Accepted for publication at the 30th IEEE European Test Symposium (ETS '25)

Automatic Test Pattern Generation for Printed Neuromorphic Circuits

Tara Gheshlaghi[1], Priyanjana Pal[1], Alexander Studt[1], Michael Hefenbrock[2], Michael Beigl[1], Mehdi B. Tahoori[1]
[1]*Karlsruhe Institute of Technology,* [2]*RevoAI GmbH,*
[1]{tara.gheshlaghi, priyanjana.pal, alexander.studt, michael.beigl, mehdi.tahoori}@kit.edu, [2]michael.hefenbrock@revoai.de

***Abstract*—Printed Electronics (PE) has emerged as a superior alternative to traditional silicon-based technologies for applications requiring mechanical, flexibility, point-of-use customization, low-cost fabrication, and energy efficiency. PE is particularly well-suited for low-power edge devices, such as wearable healthcare systems, environmental monitoring patches, and flexible diagnostic tools, where lightweight, adaptable, and efficient solutions are essential. Leveraging PE, printed analog neuromorphic circuits (pNCs) enable energy-efficient neural computations, making them ideal for classification tasks in target applications. However, the additive manufacturing processes inherent to PE increase susceptibility to defects, posing significant challenges for reliable operation and necessitating robust fault detection mechanisms. To address this, we propose a novel Automatic Test Pattern Generation (ATPG) framework tailored for pNCs, integrating fault abstraction and gradient-based test input generation. Fault modeling and abstraction reduces the fault space by clustering faults with similar transfer functions and removing untestable faults, thereby enhancing testing efficiency. In our experiments, the fault space was reduced by a factor of 2.28. The proposed framework refines test inputs through gradient-based optimization, maximizing output discrepancies between fault-free and faulty circuits. We evaluate the method on nine trained models across multiple datasets, achieving fault coverage exceeding 90% with fewer test vectors compared to random search.**

I. Introduction

In recent years, Printed Electronics (PE) has gained significant attention due to its unique benefits, including mechanical flexibility, affordability, customizability, and the ability to support on-demand fabrication. These attributes make PE particularly suitable for a wide range of applications, such as the Internet of Things (IoT), wearable devices, RFID tags, smart cards, smart labels and smart sensors [1]. Functional PE circuits, featuring organic and inorganic printed transistors, are fabricated through additive manufacturing techniques. These processes often utilize maskless, portable methods—sometimes referred to as 'fab-in-a-box' systems. This approach drastically lowers production costs and reduces manufacturing timelines. A key strength of PE lies in its ability to provide highly customized, application-specific solutions for both small and large production volumes, offering a cost-effective alternative to traditional lithography-based fabrication techniques.

To support essential analog signal processing tasks in target applications, such as classification on analog sensory inputs, pNCs serve as a natural fit. By directly processing analog input, these circuits eliminate the need for costly analog-to-digital converters (ADCs), simplifying computational processes and hardware footprint. Additionally, analog artificial neural networks (ANNs) implemented in pNCs are well-suited for PE, as they enable reduced device counts compared to digital counterparts, thus aligning with the constraints of PE,

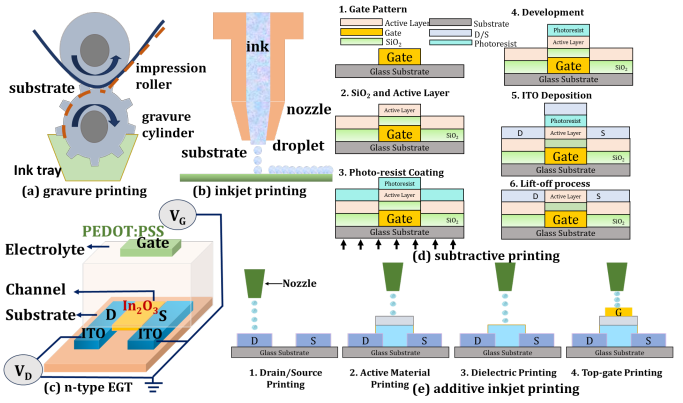


Fig. 1: Schematic of typical printing technologies: (a) gravure printing and (b) inkjet printing (c) a printed nEGT (d) Subtractive and (e) Additive printing process of PE.

which prioritize low-cost, additive manufacturing techniques and achieve lower integration density.

Despite their advantages, printed analog circuits face challenges with respect to testing. These include increased susceptibility to defects and variabilities compared to the mask-based fabrication, due to their low-cost additive manufacturing and reduced process control. Furthermore, problems such as ink smudging, delamination, and void formation can increase the challenges, reducing the overall reliability of printed circuits [2]. Unlike digital circuits, where standardized tests and ATPG methods are widely applicable [3, 4], analog testing requires customized approaches due to the specific nature of analog building blocks. In addition, bespoke architectures [5]–[7] for pNCs where model parameters are hardcoded into the circuit, and the inherent complexity of analog designs further complicate testing challenges as traditional digital fault models and test generation methods are no longer applicable to pNCs.

To address these challenges, it is crucial to focus on enhancing the manufacturing tests while ensuring the reliable performance of PE during in-field operation. As a solution, ATPG, widely used in digital circuit testing, should be developed to address fault detection of pNCs in PEs but remains widely unexplored in these target application domains. Recent advances in machine learning (ML) have enabled new possibilities to improve the testing of analog and mixed signal circuits, with ML techniques being used in Built-in Self-Test (BIST) configurations for analog devices. These developments motivate the exploration of optimization-based algorithms. For instance, [8] introduced a gradient-based approach for fault detection in analog integrated circuits using neural twins, which replicate circuit behavior through neural networks. Their method maximized output discrepancies to differentiate

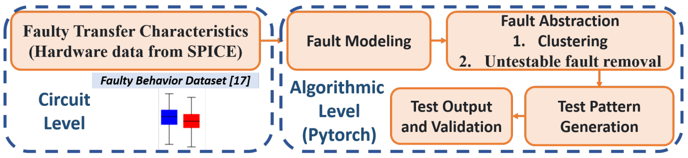

Faulty Transfer Characteristics (Hardware data from SPICE)
Faulty Behavior Dataset [17]
Circuit Level
Fault Modeling
Fault Abstraction
1. Clustering
2. Untestable fault removal
Algorithmic Level (Pytorch)
Test Output and Validation
Test Pattern Generation

Fig. 2: Proposed flow for automatic test pattern generation (ATPG) on an on-demand bespoke pNC in PE given a specification of a desired functionality.

between faulty and fault-free systems, focusing on analog IPs. Similarly, [9] examined optimization strategies employing Hopfield neural networks, Simulated Annealing, and Genetic Algorithms to enhance differentiation between nominal and faulty circuit responses. Inspired by these methodologies, our approach adapts gradient-based optimization to physical pNCs, addressing unique fault coverage challenges without relying on neural twins.

Our work addresses a critical gap in testing pNCs, which implement analog ANNs. While prior studies have explored traditional ATPG techniques and ML-based methods for testing analog circuits [8], there is no prior work specifically targeting fault detection in analog ANNs. Existing approaches face significant challenges when applied to these systems due to the complexities, variability, and unique characteristics of PE. To address this, we employ a gradient-based optimization algorithm to generate efficient test patterns for pNCs, ensuring comprehensive fault detection while minimizing redundancy—i.e., reducing overlapping test patterns—and computational overhead. This work makes two key contributions:

1) **Fault Analysis and Abstraction:** We analyze fault types in pNCs, starting from technology-level defects in the circuit, approximating fault equivalents all the way to training algorithms and identifying untestable faults to streamline abstraction and improve fault coverage.
2) **Gradient-Based Optimization:** We propose a gradient-based approach to generate optimal test patterns, achieving up to 90% fault coverage. This method is validated on nine trained pNCs, demonstrating its effectiveness in comprehensive fault detection.

The rest of this paper is structured as follows: Sec. II introduces PE, pNC, fault models in analog circuits, testing techniques, and other preliminaries. Sec. III describe the methodology of ATPG. In Sec. IV, the proposed approach is validated with extensive simulations. Finally, Sec. V summarizes this work.

II. Preliminaries

A. *Printed Electronics (PE)*

Printed Electronics (PE) is a manufacturing technology that leverages cost-effective printing methods such as gravure (as shown in (Fig. 1a) and inkjet printing (as shown in (Fig. 1b) to fabricate flexible circuits using advanced inks and materials, such as organic and oxide-based semiconductors. Non-contact techniques like inkjet printing enable precise, additive manufacturing, which is ideal for small-batch and bespoke applications, while contact methods like gravure printing are

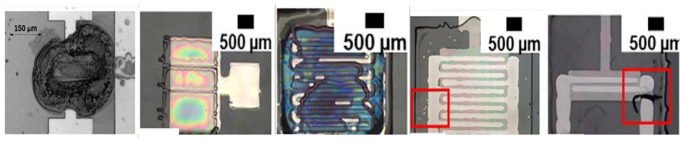


Fig. 3: (a) Transistor with exploded electrolyte (b) inkjet-nozzle clog (c) inhomogeneous layer formed (d) satellite drops of ink (e) short circuit between S-D (sourced from [12, 13]).

optimized for high-volume production. The inkjet-printed fabrication process of n-type electrolyte-gated transistors (nEGTs) (Fig. 1d) involves sequential printing of electrodes, semiconductors, gate dielectrics and gate electrodes (as shown in (Fig. 1e). This process offers simplified manufacturing process steps with reduced material waste, suitable for energy efficient and scalable electronics compared to subtractive manufacturing process (as shown in (Fig. 1c). Unlike organic semiconductors, which suffer from low field-effect mobility and high operating voltages, inorganic oxide semiconductors in nEGTs achieve superior performance, with high electron mobility and sub-1V operation due to their high gate capacitance. These attributes make nEGTs highly suitable for low-power PE applications powered by energy harvesters or small batteries [10], despite the current lack of reliable P-type EGT counterparts [10, 11].

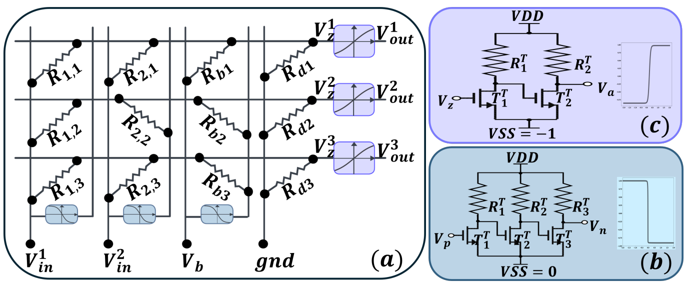

$R_{1,1}$ $R_{2,1}$ R_{b1} R_{d1}
$R_{1,2}$ $R_{2,2}$ R_{b2} R_{d2}
$R_{1,3}$ $R_{2,3}$ R_{b3} R_{d3}
V_z^1 V_{out}^1
V_z^2 V_{out}^2
V_z^3 V_{out}^3
V_{in}^1 V_{in}^2 V_b gnd
(a)
VDD
R_1^T R_2^T
V_z T_1^T T_2^T V_a
VSS = −1
(c)
VDD
R_1^T R_2^T R_3^T
V_p T_1^T T_2^T T_3^T V_n
VSS = 0
(b)

Fig. 4: Schematic of printed neuromorphic circuits. (a) example of a 3-input and 3-output printed layer based on crossbar array; (b) p-negative weight; (c) p-tanh.

B. *Printed Analog Neuromorphic Circuits (pNC)*

Neuromorphic computing has emerged as a transformative approach for solving computational tasks efficiently, leveraging artificial intelligence (AI) advancements. pNCs, in particular, provide a highly cost-effective solution for real-time sensor data processing. Compared to silicon-based hardware, pNCs offers low power [14], high customizability, and suitability for edge-computing tasks. These circuits execute basic neural operations, including weighted-sum and nonlinear activation, to enable efficient on-demand processing for PE applications.

a) Resistor Crossbars: The resistor crossbar, shown in Fig. 4a, performs weighted-sum operations based on Ohm's Law and Kirchhoff's Current Law. The output voltage V_z can be expressed as a linear combination of input voltages V_i, where the weights are determined by the conductance values:

$$V_\mathrm{z} = \frac{g_1}{G}V_1 + \frac{g_2}{G}V_2 + \frac{g_3}{G}V_3 + \frac{g_\mathrm{b}}{G}V_\mathrm{b}, \tag{1}$$

where g_i denotes the conductance of resistor R_i, and G represents the total conductance, defined as $\sum_j g_j + g_\mathrm{b} + g_\mathrm{d}$. By carefully adjusting the conductance ratios, this design achieves programmable weights, effectively enabling the weighted-sum operation akin to training in ANNs. Since the resistor crossbar can only generate positive weights, additional inverter-based circuits are incorporated to enable negative weight representation [14]. As illustrated in Fig. 4b, it achieves weight inversion by flipping the polarity of input voltages V_{i}.

In pNCs, each crossbar array's learnable parameter is denoted as $\boldsymbol{\Theta} \in \mathbb{R}^{(M+2)\times N}$, where M and N represent the number of inputs and outputs. The magnitude of each element in $\boldsymbol{\Theta}$ indicates the conductance, while its sign determines if a negative weight circuit is required. The crossbar output voltages $\boldsymbol{V}_{\mathrm{z}} = [V_{\mathrm{z}}^1, \ldots, V_{\mathrm{z}}^N]$ are given by:

$$\boldsymbol{V}_{\mathrm{z}} = \boldsymbol{V}_{\mathrm{in}} \cdot \left(\boldsymbol{W} \odot \mathbb{1}_{\{\boldsymbol{\Theta}\geq \mathbf{0}\}}\right) + \mathrm{neg}(\boldsymbol{V}_{\mathrm{in}}) \cdot \left(\boldsymbol{W} \odot \mathbb{1}_{\{\boldsymbol{\Theta}< \mathbf{0}\}}\right),$$

where $\boldsymbol{V}_{\mathrm{in}} = [V_{\mathrm{in}}^1, \ldots, V_{\mathrm{in}}^M, 1\mathrm{V}, 0\mathrm{V}]$ includes the input voltages extended by V_{b} and GND, and $\odot$ is the element-wise product. The weight matrix $\boldsymbol{W}$ is defined as:

$$\boldsymbol{W} = |\boldsymbol{\Theta}| \cdot \mathrm{diag}(|\boldsymbol{\Theta}|^{\top} \cdot \mathbf{1}_{M+2})^{-1}, \tag{2}$$

where $\mathbf{1}_{M+2}$ is a vector of ones, $|\cdot|$ applies an element-wise absolute operation, and $\mathrm{diag}(\cdot)$ generates a diagonal matrix. The output voltages $\boldsymbol{V}_{\mathrm{z}}$ are then passed through the *p-tanh* activation function (AF):

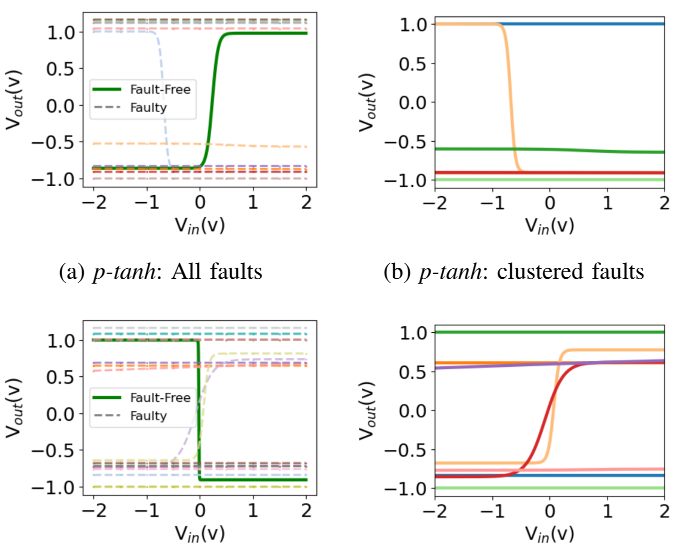
$$\boldsymbol{V}_{\mathrm{out}} = \mathrm{ptanh}(\boldsymbol{V}_{\mathrm{z}}),$$

where $\boldsymbol{V}_{\mathrm{out}} = [V_{\mathrm{out}}^1, \ldots, V_{\mathrm{out}}^N]$ represents the printed neuron outputs. By stacking these weighted-sum and AF operations, deeper pNCs can be designed efficiently.

b) Printed activation function circuit: After processing through the crossbar, the signals are passed through printed AF circuits, as shown in Fig. 4c, thus introducing the nonlinearity to the system. For example, the printed **tanh** (*p-tanh*) AF maps the input to the range [-1, 1], improving balanced gradient propagation and enabling efficient learning [6]. While these printed AFs are bespoke, learnable [15], provide nonlinearity, and maintain ANN performance, they are more susceptible to defects in the PE manufacturing process and can cause unstable behavior in pNCs [16].

C. Defects in Additive Printing and Fault Models in Printed Circuits

Faults in printed circuits can arise from defective components, signal line breaks, short circuits, and delays (as shown in Fig. 3) affecting the circuit functionality. Permanent faults, which persist and are detectable during testing, include *catastrophic faults* (e.g., open and short circuits) and *parametric faults* (caused by variations in process parameters) [12, 17]. Catastrophic faults are further classified into *stuck-open* faults, where component terminals lose contact, simulated by adding a large series resistor ($R_s = 100\,\mathrm{M\Omega}$), and *stuck-short* faults, where terminals are short-circuited, modeled by a small parallel resistor ($R_p = 1\,\Omega$) [18]. Parametric faults, on the other hand, affect component values such as resistors and capacitors, and are classified as *global* or *local*. Global faults arise from manufacturing imperfections that affect all components, whereas local faults result from specific defects, such as particles that affect the transistor channel length. These fault models help identify and simulate defects in printed circuits effectively.



(a) *p-tanh*: All faults (b) *p-tanh*: clustered faults

(c) *p-negative*: All faults (d) *p-negative*: clustered faults

Fig. 5: Fault-free and faulty transfer functions for the *p-tanh* and *p-negative* circuits. In the left subplots, all fault transfer functions are shown, including one that is significantly distant from the others (as measured by average Euclidean distance), causing visualization clutter. For clarity, this outlier transfer function and its corresponding cluster are omitted from the representative subplots (though they remain accounted for in the algorithm). This abstraction effectively highlights fault behavior while minimizing duplicate test representations.

III. Methodology

To develop an efficient test pattern generation framework for pNCs, we propose a gradient-based optimization algorithm aimed at generating test inputs. The primary goal is to maximize the output discrepancy between fault-free and faulty pNCs. Standard digital fault models, such as stuck-at or transition faults, are not applicable for analog pNCs due to two main reasons: (1) the building blocks are different and (2) the underlying defect mechanisms originating from additive manufacturing with different materials and processes are fundamentally different.

As shown in Fig. 2, the methodology begins with defect injection at the circuit level using SPICE simulations. This step accurately models technology-specific failure mechanisms, resulting in faulty characteristics that reflect realistic defect behaviors. These characteristics are then translated into algorithm-level fault models, enabling test generation using gradient-based optimization.

Given the extensive number of potential defects introduced by additive manufacturing, exhaustive testing is impractical. To make test generation feasible, we incorporate a *fault abstraction* process consisting of two main components: **fault clustering** and **untestable fault removal**. In analog pNCs, learning-based

TABLE I: Comparison of Gradient-based Algorithm and Random Search for Test Input Generation. **#Faults:** Total faults before and after clustering, and after clustering + untestable removal. **Fault Cov.:** Fault Coverage (%). **#Vec:** Number of Test Vectors. Runtime in minutes (Gradient-based Algorithm)

Dataset	**pNC #Devices**		**#Faults**			**Gradient-based Algorithm**			**Random Pattern**	
	#Transistor	**#Resistor**	**All**	**+Clustering**	**+Untestable Removal**	**Fault Cov.**	**#Vec**	**Runtime (m)**	**Fault Cov.**	**#Vec**
AcuteInflam	25	42	362	220	123 (2.94×)	93.50%	115	211	47.96%	59
BreastCancWisc	39	81	434	265	128 (3.39×)	99.22%	127	139	49.22%	63
EnergyY2	40	72	432	267	214 (2.02×)	97.66%	209	177	71.96%	154
Iris	28	57	336	207	154 (2.18×)	97.40%	150	111	67.53%	104
Pendigits	71	173	778	506	448 (1.74×)	98.21%	440	626	78.79%	32
Seeds	34	73	408	252	201 (2.03×)	96.52%	194	104	72.64%	146
TicTacToe	21	62	434	265	153 (2.84×)	86.27%	132	86	60.13%	92
VertCol2	33	65	362	220	180 (2.01×)	69.44%	125	1094	40.00%	72
VertCol3	26	56	384	237	124 (3.1×)	99.19%	123	47	69.35%	86
Average	**35.2**	**75.7**	**436.7**	**271.0**	**191.67** (2.28×)	**93.05%**	**179.4**	**288**	**61.95%**	**89.83**

TABLE II: Fault Coverage Distribution Across Different Fault Locations (Cov.: Covered, Uncov.: Uncovered faults.)

Dataset	**p-tanh (in %)**		**p-negative (in %)**		**Crossbar (in %)**	
	Cov.	**Uncov.**	**Cov.**	**Uncov.**	**Cov.**	**Uncov.**
AcuteInflam	28	0	39	5	26	2
BreastCancWisc	23	0	27	1	49	0
Cardiotoco3	20	0	49	2	29	0
Iris	27	0	32	3	38	0
Pendigits	20	1	34	0	44	1
Seeds	21	0	37	3	38	0
TicTacToe	20	3	23	0	43	11
VertCol2	19	0	27	18	23	13
VertCol3	28	0	35	1	35	0

methods are used to identify faults with similar behaviors through clustering, effectively reducing redundancy.

A. Fault Modeling

In this work, we address catastrophic faults in three components of pNCs: crossbar conductances, AFs (*p-tanh*), and inverters (*p-negative*). Given the low probability of multiple faults and to ensure feasible test generation complexity, we assume a single-fault condition per analysis step.

In the crossbar, each element represents a surrogate conductance. Faults in this component are modeled using a ternary fault mask, M_g, such that $\tilde{g} = M_g \odot g$ and $M_g \in \{1, 0, \infty\}$. Here, $M_g = 1$ represents the fault-free state, 0 corresponds to an open circuit (zero conductance), and ∞ models a short circuit (infinite conductance). Fault injection is applied to specific layers and connections.

For the *p-tanh* and *p-negative* circuits, faults occur in resistors (open or short circuits) and transistors. Transistor faults are further categorized into four subtypes: gate-drain short, gate-source short, drain-source short, and gate open. To simulate these faults, we use SPICE simulations to generate faulty input-output pairs $(V_{\text{in}}, V_{\text{out}})$, which are stored in the Faulty Behavior Dataset (FBDataset). The behavior of the *p-tanh* and *p-negative* circuits is described using the following transfer functions:

$$\text{ptanh}(V) = V_{\text{a}} = \eta_1^{\text{T}} + \eta_2^{\text{T}} \cdot \tanh\left((V - \eta_3^{\text{T}}) \cdot \eta_4^{\text{T}}\right), \quad (3)$$

$$\text{pneg}(V_{\text{z}}) = -\left(\eta_1^{\text{N}} + \eta_2^{\text{N}} \cdot \tanh\left((V_{\text{z}} - \eta_3^{\text{N}}) \cdot \eta_4^{\text{N}}\right)\right), \quad (4)$$

where $\boldsymbol{\eta}^{\text{T}} = [\eta_1^{\text{T}}, \eta_2^{\text{T}}, \eta_3^{\text{T}}, \eta_4^{\text{T}}]$ and $\boldsymbol{\eta}^{\text{N}} = [\eta_1^{\text{N}}, \eta_2^{\text{N}}, \eta_3^{\text{N}}, \eta_4^{\text{N}}]$ are auxiliary fitting parameters. These parameters are obtained using curve fitting techniques (nonlinear least squares) to minimize the error between SPICE-simulated data and the predicted transfer function outputs for each faulty circuit, considering only one fault in the circuit at a time.

Fault injection for the *p-tanh* circuit is performed at the activation stage of individual neurons—since faults at this stage directly alter the neuron's output—whereas for the *p-negative* circuit, faults are injected at the connection level, reflecting their localized effect within the weight inversion network. The resulting transfer functions, as shown in Figures 5a and 5c, enable efficient fault analysis and clustering based on the similarity of their behaviors.

B. Fault Abstraction

Testing analog pNCs with an exhaustive fault set is computationally expensive and increases the number of test patterns. To address this, we propose a fault abstraction process consisting of two key steps: (1) approximating fault equivalence through clustering and (2) removing untestable faults. This process significantly reduces the fault set while maintaining high fault coverage. Since the elements of pNCs are analog, existing fault equivalency and fault collapsing methods developed for digital circuits cannot be directly applied. Consequently, learning-based clustering groups similar faults into a representative fault.

1) Approximating Fault Equivalence: Faults exhibiting similar transfer functions are approximated as equivalent using a clustering-based approach. The transfer functions corresponding to the faults are grouped by similarity, ensuring that faults leading to equivalent circuit behavior are clustered together.

Let $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ represent the fault set and $T(f)$ the transfer function under fault f. Faults are grouped into K clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ such that:

$$\mathcal{C}_k = \{f_n \mid d(T(f_n), T(c_k)) \leq \epsilon, \, \forall n = 1, \ldots, N\}, \quad (5)$$

where $d(T(f_n), T(c_k))$ is the Euclidean distance between the transfer function under fault f_n and the cluster centroid c_k, and

$\epsilon \in \mathbb{R}^+$ is a predefined similarity threshold. Each cluster is represented by the transfer function closest to the cluster centroid c_k, ensuring that test inputs generated for c_k are applicable to other faults within the cluster $\mathcal{C}_k$. The representative transfer functions after clustering are shown in Fig. 5b and Fig. 5d, corresponding respectively to the original fault distributions in Fig. 5a and Fig. 5c.

2) Removing Untestable Faults: Faults that produce identical outputs for faulty and fault-free pNCs under any input are classified as untestable and removed. For each fault f_n, the average difference between the fault-free output logits $\mathbf{y}_{\text{fault-free}}$ and the faulty output logits $\mathbf{y}_{\text{faulty}}$ is computed across a large random input set $\mathcal{X}$ as:

$$\Delta(f_n) = \max_{\mathbf{x}\in\mathcal{X}} \|\mathbf{y}_{\text{faulty}}(\mathbf{x}, f_n) - \mathbf{y}_{\text{fault-free}}(\mathbf{x})\|_2^2. \tag{6}$$

If $\Delta(f_n) = 0$, the fault f_n is deemed untestable and excluded from further analysis. This two-step abstraction process reduces computational complexity while preserving high fault coverage.

Algorithm 1: Proposed ATPG Framework for pNCs

```
Input: Output logits of Pre-trained PNC model $y$, Fault list $\mathcal{F}$,
       Learning rate $\eta$, Max epochs $E$, Random input set $\mathcal{X}_{\text{init}}$
Output: Optimized test patterns $\mathcal{T}$, Covered faults $\mathcal{C}$
1  Step 1: Fault Abstraction (Preprocessing)
2  Reduced fault set: $\mathcal{F}_{\text{abstracted}}$;
3  Step 2: Gradient-Based Test Pattern Generation
4  for each fault $f_n \in \mathcal{F}_{abstracted}$ do
5      Generate 100 random initial inputs $\mathcal{X}_{\text{init}} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{100}\}$;
6      Select the best input $\mathbf{x} \in \mathcal{X}_{\text{init}}$ that maximizes the initial
         discrepancy;
7      Set optimizer $\mathcal{O} \leftarrow \text{Adam}(\mathbf{x}, \eta)$;
8      for epoch $e = 1$ to $E$ do
9          Compute fault-free softmax output
             $p_{\text{fault-free}}(\mathbf{x}) = \text{softmax}(y_{\text{fault-free}}(\mathbf{x}))$;
10         Compute faulty softmax output
             $p_{\text{faulty}}(\mathbf{x}, f_n) = \text{softmax}(y_{\text{faulty}}(\mathbf{x}, f_n))$;
11         Compute the KL divergence as the optimization objective:
             $\mathcal{L}_{\text{KL}}(\mathbf{x}, f_n) = -\sum_j p_{\text{fault-free},j}(\mathbf{x}) \log \frac{p_{\text{fault-free},j}(\mathbf{x})}{p_{\text{faulty},j}(\mathbf{x}, f_n)}$.
12         Update test input $\mathbf{x}$ using backpropagation:
             $\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla_{\mathbf{x}} \mathcal{L}_{\text{KL}}$.
13         Project $\mathbf{x}$ back into the range $[0, 1]$ element-wise:
             $\mathbf{x} \leftarrow \text{clip}(\mathbf{x}, 0, 1)$;
14         if fault is covered
             ($argmax(p_{fault\text{-}free}(\mathbf{x})) \neq argmax(p_{faulty}(\mathbf{x}, f_n))$) then
15             Save $\mathbf{x}$ as optimized test pattern;
16             Add $f_n$ to covered faults $\mathcal{C}$;
17             Break;
18 return Optimized test patterns $\mathcal{T}$, Covered faults $\mathcal{C}$;
```

C. Generating Test Patterns Through Gradient Optimization

To expose faults in pNCs, we optimize input patterns that maximize the output discrepancy between fault-free and faulty models. Starting from 100 random initial inputs $\mathbf{x} \in [0, 1]$, the best input is selected for a given fault f_i and iteratively refined using a gradient-based approach. Once optimized for f_i, the process is repeated for the next selected fault f_j.

The optimization objective is defined as maximizing the Kullback-Leibler (KL) divergence between the softmax output distributions of the fault-free and faulty pNC models. For a given input $\mathbf{x}$, the KL divergence is expressed as:

$$\mathcal{L}_{\text{KL}}(\mathbf{x}, f_n) = -\sum_j p_{\text{fault-free},j}(\mathbf{x}) \log \frac{p_{\text{fault-free},j}(\mathbf{x})}{p_{\text{faulty},j}(\mathbf{x}, f_n)}, \tag{7}$$

where $p_{\text{fault-free},j}(\mathbf{x})$ and $p_{\text{faulty},j}(\mathbf{x}, f_n)$ represent the softmax probabilities of the fault-free and faulty outputs for class j, respectively. While the goal is to maximize the KL divergence to amplify output discrepancies, gradient-based optimizers inherently minimize objectives. To resolve this, we negate the KL divergence, allowing gradient descent to effectively maximize the discrepancy. During each iteration, gradients of $\mathcal{L}_{\text{KL}}$ with respect to $\mathbf{x}$ are computed via backpropagation, and the input pattern is updated. To ensure physical feasibility, the test pattern is projected back into the range $[0, 1]$ element-wise, $\mathbf{x} \leftarrow \text{clip}(\mathbf{x}, 0, 1)$, respecting the technology constraints that limit voltage values to a feasible range.

The optimization process continues until convergence, producing a final test pattern $\mathbf{x}^*$ for each fault f_n. Since the task involves classification, the objective is to generate test inputs that result in different class predictions for fault-free and faulty pNCs. While the *argmax* function determines class predictions, its non-differentiability prevents direct use in gradient-based optimization. To address this, the KL divergence serves as a smooth and differentiable surrogate, amplifying dissimilarities between the softmax distributions of fault-free and faulty outputs. By leveraging the KL divergence, the proposed method efficiently generates test patterns that expose faults, achieving high fault coverage while ensuring physical feasibility (technology constraints for input voltage range). The overall method is summarized in pseudocode, as shown in Algorithm 1.

IV. Evaluation

A. Experiment Setup

1) Circuit Setup: Fault injection in *p-tanh* and *p-negative* circuits was performed using SPICE simulations with the nEGT P-PDK [19] in Cadence Virtuoso[1]. This modeling resulted in 12 fault subtypes for *p-tanh* and 18 for *p-negative* circuits.

2) Algorithm Setup: For fault modeling, curve fitting was performed using `scipy` [20]. The fault space was reduced with the Elbow Method [21], clustering faults into 7 groups for *p-tanh* and 9 for *p-negative*. The gradient-based ATPG algorithm was evaluated on 9 pNCs, trained with datasets from the UCR Time Series Archive [22], using an #input-3-#classes architecture. Fault scenarios followed Section III. Test input optimization employed the Adam optimizer [23] with a 0.01 learning rate and early stopping after 100 epochs without fault coverage improvement. The baseline *Random Pattern* approach sampled over the feasible input range to evaluate fault coverage. Faults were considered *covered* if the fault-free and faulty models produced different class predictions. Both approaches were implemented in `PyTorch` [24][2].

[1]https://www.cadence.com/en_US/home.html

[2]Code is available at https://github.com/KIT-Neuromorphic-Computing/ATPG-PNC.

B. Results

Tab. I presents a comparative evaluation of the proposed Gradient-Based Algorithm and the baseline Random Pattern generation method across 9 datasets for different pNCs. The second column reports the number of transistors and resistors, excluding inactive components such as *p-tanh* with all zero-connected θ values, *p-negative* with non-negative θ values, and resistors with zero surrogate θ values. For simplicity, our fault model assumes an inverter for every weight—even when unnecessary for non-negative weights—resulting in fault counts that exceed the number of devices. The fault counts are shown before and after clustering, and after removing untestable faults, along with reduction ratios. The table highlights fault coverage (%), test vector count, and runtime (minutes). Tab. II provides the fault coverage distribution for *p-tanh*, *p-negative*, and *Crossbar* components.

C. Discussion

The proposed method significantly improves fault coverage and testing efficiency for analog pNCs. The Gradient-Based Algorithm achieved over 90% fault coverage across datasets, outperforming Random Pattern testing. For instance, fault coverage on the *BreastCancWisc* dataset reached 99.2%, compared to 49.2% with Random Patterns, albeit with a longer runtime (139 minutes). While fault abstraction reduced the fault space significantly (e.g., by 3.39 times in *BreastCancWisc*) datasets with complex fault behaviors, such as *TicTacToe* and *VertCol2*, remain challenging. These datasets occasionally lead the optimization to local optima, requiring additional iterations to improve coverage. Potential solutions include integrating multi-objective optimization methods, which can balance fault coverage and the required number of test patterns to also reduce runtime, as well as exploring hybrid algorithms that combine gradient-based and evolutionary approaches to more effectively navigate the fault space and provide better initialization for input patterns. These strategies will be explored in future work. Tab. II shows that the *p-tanh* circuit achieved the highest coverage, while *p-negative* and *Crossbar* had more uncovered faults, particularly in *TicTacToe* and *VertCol2*.

In summary, the method combines fault abstraction and gradient-based optimization to achieve high fault coverage with reduced redundancy.

V. Conclusion

This paper presents a framework for testing analog pNCs by integrating fault abstraction with a gradient-based ATPG method. Fault abstraction, achieved via clustering and the removal of untestable faults, effectively reduced the fault space while maintaining robust fault coverage. The gradient-based optimization demonstrated superior fault coverage and test vector reduction compared to random pattern generation but still faced challenges such as local optima and increased runtime. Future work will address these limitations by exploring evolutionary algorithms for test input generation and investigating fault localization to refine test pattern generation, enabling both fault detection and precise fault isolation. These multi-objective optimization approaches aim to improve fault coverage and enhance scalability for larger and more complex pNCs.

Acknowledgment

This work has been supported by the the European Research Council (ERC).

References

[1] M. A. Leenen *et al.*, "Printable electronics: Flexibility for the future," *physica status solidi (a)*, vol. 206, no. 4, pp. 588–597, 2009.
[2] M. H. Behfar *et al.*, "Failure mechanisms in flip-chip bonding on stretchable printed electronics," *Advanced Engineering Materials*, vol. 23, no. 12, p. 2100264.
[3] A. C. and A. D., "Fault modeling and testing of spiking neural network chips," *IEEE Transactions on Neural Networks*, vol. 29, no. 5, pp. 456–468, 2020.
[4] A. E. and A. F., "Generalized markov reliability models for fault-tolerant neural networks," *Reliability Engineering & System Safety*, vol. 45, no. 2, pp. 234–245, 2018.
[5] H. Zhao *et al.*, "Aging-Aware Training for Printed Neuromorphic Circuits," in *IEEE/ACM ICCAD '22*, 2022.
[6] H. Zhao, B. Sapui, M. Hefenbrock, Z. Yang, M. Beigl, and M. B. Tahoori, "Highly-bespoke robust printed neuromorphic circuits," *2023 DATE*, pp. 1–6, 2023.
[7] G. Armeniakos *et al.*, "Cross-layer approximation for printed machine learning circuits," in *2022 DATE*, 2022, pp. 190–195.
[8] J. Talukdar *et al.*, "Automatic structural test generation for analog circuits using neural twins," in *2022 IEEE ITC*. IEEE, 2022, pp. 145–154.
[9] J. L. Bernier *et al.*, "Test pattern generation for analog circuits using neural networks and evolutive algorithms," in *International Workshop on Artificial Neural Networks*. Springer, 1995, pp. 838–844.
[10] S. K. Garlapati *et al.*, "Ink-Jet Printed CMOS Electronics from Oxide Semiconductors," *Small*, vol. 11, no. 29, pp. 3591–3596, 2015.
[11] G. Cadilha Marques *et al.*, "Progress Report on "From Printed Electrolyte-Gated Metal-Oxide Devices to Circuits"," *Advanced Materials*, vol. 31, no. 26, p. 1806483, 2019.
[12] A. T. Erozan *et al.*, "Defect detection in transparent printed electronics using learning-based optical inspection," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 8, pp. 1505–1517, 2021.
[13] E. Sowade *et al.*, "All-inkjet-printed thin-film transistors: manufacturing process reliability by root cause analysis," *Scientific reports*, vol. 6, no. 1, p. 33490, 2016.
[14] H. Zhao *et al.*, "Power-Aware Training for Energy-Efficient Printed Neuromorphic Circuits," in *42nd IEEE/ACM ICCAD*, 2023.
[15] P. Pal *et al.*, "Neural architecture search for highly bespoke robust printed neuromorphic circuits," ACM Digital Library, Tech. Rep., 2024.
[16] P. Pal, F. Afentaki, H. Zhao, G. Saglam, M. Hefenbrock, G. Zervakis, M. Beigl, and M. B. Tahoori, "Fault sensitivity analysis of printed bespoke multilayer perceptron classifiers," in *2024 IEEE ETS*. IEEE, 2024, pp. 1–6.
[17] L. Milor *et al.*, "Detection of catastrophic faults in analog integrated circuits," *IEEE TCAD of Integrated Circuits and Systems*, vol. 8, no. 2, pp. 114–130, 1989.
[18] H. Ling *et al.*, "Electrolyte-gated transistors for synaptic electronics, neuromorphic computing, and adaptable biointerfacing," *Applied Physics Reviews*, vol. 7, no. 1, p. 011307, 2020.
[19] F. Rasheed *et al.*, "Predictive modeling and design automation of inorganic printed electronics," in *2019 DATE*, 2019, pp. 30–35.
[20] P. e. a. Virtanen, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.
[21] R. Tibshirani *et al.*, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.
[22] Y. Chen *et al.*, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.
[23] D. P. Kingma *et al.*, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[24] A. Paszke *et al.*, "Pytorch: An Imperative Style, High-performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.