# A Novel AI-ML Regression Flow for SoC verification

## I. ABSTRACT

**As the development of chip design and verification goes hand in hand to meet Time-to-Market, every new feature implemented may cause failure to already passing tests. Common reasons could be, change in test bench, design, simulation environment. Over all turnaround time to debug a failing simulation in regression may take around a week of time due to manual effort involved in debug and slow simulations. As per analysis, over all verification effort can take up to 65% of whole design cycle. Especially Debug would take more than 50% of verification effort.**

**Until few years ago, there was no legitimate solution in the market which is intelligent enough to help reduce laborious and time-consuming steps in Design-Verification domain. Off-late AI is sensational in semiconductor sector, enhancing Design, Verification, bank-end process and fostering innovation. AI plays an important role in SoC Verification as the design size is huge and simulation time are considerably large. The turn-around time to debug a failure may take around a week of time. In regression suite of around ten thousand tests, it is going to take many days by many engineers to debug failures as engineers would debug one after another failures.**

**We have come up with a Novel Flow using AI-ML to analyze failure signature with the historical database and provide possible root cause analysis in the DV cycle. This flow consists of three main AI-ML engines called *CodeMiner*, *WaveMiner* and Defect Tracking Tool (*DTT)*. Each one contributes to reduce time in various possible debug cases like Repeated-failure, New-failure, new-feature. As this flow can be applied on multiple failures in parallel, over all debug time for regression can be reduced by around 10 to 20 folds, based on the complexity of the design. This flow is adaptable to IP, sub-system and SoC environment which is based on incremental design/DV process.**

## II. INTRODUCTION

Semiconductors are integral part of all modern electronics from smartphones to home appliances to complex computing systems. As chip designs become more complex by size and functionality, traditional methods are not sufficient to meet time to market targets. In the development process of a Chip, Verification effort is pretty high compared to design and sub-sequent steps. Any idea that helps to reduce verification effort would eventually help reduce overall Chip design cycle. The ultimate goal of verification is to find out bugs faster and reach coverage targets in short duration to meet product roadmap/time-to-market.

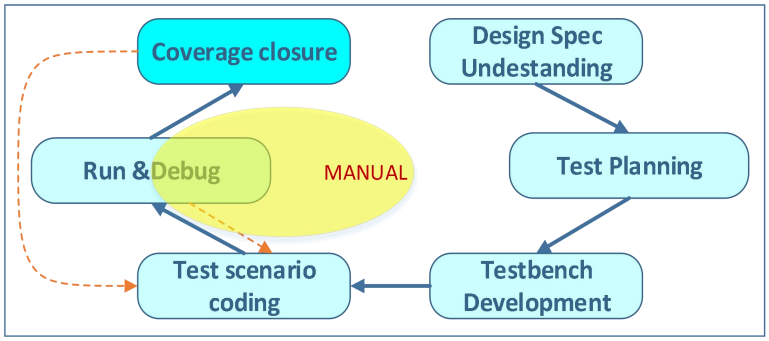


Figure-1 Conventional Verification Flow

To discuss more on verification stages, following are the steps involved in verification flow.

- Verification planning & review
- Testbench development / coding
- Running simulation
- Failures Debug & Analysis
- Review the metrics (Code coverage / Functional Coverage, **test** pass rate)

In above steps, 70% of time takes in debug and analysis. Though above steps remain same across IP, sub-system, SoC verification, Debug and Analysis step plays crucial role in SoC, as the simulation duration is pretty larger than that of IP/sub-system. Hence debug and fix time for each failure would be many folds larger than IP/sub-system. In above verification steps, some steps are backed up by tools to reduce the turnaround time. For example, **vManager** tools helps in developing Verification plan faster, while DVT **VSCode** can help in understanding and developing testbench faster. Unfortunately, Debug process is always manual right from the evolution of the chip industry. The area of focus in this paper target to address the same by utilizing AI-ML techniques to reduce the simulation debug times, hence reduces overall turn-around-time of SoC Verification.

Off late, Artificial intelligence is applied across many applications, platforms, industries. For example, AI based home security cameras, Traffic monitoring system, home appliances etc. Even in Semiconductor domain, AI can find its own way to help improve quality and turn-around-time. Especially, applying AI-ML yields better results in Verification as it takes pretty large percentage of overall chip cycle. Though Chip design & verification domain was not anticipated for AI while it started few years back, off-late chip design has been in focus. AI-ML is being considered for Specification to design, from design to verification, from verification to physical design and till to chip tape out. If AI can help us generate simulation results faster, engineer can focus on debug and find out more bugs.

In SoC verification, numerous tests are supposed to be run to cover each and every block and its cross communication channels. With conventional methods, it's difficult to debug all failures in a given time as simulation times are pretty large. Common issue may appear across multiple block –level simulations. Hence there needs a technique by which common issues and its resolutions are shared across blocks and end-to-end scenarios. In this case, **ML** technique provides a promising approach that is required to learn from past debug resolutions and apply to latest regression failures.

Since last few years, many EDA companies have been working on AI-ML and some of the tools / flows have been making rounds in the market. ***Verisium*** is one such successful tool. The Verisium Platform is a suite of applications leveraging big data and generative AI to optimize verification workloads, boost coverage, and accelerate root cause analysis of design bugs on complex SoCs. Verisium Platform uses AI on log files, RTL and test bench churn, revision control, and waveforms to accelerate and assist debugging, reducing manual effort by up to 32X [2]

Verisium accelerates the root cause analysis of design bugs, boosts coverage, and optimizes verification compute farm resources for complex SoCs. Verisium **CodeMiner** reduces the debug time by identifying the RTL logic that could be culprit for the failure. Verisium **WaveMiner** helps in finding Root-Cause by comparing Design wave dump of Pass and Fail cases. These two applications can achieve remarkable improvement in efficiency.

In house tool, called Defect Tracking Tool(***DTT***)], is a ML solution which mines the data from simulation Log, Jira/bug tracking information, failure report from older projects. This tool consists of three applications. **Abnormal Simulation Detector (ASD)** which is based on date from multiple linear regressions, to make expected run time. **Debug History Finder (DHF**), is a pattern matching technique, that is based on levenshtein distance algorithm, to

calculate similarity between errors. Last one, Debug Guide System (DGS), is based on algorithm - **BERT/XLNET/RandomForest**[8], to categorize simulation errors into 10 different categories.

## III. LITERATURE SURVEY

In this survey we provide a comprehensive investigation of the literature by examining how AI ML is utilized to enhance chip development processes, especially on SoC verification Debug time. Although AI ML is not new to Software domain, it is relatively new to Chip-design. Renowned EDA companies have been working on AI ML applications for quite some time. But the products are available to users only in past five years. To name some of them, Verisium platform by Cadence Design Systems, Synopsys.ai is suite of AI Apps by Synopsys Inc. There used to be some techniques to speed up the debug time of verification. Some of them are, auto-x detection for faster Root cause analysis, coverage based ranking to reduce regression time. But those solutions are based on the current data and simulation information. It doesn't consider past data and no learning experience taken into consideration.

Off late, EDA companies have invented AI-ML solutions that takes in to account of previous pass/ fail information and compare against current simulation. It does mining of both snapshots and figure out the Root-cause. These techniques can be applied to find out testbench issues as well [1][2]. *As those tools are business secretes, not much information is disclosed by Vendors*

To find the correlation of simulation failures, **BERT** algorithm [8] is used in DTT tool. The first step in this process is to classify the failure signature. Text classification is a subfield of ML, that discuss about classifying text into different categories. It's commonly used as a supervised learning technique, which means that the algorithm is trained on a set of texts that have already been labelled with their respective categories. Once it's been trained on this data, the algorithm can use what it's learned to make predictions about new, unlabelled texts. The algorithm looks for patterns in the text to determine which category it belongs to.

## IV. PROBLEM DEFINITION

As described in Introduction, Regression debug is challenging and time consuming task in verification, due to huge simulation timings in SoC as its large in design, complex in logic. As the features are still under development, any new addition or update, like adding new IP, enhancing the IP, adding new features can be done until last stage of verification. Any such update, would cause a failure in already Passed simulations. Engineer may have to spend time in debug again and again though the same test was Passed in previous Tag. Note, Debug is a **manual** task, that involve in generating wave dump(re-run), pull out required signals, trace drivers, until Root cause is identified. This is an inefficient way of verification! Hence we come up with an AI-ML flow to accelerate Debug task hence reduce turn-around time of Verification.

## V. PROPOSED AI-ML FLOW

The primary goals of using AI technologies are to reduce costs, save time, improve quality. The proposed flow is going to address in reducing debug time, hence accelerate over all SoC verification. This flow is an intelligent, scalable and adoptable that can fulfill the following motivations.

- Reduce or avoid Debug time
- Find out the root cause
- Reduce Overall turnaround time of debug
- Scalable to IP, Sub-system, SoC
- Adaptable to various products

Proposed flow is based on three AI/ML flows. One solution called **DTT** is in-house tool, while other 2 are from cadence, under Verisium Platform – **CodeMiner** and **WaveMiner** from Cadence. The Verisium Platform is a suite of applications leveraging big data and generative AI to optimize verification workloads, boost coverage, and accelerate root cause analysis of design bugs on complex SoCs. **Verisium CodeMiner** provides an algorithmic solution to compare multiple source code revisions of an IP or SoC, classify these revisions, and rank which updates are most disruptive to the system's behaviour to help pinpoint potential bug hotspots. **Verisium WaveMiner** Applies powerful AI engines to analyse waveforms across a full verification test suite and determine which signals, at which times, are most likely to represent the root cause of test failures. As described in Introduction, Defect Tracking Tool(***DTT***) an in-house tool, is a ML solution that register the failure information along with solution. If he same error is encountered by other user, it displays the solution to be applied.

The ***Figure-2*** depicts the Flow diagram involving building blocks of the proposed AI-ML flow. in SoC Verification as the engineer has to work on developing new features alongside. The flow involves 3-phases called Planning, Execution and Debug. In the planning phase, Test plan consists of **N** number of tests, its Passing status and the corresponding Tag in which it passed for the first time. In the Execution phase, regression is run as per the verification strategy and the results are published. Its noticed that test T2 has failed in the latest Regression on tag (N+1). The same test got passed in Tag (N-1) In the debug phase, T2 is going to debugged by using AI-ML Debugger, that leverages three Apps, called **Defect Tracking Tool** (DTT-In-house tool), **CodeMiner** (Verisium app), **WaveMiner** (Verisium App). Once the failures if fixed, Pass Tag is updated in ML Database.

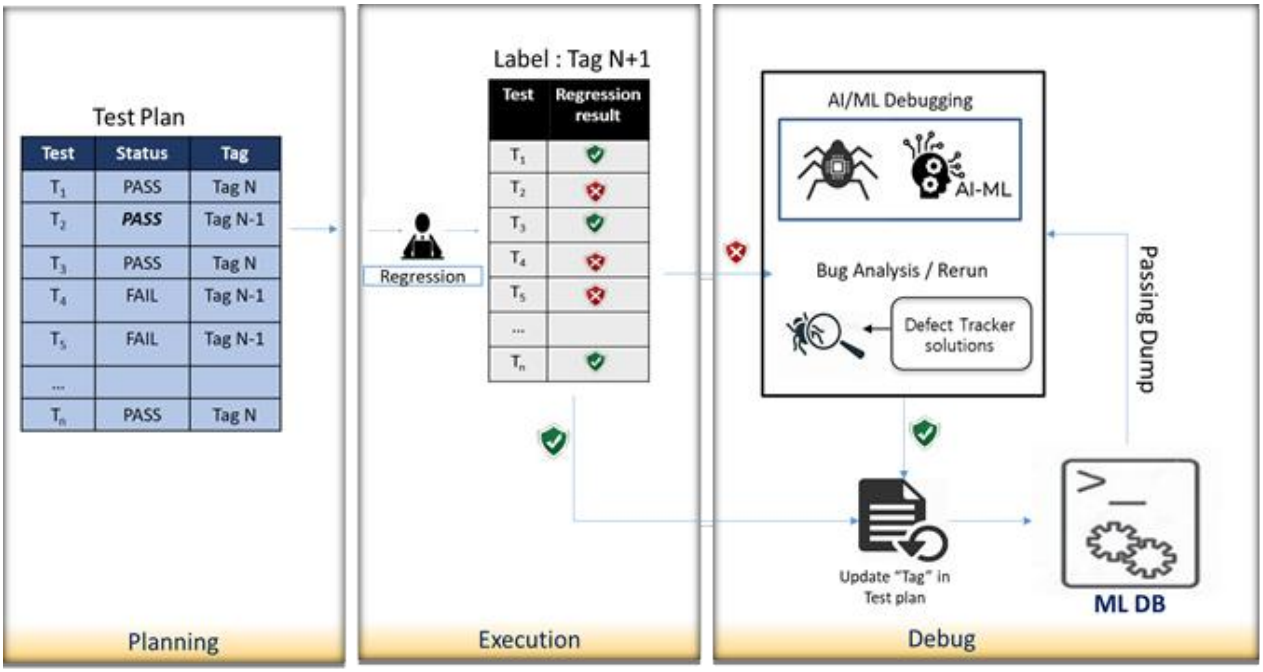


**Figure-2**: Block diagram of AI-ML Flow

Figure-3 explains the flow and steps involved in failure debug.

1. Regression manager launches the regression of tag (N+1)
2. If the test is Passed, current tag is updated in ML Database.
3. If the simulation has a Failure, the following cases are analyzed.

a. If it is Old / known failure, for which **DTT** has a solution registered, the same get applied without user intervention and given re-run.

b. If its New failure, it can be all together new test (without any past history) OR it cas got Passed in the old tag, but encountered a new Error signature.

   i. Failure with History: this is debugged with the help of **WaveMiner** that generate the wave dump and compare the signal at each time event and find out the Root-cause.

   ii. Failure with NO-history, is going to be debugged by the **CodeMiner**, that compare RTL snapshot of previous Tag and Current tag, to help localize the issue and provide the possible root causes in Raking order. Then User has to involve and do further debug to find out the exact root case.

   iii. In any case, once the test is fixed, tag is updated in ML database.

AI-ML method yields attractive results if number of failures are more.

1. In traditional method the debug process is manual and iterative, and debug TAT becomes exponential as the number of failures increase.
2. In AI-ML debug process, many steps in debug cycle are automated. In regression flow, RCA of all the failures gets generated without manual intervention, as described in Figure-4.

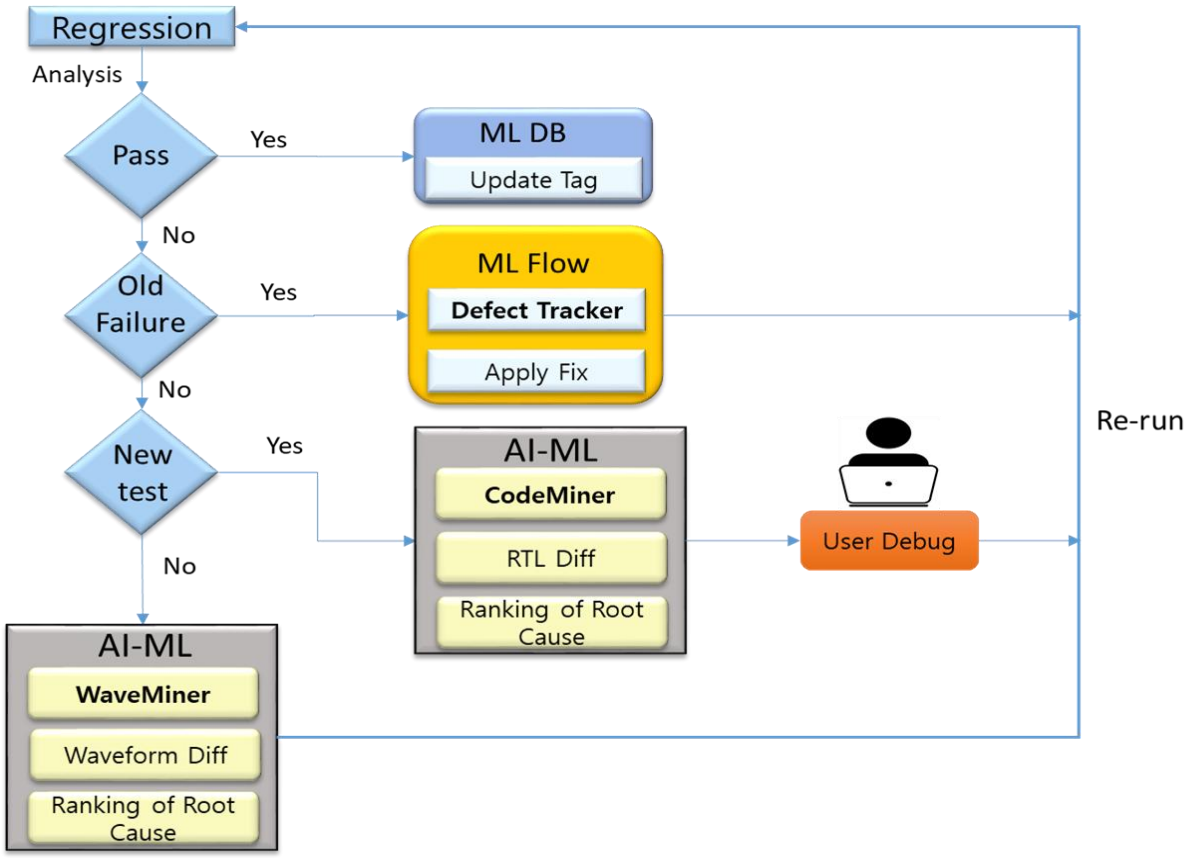


**Figure-3**: Flow chart of AI-ML Debug

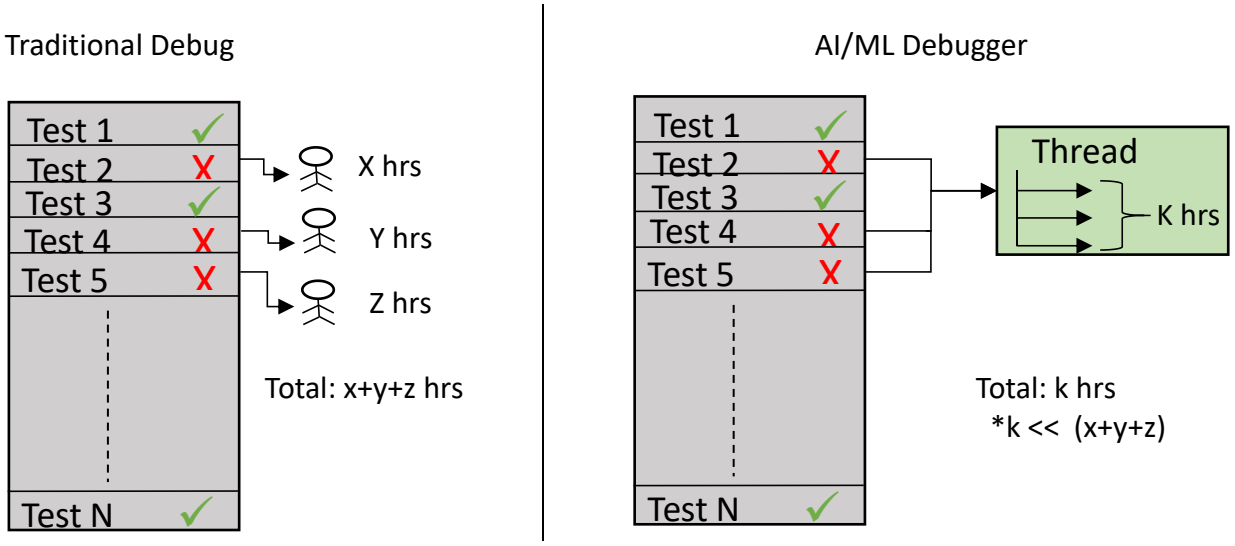


**Figure-4**: Traditional vs. AI-ML debug approach

## VI. APPLICATIONS

This methodology can be applied at various verification approaches as mentioned below

1. Debug multiple failures simultaneously
2. Easy deployment across any DV flow
3. Adaptable to IP/SoC/sub-system level

**Debug multiple failures simultaneously:** Assume there are multiple failures one user has to debug. In conventional way of debug, user will debug one by one, in sequential manner hence over all debug time of a project is tremendously large. If AI-ML flow is applied, failures are debugged in parallel (if required number of licenses are available), hence over all turnaround time of debug of a project gets reduced by many folds. It is observed that over all savings using AI-ML flow is 20 times faster than conventional debug.

**Easy deployment across other DV flow:** AI-ML flow can be applied across various DV flows like IP, Sub-system, SoC. Time taken to generate snapshot and code / wave miner results are very quick in case of IP, due to its small design size which improves TAT. Whereas for SoC, due to its heavy size, snapshot creating and mining would be little slower compared to IP.

**Adaptable to IP/SoC/sub-system level:** This AI-ML flow is based on RTL and TB code, which involves Code Miner and Wave Miner. Code Miner is used to correct RTL code automatically, which are causing failures without manual intervention. Similarly, Wave Miner also fix the bugs, which are repetitive. This will improve the TAT and reduce the manual efforts in turn can be applied easily from IP to SoC level without any difficulty.

## VII. RESULTS

In this section, testbench simulation environment is explained in detail like test access, complexity, followed by improvements in simulation Debug times using AI-ML Flow.

**Flow-Test plan:** We have considered two categories of tests. First one is of basic scenario, like register programming, connectivity check. Other one is complex test, like core scenario involving multiple blocks, multiple transaction. For example, accessing DRAM back to back. Initiating a camera system to capture picture and process it in Image Signal processor., complex test. These 2 categories of test are asked to debug manually by junior engineer with less than 5 years of experience, a senior engineer over 10 years, and a new-joiner with around 7 years of experience. Each one of them are asked to debug same test. The same is repeated for multiple test in each category of test.

**Debug time comparison:** Considering the above plan, we have collected the debug time for each case. Gain in overall Regression is as high as 20x gain, due to the reason that this flow can debug failures simultaneously, whereas engineer has to debug one after another.

**Results are detailed in the Table-I and II below.**

| Test Type | Debug effort(in hours) | | | | Total Gain ( in %) |
|---|---|---|---|---|---|
| | Low-level experience | Debug Mid-level experience | High-level experience | AI/ML Flow (in Min) | |
| Register Access Test | 3 | 2 | 1 | 10 min | ~91.6 |
| Block-specific | 4 | 2 | 1 | 30 min | ~78.6 |
| Power test | 5 | 3 | 2 | 45 min | ~77.5 |

TABLE I: Simulation Turn-around time

| TAT Metric | Gain over Traditional approach** |
|---|---|
| Debug time of simple test (Register access) | 2-10x |
| Over all TAT of simple test (debug + re-run + multiple iterations) | 2-5x |
| Debug time of a complex test | 2-5x |
| Over all TAT of Complex (debug + re-run + multiple iterations) | 2-20x |
| Regression TAT | 2-20x |

** *Based on skill set of an average engineer*

TABLE II: Simulation Turn-around time

## VIII. FUTURE SCOPE

The future of AI in the semiconductor industry is promising. As technology continues to evolve, we can expect more sophisticated AI applications to emerge. Our flow will get equipped with a new application called **DVChatBoat**. It does provide not only basic information to code or debug faster, also helps engineer to learn more about the logic and functionality of each feature. As like general Chatboat, this app contains reference to documents, list of frequently caught issues and steps involved to achieve a task, user guide to commonly used task and so on. It

## IX. CONCLUSIONS

Employing AI-ML Flow for debugging failures using a data set of previous regression simulation results presents a promising approach to enhancing the efficiency and effectiveness of debug process in hardware design verification. It assures re-debug of old or known failure is avoided and user can focus on real RTL bugs, instead of

re-inventing the wheel. The flow does accelerate verification cycle by leveraging industry standard AI-ML Apps to reduce turnaround time of debug and is scalable across various DV domains ranging from IP to SoC.

## REFERENCES


[1] Robert Ruiz, Taruna Reddy, “SoC Design Verification and Chip Debug with AI ” , Synopsys Inc.

[2] Cadence, “Verisium AI-Driven Verification Platform”, Cadence Design systems, Company website information.

[3] Achutharaman, R. (2021). “Trends Accelerating the Semiconductor Industry in 2021 and Beyond”. Available online at: https://blog.appli edmaterials.com/trends-accelerating-semiconductor-industry-2021

[4] Göke, S., Staight, K., and Vrijen, R. (2021). “Scaling AI in the sector that enables it: Lessons for semiconductor-device makers.” Available online at: https://www.mckinsey.com/industries/semiconductors/our-insights/scaling-ai-in-the-sector-that-enables-it-lessons-for-semic onductor-device-makers

[5] “Semiconductors and Artificial Intelligence” .” Available online at: https://irds.ieee.org/topics/semiconductors-and-artificial-intelligence

[6] Kiran Vittal, “SoC Design and Verification Solutions for a New Era of AI Chips”, Synopsys Inc.

[7] S. Bhattacherjee, D. Shah and D. Pal, "Deploying Artificial Intelligence in Design Verification to Accelerate IP/SoC Sign-off with Zero escape"

[8] Khang Pham. Text Classification with BERT. Blog Available from: https://medium.com/@khang.pham.exxact/text-classification-with-bert-7afaacc5e49b