Harnessing Regenerative AI and Machine Learning for Efficient Fault Simulation

Himanshu Vishwakarma, Lakshya Miglani and Gopi Srinivas Deepala

October 8, 2024

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

Harnessing Regenerative AI and Machine Learning for Efficient Fault Simulation

Himanshu Vishwakarma, Engineer – VLSI Design, Silicon Interfaces, Mumbai, India
(himvish@siliconinterfaces.com)

Lakshya Miglani, Engineer – VLSI Design, Silicon Interfaces, Mumbai, India
(lakshya@siliconinterfaces.com)

Gopi Shrinivas Deepala, Engineer – VLSI Design, Silicon Interfaces, Mumbai, India
(gopi@siliconinterfaces.com)

Abstract— **This paper introduces a novel method for design validation that merges Fault Simulation with Artificial Intelligence, leveraging Machine Learning techniques. Initially, we used all available Test Vectors to verify the Design under Test (DUT) by applying standard Fault Simulation methods such as Stuck@0 and Stuck@1, which serve as the training patterns for the ML model. These vectors, processed through an EDA Simulation tool, simulate stuck faults on each input signal and log the outputs. This logged data then feeds into an AI/ML framework to develop a predictive model. The model is trained on a randomly selected 20% (for better models we can use up to 80%) subset of the data, using densely layered neural networks optimized with activation functions like ReLU and Sigmoid, and fine-tuned through hyperparameters such as epoch length, accuracy, data loss, and learning rate. After training, the model is tested for robustness against the remaining 80% of the test vectors, using heat maps and other graphical tools to assess performance and ensure result validity. Subsequent validations of the model account for any changes or updates to the DUT, with tests conducted using only 20% of the updated test vectors to predict outcomes on the larger 80% portion. Comparisons of these results with initial runs highlight any discrepancies and fault statuses. The methodology proposed not only significantly cuts down on the duration of simulations but also reduces reliance on extensive testability tools, enhancing efficiency in the validation process.**

Keywords— Regenerative, AI, ML, EDA, ReLU, Sigmoid, Accuracy,

I. INTRODUCTION

In the rapidly advancing field of electronic design, ensuring that circuits and systems work correctly is a major challenge. Traditionally, engineers use specific testing methods, like Stuck@0 and stuck@1, to find faults in their designs. These methods simulate potential errors in the circuits, helping identify problems. However, as electronic designs become more complex, these traditional testing methods alone aren't enough to catch every error efficiently and the problem is that for large circuits, like SOC the Gate Counts are increasing beyond 100 million and need very expensive high-performance computing on distributed and parallel computing farms to resolve and even then takes a long duration of time which is detrimental to the time to market guidelines.

To improve this process, our paper introduces a new approach and method that blends traditional testing methods with modern fifth-generation Artificial Intelligence (AI) techniques, specifically using machine learning (ML). This combination aims to make testing faster and more accurate, reducing the need for exhaustive testing and saving valuable time. Here's how it works: first, we use all available test data to simulate potential faults in the design. This creates a large dataset describing how the design behaves under different fault conditions. We then use just 20% (for better models we can use up to 80%) of the sub-data to train an AI model, teaching it to recognize patterns and predict faults. This model uses complex algorithms to learn from the data, which is fine-tuned through a process of adjusting various settings to improve its accuracy and reliability.

After training, we check how well the AI model works by testing it against the remaining 80% of the data. We also use tools like heat maps to visually assess the model's performance. This model is built using random 20% or up to 80% of the data as training data and model optimized dense layers (using activation function – ReLU Sigmoid, Tanh, LeakyReLU, etc.) and working with hyperparameters (epoch, accuracy, data loss, learning rate) for forward and backward propagation of error/loss functions as well as optimizers (RMSProp and Adam) to have a robust model matches results of the simulation. The model is then further used with future run simulations of the design with 20% Test Vectors and the generative AI/ML model predicts accurately 80% of the test vector outputs. This may be compared with the gold first run and variations would highlight the fault status.

Our innovative method, combining AI with traditional simulation, not only speeds up the testing process but also reduces the reliance on extensive manual testing. This paper explores how this new approach can help electronics manufacturers more efficiently verify their designs and get products to market faster, with fewer errors.

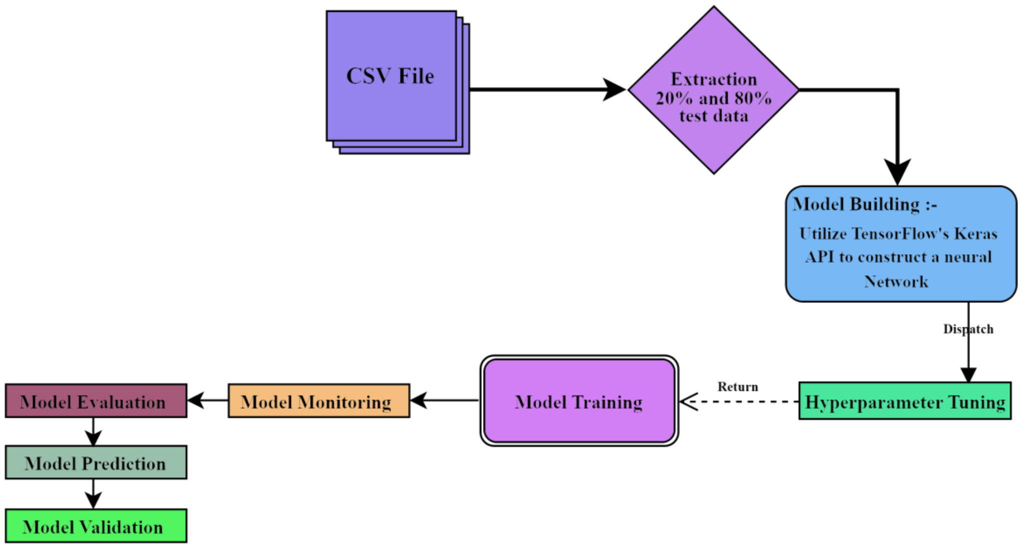


Figure 1: Process Flow: explaining data flow from simulation results to AI/ML model

II. REMODELING FAULT SIMULATION

In our recent simulation run, we focused on evaluating the robustness of our PCIe (a modified version developed in-house, with features of PCIe v2 and PCIe v3 with Gates estimated to be about 325K) design by validating Stuck-at faults, specifically Stuck@0 and Stuck@1 faults. The simulation was executed using traditional simulation tools, and the resulting data was compiled into a CSV format. Here's an explanation of the process and the significance of the results:

Fault Injection:

Stuck-at faults were generated on our PCIe design by forcing the signals to be logically high (1) and low (0). By using this method we collected and designed that are purely simulation results and are Y results by using traditional EDA simulators.

- Stuck@0 faults simulate conditions where a signal is fixed at 0, regardless of the intended logical value.
- Stuck@1 faults simulate conditions where a signal is fixed at 1.

These faults are injected into various nodes of the PCIe design to observe how the system responds under these fault conditions.

Simulation Process:

Using traditional EDA tools, we applied Stuck@0 and Stuck@1 faults at critical points in the PCIe design. Each simulation run generated a set of data reflecting the behavior of the system under these fault conditions. This data includes details such as signal integrity, timing discrepancies, and any resultant errors in the operation of the PCIe interface.

Data Collection and Refinement:

The raw data from these simulations often contains redundancy and inconsistencies, which can obscure meaningful insights. Therefore, we undertook the following steps:

1. **Redundancy Removal:** Duplicate entries and repetitive data points that did not contribute new information to the analysis were removed. This step ensures that the dataset is streamlined and focuses on unique fault impacts.

2. **Inconsistency Removal:** Inconsistencies, such as contradictory results or data points arising from transient simulation artifacts, were identified and rectified or removed. This step ensures the reliability and accuracy of the data

Here's how it can be applied practically:

1. Enhanced Design Verification: - Using a combination of fault simulation methods and AI analysis, this approach allows for quicker and more comprehensive testing of electronic circuits. Initially, all test vectors are used to simulate faults and capture data, which then helps an AI model learn to identify faults efficiently.

2. Efficient Model Training and Testing: - An AI model is trained using only 20% of the data collected, learning to predict faults accurately. This model is then validated using the remaining 80% of the data, ensuring it works effectively even with new changes in the design. This drastically cuts down the need for repetitive, exhaustive testing.

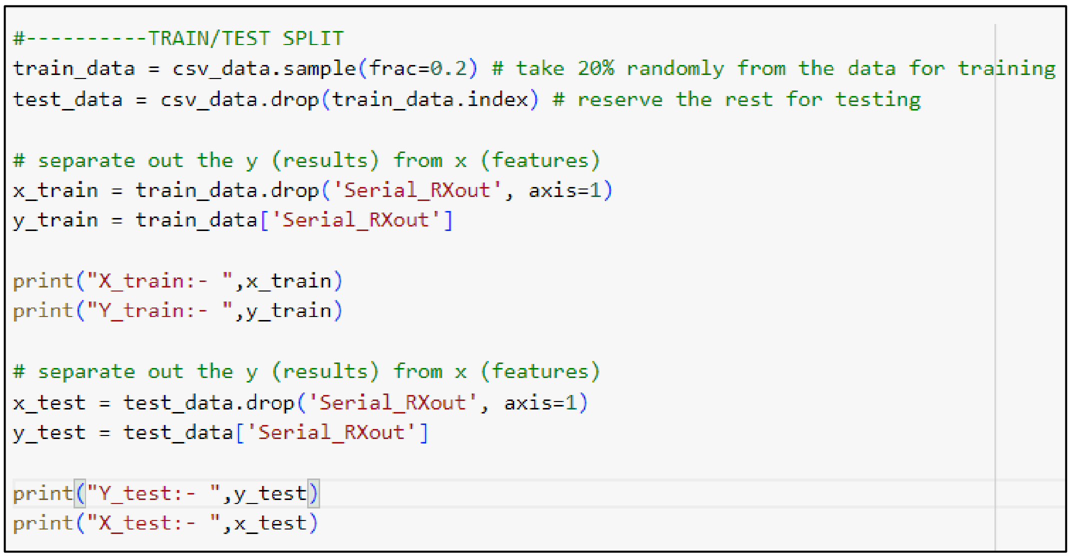
3. Faster Adaptation to Design Changes: - The AI model quickly adapts to changes in circuit designs, using less data to validate updates. This makes ongoing development faster and more efficient, a key benefit in fast-paced manufacturing environments.

4. Resource and Time Savings: - By predicting faults with a smaller set of test data, the need for full-scale simulations is reduced. This means quicker validation cycles, lower costs, and faster time-to-market for new designs.

III. AI/ML MODELING FOR FAULT SIMULATION

Using a Dataset for PCIe bus in the current papertheAI/ML Model has been developed with 5 hidden neural network layers using statistical and probabilistic functions from pandas and ML libraries, like TensorFlow function Keras and studying the effect of hyperparameters, optimizer, and activation functions. For this data set, we have the activation function "ReLU", "Sigmoid" and the optimizer selected "Adam" with a learning rate of 0.0001 furthermore the loss function "BinaryCrossEntropy" is applied to train data, and this model is fitted to train data, and the model is applied to test data for actual versus predicted values which are plotted on heat maps showing almost 98.3% accuracy.

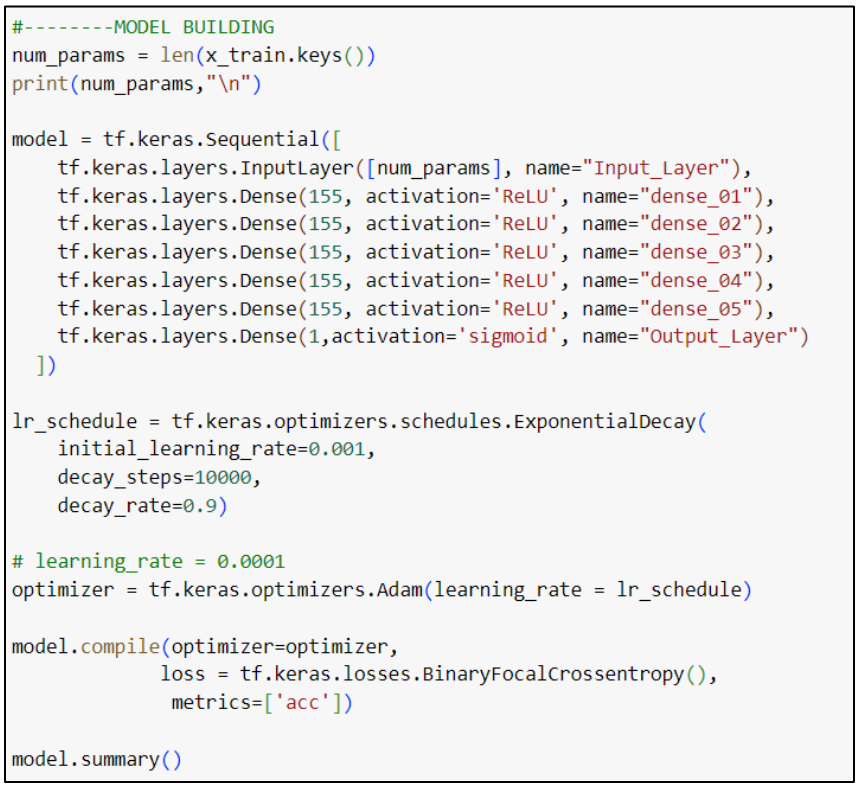
A) The dataset was split into 80:20 for train data and test data (this ratios may vary)
Further, the data in the test vector is sampled in 80% as test data and 20% as train data ratio.



```
#----------TRAIN/TEST SPLIT
train_data = csv_data.sample(frac=0.2) # take 20% randomly from the data for training
test_data = csv_data.drop(train_data.index) # reserve the rest for testing

# separate out the y (results) from x (features)
x_train = train_data.drop('Serial_RXout', axis=1)
y_train = train_data['Serial_RXout']

print("X_train:- ",x_train)
print("Y_train:- ",y_train)

# separate out the y (results) from x (features)
x_test = test_data.drop('Serial_RXout', axis=1)
y_test = test_data['Serial_RXout']

print("Y_test:- ",y_test)
print("X_test:- ",x_test)
```

Figure 2: Code for Train/Test Split Data

B) The regression ML model was built using TensorFlow ML libraries



```
#--------MODEL BUILDING
num_params = len(x_train.keys())
print(num_params,"\n")

model = tf.keras.Sequential([
    tf.keras.layers.InputLayer([num_params], name="Input_Layer"),
    tf.keras.layers.Dense(155, activation='ReLU', name="dense_01"),
    tf.keras.layers.Dense(155, activation='ReLU', name="dense_02"),
    tf.keras.layers.Dense(155, activation='ReLU', name="dense_03"),
    tf.keras.layers.Dense(155, activation='ReLU', name="dense_04"),
    tf.keras.layers.Dense(155, activation='ReLU', name="dense_05"),
    tf.keras.layers.Dense(1,activation='sigmoid', name="Output_Layer")
  ])

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.001,
    decay_steps=10000,
    decay_rate=0.9)

# learning_rate = 0.0001
optimizer = tf.keras.optimizers.Adam(learning_rate = lr_schedule)

model.compile(optimizer=optimizer,
             loss = tf.keras.losses.BinaryFocalCrossentropy(),
              metrics=['acc'])

model.summary()
```

Figure 2: Code for Model Building

IV. APPLICATION

The innovative approach we discuss in this paper, which combines traditional simulation with machine learning (ML) techniques in electronic design automation (EDA), is especially valuable for improving the efficiency and effectiveness of circuit design verification. Some areas:

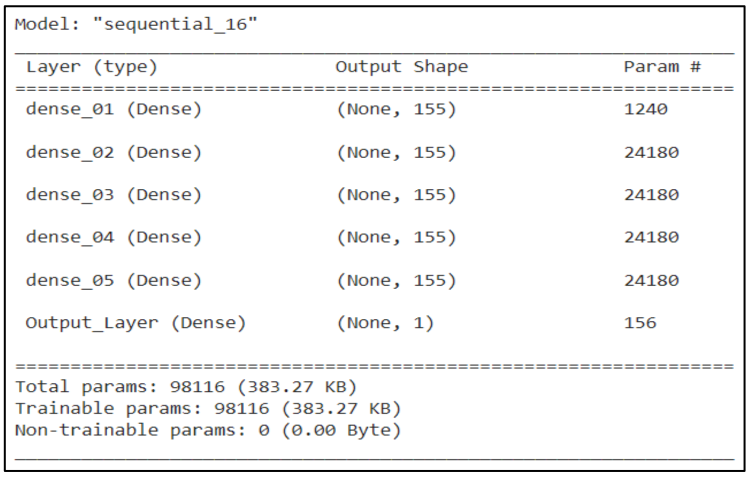
Following are some examples:

- In Automobiles, for example, Autonomous Vehicles and Electrical/Mechanical Vehicles need to be safe and in compliance with ISO26262 standards as per ASIL levels since human lives are at risk. Fault Simulation provides a method to ensure protection against Electrical and Environmental failures.
- In the Drones, Aerospace, and Space industry mission-critical programs need full proof and safety and no stone should be left unturned to ensure the multi-million programs do not fail.
- In the Health Industry, medical devices, such as pacemakers and insulin pumps, must meet strict regulatory requirements to ensure the safety and well-being of patients. Fault simulation can help identify potential failures in these devices before they are deployed, allowing for early detection and correction of design flaws.

V. RESULT

Our initial research explored integrating machine learning (ML) techniques with traditional Simulation to validate chip designs. We used a diverse dataset covering various fault scenarios, and the results were promising. Our AI-driven simulations significantly reduced the time required for fault simulation without compromising accuracy, demonstrating a marked improvement over the traditional method.

Use of the AI/ML model ((using activation function – ReLU/Sigmoid) and working with hyperparameters (epoch, accuracy, data loss, learning rate)) developed here on a standard PCIe DUT has demonstrated that it is possible to reduce time greatly and beat time-to-market forecasts by reducing the repeated, expensive and extremely time-consuming Fault Simulation tools used. The accuracy rate is 98.3%

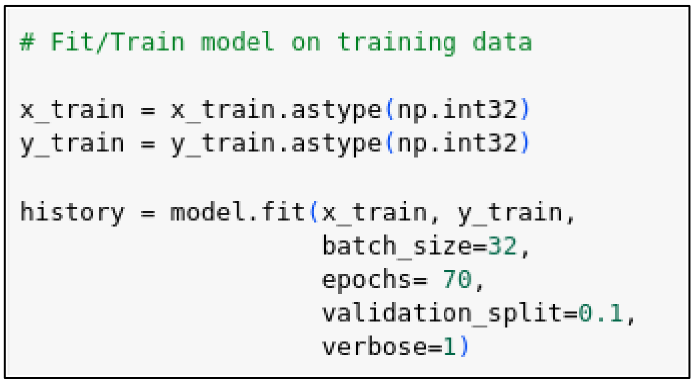


```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_01 (Dense)            (None, 155)               1240

 dense_02 (Dense)            (None, 155)               24180

 dense_03 (Dense)            (None, 155)               24180

 dense_04 (Dense)            (None, 155)               24180

 dense_05 (Dense)            (None, 155)               24180

 Output_Layer (Dense)        (None, 1)                 156

=================================================================
Total params: 98116 (383.27 KB)
Trainable params: 98116 (383.27 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 2: Model Building Result

Using Python, NumPy, pandas, TensorFlow, etc., and Colab from Google® the above stochastic behaviors were implemented by using a Data Set from Simulation Outputs for PCIe^{TM} Bus.

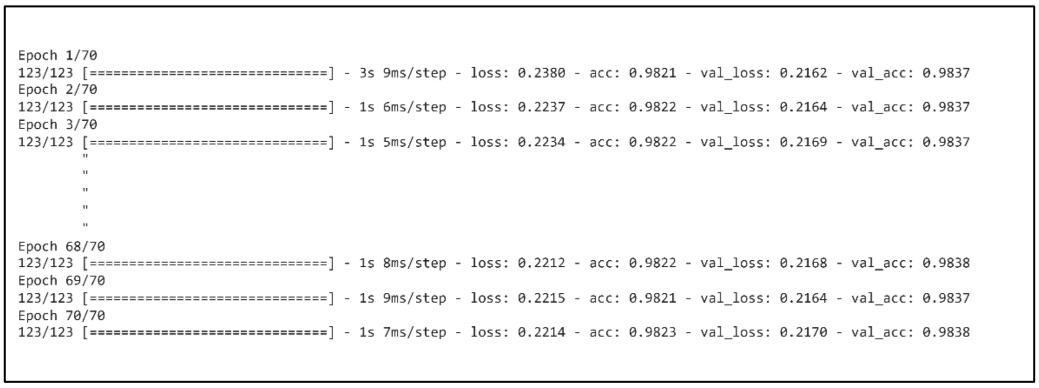
A) *We fit the model on train data over epochs runs.*



```
# Fit/Train model on training data

x_train = x_train.astype(np.int32)
y_train = y_train.astype(np.int32)

history = model.fit(x_train, y_train,
                    batch_size=32,
                    epochs= 70,
                    validation_split=0.1,
                    verbose=1)
```

Figure 3: Fit/Train Model

Using the model.fit function the train data ie. 20% sampled data is being mapped over train data. In this basically, the data is being evaluated based on accuracy, val_loss, and val_acc.



```
Epoch 1/70
123/123 [==============================] - 3s 9ms/step - loss: 0.2380 - acc: 0.9821 - val_loss: 0.2162 - val_acc: 0.9837
Epoch 2/70
123/123 [==============================] - 1s 6ms/step - loss: 0.2237 - acc: 0.9822 - val_loss: 0.2164 - val_acc: 0.9837
Epoch 3/70
123/123 [==============================] - 1s 5ms/step - loss: 0.2234 - acc: 0.9822 - val_loss: 0.2169 - val_acc: 0.9837
        "
        "
        "
        "
        "
Epoch 68/70
123/123 [==============================] - 1s 8ms/step - loss: 0.2212 - acc: 0.9822 - val_loss: 0.2168 - val_acc: 0.9838
Epoch 69/70
123/123 [==============================] - 1s 9ms/step - loss: 0.2215 - acc: 0.9821 - val_loss: 0.2164 - val_acc: 0.9837
Epoch 70/70
123/123 [==============================] - 1s 7ms/step - loss: 0.2214 - acc: 0.9823 - val_loss: 0.2170 - val_acc: 0.9838
```

Figure 3: Fit/Train Model_Result

B) *Data is being plotted against loss and epoch*

To monitor the Train data against the loss of information A graph is plotted to get clarity in graphical format.

```
#--------MONITOR
# Plot training & validation loss values
fig = plt.figure(figsize=(12,7))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validate'], loc='upper left')
plt.show()
```

Figure 4: Code to monitor data using matplotlib library

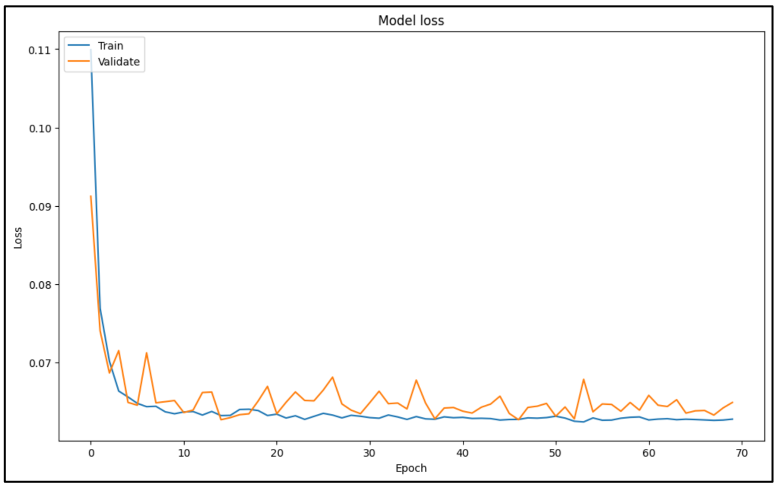


Figure 5: Model Loss

C) *At the last pass we get the heatmap as per the accuracy and the predicted data*

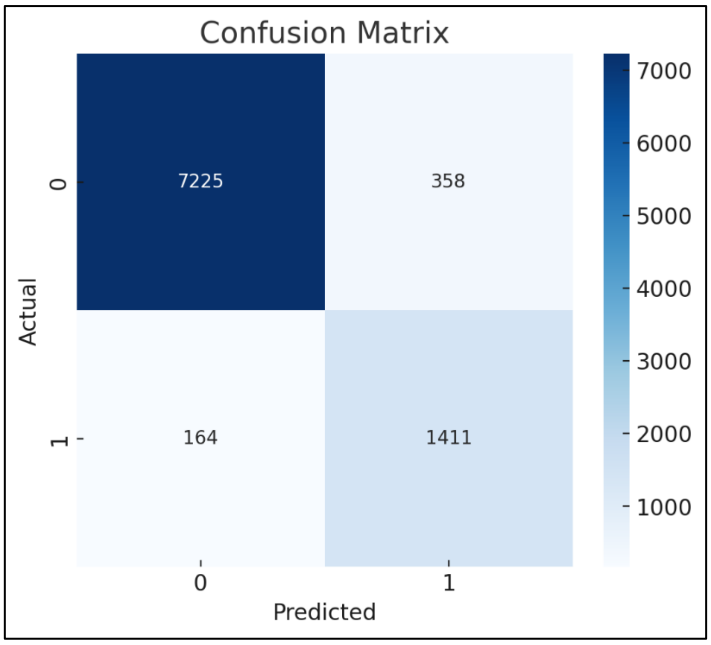


Figure 6: Actual and Prediction Heatmap

- True Negatives (TN): - The model correctly predicted the negative class (0).
  - Value: 7225
- False Positives (FP): - The model incorrectly predicted the positive class (1) for a negative instance (0).
  - Value: 358
- False Negatives (FN): - The model incorrectly predicted the negative class (0) for a positive instance (1).
  - Value: 164
- True Positives (TP): - The model correctly predicted the positive class (1).
  - Value: 1411

Moreover, the models displayed excellent adaptability, effectively managing different types of faults. These early findings suggest that our approach can streamline the fault simulation process, making chip design validation more efficient and cost-effective. This method could potentially accelerate product development cycles in the semiconductor industry, offering a versatile tool for a wide range of testing scenarios.

VI. Conclusion

Our study confirms that integrating machine learning (ML) with traditional fault simulation enhances the validation process for electronic designs. The AI model, trained on just a portion of the data, achieved high accuracy, efficiently predicting faults and adapting quickly to design changes. This method not only reduces the time and resources needed for extensive testing but also improves the overall effectiveness of fault detection.

Using visual tools like heat maps for performance assessment makes the process more intuitive and transparent. This approach represents a significant advancement in electronic design automation (EDA), indicating a promising future for AI-driven methods in streamlining and optimizing design validation. We recommend further exploration into this innovative integration to continue enhancing productivity and reliability in electronics manufacturing. Using simulation and AI/ML models to detect faults may alleviate the need for expensive, time-consuming fault simulators, and to that extent, this represents disruptive technology.

The limitation that may be considered is that the AI/ML model should not take significantly less time to give the results and the need to ensure that the AI processor usage is not becoming more expensive that the distributed parallel computers it is seeking to replace.

Future work should be to make one AI/ML model that can "fit" all designs and this may be undertaken by building the best-of-breed techniques to have a model that can be "generically" used but specific modules may still be required for complex designs.

References

[1] SiggraphMachineLearning&NeuralNetworkswithRajeshSharma(SIGGRAPHNow|Hands-on Workshop: Machine Learning and Neural Networks–Lecture1- YouTube) (https://youtu.be/gfY2LfRfE1E?si=TM3KDvTgWTOJeTPb)

[2] Functional Safety Verification Challenges for Automotive ICs| VerificationHorizons-July2022|VerificationAcademy. (https://verificationacademy.com/verification-horizons/july-2022-volume-18-issue-2/functional-safety-verification-challenges-for-automotive-ics/)

[3] Reducing simulation life cycle time of Fault Simulations using Artificial Intelligence and Machine Learning techniques on Big dataset ad DVCon Japan 2023

[4] Smart Testing: Integration Fault Simulation and AI/ML for Efficient IP Validation at 61st Design Automation Conference 2024

[5] Basic Architecture-of-an-artificial-neural-network (https://aiml.com/what-is-the-basic-architecture-of-an-artificial-neural-network-ann/)

[6] Activation-Functions (https://insideaiml.com/blog/Activation-Functions-In-Neural-Network-1089)