Sequential Printed Multilayer Perceptron Circuits for Super-TinyML Multi-Sensory Applications

Gurol Saglam
guerol.saglam@kit.edu
Karlsruhe Institute Of Technology
Germany

Florentia Afentaki
afentaki@ceid.upatras.gr
University of Patras
Greece

Georgios Zervakis
zervakis@ceid.upatras.gr
University of Patras
Germany

Mehdi B. Tahoori
mehdi.tahoori@kit.edu
Karlsruhe Institute of Technology
Germany

ABSTRACT

Super-TinyML aims to optimize machine learning models for deployment on ultra-low-power application domains such as wearable technologies and implants. Such domains also require conformality, flexibility, and non-toxicity which traditional silicon-based systems cannot fulfill. Printed Electronics (PE) offers not only these characteristics, but also cost-effective and on-demand fabrication. However, Neural Networks (NN) with hundreds of features —often necessary for target applications— have not been feasible in PE because of its restrictions such as limited device count due to its large feature sizes. In contrast to the state of the art using fully parallel architectures and limited to smaller classifiers, in this work we implement a super-TinyML architecture for bespoke (application-specific) NNs that surpasses the previous limits of state of the art and enables NNs with large number of parameters. With the introduction of super-TinyML into PE technology, we address the area and power limitations through resource sharing with multi-cycle operation and neuron approximation. This enables, for the first time, the implementation of NNs with up to 35.9× more features and 65.4× more coefficients than the state of the art solutions.

CCS CONCEPTS

• **Hardware** → **Sequential circuits**.

KEYWORDS

Approximate Computing, Electrolyte-gated FET, TinyML, super-TinyML, Multilayer Perceptron, Printed Electronics

ACM Reference Format:
Gurol Saglam, Florentia Afentaki, Georgios Zervakis, and Mehdi B. Tahoori. 2025. Sequential Printed Multilayer Perceptron Circuits for Super-TinyML Multi-Sensory Applications. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3658617.3697634

1 INTRODUCTION

The integration of intelligence into everyday objects is a growing trend driven by the fourth industrial revolution and the Internet-of-Things. Diverse application domains, from smart packaging to disposable goods, fast-moving consumer goods (FMCG), in-situ monitoring, and low-end healthcare products require low-cost, flexible, and non-toxic solutions that traditional silicon-based systems cannot fulfill [16]. The challenge lies in achieving ultra-low cost and conformality. While silicon VLSI is quite cheap for high volume production, the cost of circuits that integrate intelligence into already low-cost applications like smart bandages must be taken into account, and such a cost can even be less than a cent [6]. These ultra-low cost requirements cannot be met by silicon VLSI systems, which have inherent limitations due to their high non-recurring engineering, fabrication, and packaging costs. Moreover, silicon chips are too rigid to meet the flexibility and porosity needed in these application domains [6].

Printed electronics (PE) have emerged as a promising solution, offering low-cost and flexible hardware fabrication. PE refers to fabrication techniques using printing devices, enabling the realization of large-scale and flexible hardware at a negligible cost [16]. While PE may have limitations in precision manufacturing and feature sizes compared to silicon VLSI systems, it excels in producing cost-effective solutions for applications requiring flexibility and porosity. PE typically operates at frequencies ranging from a few Hz to a few KHz and has micrometer feature sizes [12].

For silicon VLSI systems, Machine Learning (ML) classification systems are further optimized to reduce the area overhead and power consumption. TinyML is an emerging field focusing on deploying models to low-power edge devices. Super-TinyML is an extension of this field where the focus is to enable ML on even more resource-constrained ultra-low-power edge devices. This field, with its focus on ultra-low-power edge devices, is extremely enticing for ML classification systems in PE.

ML classification systems targeted for PE [16] are typically implemented with sensor data as inputs, Analog-to-Digital Converters and a classifier circuit. However, the realization of complex ML models, like Multilayer Perceptron (MLP), induces a large area overhead, making them very challenging to realize in PE [16]. A promising approach to bridge this gap and enable super-TinyML for PE is to use bespoke classifier architectures in which the model's coefficients are hardwired into the hardware description and tailors the circuits to the specific dataset and model. However existing bespoke classifier designs for PE [2–4, 14, 16] are still far from super-TinyML requirements. In [3, 4, 14], the authors combine the bespoke paradigm with the well-known technique of Approximate Computing (AxC) in order to reduce the area and power overhead of combinational MLP classifiers. More precisely, the authors in [4] approximate the multipliers by using hardware-friendly coefficients and further netlist-pruning and voltage overscaling. The authors

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697634

in [3], approximate the weights by a printed-friendly retraining and truncate the accumulators. A comprehensive analysis of neuron minimization by approximation techniques is reported in [14], using quantization, unstructured pruning, and weight clustering.

All of the previous works focus on MLPs for small sensor-based applications due to the increasing complexity of MLPs requiring many large neurons with larger multipliers and/or larger accumulators. More Complex tasks with many input features explode area complexity and power consumption of the current state-of-the-art fully-parallel printed MLP designs. The authors in [16] evaluated also sequential architectures, but followed a conventional design approach and deduced eventually that fully-parallel designs are significantly more efficient. However, the sequential designs of [16] use a large number of registers that are very costly in PE [6].

In this work, we propose a new sequential circuit for super-TinyML customized to PE which minimizes the number of employed registers by exploiting the bespoke nature of MLPs and hardwiring the coefficients using multiplexers. In addition, we approximate the neurons to further reduce the area requirements. As a result, compared to the fully-parallel printed MLP architectures that can support limited models (up to 21 inputs and 130 coefficients), our sequential super-TinyML circuits successfully realize much larger models (up to 753 inputs and 8505 coefficients) within the constraints of PE technology. To the best of our knowledge, there is no prior work on sequential super-TinyML architectures in PE.

Our novel contributions within this work are as follows:

(1) This is the first work that proposes a sequential super-TinyML design dedicated to PE technology and its constraints.
(2) We approximate sequential classifier circuits by limiting some neurons to single-cycle computations. Using both single- and multi-cycle neurons, the proposed super-TinyML design is defined as an approximate hybrid sequential architecture. An automated framework[1] is designed to extract which neurons will be multi-cycle and single-cycle after approximation.
(3) This work enables the realization of printed ML classifiers with up to 753 inputs and 8505 coefficients, 35.9× and 65.4× more than the current state of the art, respectively.

2 BACKGROUND ON PRINTED ELECTRONICS

PE can be divided into two main categories of fabrications, the subtractive and the additive manufacturing processes, where the first has higher printing resolution and the second is simpler and more cost-effective. As depicted in the left of the Figure 1, subtractive method is similar to the traditional lithography-based fabrication technique and uses both additive and subtractive steps. On the other hand, the additive method applies only additive steps where the functional materials are directly deposited on the substrate [12]. The simplicity of the additive process and the fact that the technique is mask-less makes it extremely inviting for sub-cent circuits [7]. However, the simplicity and the mask-less fabrication approach results in low fabrication precision. Thus, the printed circuits have high device latency and low integration density.

In the right of the Figure 1, the piezoelectric drop-on-demand inkjet printer is demonstrated. The drop-on-demand mode releases

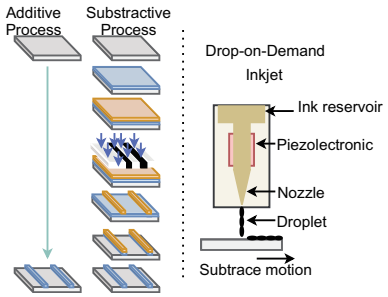

Figure 1: Printed Electronics fabrication techniques.

ink droplets at the intended printing location. Piezo inkjet utilizes piezo elements to induce a shockwave through the ink, resulting in droplet ejection. The piezoelectric type stands out due to its compatibility with various ink formulas, adjustable actuation signals, and a durable print head. Drop-on-demand printing is the most widely employed technique in printing functional materials, since it proves efficient in ink usage, featuring smaller ink droplet diameters (10-50μm) and thus relatively higher printing resolution[17].

Furthermore, while in the past PE was mainly focused on organic transistors, nowadays transparent oxide-based inorganic printed technologies have emerged [6]. Transparent oxides have one order of magnitude higher mobilities than organic semiconducting polymers. In this work, we focus on electrolyte-gated FET (EGFET) printed technology which, compared to the conventional inorganic channel based TFT, has smaller supply voltage ($1V$ instead of $>30V$) and higher mobility values ($126cm^2/Vs$ instead of $<1cm^2/Vs$), making it suitable for battery-powered applications like wearables [6].

3 PROPOSED SEQUENTIAL SUPER-TINYMLS

In this section, we describe thoroughly the proposed architecture and the extraction of model parameters. The sequential super-TinyML architecture is composed of controller logic, the hidden layer, the output layer, and the argmax. The hidden and output layers consist of the neurons of the MLP, where some neurons are multi-cycle and some are single-cycle. In the first subsection, we first describe, respectively, the architecture of the multi-cycle neuron and the single-cycle neuron. After the description of the neuron architectures, the overall architecture and the architectural decisions to minimize the number of employed registers are described. The second subsection consists of the high-level optimizations applied to the MLPs. This subsection includes quantization, redundant feature pruning, and neuron approximation.

3.1 Proposed Sequential Architecture

3.1.1 Multi-cycle neuron architecture. Figure 2 depicts the state-of-the-art [16] and the proposed neuron architectures: on the left, the state-of-the-art, in the center, the multi-cycle neuron, and on the right, the single-cycle neuron. Typically, in a sequential neuron, registers are used to store the weights and biases. However, registers impose a huge area overhead in PE, and thus, their utilization should be minimized [16]. With the low-cost fabrication of PE technology, the weights can be hardwired in bespoke circuits, as in [2]. Using the $state$ signal provided from the controller, a multiplexer is implemented to switch, at each cycle, between the different weights. As it will be discussed in Section 3.2.1, all MLPs used in this work

[1]Our framework is available at https://github.com/gurolsaglam/Sequential-Printed-MLP-Circuits-for-Super-TinyML-Multi-Sensory-Applications_ASPDAC25

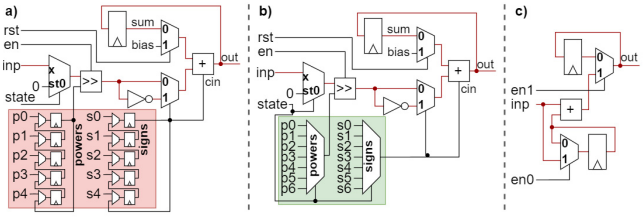

Figure 2: Neuron architectures of a) the state-of-the-art [16] and proposed: b) multi-cycle neuron, c) single-cycle neuron.

consider power-of-2 weights. Thus, the weights are stored as their powers p_i and their signs s_i such that $w_i = s \times 2^{p_i}$ to further reduce the area of the multiplexers. Thus, a barrel shifter is utilized for the multiplication, with the power p_i as input. The shifted input is connected to a multiplexer with and without inverters to subtract the product from the sum when the weight is negative $s_i = 1$. The neuron's sum is stored in a register that is reset to the bias value at the beginning of every new inference, i.e., $reset = 1$.

3.1.2 Single-cycle neuron architecture. The single-cycle neuron architecture is depicted on the right of Figure 2. As it is discussed in Section 3.2.3, based on an offline statistical analysis, we find the two most-important inputs per neuron and their corresponding expected leading-1 position using the training dataset and the absolute weights of each neuron. Two enable signals, $en0$ and $en1$, are used to signify that an important input has arrived. When $en0 = 1$, the neuron's first most-important input has arrived, and the expected leading-1 bit is stored in a 1-bit register. When $en1 = 1$, the neuron's second most-important input has arrived and is directly connected to a 1-bit adder alongside the 1-bit that was stored previously. The accumulation result is rewired according to the position of the leading-1. This step is essential to realign the approximated results with the multi-cycle neurons of the same layer. With this approximation, the results of the single-cycle neurons may become available earlier than the multi-cycled neurons of the same layer.

3.1.3 Overview of the overall Super-TinyML architecture. Figure 3 depicts the state-of-the-art and the proposed architectures of the sequential super-TinyML circuit. The controller logic utilizes a simple counter state machine to enable the different layers when the state value is within certain ranges ($0 \leq state \leq N$ for hidden layer and $N \leq state \leq N+P$ for output layer). The layers also use a reset signal generated in this logic for each inference. The state signal is also connected to the multiplexers between layers, in neurons and in argmax. In the state-of-the-art, the layers are connected using shifting registers instead. The proposed architecture assumes that there is only one input per cycle for the first N cycles, so only one ADC is active per cycle (e.g., the rest can be power-gated).

The results of each neuron are stored in registers dedicated for accumulation. When all the inputs are fed into the system, all of the neurons in the hidden layer will finish their calculations, and the state machine will produce an enable signal for the output layer to start the computations. In the output layer, the same architecture is repeated and the same logic is followed. When the results of the output layer are ready, the argmax circuit decides the winning class. As can be seen in Figure 3, the argmax is realized with a single comparator that compares the incoming values sequentially. There are two branches; the top branch has the values of the output layer

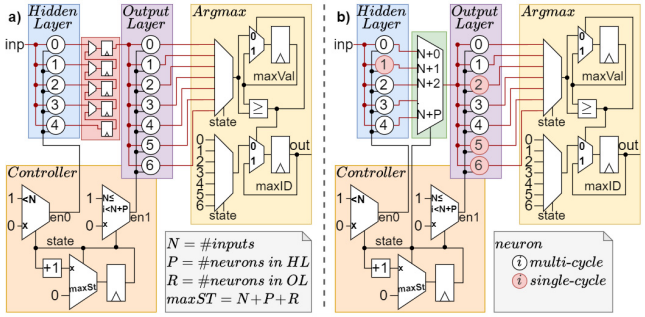

Figure 3: Overview of the sequential super-TinyML architectures: a) state-of-the-art [16], b) proposed.

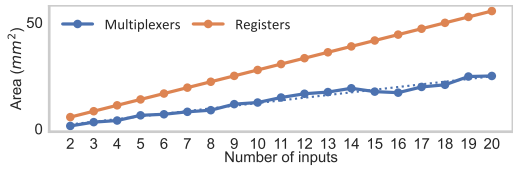

Figure 4: Area comparison of registers vs. multiplexers.

as an input and compares them to the last largest output, while the bottom branch stores the class with the highest value.

3.1.4 Architectural Choices. Conventional architecture of a sequential super-TinyML includes a controller logic for synchronizing the overall execution, i.e., dataflow between the layers and the argmax. To synchronize the data between the layers in conventional sequential MLP architectures, shifting registers are inserted between each layer, where the results of the preceding layer are stored and shifted. Additionally, in a sequential neuron, shifting registers are used to store the weights [16]. However, in the proposed architecture, multiplexers are used instead of the shifting registers to achieve the same goal with much reduced area and power. This is due to the large area overhead of the registers; the area of the shifting registers scales up linearly with the increasing number of registers required to store the output values from the preceding layer. Using a 2x1 multiplexer instead of 2 single-bit shifting registers already has less area (1 : 4 ratio), which increases by a smaller constant than the shifting registers. Figure 4 presents an area comparison between the shifting registers and multiplexers w.r.t. their number of inputs. We can observe that the shifting registers have larger area than the multiplexers for the same number of inputs. Moreover, as the number of inputs increases, the multiplexer area scales with a smaller slope than the register, leading to larger area gains. As a reference, for Arrhythmia dataset, which has 274 features and 1160 coefficients, replacing registers with muxes results in 4.4× less area. Additionally, to further minimize the area of multiplexers, the common denominator of weights for each neuron is calculated. The remainder of the weights are stored in the registers or multiplexers, and the common denominator is multiplied afterwards.

3.2 Extraction of Model Parameters

3.2.1 Quantization. The multiplier is the largest arithmetic unit within a neuron [1]. Thus, we minimize the multiplication area by using power-of-2 (pow2) quantization and replace the multiplier circuit by a barrel shifter. By leveraging pow2 values, i.e.,

$\pm w = s \times 2^{p_i}$, $p_i \in [0, 1, ..., n]$ where n is the weight bitwidth, multiplication is realized by simply shifting the input by the corresponding power of the weight. As it is described in Section 3.1, we realize the multiplication in the proposed sequential architecture by using only one barrel shifter per neuron. Further, only the power p and the sign s need to be stored in the neuron, thus increasing the saving both in storing elements, i.e., muxes; and arithmetic units, i.e., multipliers. In addition, the conventional ReLU activation function yields unbounded positive outputs, resulting in wider bit-widths at the neuron's output and subsequently larger inputs to the next layer. Thus, we adopt quantized ReLU (qReLU), a quantized-friendly variant of the ReLU, which truncates certain LSBs and applies saturation of the result to a maximum value, maintaining a manageable range without the need of re-quantization steps [10].

3.2.2 Redundant Feature Pruning. The sequential architecture flattens inputs in the time domain to reuse hardware with each input, improving hardware performance while increasing inference time. For multi-sensory applications, using MLPs with only a few neurons in the hidden layer, the flattened inputs dominate the time domain. Considering the target applications have a number of inputs ranging from 44 to 753, having 3-5 hidden neurons and a few output neurons is insignificant in terms of the time domain. Moreover, as the number of sensors increases, features become more correlated and redundant. After the quantization of the MLP, redundant feature pruning (RFP) is applied to discard the non-important features based on their statistical relevance. RFP is typically used in more complex ML systems such as convolutional neural networks [5]; however, to reduce the area overhead, the inference time, and consequently the energy consumption, we apply RFP in MLPs. The reduction of weights in the MLP is translated to a reduction in area and power since smaller multiplexers and adders are required. The inference time is shortened by a clock cycle with each pruned feature. After this process, the MLPs retain on average 81% of their original features, whereas 19% are removed.

As it is described in Algorithm 1, RFP uses the weights of the hidden layer in the MLP, the training dataset, and a given threshold to calculate N, the minimum number of inputs required to meet the threshold accuracy. With the weights and the training dataset, the algorithm calculates the relevance of each of the inputs. Although the distribution of inputs in the dataset indicates which inputs have more effect on the accumulation, some inputs have larger weights. Therefore, the products for each of the inputs are used to calculate relevance and redundancy. Algorithm 1 shows how the RFP is applied. For each input, the expected average is calculated from the training dataset, which are multiplied with the weights of each neuron in the hidden layer. Then, the average of the absolute products is calculated for each of the inputs, and then sorted in decreasing order. The weights in the hidden layer neurons and the training dataset is reordered by the order of the relevance of inputs. Afterwards, the MLP is continuously evaluated by increasing the number of features kept, until the accuracy meets the threshold. The threshold indicates the minimum desired accuracy, which is equal to the accuracy of the quantized MLP model. Even though the aforementioned algorithm is greedy, the runtime for the largest dataset with over 700 parameters it take less than one hour.

Algorithm 1: Pseudocode for Redundant Feature Pruning

```
Function prune_features MLP, x_train, threshold
    avg_prod=new float[MLP.HL.length][x_train.length];
    avg_x=average(x_train);
    for i=0; i<MLP.HL.length; i++ do
        avg_prod[i]=multiply(avg_x, HL[i].coeffs);
    [expected_avg, order]=sort(average(|avg_prod|));
    x_train=x_train[:, order];
    MLP=reorderWeights(MLP, order);
    for i=0; i<x_train.length; i++ do
        acc=MLP.eval(x_train[:, 0 : numofInputs]);
        if acc >= threshold then
            N=numofInputs;
            break;
```

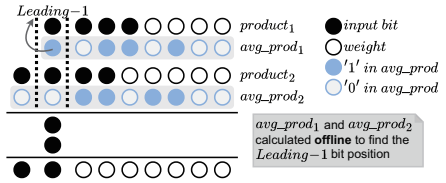

Figure 5: The proposed neuron approximation using the leading-1 of the average expected products.

3.2.3 Neuron Approximation. In order to further reduce power and area, neuron approximation is applied to the MLP, in accordance with the observation that neurons can withstand different degrees of approximation. While some neurons may need the entire input sequence to have a more accurate result, some are able to successfully estimate the result of the neuron using only a pair of bits. This pair of bits is selected based on the average expected product of each input for each neuron. Average expected product refers to the average of the absolute products from all the samples in the training dataset, the calculation of which is provided in Equation 1.

$$avg\_prod_{i,n} = \frac{\sum_{j=0}^{W}(E[x_i] * |w_{n,j}|)}{W} \quad (1)$$

In Equation 1, the absolute average product of each feature for each neuron is calculated, which is $avg\_prod_{i,n}$. $E[x_i], w_{n,j} and W$ refer to the average of feature x_i, weight j of neuron n, and the number of weights in neuron n. Our framework selects the two most important inputs as the inputs with the highest average expected products. The index of the respective leading-1 bit is calculated from the average expected products. Using the leading-1 of the two most important inputs, the result is approximated with single-bit addition. This approximation is depicted in Figure 5, where the 1-bit addition is positioned in the Leading-1 column of the binary inputs. Some neurons cause high accuracy degradation when approximated, while others have negligible impact.

We detect approximable neurons using a multi-objective genetic optimization algorithm to take advantage of its ability of effective solution space exploration. The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [13] is utilized to solve this multi-objective problem. Each candidate solution is represented as a set of boolean values to dictate whether the neuron of the corresponding value is approximated (value 1) or not (value 0). Prior to the genetic optimization, an initial population biased towards solutions with mostly non-approximated neurons are generated. Each initial solution consists of a set of boolean values, which include only 1 approximated neuron so that the solutions chosen as the parents

of following generations have higher accuracies. Each generation increases the number of approximated neurons until the solution with the highest number of approximable neurons and the highest accuracy is found. The multi-objective goal is to maximize the number of approximable neurons and the accuracy and maintain it above or equal to the desired accuracy. The maximum number of approximable neurons correspond to the area savings of the sequential super-TinyML circuit in an abstract manner, without the need for an extremely accurate hardware model. The desired accuracy refers to the tolerable minimum accuracy of the MLP. The genetic algorithm's result is a solution, a set of boolean values, used to generate the Verilog description of the super-TinyML design.

4 RESULTS

In this section, we evaluate the results of our sequential super-TinyML architecture. Section 4.1 describes the experimental setup. In Section 4.2.1, we compare our exact sequential design with the current state-of-the-art [14]. In this assessment, we use the same feature selection method also on the state-of-the-art for a more fair comparison. In Section 4.2.2, we further evaluate our sequential circuits by applying our neuron approximation. In Section 4.3, we give a comprehensive analysis regarding energy consumption.

4.1 Experimental Setup

The datasets in this work are obtained from the UCI ML repository [9]. In the case that non-sensor (categorical) features exists in the datasets, we performed pre-processing to remove them. The datasets considered in our work are SPECTF, Arrhythmia (Arr.), Gas Sensor (Gas S.), Epileptic Seizure (Epi.), Activity Recognition using Wearable Physiological Measurements (Act.), Parkinsons (Par.) and Human Activity Recognition (HAR). The MLPs are trained by following the strategy of the authors in [4, 16]. The open source Qkeras [8] framework is used for retraining the models for pow2 weights, while the open source PyGAD [11] was used for deploying the NSGA-II. Inputs and weights are quantized to 4-bit fixed-point and 8-bit power-of-2 respectively, as in previous state-of-the-art [4, 16] except HAR dataset where the weights are quantized to 14 bit. The quantization resolution for HAR dataset was chosen based on the highest accuracy results after the QAT, due to the fact that the 8-bit resolution did not reach our accuracy thresholds. For the synthesis and simulation of the circuits, Synopsys Design Compiler T-2022.03 and VCS T-2022.06 was used respectively. PrimeTime U-2022.12 is used for the power analysis. The circuits in our work are mapped to the open-source printed EGFET library [6]. The combinational circuits are all synthesized at 320ms except SPECTF, which is synthesized at 200ms. The proposed sequential circuits are synthesized at different clock periods; SPECTF at 80ms, HAR, Arr. and Gas S. at 100ms, while the rest at 120ms. The synthesis clocks are chosen based on the complexity of the MLP circuit and are in line with typical printed circuits performance [15].

4.2 Evaluation of the Sequential Super-TinyMLs

4.2.1 Sequential vs State-of-the-Art. Figure 6 depicts the comparison between the combinational state-of-the-art in [14], the sequential state-of-the-art in [16] and our multi-cycle sequential super-TinyML design, respectively, when QAT and RFP are applied. The

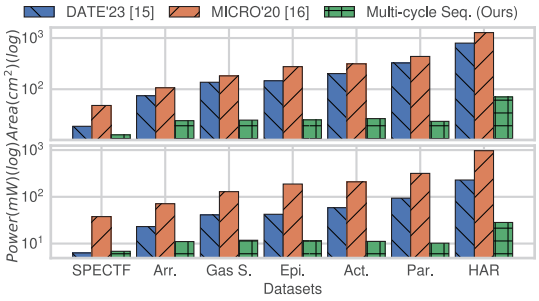

Figure 6: Comparison of State-of-the-Art Combinational [14], Sequential [16] and our Multi-cycle Sequential designs.

Table 1: Evaluation of Accuracy, Area and Power

Dataset	Accuracy	MICRO'20 [16] Area (cm^2)	MICRO'20 [16] Power (mW)	Our Multi-cycle Seq. Area Gain	Our Multi-cycle Seq. Power Gain
SPECTF	87.5	48.2	37.7	3.8×	5.5×
Arr.	61.8	106.7	71.1	4.4×	6.5×
Gas S.	90.7	182.1	128.9	7.3×	10.9×
Epi.	93.5	275.8	187.8	11.0×	16.5×
Act.	80.5	313.0	209.0	11.7×	18.7×
Par.	85.5	437.1	317.4	18.5×	31.1×
HAR	96.9	1276.2	969.2	18.1×	34.3×

results of the latter two designs are also provided with the accuracies of the models in Table 1. The datasets in the figure are ordered by the number of coefficients in each MLP model. In this work we focus on large multi-sensory inputs, that none of the previous works consider. To this end, we replicate the architectures used in the previous works on our MLP models. We compare our architecture with the combinational [14] that includes QAT; however, we also apply our RFP for a more fair comparison. We also compare our multi-cycle sequential design with the conventional sequential in [16], indicating the effect of our architectural choices. As shown in Figure 6, [16] compared to [14] is larger in area and consumes more power, 1.7× and 4.0× on average, respectively. Compared to [16], our multi-cycle sequential design is 10.7× smaller in area and consumes 17.6× less power. Additionally, the proposed sequential design has less area and power consumption compared to [14]. The proposed exact sequential architecture achieves on average 6.9× and 4.7×, area and power gains over the combinational [14]. The area and power gains of our design, compared to combinational [14], range from 1.5× to 13.8× and from 0.9× to 9.2× respectively. As depicted in Figure 6, the power of the SPECTF dataset is slightly increased. The additional components that a sequential design requires such as registers and muxes increase the required area and power, and create an overhead for smaller datasets when compared to the combinational design. In other words, the sequential design proves superior to its combinational counterpart, in terms of area and power, as number of inputs and subsequently the size of the MLP scale. Since the SPECTF dataset is the smallest of the datasets, the sequential architecture's results are closer to the state-of-the-art combinational design. Additionally, the registers consume more power in ratio to other logic gates than they occupy more area. The area in SPECTF is 1.5× smaller while the power consumption is 1.1× more in the proposed design, compared to [14].

4.2.2 Evaluation of Neuron Approximation. The results of the neuron approximation are depicted in Figure 7, highlighting the effectiveness of the proposed Neuron approximation technique using

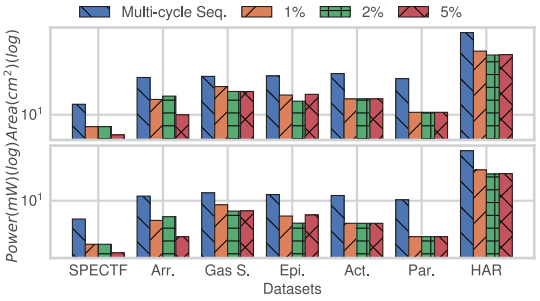

Figure 7: Impact of Neuron Approximation on the Hybrid design compared to the Multi-cycle Sequential design.

hybrid sequential architecture. First we apply only QAT and feature selection on the sequential architecture . Then, on top of these optimizations we further apply neuron approximation with 1%, 2%, and 5% accuracy drop. For 1%, 2%, and 5% the average area gains compared to multi-cycle sequential prior to Neuron Approximation are 1.7×, 1.8× and 1.9×, whereas the power gains are 1.7×, 1.7× and 1.8×, respectively. The area and power gains of all 3 accuracy constraints range from 1.3× to 2.4× and from 1.3× to 2.3×.

4.3 Energy discussion

The energy required for all architectures are represented in Figure 8. In comparison to [14], the sequential design described in [16] requires 363× more energy on average, ranging from 118× to 737×. This steep increase is due to the large number of registers that are used. The multi-cycle sequential architecture reduces the energy consumption to only 20× compared to [14], which ranges from 12× to 26×. Additionally, the proposed hybrid sequential design consumes 11.5× more energy compared to [14]. Our hybrid design, compared to the sequential design in [16], demonstrates 31.6× energy gains. The sequential architecture folds the area and subsequently reduce the power in the targeted multi-sensory datasets, but unfolds in the timing domain. That means that each neuron in the design needs multiple cycles to calculate its result and subsequently, the sequential super-TinyML circuit requires multiple cycles to compute the output class compared to its combinational counterpart where only one cycle is needed. It is important to reiterate that PE applications do not target on performance constraints. Due to this, the energy required for the designs increases with respect to the state-of-the-art combinational architectures. However, the primary issue in printed batteries is the the peak power they can deliver in the target applications, so even an increase in energy is tolerable [16].

5 CONCLUSION

Printed Electronics shows great potential for introducing computing and intelligence to application domains that have not yet undergone substantial amount of integration of computing, due to the unique characteristics such as conformality, flexibility, and non-toxicity. The aforementioned characteristics are intriguing for applications like wearables, implants and in-situ monitoring. However, the limited device count of PE impedes the integration, while the large size of registers has deterred the realization of sequential ML architectures in the past. Especially for machine learning classifiers, when the application uses multiple sensor inputs, the circuit gets more complex and subsequently the area and power consumption is greater. In this work, we present a sequential super-TinyML circuit for PE that utilize both single-cycle and multi-cycle neurons and successfully. Our evaluation shows that for MLPs with multiple input sensors, our area and power gains are on average 12.7× and 8.3× compared to the state-of-the-art combinational in [14].

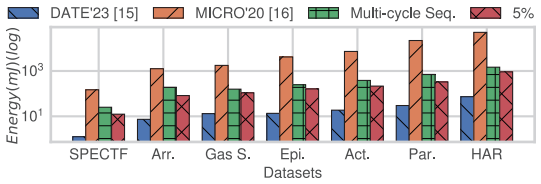

Figure 8: Evaluation of State-of-the-Art and our Multi-cycle Sequential designs in terms of energy.

ACKNOWLEDGMENT

This work is partially supported by the European Research Council (ERC), and co-funded by the H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" (H.F.R.I. Project Number: 17048).

REFERENCES

[1] Giorgos Armeniakos, Georgios Zervakis, Dimitrios Soudris, and Jörg Henkel. 2022. Hardware Approximate Techniques for Deep Neural Network Accelerators: A Survey. *ACM Comput. Surv.* 55, 4, Article 83 (nov 2022), 36 pages.
[2] Giorgos Armeniakos, Georgios Zervakis, Dimitrios Soudris, Mehdi B. Tahoori, and Jörg Henkel. 2022. Cross-Layer Approximation for Printed Machine Learning Circuits. In *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe* (Antwerp, Belgium) *(DATE'22)*. European Design and Automation Association, Leuven, BEL, 190–195.
[3] Giorgos Armeniakos, Georgios Zervakis, Dimitrios Soudris, Mehdi B. Tahoori, and Jörg Henkel. 2023. Co-Design of Approximate Multilayer Perceptron for Ultra-Resource Constrained Printed Circuits. *IEEE Trans. Comput.* 72, 9 (2023), 2717–2725.
[4] Giorgos Armeniakos, Georgios Zervakis, Dimitrios Soudris, Mehdi B. Tahoori, and Jörg Henkel. 2023. Model-to-Circuit Cross-Approximation For Printed Machine Learning Classifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 11 (2023), 3532–3544.
[5] Babajide O. Ayinde, Tamer Inanc, and Jacek M. Zurada. 2019. Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks* 118 (2019), 148–158.
[6] Nathaniel Bleier, Muhammad Husnain Mubarik, Farhan Rasheed, Jasmin Aghassi-Hagmann, Mehdi B Tahoori, and Rakesh Kumar. 2020. Printed Microprocessors. In *International Symposium on Computer Architecture*. 213–226.
[7] Joseph S Chang, Antonio F Facchetti, and Robert Reuss. 2017. A circuits and systems perspective of organic/printed electronics: review, challenges, and contemporary and emerging design approaches. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 7, 1 (2017), 7–26.
[8] Claudionor Coelho. 2019. QKeras. https://github.com/google/qkeras.
[9] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository.
[10] Abram L. Friesen and Pedro M. Domingos. 2017. Deep Learning as a Mixed Convex-Combinatorial Optimization Problem. *CoRR* abs/1710.11573 (2017). arXiv:1710.11573 http://arxiv.org/abs/1710.11573
[11] Ahmed Fawzy Gad. 2023. Pygad: An intuitive genetic algorithm python library. *Multimedia Tools and Applications* (2023), 1–14.
[12] Jörg Henkel, Hai Li, Anand Raghunathan, Mehdi B. Tahoori, Swagath Venkataramani, Xiaoxuan Yang, and Georgios Zervakis. 2022. Approximate Computing and the Efficient Machine Learning Expedition. In *Int. Conf. on Computer-Aided Design (ICCAD)*. 1–9.
[13] Deb Kalyanmoy, Pratap Amrit, Agarwal Sameer, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEETransactions on Evolutionary Computation* 6, 2 (2002), 182–197.
[14] Argyris Kokkinis, Georgios Zervakis, Kostas Siozios, Mehdi B. Tahoori, and Jörg Henkel. 2023. Hardware-Aware Automated Neural Minimization for Printed Multilayer Perceptrons. In *2023 Design, Automation and Test in Europe Conference and Exhibition (DATE)*. IEEE, Antwerp, BEL, 1–2.
[15] Cadilha Marques et al. 2017. Digital power and performance analysis of inkjet printed ring oscillators based on electrolyte-gated oxide electronics. *Applied Physics Letters* 111, 10 (2017), 102103.
[16] M. H. Mubarik, D. D. Weller, N. Bleier, M. Tomei, J. Aghassi-Hagmann, M. B. Tahoori, and R. Kumar. 2020. Printed Machine Learning Classifiers. In *Annu. Int. Symp. Microarchitecture (MICRO)*. 73–87.
[17] Feng Shao and Qing Wan. 2019. Recent progress on jet printing of oxide-based thin film transistors. *Journal of Physics D: Applied Physics* 52, 14 (02 2019), 143002.