2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Enhancing Post-Silicon Validation Through Generative Adversarial Networks (GANs) for Test Case Generation

Vidhya Sagar and Aditi Bharmaik
Intel Technology India Pvt. Ltd. Bengaluru, Karnataka
vidhya.sagar@intel.com, aditi.bharmaik@intel.com

Abstract- This paper explores the innovative application of Generative Adversarial Networks (GANs) in the realm of post-silicon validation, specifically for the generation of test cases. As semiconductor devices grow in complexity, ensuring their reliability and functionality through exhaustive validation becomes increasingly challenging. While effective, Current test case generation methods may not cover the vast space of potential defect scenarios, especially in novel or complex designs. We propose a GAN-based approach to generate diverse, comprehensive, and challenging test cases, aiming to improve defect detection rates and overall silicon reliability.

Keywords—Post silicon validation, GAN, semiconductor, SOC

I. INTRODUCTION

Post-silicon validation is a crucial phase in the semiconductor development lifecycle, bridging the gap between chip fabrication and its commercial deployment. This stage is pivotal for ensuring that the manufactured chip adheres to its design specifications and performs optimally under real-world conditions. Despite the rigor of pre-silicon validation techniques, such as simulations and formal verification, certain issues can only be detected and rectified when the physical chip is tested. Post-silicon validation not only confirms functional correctness and performance metrics like speed, power consumption, and thermal properties but also identifies manufacturing defects that could impair functionality or reliability. Efficient validation processes are instrumental in reducing time-to-market and associated costs, ensuring product quality, and facilitating system-level integration. Moreover, for chips used in critical applications across automotive, aerospace, and healthcare sectors, post-silicon validation is essential for meeting stringent industry standards and regulatory requirements. As semiconductor devices grow in complexity and become more integral to various technologies, the significance of post-silicon validation in ensuring the reliability and performance of semiconductor products continues to escalate. [1]

Current post-silicon validation methods, which rely on manually created test cases and automated test pattern generation, often struggle to cover the complex operational scenarios encountered by modern semiconductor devices. Current test case generation methods for post-silicon validation face significant challenges due to the complexity of modern semiconductor devices, the necessity for exhaustive coverage, and the limitations inherent in current approaches. These challenges include complexity, comprehensive coverage, Intermittent failures, Evolving designs, etc. [2]

a. Complexity:

Modern semiconductor devices, including CPUs, GPUs, and SoCs (System on Chips), feature billions of transistors, intricate architectures, and multifaceted functionality. This complexity makes it difficult to generate test cases that cover all possible operational scenarios and corner cases. Current methods may not efficiently explore the vast state space of these devices, potentially missing critical defects.

b. Comprehensive Coverage:

Achieving thorough coverage across functional, structural, and timing aspects is challenging, often requiring a vast number of test cases that are resource-intensive to create.

c. Evolving Designs:

Semiconductor designs and manufacturing technologies are constantly evolving, introducing new features and capabilities. Test case generation methods need to adapt to these changes, ensuring that they remain effective for validating the latest devices. This requires ongoing development and refinement of test generation techniques, which can be challenging to sustain.

d. Depth vs. Breadth:

Balancing detailed testing of specific features against broader coverage to ensure no overlooked aspect is a significant challenge.

e. Resource and Time Constraints:

The pressure to accelerate time-to-market necessitates efficient test case generation and execution within tight schedules.

f. Integration with Validation Infrastructure:

Ensuring that new test case generation methods seamlessly integrate with existing validation tools and workflows is crucial for their adoption and effectiveness.

Overcoming these challenges requires innovative, adaptive approaches, potentially leveraging AI and ML technologies, to enhance the efficiency and comprehensiveness of post-silicon validation efforts.

II. Generative Adversarial Networks: An Overview

Generative Adversarial Networks (GANs) are a groundbreaking approach in the field of artificial intelligence, particularly within the domain of unsupervised learning. Developed by Ian Goodfellow and his colleagues in 2014, GANs have revolutionized the way machines understand and generate data. The core idea behind GANs is to pit two neural networks against each other in a game-theoretic scenario, where one network strives to generate data, and the other aims to discriminate between the generated data and the real data. [3]

A. Structure of GANs

A GAN consists of two main components:

1. **Generator (G)**: This network learns to create data that mimics some distribution. The generator starts with a random noise vector (z) and transforms this into data that resembles the target distribution, attempting to fool the discriminator.
2. **Discriminator (D)**: This network acts as a judge, trying to distinguish between real data drawn from the training set and fake data produced by the generator. The discriminator's goal is to accurately classify the incoming data as real or fake.

B. Objective Function

The interaction between the generator and the discriminator can be described by the following minimax objective function:

$$\mathrm{Min}_G \mathrm{Max}_D \, E_{x\sim Pdata(x)}[\log D(x)] + E_{z\sim Pz(z)}[\log(1-D(G(z)))]$$

- The discriminator (D) maximizes this function, trying to assign the correct labels to both real and generated data.

- The generator (G) minimizes this function, trying to produce data that will be labeled as real by the discriminator.

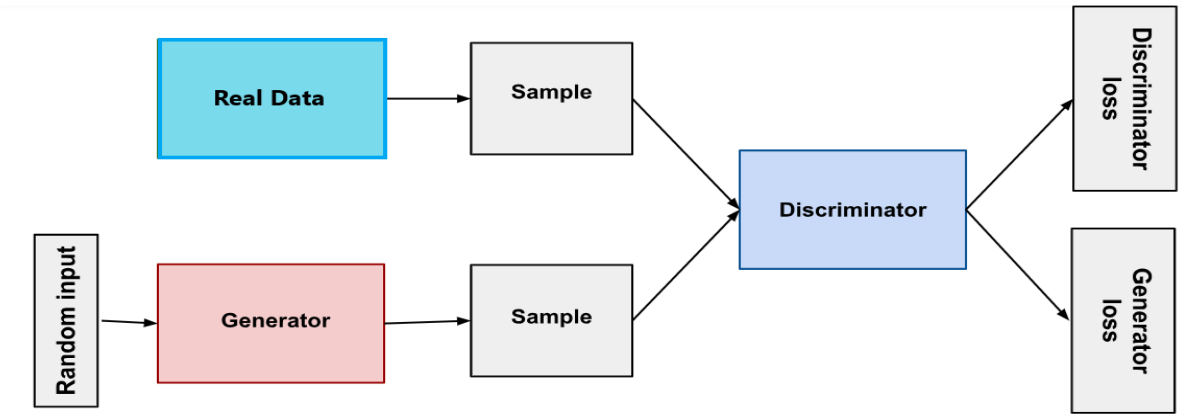

Figure 1. General GAN Architecture

C. Training Algorithm

1. **Initialization**: Begin by assigning random initial weights to both the generator and the discriminator networks.
2. **Iterative Training**: Continue the following steps iteratively until the model converges:
    - **Discriminator Training**:
        - Draw a sample batch of actual data from the dataset.
        - Generate a batch of random noise vectors.
        - Use the generator to transform these noise vectors into synthetic data.
        - Calculate the loss for the discriminator based on its ability to distinguish between the real and generated data.
        - Adjust the discriminator's weights to reduce this loss, enhancing its accuracy.
    - **Generator Training**:
        - Create a new batch of random noise vectors.
        - Produce synthetic data by processing these vectors through the generator.
        - Evaluate the generator's loss, which is determined by how well the discriminator classifies the synthetic data as real.
        - Optimize the generator's weights to decrease this loss, aiming to fool the discriminator more effectively.

This training process leverages the adversarial relationship between the generator and discriminator, refining the capabilities of both networks through continuous feedback and adjustment.

III. PROPOSED METHODOLOGY

Proposed method to generate a test plan which has optimal test scenarios and corner cases involves several steps to insure an optimum output.

a. Extraction of Features and Capabilities

It is necessary to get the features and capabilities of the products which accurately defines the functionalities and requirement of client from this product. This usually can be found from the Hardware Architecture Specification (HAS) document of the product. This includes all the new features added from previous gen to next gen and the features which are inherited from any previous or parent product.

b. Extraction of Legacy Testlines

In this step the testlines which were used by previous products are extracted. These testlines includes the details of various test scenarios associated with a test feature. These testlines are not only going to be a crucial and important aspect of the data which will be used to train the GAN model but also will be used in the testplan which is being generated in order to reduce the time taken in the generation of test scenarios.

c. Defining Test Scenarios

In this step, potential test scenarios are defined for the extracted features and capabilities. These scenarios cover normal operation as well as the boundary conditions considering following aspects: -

- Functionality: - Feature should work according to the requirement in various conditions.
- Performance: - Feature should work appropriately under heavy load and stressful conditions.
- Compatibility: - It checks that whether the function is specific to any particular device or system configuration.

d. Utilize GAN for Test Case Generation

Once the features and their potential test scenarios are defined the next step is to generate a testplan. This testplan will include different test cases generated by the GAN model which was trained on the data from legacy testplans.

e. Review & Refine

The test cases generated in the previous step are needed to be reviewed and then regenerated in order to get more précised test plans.

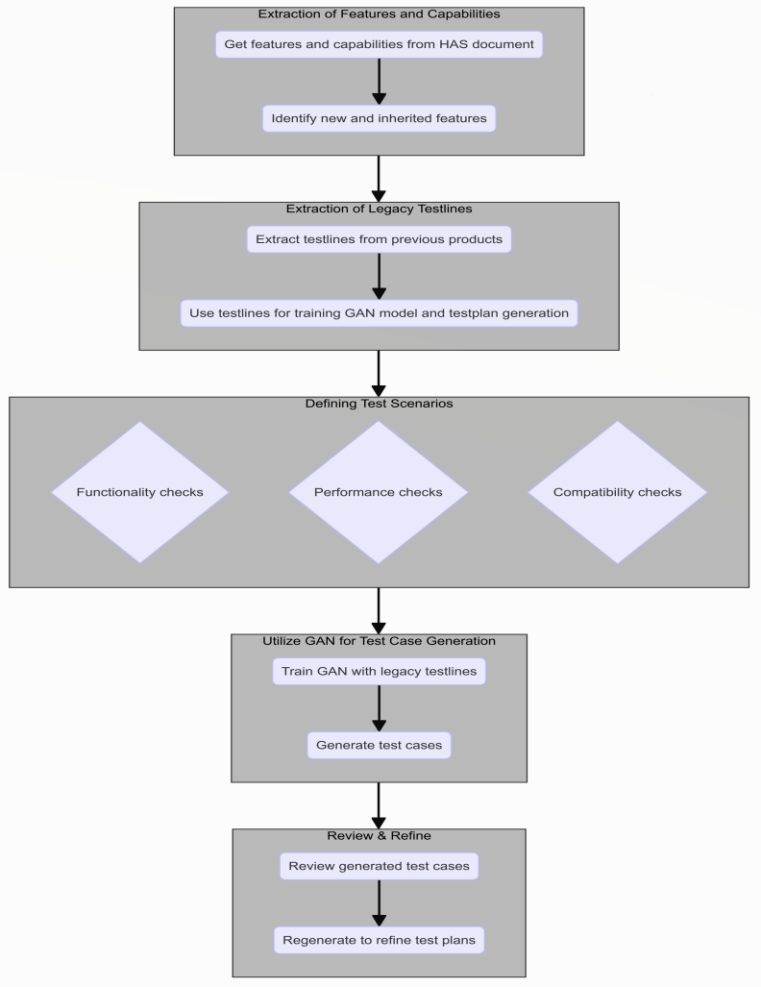

Figure 2. Test Case Generation Flow

IV. RESULTS

we utilized a legacy dataset of test cases for developing the test features using a specifically designed dummy dataset. This dataset included a variety of scenarios, from standard use cases to more complex edge cases, ensuring a comprehensive training environment for the Generative Adversarial Network (GAN).

The data after refinement looks like the following tabulation arrangement:

TABLE I
DUMMY DATASET

Feature_name	**T1**	**T2**		**T8**
Commercial OS	Install and Boot	Power Management	•••	VT-x
DMA Remapping	Address Translation	Address Translation Services (ATS)	•••	Peer to Peer
Faults and Errors	External Faults	Internal Faults (Non-recoverable)	•••	Internal Faults (Recoverble)
IOMMU VMX	IOMMU Guest Access Real Mode	IOMMU Virtual Interrupt Delivery	•••	IOMMU Virtual Processor ID (VPID)
. . .	. . .	. . .	. . .	. . .
Misc Features	Bios Compliance	PCIE Misc	•••	Perfmon checkout

The GAN, trained on this legacy data, demonstrated ability in creating test scenarios that mimic the test environment provided by the legacy dataset. This shows its potential effectiveness in practical applications. The refined dataset, post-training, is detailed in the following format, showcasing the scenarios and outcomes generated by the GAN.

```
Epoch 93/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - loss: 0.1768
Epoch 94/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.1672
Epoch 95/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - loss: 0.1582
Epoch 96/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.1496
Epoch 97/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - loss: 0.1414
Epoch 98/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.1337
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.1265
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - loss: 0.1196
completed in gan_arch ########################################
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
Test functionalities to cover for 'DMA Remapping':

1. Address Translation
2. Address Translation Services (ATS)
3. Draining of Requests to Memory
4. Pass-through for Devices
5. Peer to Peer
```

Figure 3. Generated Testplan

V. Conclusion and Future Enhancements

The research presented in the paper concludes that the integration of Generative Adversarial Networks (GANs) into the post-silicon validation process marks a significant advancement in the field of semiconductor manufacturing. The findings demonstrate that GANs can effectively generate complex and realistic test cases that traditional methods struggle to produce. This capability can lead to enhanced coverage of potential defects and operational scenarios, directly addressing the limitations of existing validation techniques.

In order to enhance the ability of the model created so far, the requirement of a refined data comes into the picture which can be created by the use of NLP models on the Hardware Architecture Specification document to extract the features. The generated data from these can be used for training of GAN, which in turn will result in the better performance of GAN model.

Another requirement that catches the eye is the need of GAN to work freely instead of conditional boundation which is being used. Because of this conditional boundation, the GAN is restricted to the data provided and mimics the behavioral pattern of input data instead of generating new test scenarios which pushes the product to it's limits in order to get the complete idea of it's capabilities and breaking points.

Therefore, we can conclude that the combination of work done so far and the suggested enhancements can result in a model which can mimic real-life conditions, thereby uncovering hard-to-find bugs and improving the reliability of semiconductor devices. Since the whole process can be automated and human resource interference is almost null the possibility of human error & chances of overlooking any possible test case scenarios are almost impossible. Overall, the research underscores the potential of GANs to revolutionize post-silicon validation by offering a more efficient, comprehensive, and automated approach to test case generation, paving the way for the development of more robust semiconductor products.

References

[1] Di Mattia, Federico, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. "A survey on gans for anomaly detection." arXiv preprint arXiv:1906.11632 (2019).

[2] Ioannides, Charalambos, and Kerstin I. Eder. "Coverage-directed test generation automated by machine learning--a review." ACM Transactions on Design Automation of Electronic Systems (TODAES) 17, no. 1 (2012): 1-21.

[3] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, "Generative Adversarial Networks: An Overview," in IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53-65, Jan. 2018, doi: 10.1109/MSP.2017.2765202.