearity		
Referred Work	[22]	[23]	[24]	[25]	This Work
Operating Regimes	WI, SI	-	WI	-	WI, MI, SI
Design based on	Current mode	Current mode	Voltage mode	-	Shape based
Technology (μm)	3	0.5	0.18	0.065	0.18
Area (μm^2)	100800	264600	-	2000000	190.46
Supply (V)	2.5	-	1.3 to 1.8	-	1.1 to 1.8
Power	-	-	149-150μW	1.2mW	18.2nW - 89.4μW
Result Type	simulated	simulated	measured	measured	measured
			Analog Multiplier		
Referred Work	[26]	[27]	[28]	[24]	This work
Operating Regimes	SI	SI	WI, MI, SI	WI	WI, MI, SI
Design based on	MOSFET Sq law	Pool circuit	Current mode	Voltage mode	Shape based
Technology (μm)	2	2	0.18	0.18	0.18
Area (μm^2)	10670	-	600/800	-	885.74
Supply (V)	5	5	1.2	1.1 to 1.8	0.7 to 1.8
-3dB Bandwidth	115kHz	7MHz	79.6 MHz/59.7 MHz	14kHz	15.12 MHz
Power	1mW	-	60μW/75μW	234μW	546nW - 268.2μW
Result Type	measured	simulated	simulated	measured	measured
			Log-DAC		
Referred Work	[29]	[30]	[31]	[32]	This Work
Operating Regime	-	-	-	WI	WI, MI, SI
Conversion Technique	Current attenuator	Pseudo log amp	Memristors	Sub-threshold transistor	Shape based
Technology (μm)	1.2	0.18	0.18	0.18	0.18
Area (mm^2)	1.5	1.5	-	0.0069	0.00127
Supply (V)	5	1.65	1.8	1.8	1.1 to 1.8
Implemented Resolution (bit)	8	4	4	8	8
Power	6mW@1MHZ	-	100μW@100kHz	3.11μW@5MHz	138nW - 536.4 μW @3.37MHz
Utility type	log-DAC	log-DAC	log-DAC	log-DAC	log-generic-compressive-DAC
Result Type	measured	measured	simulated	measured	measured

REFERENCES

[1] K. Freund, "IBM Research Says Analog AI Will Be 100X More Efficient. Yes, 100X," Sept. 23, 2021 [Online]. [Online]. Available: https://www.forbes.com/sites/karlfreund/2021/09/23/ibm-research-says-analog-ai-will-be-100x-more-efficient-yes-100x/?sh=61b5e23b129b

[2] C. Toumazou, F. J. Lidgey, and D. Haigh, *Analogue IC design: the current-mode approach.* Presbyterian Publishing Corp, 1990, vol. 2.

[3] A. James, K. Kemp, and e. a. Robertson, Dave, "Decadal Plan for Semiconductors," Jan. 2021 [Online]. [Online]. Available: https://www.src.org/about/decadal-plan/

[4] 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.

[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] J.-J. Sit and R. Sarpeshkar, "A micropower logarithmic A/D with offset and temperature compensation," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 2, pp. 308–319, 2004.

[7] M. Gu and S. Chakrabartty, "Subthreshold, varactor-driven cmos floating-gate current memory array with less than $150 ppm/\degree k$ temperature sensitivity," *IEEE journal of solid-state circuits*, vol. 47, no. 11, pp. 2846–2856, 2012.

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

[9] G. Cauwenberghs and M. Bayoumi, *Learning on silicon: Adaptive VLSI neural systems.* Springer Science & Business Media, 1999, vol. 512.

[10] J. Y. Yam and T. W. Chow, "A weight initialization method for improving training speed in feedforward neural network," *Neurocomputing*, vol. 30, no. 1-4, pp. 219–232, 2000.

[11] 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.

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

[13] 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

[14] 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.

[15] S. Wang and P. Kanwar, "BFloat16: The secret to high performance on Cloud TPUs," [Online]. [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus

[16] A. R. Nair, P. K. Nath, S. Chakrabartty, and C. S. Thakur, "Multiplierless MP-Kernel Machine For Energy-efficient Edge Devices," *arxiv*, vol. abs/2106.01958, 2021. [Online]. Available: https://arxiv.org/abs/2106.01958

[17] F. J. Kub, K. K. Moon, I. A. Mack, and F. M. Long, "Programmable analog vector-matrix multipliers," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 1, pp. 207–214, 1990.

[18] C. R. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE Journal on emerging and selected topics in circuits and systems*, vol. 1, no. 3, pp. 403–411, 2011.

[19] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Applied Physics Reviews*, vol. 7, no. 3, p. 031301, 2020.

[20] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[21] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov, "High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4782–4790, 2017.

[22] S.-Y. Lin, R.-J. Huang, and T.-D. Chiueh, "A tunable Gaussian/square function computation circuit for analog neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 3, pp. 441–446, 1998.

[23] M. Shaterian, C. M. Twigg, and J. Azhari, "An MTL-Based Configurable Block for Current-Mode Nonlinear Analog Computation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 9, pp. 587–591, 2013.

[24] R. J. D'Angelo and S. R. Sonkusale, "A Time-Mode Translinear Principle for Nonlinear Analog Computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 9, pp. 2187–2195, 2015.

[25] N. Guo, Y. Huang, T. Mai, S. Patil, C. Cao, M. Seok, S. Sethumadhavan, and Y. Tsividis, "Energy-Efficient Hybrid Analog/Digital Approximate Computation in Continuous Time," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 7, pp. 1514–1524, 2016.

[26] N. Saxena and J. Clark, "A four-quadrant CMOS analog multiplier for analog neural networks," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 746–749, 1994.

[27] S.-I. Liu and C.-C. Chang, "CMOS analog divider and four-quadrant multiplier using pool circuits," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 9, pp. 1025–1029, 1995.

[28] C. Popa, "Improved Accuracy Current-Mode Multiplier Circuits With Applications in Analog Signal Processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 443–447, 2014.

[29] J. Guilherme and J. Franca, "A logarithmic digital-analog converter for digital CMOS technology," in *Proceedings of APCCAS'94 - 1994 Asia Pacific Conference on Circuits and Systems*, 1994, pp. 490–493.

[30] S. Purighalla and B. Maundy, "84-dB Range Logarithmic Digital-to-Analog Converter in CMOS 0.18- μm Technology," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 5, pp. 279–283, 2011.

[31] L. Danial, N. Wainstein, S. Kraus, and S. Kvatinsky, "Breaking Through the Speed-Power-Accuracy Tradeoff in ADCs Using a Memristive Neuromorphic Architecture," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 5, pp. 396–409, 2018.

[32] M. G. Jomehei, S. Sheikhaei, E. H. Hafshejani, and S. Mirabbasi, "A Low-Power Logarithmic CMOS Digital-to-Analog Converter for Neural Signal Recording," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.

[33] P. Kumar, K. Zhu, X. Gao, S.-D. Wang, M. Lanza, and C. S. Thakur, "Hybrid Architecture Based on Two-dimensional Memristor Crossbar Array and CMOS Integrated Circuit for Edge Computing," *npj 2D Materials and Applications*, vol. 6, no. 1, pp. 1–10, 2022.