Faster Test Development through Accurate Digital Twinning of Loadboards

Shrey Gupta

Faster Test Development through Accurate Digital Twinning of Loadboards

by

Shrey Gupta

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday October 31, 2024 at 10:00 AM.

Student number:	5723698	
Project duration:	July 17, 2023 – October 31, 2024	
Thesis committee:	Dr. Ir. M. Taouil,	TU Delft, supervisor
	Dr. Ir. M. C. R. Fieback,	TU Delft, co-supervisor
	Dr. Ir. Qinwen Fan,	Tu Delft
	Dr. Ir. G. C. Medeiros,	NXP Semiconductors

Abstract

The increasing complexity of *Integrated Circuits* (ICs) and stringent quality requirements in industries like automotive have contributed to increasing test development time. Virtual testing methodologies, such as AMS-VT, can be used for pre-silicon debugging of test programs, optimizing the test development process. However, AMS-VT has shown limitations in their usage due to the absence of accurate loadboard models, especially in capturing site-to-site variations. To address the lack of an accurate loadboard model for pre-silicon debugging, this research proposes a framework for generating an accurate loadboard model by integrating parasitic effects and possible reflections under real-world conditions.

The proposed framework consists of three main stages. The first stage is an automated netlist generation phase, during which an ideal single-site netlist is created from a multisite loadboard schematic. The ideal netlist is generated considering the AMS-VT environment so that it can be seamlessly integrated into the current virtual testing environment. In the next stage, Cadence PowerSI, which is an *Electronic design automation* (EDA) tool for parasitic extraction, is automated to extract the parasitics (R, L, G, and C) for all the channels included in the ideal netlist. Finally, in the last stage, a methodology is developed to integrate the parasitics and reflection due to impedance mismatch in the ideal netlist. To improve the accuracy, a static model of the relay was developed that considers its characteristics, such as on-state resistance and off-state current leakage. All these stages are then automated and combined to create a framework that can effectively generate an accurate loadboard netlist that predicts site-to-site variations.

The framework is then validated, with each stage verified independently. The generated ideal single-site netlist was verified against a golden simulation. This verification displayed that the functionality of the generated netlist was accurate and could integrate with AMS-VT seamlessly. The parasitics extracted from PowerSI were validated using a test *Printed Circuit Board* (PCB) with multiple configurations. This showed that the simulation results using extracted parasitics were within 1% error compared to the IR drop analysis. The relay model was also validated by comparing it with its datasheet, proving that the model accurately considered the physical characteristics. The third stage was then validated by comparing it with PSpice simulations, demonstrating that the model's results were accurate, with the maximum error being below 1.6% for multiple voltages and frequencies. The complete framework was then validated against a physical loadboard, showing maximum errors of less than 2% across all sites and improved timing accuracy for pulse width, reducing it to 0.41% for the proposed framework from 1.6% for the previous methodology. This validation proved that the loadboard generated by the framework accurately modelled parasitics and reflections. The model also accurately accounted for temperature variations, demonstrating its effectiveness in changing operating conditions. By introducing this framework to generate an accurate loadboard model, this research enables a virtual testing environment for faster and more reliable test development, reducing the time-to-market for semiconductor products.

Acknowledgement

I would like to thank the following people, without whom I would not have been able to complete my thesis:

- First and foremost, I would like to thank my parents, whose support has been significant in my personal and academic journey. They stood behind me even when I made wrong choices and helped me to stand back up after failures. I would like to thank them wholeheartedly for all the effort they have put into me throughout their life, even when they were going through difficult times.
- Second, I would like to thank my supervisors at TU Delft, Dr. Ir. Mottaqiallah Taouil and Dr. Ir. M.C.R. Fieback, who provided me immense support during the entire course of my thesis by providing me valuable feedback on the work and helping me to look at things critically. The weekly discussions allowed me to evaluate the work and improve further to achieve better results.
- I would like to express my sincere thanks to my supervisor at NXP Semiconductors, Dr. Ir. G.C. Medeiros, for his mentorship and guidance throughout my time at NXP. He gave me insights and knowledge on virtual testing and guided me throughout the project. I would like to thank him for answering all my questions, even if they were out of context, and giving me the knowledge to complete the project and start my career in this field.

Shrey Gupta
Delft, October 2024

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The thesis delves into innovative strategies in semiconductor testing, mainly focusing on the accurate digital twinning of loadboards and virtual testing for faster test development. Section 1.1 presents the motivation behind the thesis and the importance of virtual testing. Then, Section 1.2 presents the state-of-the-art in virtual testing. This section also introduces the limitations in the current methodologies that need to be overcome to achieve accurate virtual testing. Section 1.3 presents the scientific contributions of this thesis. Finally, Section 1.4 presents the outline of the thesis.

1.1. Motivation

Integrated Circuits (ICs) have revolutionised the world of technology by enabling miniaturisation and increasing efficiency in electronic devices [1]. Due to the increasing demand for electronic devices needed to manage innovations worldwide, the demand for ICs has become numerous in recent years [2]. This trend is reflected in statistics from *World Semiconductor Trade Statistics* (WSTS), which report that the semiconductor industry's revenue grew by 16% in 2024 compared to the previous year due to the rising demand for ICs [3]. Figure 1.1 shows that the semiconductor industry's revenue significantly increased from 1987, when it was around 30 billion U.S. dollars, to around 611.23 billion U.S. dollars in 2024.

As the world is seeing increasing demand for ICs, the requirements of the consumers related to the quality of the chips are also becoming tighter [4]. These chips are used in critical sectors such as the automotive industry, where a bad-quality IC can result in disastrous consequences. The industrial term for defining the quality of a given product is *Defective Parts Per Million* (DPPM), which refers to the number of faulty chips sold to the customers in a batch of a million chips [4]. Due to the increasing number of electronic components in recent years, automotive companies like BMW have stated that even with 1 DPPM, they expect one car failure every hour [5]. This has resulted in a much more stringent quality requirement for automotive semiconductors, i.e., from DPPM to *Defective Parts Per*

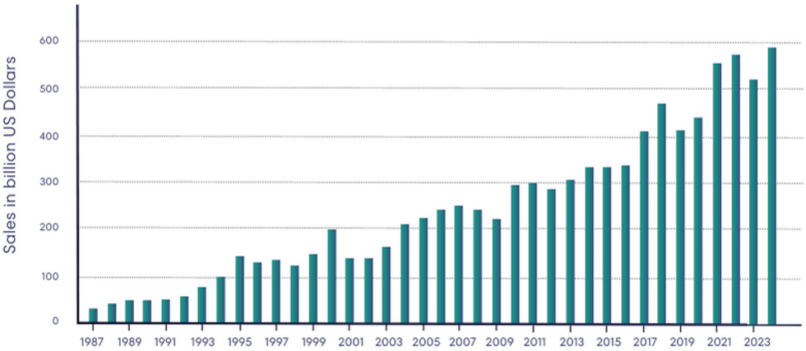


Figure 1.1: Semicondcutor Market Revenue from 1987 to 2024 [3]

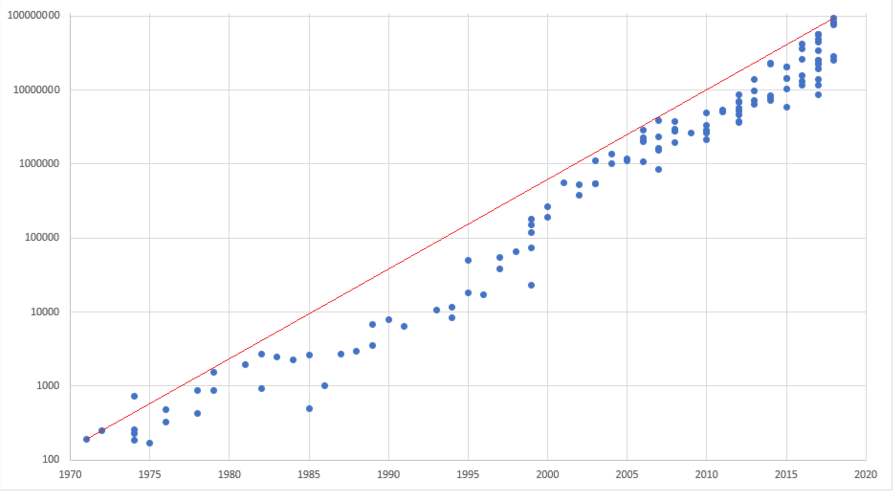


Figure 1.2: Transistor per Semiconductor Chip [9]

Billion (DPPB) level [5]. Due to the high-quality standards set by the consumers, the expected quality of the ICs is increasing. The quality of an IC is determined by testing, where it is determined if every manufactured semiconductor is functioning within the specifications [4]. The testing strategy is to develop suitable test plans while considering both quality and economy, that is, to detect most errors at the lowest cost. Depending on the product type and test requirements, the cost of testing accounts for 10–30% of the total IC cost [6].

Over the years, the complexity of IC has also increased sharply. As shown in Figure 1.2, transistor density in ICs has steeply increased over time, indicating that the complexity of chips is continuously rising. This trend aligns with Moore’s law [7], which predicts a doubling of transistors on a microchip every two years. Due to the increasing complexity and quality requirements of ICs, there is an increase in the test development time of a chip, which then contributes to increased time-to-market [8], which means the total length of time it takes to bring a product from conception to market availability. The test development for an IC requires a significant amount of time depending on the complexity, which can delay the product release by even weeks. To stay competitive and meet customer demands related to product delivery, it has become crucial for manufacturers to optimise the test development time to reduce time-to-market. Efficient test development can enhance the manufacturer’s productivity, allowing them to deliver innovative products more swiftly [8].

Testing is a complex process that involves several components such as an *Automatic Test Equipment* (ATE), Test Program, Loadboard, and the DUT. The test program is used to automate the test process, and its development depends on the DUT and the capabilities of the chosen ATE platform [8]. Its development can be conducted in parallel with the design of the DUT. However, verifying whether a test program properly conducts the intended tests is challenging before the first silicon, when the first IC is produced, is available [10]. Certain “sanity checks,” such as ensuring the correct syntax and configurations for the ATE, can be carried out beforehand. However, comprehensive checks and full validation can only be performed after the first silicon [8]. As shown in Figure 1.3, a test program development starts during the design stage and is completed pre-silicon. However, the test program debugging begins after the first silicon in the current scenario. If the test program can be debugged pre-silicon, the test development time can be optimised, decreasing the time-to-market. By using a methodology known as virtual testing, which means using virtual models of testing components to run a simulation and predict the behaviour of the test program, the test program can be debugged pre-silicon. This pre-silicon debugging of the test program using virtual testing in the test development flow speeds up the start of production testing and contributes towards time-to-market speed-up [8].

1.2. State Of The Art

Several methods have been proposed for virtual testing in the test development flow. Section 1.2.1 introduces them and describes how they are used. Then, Section 1.2.2 presents the limitations of these methodologies.

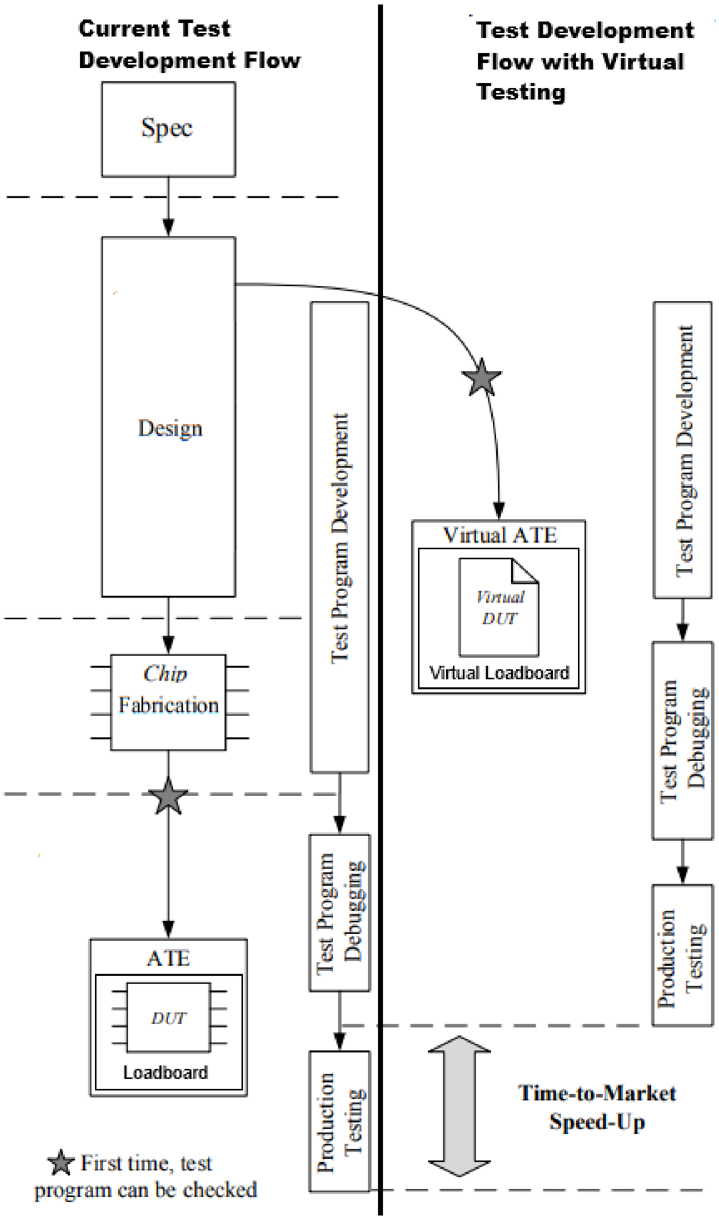


Figure 1.3: Normal Flow vs Virtual Testing FLow [8]

1.2.1. Virtual Testing Methodologies

The need for software in testing was already recognised in 1996 by Mielger and Woltz [11]. They introduced a software called VTB that allows the definition of signals with the help of pre-defined models of the test programs. This was further developed as a simulation tool known as VTest that can be integrated for proper circuit simulations [12]. VTest allowed the engineers to create a platform for designing and simulating test requirements using DUT models. It allowed design and test engineers to communicate by introducing a generic library that enables data sharing. This generic library further allowed the sharing of models between the design and testing stages and opened the door for simulating the DUT using test requirements. VHDL, a *Hardware Description Language* (HDL), was used to create a high-level netlist for the components needed for the simulation [12]. This allowed the test engineer to work sooner than they were doing by integrating the design and testing environment, and they could use VHDL models of the components to review the design and provide feedback to designers. This paved the way for test development to begin even sooner as they could access designer files and requirements, giving them more time to make a robust test program.

A simulation approach was proposed to verify and test a system before making the hardware [13]. After determining the design and requirements, a schematic was generated for the components. After this, tests were developed based on the device's requirements. These tests were then simulated to check for device performance errors. If errors were detected during the simulation, the issues were sent back to the designer for verification. Following this approach, an IC could be virtually tested for specific parameters by simulating the models. In this approach, certain complications were found, such as the absence of accurate simulation software and accurate models. This demonstrated that the DUT model generated by designers could effectively be simulated based on the testing requirements. So, a method was needed to convert test programs made by test engineers in the form of triggers, also called stimuli, which can then be used to simulate the DUT models. Using this, the behaviour of test programs can be predicted, and they can be debugged before silicon.

A virtual environment was then created that could combine the models of DUT and ATE and use test programs as stimuli to trigger voltages in the pins [14]. Using this environment, the test engineers can efficiently run simulations to design, debug and verify all test-relevant processes [14]. The virtual testing environment uses an operating system (Linux) to maintain interoperability between the *Electronic Design Automation* (EDA) tools and test system, but this requires digital twinning [15] of DUT, Loadboard, and ATE.

This new approach was successful as it solved one of the major issues: to make an integrated environment for all models and libraries. Introducing a virtual testing environment allowed companies such as Infineon Technologies to research ways using which virtual testing can be integrated into the existing test development flow [10]. The chip was modelled as a *Register Transfer level* (RTL) code, and stimuli were applied to verify how the DUT would behave under a specific test program. Infineon integrated virtual testing for pressure sensors and found that it reduced the test debugging efforts of test programs after the silicon. A digital twin of the chips was used with stimuli parallel to the chip design, which reduced the test development time. The main advantage of this is pre-silicon test program debugging, which means a more robust test program can be obtained before silicon.

Since virtual testing saves test development time, the need for accurate simulation models became even more significant. The focus moved towards accurate digital twinning of the required testing components: ATE, DUT, and the loadboard [16]. ATE manufacturers such as Testinsight started providing virtual ATE that converts the test program files into an ATE-aware Verilog model that emulates how ATE drives the DUT [17]. This virtual ATE allows pre-silicon testing by mirroring the behaviour of the physical ATE. It makes production test debugging shorter and more predictable by ensuring that tests are ATE-compatible, identifying issues in converting test patterns from design to ATE format, and allowing attention only to silicon.

After having an accurate model of ATE, designers then developed the DUT model in Verilog to communicate with the virtual ATE, and by using these models, the test program debugging was possible before the arrival of the first silicon, as shown in Figure 1.3. The virtual testing methodology was also compared with the actual ATE results for memory tests using the same test program [8]. For the test, the registers were first set to 0, then checked, and repeated after setting the registers to 1. This confirmed that precisely the same outputs were generated by both physical ATE and virtual testing except for the run-time since virtual testing takes more time [8].

In recent years, NXP has been working to integrate a virtual testing methodology called *Analog/Mixed-*

Signal Virtual Tester (AMS-VT) into their test development flow. AMS-VT, developed by TestInsight, is a mixed-signal virtual ATE that enables pre-silicon validation by emulating the actual tester environment in an EDA simulation using cadence tools [18]. It allows test engineers to debug their test programs and loadboards before *Tape-Out* (TO) when ICs are sent for industrial manufacturing, significantly reducing development time and improving confidence in the test setup. AMS-VT offers several key advantages that are [18]:

- Emulate the ATE tester.
- Predict the post-silicon test results.
- Faster test program debugging using EDA tools.
- Pre-silicon validation of loadboard design and test program.

The integration of AMS-VT into NXP's *New-Product-Introduction* (NPI) process has enabled the test engineers to perform early simulations with the entire test setup, i.e., the DUT, the loadboard, and tester instruments, and thus obtaining early knowledge of the product under development. It is also expected that AMS-VT can significantly reduce development time, leading to a shorter time-to-market for new products [18].

1.2.2. Limitations

The previous section illustrated the virtual testing methodology over the years and proved that employing virtual testing in test development flows results in a faster test development process. After using virtual testing and comparing it with the physical ATE, it was found that the results for memory testing were the same [8]. While AMS-VT used by NXP is the most recent and advanced virtual testing methodology, representing significant advances in the test development process, certain limitations still constrain it. One of the critical limitations of the current AMS-VT approach used by NXP is the inaccuracy of the loadboard model. The loadboard model currently used is an ideal netlist in Verilog AMS, which is manually created. This manual process is prone to human errors. It does not accurately reflect the physical characteristics of the loadboard, such as parasitics, which means the loadboard cannot be debugged pre-silicon.

Furthermore, a loadboard contains multiple sites, where one site means one DUT being tested, so it is vital to account for site-to-site variations for debugging the test program and the loadboard using virtual testing. However, AMS-VT cannot evaluate how the test program will behave across multiple sites as it can only simulate one site at a time [18]. To account for these variations, a loadboard model is needed to predict the site-to-site differences. Still, currently, the loadboard model being used is an ideal netlist that cannot account for the site-to-site variations. This limitation is critical because voltage differences and parasitic effects can vary from site to site, potentially leading to discrepancies not accounted for in the ideal netlist. Due to these significant limitations, AMS-VT can only debug certain aspects of the test program, such as ATE-DUT connectivity and the basic flow of the test program; it cannot thoroughly debug the test program in terms of voltage levels, signal integrity, or site-to-site variations. These limitations highlight the need for more accurate digital twinning of the loadboard. Without accurate digital twinning, the results remain idealized, missing critical details that can impact the final product's performance.

To overcome these limitations and improve the accuracy of virtual testing using AMS-VT, the following requirements for the loadboard modelling framework are proposed:

- The loadboard model should be single-site as this aligns with the current capabilities of AMS-VT. However, the model must be highly detailed to capture all relevant parasitic effects and signal integrity issues at that site.
- The loadboard should be in Verilog AMS to ensure compatibility with existing AMS-VT methodologies and EDA tools used in virtual testing.
- The framework to generate the model should be generic to accommodate different projects and loadboard designs.
- The loadboard model must be highly accurate to emulate the site-to-site variations so that the test program results can be predicted on different sites.

1.3. Thesis Contribution

This thesis addresses the critical limitations identified in the current AMS-VT methodologies employed by NXP, particularly the need for an accurate digital twin of the loadboard. By focusing on the accurate modelling of loadboard characteristics and their integration into the AMS-VT environment, this research makes the following contributions:

- Introduces a framework that automates the generation of a highly accurate virtual model of the loadboard by including its physical characteristics.
- The framework generates a unique model for each site, emulating the site-to-site variations due to the parasitics, allowing the test engineers to predict the behaviour of the test program on different sites.
- The framework seamlessly integrates the generated loadboard model into the AMS-VT environment, allowing for precise simulation.
- A loadboard model that predicts the effects of real-world operating conditions, such as frequency and temperature, on voltage and timing.
- For increased accuracy, a static relay model is developed to emulate its parasitic effects on voltage and timing.
- Provides a repository containing all the developed models and scripts. The code repository is accessible at [19].

1.4. Thesis Outline

The rest of this report is organised into seven chapters. The following list provides a detailed description of the topics discussed in each chapter.

- **Chapter 2:** It describes the process of manufacturing ICs and then delves deeper into the testing domain. It discusses the current testing timeline and briefly explains the test development flow. Then, the testing hardware is explained with a detailed overview of loadboard parasitics.
- **Chapter 3:** The literature survey and the current state of the art in virtual testing is presented. This chapter offers insights into diverse techniques and methodologies for the digital twinning of loadboards and finally presents the limitations of the current methodologies for digital twinning.
- **Chapter 4:** Based on the current methodology limitations, the requirements for the framework are derived. Then this chapter proposes a new framework for accurate loadboard modelling to meet the derived requirements.
- **Chapter 5:** The complete framework implementation to convert the loadboard into an accurate virtual model is discussed. This chapter provides the flowchart and algorithm in detail to explain the implementation of the complete framework.
- **Chapter 6:** In this chapter, the framework implemented in the previous chapter is validated thoroughly to analyse its accuracy.
- **Chapter 7:** This chapter summarises this thesis and discusses potential future work.

2

Semiconductor Testing

This chapter introduces semiconductor manufacturing, more specifically, semiconductor testing. A general introduction to the manufacturing process is presented in Section 2.1. After this, Section 2.2 introduces the test development process in detail. Section 2.3 then presents the hardware used in the industry to materialize the testing process discussed in the previous section. Finally, Section 2.4 delves into the physical properties of the laodboard that are necessary for its modelling.

2.1. Semiconductor Manufacturing Process

The semiconductor manufacturing process starts with sand as raw material and ends with the final silicon chips used in electronic devices [20]. This process can be divided into three stages: Design, Wafer Fabrication, and Testing. The design stage involves creating the layout and specification for the chips, while the Wafer Fabrication stage transforms raw materials into silicon wafers through a series of complex steps. The testing stage ensures the fabricated chips meet all performance and quality standards [21]. A single company can carry out these stages, or multiple companies can carry out each stage separately, as each requires complex equipment and highly trained engineers [21].

2.1.1. Design

A sequence of steps known as the design flow, as illustrated in Figure 2.1, is followed to design the IC. The design flow includes stages such as specification, circuit design, layout design, layout verification, and tape out [6].

Specification

The design flow begins with defining a set of specifications. These specifications include the following [6]:

- **Functional Requirements:** A detailed description of the global and local functions of the ICs. This tells how the IC should behave and what its tasks are. It also includes the software that will run on it.

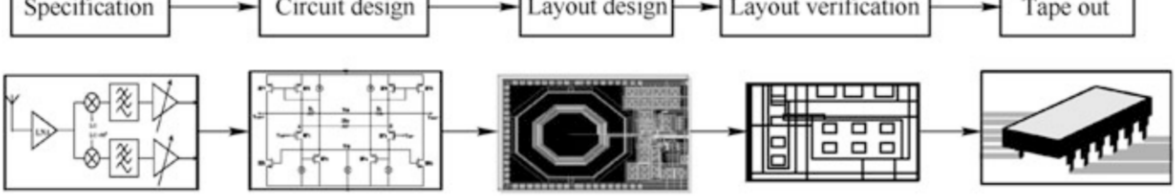


Figure 2.1: Design Flow [6]

- **Chip Architecture:** This is the software and hardware aspects of the IC. The software includes the protocol and operating system, and the hardware consists of the standard clock, memory, and system bus.

- **Input/Output Requirements:** This is the input and output signals of the target IC. The timing sequence concerning other chips, the clock pins, power/ground pins, and chip package specification and size are included.

- **Software Interface:** This is the software-accessible peripheral driver, which includes information regarding the IC programmability. The programmable registers of the chip, such as the hardware peripheral driver, are defined in this.

Circuit Design

After the specifications are defined, the circuit design begins. In this, the engineers develop the logic and circuit components for the ICs using *Hardware Description Languages* (HDL) such as VHDL or Verilog. For analog and mixed-signal ICs, languages like Verilog-AMS are used as they can handle both analog and digital signals [22]. After this, *Electronic Design Automation* (EDA) tools are used to verify the function and performance of the circuit.

Layout Design and Verification

After circuit design, EDA tools are used to draw the geometric representation of the circuit, which includes layers of different materials and structures that will be manufactured later in the process. Engineers then optimize the generated layout from the tool to achieve optimal performance such as power efficiency, speed, and area utilization [23]. After the design, the layout is verified using EDA tools. This involves ensuring that the layout adheres to the defined specifications and guidelines. Verification is essential to prevent any future manufacturing issues and electrical violations.

Tape-Out

After the validation, the final layout is sent for fabrication, called *Tape-Out* (TO). The layout is sent to the foundry with a *Process Design Kit* (PDK) that contains information such as design symbol, device model, parametric cell, and physical verification rules. The PDK includes the specifications that the foundry needs to adhere to during the manufacturing process to reach the optimum performance [23].

2.1.2. Wafer Fabrication

After the layout is completed and verified, the process moves to manufacturing. This is done in specially designed manufacturing sites known as wafer fabs; they are costly and can range from $2 billion to $4 billion based on the requirements [6]. The manufacturing process is performed in a clean room, where air is filtered periodically, as even a few impurities can affect the fabrication process.

The fabrication of chips starts with the production of silicon wafers. This begins by creating a silicon ingot by dipping and rotating a silicon seed crystal in purified molten silicon [21]. These ingots are then sliced into a thin disk-like shape called wafers, as shown in Figure 2.2, which are then used as the base for the manufacturing process. The production of ingots and wafers is handled by specialized suppliers rather than the semiconductor manufacturers.

The complete fabrication process after the wafers involves complex steps depending on the technology requirements. Below are the steps for the fabrication of planar CMOS technology [21]:

- **Oxidation:** A silicon dioxide layer is formed over a wafer using high temperatures in a furnace.

- **Photoresist Coating:** The wafer is coated with a light-sensitive material called photoresist that can be made into a pattern when exposed to *Ultraviolet* (UV) rays.

- **Lithography:** This step determines the size of planar CMOS transistors. A wafer is put inside a lithography machine, exposing it to UV light to form a specific pattern.

- **Etching:** The next step is to remove the degraded resist where UV light was exposed to show the pattern. There are two main types of etching: wet etching, which uses chemical baths to expose the pattern, and dry etching, which uses gases to form the pattern.

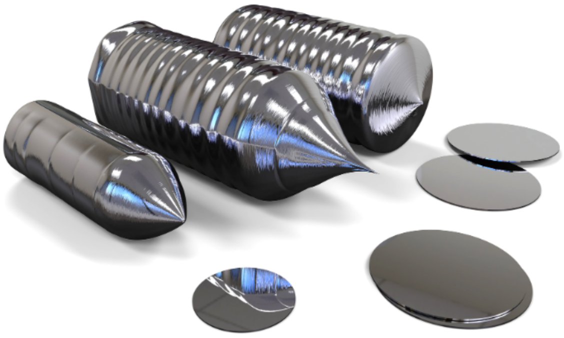


Figure 2.2: Silicon Ingot and Wafers [21]

- **Doping:** After the patterns are made visible on wafers, they are bombarded with ions to tune the conductive properties of the pattern.
- **Packaging:** After transistors are manufactured and the interconnect layers are laid out, the wafers are sliced with a diamond saw into dies. The die is then put inside a substrate that works as a baseboard and uses metal pins for the channels to produce the final chips.

The steps mentioned above are very sensitive and prone to defects. Equipment malfunctions during the process can introduce issues in the IC. For example, problems in the photoresist coating step can lead to photolithographic inaccuracies, causing under or over-exposure to UV rays. This can result in semiconductor components with incorrect dimensions and performance that is out of specification [6]. Similarly, problems during the doping process can alter the electronic properties of the semiconductor bandgap, resulting in components with inadequate electrical properties and reduced lifetimes [24]. The next stage, testing, is critical to identify and address these issues before the ICs are shipped to consumers.

2.1.3. Testing

This stage aims to ensure that the fabricated chips are reliable and meet all the requirements defined in the specifications by detecting any possible issues [24]. These issues in an IC can happen at any point in the semiconductor manufacturing process, such as product design or the fabrication process [24]. The issues can also be seen due to a faulty test being done on the IC, so engineers need to ensure that the test being done is thoroughly verified. The engineers develop suitable test plans to detect errors at the lowest cost. It can be said that testing is a process where a stimulus to the *Device Under Test* (DUT) is sent, and the response is then compared with the expected value [6]. The device is faulty if the output is inconsistent with the expected value. So, the testing process depends on the testing hardware (tester and loadboard), DUT, test stimulus generation, and test response capture. The testing process can be classified into different categories [25]:

1. **Digital Testing:** Verifies digital circuits' logical and timing operations.
2. **Analog Testing:** Verifies the performance of analog circuits in the chips.
3. **Mixed-SIgnal Testing:** Verifies both the digital and analog components of the IC.
4. **High-Speed Testing:** Verifies ICs that operate at high speed for timing and signal integrity.
5. **Radio Frequency:** Verifies the performance of the *Radio Frequency* (RF) circuits over multiple frequency ranges.
6. **Memory Testing:** Verifies the read and write operations of the memory components.
7. **System-on-Chip Testing:** Verifies the functionality of the subsystems in an *system-on-chip* (SoC).

Testing can be performed at two stages in the manufacturing process, the first after the wafer is fabricated and then after the packaging, as described below:

1. **Wafer Testing:** This testing is carried out on the wafers before they are cut into individual dies. This is performed using *Automatic Test Equipment* (ATE), which has a tester and probe cards to establish electric contact between the ATE and the wafer. Electric stimuli are applied to dies in the wafer, and their response is stored. The test results categorize the dies on the wafer into virtual bins, indicating whether they passed or failed the test. During testing, different bins can be used for different faulty behaviours, which then can be categorized and used for failure analysis to root out the problem.
2. **Final Testing:** After wafer testing, the dies are packaged, finishing the manufacturing process and resulting in finalized ICs. During the process of packaging and the addition of metal channels, there is a possibility of something going wrong, so a final test is done after the packaging to check the correctness of the chips. Since the package has many purposes, e.g., electrical insulation from surroundings, environmental protection, and heat dissipation, it is essential to check if it serves its purposes.

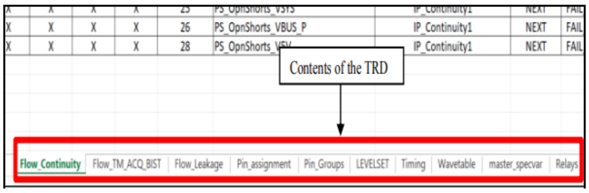
Multiple characteristics can be tested at wafer testing and final testing. The tests that can be performed are given below [25]:

1. **Parameteric Testing:** This involves testing chips to satisfy maximum current and voltage requirements.
2. **Burn-In Testing:** This is done by subjecting the chips to increased temperatures and voltages for a long time.
3. **Thermal Testing:** This checks the thermal characteristics of the chips, such as thermal resistance and thermal conductivity.
4. **Time Domain Reflection:** This involves measuring electrical parameters like capacitance using time domain reflectometry.
5. ***Radio Frequency* (RF) Testing:** This tests the chip at different frequencies to ensure their reliability and efficiency under different frequencies.
6. **Signal Integrity Testing:** This checks the device's ability to transmit and receive signals in its pins.
7. **Power Integrity Testing:** This focuses on how the power is divided into different components in the chip.
8. **Functional Testing:** This checks whether the chips perform the functions as intended. It verifies logical operations and functional correctness by applying different test patterns.
9. **System-Level testing:** This is done to check whether the chips behave correctly when integrated into a system. This helps identify the integration issues with the system and other components.

Based on the IC's requirements, a test program contains the tests needed to analyse all its aspects. This test program is then loaded on the ATE, and the tests are performed on the DUT. After performing the tests, the DUT's output is captured in real-time for data analysis so that the ATE can compare the data with the expected outputs.

2.2. Test Development Flow

Developing the test program is a crucial step in the testing process as it directly impacts the quality of the final product. Developing a test program requires significant resources, including skilled engineers, and extensive time and cost. Generally, developing a test program can take three to six months [6], depending on the complexity of IC, as it also involves the debugging and validation stages. The development is also costly, requiring advanced software tools and multiple iterations to ensure the program's robustness. This test program is then run on the ATE to check the functionality and performance of the DUT. The test program development starts during the designing stage of the IC. It contains several steps that are as follows [26]:



X	X	X	X	25	PS_OpnShorts_VSYS	IP_Continuity1	NEXT	FAIL
X	X	X	X	26	PS_OpnShorts_VBUS_P	IP_Continuity1	NEXT	FAIL
X	X	X	X	28	PS_OpnShorts_V	IP_Continuity1	NEXT	FAIL

Figure 2.3: Test Requirements Document [26]

- **Specification Definition:** The process begins with understanding the detailed specifications of the IC. This includes functional requirements, performance metrics, and the types of tests needed. Based on this metric, the test objectives are defined to outline the test requirements, such as timing requirements, electrical characteristics, and operating conditions. Based on these specifications, the objectives and scope of the tests for IC verification are established.

- **Test Planning and Analysis:** Based on the specifications, a detailed plan outlines the sequence of tests and their specific conditions. In this, a *Test Requirements Document* (TRD) shown in Figure 2.3 is developed. This sheet lists all the tests to be performed on the device. The tests are grouped in separate sheets according to the test flow. Engineers also perform an *Automatic Test Pattern Generation* (ATPG) to generate the patterns for detecting specific faults, such as stuck-at or transition faults. After this, a test coverage analysis is performed to ensure the plan covers all critical checks for the IC while balancing cost and complexity.

- **Test Program Development:** The engineers write the test program after the plan. The test program must follow a syntax specific to the ATE platform used for testing. Key components of the test program include test vectors, which are sets of inputs and expected outputs used to validate the ICs, and the sequence in which the tests are performed. Additionally, the program defines measurement parameters, such as voltage levels, timing parameters, and current measurements.

- **Test File Creation:** The test program includes files that contain data required to run the test program. These files can change based on the complexity of the IC and ATE platform. The common files are as follows [27]:

  1. **Pin Mapping File:** The file on the tester defines the pins on the DUT and how the DUT pins are connected to the tester through the loadboard.
  2. **Pattern Files:** This refers to a set of electrical inputs and outputs given to the DUT to prepare it for tests and analyse its functionality and performance.
  3. **Level Files:** Contain voltage and current levels to drive the pins of the DUT.
  4. **Test Flow:** Contains the sequences that define the test flow by specifying the test steps and the order to execute them.
  5. **Relay Set File:** Contains data on a relay that needs to be turned on or off for specific pins.

The test program combines several test suites to test the DUT's different parameters and functionalities. Its development starts before the first silicon, the initial batch of the fabricated ICs, but the test program is debugged after the initial batch is received. The debugging is done using an ATE platform to run the test program, which is connected to the loadboard that works as a bridge between the DUT and ATE [26]. This debugging is crucial to determine if there are any errors in the test program. Based on the debugging results, the test program is optimized, which includes refining test vectors and measurement parameters, known as test limits [26]. This is an iterative process that ensures the accuracy and robustness of the test program.

After the complete debugging of the test program, it is optimized for efficiency, ensuring that tests are conducted with the shortest test time feasible without compromising quality. This involves streamlining the test flow and removing redundant tests. The finalized test program is then loaded onto the ATE to test the chips. This consists of transferring the program files and configuring the ATE settings for production. Once configured, the ATE executes the test program on the ICs for mass-scale testing. This stage involves setting up the test conditions, applying test vectors, capturing results, and analyzing data in real-time.

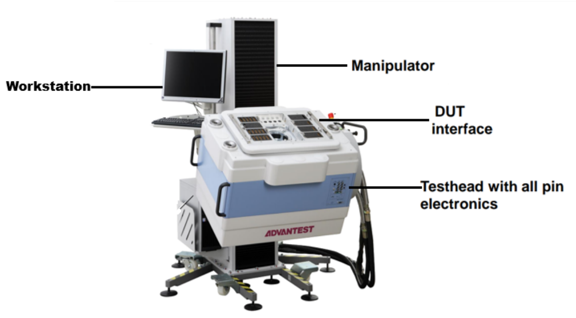


Figure 2.4: Automatic Test Equipment (ADVANTEST) [27]

2.3. Testing Hardware

Testing is a time-consuming process that requires skilled engineers and dedicated test equipment. The main hardware that makes testing possible is ATE and Loadboard. This section gives a detailed introduction to all the equipment used in testing.

2.3.1. Automatic Test Equipment

Automatic Test Equipment (ATE) is the core of semiconductor testing; it uses electrical configurations to conduct tests that assess the functionality, performance, and quality of ICs [27]. As the name implies, they automate electrical stimuli and processes while requiring minimal human interaction [27]. ATE optimises the testing process, which results in faster and more accurate tests. The ATE shown in Figure 2.4, from the semiconductor company ADVANTEST, comprises three main components, as described below. Note that most ATEs from other brands also exhibit a similar configuration. The main components of the ATE are described below [28]:

- **Manipulator:** This allows precise positioning of the DUT and the test head. The manipulator can move the test head in multiple axes, accommodating various DUT configurations and sizes. This flexibility is essential for testing a wide range of IC packages.
- **Workstation:** This refers to the computer test engineers use to interact with the ATE system. The test programs are developed, modified, and managed in the workstation. It serves as the control centre for the ATE, providing the interface through which engineers can set up tests, monitor the testing process, and analyze the results.
- **DUT Interface:** This is the place in the ATE where the loadboard is attached. The loadboard, which holds the DUT, establishes the electrical pathways between the DUT and the ATE. This interface has been developed to accommodate different types of loadboards and ensure reliable connections for testing.
- **Testhead:** It is the core of ATE, containing the electronics necessary to apply test signals to the DUT and measure the responses. This includes signal generators, measurement units, and data acquisition systems.

An ATE operates using a test program developed by test engineers based on the project's requirements.

2.3.2. Loadboard

Loadboards serve as the contact interface between the test head of the ATE and the pins of the DUT being tested. They typically comprise a *Printed Circuit Board* (PCB) that carries the test sockets or probes for the ICs under test and is mounted on the test head [29]. An image of a loadboard is shown in Figure 2.5. Loadboard design differs based on the complexity of the DUT; a typical board consists of four main components [30]:

- **Pogo Pins:** These pins connect with the ATE.

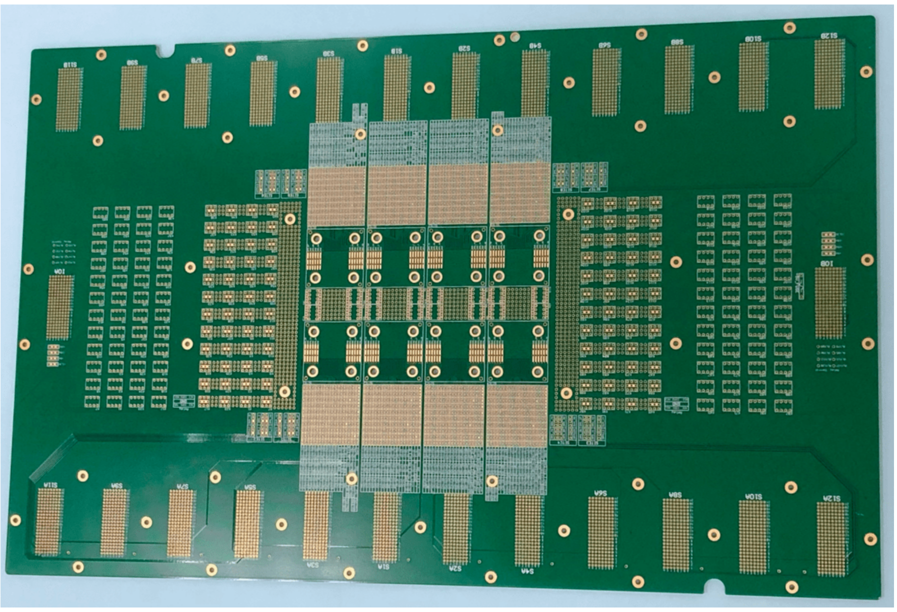


Figure 2.5: Loadboard

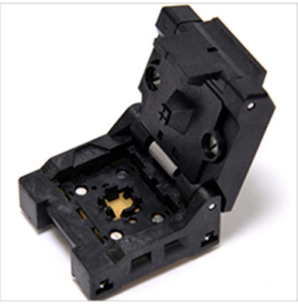


Figure 2.6: Socket [31]

- **Transmission Line:** These interconnections transmit the signals from the loadboard to the DUT. These interconnects are routed across different layers of the loadboard to accommodate complex signal paths.

- **Components:** These include active components, such as *Solid State Relay* (SSR), and passive electronic components, such as resistors, capacitors, and transistors, required for the testing process.

- **Sockets:** These are the connectors on the loadboard where the DUT is inserted as shown in Figure 2.6. They provide the mechanical and electrical interface between the DUT and the loadboard

A loadboard consists of multiple sites; a single site represents one socket that can hold one DUT. For loadboards used in development and debugging, the number of sites is generally 8 or 16, whereas loadboards for production are more extensive and can consist of up to 64 or 128 sites [32]. In some cases, a board is made of two different loadboards: the motherboard and the daughterboard, as shown in Figure 2.7 [26]. This practice is often used when a loadboard is already present, and for the new project, only minor changes are needed, which can be done by adding a daughterboard. In these cases, the motherboard has the main input signals that receive electrical input from the tester, and it is connected to the daughterboard that contains extra components and the sockets, which are then connected to the DUT [25].

A loadboard is a complex PCB containing several layers. Any element in the board can cause signal integrity issues, so it is essential to carefully design the loadboard while considering the DUT

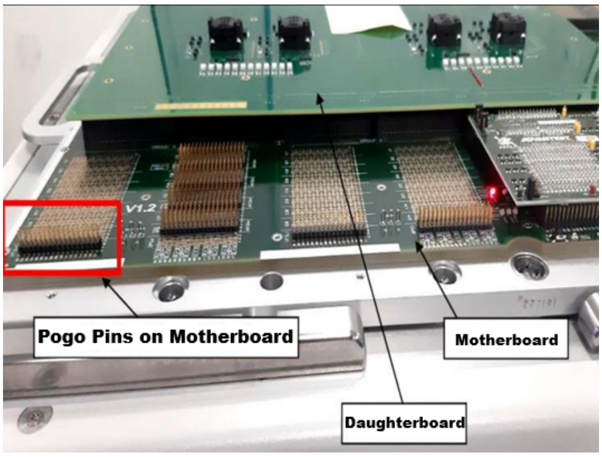


Figure 2.7: Motherboard and Daughterboard [26]

specifications [29]. First, a layout package is generated, which contains general design rules, test system mechanical guidelines, and project-specific requirements. These documents describe the design's goals, needs, and constraints [33]. Then, the schematic of the loadboard is made based on the specifications, and then an EDA tool is used to generate the board's layout. The first step in the layout process is determining the type of stack up, which refers to the arrangement of the different layers within the PCB. Loadboards generally have asymmetrical stackup to meet specific requirements and accommodate unique layout constraints. There are typically four types of layers in the stackup, which are as follows:

- **Ground Plane:** They provide grounding to the circuits and prevent interference between two signal layers.
- **Signal Layer:** The striplines are routed in this layer. They are sandwiched between the ground plane to avoid signal coupling.
- **Power Plane:** They provide power to the circuits in the loadboard.
- **Dielectric Layer:** These insulating layers separate the conductive copper layers.

After determining the board's stackup, the signals are routed in signal layers, and pads and vias are determined. The designer's decision for signal routing is based on considerations such as minimizing impedance mismatch to keep the voltage and timing variations to a minimum. Then, pads are chosen, which are the small surface areas of exposed metal on the layer to which the components are soldered. They are then selected based on the size and type of components mounted on the board to achieve mechanical stability. Finally, vias are placed on the loadboard to allow the signals to pass from one layer to another. They are placed so that the effect of vias on signal integrity is minimal. Since a loadboard often contains multiple sites, the layout needs to be created so that the site-to-site variations are as minimal as possible. The loadboards are generally significant in size due to their multisite nature.

2.4. Loadboard Parasitics

Ideally, designers try their best to minimize any impedance mismatch by making the stripline the same impedance as tester channels, but there are always some variations. This is due to parasitics, which are the unwanted elements due to the layout characteristics such as placement of the stripline, electromagnetic interference due to adjacent lines, and physical factors such as the length, width, and height of the stripline [34]. This mismatch in the impedance can result in the loss of signals while travelling through the transmission line. The test engineers need to know the possible loss due to the transmission lines to account for this error in the test program.

The loss in the signal will only occur when the line impedance does not match the tester channel impedance. So, to analyse this, the impedance of the stripline needs to be calculated. The impedance of a transmission line can be calculated by finding the four basic parasitics of the stripline [35], which are as follows:

- **Parasitic Resistance (R):** This is the inherent property of the conductive elements present in the wire used to transmit signals. Parasitic resistance can cause voltage drops and power losses and compromise signal integrity, especially in high-speed devices [36].
- **Parasitic Capacitance (C):** This occurs due to the generation of an electrical field between two close channels; this can cause coupling, leading to crosstalk and signal degradation.
- **Parasitic Inductance (L):** This parasitic occurs when current flows through conductive elements. It opposes current fluctuations, resulting in signal delays, impedance variations, and undesired coupling.
- **Parasitic Conductance (G):** This measures how easily a current can pass through the trace. A shunt resistor between the transmission line and the ground represents it.

These basic parasitic elements of the line can be calculated using EDA tools by running signal integrity and electromagnetic analysis. These parasitics depend on the physical properties of the stripline as well as the layout structure, such as having a trace nearby. After finding these four parameters, it can be used to calculate the transmission line's impedance, denoted as Z, using the formula described in Equation 2.1 [37].

$$Z = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \tag{2.1}$$

In Equation 2.1, ω (omega) stands for the angular frequency of the signal. It is defined as $\omega = 2\pi f$, where f represents the signal's frequency. The term j describes the imaginary unit, essential to deal with the phase differences caused by inductance and capacitance in the voltage and current [38]. After calculating the impedance of the stripline, it is compared with the tester channel impedance. If there is a difference between them, there is a situation known as impedance mismatch [37]. This mismatch can cause reflection in signals, which can cause a loss of voltage. These reflections can negatively impact the quality of the signal, causing loss, distortion, or interference, ultimately affecting the circuit's performance. A dimensionless parameter called the reflection coefficient, denoted as Γ, is used to measure the degree of signal reflection caused by impedance mismatches, which can be determined from Equation 2.2 [37].

$$\Gamma = \frac{Z_L - Z_S}{Z_L + Z_S} \tag{2.2}$$

In Equation 2.2, Z_L is the impedance of the loadboard stripline, and Z_S is the source impedance, which is the impedance of the ATE channel. The reflection coefficient ranges from -1 to 1, indicating the extent of reflection. A value of -1 signifies total negative reflection, equivalent to a short circuit. On the other hand, a value of 1 represents total positive reflection, similar to an open circuit. These reflections significantly impact the output voltage, as constructive or destructive interference can occur between the reflected and incident waves. One can use the formula in Equation 2.3 to determine the effective voltage. [37]. This formula helps determine the exact output voltage after being affected by the first reflected wave.

$$V_L = V_0 \times (1 + \Gamma) \tag{2.3}$$

The initial voltage of the signal is denoted as V_0. This formula helps determine the exact output voltage after being affected by the reflected wave. Knowing the effect of the loadboard on the output voltage can allow test engineers to adjust their test programs to account for these discrepancies in pre-silicon simulations and ensure accurate simulation results.

3

Virtual Testing and Digital Twinning

This chapter introduces the current techniques and their limitation in virtual testing in Section 3.1. Section 3.2 discusses the concept of digital twinning and the current methodologies with their limitations.

3.1. Virtual Testing

The increasing complexity of semiconductor chips has led to a significant rise in the cost and time required for testing. The industry is trying to find ways to solve this bottleneck, and one way is to introduce virtual testing in the test development flow [8]. The need for a change in testing scenarios was already recognised in 1996 by Mielger and Woltz [11]. The authors acknowledge the need for a simulation tool to make and debug the test program. They also highlighted the challenges of implementing such an approach, emphasising that a loadboard and a DUT model are essential components for successful simulation.

3.1.1. Evolution towards Virtual Testing

Since simulations for DUT testing started gaining recognition in the industry, several researchers have introduced methodologies to make it more efficient and accurate. The authors, in 1996 [11], introduced a virtual environment called VTB that allows defining the signals with the help of pre-defined models of the test programs. They used VTB to save time in making a test program by making signal-generating modules, rule-based consistency check modules, and import functions for foreign data formats. Building on this foundation, subsequent research led to the development of VTest in the early 2000s, a software platform for designing and simulating test requirements and DUT models [12]. However, the use of VTest revealed three significant challenges:

- Different libraries for testing and design were employed, which made it difficult to synchronize test and design requirements across different platforms.
- Lack of testing component models to simulate all the aspects of the test process, which led to inaccurate testing.
- No ways to share the models between different stages of simulation.

To address these issues, VTest introduced a more synchronized framework in 2001, standardising libraries across both the design and testing domains. This allowed for better communication and data sharing between the design and testing stages [12]. This framework also improved the integration of different models by creating a generic resource library accessible throughout the simulation process [12]. For the simulation in VTest, VHDL was used to make the DUT model in the form of a high-level netlist. VTest proved to be a system that enables simulation of the DUT model, which the designer provided, with the help of the test system. The simulation of DUT allowed the test engineer to work sooner than they were doing by integrating the design and testing environment, and they could use VHDL models of the components to review the design and provide feedback to designers. This paved the way for test development to begin even sooner, as they could access design files and requirements

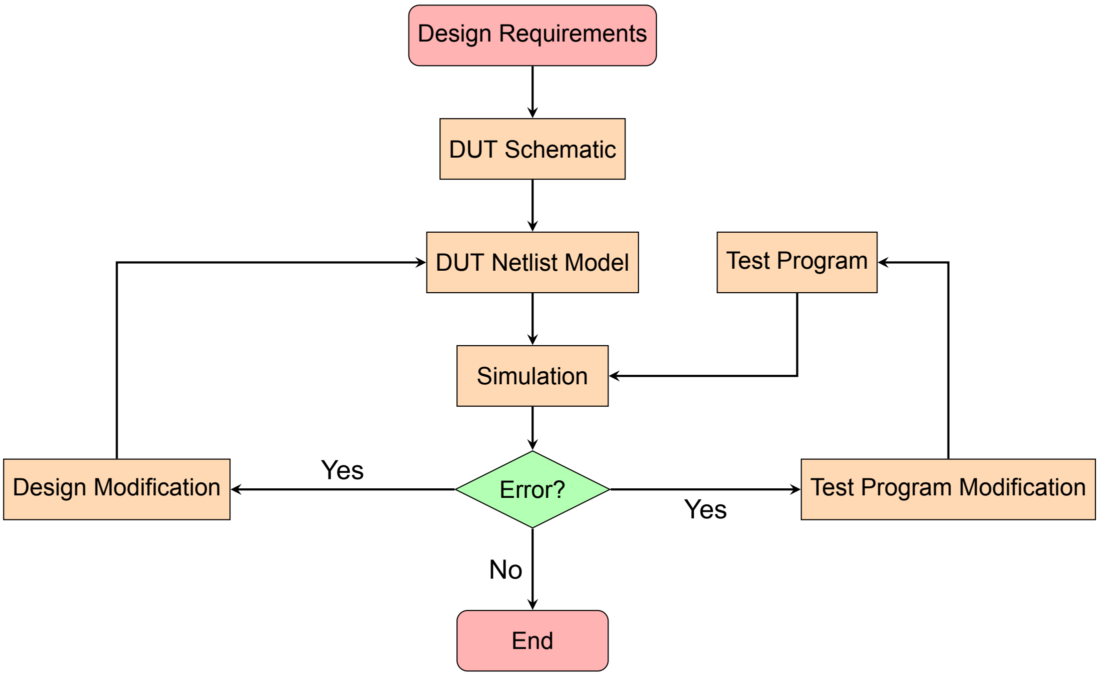


Figure 3.1: Simplified Simulation Approach [13]

earlier, giving them more time to make a robust test program. Despite these advancements, certain limitations remained, particularly in modelling the entire test development process using HDLs such as Verilog, VHDL, or SystemC. Additionally, communication barriers between different simulation tools and the accurate modelling of ATEs continued to pose significant challenges.

In response to these issues, the concept of *Virtual Prototyping* (VP) emerged in 2002 [39]. VP refers to the simulation of physical products that allows an engineer to analyse and validate the requirements and designs of the product. The VP technique at that time had some limitations, such as a lack of integration of different tools. To solve this issue, an environment that can integrate all the tools and models is required. This integration will provide a straightforward approach to testing the ICs by simulating the models from different stages using EDA tools. Considering these problems, a simulation approach was proposed to verify and test a system before making the hardware in 2003 [13]. The approach is shown in Figure 3.1, which allows virtual testing for DUT validation [13].

The approach in Figure 3.1 starts with determining the design and its requirements, based on which a schematic is generated. This schematic is then used to create a netlist of the DUT, which can be used for simulations. In parallel with this process, the test programs are developed based on the requirements, which are then used to simulate the DUT netlist. These simulation results are then analyzed, and if an error is found, feedback is provided to the relevant engineers to verify the DUT model and the test program. By following this approach, an IC can be virtually tested for specific parameters using simulation on the models of the device. This approach found certain complications, such as the absence of accurate simulation software and precise models [13].

3.1.2. Test Development Speedup through Virtual Testing

Early researchers successfully proposed an approach that allows simulation for DUT testing before the hardware is developed. This approach enables the engineers to verify the functionality of their test program by predicting the output of the DUT model. After this, ways to integrate the virtual testing approach into the test development flow were researched. Virtual testing is a subset of virtual prototyping that involves developing and debugging test programs using simulation models of the DUT and the test environment [40]. This can reduce time-to-market by reducing the debugging time needed by test engineers with the actual physical environment. Three main issues do not allow proper integration of virtual testing in the flow [40].

- It is a time-consuming and complex task to model each step of the test development process

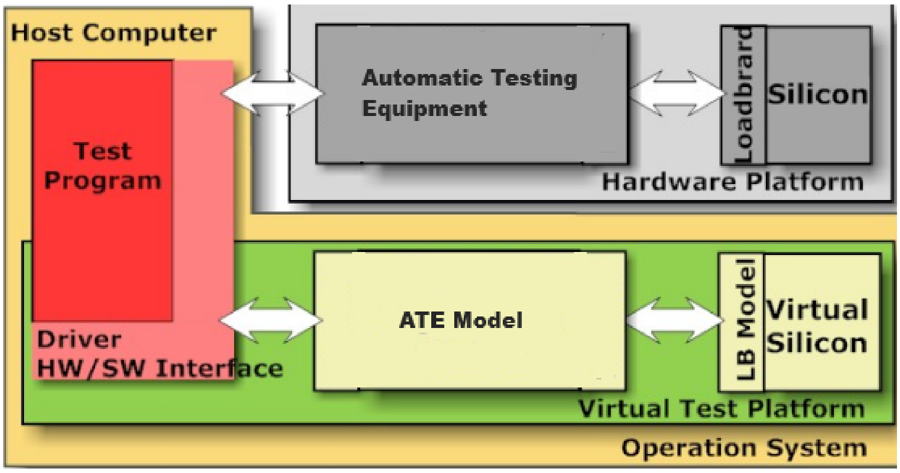


Figure 3.2: Overview of Virtual Test Environment [14]

using models in HDL (Verilog, VHDL, SystemC).

- Lack of communication between different simulation software, as designers may use different software for DUT than what will be used in testing.
- Accurate and precise modelling of ATE.

To overcome these limitations, a virtual platform was used to bridge the gap between design and test so that test program development could be more efficient [14]. The new methodology supports combining the DUT simulation software and the model of test instruments into a single simulation environment, which means the libraries used by designers can be used with the tester models. Using this environment, the test engineers can efficiently run simulations to design, debug and verify all test-relevant processes [14]. The virtual testing environment uses methodologies, models and tools to maintain interoperability between the EDA tool and test system. Figure 3.2 gives an overview of the virtual test environment's components. In Figure 3.2, the environment is divided into two main components:

1. **Hardware Platform (Top):** This is the physical testing system where the actual test program is executed on hardware components such as ATE, Loadboard, and DUT.
2. **Virtual Test Platform (Bottom):** This is the virtual test platform that replicates the functionality of the hardware platform in an operating system. This allows engineers to execute the test program using models of ATE, loadboard, and DUT before the arrival of silicon. The virtual testing environment comprises three significant layers: Test Application, Driver Interface, and Virtual Models [14].
   - **Test Application:** This layer contains the test program that specifies what kind of test needs to be performed.
   - **Driver Interface:** This layer connects the test program with the virtual models. This creates stimuli based on the test program and is based on an ATE platform.
   - **Virtual Models:** The models of loadboard and DUT are then given the stimuli based on which a voltage is forced in the models, and then the results are analysed. For the modelling, the authors used SystemC-AMS language as it provides additional support for analog signals.

In both platforms, the test program is executed on a host computer, which then communicates with the hardware components or their models using the HW/SW driver interface. In this overview, the test program is used for both the virtual test platforms and the hardware platform. For virtual testing, the testing hardware, the tester, the loadboard, and the DUT are modelled and simulated based on the test program.

This new approach proved successful as it solved one of the major issues: to make an integrated environment for all models and libraries. The limitation they found was that the same test program made for hardware cannot be directly used in simulations due to the differences in the driver interface between the simulation environment and the actual ATE. This mismatch between environments led to significant inefficiencies in test program debugging. To address this limitation, the ATE environment saw advancements that allowed for the same test programs to be used in both the simulations and physical testing. This was achieved by creating virtual ATE interfaces that closely resembled the driver interaction in the physical hardware. This allowed engineers to debug the test program using virtual testing, which made the test development process faster [17]. This new approach allowed some companies like Infineon Technologies to research the advantages of using a virtual testing environment [10]. They discussed the prospects of reducing the test debugging efforts of test programs post-silicon. For that, a digital model of the ICs can be used with stimuli parallel to the chip design, which will shorten the test development time. The chip can be modelled as an RTL code, and stimuli can then be applied to verify how the DUT will behave under a specific test program [10]. They mainly focus on using this methodology for pressure sensors and to verify sensor signal processing. After utilising virtual testing for pressure sensors, Infineon found that this approach significantly reduces the development timeline and enhances chip quality. The main advantages of using virtual testing in an industry are as follows [10].

- Once silicon is ready, quick feedback on the device’s performance can be provided.
- The feedback obtained in parallel with the design development also proved to help verify the design.
- Time for debugging can now be used for quality checks of the test program and test setup optimisations.

These advantages were further validated through industrial case studies, proving that virtual testing allows for earlier debugging of test programs, significantly reducing the time from design to verification [8]. The virtual testing methodology was also compared with the actual ATE results for memory tests using the same test program [8]. The memory test involved checking the values stored in the registers. The value in the register was first set to 0, then inspected to ensure that the correct values had been stored in the proper register [8]. This process was then repeated after changing the value from 0 to 1. These tests detect memory corruption, register addressing errors, or improper write operations. The memory test was performed using both virtual and physical testing environments. The outputs from both environments were identical, proving that virtual testing provided accurate results [8]. The only discrepancy found was that the run-time of virtual testing was more than its physical counterpart. Although virtual testing is more time-consuming than physical testing, the ability to test before silicon outweighs the disadvantages by accelerating the overall development cycle and reducing time-to-market [8].

3.1.3. AMS-VT: Overview and Limitations

The previous section highlighted the advantages of using virtual testing for test development. After being proven that virtual testing allows for faster test development by allowing pre-silicon debugging [8], different companies tried to make an advanced methodology for virtual testing that can be used in the industry. Testinsight introduced a new methodology called AMS-VT, currently the most advanced virtual testing methodology used by NXP [18]. Its development was encouraged by the increasing complexity of semiconductor designs and the corresponding demand for more complex virtual testing methodologies [18]. AMS-VT streamlines the test development flow, ensuring all the parts are in sync: test program, ATE, DUT and the loadboard. The following things can be checked with AMS-VT pre-silicon [18]:

- Validate the test program flow and the pass/fail status of the test suites.
- Validate ATE instruments and DUT setup throughout the test execution.
- Confirm ATE-DUT connectivity at every stage of the test program.

AMS-VT uses EDA software with a virtual ATE in a closed-loop system for their stability. This closed-loop system helps recognise the point at which an error occurs to accelerate test program bring-up and

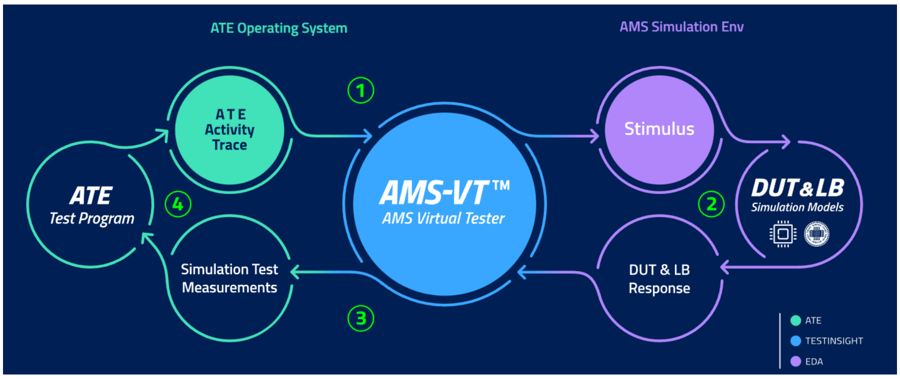


Figure 3.3: AMS-VT [18]

debugging while improving quality. The working of AMS-VT is shown in Figure 3.3. In a closed-loop system, the ATE operating system connects through AMS-VT to an EDA environment, such as the Cadence Spectre AMS simulator. It works according to the flow (also marked in Figure 3.3) given below:

1. In the ATE operating system, all the activities of the ATE test program are recorded and stored by the ATE Activity Trace in the form of stimuli.
2. The stimuli are then used to drive the DUT signals via the loadboard. The simulation environment also records the DUT response.
3. The responses are then sent back to the ATE operating systems through the connection made by AMS-VT.
4. The test program is then updated based on the response of the DUT, after which the loop is restarted.

Test engineers can significantly reduce test development time and boost confidence in the test setup by using AMS-VT to debug their loadboards and test programs before TO when ICs are delivered for industrial manufacture [18]. This allows them to gain early insight by pre-silicon debugging the developed product. One of the critical features of AMS-VT is its use of Verilog-AMS, an extension of Verilog that supports analog and mixed-signal designs [18]. This capability is crucial for accurately simulating the complex interactions between digital and analog components within the loadboard and DUT, providing a more thorough simulation environment.

Limitations of AMS-VT

The current virtual testing environments solved most of the issues by allowing the exchange of resources between different simulation software and allowing the same test program for both simulations and physical ATE. These advancements have helped companies like Infinoen to reduce time-to-market [10]. The most advanced virtual testing methodology, AMS-VT, further improved virtual testing for complex ICs, making it suitable for industries to include it in their test development flow [18]. However, despite these advancements, certain fundamental limitations constrain its effectiveness and highlight the need for further innovation.

One significant limitation is the inaccuracy of the loadboard model. The current model is represented as an ideal netlist in Verilog-AMS, which is manually created. This manual process is prone to human errors, and it fails to capture real-world characteristics like parasitics. This does not allow for accurate pre-silicon debugging. Another significant limitation of AMS-VT is its current restriction of single-site simulations [18]. This is problematic since real loadboards have multiple sites, each with unique parasitic effects and voltage differences. This also poses a significant issue in the debugging of the test program, as the test engineers cannot predict how the program will behave on different sites.

To develop a robust test program, it is important to simulate all sites, as site-to-site variations can affect the program's performance regarding voltage levels and timing. Since AMS-VT cannot handle multiple site simulations, a loadboard model that can accurately emulate these site-to-site variations is

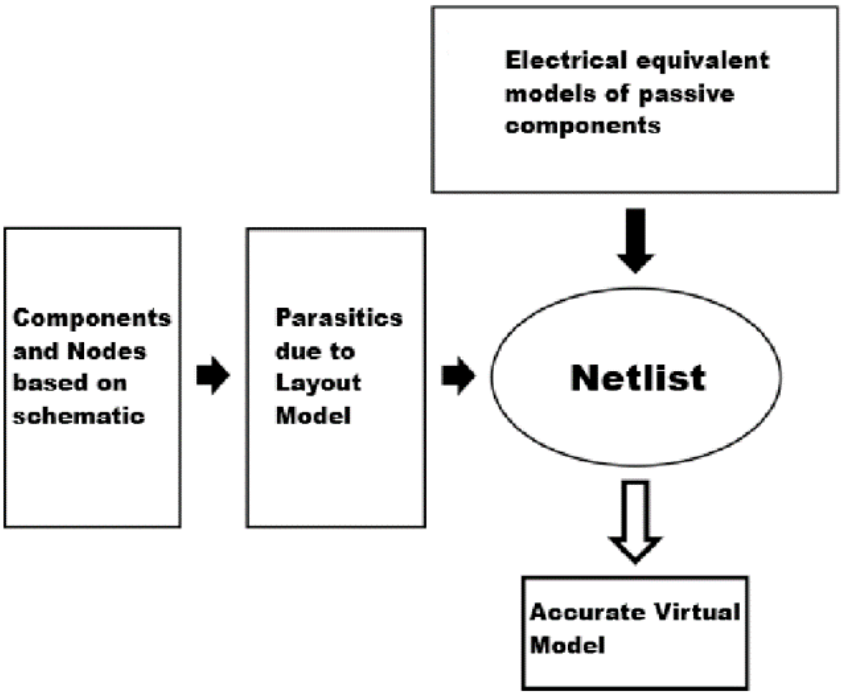

Electrical equivalent models of passive components
Components and Nodes based on schematic
Parasitics due to Layout Model
Netlist
Accurate Virtual Model

Figure 3.4: Digital Twinning Setup [43]

needed so that the test engineers can predict the behaviour of the test program on different sites. Due to these significant limitations, AMS-VT can only debug basic ATE-DUT connectivity and test flow. It cannot thoroughly debug the test program regarding site-to-site voltage and timing variations. These limitations highlight the need for more accurate digital twinning of the loadboard.

3.2. Accurate Loadboard Modelling

The importance of virtual testing methodologies was discussed in the previous section. A critical limitation was identified: the current loadboard model is an ideal netlist made manually, not considering physical effects like parasitics. To address this, a proper digital twinning method is necessary to model loadboards accurately. Digital twinning refers to creating a virtual representation of a physical device that mirrors its behaviour and characteristics in a virtual environment [15]. In this context, digital twinning involves developing a virtual model of the loadboard that accurately reflects its physical characteristics known as parasitics, allowing simulation to give results consistent with the actual physical board [16].

A systematic approach can be followed to generate a digital twin of the loadboard, shown in Figure 3.4. The initial step involves defining the physical dimensions and placements of the components on the PCB, which means making a schematic and layout of the board. Then, the layout must be used to run an electromagnetic simulation to get parasitic due to the physical characteristics of the board. Finally, the component needs to be modelled, such as models of relays, and then all of this needs to be added together as a netlist to create a digital twin of any PCB [41]. EDA tools can generate the netlist and perform electromagnetic simulations to extract parasitics. These tools allow the user to develop a complex system like PCB in a computer-based environment, which can then be analysed by running simulations [42].

The approach shown in Figure 3.4 can generate a digital twin, but this requires several complex steps, such as ideal netlist generation and parasitic extraction. Since no single EDA tool can complete all these steps, a three-stage approach can be taken, as follows [43].

1. **Schematic and Ideal Netlist:** For the first stage, the schematic can be generated and then used to create an ideal netlist.

2. **Layout and Parasitics Extraction:** Using the schematic, a layout can be generated and then parasitic can be extracted by running electromagnetic simulations.

3. **Final Netlist:** The parasitics extracted can be integrated into the ideal netlist to create an accurate netlist that accounts for the physical characteristics of the board.

Engineers are already creating the schematic and layout using an EDA tool [43]. The next step is to find EDA tools to generate an ideal netlist and run an electromagnetic simulation for parasitics. In the following sections, specific EDA tools for ideal netlist generation and parasitic extraction will be discussed, their capabilities will be compared, and why particular tools have been selected for this process.

3.2.1. EDA Tools for Ideal Netlist Generation

The process of ideal netlist generation is a critical step in developing a digital twin for a loadboard. A netlist serves as the fundamental blueprint of the circuit, detailing the electrical connections between various components on the PCB [44]. In the early stages of digital twinning, generating an ideal netlist involves creating a simplified version of the circuit in HDL that captures the logic and connections without accounting for physical effects such as parasitics. This ideal netlist is the foundation for more detailed models for simulations [44]. In the context of loadboard design, where circuits can be highly complex with numerous components, the ideal netlist simplifies the circuit to its logic state, enabling engineers to focus on the functional aspects of the schematic.

Tools Overview

There are EDA tools from multiple vendors that can capture the schematic and then generate an ideal netlist. It is crucial to analyse these tools and choose the best one for ideal netlist generation since an error in netlist will disrupt the digital twinning process before it begins. Below is a detailed analysis of four prominent EDA tools used for ideal netlist generation: Cadence Allegro, EasyEDA, Altium Designer, and Autodesk Fusion 360.

- **Cadence Allegro [45]:** It is one of the industry's most widely used EDA tools, especially for complex PCB designs. The tool is known for its robust schematic capture and netlist generation capabilities. After creating the schematic, the tool allows the user options for netlist generation, including support for various netlist formats such as SPICE, Verilog, and VHDL. The netlist generated by Allegro is known for its accuracy and completeness, ensuring that all components and connections are correctly represented.

- **EasyEDA [46]:** This cloud-based EDA tool has gained popularity due to its accessibility and ease of use. While it does not have extensive features like Allegro, it offers a straightforward approach for netlist generation. This tool was made for smaller and less complex projects or engineers looking for a quick and easy way to generate a netlist.

- **Altium Designer [47]:** This is also a powerful EDA tool widely used in the industry for PCB design and netlist generation. It offers an integrated environment allowing engineers to generate accurate netlists. Like Allgero, it can also handle large, complex schematics with numerous components and connections. The generated netlist can be exported in various formats, including SPICE and Verilog.

- **Autodesk Fusion 360 [48]:** This is primarily known as a 3D *Computer-Aided Design* (CAD) tool, but it also includes an electronics design module that supports netlist generation. The netlist is generated by capturing the schematic using the electronics design module. However, Fusion 360 is not as specialised as other EDA tools in netlist generation due to its main focus on 3D CAD.

Each of the tools mentioned above offers unique strengths for ideal netlist generation. The following sections will compare these tools based on specific metrics to choose the best available tool.

Comparison Metrics

This section describes the metrics for evaluating the EDA tools for ideal netlist generation. The metrics reflect the needs and challenges of ideal netlist generation for loadboards. The chosen metrics are as follows.

Table 3.1: Comparison of Netlist Generation Tools

Tool	Complex PCB Design	Integration	Usability	Flexibility
Cadence Allegro	+ +	– –	+ +	+
EasyEDA	–	– –	+ +	–
Altium Designer	+	– –	+ +	+
Autodesk Fusion 360	+	– –	+ +	+

1. **Complex PCB Design:** This metric evaluates how well the tool can manage complex designs involving multiple layers, numerous components, and interconnections.

2. **Integration:** This metric evaluates how well the netlist generation tool integrates with simulation tools, particularly those used in virtual testing like AMS-VT.

3. **Usability:** This metric assesses the tool's ease of use.

4. **Flexibility:** This metric assesses the tool's ability to be flexible to specific needs, such as adapting to different designs.

The tools will be graded for each metric described above. There are four levels using which the metrics will be evaluated:

- **"+ +":** This represents optimal performance, which means no more improvement is required.

- **"+":** This means satisfactory performance, indicating that some improvements can be made, but it is still acceptable.

- **"–":** This represents below marginal performance, indicating a need for improvement to reach an adequate level.

- **"– –":** This means poor performance, indicating a need for significant improvements.

Tool Comparison and Analysis

This section evaluates the EDA tools based on the metrics for ideal netlist generation. In Table 3.1, the grades for each tool across the four metrics can be seen. Cadence Allegro was the best tool for handling complex PCB designs, a grade of (+ +), due to its ability to handle multi-layer, high-density designs. EasyEDA scored the lowest, with a grade of (–), due to its inability to handle complex design as it is suitable for complex projects. The other two tools scored a grade of (+) as they both can handle more complex designs, but they are slightly less powerful than Allegro. All the tools scored the worst possible score in integration due to them generating a multisite netlist, whereas AMS-VT can only handle single-site netlists. The other three tools scored (+) in flexibility as they can handle multiple design configurations with ease. The only designs they cannot handle properly are configurations when a loadboard has two boards. However, due to constraints at NXP, which require using Cadence tools to maintain compatibility across all stages of the design and testing process, Cadence Allegro has been selected as the primary tool for ideal netlist generation.

3.2.2. EDA Tools for Parasitics Extraction

Parasitics extraction determines the accuracy of the virtual model when compared with its physical counterpart [16]. This quantifies the four basic PCB parasitics explained in Section 2.4. The process of parasitics extraction involves using specialised EDA tools to simulate the PCB layout. This is a complex process as loadboard is a multilayer PCB in which several things contribute to the increase of parasitics [29]. Another significant challenge is the frequency-dependent nature of parasitics. As the operating frequency of the signals on the loadboard increases, the parasitic effects also increase [49].

Tools Overview

Parasitics extraction needs specialised tools that can simulate a PCB layout to predict the possibilities of unwanted elements. Multiple tools in the market allow the user to run an electromagnetic simulation for parasitics extraction. The result of these tools will determine the accuracy of the digital twinning process of the loadboard. Below is a detailed analysis of four prominent EDA tools: Cadence Sigrity Aurora, Cadence Sigrity PowerSI, Keysight ADS, and Siemens hyperLynx.

1. **Cadence Sigrity Aurora [50]:** It is known for its early detection of signal and power integrity issues at every design stage. It integrates with Cadence's PCB design environment, making it a popular choice for designers within the Cadence ecosystem. While it can handle complex PCB designs, its key limitation is the lack of frequency-dependent analysis, which can limit its effectiveness in situations where frequency-dependent parasitics play a significant role.

2. **Cadence Sigrity PowerSI [51]:** It is another tool from the Sigrity suite that offers accurate electrical analysis for PCBs, enabling comprehensive studies on signal and power integrity and frequency domain simulations. Unlike Aurora, PowerSI can extract frequency-dependent parasitics, making it ideal for high-frequency applications. It is compatible with various complex layout databases where the accurate modelling of parasitics based on frequencies is essential for ensuring signal integrity.

3. **Keysight ADS [52]:** It is a tool known for its RF design capabilities. While this is not exclusively a parasitic extraction tool, it offers post-layout electromagnetic simulation features that can be used for parasitics extraction. The strength of this tool is its frequency domain analysis, which makes it highly suitable for applications involving high-frequency signals. However, its primary focus on RF applications may be more complex for specific parasitic extraction tasks than tools like PowerSI.

4. **Siemens hyperLynx [53]:** It is an EDA tool provided by Siemens that allows electromagnetic simulations for parasitics extraction. It is mainly known for its ease of use and ability to provide insights into post-layout parasitics quickly. It also includes frequency-domain analysis, making it suitable for high-frequency designs. However, while it is user-friendly, it may not offer the same accuracy of frequency-domain analysis as specialised tools like PowerSI or Keysight ADS.

Each of the tools mentioned above offers unique strengths for parasitic extraction. The following sections will compare these tools based on specific metrics to choose the best available tool.

Comparison Metrics

This section introduces the metrics that will be used to evaluate the parasitics extraction tools. These metrics are based on the requirements for digital twinning of the loadboard. The five metrics are Complex PCB Design, Integration, Frequency Domain Analysis, Run-Time, and Computing Resources.

1. **Complex PCB Design:** This metric assesses the tool's ability to manage complex PCB layouts involving multiple layers, numerous components, and interconnections.

2. **Integration:** This metric evaluates how well the parasitics extraction tool integrates with PCB design tools.

3. **Frequency Domain Analysis:** This metric assesses the tool's ability to perform frequency-domain analysis to capture the effects of frequency on parasitics.

4. **Run-Time:** This metric evaluates how long it takes for the tool to complete parasitics extraction.

5. **Computing Resource:** This metric assesses the tool's demand for computing resources, such as CPU and memory usage.

The tools will be evaluated by being assigned a grade for each metric. A tool can score four grades, which are defined below.

- **"+ +":** This represents optimal performance, which means no more improvement is required.

- **“+”:** This means satisfactory performance, indicating that some improvements can be made, but it is still acceptable.
- **“–”:** This represents below marginal performance, indicating a need for improvement to reach an adequate level.
- **“– –”:** This means poor performance, indicating a need for significant improvements.

Tool Comparison and Analysis

In this section, the four tools for parasitics extraction are evaluated based on the metrics. In Table 3.2, the grades for each tool across the five metrics can be seen. All three tools except Keysight ADS got a (+ +) grade in handling complex designs and integration with PCB design tools, as they can effectively handle multi-layer designs and integrate seamlessly with PCB design tools. KeysightADS scored a lower grade, (+), in complex designs and integration due to its limited handling of large designs and lack of interoperability with design tools. PowerSI proved to be the best tool for frequency domain analysis due to its ability to account for frequency-related changes in parasitic behaviour. At the same time, Sigrity Aurora scored the lowest in this metric, as the frequency was not considered while extracting parasitics. Siemens and PowerSI scored the worst in run-time and computing resource efficiency as they were the best in accuracy, which is why it takes several resources and time to extract parasitic. Sigrity Aurora scored the best in these parameters due to it being a frequency-independent tool and less accurate. Even though all of the tools have their advantages, due to constraints at NXP, which require Cadence tools to maintain compatibility across all stages of the design and testing process, only Cadence tools can be used. To compare the two cadence tools, it is important to determine the effects of frequency on transmission line parasitics. As the frequency increases, the parasitics see a change in their values, as described below.

- **Resistance:** As frequency increases, the resistance of the transmission line also increases. For example, the resistance for 1 mm of a transmission line at 1 Hz, which is 0.2 Ω, increases to around 0.6Ω at 5 GHz [54].
- **Inductance:** The inductance decreases with increased frequency. A 0.45 nH at 1 Hz for 1 mm of a transmission line goes down to around 0.38 nH at 5 GHz [54].
- **Capacitance:** As frequency increases, the capacitance decreases slightly. For example, a 0.12 pF capacitance for 1 mm of the transmission line decreases to around 0.1 at 5 GHz [54].
- **Conductance:** The conductance increases with the frequency, going from 0.05 S at 1 Hz to 0.1 at 5 GHz for 1 mm of the transmission line [54].

This change in the parasitic values shows that the frequency has a considerable effect on R, L, G, and C. These changes can then have a more significant effect on the further calculations for characteristic impedance. To account for these changes in parasitics extraction, it is critical to use a tool to extract the parasitics based on the frequency. So, due to the need to include frequency domain analysis for parasitics extraction, PowerSI is the better tool amongst the two cadence tools for extracting loadboard parasitics.

3.2.3. Digital Twinning Limitations

Despite the methodologies and EDA tools discussed for the digital twinning of loadboards, significant limitations still highlight the need for an automated framework to manage these processes effectively. These limitations include the complexity of the tools, the challenges in combining their outputs, and the specific requirements of adapting the results to a virtual testing environment like AMS-VT.

One of the most significant limitations is that no available EDA tools can combine the parasitics extracted from the PCB layout with the ideal netlist generated from the schematic [42]. This integration is crucial to generate a final netlist that accurately reflects the real-world behaviour of the loadboard. Without this integration, the netlist remains ideal, failing to capture the parasitic effects that can impact voltage and timing. The absence of such a framework means that this process must be done manually, which is time-consuming and prone to errors [42]. The other limitation is that the EDA tools are highly specialised and complex. Engineers must spend significant time setting up and running simulations

Table 3.2: Comparison of Parasitics Extraction Tools

Tool	Complex PCB Design	Integration	Frequency Domain Analysis	Run-Time Efficiency	Computing Resource Efficiency
Cadence Sigrity Aurora	+ +	+ +	– –	+ +	+ +
Cadence Sigrity PowerSI	+ +	+ +	+ +	–	–
Keysight ADS	+	+	+	+	–
Siemens HyperLynx	+ +	+ +	+	–	–

accurately [55]. Even if engineers complete the setup, a manual setup can lead to substantial errors, ultimately affecting the accuracy of the digital twinning process. Combining the parasitics into an ideal netlist is a problematic process that also requires a lot of time and knowledge [55]. Finally, since loadboards can be different for multiple projects, the engineers would have to make a new setup for every loadboard design, which is inefficient. Even then, the netlist would not work in AMS-VT since it will be multisite, whereas it can only handle a single-site netlist.

These limitations highlight the critical need for an automated framework to manage the digital twinning of loadboards. Such a framework would automate the integration of parasitics with the ideal netlist, ensuring the results are formatted correctly for AMS-VT and reducing the dependency on error-prone manual processes. By implementing this framework, we can significantly improve the accuracy and efficiency of the digital twin process, leading to more reliable virtual testing and faster time-to-market for semiconductor chips.

4

Framework for Accurate Loadboard Modeling

This chapter derives the requirements based on the limitations of the current methodology and then proposes a new framework for accurate loadboard modelling to meet the derived requirements. Section 4.1 derives the framework's requirements by analysing the limitations of the current methodology and operating conditions of the loadboard in the real-world. Finally, Section 4.2 introduces the framework proposed in this thesis to generate an accurate loadboard model and meet the requirements derived in 4.1.

4.1. Derivation of Requirements for Loadboard Modelling

There is a need for accurate loadboard modelling to predict site-to-site voltage variations, timing delays, and other critical parameters. The current methodology relies on manually creating an ideal netlist, which lacks accuracy due to parasitics and the effects of external parameters not being included. This can lead to discrepancies between simulation and real-world performance. By analysing the limitations of the current methodology and testing conditions under which the loadboard operates, some primary factors the framework must adhere to are derived. These factors include voltage accuracy, timing accuracy, integration with AMS-VT, automation level, component modelling, generalization, and execution speed.

4.1.1. Limitations of the Current Methodology

The current methodology to generate a loadboard is to create a single-site ideal netlist for the loadboard manually. The limitations of this methodology are:

1. **Inaccuarte voltage predictions:** The model does not account for site-to-site voltage variations since it is an ideal netlist. This leads to simulation results that do not reflect real-world behaviour, which leads to inefficient test program debugging.

2. **Neglect of Operating Conditions:** Frequency and temperature effects are not considered. These effects considerably impact voltage, and due to the absence of their inclusion, there is a mismatch between simulated and physical behaviour.

3. **Timing Inaccuracy:** Lack of parasitics in the model results in unaccounted timing delays due to the physical characteristics of the loadboard. This results in an inaccurate calculation of the pulse width, which gave an error of 1.6% when compared to physical measurement.

4. **Integration Challenges:** Manual netlist generation may lead to errors while simulating in AMS-VT.

5. **Manual Process:** The process of creating the ideal netlist is completely manual, leading to a high potential for human errors.

6. **Ideal Component Modelling of Relays:** The physical characteristics of relays are not considered as they are modelled as an ideal switch. So, their effect on voltage and timing is not taken into account, which leads to inaccuracy between simulation and real-world results.

7. **Lack of Generalization:** The methodology is not adaptable to different loadboard designs, especially unique configurations involving motherboards and daughterboards.

4.1.2. Operating Conditions Impact

The loadboard operates under varying conditions of temperature and frequency, which significantly impact the behaviour of parasitic elements:

- **Temperature:** The relation between resistance and temperature is linear, which means resistance increases linearly with temperature; this can also be seen in the resistance versus temperature graph shown in Figure 4.1. The graph demonstrates that the resistance increases from approximately 0.65 to 0.9 ohms as the temperature rises from 0 to 100. This effect of temperature on the resistance must be modelled accurately in the loadboard model to ensure the virtual model mirrors the real-world conditions.

- **Frequency:** As frequency increases, various parasitic components such as *Resistance* (R), *Inductance* (L), *Capacitance* (C), and *Conductance* (G) exhibit different behaviours.

  1. **Resistance:** As frequency increases, the resistance of the transmission line also increases. For example, the resistance for 1 mm of a transmission line at 1 Hz, which is 0.2 Ω, increases to around 0.6Ω at 5 GHz [54].
  2. **Inductance:** The inductance decreases with increased frequency. A 0.45 nH at 1 Hz for 1 mm of a transmission line goes down to around 0.38 nH at 5 GHz [54].
  3. **Capacitance:** As frequency increases, the capacitance decreases slightly. For example, a 0.12 pF capacitance for 1 mm of the transmission line decreases to around 0.1 at 5 GHz [54].
  4. **Conductance:** The conductance increases with the frequency, going from 0.05 S at 1 Hz to 0.1 at 5 GHz for 1 mm of the transmission line [54].

  This behaviour of the parasitic elements due to frequency affects the parasitic elements, affecting the impedance calculated using these elements and angular frequency as shown in Equation 2.1. The effect of frequency on impedance can be seen in Figure 4.2. According to the graph, there is a slight decrease in the trace impedance when the frequency goes from 10 MHz to 100 MHz. This change in impedance can lead to impedance mismatch, which can lead to reflections.

4.1.3. Deriving Requirements

Based on the limitations and the impact of the operational conditions, the following requirements for the loadboard modelling framework are derived.

1. **Voltage Accuracy:** The model should be able to predict site-to-site voltage variations within a tolerance of ± 0.1 V for a 5 V input signal, ensuring less than 2% deviation.

2. **Timing Accuracy:** The loadboard model should account for the timing delays due to the physical characteristics of the loadboard, with a timing accuracy within 1% for the pulse width.

3. **Operating Conditions:** The model must include the effects of temperature (from $-40°$C to $125°$C) as that is the operating range for a loadboard. The model should also account for the effects on parasitics and impedance due to frequency.

4. **Component Modelling:** The effects of the relay on voltage and timing should be considered. The relay model should accurately represent the characteristics such as output capacitance, on-state resistance, off-state leakage, and switching times.

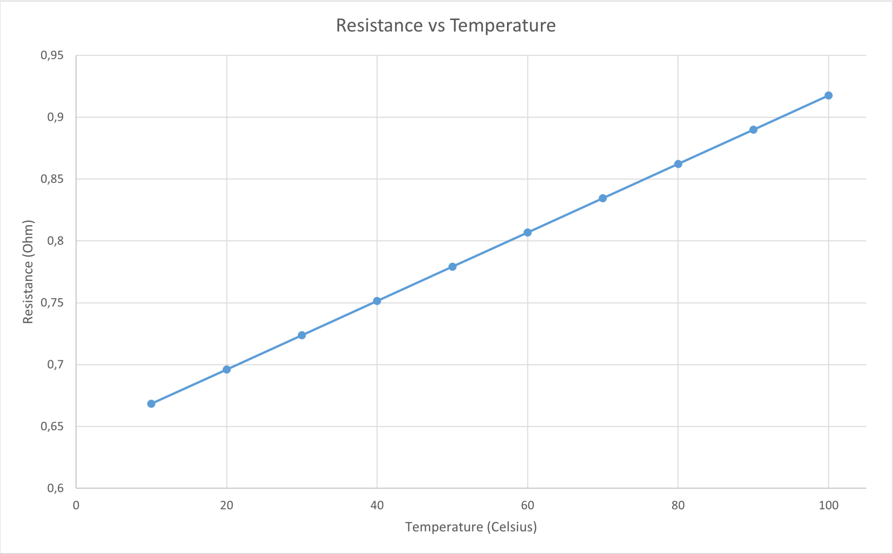


Figure 4.1: Resistance vs Temperature Graph

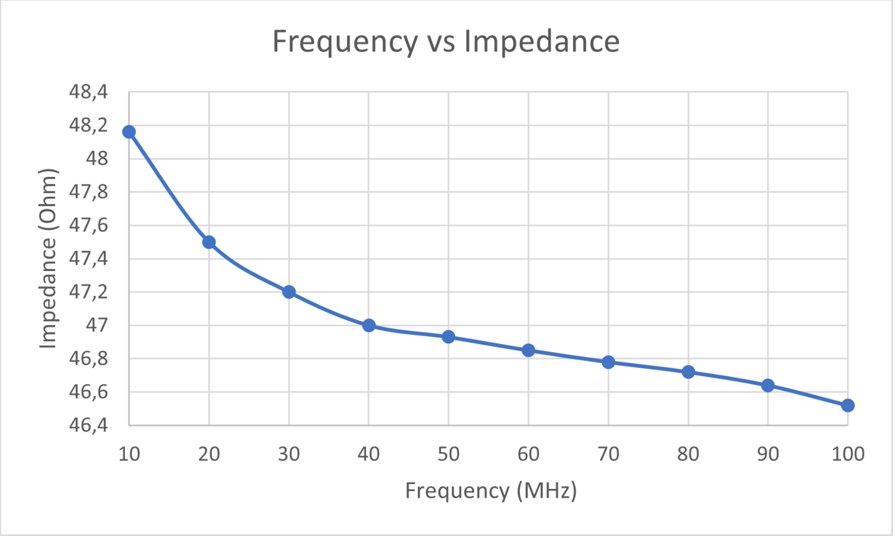


Figure 4.2: Frequency vs Impedance

Table 4.1: Inputs and Outputs for Ideal Netlist Generation

Inputs (User)	Outputs
Allegro Netlists	Single-site netlist (ideal)
Module Name	Channels list (to be used in stage 2)
Pin Mapping	
Connector Mapping	

5. **Automation Level:** The steps in the framework should be completely automated to reduce human errors. The ideal framework would automate schematic conversion to single-site netlist, parasitic extraction, and parasitics integration, thereby minimizing manual intervention.

6. **Integration with AMS-VT:** The model generated by the framework should be able to integrate into the AMS-VT environment. For example, the model should be single-site as AMS-VT can only simulate one site at a time.

7. **Generalization:** The framework should be able to handle the different designs of the loadboard without any manual changes. Loadboards can have different complexity based on the project requirements, and there can also be cases where a loadboard is a combination of motherboard and daughterboard. The framework must work for all these different configurations of the loadboard.

8. **Speed of Execution:** With automation, the framework should complete the run within 2 hours to not hinder any project timelines.

4.2. Overview of Proposed Framework

This section introduces the proposed framework to address the limitations in the current methodology to create an accurate loadboard model. This framework is designed to integrate the parasitics and other physical effects of the loadboard into a netlist that can be seamlessly used within the AMS-VT environment. The framework is structured into three primary stages: Ideal Netlist Generation, Parasitics Extraction, and Parasitics-Reflection Integration. These stages ensure that the final model accurately reflects the physical characteristics of the loadboard, thereby improving the accuracy and reliability of pre-silicon validation.

4.2.1. Key Stages of the Framework

The three critical stages of the framework are briefly explained in this section.

Ideal Netlist Generation

The first stage involves generating a formatted, single-site netlist from the schematic of the loadboard. This process uses Cadence Allegro to capture the schematic and generate a multisite ideal netlist in Verilog. This Allegro netlist is then formatted using a Python script that converts the multisite netlist to a single site based on the user input of the site number. This single-site netlist is then converted to Verilog-AMS by adding an analog block to handle both digital and analog signals of the loadboard. This stage includes validation steps to ensure the netlist meets the integration requirements. The ideal netlist is checked for errors in its structure and formatting. This validation ensures that the netlist can be integrated with the existing AMS-VT environment without issues. The inputs and outputs of this stage are shown in Table 4.1. This stage addresses **Requirement R6 (Integration with AMS-VT)** and contributes to **Requirement R5 (Automation Level)** by automating single-site ideal netlist generation from the schematic.

Parasitics Extraction

Once the ideal netlist is generated, the next stage involves extracting the parasitics of the loadboard channels. This stage uses the PowerSI Cadence tool to extract the parasitics of the channels. These channels are taken by the formatted ideal single-site netlist generated in the first stage. The framework also takes the frequency of the project by the user and feeds it to the Cadence tool to extract the parasitics based on the frequency. The cadence tool setup is automated using Python and *Tool*

Table 4.2: Inputs and Outputs for Parasitics Extraction

Inputs	Outputs
Frequency (User)	R, L, G, C of each trace
Channels List (1st Stage)	

Table 4.3: Inputs and Outputs for Parasitics and Reflection Integration

Inputs	Outputs
Temperature (User)	Final Accurate Netlist
DUT Impedance (User)	
Ideal Netlist (1st Stage)	
R, L, G, C (2nd Stage)	

Command Language (TCL) commands. This stage provides the channels' parasitic resistance, capacitance, inductance, and conductance in an Excel sheet. The inputs and outputs of this stage are shown in Table 4.2. This stage addresses requirement: **R3 (Operating Conditions)** by considering frequency effects on parasitics and enhances **Requirement R5 (Automation Level)**.

Parasitics and Reflection Integration

The final stage involves the integration of parasitics and reflections into the ideal netlist. A Python script calculates the impedance of each channel using the parasitics obtained in Stage 2 and adjusts the resistance value based on Equation 5.1. In this equation, R_0 is the resistance at a reference temperature, R(T) is resistance at temperature T, and α is the temperature coefficient of resistance for the conductive material.

$$R(T) = R_0\,(1 + \alpha(T - T_0)) \tag{4.1}$$

After calculating the impedance of the channel of the loadboard, it is then used to find impedance discontinuities along the way of propagation. The script calculates the reflection coefficient for the impedance mismatch between ATE and loadboard, mismatch due to passive components, and between loadboard and DUT. Suppose a mismatch is found at any point. In that case, the framework calculates the amount of reflection in that particular channel, and its effects on the voltage are then included in the final netlist using a Verilog module. Using a Python script, the parasitics and reflections are added to the ideal netlist as a Verilog-AMS module. After this stage, an accurate model of the loadboard is generated that can predict the timing delays and site-to-site voltage variations. The inputs and outputs of this stage are shown in Table 4.3. This stage addresses requirements: **R1 (Voltage Accuracy)**, **R2 (Timing Accuracy)**, **R3 (Operating Conditions)**, and contributes to **R5 (Automation Level)**.

4.2.2. Component Modelling

The basic components, such as resistors and capacitors, were already modelled in the existing methodology. But components like *Solid State Relays* (SSR) have ideal models in the current scenario that work like a perfect switch. SSR is an electronic switching device that has no moving parts. It consists of an input circuit, a drive circuit, and an output circuit, which also acts as the switch [56]. The framework includes detailed modelling of SSR as they have factors such as output capacitance, On-State Resistance, Off-State Current Leakage, and switching time that can impact signal integrity. The values at room temperature ($25^{\circ}C$) and definition of these factors are given in Table 4.4 [57]. The framework generates a model for SSR that incorporates its characteristics. The new model ensures that the physical behaviour of the SSR is captured accurately during simulations to mirror their effects on voltage and timing. This addresses requirement: **R4 (Component Modeling)**.

4.2.3. Platform

The entire framework was automated using Python since it has robust libraries for automation, data processing, and integration tasks [58]. Pyhton was also used to format the ideal netlist generated by Allegro to a single-site netlist as required by AMS-VT. This automation significantly reduces the manual

Table 4.4: SSR Relay Characteristics

Parameter	**Value**	**Unit**	**Definition**
Output Capacitance	27	pF	The capacitance the relay must drive to charge the MOSFET.
On-State Resistance	0.8	Ohms	Resistance that occurs when the relay is turned on.
Off-State Current Leakage	10	nA	Small amount of current can still flow through the relay even when turned off.
Switching On Time	0.12	ms	The time it takes for the relay to switch from off to on.
Switching Off Time	0.08	ms	The time it takes for the relay to switch from on to off.

effort needed for the current methodology, minimizing the potential for human error and speeding up the overall process. Verilog-AMS was used as the language for the netlist due to its ability to model both analog and digital components, making it ideal for the mixed-signal nature of loadboards. Verilog-AMS also allows for seamless integration of the netlist with AMS-VT.

5

Framework Implementation

Following the overview of the proposed framework in Section 4.2, this chapter describes the implementation of the framework in detail. The first stage in which a single-site ideal netlist is generated is explained in Section 5.1. The implementation is continued with Section 5.2, where the methodology to extract the parasitics of the loadboard is explained. Section 5.4 continues with the model for calculating the impedance, reflection coefficient, and integration of parasitics and reflection in the ideal netlist. Finally, Section 5.5 explains the automation methodology, which combines the algorithms of all three stages.

5.1. Ideal Netlist Generation

This section describes the detailed implementation of the first stage of the framework that generates an ideal single-site netlist in Verilog-AMS from a loadboard schematic. The first stage can be divided into two steps, which are as follows.

1. **Schematic to Multisite Netlist:** The first step is transforming the loadboard schematic into a multisite netlist. During this step, an ideal netlist is created in Verilog that reflects all the physical connections and components of the loadboard. This generated multisite netlist cannot be integrated with AMS-VT since it can only handle single-site simulations.

2. **Multisite to Single-Site Netlist:** After getting the initial multisite netlist, the next step is to convert it to a single-site netlist, as AMS-VT can only simulate one site at a time. After converting it to a single-site, the netlist is converted to Verilog-AMS to handle analog, mixed signals.

5.1.1. Schematic to Multisite Netlist

In this step, a complete schematic of the loadboard is used to generate a preliminary netlist. In Section 3.2, four prominent industrial tools, Cadence Allegro, EasyEDA, Altium Designer, and Autodesk Fusion, were introduced that can convert a schematic to its equivalent ideal netlist. Among these tools, Cadence Allegro proved to be the best tool available due to NXP constraints, which could be used for ideal netlist generation.

The loadboard schematic contains all the details regarding its functionality. The first few pages of the schematic file comprise all the signals the loadboard includes, which are made as block components where the signals are given numbers starting from one. While designing, the engineer can use any signal from any group based on the project's requirements. The unused signals are still present in the schematic as they are a part of the loadboard, even though they are not used. One of the signal description blocks named "M31" is shown in Figure 5.1.

After the signal description blocks, the circuit diagrams of the sites are present. A schematic of Site-1 for a loadboard design is shown in Figure 5.2. A single page can have multiple sites based on the complexity of the loadboard. All the sites are identical, the only difference being the signals used and the numbering of passive components. The main aim of this step is to convert this complete schematic into a netlist format, which can then be used for further formatting.

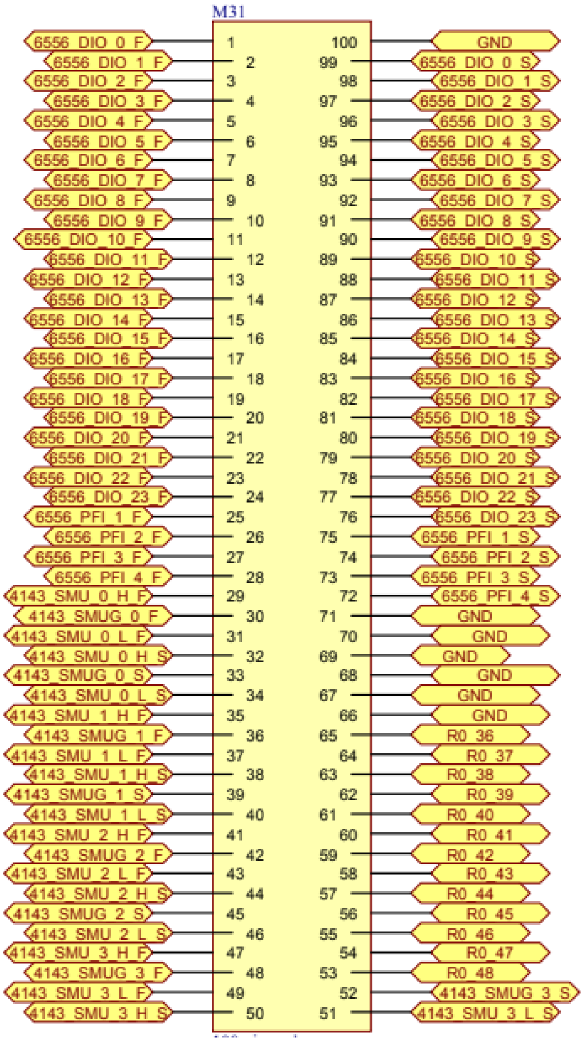

M31
6556_DIO_0_F 1 100 GND
6556_DIO_1_F 2 99 6556_DIO_0_S
6556_DIO_2_F 3 98 6556_DIO_1_S
6556_DIO_3_F 4 97 6556_DIO_2_S
6556_DIO_4_F 5 96 6556_DIO_3_S
6556_DIO_5_F 6 95 6556_DIO_4_S
6556_DIO_6_F 7 94 6556_DIO_5_S
6556_DIO_7_F 8 93 6556_DIO_6_S
6556_DIO_8_F 9 92 6556_DIO_7_S
6556_DIO_9_F 10 91 6556_DIO_8_S
6556_DIO_10_F 11 90 6556_DIO_9_S
6556_DIO_11_F 12 89 6556_DIO_10_S
6556_DIO_12_F 13 88 6556_DIO_11_S
6556_DIO_13_F 14 87 6556_DIO_12_S
6556_DIO_14_F 15 86 6556_DIO_13_S
6556_DIO_15_F 16 85 6556_DIO_14_S
6556_DIO_16_F 17 84 6556_DIO_15_S
6556_DIO_17_F 18 83 6556_DIO_16_S
6556_DIO_18_F 19 82 6556_DIO_17_S
6556_DIO_19_F 20 81 6556_DIO_18_S
6556_DIO_20_F 21 80 6556_DIO_19_S
6556_DIO_21_F 22 79 6556_DIO_20_S
6556_DIO_22_F 23 78 6556_DIO_21_S
6556_DIO_23_F 24 77 6556_DIO_22_S
6556_PFI_1_F 25 76 6556_DIO_23_S
6556_PFI_2_F 26 75 6556_PFI_1_S
6556_PFI_3_F 27 74 6556_PFI_2_S
6556_PFI_4_F 28 73 6556_PFI_3_S
4143_SMU_0_H_F 29 72 6556_PFI_4_S
4143_SMUG_0_F 30 71 GND
4143_SMU_0_L_F 31 70 GND
4143_SMU_0_H_S 32 69 GND
4143_SMUG_0_S 33 68 GND
4143_SMU_0_L_S 34 67 GND
4143_SMU_1_H_F 35 66 GND
4143_SMUG_1_F 36 65 R0_36
4143_SMU_1_L_F 37 64 R0_37
4143_SMU_1_H_S 38 63 R0_38
4143_SMUG_1_S 39 62 R0_39
4143_SMU_1_L_S 40 61 R0_40
4143_SMU_2_H_F 41 60 R0_41
4143_SMUG_2_F 42 59 R0_42
4143_SMU_2_L_F 43 58 R0_43
4143_SMU_2_H_S 44 57 R0_44
4143_SMUG_2_S 45 56 R0_45
4143_SMU_2_L_S 46 55 R0_46
4143_SMU_3_H_F 47 54 R0_47
4143_SMUG_3_F 48 53 R0_48
4143_SMU_3_L_F 49 52 4143_SMUG_3_S
4143_SMU_3_H_S 50 51 4143_SMU_3_L_S

Figure 5.1: Signals Description in Schematic

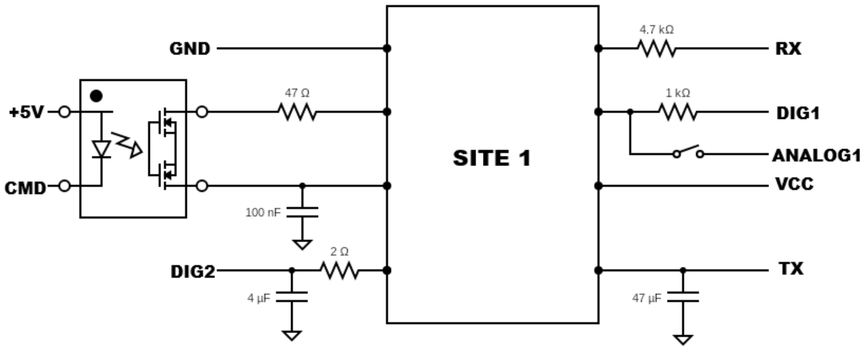


Figure 5.2: Single site loadboard schematic

The complete schematic made by designers is loaded in the Cadence Allegro tool. After loading the schematic, Allegro can generate a netlist in multiple formats. As discussed in Section 3.1, Verilog is the most suitable HDL for integration with AMS-VT. Verilog language was chosen as the netlist language for seamless integration with the current virtual testing methodology. After selecting the appropriate format for the netlist, Allegro creates a multisite netlist within seconds. The generated netlist contains all the sites and components for signal description, such as M31.

5.1.2. Multisite to Single-Site Netlist

Allegro created a multisite netlist, but several issues arose when that netlist was used to do the simulations. The main issue was that Allegro included the unused components, such as “M31”, in the netlist, which gave an error as they were undeclared. Since AMS-VT could only handle a single-site, the signals of only one site were declared in the simulation. Another issue was the netlist’s inability to handle mixed signals due to its Verilog format. This issue restricted the netlist usability as a loadboard consists of analog and digital signals. The simulation environment flagged numerous modules that were either unnecessary for the simulation or not properly declared within the netlist. So, to resolve these issues, an algorithm was developed to convert the multisite to a single-site netlist and add an analog block to handle analog mixed signals.

The flowchart of the algorithm is shown in Figure 5.3. The algorithm takes five inputs from the user to start the netlist formatting, described below.

- **Site Number:** Since the main aim of this algorithm is to convert the multisite Allegro netlist to a single site, it is necessary to determine the site on which the netlist is to be created. This is a number input by the user, such as “1” or “2”, that tells the algorithm the site on which the single-site netlist will be created.

- **Dual-Board:** This input determines whether the loadboard comprises two different boards. The user needs to give a “Yes” or “No” input, which will determine the configuration of the loadboard based on which the algorithm will run.

- **Connector Mapping:** This Excel file contains information on how the two boards are connected when a loadboard is made of two different boards. This determines the signals from each board that connect to make a complete loadboard for that project.

- **PIN and Module Name:** This is an input given by users where they can determine the output PIN and module name for their convenience.

- **Allegro Netlist Location:** This contains the path to the netlist generated by Allegro. If this is a dual-board configuration, then two paths will be provided that contain the location of Allegro netlists for both boards.

After determining these inputs, the user can run the complete algorithm. This algorithm was developed in Python since it has robust libraries for automation, data processing, and integration tasks. As shown in Figure 5.3, the algorithm can be further divided into three stages to generate a single-site ideal netlist. The stages are described below in detail.

1. **Stage-1 (Unique Board Design):** This is the first stage of the algorithm whose steps are highlighted in yellow in Figure 5.3. This step runs based on the “Dual-Board” input provided by the user. This step deals with cases where a loadboard of a project is made of two different boards. If the user provides “No” as the input to the “Dual-Board” parameter, the step is skipped as the allegro netlist generated is the final one and does not require any modifications. If the input is “Yes”, then this step takes the location of the allegro netlists and the connector mapping Excel file (shown in Figure 5.4) containing information on how the two boards are connected. Based on the Excel file, the algorithm searches for the “X4” component in both the netlists. Then, it connects the connections mentioned in the file, such as PIN 18 of the Motherboard will be connected to PIN 15 of the Daughterboard. To connect these two, the algorithm writes a new module named “alias_bit” that connects the two signals by indicating a zero voltage drop between them. Figure 5.5 shows motherboard and daughterboard signals on the left and the final output on the right. The PINs 18 from the motherboard and 15 from the daughterboard, marked by orange, are connected using

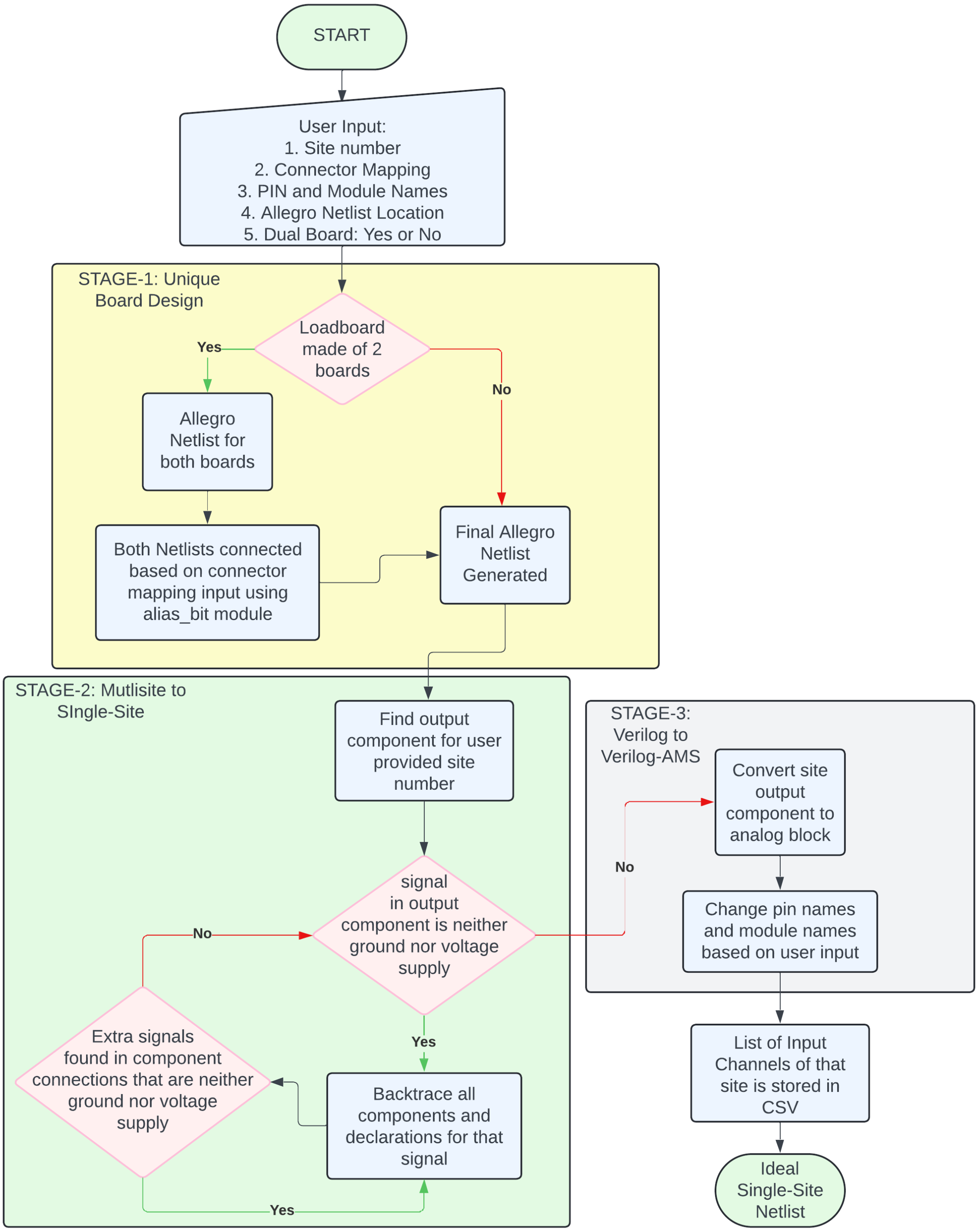
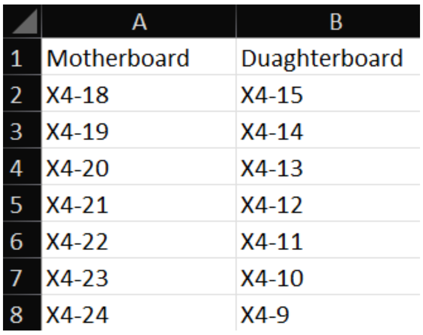

START
User Input:
1. Site number
2. Connector Mapping
3. PIN and Module Names
4. Allegro Netlist Location
5. Dual Board: Yes or No
STAGE-1: Unique Board Design
Loadboard made of 2 boards
Yes
No
Allegro Netlist for both boards
Both Netlists connected based on connector mapping input using alias_bit module
Final Allegro Netlist Generated
STAGE-2: Mutlisite to SIngle-Site
Find output component for user provided site number
signal in output component is neither ground nor voltage supply
No
Yes
Extra signals found in component connections that are neither ground nor voltage supply
Backtrace all components and declarations for that signal
Yes
STAGE-3: Verilog to Verilog-AMS
No
Convert site output component to analog block
Change pin names and module names based on user input
List of Input Channels of that site is stored in CSV
Ideal Single-Site Netlist

Figure 5.3: Mutlisite to Single-Site Flowchart



	A	B
1	Motherboard	Duaghterboard
2	X4-18	X4-15
3	X4-19	X4-14
4	X4-20	X4-13
5	X4-21	X4-12
6	X4-22	X4-11
7	X4-23	X4-10
8	X4-24	X4-9

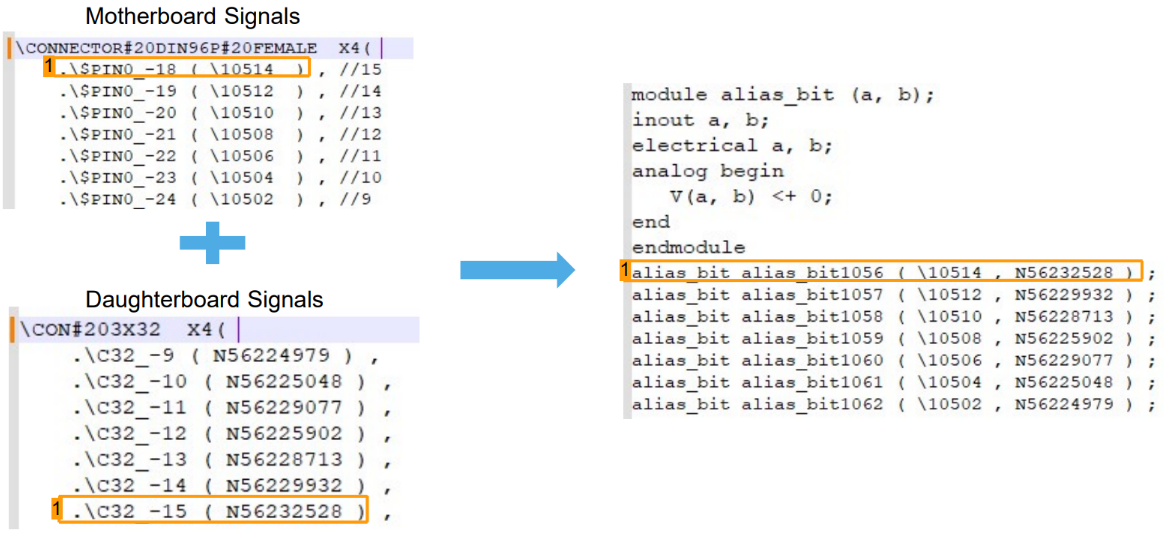
Figure 5.4: Connector Mapping Excel File

the alias_bit module, and an orange box on the right-hand side of the image marks the resulting connection. After running this stage, the algorithm makes one combined netlist where both the netlists are connected.

2. **Stage-2 (Multisite to Single-Site):** After stage-1 of the algorithm, a final Allegro netlist is generated. The next step is to convert this netlist to a single-site netlist. This step is highlighted by green in Figure 5.3; this works as a recursive backtracking algorithm based on the user input of the site number. The user provides the site number whose netlist will be created; based on the site number, the algorithm searches for the component with that particular site number. This component contains the final output signals of that specific site, and the algorithm stores all the signals of that component in a list. Based on the list of signals, the algorithm backtraces the connection that led to that particular output. This backtracking algorithm works only if the signal in the list is neither ground nor voltage supply, as this signal means that the output is directly connected to either ground or voltage supply, so there is no need for further tracing. If the signal is neither ground nor voltage supply, then the algorithm traces all the components and connections that the signal has in the netlist. If a component is found, the algorithm stores it in a set and then checks its connections. If the connections of the components have new signals, then the algorithm will trace the components and declarations of the extra signal. This loop continues until all the components and their connection to that output are included, after which the algorithm moves on to the following output signal from the list. If the output has no subsequent connection, the algorithm will automatically move on to the next output in the list. The algorithm continues this operation for all the signals so that all connections and components of that particular site are included in the netlist. When no signal remains in the list, the algorithm writes all the declarations and components in a new file. After writing all the connections stored previously, an ideal single-site netlist is generated from an Allegro multisite netlist.

3. **Stage-3 (Verilog to Verilog-AMS):** After completing stage-2 of the algorithm, a single-site ideal netlist is created in Verilog format. The next step is to convert this netlist into Verilog-AMS to handle analog mixed signals. This step is the final step and is highlighted by grey in Figure 5.3. After creating the single-site netlist, the algorithm takes the output component, the site number component. Then, it stores the output signal and its connections in a list. After storing all the signals and their connections, the algorithm adds an analog block at the end of the netlist. It adds those connections to show that the voltage drop between them is zero, indicating the signals are connected. After converting the component to an analog block, the algorithm checks if the user has provided output PIN names as input. If an input is provided, the algorithm changes the output connection to the desired name based on the input. This makes it easier for the users to differentiate between the projects and makes it easy to understand.

After carefully going through the three-stage algorithm, the Allegro netlist was converted into an ideal single-site netlist easily integrated with AMS-VT. After running this algorithm, the user generates an ideal single-site netlist, which can be used for simulation if there are no accuracy requirements.



```
\CONNECTOR#20DIN96P#20FEMALE  X4(
    .\$PIN0_-18 ( \10514  ) , //15
    .\$PIN0_-19 ( \10512  ) , //14
    .\$PIN0_-20 ( \10510  ) , //13
    .\$PIN0_-21 ( \10508  ) , //12
    .\$PIN0_-22 ( \10506  ) , //11
    .\$PIN0_-23 ( \10504  ) , //10
    .\$PIN0_-24 ( \10502  ) , //9
```

```
\CON#203X32  X4(
    .\C32_-9 ( N56224979 ) ,
    .\C32_-10 ( N56225048 ) ,
    .\C32_-11 ( N56229077 ) ,
    .\C32_-12 ( N56225902 ) ,
    .\C32_-13 ( N56228713 ) ,
    .\C32_-14 ( N56229932 ) ,
    .\C32_-15 ( N56232528 ) ,
```

```
module alias_bit (a, b);
inout a, b;
electrical a, b;
analog begin
   V(a, b) <+ 0;
end
endmodule
alias_bit alias_bit1056 ( \10514 , N56232528 ) ;
alias_bit alias_bit1057 ( \10512 , N56229932 ) ;
alias_bit alias_bit1058 ( \10510 , N56228713 ) ;
alias_bit alias_bit1059 ( \10508 , N56225902 ) ;
alias_bit alias_bit1060 ( \10506 , N56229077 ) ;
alias_bit alias_bit1061 ( \10504 , N56225048 ) ;
alias_bit alias_bit1062 ( \10502 , N56224979 ) ;
```

Figure 5.5: Connecting Motherboard and Daughterboard Signals

Table 5.1: PowerSI Parasitics for different frequencies

Frequency (MHz)	R (Ω)	L (nH)	C (pF)	G (S)	Impedance (Ω)
10	0.922	63.2	27.951	0.0000675	47.67
20	0.995	62.7	27.947	0.000135	47.25
30	1.083	62.2	27.941	0.000202	46.98
40	1.182	61.8	27.932	0.000269	46.77
50	1.287	61.4	27.921	0.000336	46.61

5.2. Parasitics Extraction

An ideal single-site netlist was successfully generated in section 5.1. This netlist can be used for simulation, but it cannot predict site-to-site variations. The reason behind this is the absence of consideration of parasitics due to the physical layout of the loadboard, as explained in section 2.4. To include the parasitic in the netlist, the first step is to extract them (R, L, G, C) from the physical layout of the loadboard. An example layout file for a loadboard is shown in Figure 5.6.

Specialised EDA tools can simulate a PCB layout and extract their parasitics. In section 3.2, four prominent tools were introduced that can be used for parasitics extraction, which are Cadence Sigrity Aurora, Cadence Sigrity PowerSI, Keysight ADS, and Siemens hyperLynx. Due to the constraints at NXP, only cadence tools were available for parasitics extraction. Among the two cadence tools, PowerSI proved to be the best tool as it could extract parasitics based on frequency, and frequency has a considerable effect on R, L, G, and C [54]. An NXP loadboard layout was uploaded to the PowerSI tool to check the impact of frequency on R, L, G, and C of a channel of the loadboard. The parasitics extraction for one channel of loadboard from PowerSI for multiple frequencies can be seen in Table 5.1. As frequency increases, the resistance and conductance increase, whereas inductance and capacitance decrease.

To extract parasitic parameters using PowerSI, several steps need to be done in a proper sequence. These steps require a detailed understanding of how the tool works, so to make it easier for engineers and reduce manual errors, the extraction was automated using a combination of Python scripts, *Tool Command Language* (TCL) commands, and PowerShell execution. The algorithm for the automation of parasitic extraction is shown in Algorithm 1. The first step is to read the list of channels provided as a CSV input, provided from stage 1, and store them in a list. The script then starts generating TCL commands that contain the necessary commands to set up and run the simulation for the automation of PowerSI. This starts with base commands to open the correct workspace in PowerSI for parasitics extraction, and then the command for opening the laodboard layout is added. The script then selects only the channels provided and ground nets, which are moved to a designated group to create ports. After this, the frequency of the loadboard is specified in commands based on which the parasitics will

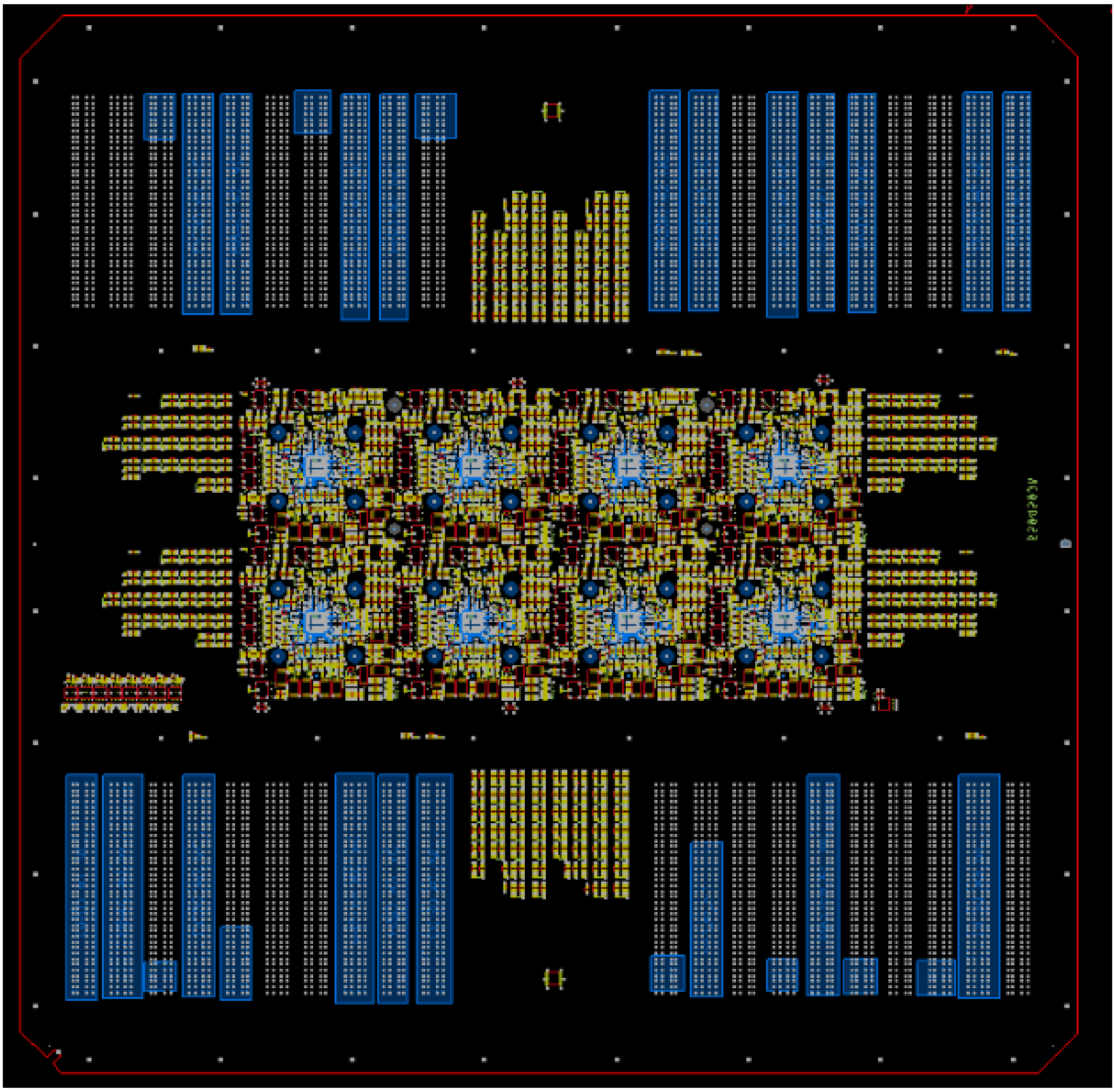


Figure 5.6: Loadboard Layout

Table 5.2: Extracted Parasitic Values for Loadboard Site 1

Net	R (Ω)	L (nH)	C (pF)	G (S)
1	0.922	63.2	27.951	0.000067455
2	1.131893	109.28	48.208	0.0000988
3	1.033932	102.38	45.948	0.0000128
4	1.186084	112.33	49.506	0.00007856
5	0.9825	94.65	38.099	0.0000054
6	1.161864	117.37	61.753	0.0000872
7	1.044306	97.884	42.967	0.0000198
8	0.808466	58.217	25.931	0.0000471

be extracted. Finally, the command for the start of the simulation and the export of results to a CSV file is generated. All these TCL commands are saved in a file, which will be executed in PowerShell. The script creates a PowerShell command that calls PowerSI with the TCL script as an argument. The PowerShell command is executed using Python's subprocess module, which runs PowerSI in batch mode and then executes the TCL script. After the complete run, a CSV file is generated with the parasitics of the channels as shown in Table 5.2.

Algorithm 1 Automated Parasitic Extraction using PowerSI

```
Input:
- channels.csv: CSV file containing the list of channels.
- layout.spd: Loadboard layout file for PowerSI.
- frequency: Frequency at which parasitics are to be extracted.
Output: Parasitic parameters ($R$, $L$, $G$, $C$) for each channel.

function ParasiticExtraction
    Step 1: Read the CSV file
    Read the csv file channels.csv.
    Store the channels in a list.

    Step 2: Generate TCL Script for PowerSI
    Initialize a list commands with base TCL commands for PowerSI.
    Include commands to open the layout file layout.spd.
    Set up the PowerSI workflow for parasitic extraction.
    Add commands to select and configure the nets for the list of channels.
    Set the simulation frequency to frequency.
    Add commands to run the simulation and export the results in a CSV file.
    Write the TCL commands to a file cadence.tcl.

    Step 3: Execute PowerSI with TCL Script
    Use PowerShell to execute PowerSI with the generated TCL script.
    Monitor the execution for completion or errors.
    After execution, R, L, G, and C for all channels are stored in a CSV file.
end function
```

5.3. Component Modelling

The passive components on the loadboard are the resistor, capacitor, and *Solid-State Relay* (SSR). Accurate modelling of passive components is crucial in loadboard modelling to ensure that the model closely reflects real-world performance. While the models of resistors and capacitors have already been implemented in Verilog-AMS, SSR requires a more detailed model to account for its physical characteristics. The initial SSR model is just a standard switch, but the SSR has on-state resistance, off-state leakage, output capacitance, and switching time. SSR is quite different from standard switches and mechanical relays as they use photodiodes to switch between channels electrically instead of switching using physical parts [57]. This difference in operation leads to certain parasitic characteristics. The parasitic values for a Panasonic relay, the most common industrial-use SSR, are mentioned below.

- **On-State Resistance (R_{on}):** It is the resistance that occurs when the relay is turned on. R_{on} is approximately $0.8\ \Omega$ at $25^{\circ}C$. This resistance increases with temperature due to a positive temperature coefficient, which is 0.006818.

- **Off-State Current Leakage:** Refers to the small amount of current that can still flow through the relay even when turned off. The current leakage sees a slight increase with increasing voltage reaching 10 nA at a voltage value of 60 V.

- **Output Capacitance (C_{out}):** It is a parameter that affects how much charge the relay needs to drive the MOSFET. This is voltage-dependent, decreasing exponentially with increased applied voltage. At 0 V, C_{out} is 27 pF, decreasing to 5.6 pF at 60 V.

- **Switching Time:** This is the time the relay takes to switch to a different state, which means the time taken for the relay to go from off-state to on-state and vice-versa. The *turn-on time* (t_{on}) is 0.12 ms and the *turn-off time* (t_{off}) is 0.08 ms.

5.3.1. SSR Model Implementation

A behavioural model in Verilog-AMS was developed to account for the above-mentioned physical characteristics and accurately represent the relay model in the netlist. The model created was a static model that does not consider the power transfer during switching, so the parasitics are only considered at the time of either on or off state and not in between switching. This model also captures the electrical behaviour of SSR by accounting for the temperature and voltage dependencies of the physical attributes.

Control Signal Handling and Switching Delays

The relay state is controlled by a voltage signal, V_{CMD}, which is applied to the control terminal of the relay. The model changes the state of the relay based on V_{CMD} and then switches the state after accounting for the appropriate delay. The *relay state* S(t) is defined in Equation 5.1, where V_{th} is the threshold voltage value set for the model above which the relay is turned on and t_{cmd} is the time when the control signal changed.

$$S(t) = \begin{cases} 1, & \text{if } V_{\text{cmd}} > V_{\text{th}} \text{ and } t \geq t_{\text{cmd}} + t_{\text{on}} \\ 0, & \text{if } V_{\text{cmd}} \leq V_{\text{th}} \text{ and } t \geq t_{\text{cmd}} + t_{\text{off}} \end{cases} \tag{5.1}$$

Voltage-Dependent Output Capacitance

This is the capacitance between the relay's output terminals; the value of the capacitance changes with voltage. This behaviour can be modelled using depletion capacitance of a reverse-biased p-n junction [59]. The junction capacitance $C(V_{\text{out}})$ is shown in Equation 5.2. Here C_o is zero-bias capacitance at 0 V, which is 27 pF, V_{out} is the voltage across the terminal, V_j is the built-in junction potential, and m is the grading coefficient, which is 0.5 [59].

$$C(V_{\text{out}}) = \frac{C_0}{\left(1 + \frac{|V_{\text{out}}|}{V_j}\right)^m} \tag{5.2}$$

The voltage across output terminals of the relay, which can be calculated by Equation 5.3, where $V_{\text{IN\_P}}$ and $V_{\text{IN\_N}}$ are the voltages at the positive and negative output terminals of the relay, respectively.

$$V_{\text{out}} = V_{\text{IN\_P}} - V_{\text{IN\_N}} \tag{5.3}$$

Based on the literature [59], the built-in junction potential, V_j, for reverse-biased p-n junction is 2.5 V. The output capacitance can be expressed as a function of voltage as shown in Equation 5.4.

$$C(V_{\text{out}}) = \frac{27\text{ pF}}{\left(1 + \frac{|V_{\text{out}}|}{2.5\text{ V}}\right)^{0.5}} \tag{5.4}$$

On-State and Off-State Resistance

The conductive path of the relay is modelled using a resistance that switches from R_{on} to R_{off} based on the relay state. The on-state resistance is calculated using the equation shown in Equation 5.5, where the Temperature is in degrees Celsius.

$$R_{\text{on}} = 0.8 \times [1 + 0.006818(Temperature - 25°\text{C})] \tag{5.5}$$

In the off-state of the relay, there is a current leakage that changes based on the voltage. This slight increase in current leakage corresponds to an *off-state resistance* ((R_{off})). The value of this resistance can be calculated by considering that the leakage at 60 V is around 10 nA. Using this data, the value of R_{off} can be calculated using Equation 5.6, which came out to be 6 GΩ.

$$R_{\text{off}} = \frac{V_{\text{off}}}{I_{\text{leak}}} = \frac{60\text{ V}}{10\text{ nA}} = 6\text{ G}\Omega \tag{5.6}$$

5.3.2. Algorithmic Representation of the SSR Model

The complete working of the relay verilog model is explained in Algorithm 2. Firstly, the model takes control voltage ($V_{\text{cmd}}(t)$) and temperature as input. The control voltage then determines the state of the relay using Equation 5.1, which determines if the relay is on or off and accounts for the switching delay. The model monitors $V_{\text{cmd}}(t)$ for changes and updates the relay state after the switching delay. After this, the temperature-dependent resistance is calculated using Equation 5.5. Then based on the relay state R_{eff}, which is the effective resistance, is decided. Next, the voltage across relay terminals is calculated using Equation 5.3, which is then used to find the effective output capacitance using Equation 5.4. Finally, the resistive and capacitive currents are calculated and added to get the total current travelling through the relay. The equivalent circuit diagram for the algorithm is shown in Figure 5.7.

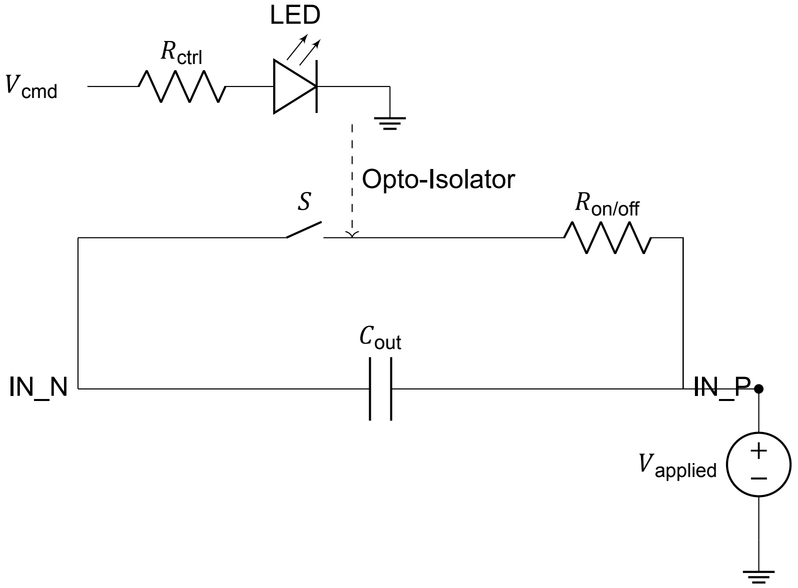


Figure 5.7: SSR Circuit Diagram

5.4. Parasitics and Reflection Integration

In loadboard, impedance mismatches can occur, leading to reflection, ultimately affecting the transmitted voltage as explained in Section 2.4. For an accurate loadboard model, it is crucial to calculate the

Algorithm 2 SSR Relay Model Algorithm

```
Input: Control voltage $V_{\text{cmd}}(t)$, Temperature $T$
Terminals: Positive Terminal IN_P, Negative Terminal IN_N
Initialize: Last control voltage $V_{\text{cmd, last}} \leftarrow V_{\text{cmd}}(0)$, relay state $S(0)$, command change time $t_{\text{cmd}} \leftarrow 0$
while Simulation is running do
    Monitor control voltage $V_{\text{cmd}}(t)$
    if $V_{\text{cmd}}(t) \neq V_{\text{cmd, last}}$ then
        Record command change time: $t_{\text{cmd}} \leftarrow t$
        Update last control voltage: $V_{\text{cmd, last}} \leftarrow V_{\text{cmd}}(t)$
        if $V_{\text{cmd}}(t) > V_{\text{th}}$ then
            Set switching delay: $delay \leftarrow 0.12$
        else
            Set switching delay: $delay \leftarrow 0.08$
        end if
    end if
    if $t \geq t_{\text{cmd}} + delay$ then
        if $V_{\text{cmd}}(t) > V_{\text{th}}$ then
            Set relay state: $S(t) \leftarrow 1$
        else
            Set relay state: $S(t) \leftarrow 0$
        end if
    end if
    Compute temperature-adjusted on-state resistance:
        $R_{\text{on}}(T) = 0.8\,[1 + 0.006818(T - 25°\text{C})]$
    Determine effective conductance $G_{\text{eff}}$:
    if $S(t) = 1$ then
        $R_{\text{eff}} \leftarrow \frac{1}{R_{\text{on}}(T)}$
    else
        $R_{\text{eff}} \leftarrow \frac{1}{6G\Omega}$
    end if
    Calculate voltage across output terminals:
        $V_{\text{out}} = V_{\text{IN\_P}} - V_{\text{IN\_N}}$
    Compute voltage-dependent output capacitance:
        $C(V_{\text{out}}) = \frac{27\text{ pF}}{\left(1 + \frac{|V_{\text{out}}|}{2.5\text{ V}}\right)^{0.5}}$
    Compute resistive current:
        $I_{\text{res}} = \frac{V_{\text{out}}}{R_{\text{eff}}}$
    Compute capacitive current:
        $I_{\text{cap}} = \frac{d}{dt}\,[C_{\text{out}}(V_{\text{out}}) \times V_{\text{out}}]$
    Compute total current:
        $I_{\text{total}} = I_{\text{res}} + I_{\text{cap}}$
    Update simulation time $t$
end while
```

Table 5.3: Parasitic Elements of Loadboard Transmission Line

Frequency (MHz)	R (Ohm)	L (nH)	C (pF)	G (Siemens)
10	0.922	63.2	27.951	0.0000675

possible reflections in the loadboard and include their effects in the netlist. This section first explains in detail the locations in the loadboard where reflections can occur and how the transmitted voltage is affected by those reflections. Then, it describes the algorithm that calculates these reflections for all the loadboard channels and adds its effects to the ideal netlist.

5.4.1. Transmission Line Parasitics and Characteristic Impedance

The transmission line of the loadboard can be characterized by R, L, G, and C, which are frequency dependent. The parasitics of a transmission line of a loadboard, which are extracted using PowerSI, are shown in Table 5.3. The accuracy of the PowerSI tool for parasitics extraction is validated in the following chapter. These parasitics can be used to calculate the characteristics impedance of the trace using Equation 5.7, where $\omega = 2\pi f$. The characteristic impedance of the trace is independent of its length and only depends on the parasitics and frequency. Using Equation 5.7 with the values given in Table 5.3, the characteristic impedance at 10 MHz can be calculated as approximately 48.16Ω. This characteristic impedance can now be used to calculate the mismatch and reflections that can occur in a loadboard transmission line further.

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \tag{5.7}$$

5.4.2. Reflection Calculation

Reflection in a transmission line can occur due to impedance mismatches at different points in a circuit. Based on the type of reflection, the reflected wave can have a destructive or constructive interference with the incident wave, thereby changing the transmitted voltage. Reflection can be quantified using *reflection coeffecient* (Γ), which can be calculated using Equation 5.8, where Z_S is the source impedance (impedance of the component from which the voltage is coming) and Z_L is the load impedance (impedance of the component where the voltage is going). After the reflection the final *transmitted voltage* (V_T) can be calculated using Equation 5.9.

$$\Gamma = \frac{Z_\text{L} - Z_\text{S}}{Z_\text{L} + Z_\text{S}} \tag{5.8}$$

$$V_\text{T} = V_\text{incident} \times (1 + \Gamma) \tag{5.9}$$

This reflection can occur at different points in a loadboard due to impedance mismatches; three primary cases where reflection appears in a loadboard, as shown in Figure 5.8, are given below.

1. **Case-1: ATE to Loadboard:** This reflection occurs due to the impedance mismatch between ATE and loadboard, Case-1 in Figure 5.8. In this case, ATE is the source impedance, and loadboard is the load impedance.

2. **Case-2: Passive Components in Loadboard:** When a passive component is present in the loadboard, the impedance of that segment changes, leading to an impedance mismatch, which can also lead to reflection. This reflection occurs at two points, first when the signal enters the segment of the transmission line where the passive component is present and when the signal exits the segment with the passive component as shown by Case-2 in Figure 5.8.

3. **Case-3: Loadboard to DUT:** This is the final point where reflection can occur when the signal goes to the DUT from the loadboard, which is Case-3 in Figure 5.8. In this case, the source is the loadboard, and DUT is the load.

These three are the primary cases where reflection can occur in a loadboard. The calculation in each case differs due to the type of reflection. The three cases are explained below with example calculations explaining their effects on the incident voltage.

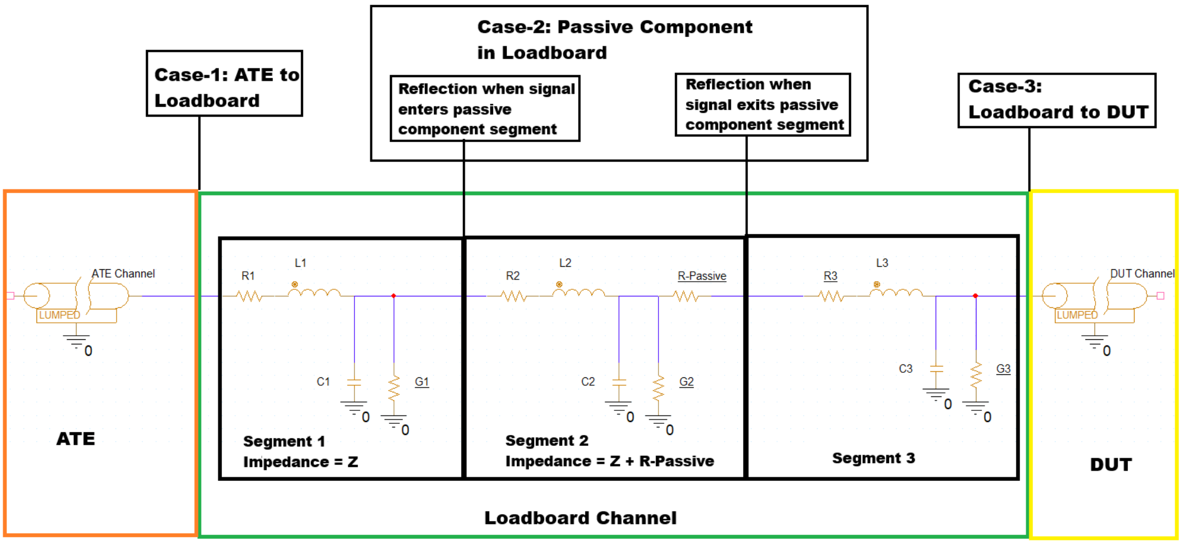


Figure 5.8: Reflection in Loadboard

Case 1: ATE to Loadboard

At the interface between ATE and loadboard, a reflection can occur if the impedance of ATE (Z_S) is different from the impedance of the loadboard transmission line (Z_L). This mismatch can change the voltage that is transmitted to the loadboard. To calculate the effect of this reflection on the voltage, the impedance of ATE is taken as 50Ω, and the impedance of the laodboard channel is 48.16Ω calculated using values in Table 5.3.

First, the reflection coefficient is calculated using the impedance values in Equation 5.8. The reflection coefficient is then calculated, which comes out to be -0.01875 as calculated in Equation 5.10. The negative sign indicates that the reflected wave is inverted and related to the incident wave, which will result in a destructive interference. After reflection, the transmitted voltage to the loadboard can be calculated using Equation 5.9. For an input of 5 V, the transmitted voltage after reflection is 4.906 V as calculated in Equation 5.11. So, due to the reflection from ATE to the loadboard, there is a slight reduction in the transmitted voltage.

$$\Gamma_{\text{ATE-LB}} = \frac{48.16\ \Omega - 50\ \Omega}{48.16\ \Omega + 50\ \Omega} = \frac{-1.84\ \Omega}{98.16\ \Omega} \approx -0.01875 \tag{5.10}$$

$$V_{\text{LB}} = V_{\text{S}} \times (1 - 0.01875) = 0.98125 \times 5\ \text{V} = 4.906\ \text{V} \tag{5.11}$$

Case 2: Passive Components in Loadboard

Passive components such as resistors, capacitors, and relays introduce impedance discontinuities in the loadboard channel, which leads to reflections. These components change the characteristic impedance of the segment of the transmission line where they are present, leading to mismatches between the previous segment and the passive component segment. The reflections occur at two points due to passive components, first when the signal enters and leaves the component. So, to accurately calculate the reflections due to passive components, the transmission line needs to be divided into three segments as shown in Figure 5.8.

1. **First Segment:** This segment is not impacted due to passive components; the impedance is the same as the transmission line, $Z1 = 48.16\Omega$. This segment is connected with the ATE, so it can have reflections due to the mismatch between the ATE and loadboard.

2. **Second Segment:** This is the segment where the passive component is present. The impedance of this segment (Z2) changes due to the component, and a reflection occurs when the signal enters from the first segment to this one.

3. **Third Segment:** This is the final segment, which connects to the DUT, and its impedance is also the same as the transmission line, $Z3 = 48.16\Omega$. In this segment, the reflection occurs when the signal enters from the second segment, as there is an impedance mismatch due to the passive component in the second segment.

If multiple passive components are present in a loadboard channel, all of them can be combined in the second segment to calculate the total reflection due to them. This approach allows us to consider the reflections due to all components while keeping the computational resources minimal. Two example scenarios will be taken to understand the calculations of the transmitted voltage after reflection due to passive components.

1. **Example 1 (A series resistor is present of 10Ω):** Due to the presence of a resistor, the impedance of the second segment changes to 58.16Ω as calculated in Equation 5.12.

$$Z_2 = Z_0 + R = 48.16\ \Omega + 10\ \Omega = 58.16\ \Omega \tag{5.12}$$

The reflection will happen at the first interface when the signal goes from the first to the second segment, whose reflection coefficient comes out to be 0.094 from Equation 5.13. Using this reflection coefficient, the transmitted voltage to the second segment when the incident voltage is 4.906 V is 5.368 V as calculated in Equation 5.14.

$$\Gamma_{12} = \frac{Z_2 - Z_1}{Z_2 + Z_1} = \frac{58.16\ \Omega - 48.16\ \Omega}{58.16\ \Omega + 48.16\ \Omega} = \frac{10\ \Omega}{106.32\ \Omega} \approx 0.094 \tag{5.13}$$

$$V_{12} = V_{\text{LB}} \times (1 + \Gamma_{12}) = 4.906\ \text{V} \times (1 + 0.094) = 4.906\ \text{V} \times 1.094 \approx 5.368\ \text{V} \tag{5.14}$$

After the reflection at the first interface, the next reflection will occur at the second interface between the second and third segments. In this case, the reflection coefficient is -0.094, as from Equation 5.13, the source and load impedance are just interchanged, thereby only changing the sign of the coefficient. The final transmitted voltage to the third segment is 4.866 V as calculated in Equation 5.15. There is a minor reflection compared to the initial voltage before the component, which was 4.906 V. This is because constructive interference increases the voltage at the first interface. Still, at the second interface, destructive interference decreases the voltage, cancelling most of the reflection effect. As the value of the resistor increases, the effect due to reflection on the incident voltage will also increase.

$$V_{23} = V_{12} \times (1 + \Gamma_{23}) = 5.368\ \text{V} \times (1 - 0.094) = 5.368\ \text{V} \times 0.906 \approx 4.866\ \text{V} \tag{5.15}$$

2. **Example 2 (Solid State Relay and Series Resistance (10Ω)):** A series resistance (10Ω) and a relay with on-state resistance as 0.8Ω and output capacitance $C_{\text{out}}(V)$ calculated using Equation 5.4 are present in the loadboard channel. Due to the presence of a relay with an on-state resistance output capacitance and series resistance, there will be an impedance mismatch, resulting in reflection. The *on-state resistance* (R_{on}) is 0.8Ω and output capacitance is 19.28 pF which is calculated using Equation 5.4 with voltage being 4.906 V (from Case-1). First, the impedance of the relay needs to be calculated, and since the on-state resistance and output capacitance are in parallel, Equation 5.16 will be used for the impedance calculation.

$$\frac{1}{Z_{\text{SSR}}} = \frac{1}{R_{\text{on}}} + \frac{1}{X_{C_{\text{out}}}} \tag{5.16}$$

To calculate the impedance, the reactance due to the output capacitance, which is $X_{C_{\text{out}}}$, needs to be determined. This reactance is 824Ω after the calculation shown in Equation 5.17.

$$X_{C_{\text{out}}} = \frac{1}{2\pi f C_{\text{out}}} = \frac{1}{2\pi \times 10 \times 10^6\ \text{Hz} \times 19.28 \times 10^{-12}\text{F}} \approx 824\Omega \tag{5.17}$$

Now putting the values in Equation 5.16, the *relay impedance* (Z_{SSR}) is found to be approximately 0.8Ω. Now, this relay impedance will be added with the series resistance and impedance of segment two to get the total impedance of the second segment, which comes out to be 58.96Ω.

The reflection at the first interface, where the signal goes from the first to the second segment, will be calculated. The reflection coefficient comes out to be 0.1 as derived in Equation 5.18. Using this reflection coefficient, the transmitted voltage to the second segment when the incident voltage is 4.906 V is 5.3966 V as calculated in Equation 5.19.

$$\Gamma_{12} = \frac{Z_2 - Z_1}{Z_2 + Z_1} = \frac{58.96\ \Omega - 48.16\ \Omega}{58.96\ \Omega + 48.16\ \Omega} = \frac{10.8\ \Omega}{107.12\ \Omega} \approx 0.1 \tag{5.18}$$

$$V_{12} = V_{\text{LB}} \times (1 + \Gamma_{12}) = 4.906\ \text{V} \times (1 + 0.1) = 4.906\ \text{V} \times 1.1 \approx 5.3966\ \text{V} \tag{5.19}$$

After this, the reflection will occur at the second point when the signal goes from the second to the third segment. The reflection coefficient comes out to be -0.1 as the source and load impedance are interchanged. Using this, the final voltage that goes into the third segment is calculated, which comes out to be 4.85 V as derived in Equation 5.20. In this case, both the passive components were combined in the second segment, and the reflections were calculated, resulting in a slight voltage decrease from 4.906 to 4.85 V.

$$V_{23} = V_{12} \times (1 + \Gamma_{23}) = 5.3966\ \text{V} \times (1 - 0.1) = 5.3966\ \text{V} \times 0.9 \approx 4.85\ \text{V} \tag{5.20}$$

Case 3: Loadboard to DUT

Finally, the last reflection can occur if there is an impedance mismatch between the loadboard and the DUT. This can only be calculated if the user provides the impedance value of the DUT for which the calculation will be made. For an example scenario, the impedance of the DUT is assumed to be 52Ω. The incident voltage will be the voltage at segment three in the previous case, which is 4.85 V (the results of the second example from case 2 are taken for this case). The source impedance is the impedance of the third segment, which is 48.16Ω; this leads to an impedance mismatch between the loadboard and DUT. So, the reflection coefficient will be calculated using Equation 5.8, which comes out to be 0.0383. The final voltage that reaches the DUT is calculated as shown in Equation 5.21. The final voltage that reaches the DUT is 5.03 V, slightly higher than the incident voltage from the ATE, which was 5 V. The signal goes through three reflections from ATE to DUT before reaching its final destination.

$$V_{\text{DUT}} = V_{23} \times (1 + \Gamma_{\text{DUT}}) = 4.85\ \text{V} \times (1 + 0.0383) = 4.85\ \text{V} \times 1.0383 \approx 5.03\ \text{V} \tag{5.21}$$

5.4.3. Integration Automation

The previous subsection discussed the calculations required to model reflection due to impedance mismatches. After understanding the points in the loadboard where reflection can occur and how to calculate it, the next step is to integrate these parasitics and reflections in the ideal netlist generated in Stage 1. A Python script and a Verilog-AMS module were used to automate the integration of these parameters in the ideal netlist.

The Python script automatically includes parasitics effects and reflection in the ideal netlist by calculating the reflection and adding transmission line segments. The script divides each channel into three segments, with passive components in the second segment. If a relay is present, the script calculates the output capacitance based on the voltage to derive its impedance. The script's algorithm is shown in Algorithm 3. The script performs the following main tasks:

1. **Read Input Data:** Load R, L, G, and C parameters, DUT impedance (if provided), temperature and ideal netlist from the first stage.

2. **Process Each Channel:** primarily, the script changes the resistance based on the temperature using equation Equation 5.22.

$$R(T) = R_0 \times (1 + 0.00385 \times (T - 25)) \tag{5.22}$$

After this, the channel is traced, and the script identifies all the passive components present in the channel. Then, the R, L, G, and C are used to calculate the impedance of the trace using Equation 5.7. The channel is then divided into three segments: the first segment is connected with the ATE, the second segment contains all the passive components, and finally, the third segment is connected to the output node that goes to the DUT. If a passive component is present, the script calculates the total impedance of the second segment by combining it with the impedance of the passive element. The script then calculates the reflection coefficient at each point for all three segments.

3. **Write Final Netlist:** Finally, after all the calculations, the netlist is written with three verilog module calls for each channel. These module calls contain the R, L, G, C, and Γ. After the Verilog module calls, the connections are changed based on the new nodes and their declarations are added.

As mentioned in Algorithm 3, the Python script adds verilog module calls to the ideal netlist, defined in Algorithm 4. The Verilog module takes in the R, L, G, C, and Γ as inputs from the Python script. Based on these inputs, the module applies the reflection coefficient to the input voltage to model reflections. It adds a resistor, inductor, capacitor, and conductance between the input and output nodes to make a lumped transmission line circuit. The conductance is modelled as a shunt resistance whose values is the inverse of G. The Python script and Verilog-AMS module work together to automate the integration of parasitic effects and reflections into the ideal netlist. After this integration, a final netlist is generated that accurately represents the physical characteristics of the loadboard.

5.5. Framework Automation

The methodology to generate a netlist with the parasitics and impedance analysis was divided into three stages: Ideal Netlist Generation (Section 5.1), Parasitics Extraction (Section 5.2), and Parasitics and Reflection Integration (Section 5.4). After the third stage, a final netlist with parasitics for each channel was generated, and the effect of signal reflections was also included. It can be said that the final netlist emulates most of the critical parameters of the physical loadboard. The algorithm of each stage is connected, making it difficult for the engineers to operate the framework, as they will have to run the scripts separately. So, to tackle this issue and make the process more streamlined, an automation approach was considered and employed.

5.5.1. Proposed Methodology

To make the whole process easier for users, a master script was created that coordinates with the three stages, which passes on the input from the user to the stages and the outputs of the stages to the following stages. Firstly, the master script takes all the user inputs from one Excel file, as shown in Table 5.4, making it easier for users to give the inputs. The master script will then pass the inputs to the stages based on their requirements. The Excel file contains essential parameters such as frequency, pin and connector mappings, schematic and layout locations, and temperature settings, among other things. The inputs for the three stages are stated below:

- **Ideal Netlist Generation:** Inputs like Allegro Netlists, Module Names, and Pin Mappings are used in this stage to produce an ideal single-site netlist.
- **Parasitics Extraction:** Based on the ideal netlist from stage one and the frequency specified by the user, this step extracts the parasitic values (R, L, G, C) for every trace.
- **Parasitics and Reflection Integration:** The user provides inputs such as temperature and DUT impedance combined with the parasitic values calculated from the previous stage and the ideal netlist from the first stage. These inputs are used to determine the final accurate netlist.

The master script determines how many stages will be executed depending on the “Stage to Run” setting. This allows the user to finish after creating the ideal netlist or continue to the netlist that includes parasitics.

5.5.2. Execution Flow

The execution flow of the master script has two main modes, as mentioned below.

- **Mode 1:** This mode runs only the script to generate the ideal netlist. It is useful when engineers only need the netlist without the parasitics.

Algorithm 3 Netlist Modification Script

```
Input: RLGC parameters file (RLGC.csv), netlist file (netlist.vams), DUT impedance file (DUT.csv), temperature (Temp)
Output: Modified netlist file (modified_netlist.vams)

function ModifyNetlist
    Load RLGC parameters into data frame df_rlgc
    Load DUT impedance values into dictionary dut_impedance_dict
    Parse components and connections into list: components, ideal netlist lines into list: netlist
    for each channel channel in df_rlgc do
        Extract total $R$, $L$, $G$, $C$ for the channel
        Adjust Resistance for Temperature:
          Calculate $R_{\text{temp}} = R \times (1 + \alpha \times (Temp - 25))$
          Where $\alpha = 0.00385$ per °C (temperature coefficient of resistance for copper)
        Trace the channel to identify connected passive components passive_components
        Calculate the characteristic impedance $Z_0$ using Equation with adjusted $R_{\text{temp}}$:

            $$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

        Divide the channel into three segments:
          Segment 1: from ATE to passive components ($Z_1 = Z_0$)
          Segment 2: includes passive components ($Z_2$)
          Segment 3: from passive components to DUT ($Z_3 = Z_0$)
        Calculate impedance of Segment 2 ($Z_2$):
        if passive components are present then
            Combine impedances of passive components in Segment 2
            Calculate $Z_2$ using series and parallel combinations
            if a relay is present then
                Calculate $C_{\text{out}}$ using Equation (??)
                Include $R_{\text{on}}$ and $C_{\text{out}}$ in $Z_2$ calculation
            end if
        else
            $Z_2 = Z_0$
        end if
        Calculate reflection coefficients:
          ATE and Segment 1: $\Gamma_{ATE1} = \frac{Z_1 - 50}{Z_1 + 50}$
          At interface between Segment 1 and Segment 2: $\Gamma_{12} = \frac{Z_2 - Z_1}{Z_2 + Z_1}$
          At interface between Segment 2 and Segment 3: $\Gamma_{23} = \frac{Z_3 - Z_2}{Z_3 + Z_2}$
        if DUT impedance provided then
            Segment 3 and DUT: $\Gamma_{3DUT} = \frac{Z_{DUT} - Z_3}{Z_{DUT} + Z_3}$
        else
            Ignore DUT reflection
        end if
        Modify the netlist:
          Insert Verilog-A module calls for each segment
            First segment: between source and node 1
            Second segment: between node one and first passive component
            Third segment: between last passive component and output node
          Update component connections:
            Change the input node of the first passive component to the output node of the second segment
          Update analog block:
            Change the node connected to the DUT to the output node of the third segment
          Add declarations for newly created nodes
    end for
    Write the modified netlist to the output file
end function
```

Algorithm 4 Verilog-A Module for Transmission Line Segment

```
Module Declaration:
    module lumped_circuit (in_signal, out_signal, gnd);
Ports and Disciplines:
    inout in_signal, out_signal, gnd;
    electrical in_signal, out_signal, gnd;
Internal Nodes:
    electrical node1;
Parameter Definitions:
    parameter real R = 0;
    parameter real L = 0;
    parameter real C = 0;
    parameter real G = 0;
    parameter real Gamma = 0;
Analog Block:
    analog begin
    V(out_signal) <+ V(in_signal) * (1 + Gamma);
    end
Parasitic Elements:
    resistor #(.r(R_temp)) R1 (in_signal, node1);
    inductor #(.l(L)) L1 (node1, out_signal);
    capacitor #(.c(C)) C1 (out_signal, gnd);
    if (G > 0) begin
    resistance #(.r(1/G)) G1 (out_signal, gnd);
End Module
```

Table 5.4: Description of User Parameters

Parameter	Description
Frequency	At which project is running in Hz
Pin Mapping [1]	File that gives user defined outputs based on chip
Dual Board	If the loadboard is made of two boards then “yes” otherwise “no”
Connector Mapping [2]	If the above is yes, then connector mapping to connect the two boards
Schematic Location	Schematic of loadboard (in case of Allegro automation)
Layout Location	Location of Layout of loadboard for parasitics extraction
Allegro netlist location	Netlist generated by allegro location
Allegro Motherboard	Motherboard allegro netlist
Allegro Daughterboard	Daughterboard allegro netlist
Temperature	In degree Celsius
Stage to run	If just want schematic to netlist then “1”, otherwise “2”
Module name	Name of module

- **Mode 2:** This mode runs the entire process from the netlist generation to the extraction of parasitics and the impedance analysis.

The master script controls the whole framework and streamlines the process of generating the netlist. It also gives users options as to what kind of netlist they require. This automation makes the data exchange between stages easier and free of manual errors. It also allows users to quickly provide input in an Excel file instead of giving input to each stage separately.

Algorithm 5 Master Script for Loadboard Model Workflow

```
Load user inputs from Excel using pandas
if config[Dual Board] == yes then
    Set script path and arguments for dual-board configuration
else
    Set script path and arguments for single-board configuration
end if
if config[Stage to Run] == 1 then
    Execute netlist generation script
else
    Execute full simulation workflow, including parasitics extraction
    Continue with integrating parasitics into the netlist
end if
```

After running the full execution flow, this approach generates a netlist formatted according to the requirements of the simulation environment. The netlist also contains parasitic and a model of reflection, which integrates the effects caused by an impedance mismatch. After successfully generating a virtual model of the loadboard, the next crucial step is to validate the flow to test its accuracy.

6

Framework Validation

The framework implemented in the previous chapter is validated in this chapter. The chapters start with validating the generation of an ideal single-site netlist in Section 6.1. Next, Section Parasitics Validation validates the parasitics extraction algorithm with the help of a custom PCB. After this, Section 6.3 verifies the relay model created in Verilog. Stage 3, where the parasitics and reflection are integrated into the netlist, is validated in Section 6.4. Finally, the full framework is validated in Section 6.5 based on the requirements set in Chapter 4.

6.1. Ideal Netlist Validation

Validation is crucial in determining whether the model's results are usable. The first step in the validation process is to verify the ideal netlist generated in Section 5.1. The validation aims to verify the requirements **R6 (Integration with AMS-VT)** and **R5 (Automation Level)**.

6.1.1. Experimental Setup

To properly validate the ideal netlist, it is crucial to determine the conditions under which the validation was done. The key components of the experimental setup are as follows:

Simulation Environment

The simulation was conducted in the AMS-VT environment, a closed-loop system in which the ATE model is connected to the Cadence Spectre AMS simulator. In the ATE model, all the activities of the ATE test program are recorded and stored as stimuli. The stimuli are then used to drive the DUT signals via the loadboard. The simulation environment also records the DUT response. This allows for realistic simulation of testing scenarios, including forcing voltages, measuring responses, and activating components like relays.

Test Program

A test program called "Max rating" checks the voltage level by pushing the loadboard pins to their maximum voltages and relays for all the pins, checking the electrical behaviour of all outputs in the netlist. The basic structure of the test program is shown in Algorithm 6.

Algorithm 6 Max rating Test Code for Loadboard Validation

```
Initialize all pins to 0 V.
Force pins to 5 V.
Force pins to 6.5 V.
Force pins back to 5 V.
Record the outputs.
```

The pins were first forced to 5 V, which is the nominal voltage, and then they were forced to 6.5 V, which is the maximum voltage the pins can go to. The test program is executed in an AMS-VT environment with the ideal loadboard netlists. After the complete execution SimVision, a Cadence waveform viewer and analysis tool, was used for post-simulation analysis. It allows a detailed check of signal behaviours and voltage levels.

6.1.2. Validation Methodology

The validation process consists of two methods: manual verification and simulation verification.

Manual Verification

This involves a detailed comparison between the generated ideal netlist and the original loadboard schematic. The steps taken in the manual verification process include:

1. **Component Verification:** Each component in the netlist was cross-checked with the schematic to check its presence. This included verifying components such as resistors, capacitors, and relays.

2. **Connection Verification:** The electrical connections between the components were checked. This involved checking if the nodes in the netlist were the same as those in the schematic and if any connection was missing.

3. **Parameter Verification:** The values of the components, such as resistance and capacitance values, were checked to match those specified in the schematic.

4. **Hierarchical Structure Verification:** For loadboards with multiple boards (e.g., motherboard and daughterboard configurations), the hierarchy in the netlist was compared to check for the correct representation of the interconnections.

Simulation-Based Validation

This aimed to verify the functional correctness of the loadboard netlist under realistic testing conditions. The DUT model was removed to keep the focus on the loadboard netlist and remove any external variables. This ensures that any errors observed during simulation are due to the loadboard netlist.

1. **Simulation Setup:** The generated ideal netlist in Stage-1 was integrated into the simulation environment explained above.

2. **Execution of Simulations:** After executing the simulations, data such as voltage levels and signal transitions are analysed using SimVision.

3. **Golden Simulation Comparison:** The simulation results from the generated netlist are compared against the golden simulation. The golden simulation was executed in the same environment using a loadboard netlist, which was made and verified by test engineers.

4. **Analysis of Parameters:** The following parameters are analysed for the simulations:

   - **Voltage Levels:** Verification of the voltage levels on all pins matched between the simulations at corresponding times.
   - **Pulse Width:** Verification of the duration for which signals remained at specific voltage levels before transitioning. The pulse width is measured at 50% of the voltage increase, which means the measurement was taken from 5.75 to 5.75 V for rise and fall events.
   - **Events:** Verification of all the events in the simulation with the events in the test program.

6.1.3. Simulation Results

The above-mentioned experimental setup and validation methodology were used to properly validate stage 1 of the design that generates an ideal netlist. This validation included manual and simulation checks.

Table 6.1: Project Validation and Design Specifications

Projects	Design Type	Tool Working	Validation Method
1	Single Board	Yes	Manual and Simulation
2	Single Board	Yes	Manual
3	Single Board	Yes	Manual
4	Single Board	Yes	Manual
5	Double Board	Yes	Manual and Simulation

Manual Validation Results

After the manual validation, the following findings were made:

1. **Components Accuracy:** All components in the netlist matched the schematic in terms of type, nodes and configuration.
2. **Connection Integrity:** The netlist had correct connectivity with no missing or extra connections, as each net was correctly represented in the netlist.
3. **Hierarchical Consistency:** The interconnections were correctly modelled in dual-board configurations.
4. **Parameter Consistency:** All parametric values of the components in the netlist matched with the schematic.

These results confirm that the ideal netlist accurately represents the schematic without structural discrepancies.

Simulation Results

The simulation results demonstrated high accuracy between the generated netlist and the golden simulation. Figure 6.1 shows the comparison for a set of signals.

- **Markers:** Indicate key events and transitions with precise timing.
- **Red Markers:** Represent signals from the golden simulation.
- **Yellow Markers:** Represent signals from the simulation using the generated netlist.

After analysing the signals, it was found that all signals show identical voltage in both simulations. For example, signals are forced to a nominal voltage of 5 V and then reach a maximum voltage of 6.5 V, matching exactly in both cases. Signal transitions occur at the same time as written in the text program. The pulse width from when the signals go from 5 to 6.5 V and then back to 5 V was also identical for both the simulation results, which was 10.685 ms. The activation and deactivation of relays and other components occur as expected, with no discrepancies observed. A critical aspect of this validation was checking if the generated ideal netlist could be integrated with the AMS-VT environment without errors. No errors were encountered during the integration and simulation with the environment. The validation in this section confirms the ideal single-site netlist generated by the algorithm adheres to the requirements **R6 (Integration with AMS-VT)** and **R5 (Automation Level)**.

Stage 1 was verified for multiple board configurations using manual or simulation validation, as shown in Table 6.1. The framework generated correct netlists for multiple loadboards, even for a loadboard of dual configuration. This verifies that the framework adheres to requirement **R7 (Generalization)**.

6.2. Parasitics Validation

This section explains the setup, methodology and results of validating the parasitic extraction algorithm used in Stage 2 of the framework described in Section 5.2. The validation process involves creating a custom PCB with different trace configurations, extracting parasitic using the automation algorithm from PowerSI, simulating the trace in PSpice and PowerDC, and then comparing the results to ensure accuracy.

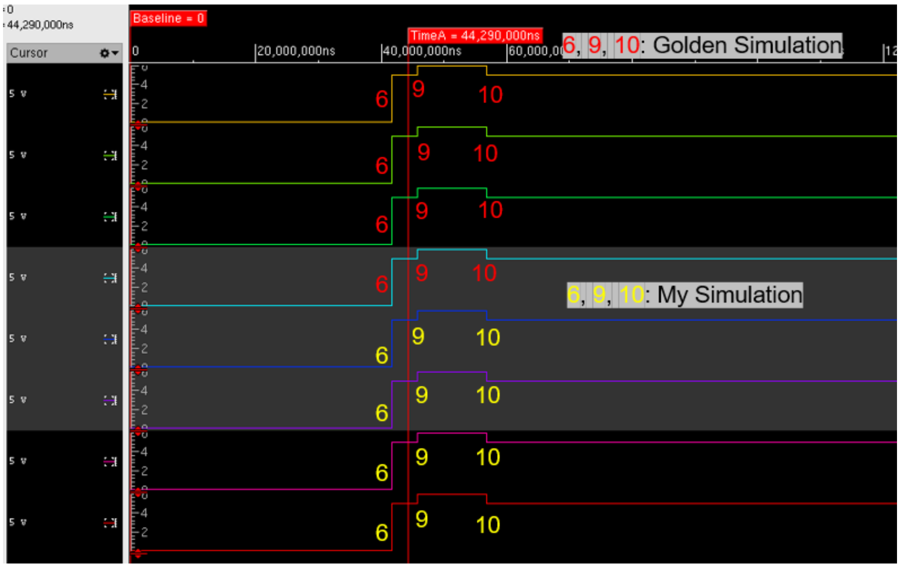


Figure 6.1: Simulation Validation

6.2.1. Experimental Setup

The setup contains a custom-designed PCB with different trace configurations to evaluate the parasitic extraction accuracy under multiple conditions. The key components of the setup are as follows:

Custom PCB Design

The PCB was made using Cadence Allegro PCB Designer, a tool that allows for an easy design with inbuilt functions to add traces, components, and layers to a PCB. The PCB was fabricated with the following specifications:

- **Material:** Copper traces on an FR-4 substrate chosen for its common use in the industry.
- **Trace Thickness:** The thickness of the trace was set to 35 um, representing a normal trace thickness.
- **Number of Layers:** This was a 4-layer PCB, in which layers two and three were ground planes to minimize electromagnetic interference.
- **Trace Configurations:** Five different configurations were implemented, shown in Figure 6.2, that are as follows:
  1. **Baseline Trace (1 in Figure 6.2):** This was used as a baseline for comparison, created with a straight line and regular measurements. This trace has a 170 mm length and 0.25 mm width.
  2. **Traces with varying lengths (2 in Figure 6.2):** Different lengths of traces were studied to see how they affect parasitic. Trace lengths varied from 50 mm to 350 mm to check its effect on parasitics.
  3. **Traces with widths (3 in Figure 6.2):** Different widths of traces were studied to see how they affect parasitic. Trace widths varied from 0.25 mm to 0.45 mm to check its effect on parasitics.
  4. **Proximity effects due to traces (4 in Figure 6.2):** Traces were arranged closely to test for mutual capacitance and the impact of electromagnetic interference between adjacent traces.
  5. **Proximity effect due to Ground Planes (5 in Figure 6.2):** For the above six configurations, the layer that contains the trace is sandwiched between two ground planes. In this case, the two ground planes are removed and replaced by two ground nodes to see if the model considers their proximity effect.

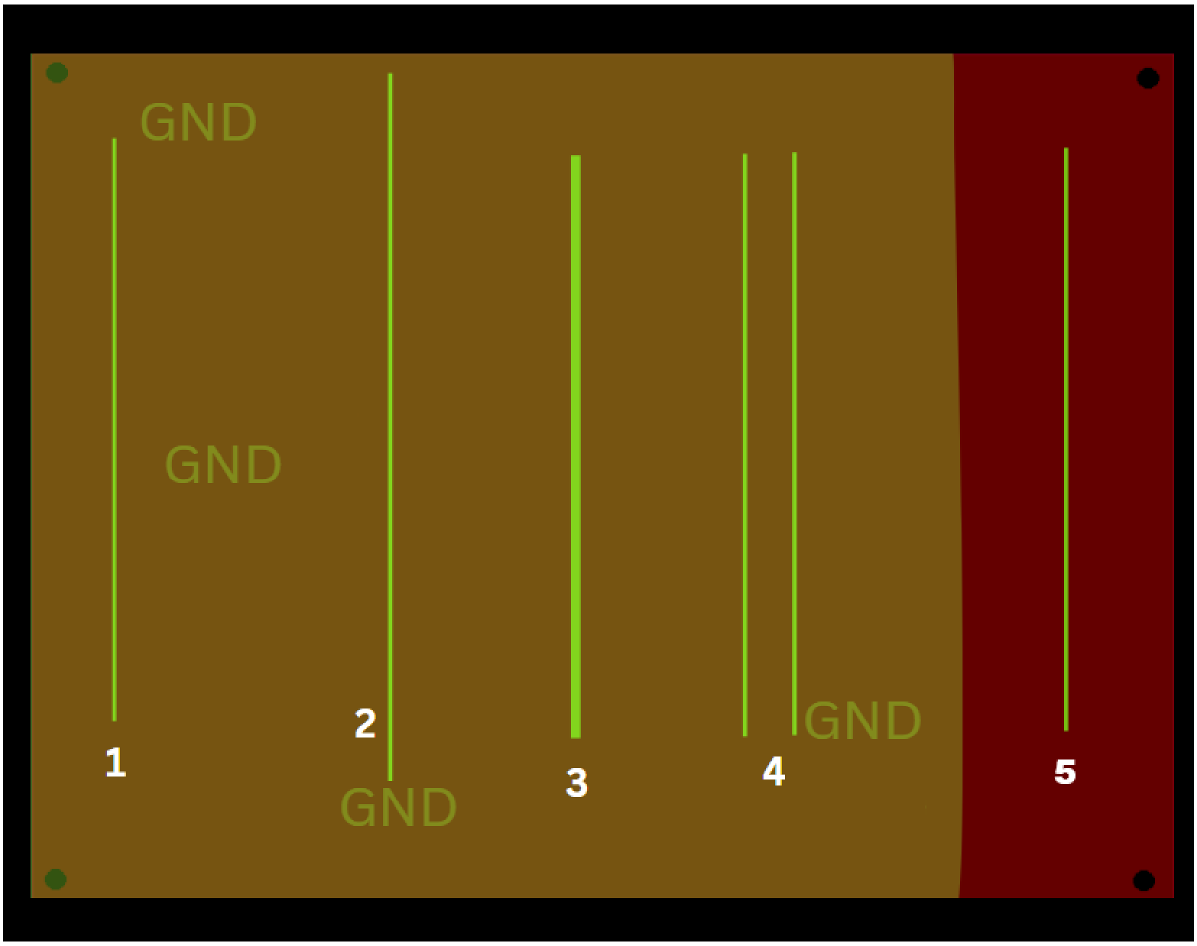


Figure 6.2: Test PCB

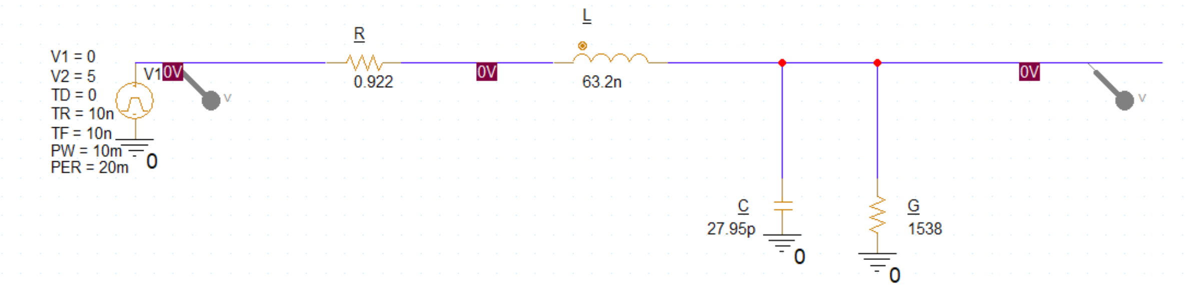


Figure 6.3: PSpice circuit of Loadboard Trace for Parasitics Validation

Parasitics Extraction

The parasitics for the trace were extracted from PowerSI using the automation algorithm that was developed. Three different frequencies, 10, 50, and 100 MHz, were used to extract the parasitics to analyse their frequency-dependent nature.

Simulation Tools

Two simulation tools were used to validate the parasitics.

- **PSpice:** This was used to make a lumped circuit of the traces using the extracted R, L, G, and C as shown in Figure 6.3. This allows for the simulation of the trace using parasitic to check their effect on the voltage.
- **PowerDC:** This was used on the PCB to conduct an IR drop analysis on the PCB traces for different frequencies.

Operating Conditions

The simulations were performed under certain conditions to ensure consistency.

- **Operating Frequencies:** Three frequencies, 10, 50, and 100 MHz, were chosen to check the effects of frequency on voltage.
- **Input Voltages:** Voltage of 5V, 20V, and 40V were provided as input to analyse the effect of parasitics on different voltages.
- **Environmental Conditions:** Simulation was performed considering room temperature (25 °C) conditions.

6.2.2. Validation Methodology

The validation methodology followed a proper approach to simulate the parasitics accurately. The first step was the PCB design, in which each trace was carefully configured in Allegro Design. The layout location was provided to the algorithm for each configuration, and parasitics were extracted at the specified frequencies. Each trace was modelled as a lumped circuit in PSpice using these extracted parasitics. Then, PowerDC was used to analyse the voltage drop across the traces using different input voltages and frequencies. The baseline configuration was used as a reference point. Then, the simulation results from PSpice and PowerDC were compared for each trace configuration. The impact of frequency on parasitics was evaluated by comparing simulation results across three frequencies.

6.2.3. Results and Analysis

The validation process provided results across all trace configurations for three frequencies. The results and their analysis for all trace configurations are explained below in detail.

Table 6.2: Parasitic Parameters for Baseline Trace at Different Frequencies

Frequency (MHz)	R (Ω)	L (nH)	C (pF)	G (S)
10	0.922	63.2	27.95	0.00065
50	1.287	61.4	27.92	0.00066
100	1.812	59.15	27.88	0.00067

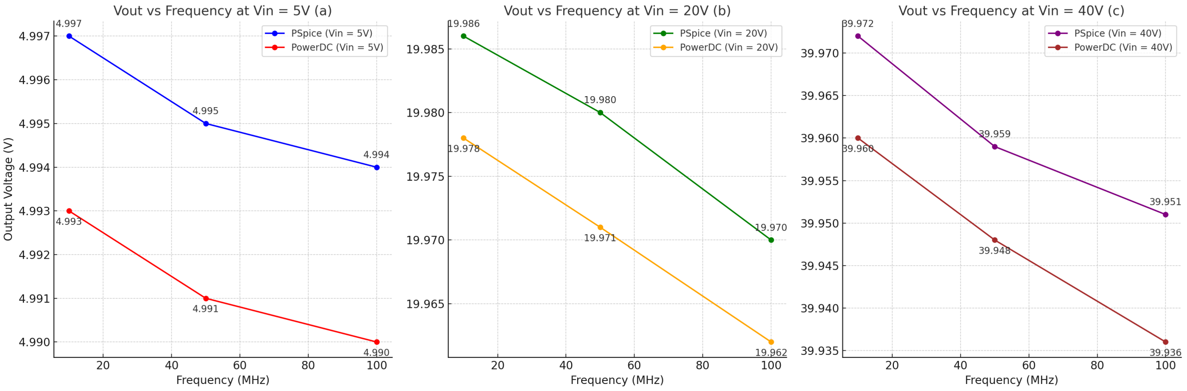


Figure 6.4: PSpice vs. PowerDC Output Voltages for Baseline Trace

Baseline Configuration (Configuration 1

A standard trace is set up as the benchmark to validate the simulation framework thoroughly. This benchmark is used to determine the types of physical characteristics that the model considers. The standard trace was created with no bends representing a straight line as shown in Figure 6.2 marked by ”1”. The baseline trace had no bends and directly went from the input to the output node, and the length was set as 170 mm since this is the most common length for the traces of loadboards. The width was set to 0.25 mm, a typical trace starting width. This trace was intentionally selected for its straightforwardness and consistency, making it perfect for baseline configuration. It does not involve intricate elements, such as curves or integrated parts, guaranteeing that extra factors do not affect the outcomes. Table 6.2 presents the extracted parasitic parameters for the baseline trace at different frequencies. As expected, R increases with frequency due to the skin effect, which causes the current to flow towards the edge of the trace; it saw around 40% increase when the frequency increased. Also, L slightly decreases with increasing frequency, which matches the theory that higher frequencies can lead to reduced inductive effects due to losses in the trace’s magnetic field. The C and G remain see a minimal change through the frequencies, with C seeing a minor decrease and G seeing a minor increase.

Figure 6.4 compares the output voltages obtained from PSpice and PowerDC simulations using the extracted parasitics. The simulation results of both of them are pretty similar. At 10 MHz for 5 V, Both simulations predict an output voltage very close to the input, with PSpice at 4.997 V and PowerDC at 4.993 V, resulting in a difference of approximately 0.08%. Similarly, for 40 V, the output voltages are 39.972 V (PSpice) and 39.960 V (PowerDC), a negligible difference of 0.03%. For 50 MHz, PSpice predicts 4.995 V, and PowerDC predicts 4.991 V, a difference of 0.08%, and output voltages are 19.980 V (PSpice) and 19.971 V (PowerDC), a minimal difference of 0.04% for 20 V. At a higher frequency of 100 MHz, Output voltages are 19.970 V (PSpice) and 19.960 V (PowerDC), a difference of 0.05%, and for 40 V, Output voltages are 39.951 V (PSpice) and 39.936 V (PowerDC), a difference of 0.06%. The difference observed for the baseline was minimal within the range from 0.03% to 0.08%, indicating that the extracted parasitics resulted in an accurate voltage drop for the baseline.

Increased Trace Length (Configuration 2)

After getting the baseline results, the physical properties of the trace can be changed to determine if the model can emulate the changes these effects will bring. The first change was done by increasing the trace length while keeping all other parameters just like the baseline, as shown in Figure 6.2 marked by ”2”. To analyse the impact of trace length on parasitic elements, traces of varying lengths, 50, 150,

Table 6.3: Parasitic Parameters for Increased Trace Lengths at Different Frequencies

Trace Length (mm)	Frequency (MHz)	R (Ω)	L (nH)	C (pF)
50	10	0.462	31.6	13.975
50	50	0.643	30.7	13.960
50	100	0.906	29.575	13.940
150	10	1.387	94.8	41.925
150	50	2.064	92.1	41.760
150	100	2.718	88.725	41.640
250	10	2.305	157.0	69.875
250	50	3.445	152.25	69.700
250	100	4.525	146.95	69.550
350	10	3.227	219.2	97.825
350	50	4.835	222.375	97.450
350	100	6.345	224.125	97.350

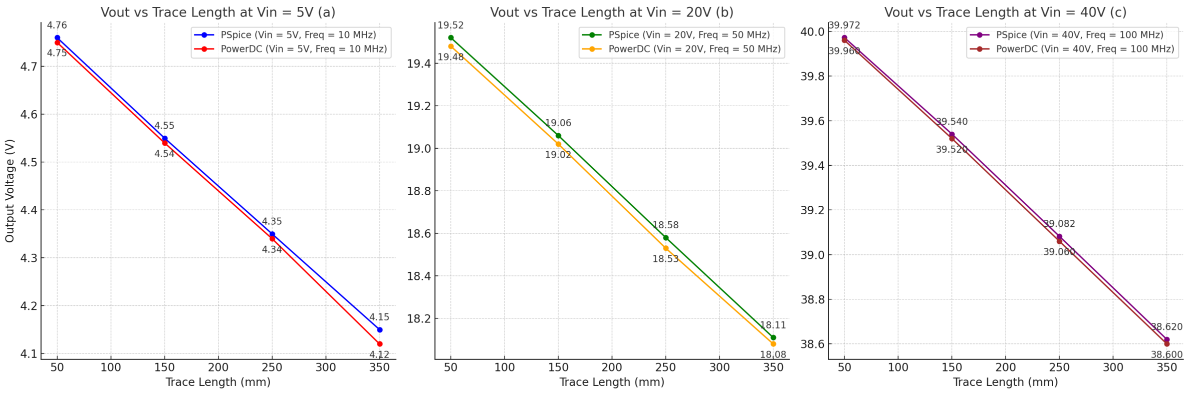


Figure 6.5: PSpice vs. PowerDC Output Voltages for Increased Trace Lengths

250, and 350 mm, were simulated while keeping the thickness and width of the trace the same as the baseline. Table 6.3 presents the extracted parasitic parameters for the baseline and increased trace lengths at different frequencies. Resistance increases linearly with the trace length; for example, at 10 MHz, R goes from 0.462 Ω (50 mm) to 3.227 Ω (350 mm). Inductance also increases with the trace length but is not linear; for example, at 10 MHz, L increases from 31.6 nH (50 mm) to 219.2 nH (350 mm). Capacitance increases with length as the trace interacts more with the ground plane. This confirms that parasitics extraction accurately captures the effects of trace length.

Figure 6.5 shows the output voltages from PSpice and PowerDC for traces of different lengths at 10 MHz, 50 MHz, and 100 MHz frequencies. At 10 MHz, the output decreases from 4.76 V (50 mm) to 4.15 V (350 mm) for an input of 5 V, which indicates that the voltage drop percentage ranges from approximately 4.8% to 3% as trace length increases. For a 20 V input, the output voltage goes from 19.52 V (50 mm) to 18.11 V (350 mm), showing a drop percentage from approximately 2% to 1.9%. Similarly, for 40 V, the drop percentage ranges from approximately 1.9% to 1.8%. The difference between PSpice and PowerDC is minimal, with the maximum difference being 0.02 V.

Increased Trace Width (Configuration 3)

The trace width also plays a critical role in determining the signal behaviour in a PCB. As the trace width increases, the current gets more area to flow, leading to decreased resistance. This configuration was taken with varying widths of 0.25, 0.35, and 0.45 mm for simulation to analyse the effect of changing trace width. The length of each trace was fixed at 170 mm, the baseline length. Table 6.4 shows the extracted parasitics for traces with different widths at multiple frequencies. R decreases with increasing width, which matches with their, according to which wider traces have lower resistance due to the increased cross-sectional area. Inductance and capacitance also decrease with increasing width. For example, C goes from 27.95 pF (0.25 mm) to 27.80 pF (0.45 mm) at 10 MHz.

Table 6.4: Parasitic Parameters for Increased Trace Widths at Different Frequencies

Trace Width (mm)	Frequency (MHz)	R (Ω)	L (nH)	C (pF)
0.25	10	0.922	63.2	27.95
0.35	10	0.660	41.2	27.85
0.45	10	0.515	31.4	27.80
0.25	50	1.287	61.4	27.92
0.35	50	0.925	38.5	27.80
0.45	50	0.720	29.0	27.75
0.25	100	1.812	59.15	27.88
0.35	100	1.302	37.3	27.70
0.45	100	1.010	28.8	27.65

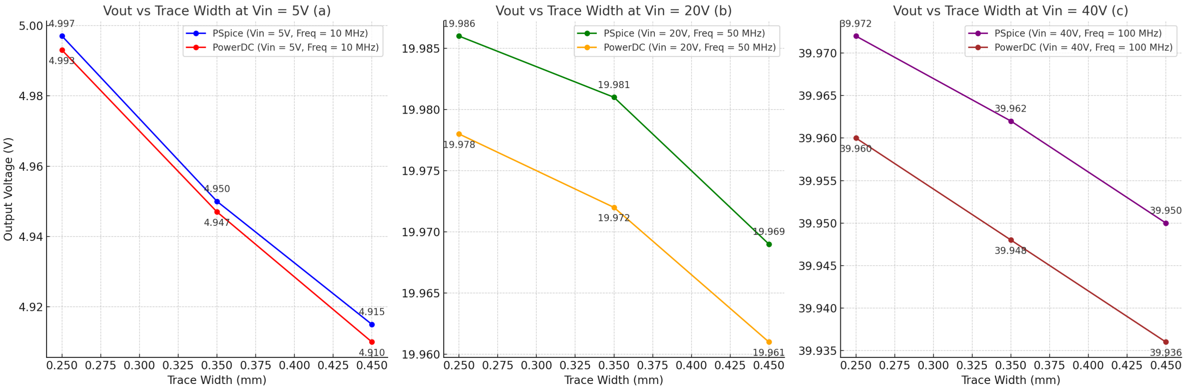


Figure 6.6: PSpice vs. PowerDC Output Voltages for Increased Trace Widths

Figure 6.6 shows the output voltages from PSpice and PowerDC for traces with varying widths at 10 MHz, 50 MHz, and 100 MHz frequencies. At a 5 V input for 10 MHz, the voltage goes from 4.997 to 4.915 as the width increases, showing a minimal voltage drop. Similarly, for 100 MHz, the output voltage goes from 39.72 V to 39.48 V, showing a voltage drop from approximately 0.06% to 0.07%, demonstrating the negligible impact of trace width on voltage levels at high frequencies. The difference between PSpice and PowerDC is minimal within an error of 1%, proving that the parasitics extraction algorithm works well for varying widths for multiple frequencies.

Proximity Effects Due to Adjacent Traces (Configuration 5)

One of the main contributors to the parasitic, especially resistance and capacitance, is the proximity effect. This refers to the parasitic due to traces being placed very close to each other. For this case, one trace was added at a distance of 1 mm to the baseline trace with the same physical characteristics. Table 6.5 shows the extracted parasitics for traces with adjacent proximity. Resistance increases by a huge number due to the combined effects of trace length and mutual coupling between adjacent traces. Inductance decreases with frequency from 75.0 nH at 10 MHz to 71.0 nH at 100 MHz, and capacitance increases marginally with frequency, from 35.50 pF at 10 MHz to 35.40 pF at 100 MHz.

Figure 6.7 shows the output voltages from PSpice and PowerDC for traces with adjacent proximity at 10 MHz, 50 MHz, and 100 MHz frequencies. For an input of 5 V at 10 MHz, the output decreases from 4.997 V (baseline) to 4.800 V (adjacent) in PSpice and from 4.993 V to 4.790 V in PowerDC, representing a voltage drop of approximately 4.8% and 4.9%, respectively. At 20 V, the output goes

Table 6.5: Parasitic Parameters for Adjacent Traces at Different Frequencies

Frequency (MHz)	R (Ω)	L (nH)	C (pF)	G (S)
10	1.200	75.0	35.50	0.00070
50	1.700	73.5	35.45	0.00072
100	2.400	71.0	35.40	0.00074

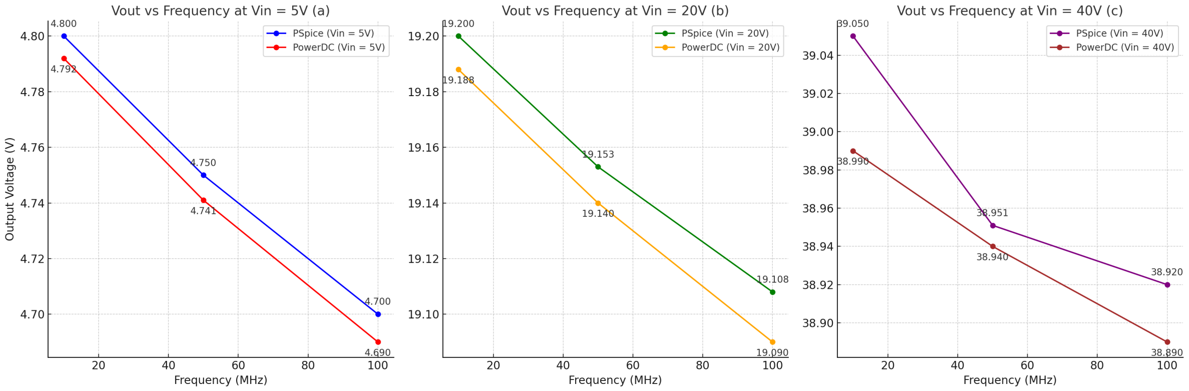


Figure 6.7: PSpice vs PowerDC Output Voltages for Adjacent Traces

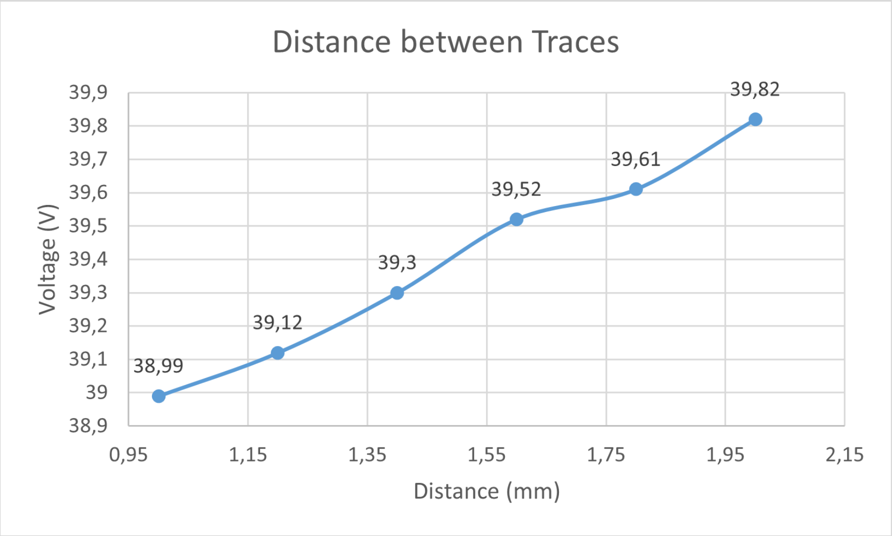


Figure 6.8: Length between traces vs Voltage

from 19.986 V (baseline) to 19.200 V (adjacent) in PSpice and from 19.978 V to 19.190 V in PowerDC, representing voltage drops of approximately 4.0% for both. Similarly, at 40 V, there is a drop of roughly 2.6% in both PowerSI and PowerDC. This represents a minimal difference between PSpice and PowerDC, showing that the parasitics consider the effects due to adjacent traces.

The graph in Figure 6.8 shows how the voltage changes based on the distance between two traces when a voltage of 40V is forced at 10 MHz. It starts at 39.99V when the trace is only 1 mm apart. As the distance increases, the voltage gets closer to the baseline, which is 39.96 V. As the distance increases and reaches 2 mm, the voltage reaches 39.82 V, showing a decrease in mutual trace effects. This indicates that stage 2 correctly models the impact of distance between the adjacent traces.

Proximity Effects Due to Ground Planes (Configuration 6)

For baseline configuration, the trace was sandwiched between two ground planes and this plane also caused an increase in parasitic capacitance. For the last case, the ground planes were replaced with two ground nodes near the trace so that there is no proximity effect due to them. Table 6.6 shows the extracted parasitic parameters for traces with and without ground planes at different frequencies. Based on extracted parasitics, it can be seen that there is no difference between the parasitics with and without ground planes. This means that the output voltage from PSpice will be the same as the baseline, whereas PowerDC sees a difference in output voltage as shown in Table 6.7. The PowerDC tool incorporates the effect of removing the ground planes as the output voltage sees a decrease across all input voltages. This means that the parasitics extraction algorithm does not consider the effects of the ground plane while extracting parasitics.

Table 6.6: Parasitic Parameters with and without Ground Planes at Different Frequencies

Trace Configuration	Frequency (MHz)	R (Ω)	L (nH)	C (pF)	G (S)
With Ground Planes	10	0.922	63.2	27.95	0.00065
Without Ground Planes	10	0.922	63.2	27.95	0.00065
With Ground Planes	50	1.287	61.4	27.92	0.00066
Without Ground Planes	50	1.287	61.4	27.92	0.00066
With Ground Planes	100	1.812	59.15	27.88	0.00067
Without Ground Planes	100	1.812	59.15	27.88	0.00067

Table 6.7: Output Voltages after removing Ground Planes at 10 MHz for PowerDC

Input Voltage (V)	Baseline - PowerDC (V)	Out Voltage - PowerDC (V)
5	4.993	4.9
20	19.978	19.86
40	39.960	39.74

Summary of Parasitics Validation

The parasitics extraction algorithm from stage 2 went through validation for multiple trace configurations, which accounted for most of it except the effects due to the ground plane. The algorithm properly considers the trace's physical parameters, such as length, width, and proximity effects due to adjacent traces. The result of the PSpice simulation that uses the extracted parasitics closely matched the results of IR drop analysis from PowerDC for multiple frequencies, proving the ability of the algorithm to extract parasitics based on frequency. This verifies that the framework adheres to requirement **R3 (Operating Conditions)** for frequency.

6.3. SSR Model Validation

The validation of the SSR model developed in Section 5.3 can be done by comparing the model's results with the manufacturer's datasheet. The verified characteristics are:

1. On-State resistance vs Temperature
2. Off-State Leakage Current vs Applied Voltage
3. Output capacitance vs Applied Voltage
4. Switching Behaviour

By verifying their parameters, it can be checked whether the SSR model is reliable and matches the datasheet.

6.3.1. Experimental Setup

A testbench was created in Verilog-AMS to simulate the SSR model under different conditions. The testbench adjusts temperature and applied voltage and records the SSR module's result. It also validates the switching behaviour by applying control signals and checking the output.

Testbench Algorithm

The algorithm for the testbench is shown in Algorihtm 7. The algorithm starts by setting a temperature range from -40 to 80 °C and a range of voltage from 0 to 60 V. First, the on-state resistance is measured by applying a control voltage of 5 V to run SSR on and an applied voltage of 1 V. The current is then measured, and resistance is calculated using the ohms law. Then, the SSR is switched off, and the current is measured for different applied voltages. Then the relay is switched on, and the output capacitance is calculated by measuring reactive current.

The simulation conditions are defined below:

- *Threshold Voltage* V_{th} set to 2.5 V.
- Temperature range set from -40 to 80 °C.

Algorithm 7 SSR Model Testbench Algorithm

```
Initialize: Define temperature range $T = \{-40°\text{C}, -20°\text{C}, 0°\text{C}, 20°\text{C}, 40°\text{C}, 60°\text{C}, 80°\text{C}\}$
Define applied voltage range $V = \{0\text{ V}, 10\text{ V}, 20\text{ V}, 30\text{ V}, 40\text{ V}, 50\text{ V}, 60\text{ V}\}$
Set control voltage threshold $V_{th} = 2.5\ V$
Verify Switching Behavior:
Apply control voltage $V_{cmd} = 5\ V$ (above threshold)
Apply small voltage $V_{out} = 1\ V$ across output terminals
Measure current $I_{on}$
Ensure $I_{on} > 0$ (relay is conducting)
Apply control voltage $V_{cmd} = 0\ V$ (below threshold)
Measure current $I_{off}$
Ensure $I_{off} \approx 0$ (relay is not conducting)
for each temperature $T_i$ in $T$ do
    Set SSR model temperature to $T_i$
    On-State Resistance Measurement:
    Apply control voltage $V_{cmd} = 5\ V$ (relay on)
    Apply small voltage $V_{out} = 1\ V$ across output terminals
    Measure current $I_{on}$
    Calculate $R_{on} = \frac{V_{out}}{I_{on}}$
end for
for each applied voltage $V_j$ in $V$ do
    Off-State Leakage Current Measurement:
    Set SSR model temperature to $25°\text{C}$
    Apply control voltage $V_{cmd} = 0\ V$ (relay off)
    Apply voltage $V_{applied} = V_j$ across output terminals
    Measure leakage current $I_{leak}$
    Output Capacitance Measurement:
    Apply a small AC voltage $v_{ac}$ superimposed on $V_{applied}$
    Measure reactive current $I_{cap}$
    Calculate $C_{out} = \frac{I_{cap}}{2\pi f v_{ac}}$
end for
Output: Compile measured $R_{on}$, $I_{leak}$, and $C_{out}$ into tables
```

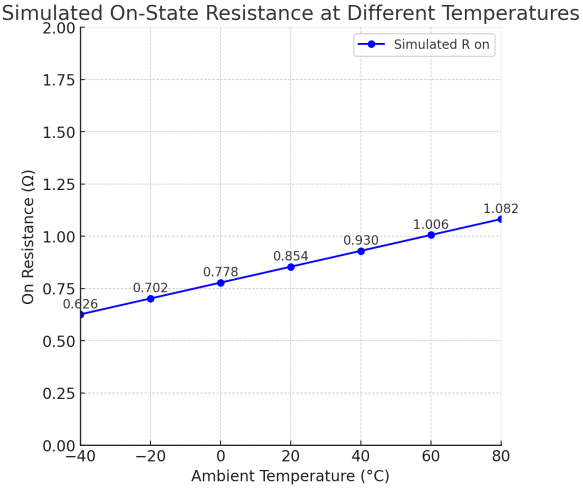


Figure 6.9: Simulation: On-State vs Temperature

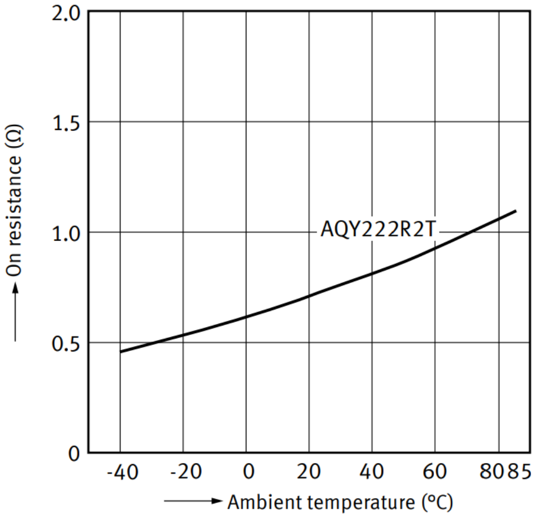


Figure 6.10: Datasheet: On-State Resistance vs Temperature [57]

- Applied voltage set from 0 to 60 V
- ModelSim Verilog-AMS simulator was used for the simulation.
- Frequency to find output capacitance was 1 MHz.

6.3.2. Result and Analysis

First, the testbench verifies the model's switching behaviour. The applied voltage was also measured at the output when a control voltage of 5 V was applied, and there was no conduction when the control voltage was 0 V. The validation results of the SSR are presented below:

On-State resistance

The on-state resistance of the SSR model at different temperatures is shown in Figure 6.9. The on-state resistance increases linearly with temperature, which is consistent with the graph in the datasheet showing in Figure 6.10. At 25 °C, both measure approximately 0.8 Ω, resulting in a negligible difference. Similarly, at 45 °C, the resistance increases to 0.91 Ω both. This indicates that the model captures the temperature-dependent behaviour of the SSR model.

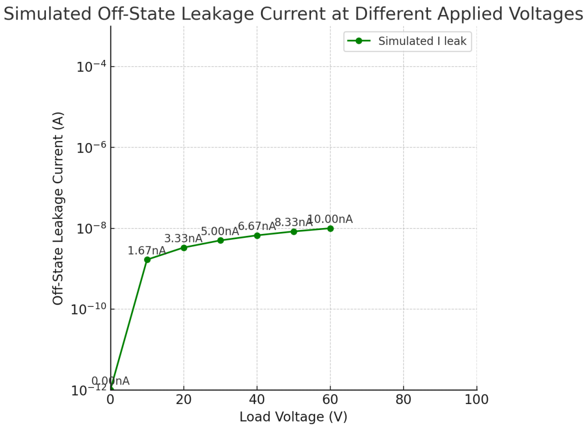


Figure 6.11: Simulation: Off-State Current vs Voltage

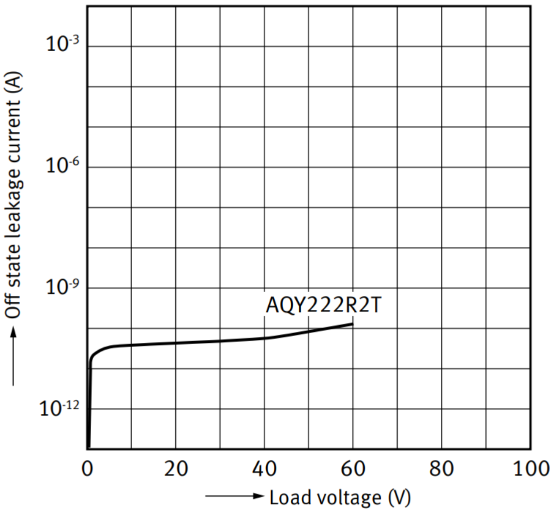


Figure 6.12: Datasheet: Off-State Current vs Voltage [57]

Off-State Current Leakage

The off-state current leakage measured is shown in Figure 6.11 for different voltages. The simulated results increase with the applied voltage, which matches the datasheet graph shown in Figure 6.12. At 20 V, PSpice measures the leakage current as 3.33 nA which matches with the datasheet. Similarly, at 60 V, both leakage is 10 nA, with the off-state resistance being 6 GΩ.

Output Capacitance

The output capacitance was measured for different voltages, and the result is shown in Figure 6.13. The simulated output capacitance decreases exponentially with increasing voltage, which matches the datasheet graph shown in Figure 6.14. At 0 V, both give a capacitance of 27 pF. Similarly, at 60 V, they measure 5.6 pF, showing negligible differences.

validation Summary

This validation depicts that the SSR model accurately calculates the key characteristics compared to the datasheet. The characteristics were accurate for different temperatures and applied voltages. This validation confirms that the model adheres to the requirement **R4 (Component Modeling)**.

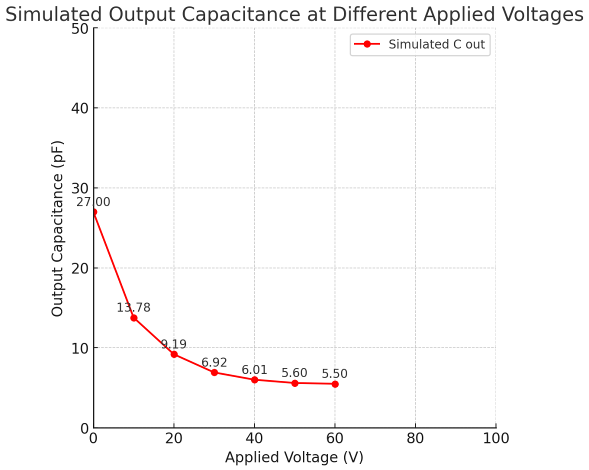


Figure 6.13: Simulation: Output Capacitance vs Voltage

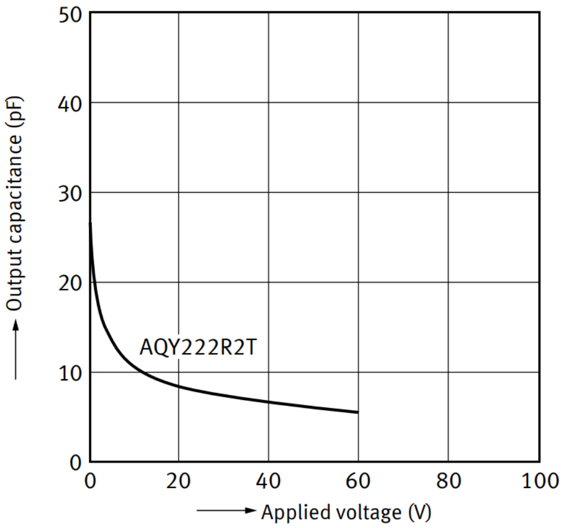


Figure 6.14: Datasheet: Output Capacitance vs Voltage [57]

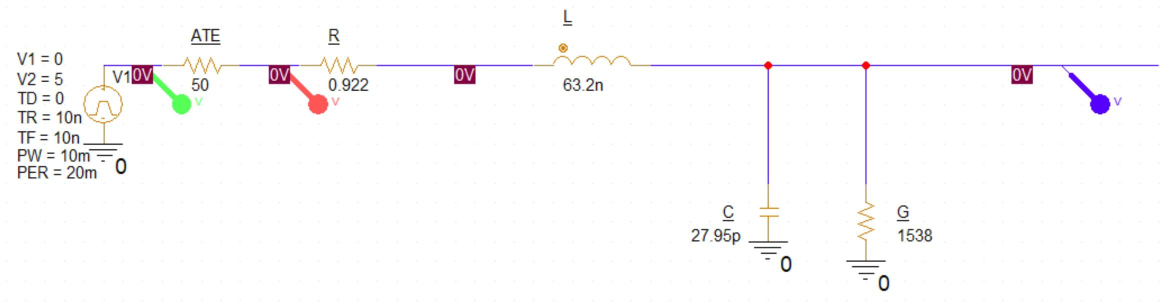


Figure 6.15: Configuration 1: ATE present

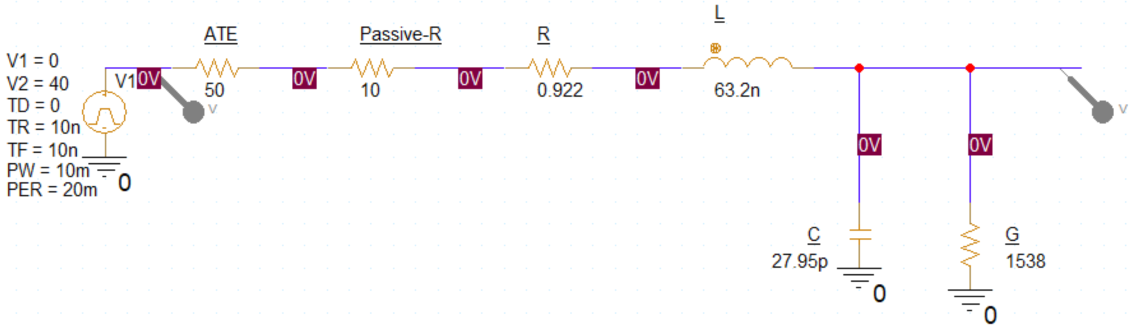


Figure 6.16: Configuration 3: ATE with Resistor (10 Ω)

6.4. Parasitics and Reflection Integration Validation

This section focuses on validating Stage 3 implemented in Section 5.4. Stage 3 involves adding the parasitics and reflection effects into the ideal netlist to mirror real-world conditions. This validation will ensure that this stage replicates the expected voltage levels for different configurations.

6.4.1. Experimental Setup

The validation was done by taking the custom-made PCB from Section 6.2. The extracted parasitics for different frequencies of the baseline configuration were taken for this validation. Three different configurations were considered for the validation:

1. **Configuration 1: ATE Present**—This configuration simulates the scenario where only the ATE is connected to the loadboard, as shown in Figure 6.15. The ATE represents a 50 Ω resistance after the voltage source.

2. **Configuration 2: ATE and DUT Present** – Emulates the ATE and DUT presence with the load-board. The DUT impedance is taken as 45 Ω as it aligns with the typical DUT impedance in the testing environment.

3. **Configuration 3: ATE with Resistor (10 Ω)**—This configuration incorporates a passive resistor component (10 Ω) alongside the ATE to assess the model's handling of reflections due to the passive component, as shown in Figure 6.16.

Simulation Tools

PSpice was used to simulate the PCB trace using the extracted parasitics, and the ATE and DUT were modelled as the resistors at both ends of the trace. The ideal netlist was created for the PCB, and stage

Table 6.8: Parasitic Parameters for Baseline Trace at Different Frequencies

Frequency (MHz)	R (Ω)	L (nH)	C (pF)	G (S)
10	0.922	63.2	27.95	0.00065
50	1.287	61.4	27.92	0.00066
100	1.812	59.15	27.88	0.00067

3 was used to add parasitics and reflections to the netlist. The complete netlist was then simulated using Cadence Spectre. ATE impedance was taken as 50 Ω, and DUT impedance was assumed to be 45 Ω.

6.4.2. Validation Methodology

The validation of stage 3 involved the following steps:

Simulation in PSpice and Stage 3 Model

The three configurations were modelled in PSpice using the extracted parasitic as shown in Table 6.8. The ATE and DUT were modelled as sources and load resistors in PSpice. For the model simulation, an ideal netlist was created for the baseline trace of the custom PCB, in which parasitics and reflection were added using the Python script and Verilog module. After this, the stage 3 model was simulated using Cadence Spectre and compared with the PSpice simulation under identical conditions (same input voltages and frequencies).

.

6.4.3. Results and Analysis

The validation results compare the output voltage of the PSpice and the stage 3 simulation.

Configuration 1: ATE Present

The ATE is present with the loadboard in this configuration, modelled as a 50 Ω resistor after the voltage source. Figure 6.17 compares the output voltages obtained from PSpice and the stage 3 model for the first configuration across different input voltages and frequencies. The simulation results show a high degree of accuracy, with the error percentage being less than 1%. For a 5 V input at 10 MHz, PSpice measures 4.83 V while stage 3 measures 4.82 V, resulting in a difference of 0.21%. Similarly, the difference between the other voltages for 10 MHz is less than 0.3%. For a frequency of 50 MHz, a 10 V input gives an output of 9.68 V from PSpive versus 9.65 V from stage 3, resulting in a difference of 0.31%. The maximum percentage error for 50 MHz is measured at an input of 20 V when the error is around 0.86% as PSpice measures 19.36 V whereas stage measures 19 V. Finally, for the frequency of 100 MHz, PSpice records 38.87 V. In contrast, the stage records 38.76 V for a 40 V input, the error is 0.28. Overall, for the first configuration, the maximum error is recorded to be 0.86% for a frequency of 50 MHz. These minor differences indicate that the stage 3 model accurately models the reflection due to the presence of an ATE.

Configuration 2: ATE and DUT Present

Figure 6.18 compares the output voltages obtained from PSpice and the Stage 3 model for Configuration 2 across different input voltages and frequencies. For an input of 5 V, at 10 MHz, stage 3 measures 4.83 V with an error percentage of around 0.41%; at 50 MHz, PSpice records 4.75 V, whereas stage 3 records 4.73 V. For a frequency of 100 MHz, at 10 V the error percentage comes out to be 0.84% as PSpice measures 9.55 V and stage 3 measures 9.47 V. At the same frequency, PSpice gives an output of 38.08 V and stage 3 gives 37.92 V for an input voltage of 40 V, with the error percentage being 0.42%. The maximum error percentage in this configuration is 0.84%, measured at 10 V and 100 MHz.

Configuration 3: ATE with Resistor (10 Ω)

Figure 6.19 compares the output voltages obtained from PSpice and the Stage 3 model for Configuration 3 across different input voltages and frequencies. For 10 MHz, at an input voltage of 5 V, output voltages are 4.80 V (PSpice) and 4.78 V (Stage 3), reflecting a 0.42% difference. This slight discrepancy arises from the added impedance of the 10 Ω resistor, which introduces additional reflections that

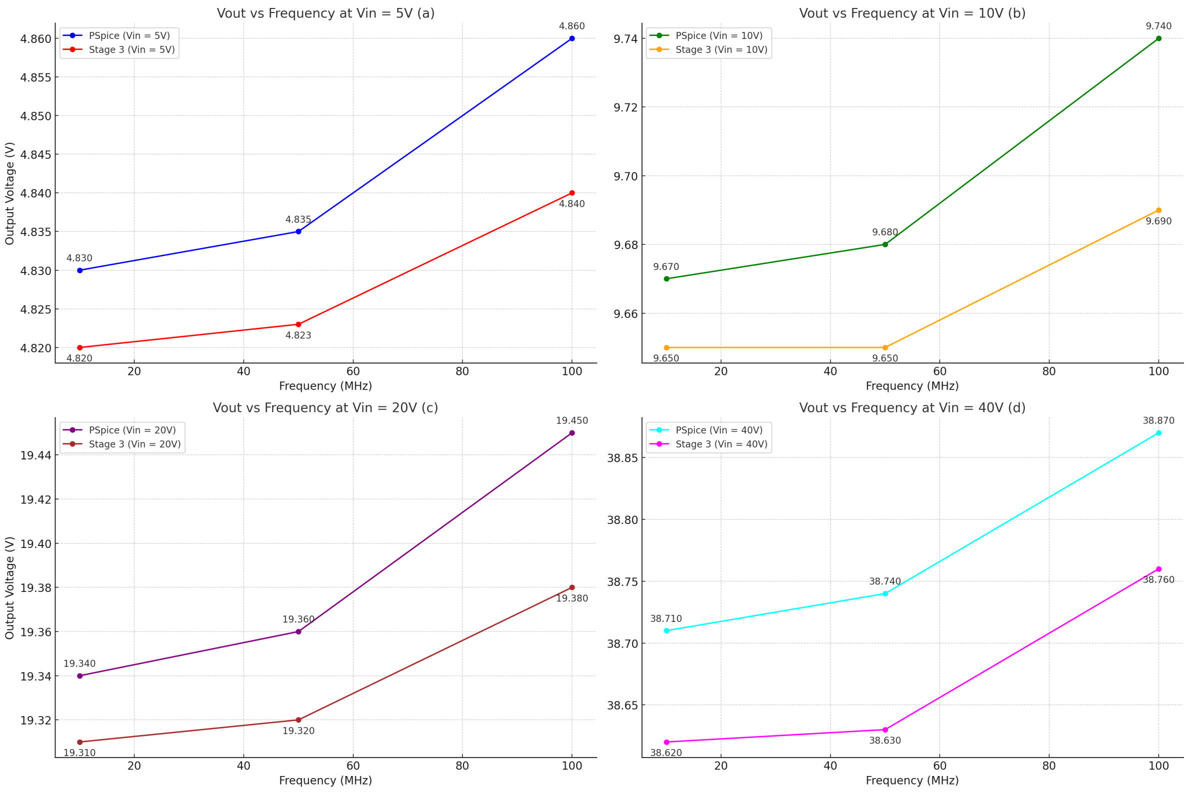


Figure 6.17: PSpice vs. Stage 3 Model Output Voltages for Configuration 1 (ATE Present)

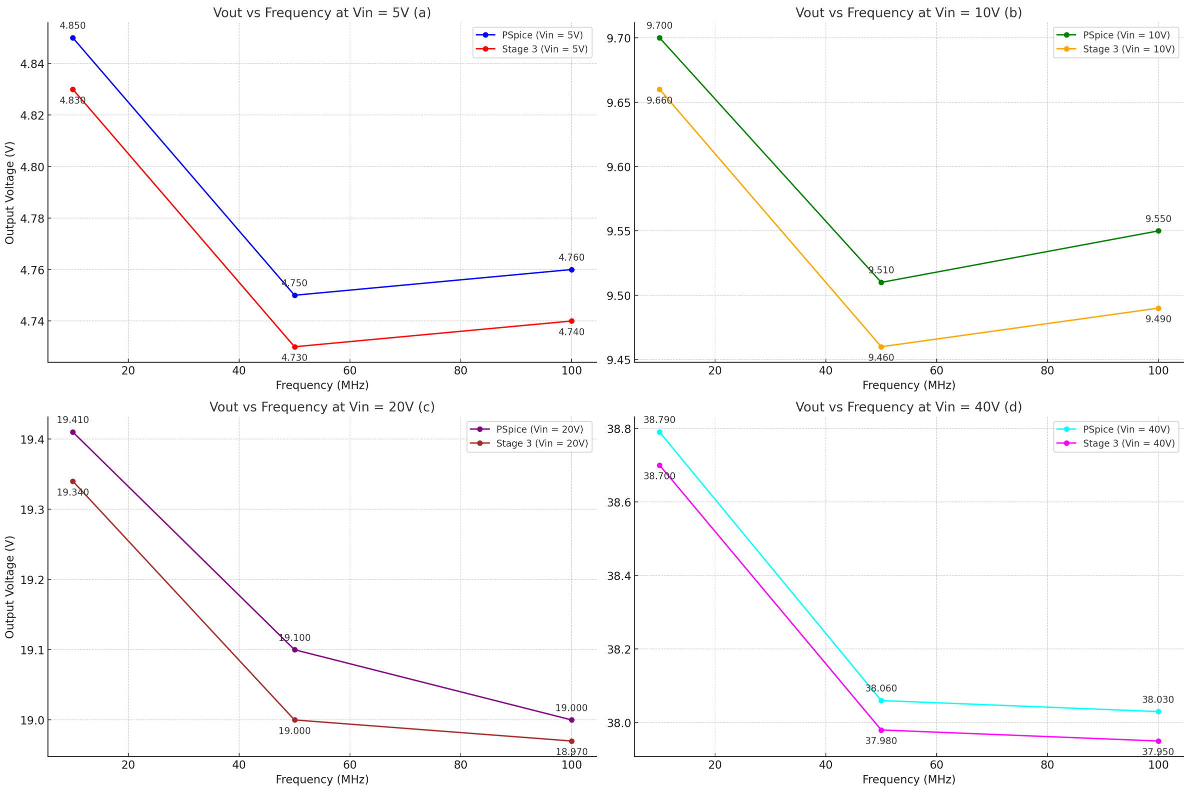


Figure 6.18: PSpice vs. Stage 3 Model Output Voltages for Configuration 2 (ATE and DUT Present)

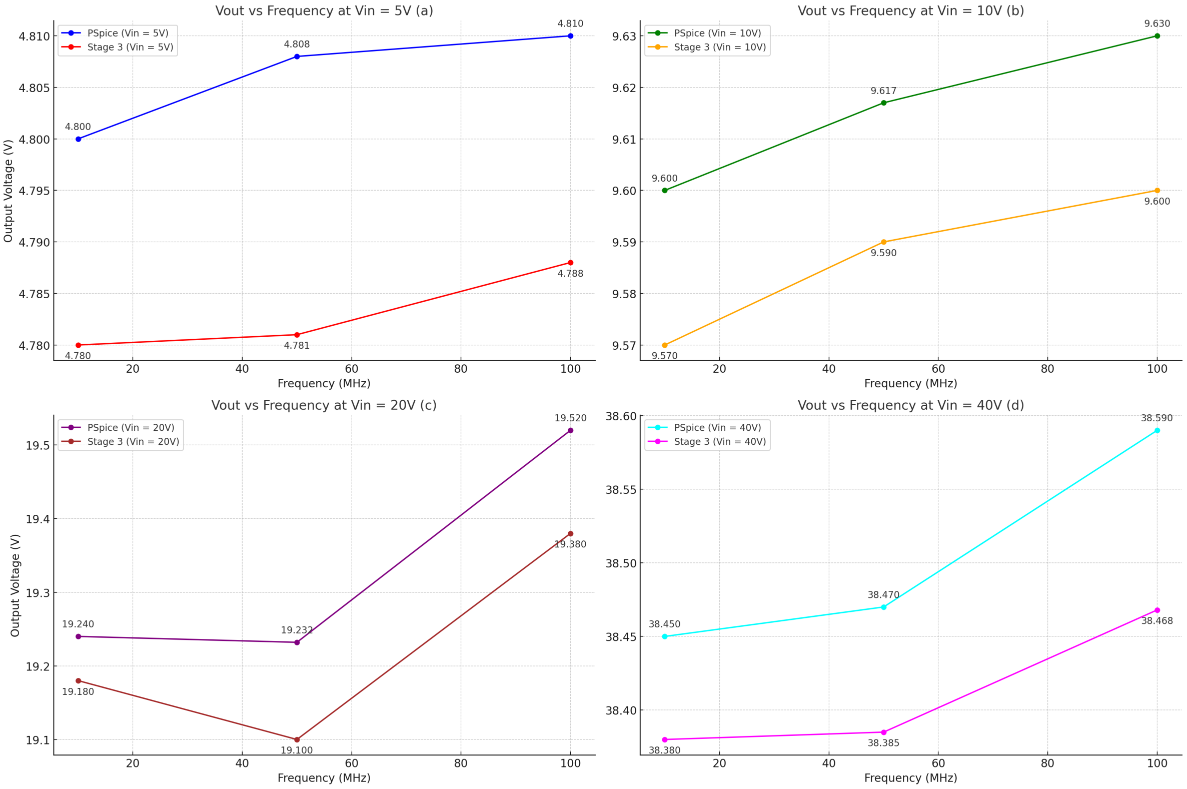


Figure 6.19: PSpice vs Stage 3 Model Output Voltages for Configuration 3 (ATE with Resistor)

the Stage 3 model accurately captures. At 100 MHz, for a 40 V input, output voltages are 38.59 V (PSpice) and 38.28 V (Stage 3), resulting in a 0.81% difference. In the same frequency for an input of 10 V, the record error percentage is 1.56%, the highest error for this configuration as PSpice records 9.63 V and stage records 9.48 V. The percentage differences remain consistently low across all frequencies and input voltages, demonstrating that the Stage 3 model effectively integrates parasitics and reflections introduced by passive components.

Validation Summary

The validation of stage 3 proved that the reflection and parasitic accounted for by the model are very accurate when compared with the PSpice simulation. The maximum error percentage across all three configurations was 1.56% for the third configuration at 10 V and 100 MHz. The overall higher error percentage in configuration three is due to the presence of the passive resistor, which introduces more complex signal interactions and reflections, especially at higher frequencies.

6.5. Full Framework Validtion

After validating the three stages of the framework and the component modelling, it was confirmed that the stages work individually. So, the next step is to validate the full framework against a real-world loadboard to validate how close the model is to its real-world counterpart regarding voltage and timing characteristics.

6.5.1. Experimental Setup

A complex setup with the physical devices was established in the lab for the physical board, and AMS-VT was used for the simulation. The goal is the comparison of the simulation and hardware results.

Simulation Setup

The simulation was conducted in the AMS-VT environment, a closed-loop system in which the ATE model is connected to the Cadence Spectre AMS simulator. In the ATE model, all the activities of the

ATE test program are recorded and stored as stimuli. The stimuli are then used to drive the loadboard signals. The simulation environment also records the response. This allows for realistic simulation of testing scenarios, including forcing voltages and measuring responses.

Physical Setup

The physical setup is shown in Figure 6.20, whose components are:

- **ATE:** The ATE used was an Advantest V93000 system. It was used as the ATE to execute a test program. This ATE sends the signals to the loadboard. The loadboard is attached to the ATE, which ends various voltage levels and signal patterns as the test program directs.
- **Loadboard:** This physical board was available at the NXP lab. The loadboard used is of the dual board configuration to thoroughly check the framework. The frequency of the loadboard was 10 MHz. This was attached to the
- **Oscilloscope:** A Keysight DSOX3054T oscilloscope was used. This measures voltage levels and pulse width at the loadboard output pins. The oscilloscope captures waveform details, precisely comparing simulated and physical signals.

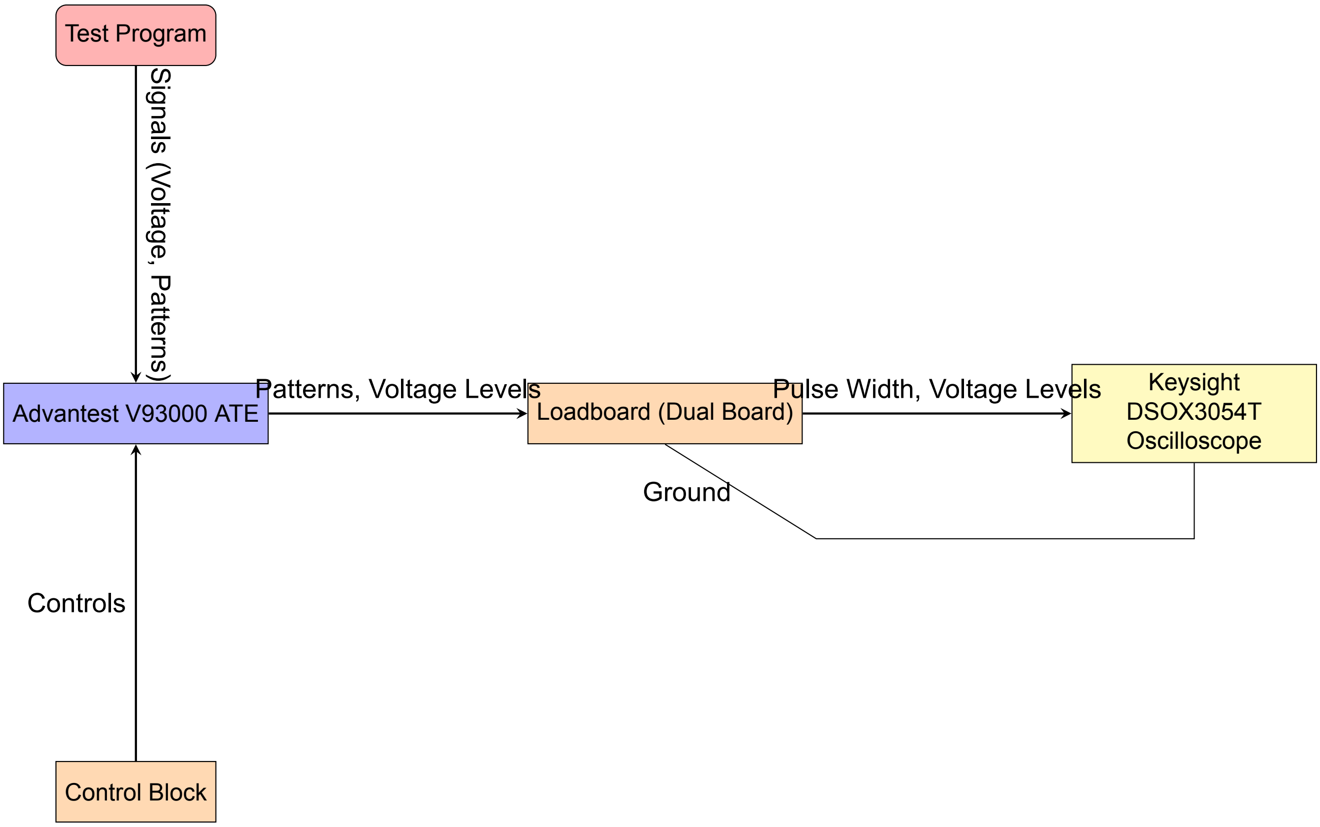


Figure 6.20: Physical Setup for Full Framework Validation

Test Program

The same test program is used in simulation and the physical environment. A new test code was written for this in which the pin is stabilised and then forced to 5V, as shown in Algorithm 8.

Oscilloscope Configuration

The oscilloscope was configured for the following setting:

- **Bandwidth:** 500 MHz, ensuring the capture of signals without attenuation.

Algorithm 8 Pin Voltage Control Test Code

```
Begin
Initialize Pin Voltage:
Set pin voltage to 0V to ensure a clean start and eliminate residual effects.
Execute RDI Block:
Enter Register Direct Interface (RDI) block.
Force pin to a target voltage of 5V using direct register control.
Monitor and Control:
Continuously monitor the pin voltage.
Ensure voltage stability and check for consistency.
Error Checking:
Implement error-checking mechanisms.
Detect and respond to any voltage setting failures or anomalies.
End
```

- **Sampling Rate:** 2 GS/s, providing high-resolution waveform details.

- **Input Impedance:** 1 MΩ, minimizing loading effects on the circuit under test.

- **Triggering Mode:** Edge triggering is set to detect rising edges at a threshold of 1.5 V.

- **Vertical Sensitivity:** 1 V/div, allowing clear observation of voltage levels.

- **Horizontal Scale:** 1 ms/div, allowing clear observation of pulse widths.

6.5.2. Validation Methodology

The framework validation is conducted by evaluating each requirement derived in Chapter 4. This methodology involves executing the test program in the physical lab and AMS-VT to compare their results. The requirements based on which the full framework validation will be done are:

1. **Voltage Accuracy:** The model should be able to predict site-to-site voltage variations within a tolerance of ± 0.1 V for a 5 V input signal, ensuring less than 2% deviation.

2. **Timing Accuracy:** The loadboard model should account for the timing delays due to the physical characteristics of the loadboard, with a timing accuracy within 1% for the pulse width.

3. **Temperature Validation:** The model must include the effects of temperature (from $-40°$C to $125°$C) as that is the operating range.

4. **Automation Level:** The steps in the framework should be completely automated to reduce human errors. The ideal framework would automate schematic conversion to single-site netlist, parasitic extraction, and parasitics integration, thereby minimizing manual intervention.

5. **Speed of Execution:** With automation, the framework should complete the run within 2 hours to not hinder any project timelines.

For the validation, the test program was executed on the ATE, which then generated a 5 V signal across the loadboard. The applied voltages were then measured by an oscilloscope. The same test program was then executed in AMS-VT, and the results were analysed using SimVision. Timing delays were measured by applying a pulse signal from 5 to 6.5 V and then back to 5V, measuring the time taken for the signal to reach 6.5 V and then go back to 5 V. The pulse width is measured at 50% of the voltage increase, which means the measurement was taken from 5.75 to 5.75 V for rise and fall events.

6.5.3. Result and analysis

The validation results are analysed against each framework's requirements. The following subsections present the findings for each requirement.

Table 6.9: Physical vs. Full Framework Output Voltages at 5V Input

Site	Physical (V)	Framework (V)
1	5.09	5.11
2	5.08	5.1
3	5.02	5.06
4	5.06	5.12
5	5.11	5.13
6	5.05	5.09
7	5.11	5.15
8	5.12	5.14

Table 6.10: Physical vs. Ideal vs. Final Framework Pulse Widths

Measurement Type	Pulse Width (ms)	Difference from Physical (ms)
Physical	10.865	0
Simulation Ideal Netlist	10.685	0.18
Simulation with Parasitics and Reflections	10.82	0.045

Voltage Accuracy

Table 6.9 compares the voltages measured physically and in the simulation when an input of 5 V is provided across all eight sites of the loadbord. The framework voltage prediction proves to be of high accuracy with percentage difference for Site 1 being 0.39%, Site 2 being 0.3%, Site 3 being 0.8%, Site 4 being 1.17%, Site 5 being 0.39%, Site 6 being 0.79%, Site 7 being 0.78%, and Site 8 being 0.37%. The framework achieves a voltage accuracy with a tolerance of 1.17%, which is within the requirements of less than 2% deviations for a 5 V input. The highest discrepancy was observed at Site 4, where the physical result was 5.06 V. The framework predicts 5.12 V. This verifies that the framework adheres to requirement **R3 (Voltage Accuracy)**.

Timing Accuracy

Table 6.10 compares the pulse widths of the signal from the time it goes from 0 to 5 V until it goes back to 0 V. The pulse width of the physical lab is compared with the ideal netlist and the entire framework netlist. The timing accuracy is evaluated based on the deviation of simulated pulse widths from physical measurements. The ideal netlist shows a 1.66% deviation from the physical value. However, the final framework incorporating parasitics and reflections significantly improves the accuracy, reducing the discrepancy to 0.41%. This meets the requirements of having a tolerance of less than 1% for the timing accuracy of pulse width. This verifies that the framework adheres to requirement **R3 (Timing Accuracy)**.

Temperature Validation

The thermal simulation in PowerDC for baseline trace in custom PCB made in Section 6.2, for 10 MHz, was executed to check the effects of temperature on the overall voltage. The temperature sweep was done from -10 to 50 °C for an input voltage of 5 and 40 V. The output from both simulations can be seen in Figure 6.21. As expected, the output voltage decreases slightly when temperature increases due to increased resistance. At 5 V and -10°C, the output voltage is approximately 4.9931 V in PowerDC and 4.993 V in the framework showing a negligible difference. As temperature increases, the output voltage decreases to 4.9912 V in PowerDC and 4.9911 V in the framework. Similarly, at 40 v, the difference between them is 0.0017 V, which is higher than at 5 V but still negligible. This decrease in output voltage due to temperature is consistent in the results of both simulations. There is a slight discrepancy between the output voltages of PowerDC and the framework; this is because the framework only accounts for temperature effects on R, while PowerDC also accounts for the very minor temperature effects on L, C, and G. This verifies that the framework adheres to requirement **R3 (Operating Conditions)** for temperature.

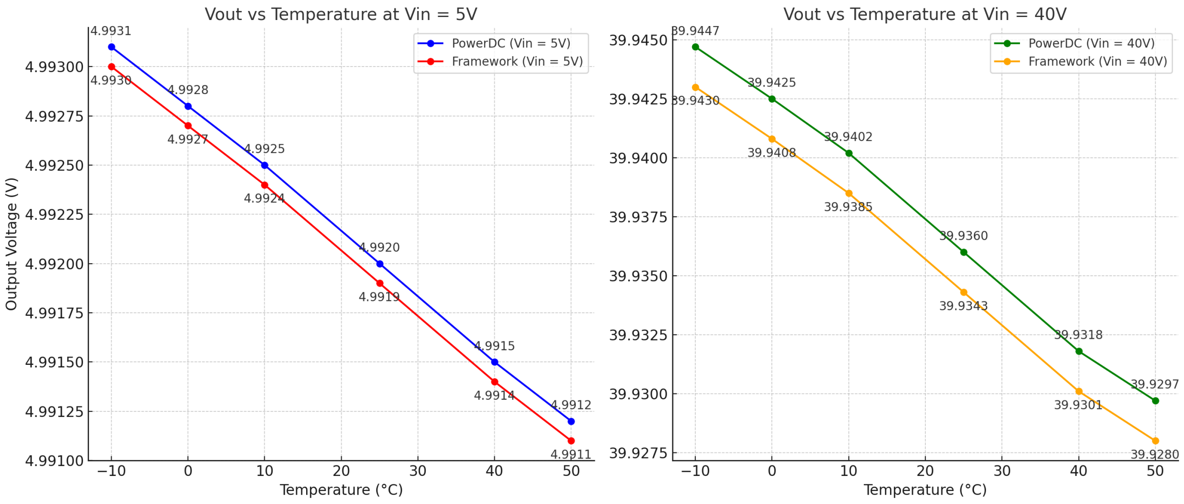


Figure 6.21: Temperature: Comparison between PowerDC and Framework

Automation Level

The framework automation capabilities are evaluated based on the following criteria:

- **Schematic to Ideal Single-Site:** The framework automates the conversion of Allegro's multisite netlist to a single-site netlist. Previously, generating a multisite netlist from Allegro must be done manually. Still, since this does not require any changes and is not time-consuming, the overall effect on the automation level is negligible.
- **Parasitics Extraction:** A Python script was developed that automates the extraction of frequency-based parasites from PowerSI without requiring manual intervention.
- **Parasitics and Reflection Integration:** The integration of parasitics and reflection in the ideal single-site netlist is fully automated using a Python script and a Verilog-AMS module.

Overall, the framework fulfils the automation requirements as it automates all the complex processes. The only manual process remaining is the generation of the Allegro netlist, which only requires loading the schematic to Allegro and then generating the netlist. This verifies that the framework adheres to requirement **R5 (Automation Level)**.

Execution Speed

The framework's execution speed was measured to ensure that it completed the generation of the model within the set duration, which was 2 hours. The framework completed the execution for a dual board configuration in approximately 1.5 hours, which is well within the required criteria. High levels of automation optimized parasitic extraction algorithms, making them faster as they took most of the run time. This verifies that the framework adheres to requirement **R8 (Speed of Execution)**.

6.6. Discussion

The comprehensive validation of the complete framework against the specified requirements proves that it generates an efficient, accurate, and reliable laodboard model that can be used directly in virtual testing. The framework successfully calculated the site-to-site variations and decreased the error percentage in the pulse width. The framework can be graded based on the requirements. Four levels are used to grade the framework:

- **"+ +":** This represents optimal performance, which means no more improvement is required.
- **"+":** This means satisfactory performance, indicating that some improvements can be made but still acceptable.

Table 6.11: Framework Requirements

Metric	Required Score	Justification
Voltage Accuracy	+ +	The model is able to predict site-to-site voltage variations for all the sites on the loadboard.
Timing Accuracy	+	There is a substantial decrease in the error percentage of pulse width to 0.41%. The rise and fall time can also be considered for more accuracy in timing.
Frequency and Temperature Effects	+ +	Frequency and temperature effects are included in the generated model.
Automation Level	+	The complex steps in the framework are completely automated to reduce human errors. There is a simple one-step that is not automated.
integration with AMS-VT	++	The model generated by the framework integrates into the AMS-VT environment with minimal effort.
Component Modeling	+	The effects of the relay on voltage and timing are considered. It is a static model that can be converted to a dynamic one if power loss is considered.
Generalization	++	The framework handle different configurations of the loadboard with ease.
Speed of Execution	++	With automation, the framework takes 1.5 hours.

- **"–":** This represents below marginal performance, indicating a need for improvement to reach an adequate level.
- **"– –":** This means poor performance, indicating a need for significant improvements.

The grade of the framework based on the requirements and their reasoning is shown in Table 6.11.

7 Conclusion

7.1. Summary

Due to the increasing use of ICs in critical sectors, the semiconductor industry's quality requirements have become tighter. Due to the higher quality requirements, chip testing to ensure quality is becoming increasingly essential in semiconductor manufacturing. The industry is also seeing an increase in the complexity of the ICs due to the demands of functional requirements. This increasing complexity, combined with tighter quality standards, is leading to a rise in the test development time of an IC. This increase in test development time is further contributing towards an increasing time-to-market. So, there is an urgent need to optimise test development flow to reduce the time-to-market. Virtual testing is a methodology that can be employed in the test development flow to start the debugging process pre-silicon which is currently being done post-silicon. However, it is currently not being utilised due to the absence of an accurate loadboard model to connect the tester and DUT model. Therefore, this thesis created a framework that integrates parasitic effects and signal reflections due to impedance mismatch into the loadboard netlist and creates an accurate loadboard model for pre-silicon virtual testing environments. Thus, the primary research question was centred around integrating different methodologies in a single flow to generate a virtual model of loadboard. Further, how can this flow be automated to reduce the probability of human errors? The thesis aimed to address these concerns by streamlining the flow and integrating the different EDA tools with some developed models to achieve digital twinning of the loadboard.

Chapter 2 explored the fundamentals of semiconductor testing. It briefly introduces the complete semiconductor manufacturing process, explaining the design and wafer fabrication stages. Then, the testing stage is introduced, where two primary types of testing are discussed: wafer and final testing. After this, the test development flow is discussed, which explains the complete process of test development, which starts from defining specifications and then goes on to test planning, test program development, test file creation, and test program debugging. After describing the test development process, the testing hardware, ATE, and loadboard were discussed in detail. Finally, the chapter discusses the parasitic in the loadboard and how they can introduce reflections due to impedance mismatching, which affects signal integrity.

Chapter 3 then discusses the state-of-the-art methodologies in virtual testing and digital twinning. First, the development of virtual testing over the years is discussed. It was noted that companies like Infineon tried using virtual testing in their test development flow, and they saw a decrease in their time-to-market by weeks. After this, the latest virtual testing methodology, the AMS-VT developed by Testinsight, is discussed, and NXP currently uses it. AMS-VT had certain limitations, which prevented it from being used to its full capability, and the primary limitation was the absence of an accurate loadboard model. The concept of digital twinning can be employed to create an accurate model. Then, a digital twinning technique was presented, which involved creating an ideal netlist and combining parasitics extraction to make an accurate model. Then, the possible EDA tools Cadence Allegro and PowerSI were selected for ideal netlist generation and parasitics extraction, respectively.

Chapter 4 analysed the current methodology for loadboard modelling and derived requirements. The requirements are voltage accuracy, timing accuracy, operating conditions, component modelling, automation level, integration with AMS-VT, generalization, and execution speed. After the derivation of requirements, an overview of the proposed solution was presented that could be used to counter the inadequacies found in the previous chapter. These solutions focused on developing an accurate netlist that could capture the loadboard's parasitics, integrate reflections caused by mismatches, and be compatible with tools like AMS-VT. Chapter 5 goes deeper into implementing the proposed solution by dividing it into three stages. The first stage was the generation of an ideal single-site netlist with the format of the current virtual testing environment, ensuring compatibility with the existing virtual testing setup. After generating an ideal netlist, tools like PowerSI extracted parasitic parameters (R, L, C, G) from the loadboard layout, capturing critical parasitic effects for the signal traces. The parasitic extraction from PowerSI was further automated to reduce manual errors during simulations. The final stage involved parasitic analysis to calculate the channel's characteristic impedance, which was then compared with the source or load impedance, which showed the possibility of a mismatch and resulting reflection, consequently changing voltage magnitude at the output. A Verilog module was developed and added to the ideal netlist by a Python script to model this change. This module successfully added the parasitic as distributed elements for all channels of that site and also added the effects of an impedance mismatch to calculate the output voltage accurately. A static model of the SSR relay was also developed that considers the effects of main parameters, such as on-state resistance, output capacitance, off-state current leakage, switching times, and voltage and timing. Finally, a master script was designed to communicate between different stages and make the process streamlined and error-free.

The implementation was then validated in Chapter 6, where each stage of the framework was validated separately, and then a full framework validation was done. The first stage, where an ideal netlist is generated, was validated by comparing it with a golden simulation, proving the netlist generated can be easily integrated into AMS-VT without requiring manual changes. Then, the extracted parasitic was validated by creating a test PCB with multiple configurations. The extracted parasitic from the algorithm were simulated in PSpice and compared with the results from PowerDC where an IR drop analysis was performed on the channel of the test PCB. This proved that the extracted parasitic from the algorithm matched with the PowerDC results with an error of less than 1%. Stage 3 was then validated, where its algorithm was executed on the baseline configuration of the test PCB, and the final netlist was then simulated for multiple voltages. The results of this simulation were compared with the PSpice results, which proved that the algorithm was very accurate with an error of less than 1%. Finally, the entire framework was validated by comparing it with the results of a physical loadboard of NXP. The generated netlist gave an error of less than 2% for voltage accuracy, and the timing error for a pulse width was reduced to 0.41% from 1.6%. The generated model was also verified for frequency and temperature, proving that the model accounts accurately for both these factors. The framework generated an accurate model of the loadboard in 1.5 hours, which was also well within the requirement of 2 hours. The framework was used on multiple loadboard configurations, proving its flexibility. The SSR model was also validated with its datasheet to check if the model accurately represents its characteristics.

7.2. Limitations and Future Work

While the proposed framework provides accurate modelling of loadboard for pre-silicon virtual testing, some limitations present opportunities for future work.

Firstly, the framework currently provides an accurate model for lower-frequency circuits (below 1 GHz), but it faces limitations when applied to higher frequencies, especially radio frequencies. These limitations arise because, as frequency goes beyond 1 GHz, the length of the loadboard traces becomes comparable to the wavelength. Beyond this frequency, the model loses accuracy because additional effects such as multiple reflections, ringing, and crosstalk become more apparent. The current framework assumes that reflections occur primarily due to impedance mismatches at the ends of the loadboard and at passive components, which is valid for low frequencies. However, multiple reflections occur at higher frequencies along the trace's length due to bends and vias. These reflections cause ringing and other signal integrity issues, which are not accounted for in the current framework. To address this, the effects of bends and vias along the trace lengths can be incorporated to consider their impact on the signal. This can be achieved by further segmenting the trace into smaller sections and quantifying the effects of bends and vias on the signal by calculating their reflection. Second, the framework can be extended to include crosstalk and coupling effects by incorporating electromagnetic simulations that capture these interactions between ground planes.

Secondly, the static model of the SSR relay used in the current framework does not accurately represent its dynamic behaviour. The SSR's dynamic characteristics in high-speed applications, such as switching transients, delay variations, and frequency-dependent effects, also become significant. The relay's switching states introduce parasitic effects that affect signal timing and voltage, which are not modelled in the current framework. Developing a dynamic SSR model would overcome these limitations. This can be done by considering the loss of power during the switching of the relay, which can then be used to calculate its effects on different parameters.

Thirdly, the current framework assumes that temperature only affects resistance, whereas the inductance and capacitance are also affected by temperature, although the effect is less. This effect of temperature on L and C becomes more apparent in environments with significant temperature variations, which can then affect the signal integrity. The framework can be enhanced to account for temperature-dependent parasitic behaviour in inductance and capacitance. This can be done by introducing temperature-dependent functions for these parasitics, ensuring the model remains accurate on a broader range of operating conditions.

Additionally, the framework can be expanded by adding other minor effects that are:

- **Power Integrity Issues:** Parameters such as power supply noise and ground bounce can be added to account for their effects, especially in power-sensitive applications.
- **Manufacturing Variations:** The effect due to variations in dielectric constant can also be added to make the model more robust

Finally, the framework can be validated for various loadboards, even with designs outside of NXP, to check its flexibility in handling different configurations. The framework can also be tested for high-frequency and high-power applications to verify its accuracy. These boards introduce additional complexities such as trace bends, vias, and plane effects, which must be modelled to simulate signal behaviour accurately. The framework can be further refined to handle different applications by addressing these limitations. In conclusion, while the current framework provides an accurate model of loadboard, future enhancements are necessary to broaden its capability for various applications.

Bibliography

[1] Mario A Bolanos. “Semiconductor integrated circuit packaging technology challenges-next five years”. In: *2005 International Symposium on Electronics Materials and Packaging*. IEEE. 2005, pp. 6–9.

[2] Shubham Kumar and Ankaj Gupta. *Integrated circuit fabrication*. Taylor and Francis, CRC Press, 2021, p. 352. doi: `https://doi.org/10.1201/9781003178583`.

[3] *Semicondutor Revenue*. Accessed: 25th July 2024. url: `https://www.wsts.org/67/Historical-Billings-Report`.

[4] Stefan R Vock et al. “Challenges for semiconductor test engineering: a review paper”. In: *Journal of Electronic Testing* 28.3 (2012), pp. 365–374.

[5] Chen He. “Automotive Semiconductor Test: Challenges and Solutions towards Zero Defect Quality”. In: *2023 45th Annual EOS/ESD Symposium (EOS/ESD)*. Vol. EOS-45. 2023, pp. 1–11. doi: `10.23919/EOS/ESD58195.2023.10287730`.

[6] Yangyuan Wang et al. *Handbook of Integrated Circuit Industry*. Springer Nature, 2023.

[7] Gordon E Moore. “Cramming more components onto integrated circuits”. In: *Proceedings of the IEEE* 86.1 (1998), pp. 82–85.

[8] Sebastian Pointner et al. “Test Your Test Programs Pre-Silicon: A Virtual Test Methodology for Industrial Design Flows”. In: *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2019, pp. 241–246. doi: `10.1109/ISVLSI.2019.00052`.

[9] “Transistors per microprocessor”. In: *Karl Rupp, Microprocessor Trend Data (2022) – processed by Our World in Data*. Accessed: 25th July 2024.

[10] Adrian I. Voinea and Stefan Kampfer. “Rapid prototyping and test before silicon of integrated pressure sensors”. In: *2015 IEEE International Test Conference (ITC)*. 2015, pp. 1–9. doi: `10.1109/TEST.2015.7342402`.

[11] M. Miegler and W. Wolz. “Development of test programs in a virtual test environment”. In: *Proceedings of 14th VLSI Test Symposium*. 1996, pp. 99–103. doi: `10.1109/VTEST.1996.510842`.

[12] W.R. Dearborn et al. “The Virtual Test program (VTest)”. In: *1998 IEEE AUTOTESTCON Proceedings. IEEE Systems Readiness Technology Conference. Test Technology for the 21st Century (Cat. No.98CH36179)*. 1998, pp. 149–159. doi: `10.1109/AUTEST.1998.713435`.

[13] Q.M. Li, F.C. Lee, and T.G. Wilson. “Design verification and testing of power supply system by using virtual prototype”. In: *IEEE Transactions on Power Electronics* 18.3 (2003), pp. 733–739. doi: `10.1109/TPEL.2003.810845`.

[14] Ping Lu et al. “Mixed-signal test development using open standard modeling and description languages”. In: *2009 IEEE Behavioral Modeling and Simulation Workshop*. 2009, pp. 78–83. doi: `10.1109/BMAS.2009.5338885`.

[15] Stefan Mihai et al. “Digital twins: A survey on enabling technologies, challenges, trends and future prospects”. In: *IEEE Communications Surveys & Tutorials* 24.4 (2022), pp. 2255–2291.

[16] Saleh Kalantari et al. “Developing and user-testing a “Digital Twins” prototyping tool for architectural design”. In: *Automation in Construction* 135 (2022), p. 104140.

[17] *Virtual ATE*. Accessed: 25th July 2024. url: `https://www.testinsight.com/vt-virtual-ate/`.

[18] *TestInsight AMS-VT*. Accessed: 6th August 2024. url: `https://www.testinsight.com/ams-vt-mixed-signal-virtual-ate/`.

[19] Shrey Gupta. *Loadboard Digital Twin Framework Code Repository*. Accessed: 21st October 2024. url: `https://gitlab.tudelft.nl/msc/shrey-gupta-loadboard-simulation.git`.

[20] Chue San Yoo. *Semiconductor manufacturing technology*. Vol. 13. World Scientific, 2008.

[21] Badih El-Kareh and Lou N Hutter. *Fundamentals of semiconductor processing technology*. Springer Science & Business Media, 2012.

[22] Ian A Grout. *Integrated circuit test engineering: modern techniques*. Springer Science & Business Media, 2005.

[23] R Jacob Baker. *CMOS: circuit design, layout, and simulation*. John Wiley & Sons, 2019.

[24] Michael L Bushnell, Vishwani D Agrawal, et al. "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits [electronic resource]". In: ().

[25] Sudarshan Bahukudumbi and Krishnendu Chakrabarty. *Wafer-level testing and test during burn-in for integrated circuits*. Artech House, 2010.

[26] N Sameeksha Rai, Namita Palecha, and Mahesh Nagarai. "A brief overview of Test Solution Development for Semiconductor Testing". In: *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. 2019, pp. 205–209. doi: `10.1109/RTEICT46194.2019.9016857`.

[27] Keith Brindley. *Automatic test equipment*. Elsevier, 2013.

[28] *Advantest ATE*. Accessed: 28th July 2024. url: `https://www.advantest.com/en/`.

[29] Justin Moses. "Design and Testing of an Electronic Load Board for Power Supply Validation and Verification". PhD thesis. Auburn University, 2014.

[30] David E McFeely. "The process and challenges of a high-speed DUT board project". In: *Proceedings. International Test Conference*. IEEE. 2002, pp. 565–573.

[31] Yolando G. Yabut et al. "Device Under Test (DUT) Socket Conversion to Improve Analog Loadboard Performance Efficiency". In: *2006 Thirty-First IEEE/CPMT International Electronics Manufacturing Technology Symposium*. 2006, pp. 492–495. doi: `10.1109/IEMT.2006.4456500`.

[32] Chongjun Sun and Minghu Zhang. "Automated Calibration Method of Loadboard of Integrated Circuit ATE". In: *2018 IEEE International Conference on Electronics and Communication Engineering (ICECE)*. 2018, pp. 130–133. doi: `10.1109/ICECOME.2018.8644921`.

[33] NA Mutalib, MS Mispan, and N Hashim. "INTELLIGENT INTERFACE BOARD COMPONENT CODE GENERATOR FOR AUTOMATED TESTING EQUIPMENT". In: *Journal of Engineering and Technology (JET)* 14.1 (2023), pp. 121–135.

[34] S.M. Low et al. "Verification of trace length and trace impedance of fabricated load board using TDR". In: *2009 1st Asia Symposium on Quality Electronic Design*. 2009, pp. 401–404. doi: `10.1109/ASQED.2009.5206229`.

[35] David Reusch and Johan Strydom. "Understanding the effect of PCB layout on circuit performance in a high-frequency gallium-nitride-based point of load converter". In: *IEEE Transactions on Power Electronics* 29.4 (2013), pp. 2008–2015.

[36] Tonio Biondi et al. "Distributed modeling of layout parasitics in large-area high-speed silicon power devices". In: *IEEE Transactions on Power Electronics* 22.5 (2007), pp. 1847–1856.

[37] U.A. Bakshi and L.A.V. Bakshi. *Communication Network & Transmission Lines*. Technical Publications, 2020. isbn: 9789333223737. url: `https://books.google.co.in/books?id=8E4bEAAAQBAJ`.

[38] Wei-Da Guo et al. "Reflection enhanced compensation of lossy traces for best eye-diagram improvement using high-impedance mismatch". In: *IEEE Transactions on Advanced Packaging* 31.3 (2008), pp. 619–626.

[39] G. Gary Wang. "Definition and Review of Virtual Prototyping ". In: *Journal of Computing and Information Science in Engineering* 2.3 (Jan. 2003), pp. 232–236. issn: 1530-9827. doi: `10.1115/1.1526508`. eprint: `https://asmedigitalcollection.asme.org/computingengineering/article-pdf/2/3/232/5637643/232\_1.pdf`. url: `https://doi.org/10.1115/1.1526508`.

[40] Ping Lu et al. “A novel approach to entirely integrate Virtual Test into test development flow”. In: *2009 Design, Automation & Test in Europe Conference & Exhibition*. 2009, pp. 797–802. doi: 10.1109/DATE.2009.5090772.

[41] Michel Nagel et al. “Virtual PCB Layout Prototyping: Importance of Modeling Gate Driver and Parasitic Capacitances”. In: *2022 IEEE Design Methodologies Conference (DMC)*. 2022, pp. 1–5. doi: 10.1109/DMC55175.2022.9906542.

[42] Dwight Howard. “The Digital Twin: Virtual Validation In Electronics Development And Design”. In: *2019 Pan Pacific Microelectronics Symposium (Pan Pacific)*. 2019, pp. 1–9. doi: 10.23919/PanPacific.2019.8696712.

[43] Pablo González-de-Aledo et al. “Towards a Verification Flow Across Abstraction Levels Verifying Implementations Against Their Formal Specification”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.3 (2017), pp. 475–488. doi: 10.1109/TCAD.2016.2611494.

[44] Jens Lienig and Juergen Scheible. *Steps in Physical Design: From Netlist Generation to Layout Post Processing*. Cham: Springer International Publishing, 2020, pp. 165–211. isbn: 978-3-030-39284-0. doi: 10.1007/978-3-030-39284-0_5. url: https://doi.org/10.1007/978-3-030-39284-0_5.

[45] *Allegro System Capture*. url: https://www.cadence.com/en_US/home/tools/pcb-design-and-analysis/design-authoring/allegro-system-capture.html.

[46] *EasyEDA*. Accessed: 14th August 2024. url: https://easyeda.com/.

[47] *Altium Designer*. Accessed: 14th August 2024. url: https://www.altium.com/altium-designer.

[48] *Autodesk Fusion 360*. Accessed: 14th August 2024. url: https://www.autodesk.com/in/products/fusion-360/overview?term=1-YEAR%5C&tab=subscription.

[49] Taigon Song et al. “Full-chip signal integrity analysis and optimization of 3-D ICs”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.5 (2015), pp. 1636–1648.

[50] *Sigrity-Aurora*. Accessed: 14th August 2024. url: https://www.cadence.com/en_US/home/tools/sigrity-x.html#sigrity-x-aurora.

[51] *Sigrity-PowerSI*. Accessed: 14th August 2024. url: https://www.cadence.com/en_US/home/tools/sigrity-x.html#sigrity-x-powersi.

[52] *Keysight ADS*. Accessed: 14th August 2024. url: https://www.keysight.com/in/en/products/software/pathwave-design-software/pathwave-advanced-design-system.html.

[53] *Siemens hyperLynx*. Accessed: 14th August 2024. url: https://eda.sw.siemens.com/en-US/pcb/hyperlynx/.

[54] Joonhyun Kim and Yungseon Eo. “IC Package Interconnect Line Characterization Based on Frequency-Variant Transmission Line Modeling and Experimental S-Parameters”. In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 9.6 (2019), pp. 1133–1141. doi: 10.1109/TCPMT.2019.2898671.

[55] Louis Scheffer, Luciano Lavagno, and Grant Martin. *EDA for IC system design, verification, and testing*. CRC press, 2018.

[56] Vladimir Gurevich. “The International Standard on Solid-State Relays (IEC 62314, Ed. 1) Critical Review”. In: *8th January* (2024).

[57] *Solid State Relays*. url: https://www.ia.omron.com/support/guide/18/introduction.html.

[58] Wes McKinney. *Python for data analysis*. ” O’Reilly Media, Inc.”, 2022.

[59] Katherine A Kim et al. “A dynamic photovoltaic model incorporating capacitive and reverse-bias characteristics”. In: *IEEE Journal of photovoltaics* 3.4 (2013), pp. 1334–1341.