Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Description
RTL	Register Transfer Level
ML	Machine Learning
IC	Integrated Circuit
ICs	Integrated Circuits
TB	Test Bench
CRV	Constraint Random Verification
FPV	Formal Property Verification
ABV	Assertions Based Verification
AI	Artificial Intelligence
UVM	Universal Verification Methodology
LLMs	Large Language Models
SVA	SystemVerilog Assertions
VLSI	Very Large-Scale Integration
DUT	Design Under Test
DV	Design Verification
DNNs	Deep Neural Networks
RL	Reinforcement Learning
SV	SystemVerilog
GUI	Graphical User Interface
PPM	Post-Processing Module
SVMs	Support Vector Machines
HDL	Hardware Description Language

Table 1 List of Abbreviations

Abstract
Translating high-level digital design specifications into Register Transfer Level (RTL) implementations poses a significant challenge in digital design. Discrepancies between specifications and RTL code often arise, including redundant, unused, or invalid code, compromising design quality and efficiency. To address this challenge, using Large Language Models (LLMs) such as GPT-4 presents a promising opportunity for automated equivalence checking in digital design workflows. This research project aims to explore the potential of GPT-4 for equivalence checking and proposes a novel methodology that capitalizes on the model's ability to process and comprehend complex information. The proposed methodology involves training GPT-4 on a comprehensive dataset of paired design specifications and corresponding RTL implementations to establish a reference point for identifying potential discrepancies during the design process. Subsequently, the trained LLMs will analyze new design specifications and compare them to existing RTL code, automatically highlighting any inconsistencies. This approach holds significant promise for enhancing design quality by ensuring accurate implementation, eliminating unnecessary code, and improving communication between architects, designers, and validators. The research project will collect and preprocess a diverse dataset of paired digital design specifications and RTL designs to train GPT-4, enabling it to learn their relationships. An automated equivalence checker utilizing the trained LLMs will be developed and evaluated on representative designs. The project's overarching goal is to enhance design quality, reduce design time, improve communication, and streamline the design workflow. Leveraging the powerful capabilities of GPT-4, this research project has the potential to revolutionize the verification of digital designs, contributing significantly to the field of digital design by ensuring higher quality and more efficient design processes.

1.0 Introduction

The digital design process begins with defining the architecture or specifications before delving into the Register Transfer Level (RTL) designs. However, discrepancies between the design specifications and the RTL designs can arise due to miscommunications among architects, designers, and validators. It could lead to redundant, irrelevant, unused, or invalid RTL code issues. These discrepancies may result from miscommunications among the architects, designers, and validators. Large Language Models (LLMs) can be leveraged to compare and validate the RTL designs against the design specifications to address this challenge. This approach holds the potential to ensure that the RTL designs align accurately with the specified requirements, thereby mitigating the occurrence of discrepancies and enhancing the overall digital design flows (Hu et al., 2020).

2.0 Research Background

Register Transfer Level (RTL) is a crucial aspect of digital design, serving as an intermediate step between defining the architecture or specifications and the actual digital design flows (Nane et al., 2016). It involves transferring data between registers within a digital system, and it is a fundamental abstraction level for digital design (Zergainoh et al., 2006). Digital design encompasses creating, implementing, and testing digital circuits or systems composed of logic gates and interconnected components (Zemva et al., 1998). The digital design flows typically commence with the definition of architecture or specifications, followed by RTL designs, verification, synthesis, and the implementation of the physical design (Trost & Zemva, 2012). The architects play a pivotal role in defining the high-level architecture and specifications of digital design, providing the foundational framework for the subsequent design stages (Jutraz & Zupancic, 2014). The designers translate the architectural specifications into RTL designs, ensuring the digital system meets the specified requirements (Becker et al., 2014). Conversely,

the validators are tasked with rigorously assessing and verifying the RTL designs to ensure compliance with the defined specifications, thereby identifying discrepancies or errors (Rodrigues et al., 2008).

LLMs are characterized by their ability to understand and generate human-like text. These models are trained on vast amounts of text data, enabling them to capture complex patterns and relationships within the language. The capabilities of LLMs, such as GPT-4, have been demonstrated across various domains, including medicine, natural language processing, and scientific research. For instance, studies have shown that GPT-4 achieves high accuracy in medical knowledge assessment, with performance exceeding the threshold for passing scores in medical examinations. Furthermore, the model has been leveraged to identify clinical phenotypes in electronic health records and demonstrate its effectiveness in processing and analyzing medical data. In digital design, these capabilities translate to the potential for GPT-4 to accurately validate RTL designs against design specifications, thereby addressing the challenges associated with discrepancies/miscommunications in the design review and enhancing the accuracy of design specifications/RTL designs.

3.0 Problem Statement

The digital design flows begin with establishing architecture or specifications before the RTL is designed. However, a common issue arises when discrepancies occur between the design specifications documented by the architects and the actual RTL designs despite the completion of design reviews. This discrepancy can lead to the inclusion of redundant, irrelevant, unused, or invalid RTL code. Miscommunications among the architects, designers, and validators further compound the problem, as the limited time available for design reviews often results in an incomplete review of the RTL code. Additionally, the lack of a precise design review between the architects and designers means that not every line of RTL code is thoroughly reviewed. Consequently, the validators, who typically communicate with the designers, face challenges properly verifying the digital design to ensure better or full coverage. This challenge is exacerbated by the designers' potential lack of the correct/accurate mindset regarding the digital design specifications, which may lead to the transmission of incorrect messages to the validators, further complicating the verification process, which has missed some of the important test sequences/assertions/checkers.

The need for this research study arises from the critical issues in digital design verification that have not been adequately addressed in existing literature. Specifically, the study aims to tackle the challenges associated with discrepancies between design specifications and the actual RTL designs and the communication gaps among architects, designers, and validators. These issues include redundant, irrelevant, unused, or invalid RTL code, which can go undetected due to incomplete design reviews and miscommunications. By addressing these specific issues and exploring the novel application of LLMs in equivalence checking, this research study aims to contribute significantly to advancing digital design verification methodologies.

4.0 Research Questions

The following are the research questions that can arise from this evaluation.

- Which are the most suitable LLMs that can be used for comparing and validating RTL designs against digital design specifications, considering the latest LLMs available in the market, such as GPT-4, Cohere Command, Falcon, LaMDA, LLaMA, Guanaco-65B, Vicuna 33B, Galactica, GPT-3, GPT-3.5, BLOOM, XLM-RoBERTa, XLNet, Orca, PaLM, Phi-1, and the new LLMs from Google Gemini?

- If LLMs manage to perform the required tasks, are separate DNNs/RL/RLHF required to be used for better accuracy/reliability?
- What is the optimal setup/fine-tuning for LLMs and PPM in the context of equivalence checks?
- Is there an automated methodology for fine-tuning LLMs and PPM in the context of equivalence checks?
- Which platforms are suitable for running equivalence checkers?
- Are there any new possibilities for LLMs to understand digital design specifications and RTL, such as simulating RTL, building test benches, or finding potential digital design bugs?
- Any other AI/ML components that must be added to achieve accurate/reliable equivalence checks?
- Is a digital design simulator like Synopsys VCS, Cadence Xcelium, or Siemens ModelSim required to be integrated with the context of equivalence checks? It would be best to work without a digital design simulator.
- Any preprocessing required for digital design specifications and RTL designs.
- To what extent can LLMs analyze digital design specifications and RTL designs?

5.0 Aim and Objectives

The following are the aims and objectives that can arise from this evaluation.

- The main purpose of this evaluation is to ensure the alignment between the architectural specifications and the RTL designs as part of the critical digital design verification process and to ensure the quality/robustness/optimization/accuracy of RTL designs.
- Evaluate the strengths and limitations of LLMs in analyzing digital design specifications and RTL designs.
- Implement an equivalent checker to align digital design specifications and RTL designs.

6.0 Scope of the Research

The scope of the research encompasses a comprehensive comparison of various aspects of the RTL designs, focusing on basic/intermediate/advance elements such as the following items.

- Basic
  - Hierarchy
  - Interface
  - Clock Domain
  - Reset Domain
  - State Machine
  - Data Path
  - Clock Path
  - Reset Path
- Intermediate
  - Combinational Logic
  - Clock Domain Crossing
  - Reset Domain Crossing
- Advance
  - Functional Behavior
  - Timing Behavior
  - Clock Behavior

- Reset Behavior
- Register Structure

The research aims to leverage LLMs to flag discrepancies between the digital design specifications and the RTL designs. This involves meticulously examining the design hierarchy and interface, focusing on the clock domain to ensure synchronization and minimize clock skew. Additionally, the research will investigate combinational logic, state machine, data path, clock path, and reset path to identify any deviations from the design specifications. The LLMs will play a pivotal role in facilitating the detection of inconsistencies, thereby enabling the architects, designers, and validators to collectively review and address any disparities between the digital design specifications and the RTL designs. By encompassing these critical elements and leveraging LLMs, the research aims to provide a robust framework for comprehensively validating digital designs at the RTL level.

7.0 Significance of the Research

In the rapidly evolving field of digital design, ensuring the alignment of digital design specifications with RTL designs is paramount. Traditional methods often suffer from miscommunications among architects, designers, and validators, leading to discrepancies between digital design specifications and RTL designs. This project proposes the use of LLMs as a tool to bridge this gap, ensuring that RTL designs are aligned with the digital design specifications.

7.1 Improving Accuracy and Efficiency

The application of LLMs in validating RTL against digital design specifications represents a significant advancement in digital design. By automating the comparison process, LLMs can detect redundant, irrelevant, unused, or invalid RTL codes that might be overlooked in manual design reviews. This enhances the accuracy of RTL designs and accelerates the digital design verification development cycle, a crucial factor in a highly competitive industry.

7.2 Reducing Miscommunication

Miscommunications in the digital design verification process can lead to costly errors. By establishing a 'golden point' of comparison through LLMs, all stakeholders, including the architects, designers, and validators, can refer to a consistent, unambiguous standard. This reduces the risk of misinterpretation and ensures that all parties are aligned in their understanding of the digital design specifications.

7.3 Cost-Effectiveness

In digital design verification, errors in RTL designs can be expensive to rectify, especially if detected late in the IC design flows. The proactive validation offered by LLMs can identify potential issues early, significantly reducing the costs associated with post-silicon debugging and fixes.

7.4 Contribution to Knowledge Base

This research contributes to the existing knowledge base by demonstrating the practical application of LLMs in a new domain. It opens avenues for further research into integrating AI and machine learning technologies in digital design verification processes.

7.5 Audience and Beneficiaries

The primary beneficiaries of this research are digital design verification professionals, including architects, designers, and validators. Academia and research institutions focusing on digital design verification and AI applications will also find this study valuable. Additionally,

industries reliant on digital design, such as semiconductors and consumer electronics, will benefit from the improved efficiency and accuracy in digital design verification flows.

8.0 Research Methodology

This research utilizes a structured methodology, encompassing block-level descriptions of key components, implementation/integration procedures, and operational/validation flow. The following section provides an in-depth exploration of each element, ensuring transparency and reproducibility of the research approach.

8.1.0 Block Level Description

The core components of this research framework are thoroughly outlined in the following section. This detailed block-level description delves into each key element, clearly understanding their functionalities and interactions within the system.

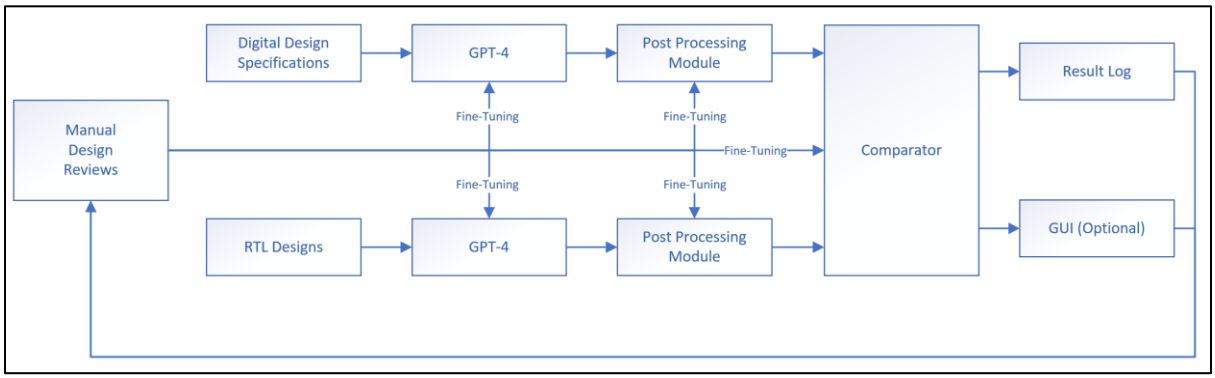
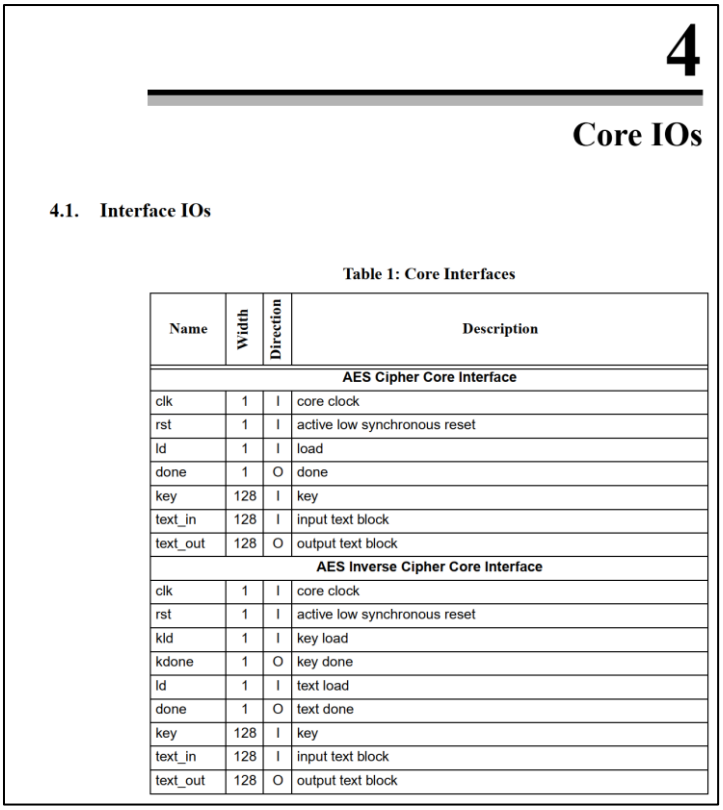


Figure 1 Top-level block diagram for equivalence checker for digital design specifications and RTL designs.

8.1.1 Digital Design Specifications

Digital design specifications are a set of documents that are commonly documented in a Word file and define the architecture and functionality of a digital system. They are typically written in natural language and include diagrams, tables, and other supporting information. The digital design specifications for the proposed system will be sourced from OpenCores (OpenCores, 2023), a free and open-source repository of digital designs.

The digital design specifications are a critical input to the proposed system. They provide GPT-4 with the necessary information to understand the design of the digital system. GPT-4 will use this information to extract features from the digital design specifications and RTL designs. These features will then compare the two and identify any inconsistencies.



4

Core IOs

4.1. Interface IOs

Table 1: Core Interfaces

Name	Width	Direction	Description
AES Cipher Core Interface			
clk	1	I	core clock
rst	1	I	active low synchronous reset
ld	1	I	load
done	1	O	done
key	128	I	key
text_in	128	I	input text block
text_out	128	O	output text block
AES Inverse Cipher Core Interface			
clk	1	I	core clock
rst	1	I	active low synchronous reset
kld	1	I	key load
kdone	1	O	key done
ld	1	I	text load
done	1	O	text done
key	128	I	key
text_in	128	I	input text block
text_out	128	O	output text block

Figure 2 Example of digital design specifications.

8.1.2 RTL Designs

RTL designs are a hardware description language (HDL) representation of a digital system. They are typically written in Verilog or VHDL. The RTL designs for the proposed system will be sourced from OpenCores (OpenCores, 2023).

The RTL designs are another critical input to the proposed system. They provide GPT-4 with the necessary information to understand the implementation of the digital system. GPT-4 will use this information to extract features from the RTL designs and compare them to those extracted from the digital design specifications. This comparison will help to identify any discrepancies between the two.

```
module aes_cipher_top(clk, rst, ld, done, key, text_in, text_out );
input       clk, rst;
input       ld;
output      done;
input   [127:0] key;
input   [127:0] text_in;
output  [127:0] text_out;

//////////////////////////////////////////////////////////////////
//
// Local Wires
//

wire    [31:0]  w0, w1, w2, w3;
reg [127:0] text_in_r;
reg [127:0] text_out;
reg [7:0]   sa00, sa01, sa02, sa03;
reg [7:0]   sa10, sa11, sa12, sa13;
reg [7:0]   sa20, sa21, sa22, sa23;
reg [7:0]   sa30, sa31, sa32, sa33;
wire    [7:0]   sa00_next, sa01_next, sa02_next, sa03_next;
wire    [7:0]   sa10_next, sa11_next, sa12_next, sa13_next;
wire    [7:0]   sa20_next, sa21_next, sa22_next, sa23_next;
wire    [7:0]   sa30_next, sa31_next, sa32_next, sa33_next;
wire    [7:0]   sa00_sub, sa01_sub, sa02_sub, sa03_sub;
wire    [7:0]   sa10_sub, sa11_sub, sa12_sub, sa13_sub;
wire    [7:0]   sa20_sub, sa21_sub, sa22_sub, sa23_sub;
wire    [7:0]   sa30_sub, sa31_sub, sa32_sub, sa33_sub;
wire    [7:0]   sa00_sr, sa01_sr, sa02_sr, sa03_sr;
wire    [7:0]   sa10_sr, sa11_sr, sa12_sr, sa13_sr;
wire    [7:0]   sa20_sr, sa21_sr, sa22_sr, sa23_sr;
wire    [7:0]   sa30_sr, sa31_sr, sa32_sr, sa33_sr;
wire    [7:0]   sa00_mc, sa01_mc, sa02_mc, sa03_mc;
wire    [7:0]   sa10_mc, sa11_mc, sa12_mc, sa13_mc;
wire    [7:0]   sa20_mc, sa21_mc, sa22_mc, sa23_mc;
wire    [7:0]   sa30_mc, sa31_mc, sa32_mc, sa33_mc;
reg     done, ld_r;
reg [3:0]   dcnt;

//////////////////////////////////////////////////////////////////
//
// Misc Logic
//

always @(posedge clk)
    if(!rst)    dcnt <= #1 4'h0;
    else
    if(ld)      dcnt <= #1 4'hb;
    else
    if(|dcnt)   dcnt <= #1 dcnt - 4'h1;

always @(posedge clk) done <= #1 !(|dcnt[3:1]) & dcnt[0] & !ld;
always @(posedge clk) if(ld) text_in_r <= #1 text_in;
always @(posedge clk) ld_r <= #1 ld;
```

Figure 3 Example of RTL designs in SystemVerilog code.

8.1.3 GPT-4

GPT-4 is a LLMs developed by OpenAI. It is a powerful tool for natural language processing tasks such as text generation, translation, and summarization. GPT-4 will be used in the proposed system to extract features from the digital design specifications and RTL designs.

GPT-4 is a well-suited tool for this task because it can understand the semantics of the digital design specifications and RTL designs. It can also identify relationships between the two. This ability is essential for extracting the features necessary to compare the digital design specifications and RTL designs.

8.1.4 Post-Processing Module (PPM)

The Post-Processing Module (PPM) bridges the gap between GPT-4 and the Comparator by extracting relevant features from GPT-4 processed features from digital design specifications and RTL designs. It performs four key tasks:

- Feature Identification: Identifies key design elements like modules, signals, ports, and logic operations.
- Feature Representation: Converts features into a structured format suitable for comparison.
- Data Cleaning and Standardization: Ensures consistency and accuracy of extracted data.
- Feature Aggregation and Association: Aggregates related features and establishes associations for a holistic design view.

By transforming raw GPT-4 output, the PPM prepares data for efficient and accurate comparison by the Comparator. The PPM can be phased out after multiple rounds of fine-tuning until GPT-4's output format becomes consistently compatible with the comparator, streamlining the process.

8.1.5 Comparator

The comparator is a Python module that compares the features extracted from the PPM for digital design specifications and RTL designs. It outputs a list of inaccuracies/discrepancies between the two. The comparator is another critical component of the proposed system. It identifies inaccuracies or discrepancies between the digital design specifications and RTL designs. The comparator is implemented using Python and a variety of machine-learning algorithms.

The comparator will work as follows:

- It will load the features extracted from the PPM's digital design specifications and RTL designs.
- It will then compare the two sets of features using a variety of machine learning algorithms.
- The comparator will list inaccuracies/discrepancies between the digital design specifications and RTL designs.

8.1.6 Result Log

The result log is a file that contains the list of inaccuracies/discrepancies output by the comparator. It also contains other information, such as the time and date of the comparison. The result log is an important output of the proposed system. It provides the user with a detailed list of inaccuracies or discrepancies between the digital design specifications and RTL designs. The user can use the result log to debug the design or make necessary changes.

8.1.7 Graphical User Interface (GUI)

The Graphical User Interface (GUI) displays the results of the comparison to the user. It is optional and can be developed using a Python library like Tkinter. The GUI is a convenient way for the user to view the comparison results. It can also interact with the system to filter the results or view more details about a particular inaccuracy/discrepancy.

8.1.8 Manual Design Reviews

The equivalence checker's effectiveness hinges on meticulous manual reviews by experienced architects, designers, and validators. These experts delve deep into the generated results, leveraging the detailed report as a roadmap to identify and address any discrepancies between the high-level digital design specifications and their concrete RTL implementation. Their keen eyes and insightful feedback form the cornerstone for iteratively refining the GPT-4 and the PPM, paving the way for increasingly accurate and reliable comparisons in future projects.

This comprehensive review process goes beyond simply verifying individual discrepancies. The reviewers analyze trends and patterns in the identified inconsistencies to uncover potential biases or limitations within the GPT-4. This insightful analysis informs targeted fine-tuning efforts, ensuring the GPT-4's understanding of design specifications and RTL code becomes increasingly nuanced and accurate.

Furthermore, the reviewers' expertise extends to evaluating the effectiveness of the post-processing module. They assess its ability to refine and structure the extracted features for optimal comparison by the GPT-4. By identifying areas where the post-processing module can be enhanced, the reviewers contribute to its ongoing development and refinement, leading to a more robust and reliable equivalence checker.

In essence, the manual review process serves as a critical feedback loop, constantly pushing the boundaries of the equivalence checker. Through the combined efforts of the GPT-4, the post-processing module, and the expert reviewers, the equivalence checker evolves into a powerful tool for ensuring the equivalence between digital design specifications and their RTL implementations, paving the way for a future where hardware development is both efficient and error-free.

8.2.0 Implementation/Integration

This section details the implementation and integration of the equivalence checker, focusing on the technical aspects and providing a comprehensive overview of the process.

8.2.1 Digital Design Specifications

The digital design specifications will be parsed and preprocessed to ensure compatibility with GPT-4's input format. Relevant information like architecture, functionality, interfaces, data flow, control flow, and timing constraints will be extracted using the GPT-4/PPM module. Additionally, specific keywords and technical terms related to the digital design domain will be identified and fed into GPT-4/PPM to enhance its understanding of the digital design specifications. From the implementation point of view, it is necessary to explore the need for parsing, preprocessing, and sequencer of digital design specifications before they are sent to GPT-4. There could be possibilities that parsing, preprocessing, and sequencer of digital design specifications can be done by GPT-4 API.

8.2.2 RTL Designs

Similar to the digital design specifications, relevant information in RTL designs will be extracted using the GPT-4/PPM module. Additionally, specific signal names, component names, and hardware-related keywords will be identified and utilized to improve GPT-4/PPM understanding of the RTL designs. From the implementation point of view, it is necessary to explore the need for parsing, preprocessing, and sequencer of RTL designs before they are sent to GPT-4. There could be possibilities that parsing, preprocessing, and sequencer of RTL designs can be done by GPT-4 API.

8.2.3 GPT-4 Integration

The GPT-4 API will be used to access the GPT-4 model and perform the required tasks:

- Extract features from both the digital design specifications and RTL designs.
- Generate natural language descriptions of identified discrepancies.
- Answer user queries about the design comparison process.

The specific API calls and parameters will be optimized for efficient information extraction and accurate results. To improve its performance and accuracy, fine-tuning GPT-4 on a dataset of similar digital design specifications and RTL designs might be necessary.

8.2.4 Post-Processing Module Development

Built on the foundation of Python libraries like pandas and scikit-learn, the post-processing module plays a vital role in refining and preparing the features extracted by GPT-4. Its responsibilities include:

- Data Cleaning: Removing noise and irrelevant information from the raw features extracted by GPT-4. Techniques like normalization and outlier detection are applied to ensure data quality.
- Feature Transformation: Transforming the features into a format best suited for the Comparator. This may involve dimensionality reduction techniques like PCA or t-SNE.
- Feature Selection: Carefully selecting a subset of the most informative features for comparison. This optimizes the accuracy and efficiency of the equivalence checking process.

8.2.5 Comparator Development

The Comparator, developed using Python libraries like difflib and fuzzywuzzy, takes center stage in identifying discrepancies between extracted features. Its key functionalities include:

- Feature Comparison: Employing similarity measures, distance metrics, and rule-based matching techniques to compare features extracted from the digital design specifications and the RTL designs.
- Discrepancy Detection: Using thresholds for similarity or outlier detection techniques to identify discrepancies between the compared features.
- Result Log Generation: Creating a comprehensive and informative result log detailing all identified discrepancies. This log includes details like the nature of the discrepancy, its severity, the specific locations within the digital design specifications document, and the RTL code where it occurs.

The result log should be meticulously structured for easy user review. Information should be organized in distinct columns for features, discrepancies, severity levels, and location details. The format should be flexible and customizable to accommodate diverse user preferences and project requirements.

8.2.6 Result Log Generation

The results of the comparison will be logged in a clear and concise format, including:

- List of identified discrepancies with descriptions and confidence scores.
- Specific locations in the digital design specifications and RTL designs where discrepancies occur.
- Timestamps and user information for traceability and version control.
- Links to relevant resources or documentation for further investigation.

8.2.7 GUI Development

A basic GUI can be developed using libraries like Tkinter to:

- Display the results of the comparison in an interactive and user-friendly way.
- Allow users to filter and search for specific discrepancies based on their interests.
- Provide access to additional information and documentation related to the design and discrepancies.

- Facilitate communication and collaboration among architects, designers, and validators.

8.3.0 Operational/Validation Flow

This section provides a high-level overview of the two key workflows within the system: the operational flow and the validation flow. The operational flow describes the continuous process of the equivalence checker, while the validation flow ensures the system's accuracy and robustness. Understanding both flows is crucial for appreciating the system's effectiveness and reliability. The complete details of these flows, including a comprehensive flowchart, are provided in a subsequent section for further exploration.

8.3.1 Design Upload

The user begins by uploading the digital design specifications (Word, PDF, text) and the corresponding RTL designs (zipped SystemVerilog files).

8.3.2 Send to GPT-4 for Analysis and Features Extraction

GPT-4 directly ingests the uploaded digital design specifications in a Word document and zipped RTL designs (SystemVerilog files) without requiring prior parsing. Utilizing its deep learning abilities, GPT-4 analyzes the design content and extracts crucial features that capture the intent and functionality of the digital system. These extracted features are then forwarded to the post-processing module for further refinement and preparation for comparison.

8.3.3 Post-Processing and Features Refinement

The extracted features are passed through a post-processing module, cleaned, transformed, and selected to ensure high-quality data for accurate comparison.

8.3.4 Features Comparison and Discrepancy Detection

A dedicated comparator module compares the refined features to identify discrepancies between the design specifications and the RTL implementation. To achieve this, it utilizes various similarity measures, distance metrics, and rule-based matching techniques.

8.3.5 Result Generation

The comparator generates a detailed result log that lists all identified discrepancies, including their nature, severity level, and specific location within the design documents.

8.3.5 Manual Design Review

This log is then reviewed by experts, such as architects, designers, and validators, to ensure its accuracy and completeness. They evaluate the severity of each discrepancy for prioritization. Their feedback helps refine the GPT-4 and PPM.

8.3.6 System Improvement and Fine-Tuning

Based on the review results and identified discrepancies, the system undergoes fine-tuning. This may involve adjusting GPT-4 parameters, optimizing the post-processing module, or refining the comparator's algorithms. The goal is to continuously improving the system's accuracy and robustness over time.

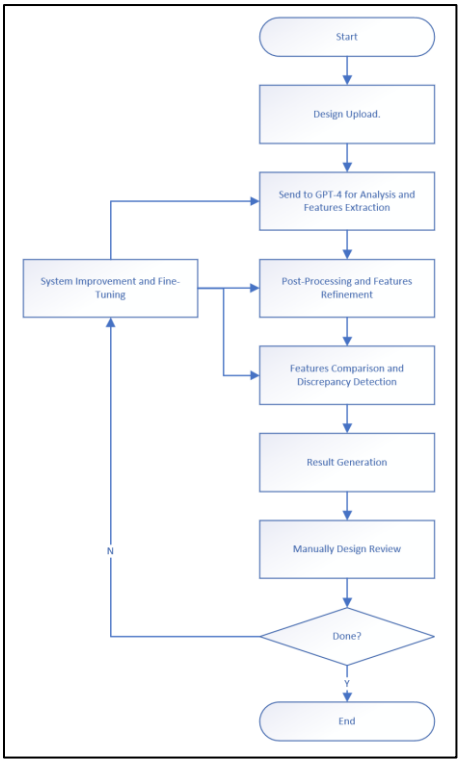


Figure 4 The flowchart of the equivalent checker for digital design specifications and RTL designs.

9.0 Research Plan

The research project will be conducted over 46 weeks, starting on January 1, 2024 (Week 1) and ending on November 17, 2024 (Week 46). The project plan categorizes tasks into stages: Planning, Setup, Development, Validation, Evaluation, and Project Report. Within each category, specific tasks are outlined with their corresponding start and end weeks, along with the duration in weeks.

9.1 Planning

Planning related tasks involves the development of a comprehensive research plan, taking 2 weeks (weeks 1-2).

9.2 Setup

Setup encompasses tasks like dataset preparation, data preprocessing, and environment setup for GPT-4, LLMs, PPM, and other tools, spanning 6 weeks (weeks 3-9), excluding week 7, when it is a public holiday for Chinese New Year.

9.3 Development

Development focuses on constructing key components like LLMs, PPM, Comparator, and Result Log Generation, each requiring 3 weeks (weeks 12-14, 16-18, 22-24, and 25-27, respectively).

9.4 Validation

Validation ensures the proper functioning of the developed components through continuous fine-tuning of LLMs, PPM, and Comparator, lasting 4 weeks (weeks 28-30 and week 33).

9.5 Evaluation

The evaluation assesses the effectiveness of the proposed methodology, involving methodology evaluation and evaluation analysis taking 4 weeks (weeks 34-37).

9.6 Project Report

The concluding phase of the project involves drafting the final report and preparing the presentation over three weeks (weeks 38-40). This phase leads to a series of presentations delivered to the supervisor in weeks 41, 43, and 45. Concurrently, from week 42 to week 45, the project report undergoes continuous revisions. The submission of the finalized project report is scheduled for week 46.

9.7 Public Holiday and Examination Leave

Examination Leave is scheduled for weeks 10-11, 20-21, and 31-32, while public holidays for Chinese New Year and Hari Raya Aidifitri fall on week 7 and week 15.

The time allocated for each task varies depending on its complexity and available resources. It falls within the following ranges: planning (2 weeks), setup (6 weeks), development (12 weeks), validation (4 weeks), evaluation (4 weeks), and project report (9 weeks). This comprehensive schedule provides a roadmap for the research project, allowing for efficient project management and timely completion. The project timeline Gantt charts are drafted as shown in the following figures.

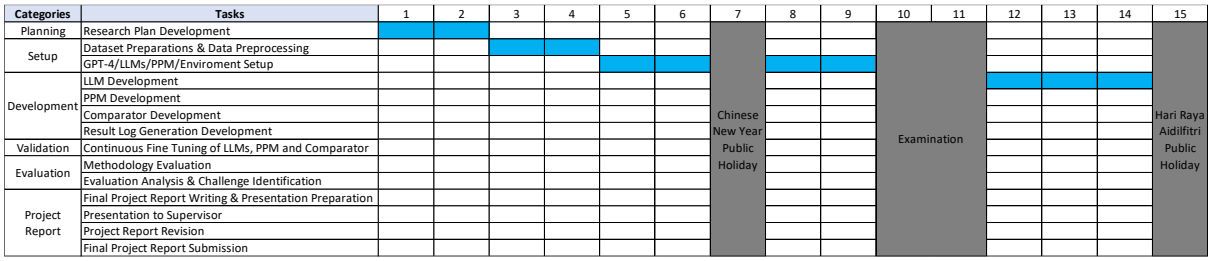


Figure 5 Project timeline Gantt chart from week number 1 to week number 15 in the year 2024.

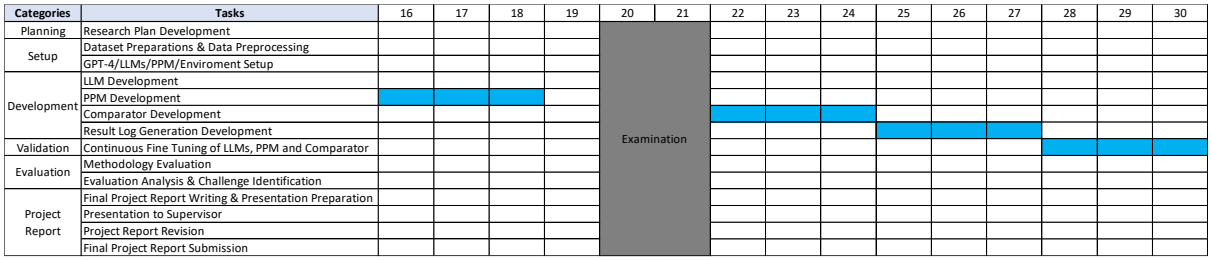


Figure 6 Project timeline Gantt chart from week number 16 to week number 30 in the year 2024.

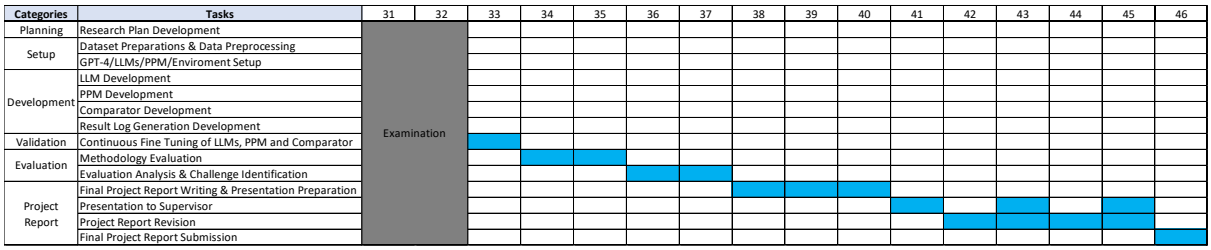


Figure 7 Project timeline Gantt chart from week number 31 to week number 46 in the year 2024.

10.0 Summary

The digital design process involves several key stages, starting with the definition of high-level architecture and specifications by architects. Subsequently, designers translate these specifications into RTL designs, ensuring the digital system meets the specified requirements. RTL serves as an intermediate step between defining the architecture and the actual digital

design flows, involving data transfer between registers within a digital system. However, discrepancies between the digital design specifications and RTL code often arise, including redundant, unused, or invalid code, compromising design quality and efficiency.

To address this challenge, using LLMs such as GPT-4 presents a promising opportunity for automated equivalence checking in digital design workflows. Leveraging LLMs can ensure that RTL designs align accurately with the specified requirements, thereby mitigating the occurrence of discrepancies and enhancing the overall digital design flows. The proposed methodology involves training GPT-4 on a comprehensive dataset of paired design specifications and corresponding RTL implementations to establish a reference point for identifying potential discrepancies during the design process. Subsequently, the trained LLMs will analyze new design specifications and compare them to existing RTL code, automatically highlighting any inconsistencies.

This approach holds significant promise for enhancing design quality by ensuring accurate implementation, eliminating unnecessary code, and improving communication between architects, designers, and validators. The research project aims to explore the potential of GPT-4 for equivalence checking and proposes a novel methodology that capitalizes on the model's ability to process and comprehend complex information. Furthermore, the research project will collect and preprocess a diverse dataset of paired digital design specifications and RTL designs to train GPT-4, enabling it to learn their relationships. An automated equivalence checker utilizing the trained LLMs will be developed and evaluated on representative designs to enhance design quality, reduce design time, improve communication, and streamline the design workflow.

In summary, utilizing LLMs, particularly GPT-4, for equivalence checking in digital design workflows presents a transformative opportunity to ensure higher quality and more efficient design processes, revolutionizing the verification of digital designs.

References

Hu, J., Hu, Y., Lv, Q., Wang, W., Wang, G., Chen, G., … & Yang, H. (2020). A path-based equivalence checking method between system level and RTL descriptions using machine learning. Journal of Circuits, Systems and Computers, 30(04), 2150074. https://doi.org/10.1142/s0218126621500742

Becker, A., Novo, D., & Ienne, P. (2014). Sketchilog: sketching combinational circuits.. https://doi.org/10.7873/date2014.165

Jutraz, A. and Zupancic, T. (2014). The role of architect in interdisciplinary collaborative design studios. Igra Ustvarjalnosti - Creativity Game, 2014, 034-042. https://doi.org/10.15292/iu-cg.2014.02.034-042

Nane, R., Sima, V., Pilato, C., Choi, J., Fort, B., Canis, A., … & Bertels, K. (2016). A survey and evaluation of fpga high-level synthesis tools. Ieee Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(10), 1591-1604. https://doi.org/10.1109/tcad.2015.2513673

Rodrigues, C., Morais, F., Silva, L., Silva, K., Figueiredo, J., Guerrero, D., … & Melcher, E. (2008). Functional verification methodology using hierarchical coloured petri nets-based testbenches.. https://doi.org/10.1109/icsmc.2008.4811600

Trost, A. and Zemva, A. (2012). Teaching design of video processing circuits. International Journal of Electrical Engineering Education, 49(2), 170-178. https://doi.org/10.7227/ijeee.49.2.7

Zemva, A., Trost, A., & Zajc, B. (1998). Educational programmable system for prototyping digital circuits. International Journal of Electrical Engineering Education, 35(3), 236-244. https://doi.org/10.1177/002072099803500306

Zergainoh, N., Tambour, L., & Jerraya, A. (2006). Automatic delay correction method for ip block-based design of vlsi dedicated digital signal processing systems: theoretical foundations and implementation. Ieee Transactions on Very Large Scale Integration (Vlsi) Systems, 14(4), 349-360. https://doi.org/10.1109/tvlsi.2006.874360

OpenCores. (n.d.). Home. Retrieved December 10, 2023, from https://opencores.org/