

DEEP-LENS: Deep-Learning Powered Layout Extraction and Novel Segmentation for IC Assurance and Security

Shuvodip Maitra

Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India
shuvodipmaitra@iitkgp.ac.in

Abhishek Chakraborty

Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India
achakraborty25@kgpian.iitkgp.ac.in

Debdeep Mukhopadhyay

Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India
debdeep@cse.iitkgp.ac.in

Abstract—Ensuring the physical integrity of Integrated Circuits (ICs) at the microscopic level is essential for defending against threats like Hardware Trojans and counterfeiting in the electronics supply chain. This study presents enhanced segmentation and layout modification detection capability using electron microscopy images of a delayered IC. We have developed DEEP-LENS, a robust segmentation method combining Conditional Pixel Diffusion with a Dual Residual Shifted Window U-Net architecture. This approach effectively extracts layout features, such as standard cells, from noisy SEM images. DEEP-LENS achieved impressive performance metrics, including an Intersection over Union (IoU) of 0.908, a Mean Pixel Accuracy (mPA) of 0.955, and a Dice score of 0.952 on the test dataset. IoU measures mask overlap; mPA assesses class-wise pixel accuracy; and the Dice score emphasizes true positives, enhancing sensitivity to small segmentation errors. Additionally, it detected polygon modifications with over 91% accuracy compared to the original layout designs.

Index Terms—Hardware Trojan, layout extraction, IC gate-layer, image diffusion, segmentation

I. INTRODUCTION

In the context of Hardware Trojan detection, physical assurance-based techniques have gained prominence. These invasive techniques rely on depackaging, delayering, and Scanning Electron Microscope (SEM) imaging of IC layers to compare the extracted gate-level structures against the original GDSII layouts. The main challenges in these techniques are that these methods are labor-intensive, requiring significant manual intervention, to produce accurate results. There is a growing need for automation in the IC image analysis. Here, the main challenges result from SEM image noise, milling debris, and textural variations. Traditional image processing methods such as thresholding and Canny edge detection often fail under such conditions due to parameter sensitivity. Deep learning have been used in related IC imaging tasks like line edge roughness reduction [1], metal layer extraction [2], circuit annotation [3], standard cell component identification [4], and graphical extraction of standard cell connections [5].

This work is supported by the Centre on Hardware Security Entrepreneurship Research and Development (CHERD) project (HSE), and Information Security Education and Awareness Project (YAP); funded by MeitY, India.

II. BACKGROUND

Architectures such as U-Net [6] and Generative Adversarial Networks (GANs) [4] have shown promise for IC segmentation purposes. Beyond these, generative models like Variational Autoencoders (VAEs), Normalizing Flows, and Diffusion models [7] are increasingly applied in biomedical imaging, multimodal signal processing, and vision-language tasks, but remain underexplored for IC image segmentation. Diffusion models, including (DDPM) [8] and (DDIM) [9], have recently emerged as the gold standard in generative vision tasks, surpassing VAEs and GANs in capturing complex data distributions. Unlike VAEs, which often suffer from blurry outputs, and GANs, which face mode collapse, Diffusion models achieve superior fidelity and diversity by iteratively denoising noisy samples. These models are well-suited for noisy images because they smooth out noise while preserving important edges and structures. In this work, we investigate diffusion-based approaches to solve the challenges in IC image segmentation and layout modification detection tasks.

III. OUR CONTRIBUTIONS

In this work, we propose a Conditional Pixel Diffusion Model called **DEEP-LENS** to reconstruct highly accurate layout images from noisy SEM inputs. The main novelty and contributions are summarized below:

(1) We develop a novel self-attention that improves the network’s ability to detect subtle structural patterns in SEM images. It containing a combination of Rotary Positional embeddings and a Bilinear attention mechanism integrated into a shifted windowing-based hierarchical approach used in the Swin transformer [10].

(2) Using our developed attention mechanism, we have constructed a unique transformer architecture (having a shifted window-based hierarchical framework) that captures spatial features, as well as spatio-temporal features (using a separate cross-attention module), which is useful for guiding a diffusion-based process.

(3) In our novel transformer architecture, we have used novel dual-residual connections to improve and stabilize the

training process. Hence, we have named this as **Dual Residual Shifted Window (DRSW) transformer**.

(4) We have used a conditional Diffusion approach, where we have used ground truth images as conditioning input to guide the segmentation process. This has been accomplished for the first time in the context of IC image segmentation.

(5) We have constructed a novel **DRSW** based U-Net architecture to perform the crucial denoising process of the conditional diffusion approach. This U-net not only processes space tokens, but also time tokens, to develop a cross-modal spatio-temporal feature processing on delayed IC images.

IV. DEEP-LENS SEGMENTATION MODEL

The **DEEP-LENS** segmentation model is composed of a conditional Diffusion model, which is trained on ground truth image dataset. It is trained to generate GDS like images, from the delayed IC SEM images of standard cell layer. The **DEEP-LENS** model consists of a denoising U-Net architecture backbone, composed of customized **Dual Residual Shifted Window (DRSW) transformer** modules.

A **DRSW transformer** module is composed of a Swin Transformer type architecture (shown in Fig. 1), which breaks down input images into patches, where patches are of size $(2 \times 2) \text{ pixels}^2$, a shift size of 1 patch, a window size of (2×2) patches. We use rotary embeddings (RoPE) to assign the positional information of each of the image tokens. Firstly, from each image token (Z), we compute the Query (Q), Key (K), and Value (V) vectors along with the associated learnable weight matrices (W^Q, W^K, W^V). Then, the resultant vectors are multiplied by the RoPE embedding of the patches. Next, we compute the Bilinear Attention scores between the transformed Query (Q_{rot}) vector and the transformed Key (K_{rot}) vector, which is normalized by the dimensionality vector. Mathematically,

$$\text{Modified Self-Attention} = \text{softmax} \left(\frac{Q_{rot} W K_{rot}^T}{\sqrt{d_k}} \right) V_{rot} \quad (1)$$

Lastly, the resultant vector is multiplied by the transformed Value (V_{rot}) vector. This modified-self attention block described above for calculating the attention scores within the window is called Window Self-attention (WSA), and S-WSA denotes the corresponding shifted window version.

Bilinear mechanism captures richer and more complex interactions by allowing the relationship between Q and K to be modeled beyond simple dot products [11]. Bilinear attention provides greater flexibility in attending to multiple aspects of the input tokens simultaneously. The learnable matrix W provides an additional degree of freedom, allowing the attention mechanism to adapt dynamically to different tasks and datasets. This increased capacity can improve performance on tasks requiring understanding the complex interrelation between vision tokens for image interpretation.

The time embeddings are generated using sinusoidal embeddings [12]. In the cross-attention mechanism, the Query (Q) vectors are derived from these time embeddings, while the Key (K) and Value (V) vectors come from spatial embeddings

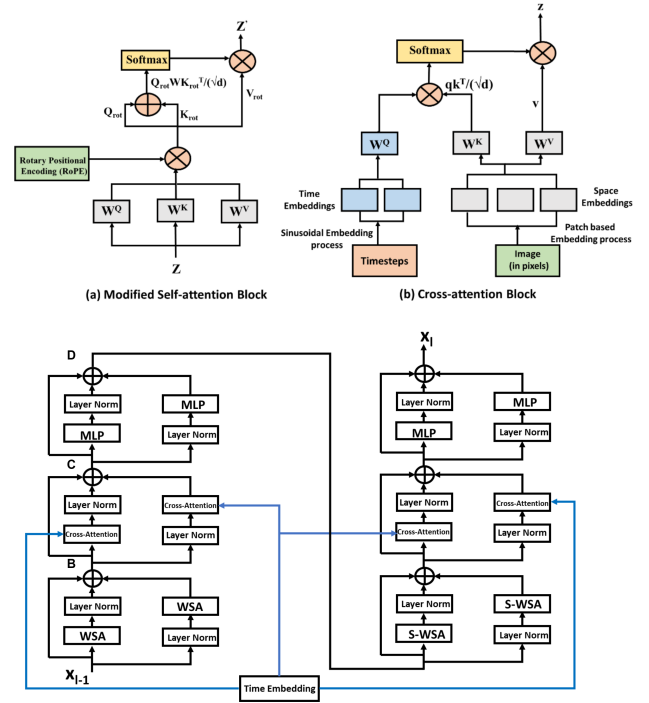


Fig. 1. Top: (a) Modified Self attention block; (b) Cross-attention block; (Bottom) DRSW architecture

obtained through patchification. Attention is computed via the dot product of Q and K , followed by a softmax and multiplication with V . This cross-attention captures spatio-temporal relationships between patches across different timesteps in the diffusion process, enabling effective modeling of multi-modal changes in vision tokens. The **Dual Residual Shifted Window (DRSW) Transformer** architecture comprises two sequential sub-blocks, where the output of the first sub-block is passed as the input to the second sub-block. Each sub-block consists of three core sub-modules. The first sub-module is the **Window/Shifted Window Self-Attention** ($L_{(S)-WSA}$) that computes self-attention within spatial windows or their shifted versions. Next sub-module is the **Cross attention** (L_{cross}) that models spatiotemporal relationships using time embeddings. Last sub-module is the **Multi-Layer Perceptron (MLP)** (L_{MLP}) which captures non-linear transformations using a feed-forward network consisting of a linear layer, followed by a Gaussian Error Linear Unit (GeLU) activation (defined as: $\text{GeLU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ denotes the cumulative distribution function of the standard normal distribution); and another linear layer. Let x_{l-1} denote the input to a sub-block, and $L_N(\cdot)$ denote the layer normalization operation. The intermediate outputs of the three sub-modules, denoted as B , C , and D , are computed as follows:

$$B = x_{l-1} + L_N (L_{(S)-WSA}(x_{l-1}) + L_{(S)-WSA}(L_N(x_{l-1}))) \quad (2)$$

$$C = B + L_N (L_{cross}(B, \text{Time embedding}) + L_{cross}(L_N(B), \text{Time embedding})) \quad (3)$$

$$D = C + L_N (L_{MLP}(C)) + L_{MLP}(L_N(C)) \quad (4)$$

In the overall architecture, the first sub-block uses standard window attention, while the second sub-block applies attention to shifted windows. The output of the second sub-block is denoted by x_l . Combining Pre-Layer Normalization (Pre-LN) and Post-Layer Normalization (Post-LN) optimizes training stability and output quality by leveraging the strengths of both approaches [13]. Pre-LN enhances gradient flow stability, enabling effective propagation through residual connections, while Post-LN ensures well-conditioned outputs for subsequent layers, improving representation learning. This hybrid approach offers flexibility in training dynamics, making it robust and adaptable to various architectures and tasks.

Algorithm 1: Algorithm for Conditional Diffusion Training

Input: Training dataset $\mathcal{D} = \{(c_i, x_i^{(0)})\}_{i=1}^N$, where c_i is the conditional image and $x_i^{(0)}$ is the target image;
Number of timesteps T , noise schedule $\{\beta_t\}_{t=1}^T$;
U-Net neural network ϵ_θ with parameters θ ;
Learning rate α , batch size B , number of epochs E
Output: Trained U-Net ϵ_θ

- 2 Initialize: $\alpha_t = 1 - \beta_t$ for $t = 1, \dots, T$;
- 4 Initialize: $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ for $t = 1, \dots, T$;
- 6 Initialize: Network parameters θ ;
- 8 **for** $epoch = 1$ to N **do**
- 10 **foreach** batch $\{(c_j, x_j^{(0)})\}_{j=1}^B$ from \mathcal{D} **do**
- 12 **for** $j = 1$ to B **do**
- 13 Sample timestep $t_j \sim \text{Uniform}(\{1, \dots, T\})$;
- 14 Sample noise $\epsilon_j \sim \mathcal{N}(0, \mathbf{I})$;
- 15 Compute noisy target: $x_j^{(t_j)} = \sqrt{\bar{\alpha}_{t_j}} x_j^{(0)} + \sqrt{1 - \bar{\alpha}_{t_j}} \epsilon_j$
- 16 Generate time embedding: $\text{emb}_j = \text{TimeEmbed}(t_j)$;
- 17 Predict noise: $\hat{\epsilon}_j = \epsilon_\theta(x_j^{(t_j)}, c_j, \text{emb}_j)$;
- 18 **end**
- 20 Compute loss: $\mathcal{L}_{\text{batch}} = \frac{1}{B} \sum_{j=1}^B \|\epsilon_j - \hat{\epsilon}_j\|_2^2$
- 21 where $\epsilon_j =$ true noise added to sample j
- 22 and $\hat{\epsilon}_j =$ predicted noise for sample j
- 24 Update parameters: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$;
- 25 **end**
- 26 **end**
- 27 **return** Trained U-Net ϵ_θ

For Diffusion based generative models, the goal of the forward diffusion process is to gradually add Gaussian noise to the input image data (x_0) over $1 \dots T$ timesteps, producing noisy versions (x_t) of the original data. The forward process follows the equation [9]:

$$q(x_t | x_0) := \int q(x_{1:t} | x_0) dx_{1:(t-1)} = \mathcal{N}(x_t; \sqrt{\alpha_t} x_0, (1 - \alpha_t) \mathbf{I}) \quad (5)$$

Here, α_t represents a noise schedule controlling how much noise is added at each step, and \mathbf{I} represents the identity matrix, which plays a crucial role in defining the covariance structure of the Gaussian noise added during each step. In Denoising Diffusion Probabilistic Models (DDPM), the reverse generative process is modeled as a Markov chain that progressively denoises a noisy sample to recover the original data [8]. A Markov chain is a stochastic process where the transition to the next state depends solely on the current state, satisfying the Markov property. This means that no memory of previous states is required, making the process memoryless. The reverse process in DDPM can be mathematically expressed as:

$$p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T}, \quad \text{where} \quad (6)$$

$$p_\theta(x_{0:T}) := p_\theta(x_T) \prod_{t=1}^T p_\theta^{(t)}(x_{t-1} | x_t) \quad (7)$$

Here, $p_\theta(x_0)$ represents the marginal probability of the original data, computed by integrating over the full reverse trajectory from x_T to x_0 , using the parameterized transition distributions $p_\theta^{(t)}(x_{t-1} | x_t)$ at each timestep.

Algorithm 2: Algorithm for Conditional DDPM Sampling

Input: Conditional image c , trained denoising network ϵ_θ ;
Number of timesteps T , noise schedule parameters $\{\beta_t, \alpha_t, \bar{\alpha}_t\}_{t=1}^T$;
Target image dimensions (C, H, W)
Output: Conditional generated image x_0

- 2 Precompute coefficients: $\alpha_t = 1 - \beta_t$ for $t = 1, \dots, T$;
- 4 Precompute coefficients: $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ for $t = 1, \dots, T$;
- 6 Precompute coefficients: $\beta_t = \beta_t$ for $t = 1, \dots, T$;
- 8 Sample $x_T \sim \mathcal{N}(0, \mathbf{I})$ with shape (C, H, W) ;
- 10 **for** $t = T, T - 1, \dots, 1$ **do**
- 12 Generate time embedding: $\text{emb}_t = \text{SinusoidalTimeEmbed}(t)$;
- 14 **Predict noise:** $\hat{\epsilon}_t = \epsilon_\theta(x_t, c, \text{emb}_t)$;
- 16 **Compute posterior mean:** $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_t \right)$;
- 18 **Compute posterior standard deviation:** $\sigma_t = \sqrt{\beta_t} = \sqrt{\beta_t}$;
- 20 **if** $t > 1$ **then**
- 21 Sample noise: $z \sim \mathcal{N}(0, \mathbf{I})$;
- 22 $x_{t-1} = \mu_\theta(x_t, t) + \sigma_t \cdot z$;
- 23 **end**
- 25 $x_0 = \mu_\theta(x_t, t)$; No noise for final step
- 26 **end**
- 27 **return** Generated conditional image x_0

In Denoising Diffusion Implicit Model (DDIM), the reverse process is made deterministic by introducing an implicit dependency across timesteps, leading to a more efficient and controllable sampling process. The reverse process actually prepares a rough sketch of the original image (x_0) and then slowly modifies it with the reverse process. The reverse process in DDIM is mathematically expressed as [9]:

$$x_{t-1} = \underbrace{\sqrt{\alpha_{t-1}} \left(\frac{x_t - \sqrt{1 - \alpha_t} \epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{"predicted } x_0\text{"}} + \underbrace{\sqrt{1 - \alpha_{t-1} - \sigma_t^2} \epsilon_\theta^{(t)}(x_t)}_{\text{"direction pointing to } x_t\text{"}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}} \quad (8)$$

$\epsilon_\theta^{(t)}(x_t)$ represents the predicted noise at timestep t , estimated by a neural network. σ_t is the variance scaling coefficient, which controls the noise introduced at each timestep. ϵ_t is the random noise sampled from $\mathcal{N}(0, \mathbf{I})$, used to introduce stochasticity when $\sigma_t > 0$. The neural network model is trained to predict the Gaussian noise added during forward diffusion to help recover the original data by iterative denoising. In DDIM, the choice of σ_t determines whether the sampling process includes stochasticity (like DDPM) or follows a deterministic path. Unlike DDPM, which requires a large number of sampling steps, DDIM can generate high-quality samples in fewer steps. This is achieved by skipping

intermediate timesteps while still maintaining data consistency, making DDIM much faster for inference.

Conditioning in generative models can be achieved through several approaches [14] such as class-based conditioning, which generates images based on class labels; text or image conditioning that uses textual descriptions or reference images to guide the generation process; and latent space conditioning that uses latent representations to influence and control the output. The forward diffusion process remains largely unchanged compared to the unconditional case. But, the reverse process is modified to incorporate the conditioning variable 'y' [14]:

$$p_{\theta}(\mathbf{x}_{0:T} | y) = p_{\theta}(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t, y) \quad (9)$$

Here, y represents the conditioning information, such as a class label, a text prompt, or a reference image. This condition is applied at every timestep in the reverse process to guide the generation in accordance with the desired attributes.

Algorithm 3: Algorithm for Conditional DDIM Sampling

Input: Conditional image c , trained denoising network ϵ_{θ} ;
Number of inference steps S , training timesteps T ;
Target image dimensions (C, H, W) , ;
determinism parameter $\eta \in [0, 1]$;
Output: Conditional generated image x_0

- 2 Compute step ratio = T/S ;
- 4 Sample $x_S \sim \mathcal{N}(0, \mathbf{I})$, with shape (C, H, W) ;
- 6 Set $\bar{\alpha}_0 = 1.0$:Final cumulative product ;
- 8 **for** $i = S, S-1, \dots, 1$ **do**
- 10 **Map inference step to training timestep:** $t = i \times \text{ratio}$;
- 12 $t_{\text{prev}} = (i-1) \times \text{ratio}$;
- 14 Generate time embedding: $\text{emb}_t = \text{SinusoidalTimeEmbed}(t)$;
- 16 **Predict noise:** $\hat{\epsilon}_t = \epsilon_{\theta}(x_i, c, \text{emb}_t)$;
- 18 **Estimate original sample:** $\hat{x}_0 = \frac{x_i - \sqrt{1 - \bar{\alpha}_t} \cdot \hat{\epsilon}_t}{\sqrt{\bar{\alpha}_t}}$;
- 20 **Get previous timestep cumulative product:** ;
- 21 **if** $t_{\text{prev}} \geq 0$ **then**
- 22 $\bar{\alpha}_{t_{\text{prev}}} = \bar{\alpha}_{t_{\text{prev}}}$;Access from precomputed values
- 23 **end**
- 24 $\bar{\alpha}_{t_{\text{prev}}} = 1.0$;Boundary condition
- 26 **Compute variance for stochastic sampling:**
- $\text{variance} = \frac{1 - \bar{\alpha}_{t_{\text{prev}}}}{1 - \bar{\alpha}_t} \cdot \left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t_{\text{prev}}}}\right)$;
- 28 $\sigma_t = \eta \cdot \sqrt{\text{variance}}$;
- 30 **Compute predicted previous sample:**
- $\text{pred_sample_direction} = \sqrt{1 - \bar{\alpha}_{t_{\text{prev}}} - \sigma_t^2} \cdot \hat{\epsilon}_t$;
- 32 $x_{i-1} = \sqrt{\bar{\alpha}_{t_{\text{prev}}}} \cdot \hat{x}_0 + \text{pred_sample_direction}$;
- 34 **if** $\eta > 0$ **and** $i > 1$ **then**
- 35 Sample noise: $z \sim \mathcal{N}(0, \mathbf{I})$;
- 36 $x_{i-1} = x_{i-1} + \sigma_t \cdot z$
- 37 **end**
- 38 **end**
- 39 **return** Generated conditional image x_0

A U-Net is widely employed in denoising tasks for diffusion models such as DDIM and DDPM due to its encoder-decoder structure, which effectively captures both global context and local details; the encoder path extracts high-level features while the decoder reconstructs fine-grained details, enabling the modeling of complex noise distributions. In this work, we utilize a **DRSW-based U-Net architecture** (see Fig. 2) for diffusion denoising, where two inputs—an original SEM image and a condition image (ground truth)—are patchified, linearly

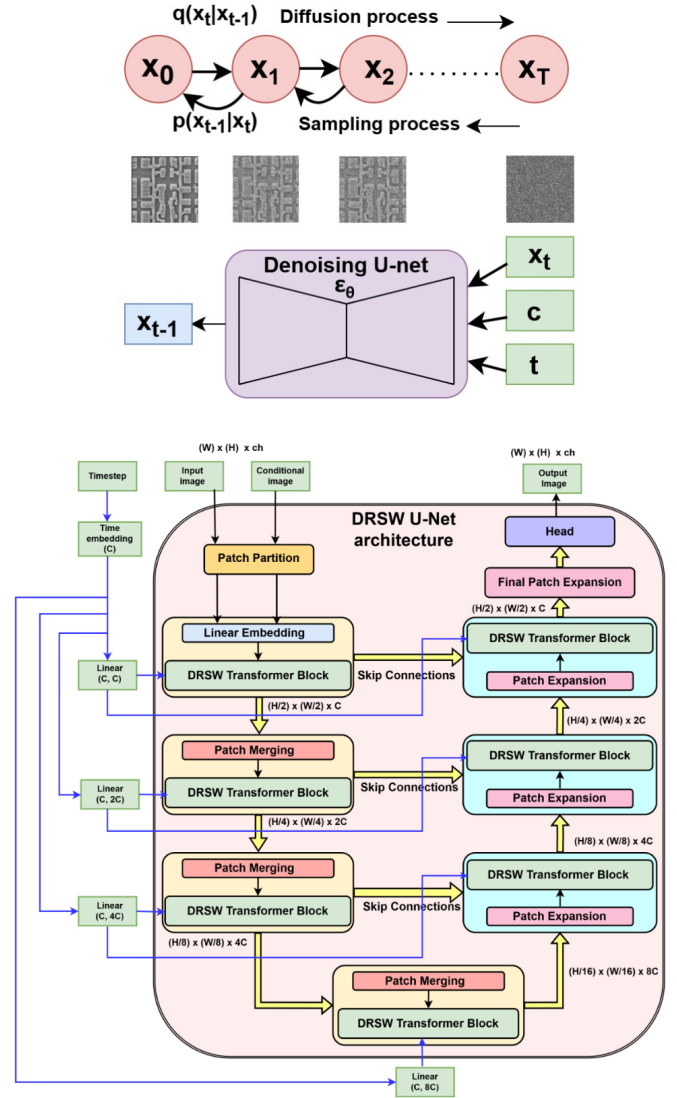


Fig. 2. (Top): Conditional Diffusion sampling; and (Bottom): DRSW U-net architecture for denoising

embedded into visual tokens, and fused via a linear layer. The encoder comprises three sub-blocks: the first with a linear embedding and DRSW transformer, and the next two with patch-merging layers followed by DRSW transformers, facilitating multi-scale representation while reducing computational load. A bottleneck layer further processes features through patch-merging and DRSW blocks. The decoder mirrors the encoder with three sub-blocks incorporating patch expansion layers and DRSW transformers, where patch expansion upsamples features by splitting patches and reducing channel dimensions. It also employs layer normalization and linear layers for $2\times$ up-sampling, ultimately restoring full-resolution outputs suitable for pixel-level tasks such as segmentation. Time embeddings are projected and added to each sub-block to align temporal and spatial embeddings for cross-attention. The architecture, embedding dimension ($C = 96$), and optimized hyperparameters were chosen to balance the model's capacity to learn complex patterns, overfitting risk, computational complexity,

and memory usage. The U-Net has 66.97 million trainable parameters with a model size of 255.51 MB in float32 format. The training process for noise prediction using the conditional diffusion framework is presented in Algo. 1, while DDPM and DDIM-based sampling for conditional image generation is detailed in Algo. 2 and Algo. 3, respectively.

V. EXPERIMENTAL SECTION

The IC chip microprocessor used in our experiments was extracted from an IoT embedded system board and initially packaged in a Quad Flat No Lead (QFN) package. The chip was desoldered using a hot air gun and de-packaged through a wet chemical process involving nitric acid and sulfuric acid, as described in [15], exposing the bare silicon die. Hydrofluoric acid (HF) was then applied to selectively remove passivation layers, metal layers, and inter-layer dielectric (ILD), leveraging its high selectivity for etching silicon dioxide (SiO_2) over silicon (Si), which revealed the gate layer containing the standard cell structure within minutes. The etched gate layer was rinsed with distilled water to ensure proper exposure for imaging. High-resolution images of the gate layer were acquired using a Zeiss CrossBeam 340 SEM in Secondary Electron (SE) mode at 5 kV, producing grayscale images of size 256×256 pixels with a resolution of 23 nm per pixel, resulting in a dataset of 1296 images covering an area of approximately $156\mu m \times 156\mu m$. These images were processed using the **DEEP-LENS** model to distinguish gate-layer structures from noisy and debris-affected SEM data. The model was trained on two Nvidia A5000 GPUs (24 GB VRAM each) using the PyTorch Lightning framework and Adam optimization in data-parallel mode. Its performance was evaluated through various metrics after the training, in model evaluation mode.

VI. RESULTS AND DISCUSSION

After de-packaging the chip (see Section V), the bare silicon die is extracted and imaged using a Scanning Electron Microscope (SEM), as shown in Fig. 3(a). The die measures approximately $(3.63 \text{ mm} \times 3.63 \text{ mm})$ and is initially covered with an opaque passivation layer. Following this, delayering is performed to expose the gate layer (as shown in Fig. 3(b)), imaged at 2.5 kX magnification. The image reveals repetitive standard cell structures separated by vertical power lines (V_{DD} and V_{SS}), consistent with known VLSI physical design practices. A further magnified view at 5 kX (depicted in Fig. 4) highlights standard cells—polygonal or rectangular with rounded edges—measuring about 700–800 nm in width and several microns in length. The image also displays texture variations and etching debris due to the delayering process, complicating segmentation of standard cell layouts from the noisy SEM data.

During training, both SEM and ground truth images are fed as input into the **DRSW U-Net** denoising network backbone of the **DEEP-LENS** model, where the network learns the noise distribution. The dataset follows an 80:20 train-test split. Training used a learning rate of 0.0001 and a batch size of 16, constrained by available CUDA memory on dual Nvidia

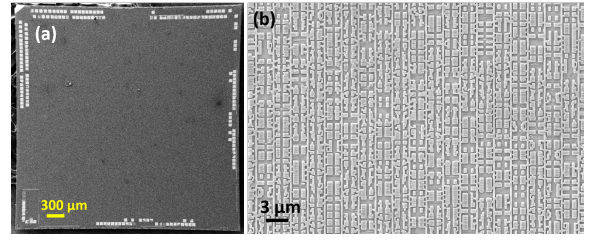


Fig. 3. SEM images of (a) Bare Silicon Die; and (b) exposed Gate layer of the IC

TABLE I
EFFECT OF SAMPLING PROCESS ON EVALUATION METRICS
(MODEL TRAINED FOR 2000 EPOCHS)

Type of Sampling	Number of samples	IOU score	mPA score	Dice score
DDPM	4	0.769	0.878	0.870
DDPM	10	0.842	0.923	0.914
DDPM	25	0.800	0.898	0.889
DDIM	10	0.920	0.965	0.958
DDIM	25	0.912	0.961	0.959
DDIM	50	0.914	0.962	0.955

A5000 GPUs. The predicted noise guides the reverse Diffusion process to reconstruct the segmented image.

The sampling or reverse process plays a critical role in segmentation quality. DDPM sampling, being stochastic and Markovian, requires many timesteps (1000 in our setup), leading to slow inference and variable output quality. In contrast, DDIM sampling is deterministic and non-Markovian, using fewer steps (200 in our case) while producing consistent, sharper segmentations. After training our **DEEP-LENS** model using the **DRSW U-Net** denoising architecture for 2000 epochs, we evaluated both sampling methods on a single SEM input image by generating multiple outputs, averaging them, and applying binary thresholding to obtain the final segmentation. As shown in Table I, DDIM outperformed DDPM with superior IOU, mPA, and Dice scores. A visual comparison is presented in Fig. 5.

Subsequently, we assessed the impact of the number of sampling steps on DDIM performance (shown in Table II left). Using 10 generated outputs from one SEM input image, we found that increasing the sampling steps improves texture and detail at the cost of inference time. An optimal balance was achieved at 400 steps, providing high-fidelity output with reasonable speed. We then examined the effect of training epochs of the **DRSW U-Net**. Underfitting at low epochs and

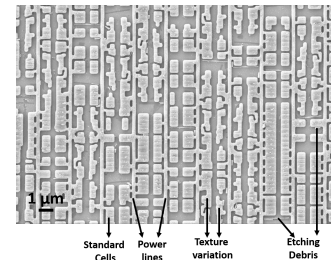


Fig. 4. High resolution SEM image of the IC showing noise and imperfections

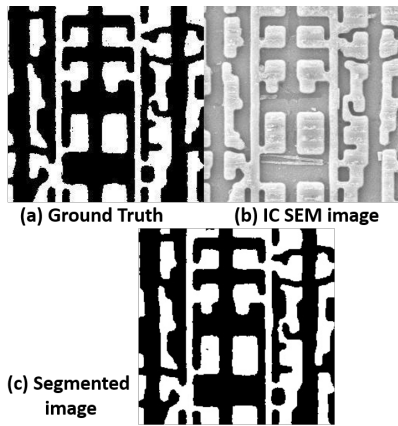


Fig. 5. (a) Ground Truth; (b) Original SEM image and (c) Segmented image (using our DEEP-LENS model)

TABLE II

(LEFT) EFFECT OF DDIM STEPS ON EVALUATION METRICS (10 SAMPLES FOR SINGLE TEST IMAGE); (RIGHT) EFFECT OF EPOCHS ON EVALUATION METRICS (10 SAMPLES FOR THE WHOLE TEST DATASET)

DDIM Steps	IoU	mPA	Dice	Epochs	IoU	mPA	Dice
200	0.920	0.965	0.958	500	0.845	0.924	0.916
300	0.902	0.956	0.948	1000	0.900	0.951	0.947
400	0.923	0.966	0.960	2000	0.908	0.955	0.952
500	0.921	0.965	0.959	4000	0.903	0.952	0.949

overfitting at high epochs were observed, which degraded segmentation quality, whereas 2000 epochs yielded optimal results (shown in Table II right), averaged over the test dataset.

To test the model’s robustness in detecting layout modifications, we simulated Hardware Trojan scenarios by randomly modifying polygons from ground truth GDSII images (Fig. 6). The SEM inputs were then segmented using DDIM, and polygon differences between outputs and modified ground truths were measured. Table III shows that for polygons sized 1000–5000 $pixels^2$, our model achieved over 91% addition detection accuracy, and over 97% deletion detection accuracy on high-resolution (2048×1536) images, demonstrating effectiveness even under noise and debris. In Table III, the first three columns (from the left) represent the experiments to simulate the situation where some additional polygons are present in the manufactured IC (compared to the ground truth GDSII images). In comparison, the last three columns (from left) of Table III represent the experiments to simulate the

TABLE III
POLYGON MODIFICATION DETECTION TABLE

Area ($pixel^2$)	Number of modified images	Addition detection (%)	Area ($pixel^2$)	Number of modified images	Deletion detection (%)
1000	10	90.45 %	1000	10	98.73 %
	100	91.28 %		100	97.35 %
2000	10	90.98 %	2000	10	97.41 %
	100	91.44 %		100	97.41 %
3000	10	91.62 %	3000	10	97.72 %
	100	91.38 %		100	97.71 %
4000	10	91.28 %	4000	10	97.28 %
	100	91.37 %		100	97.06 %
5000	10	91.36 %	5000	10	98.07 %
	100	91.43 %		100	97.39 %

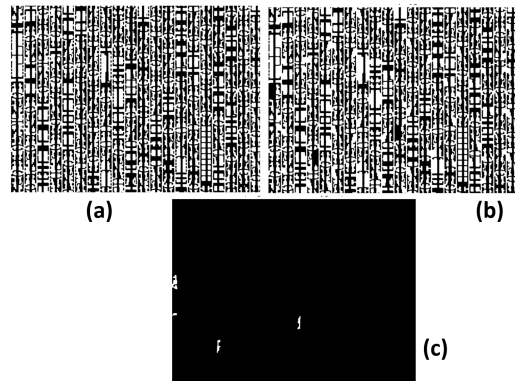


Fig. 6. (a) Original ground truth image; (b) Deleted polygon image; and (c) Difference image

TABLE IV
PERFORMANCE COMPARISON TABLE

Segmentation Method	IoU score	mPA score	Dice score
Canny Edge detection	0.0563	0.4778	0.1066
Ridge detection	0	0.5185	0
Sobel filter	0	0.4629	0.0120
Laplace of Gaussian (LOG)	0	0.5199	0
Replicated method [16]	0.8978	0.232	0.9461
DeepLab V3	0.1283	0.5202	0.1840
Segformer	0.4612	0.5559	0.6223
SAM (ViT-B)	0.4330	0.6316	0.5981
SAM (ViT-L)	0.48225	0.5981	0.6475
SAM (ViT-H)	0.4963	0.6506	0.6491
DEEP-LENS (Ours)	0.908	0.955	0.952

situation where additional polygons are present in the ground truth (GDSII images) images which are not found in the SEM images of the manufactured IC.

Finally, the performance of the **DEEP-LENS** segmentation model is compared with state-of-the-art image filtering methods used in semiconductor failure analysis (Table IV). We also replicated the deep-learning-based SEM2GDS model [16], which uses two connected U-Nets with weighted Dice loss, training it on our SEM dataset for 100 epochs, batch size 16, and learning rate 0.0001, with its metrics reported in Table IV. Additionally, we evaluated three popular pre-trained models from computer vision: DeepLab V3 [17], Segformer [18], and SAM [19]. The results show that, compared to these traditional and deep-learning models, **DEEP-LENS** consistently achieves superior evaluation performance. The process workflow is easily scalable to 16/7/2 nm technology node chip too, limited by the 1 nm minimum resolution of our SEM imaging system.

VII. CONCLUSION

This work introduces **DEEP-LENS**, a custom segmentation method tailored for noisy SEM images of the gate layer in IC chips which achieved high segmentation accuracy, with an IoU of 0.908, mPA of 0.955, and Dice score of 0.952 across the test dataset; and also demonstrated over 91% polygon modification detection capability. These results underscore **DEEP-LENS** model’s accurate performance, and make it a highly valuable tool for the physical assurance-based cyber forensics domain.

REFERENCES

- [1] Y.-Y. Tee, D. Cheng, Y. Shi, T. Lin, and B.-H. Gwee, "Integrated circuit mask–generative adversarial network for circuit annotation with targeted data augmentation," *IEEE Intelligent Systems*, vol. 39, no. 1, pp. 37–45, 2024.
- [2] S. Maitra, T. S. Sarkar, A. Chakraborty, and D. Mukhopadhyay, "Physical layout extraction via ion milling based ic delayering for reverse engineering applications," in *2024 IEEE Physical Assurance and Inspection of Electronics (PAINE)*, 2024, pp. 1–9.
- [3] Y.-Y. Tee, X. Hong, D. Cheng, C.-S. Chee, Y. Shi, T. Lin, and B.-H. Gwee, "Patch-based adversarial training for error-aware circuit annotation of delayered ic images," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 9, pp. 3694–3698, 2023.
- [4] M. M. Al Hasan, N. Vashistha, S. Taheri, M. Tehranipoor, and N. Asadizanjani, "Generative adversarial network for integrated circuits physical assurance using scanning electron microscopy," in *2021 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, 2021, pp. 1–12.
- [5] E. Huang, X. Hong, T. Lin, Y. Shi, and B. H. Gwee, "Gracer: Graph-based standard cell recognition in ic images for hardware assurance," *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:265256528>
- [6] D. Cheng, Y. Shi, T. Lin, B.-H. Gwee, and K.-A. Toh, "Delayed ic image analysis with template-based tanimoto convolution and morphological decision," *IET Circuits, Devices & Systems*, vol. 16, no. 2, pp. 169–177, 2022. [Online]. Available: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cds2.12093>
- [7] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 850–10 869, 2023.
- [8] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf
- [9] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022. [Online]. Available: <https://arxiv.org/abs/2010.02502>
- [10] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021. [Online]. Available: <https://arxiv.org/abs/2103.14030>
- [11] Z. Li, T. Gu, B. Li, W. Xu, X. He, and X. Hui, "Convnext-based fine-grained image classification and bilinear attention mechanism model," *Applied Sciences*, vol. 12, no. 18, 2022. [Online]. Available: <https://www.mdpi.com/2076-3417/12/18/9016>
- [12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: <https://arxiv.org/abs/1706.03762>
- [13] S. Xie, H. Zhang, J. Guo, X. Tan, J. Bian, H. H. Awadalla, A. Menezes, T. Qin, and R. Yan, "Residual: Transformer with dual residual connections," 2023. [Online]. Available: <https://arxiv.org/abs/2304.14802>
- [14] Z. Zhan, D. Chen, J.-P. Mei, Z. Zhao, J. Chen, C. Chen, S. Lyu, and C. Wang, "Conditional image synthesis with diffusion models: A survey," 2024. [Online]. Available: <https://arxiv.org/abs/2409.19365>
- [15] S. Maitra, A. Chakraborty, and D. Mukhopadhyay, "Microstructural investigation and serial section tomography on asic chips for assurance through reverse engineering applications," in *2023 IEEE Physical Assurance and Inspection of Electronics (PAINE)*, 2023, pp. 1–8.
- [16] T. Lin, Y. Shi, and B. H. Gwee, "Sem2gds: A deep-learning based framework to detect malicious modifications in ic layout," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.
- [17] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017. [Online]. Available: <https://arxiv.org/abs/1706.05587>
- [18] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," 2021. [Online]. Available: <https://arxiv.org/abs/2105.15203>
- [19] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023. [Online]. Available: <https://arxiv.org/abs/2304.02643>