Function Approximation Using Analog Building Blocks in Flexible Electronics

Paula Carolina Lozano Duarte *, Aradhana Dube *, Georgios Zervakis [†], Mehdi Tahoori *, Sani Nassif [‡]

*Karlsruhe Institute of Technology, [†]University of Patras, [‡]Radyalis LLC

*{paula.duarte, aradhana.dube, mehdi.tahoori} @kit.edu,
[†]zervakis@ceid.upatras.gr, [‡]srn@radyalis.com,

Abstract—**Function approximation is crucial in Flexible Electronics (FE), where applications demand efficient computational techniques within strict constraints on size, power, and performance. Devices like wearables and compact sensors are constrained by their limited physical dimensions and energy capacity, making traditional digital function approximation challenging and hardware-demanding. This paper addresses function approximation in FE by proposing a systematic and generic approach using a combination of Analog Building Blocks (ABBs) that perform basic mathematical operations such as addition, multiplication, and squaring. These ABBs serve as the foundation for constructing splines, which are then employed in the creation of Kolmogorov-Arnold Networks (KANs), improving the approximation. The analog realization of KAN offers a promising alternative to digital solutions, providing significant hardware benefits, particularly in terms of area and power consumption. Our design achieves a 125× reduction in area and a 10.59% power saving compared to a digital spline with 8-bit precision. Results also show that the analog design introduces an approximation error of up to 7.58% due to both the design and parasitic elements. Nevertheless, KANs are shown to be a viable candidate for function approximation in FE, with potential for further optimization to address the challenges of error reduction and hardware cost.**

Index Terms—**Function Approximation, Analog Building Blocks, Kolmogorov-Arnold Networks, Flexible Electronics**

I. INTRODUCTION

Flexible Electronics (FE) have emerged as a promising technology with the potential to revolutionize a wide range of applications, including wearable devices, sensors, and next-generation medical technologies [1]–[3]. FE, based on materials like Indium Gallium Zinc Oxide (IGZO), offers advantages such as lightweight, low cost manufacturing, and adaptable form factors, making it suitable for applications where traditional silicon-based electronics may fall short [4, 5]. However, the integration of FE into practical, high-performance systems requires overcoming several inherent challenges, primarily the need for low power, small footprint, and efficient near-sensor or on-sensor processing [6, 7].

One of the key limitations of FE is the larger area required for complex circuits compared to traditional silicon-based systems due to larger feature sizes, which makes adopting digital processing techniques bulky and costly for certain applications. As FE applications demand greater functionality, digital hardware becomes increasingly challenging in terms of both cost and performance [4, 6, 8, 9].

The ability to approximate functions efficiently is crucial for many FE applications, such as monitoring biological signals in wearables (e.g., calculating average heart rate or temperature) or detecting environmental parameters in compact IoT sensors (e.g., approximating pollution levels or humidity) [1, 6, 10]. Function approximation enables systems to process signals in real-time while using minimal power and hardware. However, traditional digital function approximation methods can be too power-hungry and hardware-demanding for small-scale, low-cost FE applications. Kolmogorov-Arnold Networks (KANs) offer a powerful framework for function approximation, providing an efficient and systematic way to model complex relationships in data through the use of spline functions [11]. KANs offer the flexibility to approximate any given function, making them ideal for use in a variety of applications [12].

The key challenge, however, lies in efficiently realizing these approximations in hardware, particularly given the constraints of FE. While digital implementations of KANs are straightforward, their associated power consumption and hardware costs often present significant barriers, and this has been studied as a co-optimization problem [13, 14]. Analog realizations of KANs must address issues such as maintaining precision, minimizing power consumption, and dealing with parasitic elements inherent to circuit designs.

This work proposes a novel solution by using analog circuits to implement KANs for function approximation in FE. The key feature of our approach is that it is systematic and generic, enabling analog approximations of any function through a standardized design process that brings structure and precision to analog function approximation. As a result, our approach overcomes the limitations of traditional methods and provides a reliable way to implement analog function approximation in different applications. We present the development of Analog Building Blocks (ABBs) that perform fundamental mathematical operations—such as addition, subtraction, multiplication, and squaring—which are essential for creating splines. By using these ABBs, we construct a hardware-efficient KAN implementation that reduces area by 125 × and power consumption by 10.59% compared to a digital spline with 8-bit precision. Despite introducing an approximation error of up to 7.58%, our approach provides a promising pathway for more efficient and scalable function approximation in FE.

In summary, the main contributions of this work are:

1) Creation of ABBs in FE to perform basic mathematical operations.
2) A systematic and generic approach for creating splines and KANs, enabling analog approximation of any given

function.

3) Comparison of the digital and analog cost of each spline in hardware, including the physical implementation of the analog spline.
4) An analysis of the hardware error and its impact on the overall network performance.

The structure of this paper is as follows: Sec. II, discusses the background. Sec. III presents the methodology we followed from the ABB to the approximation of the splines. Sec. IV analyzes the cost of our implementation and compares it with the digital cost of it, we present as well the error of the approximation and limitations of the work. Finally, Sec. V concludes the work with future research directions.

II. BACKGROUND

A. Flexible Electronics

FE encompass electronic devices designed to maintain functionality while bending, stretching, or conforming to various surfaces [15]. They are typically built on flexible substrates, such as polyimide, plastics, or thin metal foils, enabling them to adapt to a wide range of applications, from wearable health monitors to compact, portable sensors used in consumer and medical electronics [1]–[3].

The manufacturing process for FE involve modifications of traditional semiconductor fabrication techniques: Instead of rigid substrates, FE are constructed on pliable materials, and certain complex steps—like ion implantation and high-temperature annealing—are often eliminated [4]. For example, PragmatIC Semiconductor has optimized a streamlined version of silicon lithography specifically for flexible devices and by re-moving these costly steps, the manufacturing process becomes faster, more cost-effective, and environmentally sustainable due to reduced resource demands [16]. Furthermore, PragmatIC's FlexIC technology integrates Thin-Film Transistors (TFTs) on polyimide substrates, achieving critical dimensions of 600 nm. This process utilizes a high-k dielectric material that sup-ports standard IC input voltages, enabling compatibility with existing systems while prioritizing ultra-thin and low-energy designs [16].

One of the materials used in FE is Indium Gallium Zinc Oxide (IGZO), an amorphous oxide semiconductor with high electron mobility, transparency, and compatibility with low-temperature processes [17, 18]. These characteristics make IGZO suitable for flexible applications, though its limitations, including its restriction to N-type transistors, impose design constraints [5]. As such, FE is unlikely to replace silicon-based technologies but to complement them, enhancing hybrid systems where flexibility, low power, and form factor are priori-tized over the ultra-precision found in silicon-based circuits [7].

FE are well-suited for use cases that require real-time, local-ized processing in constrained environments, such as monitor-ing biological signals in wearable health devices or detecting environmental parameters in compact IoT sensors [1, 4, 6]. To meet the specific demands of these applications, FE devices often need efficient function approximation capabilities to perform tasks like signal smoothing, feature extraction, and

basic mathematical transformations. For instance, wearables may need to compute average heart rate or temperature values, while environmental sensors might use threshold-based approx-imations for pollution levels or humidity [10].

To address the need for efficient, low-power function approx-imation in FE, we propose an analog approach using KANs, which can effectively model complex, continuous functions in a hardware-efficient manner. KANs offer a promising solution for systematically implementing these functions in FE, as they allow for real-time analog computation while minimizing the area and power consumption of the system.

In this work, we use PragmatIC's FlexLogIC process for rapid, FE production. This platform allows for a fast turnaround, typically achieving custom designs in under six weeks [16]. The FlexIC platform provides efficient, pragmatic solutions for FE, especially in applications that benefit from a low-cost, high-volume production model.

B. Kolmogorov-Arnold Networks

KANs are inspired by Kolmogorov-Arnold representation theorem [19]. In [11], a neural network model is proposed where learnable activation functions are used, unlike Multilayer Perceptron (MLP) which has a fixed activation function. This replaces the linear weight matrix in MLP with a learnable function dependent on a single variable. KAN approximates a multivariate continuous function as a composition of continuous functions of a single variable added together. To implement these univariate functions, KAN utilizes Bézier spline with learnable coefficients [11]. The KAN layer uses the following equation:

$$\phi(x) = w_b b(x) + w_s \, \text{spline}(x) \tag{1}$$

where $\phi(x)$ is the sum of basis function b(x) and the spline function. The spline in eq. 1 is a linear combination of Bézier splines such that:

$$\text{spline}(x) = \sum_i c_i B_i(x) \tag{2}$$

While the basis function and multiplication can be implemented by multiply and accumulation using a crossbar array [20], the implementation of spline in analog domain is the focus. The Bézier splines have a set of discreet control points by which the smooth continuous curve is defined. The spline can have n-control points which define the order of the spline. In this work we implement a second-order Bézier spline [21] given by:

$$\mathbf{B}(x) = P_0(1 - x)^2 + 2P_1(1 - x)x + P_2 x^2 \tag{3}$$

where P_0, P_1, P_2 are the control points of the second order Bézier spline and, x the input. Choosing a second-order Bézier spline reflects a balance between hardware cost and approxima-tion accuracy: while higher-order splines could improve accu-racy, they would require more hardware resources, increasing area and power consumption. In our case, the second-order spline minimizes hardware and simplifies Eq. 3 to:

$$\mathbf{B}(x) = P_0 + (P_1 - P_0)2x + (P_0 - 2P_1 + P_2)x^2 \tag{4}$$

Eq. 4 becomes the basis for the implementation of second order Bézier splines in the analog domain using FE.
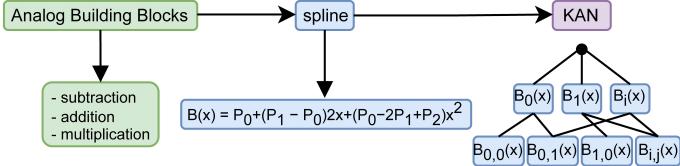
Fig. 1. Proposed methodology to build KAN. The basis for KAN is spline which is interpreted by the second order equation.

C. Related work

Due to the recent publication of [11], at the time of this work, only two studies have explored the hardware implementation of KAN. The first study [14] proposes a mixed analog-digital circuit approach, utilizing Look-up Tables (LUT) and Analog Computing-In-Memory (ACIM). However, LUTs face scalability challenges, as their size grows exponentially with the number of classes, making them memory-intensive. The second study [13] explores a fully digital circuit implementation, which also relies on LUTs. Both studies incorporate a co-design methodology with software optimization. In this work we do not replicate the designs presented in [14] or in [13], as FE require simpler and more resource-efficient designs due to power and area constrains. Instead, to establish a baseline for comparison, we opted for a digital implementation of Eq. 4 in FE, incorporating the constraints of this technology while excluding the co-optimization strategies used in prior studies.

Our work presents the first research of a fully analog-domain implementation, leaving the exploration of co-optimization with software for future research. Furthermore, we are the first to explore this design using non-silicon-based technology, which reduces power consumption and requires the creation of simpler and more efficient circuit designs.

III. METHODOLOGY

This section describes the methodology employed to implement the KAN in the analog domain, starting with the selection of basic analog components. These components, referred to as Analog Building Blocks (ABBs), are designed to perform essential operations such as inversion, subtraction, addition, multiplication, and squaring. We explain the design choices behind each operation and discuss how they contribute to the overall system's functionality. After covering the ABBs, we outline the hardware implementation approach for spline approximation, highlighting how the chosen formula reduces the number of required blocks and enhances both area and power efficiency. Finally, we present the KAN implementation derived from these splines, establishing a systematic framework for an analog approximation of any given function.

A. Analog Building Blocks

To implement KAN in the analog domain, we started with basic analog components, which we refer to as "Analog Building Blocks". These ABBs perform simple mathematical operations and form the foundation of our design approach, in Fig. 1 we present the methodology followed. In this section, we present only the ABBs used to approximate second-degree

splines(Eq. 4). However, we have also developed nonlinear functions, such as softmax, sigmoid, and tanh functions, and more complex operators like integrators, which could be applied in future iterations and other applications.

Due to large feature sizes in the target flexible IGZO technology which only includes N-type transistors, we adopted a Resistor-Transistor Logic (RTL) design strategy to minimize the number of necessary components for building blocks. Another constraint we faced was the limitation to only N-type transistors, resistors, and capacitors. Considering these limitations, Fig. 2 presents our proposed designs for inversion, subtraction, addition, multiplication, and squaring operations.

Inversion: we present a design with only two transistors and one resistor. The schematic of the design is shown in Fig. 2 a).

Subtraction: the design uses two transistors, one resistor, and a bias voltage, V_{bias}, which is necessary to expand the operating range. The schematic of this design is shown in Fig. 2 b).

Addition: we employ the logic of $A + B = A - (-B)$. We first use the inversion design on B, followed by the subtraction design as $A - (-B)$. The schematic of this design is shown in Fig. 2 c).

Multiplication: we base our design on the Gilbert cell [22, 23], which provides an optimal solution for analog multiplication in the voltage domain. We adapted the design to use only N-type transistors and added a final subtraction $V_{dd} - A$ to improve accuracy across multiple values of A. The schematic of this design is shown in Fig. 2 d).

Squaring: we further optimized the multiplication design. Knowing the input range in advance allows us to anticipate the output range, simplifying design optimization and reducing area and power consumption. We removed the final subtraction block, $V_{dd} - A$, and replaced it with a fixed V_{bias}. The schematic of this design is shown in Fig. 2 e).

B. Implementation of the spline from multiple ABBs

For the hardware implementation of each spline, we use the formula presented in Eq. 4. We chose this equation because it allows us to reduce the number of blocks required, needing only two multiplications, one squaring, and two additions (a total of five blocks). In contrast, using Eq. 3 would require two subtractions, three multiplications, two squaring, and two additions (a total of nine blocks). In Sec. IV-B, we present the savings achieved by making this choice.

First, we decompose the expression into three additive sub-blocks: a) P_0, b) $(P_1 - P_0)2x$ and c) $(P_0 - 2P_1 + P_2)x^2$.

- The first sub-block is simply P_0, which, as a constant, we interpret as a voltage input.
- The second sub-block consists of $(P_1 - P_0)2x$, where $(P_1 - P_0)2$ is a constant. We handle this constant as a voltage input and then multiply it by x.
- The last sub-block, $(P_0 - 2P_1 + P_2)x^2$. Here, $P_0 - 2P_1 + P_2$ is also a constant and will be provided as a voltage input. To implement this final sub-block, we first square x by multiplying it by itself. We then apply our design shown in Fig. 2 e) to perform the multiplication with the constant $(P_0 - 2P_1 + P_2)x^2$, which helps to reduce area and power consumption while maintaining precision.
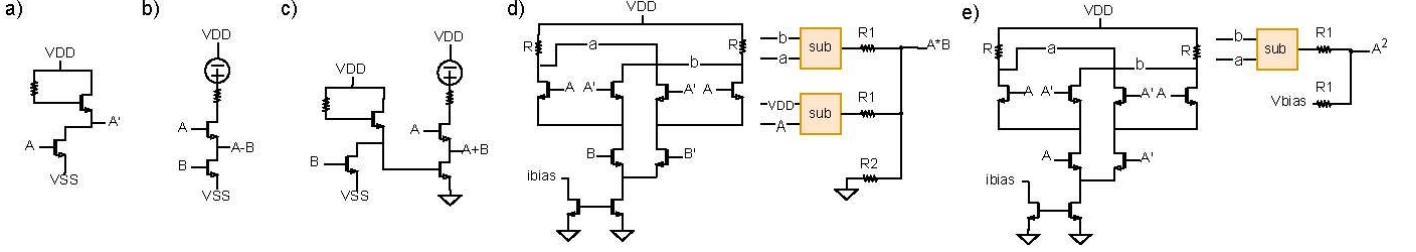
Fig. 2. Schematic of: a) Inversion b) Subtraction c) Addition d) Multiplication e) Squaring
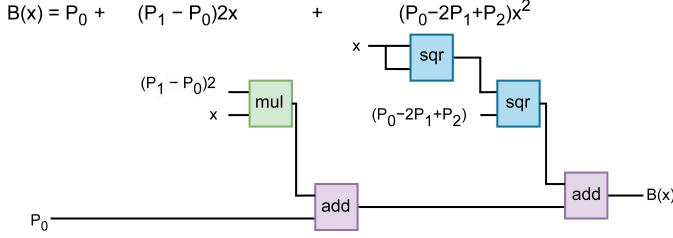
Fig. 3. Architecture of the spline implementation

Fig. 4. Architecture of the KAN implementation

Finally, we sum each of the sub-blocks to construct the spline, resulting in the approximate second-degree function. The full architecture of the implementation is presented in Fig. 3.

To ensure proper functionality, we carefully designed the input ranges. All ABBs have input ranges from $[-0.5, 0.5]$ as $[-V_{dd}/2, V_{dd}/2]$. We need to consider this for the implementation of Eq. 4.

For clarity, we examine each of the sub-blocks used to construct the spline, resulting in an approximate second-degree function. Starting with the first sub-block P_0, we know there will be no range issues with this term. For the second sub-block, $(P_1 - P_0)2x$, we know that $(P_1 - P_0)2 \in [-0.5, 0.5]$ leading to the first range constraint:

$$P_1 - P_0 \in [-0.25, 0.25]$$

The second range constraint concerns our input $t \in [-0.5, 0.5]$ For the third sub-block, by reformatting, we obtain:

$$P_2 - 2P_1 + P_0 = P_2 - P_1 - (P_1 - P_0)$$

Thus, the range of $P_2 - P_1 - (P_1 - P_0)$ is constrained to $[-0.5, 0.5]$To determine the range of $P_2 - P_1$, we express it as:

$$P_2 - P_1 = (P_2 - 2P_1 + P_0) + (P_1 - P_0)$$

Since the ranges for both $P_2 - 2P_1 + P_0$ and $(P_1 - P_0)$ are known, we can calculate the range of $P_2 - P_1$ by summing the maximum and minimum values of these terms.

- Minimum value of $P_2 - P_1$:

$$-0.5 + (-0.25) = -0.75$$

- Maximum value of $P_2 - P_1$:

$$0.5 + 0.25 = 0.75$$

Therefore, we have:

$$P_2 - P_1 \in [-0.75, 0.75]$$

Finally, it is important to mention that by considering the trainable values of KAN (P_i) as inputs in our architecture, the hardware design we present is independent of both the training process and these values. Therefore, no redesign is necessary when retraining the model; only the inputs need to be updated. For future work, it would be beneficial to perform a co-optimization with fixed P_i values, where optimization occurs within each ABB, as was done with the squaring block optimization, based on known input and output ranges.

C. Implementation of the KAN from multiple splines

Once the splines are created, constructing the KAN requires applying Multiply-Accumulate (MAC) operations. In analog design, this MAC is achieved by connecting resistors in series with each spline and placing them in parallel to each other. To implement a weighted MAC, the resistor values can be adjusted according to the desired weighting for each spline. Alternatively, if uniform weighting is needed, the resistors should have the same value, ensuring each spline contributes equally to the MAC output [24]. This approach enables flexibility in controlling the contribution of each spline within the KAN structure, aligning with a future co-design. The architecture of the KAN is presented in Fig. 4.

IV. RESULTS AND ANALYSIS

A. Simulation Setup

Simulations were made using the Cadence Spectre simulator. For the subtractive flexible technology, we employ the PragmatIC FlexICs PDK second-generation Helvellyn 2.1.0 technology [16]. All simulations are performed with V_{dd} of 1.0V, V_{ss} of -1.0V, temperature of 27°C and our V_{in} is a slope from -0.5V to 0.5V with a rise up time of 10ms. In Tab. I we present the sizes of each design used in the spline. We do not add the block of the adder, since its design is the result of combining one inversion and one subtraction block.

TABLE I
DESIGN PROPERTIES OF THE ANALOG SPLINE

Block	Component	Size
INV	Transistors	W = 20 μm, L=600nm
	R	r = 250k Ω, W=2.4μm, L=3μm
SUB	Transistors	W = 20 μm, L=600nm
	R	r = 30K Ω, W=20μm, L=3μm
	vbias = 0.5V	
MUL	Transistors	W = 5 μm, L=600nm
	R	r = 99.97M Ω, W=1.4μm, L=719.4μm
	R1	r = 428.57k Ω, W=1.4μm, L=3μm
	R2	r = 40M Ω, W=1.4μm, L=291.8μm
	ibias = 1 μA	
SQR	Transistors	W = 5 μm, L=1.3μm
	R	r = 25.97M Ω, W=1.4μm, L=201.4μm
	R1	r = 15.45M Ω, W=1.4μm, L=123.8μm
	ibias = 1 μA	
	vbias = -210mV	

TABLE II
HARDWARE COST OF AN ANALOG AND DIGITAL SPLINE

Analog spline	area (mm²)	0.073
	power (μW)	238.5
Digital spline	area (mm²)	9.111
	power (μW)	266.735

Since the range of the ABBs inputs was known from the implementation, we were able to optimize each circuit for this specific range. For example, the input of the squaring circuit is always between [-0.5,0.5] with an output range of [0,0.25]. In contrast, the multiplication circuit's output range can include negative as well as positive values, requiring a more robust circuit. This explains why the squaring circuit has lower resistor values and a higher number of transistors, and why in some cases, is more convenient to use this circuit to perform multiplications instead of the multiplication circuit.

To compare the hardware cost of the analog spline with the digital, we created a standard cell library using PragmatIC PDK with a V_{dd} of 1V, synthesized it using Synopsys Design Compiler S-2021.06. For simulation and power analysis, we used VCS T-2022.06 and PrimeTime T-2022.03. Based on Eq. 4 we developed an 8-bit fixed-point digital spline with 6-bit decimal bits and one signed bit, implemented with sequential logic using verilog. The architecture of the digital spline was determined by the synthesis tool.

B. Hardware costs

In table Tab. II, we compare the hardware cost of the analog spline presented in this work to a 8-bit digital spline. We can see that the analog implementation's area cost is significantly lower than the digital one, achieving a 125 × reduction. In terms of power, the analog implementation is also more efficient than the digital, with a 10.59% reduction. The physical implementation of the analog spline is presented in Fig. 5. As mentioned in the
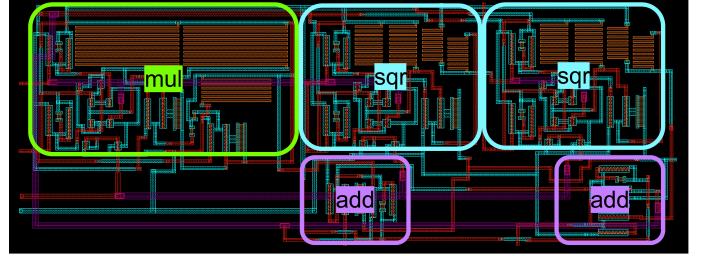

Fig. 5. Layout of a spline following the architecture presented in Fig. 3

TABLE III
ERROR OF EACH ABB

Block	Normalized Mean Percentage Error (NMPE)
INV	2.66%
SUB	-4.7%
ADD	-4.23%
MUL	-6.99%
SQR	-7.93%

implementation section, the selection of implementing Eq. 4 and not Eq. 3, reduces area up to 46% and achieves power savings of 45.7%.

C. Approximation error

Since the target of our ABBs is to approximate functions, we expect an error from this approximation. We also analyze where the error originates. We use the Normalized Mean Percentage Error (NMPE) metric, since it quantifies the average error as a percentage of the function's range, measuring the magnitude and direction of errors. This relative measure is particularly useful for evaluating function approximation within normalized signal ranges.

Error from each ABB: We first measured the error in each block, as presented in Tab. III. A negative NMPE indicates that the approximation tends to underestimate the function, while a positive value suggests that the approximation overestimates it.

Error of the full spline: When we integrated all our ABBs together, we observed that the error is not commutative, but in many cases, it compensates. We ran three different scenarios with predefined values for P_0, P_1 and P_2, and the maximum error was -7.58%, observed in Scenario C. The results for each scenario, including the values of P_0, P_1 and P_2, along with the corresponding NMPE, are summarized in Tab. IV..

Error of the physical implementation: We also need to consider the error introduced by the layout and its parasitic elements. In the example with the most error, the NMPE from the schematic simulation was -6.34% but with the post-layout simulations, this error increased to -7.58% (value presented in Tab. IV). A comparison between the schematic and post-layout simulations can be seen in Fig. 6.

Impact of Hardware-Induced Error on Function Approximation Using KAN: To investigate the sensitivity of KAN-based function approximation to hardware errors, we developed a Python script using numpy, matplotlib, and scipy CubicSpline. This script simulates a simplified KAN model with one input

TABLE IV

ERROR ANALYSIS OF THE FULL SPLINE ACROSS DIFFERENT SCENARIOS

Scenario	Parameter (P_i)	Value	NMPE (%)
A	P_0	0.1	-4.84
	P_1	0.3	
	P_2	0.8	
B	P_0	0.2	-5.02
	P_1	0.4	
	P_2	0.6	
C	P_0	0.15	-7.58
	P_1	0.4	
	P_2	0.85	

Fig. 7. Comparison of KAN approximations for two target functions, x^2 and e^x, with and without the hardware error. The corresponding Normalized Mean Percentage Errors (NMPE) are included for both cases.
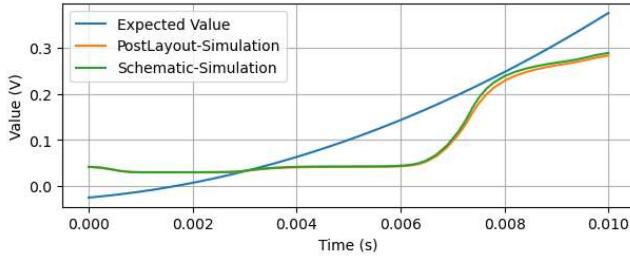
Fig. 6. Comparison of error between schematic and post-layout simulations. The plot shows the impact of parasitic elements and layout errors on the model's accuracy. The 'Expected Value' represents the mathematical result obtained from Eq. 4.

and multiple basis splines. The network was trained to approximate two chosen target functions, x^2 and e^x, using a fixed range of input values $[-0.5, 0.5]$

The KAN implementation features three basis splines, each represented by a cubic spline. A controlled noise factor of -7.58% was applied to each spline during training to emulate hardware error, introducing a stochastic perturbation that mirrors potential inaccuracies in hardware approximations. For comparison, we ran a parallel experiment without noise to observe the baseline performance of the KAN. The output of these simulations is shown in Fig. 7. As we can see, the effect of the hardware error is minimal: less than 1% for x^2 and less than 5% for e^x. It is also important to note that expanding the input range to higher values, such as [-1,1] or [-5,5], significantly reduces the NMPE to below 1% in the error-induced KAN in both cases.

D. Discussion of the results

This work has several limitations that suggest promising directions for future research. First, our focus is limited to approximating second-degree functions, which restricts the range of applications. Extending this approach to higher-order functions would increase its applicability. For higher precision requirements or more complex function approximations, it would be necessary to either use more splines—thus increasing the KAN network—or to improve and add more ABBs.

Since this approach relies on approximation, it naturally introduces some degree of error. While these results demonstrate the resilience of KAN-based approximation to hardware errors, minimizing hardware error requires a robust design that maintains simplicity and is specifically tailored to our FE implementation. This presents a trade-off between error reduction and preserving the hardware benefits of analog design. Balancing these factors is critical in ensuring an optimal approximation with minimal complexity.
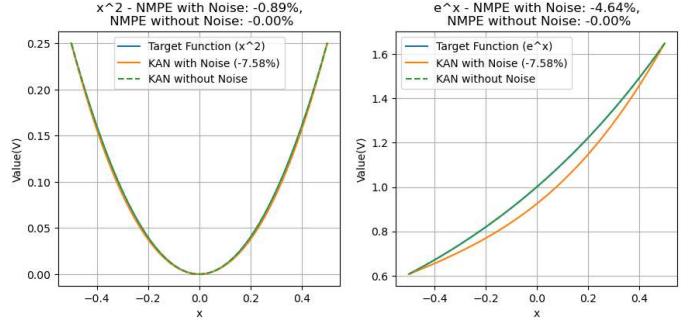
Another limitation is that we did not incorporate co-design strategies, which might have enabled more optimized and efficient configurations, increasing accuracy and versatility, as seen in [13, 14]. Lastly, our design operates in the voltage domain, which limits the operating range. Future studies might explore shifting to the current domain to extend this range and adapt to broader application needs.

These results highlight the benefits of analog computing for target applications such as ABBs and the implementation of splines to create KAN. Nevertheless, it is important to mention that while the objective of this work is to create an analog function approximation, which provides benefits in area and power savings, it comes at the cost of higher error compared to the digital equivalent. Therefore, digital computing may be a better solution when high precision is mandatory.

V. CONCLUSION

Flexible Electronics (FE), while flexible and lightweight, impose challenges such as limited conductivity and larger footprints. Analog approximations offer a promising solution by reducing both power and area. This paper explores the potential of analog Kolmogorov-Arnold Networks (KANs) for function approximation in FE, presenting a systematic and generic design process that enables approximations for any given function. By developing Analog Building Blocks (ABBs) for basic operations such as addition, multiplication, and squaring, we created splines that serve as the foundation of the KAN. Our results show significant hardware improvements, including a 125 × reduction in area and a 10.59% decrease in power consumption compared to a digital 8-bit spline implementation. Although the analog design introduces an approximation error of up to 7.58%, it does not substantially degrade the overall performance of the system. This analog approach offers considerable advantages, particularly in applications where power efficiency and area optimization are prioritized over precision.

REFERENCES

[1] W. Gao, H. Ota, D. Kiriya et al., "Flexible electronics toward wearable sensing," *Accounts of Chemical Research*, vol. 52, 02 2019.

[2] W. Heng, S. Solomon, and W. Gao, "Flexible electronics and devices as human–machine interfaces for medical robotics," *Advanced Materials*, vol. 34, no. 16, p. 2107902, 2022.

[3] W. Gao, S. Emaminejad, H. Y. Y. Nyein et al., "Fully integrated wearable sensor arrays for multiplexed in situ perspiration analysis," *Nature*, vol. 529, pp. 509–514, 2016.

[4] R. Baruah, H. Yoo, and E. Lee, "Interconnection technologies for flexible electronics: Materials, fabrications, and applications," *Micromachines*, vol. 14, p. 1131, 05 2023.

[5] H. Y. Jeong, S. H. Nam, K. S. Park et al., "Significant performance and stability improvements of low-temperature igzo tfts by the formation of in-f nanoparticles on an sio2 buffer layer," *Nanomaterials*, vol. 10, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219850719

[6] A. Nathan, A. Ahnood, M. T. Cole et al., "Flexible electronics: The next ubiquitous platform," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1486–1517, 2012.

[7] P. C. Lozano Duarte, F. Afentaki, G. Zervakis et al., "Design and in-training optimization of binary search adc for flexible classifiers," in *2025 ASPDAC*, 01 2025.

[8] E. Özer, J. Kufel, J. Myers et al., "A hardwired machine learning processing engine fabricated with submicron metal-oxide thin-film transistors on a flexible substrate," *Nature Electronics*, vol. 3, pp. 1–7, 07 2020.

[9] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[10] M. Haghi, S. Danyali, S. Ayasseh et al., "Wearable devices in health monitoring from the environmental towards multiple domains: A survey," *Sensors*, vol. 21, no. 6, 2021.

[11] Z. Liu, Y. Wang, S. Vaidya et al., "Kan: Kolmogorov-arnold networks," *arXiv preprint arXiv:2404.19756*, 2024.

[12] S. Sidharth, A. Keerthana, R. Gokul et al., "Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation," 2024.

[13] D. T. Van, X. H. L. Tran, D. T. Thi et al., "Exploring the limitations of kolmogorov-arnold networks in classification: Insights to software training and hardware implementation," 2024.

[14] W.-H. Huang, J. Jia, Y. Kong et al., "Hardware acceleration of kolmogorov-arnold network (kan) for lightweight edge inference," in *30th Asia and South Pacific Design Automation Conference (ASPDAC '25)*, 09 2024.

[15] L. Kadge. Flexible electronics. [Online]. Available: https://edu.ieee.org/in-dscepes/2019/12/11/flexible-electronics/

[16] Europractice — flexible electronics. [Online]. Available: https://europractice-ic.com/technologies/flexible-electronics/

[17] Z. Ying, H. Yongli, J. Shanshan et al., "Indium–gallium–zinc–oxide thin-film transistors: Materials, devices, and applications," *Journal of Semiconductors*, vol. 42, no. 3, p. 031101, 2021.

[18] Z. Pan, H. Yifan, J. Chen et al., "Approaches to improve mobility and stability of igzo tfts: A brief review," *Transactions on Electrical and Electronic Materials*, vol. 25, 04 2024.

[19] A. Kolmogorov, *On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables*. American Mathematical Society, 1961.

[20] I. Chakraborty, M. Ali, A. Ankit et al., "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2276–2310, 2020.

[21] K. L. Panchuk, T. M. Myasoedova, and M. N. Odinets, "Construction of a discrete planar contour by fractional rational bezier curves of second order," in *Journal of Physics: Conference Series*, vol. 1441, no. 1. IOP Publishing, 2020, p. 012072.

[22] R. Pawase and N. P. Futane, "Cmos gilbert mutiplier for analog signal processing and its fpaa based implementation," in *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, 2018, pp. 623–626.

[23] A. Diaz-Sanchez, J. C. Mateus-Ardila, G. Zamora-Mejia et al., "A four quadrant high-speed cmos analog multiplier based on the flipped voltage follower cell," *AEU - International Journal of Electronics and Communications*, vol. 130, p. 153582, 2021.

[24] D. Weller, M. Hefenbrock, M. Beigl et al., "Realization and training of an inverter-based printed neuromorphic computing system," *Scientific Reports*, vol. 11, 05 2021.