

# Lab Manual for Hardware Trojan Detection in ICs Using SEM Images

## 1 Objective

The learning objective of this module is for trainees to gain hand experiences on IC level Hardware Trojan Detection by using nano-image analysis and artificial intelligence. Trainees will learn step by step, how to use physical inspection methods such as scanning electron microscopy (SEM) to detect any malicious changes from the backside of an IC. Hardware Trojans are malicious changes to the design of integrated circuits (ICs) at different stages of the design and fabrication processes. Advanced computer vision algorithms in combination with the neural network models are used to classify authentic and malicious cells from an IC under authentication by assigning as a unique descriptor for each type of logic cells/gates. These descriptors are compared with a golden standard to detect any subtle changes on the active region, which can raise the flag for the existence of a potential hardware Trojan.

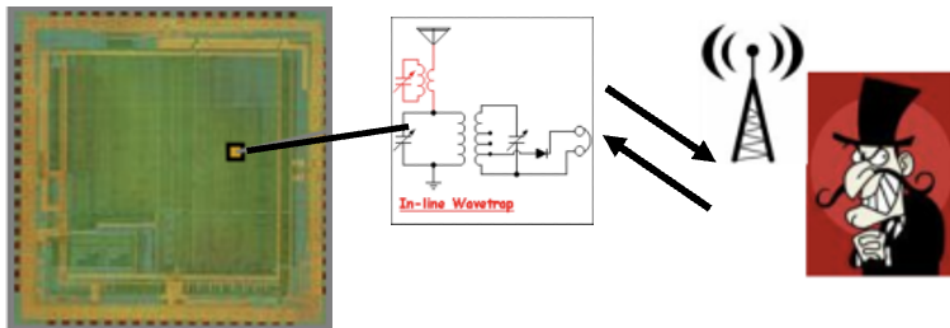


Figure 1: IC Level Hardware Trojan

## 2 Equipment and Software Needed for this Module:

This experiment involves multi-disciplinary knowledge of the following subject and requires following equipment's, samples and software:

- IC Samples: IC samples under authentication in which presence of hardware Trojan needs to be investigated.
- Golden Layout: Trusted Trojan Free Layout Design
- IC Sample Preparation Instrument: Allied X-Prep
- Dual Beam SEMs: An SEM (scanning electron microscope) with FIB (focused ion beam)
- Software: Conda Environment that supports Python v 3.9 or later running on a Windows/Mac based machine.

## 3 Background

Hardware Trojans are malicious modifications in integrated circuits (ICs) with an intent to breach security and compromise the reliability of an electronic system. This experiment uses, advanced nano-imaging, and image processing with neural networks to detect hardware Trojans inserted by untrusted foundries. An IC with on-chip trusted test structures (logic cells) with layout OR SEM images of the IC with trusted (golden circuits) and not investigated logic cells will be provided as a starter data set. The on-chip golden circuits provide authentic samples for image-based Trojan detection. The experiment will be performed on a 28nm backside thinned FPGA.

### 3.1 Prerequisites

- Fundamental Knowledge of Semiconductor device layer and packaging.
- IC Sample Preparation (Optional if Polished IC Sample is available)
- M14. Scanning Electron Microscope Training
- Fundamentals of image processing and data science using Python

## 4 Experimental Setup

- Sample Preparation Station (Optional)
- SEM Imaging Station (See Manual Module M14)
- A computer setup with conda python 3.9 or later environment with OpenCV, pandas, matplotlib and scikit-learn python packages installed.

## 5 Procedure

This experiment will be performed in the following steps as sub-modules:

- Sample Preparation (Optional, if sample is ready).
- FIB and SEM Imaging for data images collection.
- Trojan Detection System using image analysis and artificial intelligence.

## 6 Sample Preparation

Decapsulation: Internal die, lead frame and die connection components – bond wire, ball grid arrays – are all exposed when the die is decapsulated. Decapsulation methods such as mechanical polishing and computer numerical control (CNC) multi-tool milling are non-selective. Scanning electron microscopy (SEM) imaging is impossible even after exposing the bare die because electrons cannot penetrate a thick silicon substrate layer.

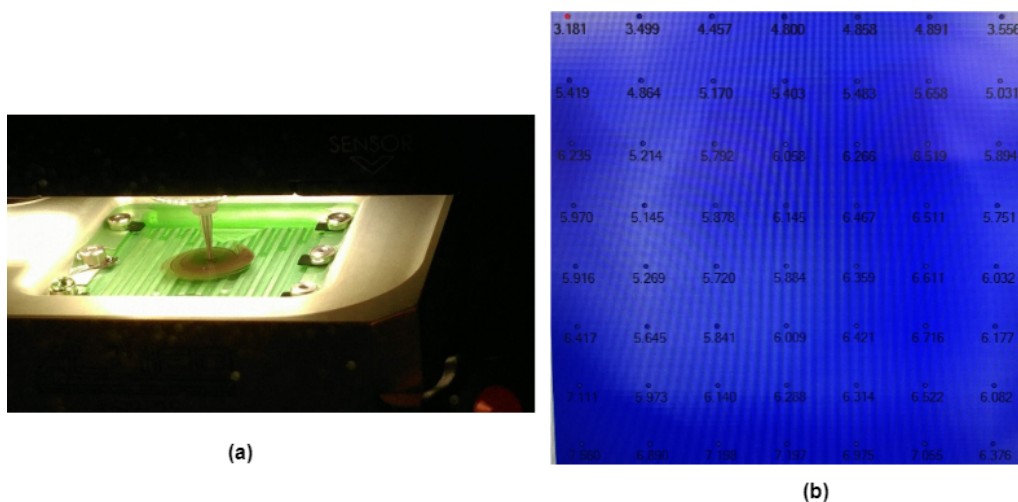


Figure 2: (a) chamber view of Allied X-prep (b) 2D thickness measurement mapping after polishing

The substrate must be thinned even more using precise polishing techniques. Furthermore, the bare die employed in the process is not flat, and its curvature changes during polishing, potentially resulting in uneven silicon substrate removal over the chip. Advanced Sample preparation machine, Allied X-Prep, sophisticated silicon die polishing technology, can be utilized to accomplish backside thinning of up to 1-2  $\mu\text{m}$  to mitigate these difficulties. Two common selective decapsulation methods are wet etching and plasma etching. We employ a 28nm FPGA in this experiment. FPGAs are widely used in communication systems, military systems television boxes. By embedding a Trojan in the circuitry, an adversary can acquire sensitive or confidential information, cause a data breach, and inflict financial loss to a entity.

A FPGA die can be flip chip or enclosed in a mold epoxy resin. The first step in decapsulating (if packaged) the FPGA chip to expose the die by grinding the top surface followed by the exposed die can be polished and thinned down to less than  $1\mu m$  by using precise polishing.

## 7 FIB and SEM Imaging

### 7.0.1 FIB Delayering

For SEM imaging and analysis, the silicon substrate needs to be thinned below  $1\mu m$ . The advanced polishing machines cannot be used for thinning sub-micron, so a plasma assisted focused ion beam will be used for further silicon thinning. In this step, a trainee has expected to finished experiement module M14 earlier.

### 7.0.2 SEM Imaging

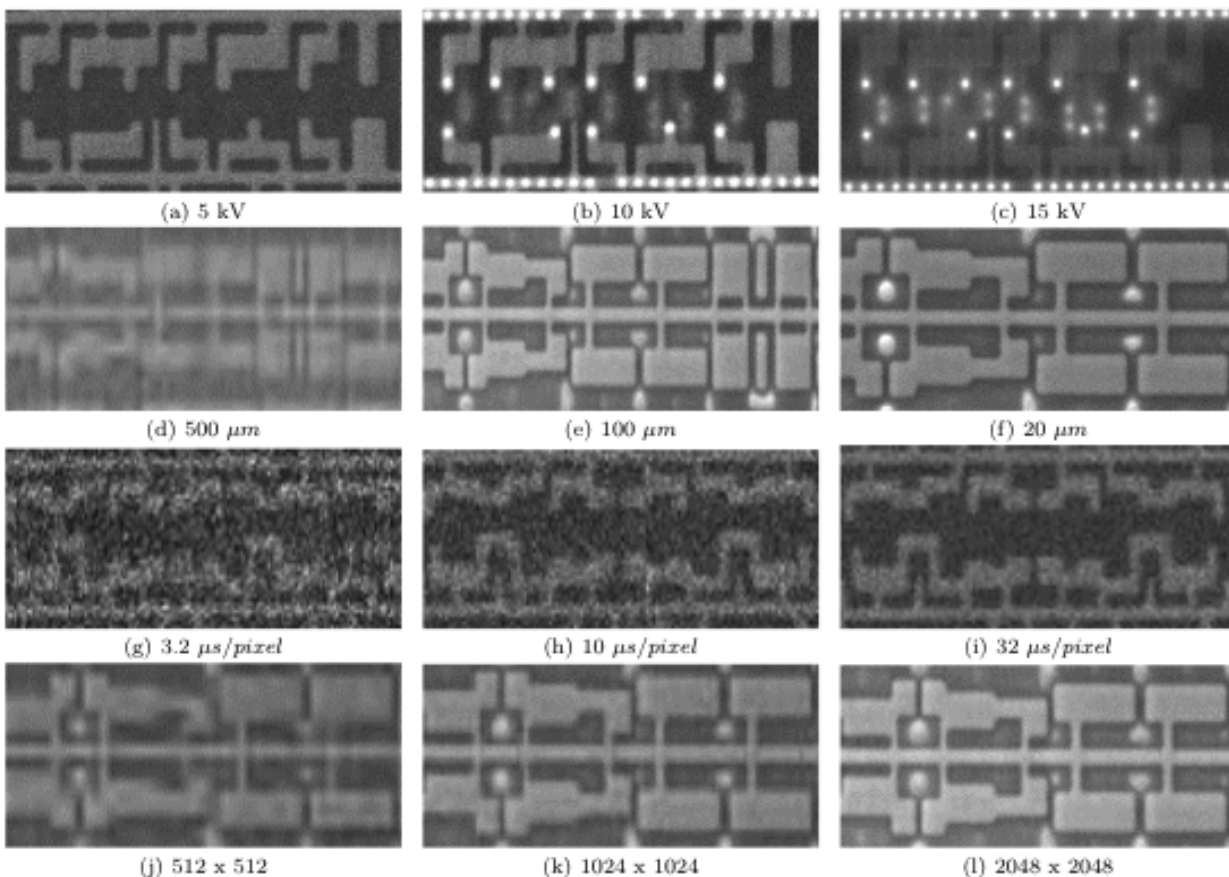


Figure 3: SEM images variations with different Beam Voltages [(a),(b) and (c)], Field of Views [(d),(e) and (f)], Dwelling Times [(g),(h) and (i)] and Resolutions [(j),(k) and (l)].

High-resolution images from the FPGA are needed and can be taken with a scanning electron microscope (SEM). The goal is to scan the entire die as quickly as possible while obtaining enough feature details to compare with the SEM image of authentic logic cells. The following SEM Parameters influence the timing and quality of SEM images. By altering one parameter at a time while leaving all other factors constant, we can see the effect of one parameter on SEM images.

- Beam voltage - The electron's beam accelerating voltage (in kV) determines the penetration depth of electrons inside an item. While imaging from the backside, a 5kV beam can expose active regions, whereas 10kV can further expose sub-surface features, including the polysilicon and higher metal layers.
- Field of view (FOV) - The magnification of the image is inversely proportional to the field of view. Because of the low magnification, a large field of view covers more features, but they are fuzzy. With a smaller field of view, imaging duration increases.
- Dwelling time (speed) - A higher dwelling time increases the image's signal-to-noise ratio, hence better quality of SEM images, but it increases the time to capture images. Meanwhile, it also affects the surface charging that can introduce artifacts in imaging.

The microscope can also be configured to scan the entire die in the form of small windows of images, which are then stitched together to make a complete panorama image when the above-mentioned parameters have been specified. Images captured with a large field of view and a short dwell period take less time to imaging, but their quality is not good. A small field of view, extended dwell time, and high-resolution capture more excellent image quality, but it collects more data than necessary and lengthens the imaging process. As a result, there is a trade-off between imaging time and image quality in order to achieve higher Trojan detection findings. SEM parameters optimization is done to balance time consumption and detection confidence.

## 8 Trojan Detection System

In this section, we are going to design an end-to-end real-time trojan detection system. Our computer vision-based approach consists of a logical cell detection and a cell recognition unit. A schematic diagram of the system is presented in Fig. 4.

Each SEM image is pre-processed and then analyzed to separate out cell rows. Cell images are extracted from each of those rows. Extracted cell images are passed through the cell recognition unit. Based on the output of the cell recognition unit and the corresponding entry in DEF file, possible Trojan presence is decided as mentioned in Fig. 4.

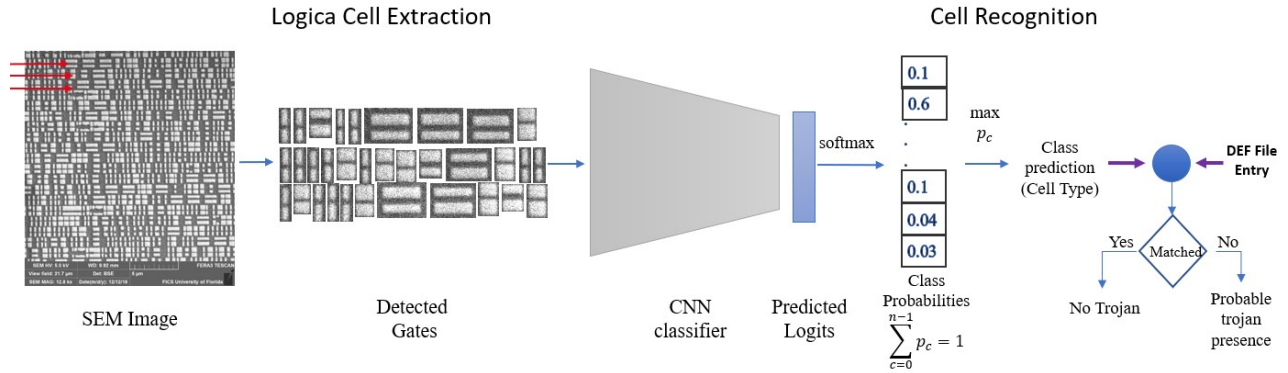


Figure 4: Full System Architecture

### 8.1 Pre-processing

Components of an image are detected by grouping out same intensity pixels. Each image is binarized to separate out gate regions (foreground pixels) from background easily. Image is denoised beforehand to escape noisy binarization. Accordingly, our pre-processing stage starts with denoising the input image followed by the binarization and grouping out foreground or gate regions through findings of connected components (see Fig. 5)

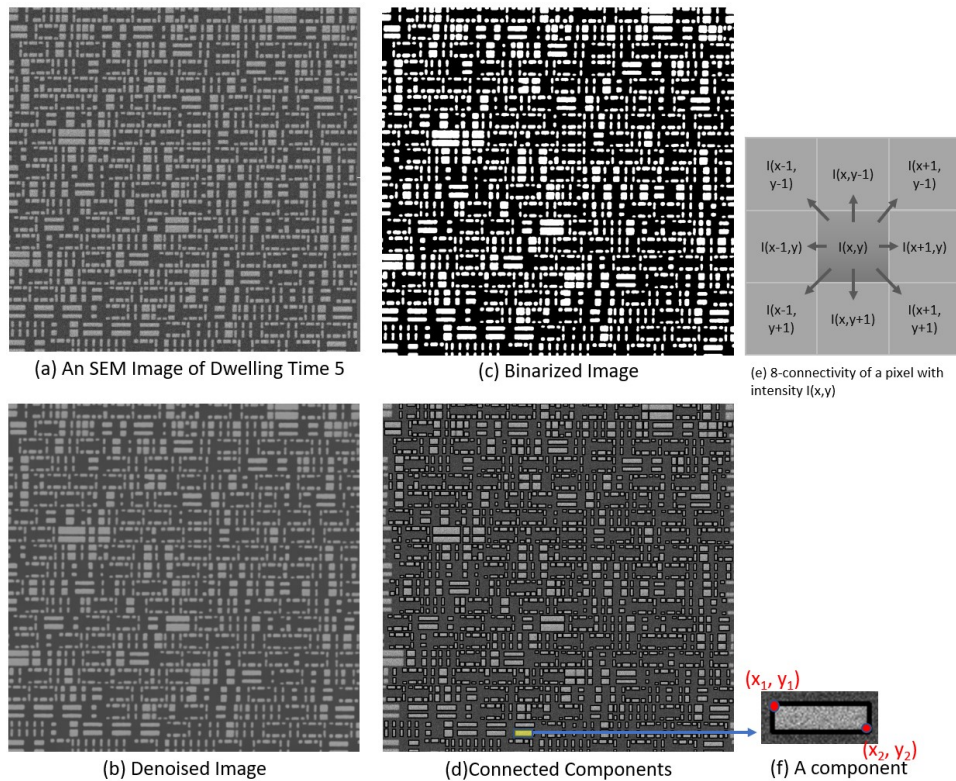


Figure 5: Pre-processing stages

Denoising Images are denoised by using non-local means of pixels. Instead of averaging out a group of pixels(say, a 5x5 window) surrounding a target pixel to smooth the image, similar patches are found throughout the image and averaged out to calculate the value of the target pixel. The whole process and parameter set-up are followed as discussed in [?] and applied on the input image I to obtain the denoised image,  $I_D$  (Fig. 5(b)).

$$I_D = \text{FastNonLocalMeans}(I)$$

Binarization Binarized Image  $I_B$  (Fig. 5(c)) is generated from denoised image  $I_D$  instead of original image I to avoid disturbance caused by probable salt-and-pepper noise of I. To select the appropriate threshold for binarization, we explored both global and local adaptive thresholding method. Otsu's method [?] is used while finding the global threshold. For local thresholding i.e., thresholding images part by part, fast binarization is followed as in [?].

$$I_B = \text{Binarization}(I_D)$$

Components Obtained binary image,  $I_B$  (pixel value set,  $P_B = \{0, 255\}$ ) is used to group out gate regions in the foreground pixels (pixels with value 255).  $I_B$  is scanned pixel-to-pixel(from top to bottom and left to right) and adjacent pixels with same intensity values are grouped together. The labelling works on different measures of connectivity. In our case, 8-connectivity of each pixel is considered for grouping (Fig. 5(e)). The background pixel was not considered while calculating connected components. A set of distinctively labelled connected components,  $C = \{c_0, c_1, c_2, \dots, c_{T-1}\}$  is obtained from this stage (Fig. 5(d)) where each  $c_i$  represents a rectangular component region with top-left point  $(x_1, y_1)$  and bottom-right point  $(x_2, y_2)$  as shown in Fig. 5(f) and T represents total number of components in the image I. C's are sorted vertically by  $y_1$ .

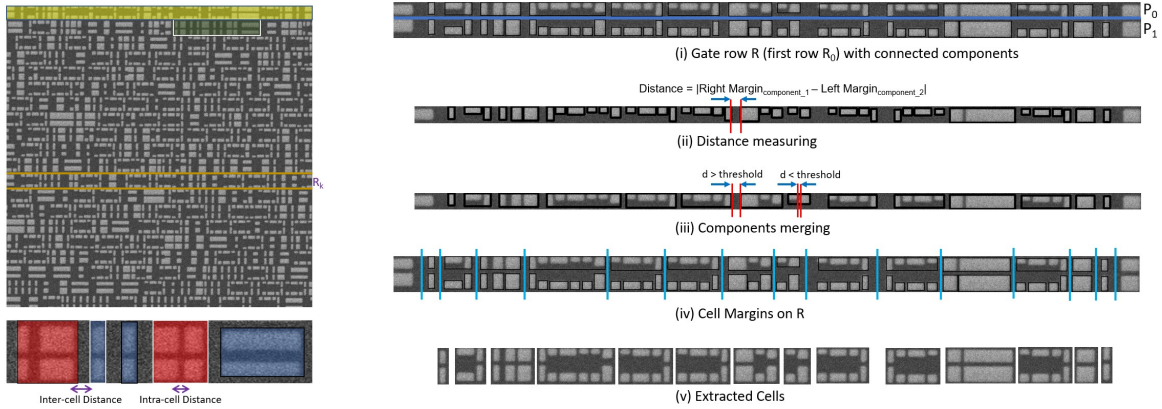
## 8.2 Cell Extraction

Each image  $I$  along with its connected component list C are used to extract out cell images. Image  $I$  is divided into N rows  $R = R_1, R_2, R_3, \dots, R_N$  where each  $R_i$  represents a row of gate (Fig. 6). This row division can be done by first vertically sorting the component list C and then comparing y co-ordinates between each consecutive components.

Cell can be made of single or multiple entities (Fig. 6). So we need to check whether any component is itself a cell or part of a cell. Cells are extracted by scanning R's one-by-one. Each R is composed of two vertically symmetric parts P's (as shown in Fig. 6b(i)). We proceeded with one of the P's (see Fig. 6b(ii)) for the convenience. Component list  $C_i = c_0, c_1, \dots, c_{m-1}$  associated with row  $P_i$  is scanned from left to right, distances D between consecutive components are measured (Fig. 6b(ii)) where

$$D = \{d_1, d_2, \dots, d_{m-1}\}$$

$$d_j = |c_j.x_1 - c_{j-1}.x_2|; \quad 1 \leq j \leq m - 1$$



(a) top - An SEM image, bottom-green highlighted portion from top image with single(blue shaded) and composite cells(red shaded) marked

(b) cell extractions from a row(yellow shaded in (a) top)

Figure 6: SEM image and cell extraction from its rows

If distance between two consecutive components falls below a threshold, those components are merged. That is,

$$c_j = \text{merge}(c_{j-1}, c_j) \quad \text{if } d_j < \text{threshold}$$

In our experiments, we set the threshold 9 pixel but it differs from image to image. A global threshold can be determined by analyzing gaps from all available images which will be done in future.

After merging we end up with margins for both composite and single components in  $P$ . Using the same margin, cells are identified and extracted from the corresponding gate row  $R$  (see Fig. 6b((iv)-(v))). Cells are annotated manually. These annotated cells constitute the real image dataset  $D_R$ .

### 8.3 Synthetic Cell Image Generation

To build a comprehensive recognition system, we need to train a robust classifier. However, a comprehensive dataset is required to achieve that objective, which is not always available. Building a dataset is a critical limitation that cannot solely rely on the IC SEM image acquisition process. A number of specific parameters involved in this process include, dwelling time, magnification, brightness, and contrast that can negatively impact the acquisition time. Besides, the IC SEM images include variations in images due to variation in pixel-intensity, noise, brightness, and contrast. Along with the imaging variations due to the quality of sample polishing process, this further reduces the possibility of using conventional image augmentation methods.

So, we are going to generate synthetic version of cell images using the extracted ones to address the insufficient data problem. Out of several ways of synthetic data generation, the generative adversarial network (GAN) [?] training set-up is preferred as it holds the current state-of-the-art in the respective arena.

**Generative Adversarial Network** In the adversarial set-up, a generator function  $G$  produces synthetic data by mapping random noise variables to data-space. In our case, random variables are sampled from Gaussian distribution.

$$\begin{aligned} z &\sim N(0, 1) \\ I_s &= G(z) \end{aligned} \tag{1}$$

where  $N(0,1)$  represents normal distribution with zero mean and unit variance.

On the other hand, a discriminative function  $D$  acts as a critic and outputs a probability value indicating the input being close to real data distribution or not.

$$\begin{aligned} D(x) &= 1 \quad \text{if } x \in D_R \\ D(x) &= 0 \quad \text{if } x \in G(z) \end{aligned} \tag{2}$$

Over the period of learning,  $D$  is trained to maximize the output of assigning correct labels to both real images and synthetic images by  $G$ . That is,  $D$  is trained to maximize the expectation for both real and synthetic data.

$$E[D] = \max E_{x \sim D_R}[\log D(x)] + E_{z \sim N(0,1)}[\log(1 - D(G(z)))] \tag{3}$$

The above expectation maximizes when  $D$  outputs values close to 1 for real data samples and 0 for synthetic data samples.

On the other hand,  $G$  is trained to fool the discriminator by generating images similar to real images. At that point  $D$  starts outputting higher values for synthetic images  $G(z)$  that leads to minimize  $\log(1-D(G(z)))$ . The following expectation is minimized for generator.

$$E[G] = \min E_{z \sim N(0,1)}[\log(1 - D(G(z)))] \tag{4}$$

Putting equations 3 and 4 together, we see that  $G$  and  $D$  play two-player minimax game with value  $V(D,G)$ .

$$\min_G \max_D V(D, G) = E_{x \sim D_R}[\log D(x)] + E_{z \sim N(0,1)}[\log(1 - D(G(z)))] \tag{5}$$

**Choice of GAN** Our training scheme of GAN needs to be selected based on two design decisions.

- Images need to be generated class-wise to reduce human efforts of labelling.

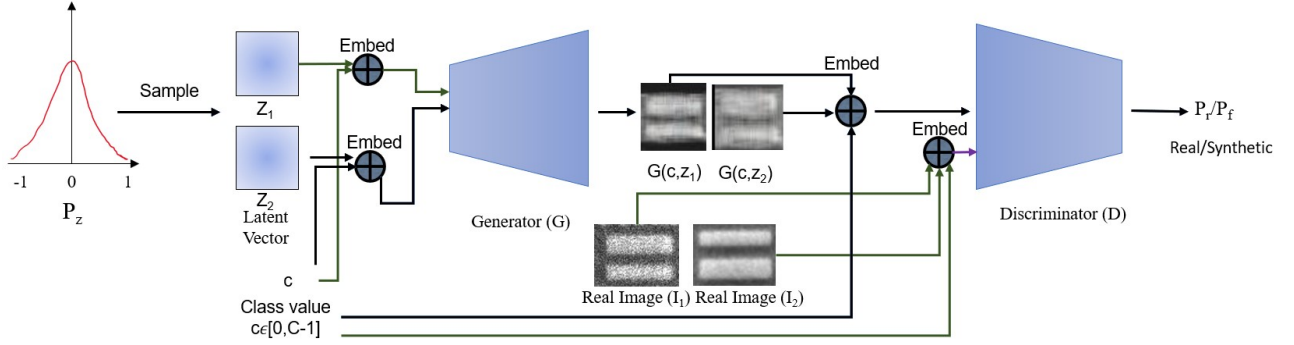


Figure 7: Implementation of MSGAN for our system

- Prevent mode-collapse i.e., refrain generator from generating samples for only one class.

Firstly, to generate synthetic images conditioned on classes, we have encoded class label information with the sampled random variables (see Fig. 7) and passed it to the generator. The class conditional formulation (Eqn. 6) is followed from the conditional GAN [?].

$$L_{cGAN} = E_{x \sim D_R}[\log D(x, c)] + E_{z \sim N(0,1)}[\log(1 - D(G(c, z)))] \quad (6)$$

Secondly, in case of mode collapse, mapped images are collapsed into a few modes i.e., two closely sampled latent codes  $z_1$  and  $z_2$  are highly likely to be mapped to same mode of images (Eqn. 7).

$$G(c, z_1) \approx G(c, z_2) \text{ where } z_1, z_2 \approx N(0, 1) \text{ and } z_1 \approx z_2 \quad (7)$$

To address this issue, the distinctive mapping of two closely sampled random variables ( $z_1, z_2$ ) can be enforced by mode-seeking regularization term (Eqn. 8) as in Mode-Seeking GAN [?].

$$L_{ms} = \frac{\max d_I(G(c, z_1), G(c, z_2))}{d_z(z_1, z_2)} \quad (8)$$

The overall objective function can be written as following.

$$L = L_{cGAN} + \lambda_{ms} L_{ms} \quad (9)$$

where  $\lambda_{ms}$  controls the weight of the mode seeking ratio.

Synthetic data  $D_S$  generated this way is used alongside  $D_R$  to train the CNN classifier.

## 8.4 Logical Cell Recognition

A CNN classifier will be trained on the real ( $D_R$ ) and synthetic ( $D_S$ ) dataset. Any popular architecture can be chosen as the backbone of the network. The network should generalize on the logical cell images enough to differentiate even among classes with less inter-class variability.

Two things should be kept in mind while designing the network.

- Attacker can change the gate shape in a way that it may closely resemble one of the existing cells. The classifier should be able to identify if any cell image is even slightly out of distribution of known cells. Domain adaptation techniques can be explored to solve this.
- Different cell images come with drastically different aspect ratio. The classifier should be size invariant.

On testing time, we will pass cell images one-by-one from each location of an SEM image into the classifier. The classifier will generate a probability maps over classes from where the class with maximum probability value will be selected (see Fig. 4). The output will be matched with the entry from DEF file for the corresponding location. Based on the agreement between two values, the presence of trojan can be detected.

## 9 Learning Outcome

This work has been done to detect malicious changes by an untrusted foundry inside an IC using Xilinx FPGA. After successfully finishing this experiment, a trainee will learn understand, what can be an hardware Trojan, Sample Preparation Process, various SEM imaging modalities. Also how to use image analysis and artificial intelligence methods for hardware assurance.