Design And Verification of An AMBA-APB Protocol Based System In SOC Using UVM With Python Regression Script

Mr. Gurubasava Shivanand Honnu[1], Dr. Bhagya P[2]
[1, 2] Associate Professor, Dept of Electronics & Communication Engineering
[1, 2] Don Bosco Institute of Technology, Bengaluru, Karnataka-560074.

Abstract- *The Project introduces the verification of the Advanced Microcontroller Bus Architecture (AMBA) which includes Advanced Peripheral Bus (APB) protocol in System on Chip (SOC) Design. As per we know that Verilog is outdated technology and System Verilog (SV) is come to existence, in Present SV architecture has many issues with efficiency and reusability and also taking huge time to Verify the Design Effectively. As a result, UVM support and promises all the requirements like high reusability, library files and reducing the verification time.*

The Designed AMBA-APB protocol will be verified using UVM architecture and verify with N number of valid test cases which create the same N number of log files which includes the UVM verbosity and severity, this helps the verification engineer about the statues of the design for manufacture. When the log files are huge that even takes lot of time to check, So the project also includes the feature of Python Regression Script.

The Python Regression Script utilize all the log files and takes a copy of all the important key results and make the excel data sheet and prepare the bar chat by its own, which helps for an effective analysis of the Design status, So the Verification time reduces drastically.

Keywords- Universal Verification Methodology (UVM). Design Under Test (DUT) , graphical user interface (GUI) , System on Chip (SOC). Advanced Peripheral Bus (APB) Advanced Microcontroller Bus Architecture (AMBA)

I. INTRODUCTION

A communication protocol is a set of rules and conventions that govern how data is transmitted and received between devices in a System on Chip (SOC). These protocols define the format, timing, sequencing, and error handling procedures necessary for successful data exchange. Communication protocols are essential for enabling devices from different manufacturers and with different functions to communicate and collaborate effectively.

Purpose of Communication Protocols: Communication protocols serve several essential purposes:

- **Data Exchange**: They enable devices to exchange data, whether it's text, images, audio, or any other form of information.
- **Error Handling:** Protocols include mechanisms for detecting and correcting errors that may occur during transmission.
- **Efficiency:** They optimize data transfer for speed, bandwidth utilization, and minimal overhead.
- **Interoperability:** Communication protocols allow devices from various vendors and technologies to communicate seamlessly.

In this project we are using AMBA-APB Protocol as a communication media between the devices. **Advanced Microcontroller Bus Architecture** (AMBA) is the belongs to ARM Developer. In this AMBA, we have various communication protocol used inside the chip and we are using **Advanced Peripheral Bus** (APB) as a communication protocol in this project. APB Design is needed to test or verify before chip manufacturing. So, we are using **Universal Verification Methodology** (UVM).

The UVM is the updated verification tool to verify this APB protocol DUT (**Design Under Test**). After verifying the DUT, we will get huge test case (DUT check results) and these test cases data is organized and have a key note in the Microsoft Excel file with Bar chat using Python Regression Script.

Regression analysis is a powerful statistical technique used to understand and model the relationship between a dependent variable (often referred to as the "target" or "outcome") and one or more independent variables (often referred to as "predictors" or "features"). It is a fundamental tool in data science and machine learning tasks such as prediction, forecasting, and understanding the impact of variables on an outcome.

Python, as a versatile and popular programming language, offers a wide range of libraries and tools that make it a natural choice for performing regression analysis. With libraries like xlsxwriter, openpyxl, orderedDict, NumPy and pandas, you can easily implement various regression models, visualize results, and draw valuable insights from the data. Openpyxl support to generate the Bar chat from the specific assigned data.

- **AMBA-APB Communication Protocol**

The (APB) Advanced Peripheral Bus forms an integral part of the Advanced Microcontroller Bus Architecture (AMBA) family and serves as a cost-effective interface that has been optimized for low power consumption and less complexity in interface. The APB is designed for connect with peripherals that possess low bandwidth requirements and less demand the high-performance characteristic of a pipelined bus interface.

Notably, the APB follows an unpipelined protocol, meaning that all signal transitions are synchronized with the clock rising edge. This design choice facilitates the seamless integration of APB peripherals into various design workflows. It's important to note that each data transfer within the APB framework consumes at least two clock cycles. The below Fig 1 shows the state diagram of AMBA-APB protocol. The complete way of designing the DUT is based on this sate diagram.

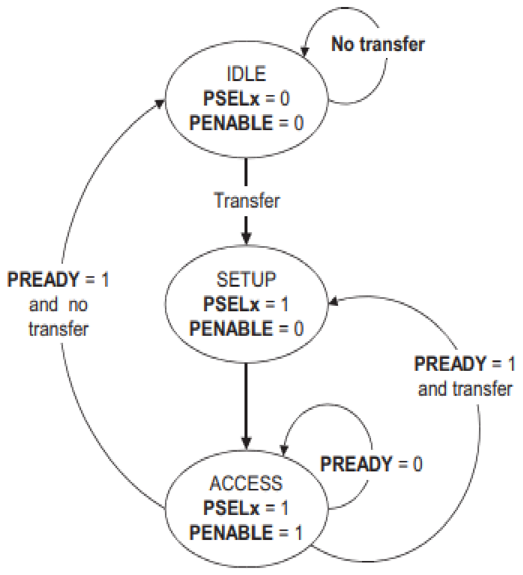


Fig. 1 *State diagram of AMBA-APB Protocol*

The finite state machine within the system functions according to the following states:

1. **IDLE:** This represents the default and initial state of the APB.
2. **SETUP:** When a data transfer is needed, the bus transitions to the SETUP state, during which the relevant select signal, denoted as PSELx, is activated. It's noteworthy that the bus stays in the SETUP state for just one clock cycle, invariably advancing to the ACCESS state on the subsequent rising edge of the clock.
3. **ACCESS:** In the ACCESS state, the enable signal, known as PENABLE, is activated. During the transition from the SETUP state to the ACCESS state, it is imperative that the address, write, select, and write data signals remain steady and unchanged. Exiting the ACCESS state is governed by the PREADY signal received from the slave device:

- If the slave device holds PREADY at a LOW level, the peripheral bus will persist in the ACCESS state.
- Conversely, if the slave device drives PREADY to a HIGH state, the ACCESS state concludes, and the bus returns to the IDLE state if no further transfers are necessary. Alternatively, if another transfer is required, the bus directly transitions to the SETUP state.

II. PROBLEM FORMULATION AND OBJECTIVE

The Updated technology has lot of efficiency, reusability and less Processing time which is very important in competitive world. UVM is the helps for Digital verification and Python regression script helps to check the status of lakhs of data in few seconds.

The main goal of this project is to reduce the execution time or processing time of a VLSI Digital Circuit verification using UVM and Python Regression Script.

III. PROPOSED METHODOLOGY

The basic idea behind the new approach is to use a updated architecture that is UVM (Universal Verification Methodology) and Python Based Code for Excel sheet and Bar chat generation

UVM Testbench Architecture: Testbench or Verification Environment is used to check the functional correctness of the **D**esign **U**nder **T**est (**DUT**) by generating and driving a predefined input sequence to a design, capturing the design output and comparing with-respect-to expected output.

Verification environment is a group of class’s performing specific operation. i.e, generating stimulus, driving, monitoring, etc. and those classes will be named based on the operation. The below Fig 2 shows the UVM testbench architecture which helps the design verification to write the code according to the architecture.

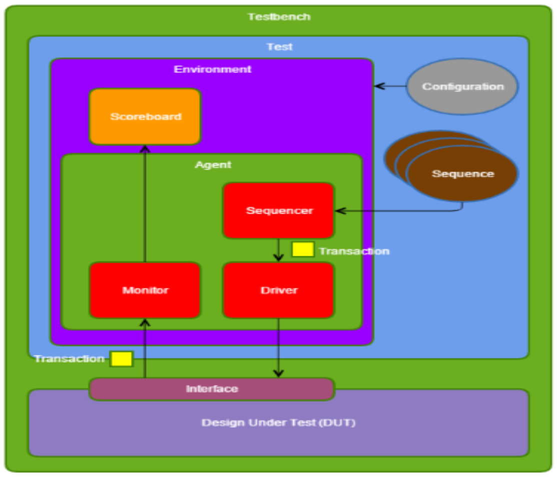


Fig. 2*UVM Testbench Architecture*

The Universal Verification Methodology (UVM) is a widely adopted standard for creating robust and reusable testbenches in the field of semiconductor design verification. UVM provides a structured and standardized approach to creating testbenches, making it easier to design, simulate, and verify complex digital hardware designs. UVM testbench architecture typically follows a hierarchical and modular structure. Here's an overview of the key components and layers in a UVM testbench architecture:

- **Testbench Hierarchy:**

Testbench: The top-level component that instantiates and manages all other components in the testbench.
DUT (Device Under Test): The hardware design being verified. The DUT is connected to the testbench through interfaces or ports.

- **Test Sequences:**

Test Sequence: Represents a specific verification scenario or test case. It defines a sequence of transactions to be applied to the DUT.
Sequencer: Manages the scheduling and execution of test sequences. Multiple sequencers may be used for different types of sequences.
Sequence Item: Represents a single transaction or data item to be transferred to or from the DUT.

- **Drivers and Monitors:**

Driver: Sends transactions generated by the sequencer to the DUT's input ports or interfaces.
Monitor: Observes transactions on the DUT's output ports or interfaces and converts them into transaction objects for analysis.

- **Interfaces and Agents:**

Interface: Represents a specific communication protocol or bus interface between the testbench and the DUT. It defines signal and data lines.
Agent: Combines a driver and a monitor for a specific interface. Multiple agents may be used to interface with different parts of the DUT.

- **Scoreboard and Coverage:**

Scoreboard: Compares the expected results (golden reference model) with the actual results produced by the DUT. It checks the correctness of the DUT's behavior.
Coverage Collector: Collects coverage data to ensure that different parts of the design have been adequately exercised by the test cases.

- **Configuration and Control:**

Test Configuration: Specifies the parameters and settings for a test case, such as clock frequencies, test duration, and test conditions.
Test Control: Coordinates the execution of different test cases and manages the overall testbench control flow.

- **Functional Coverage:**

Coverage Models: Define coverage bins and points of interest in the design. These helps track which parts of the design have been tested.
Coverage Goals: Define specific coverage objectives that must be met to consider a test case complete.

- **Assertions and Checkers:**

Assertions: Specify design constraints or requirements that must hold true during simulation. Assertions are used to catch design bugs.
Checkers: Monitor and verify that assertions are satisfied during simulation.

- **Logging and Reporting:**

Logging: Records simulation activity, including important events, messages, and warnings for debugging and analysis.
Reporting: Generates test results, coverage reports, and other verification metrics.

- **Testbenches Components:**

These are the building blocks of the testbench and can be customized and extended to suit the specific needs of the verification project.

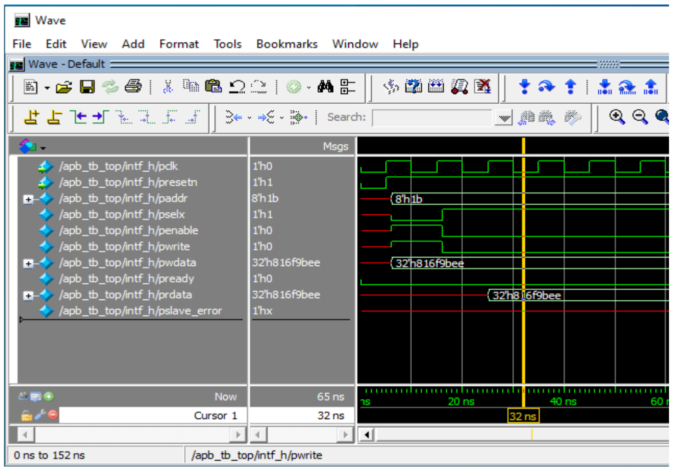
UVM encourages the use of object-oriented programming principles, such as inheritance and polymorphism, to create reusable and modular testbench components. By structuring the verification environment in this way, engineers can more effectively verify complex hardware designs and achieve better reusability and maintainability of their testbenches.

IV. RESULTS

In this section we will discuss the implementation results of our proposed methodology. Here we will explain each intermediate output of our proposed methodology, and also the final results from verification tool and results of python regression script.

➢ AMBA-APB Write and Read Transfer

APB write and read transfer test case results are shown in Fig 4.3. In waveform we can see the random sequence_item values of data and address [paddr = 1b, pwrite = 816f9bee] this write operation happens only when pwrite = 1. After that read operation of same memory happen and DUT stored value of address “1b” is “816f9bee” will sent to monitor to check whether the received data is right or wrong through scoreboard.



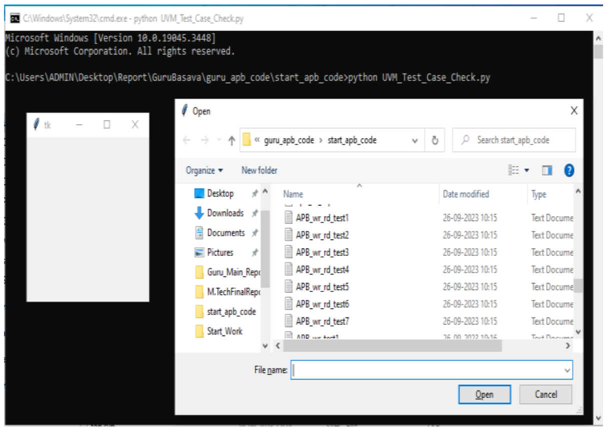
```
/apb_tb_top/intf_h/pclk            1'h0
/apb_tb_top/intf_h/presetn         1'h1
/apb_tb_top/intf_h/paddr           8'h1b
/apb_tb_top/intf_h/pselx           1'h1
/apb_tb_top/intf_h/penable         1'h0
/apb_tb_top/intf_h/pwrite          1'h0
/apb_tb_top/intf_h/pwdata          32'h816f9bee
/apb_tb_top/intf_h/pready          1'h0
/apb_tb_top/intf_h/prdata          32'h816f9bee
/apb_tb_top/intf_h/pslave_error    1'hx
                           Now     65 ns
                      Cursor 1     32 ns
0 ns to 152 ns    /apb_tb_top/intf_h/pwrite
```

Fig. 3*Simulation output of Write and Read Transfer.*

➢ Python Regression Script

We import the necessary libraries which includes **re, os, OrderedDict, xlsxwriter, openpyxl** and **filedialog** for File open, compare each line, generate Excel file and generate Bar chart using data. The Fig 4 shows the List of Library files Python 3.11 (64-bit) software do follow works with respect to code execution: -

- Firstly, python Code Open the File Dialog when it starts executing, which is shown in Fig 4.
- User need to select the UVM log files. The user has full freedom to select any number of files at once and even in any file folder.
- Python code automatically read each file one by one and start checking the error status of each file.
- In each file keys data is collected and make a note on the excel sheet automatically and Microsoft excel sheet has huge data base of all selected files, seed values, error value and fatal values
- Another key role of the python code is to check the error status of each file in data base and decide the Pass and Fail of each file
- The code itself take the excel data and prepare the bar chart which helps to analysis hundreds of test results at ones which is shown in Fig 5.

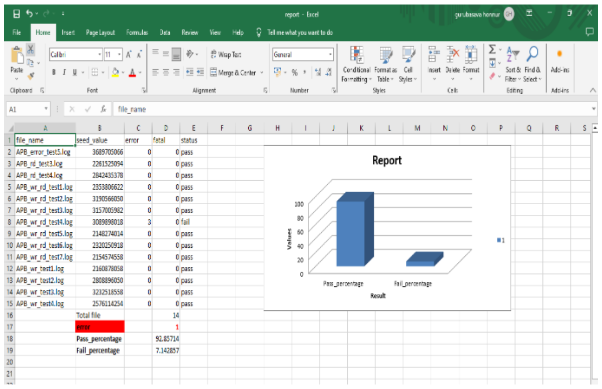


```
C:\Windows\System32\cmd.exe - python UVM_Test_Case_Check.py
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN\Desktop\Report\GuruBasava\guru_apb_code\start_apb_code>python UVM_Test_Case_Check.py
```

Name	Date modified	Type
APB_wr_rd_test1	26-09-2023 10:15	Text Docume
APB_wr_rd_test2	26-09-2023 10:15	Text Docume
APB_wr_rd_test3	26-09-2023 10:15	Text Docume
APB_wr_rd_test4	26-09-2023 10:15	Text Docume
APB_wr_rd_test5	26-09-2023 10:15	Text Docume
APB_wr_rd_test6	26-09-2023 10:15	Text Docume
APB_wr_rd_test7	26-09-2023 10:15	Text Docume

Fig. 4 *Command Prompt, File Dialog and Files in folder*



	A	B	C	D	E
1	file_name	seed_value	error	fatal	status
2	APB_error_test5.log	3689705066	0	0	pass
3	APB_rd_test3.log	2261525094	0	0	pass
4	APB_rd_test4.log	2842435378	0	0	pass
5	APB_wr_rd_test1.log	2353806622	0	0	pass
6	APB_wr_rd_test2.log	3190566050	0	0	pass
7	APB_wr_rd_test3.log	3157005982	0	0	pass
8	APB_wr_rd_test4.log	3089898018	3	0	fail
9	APB_wr_rd_test5.log	2148274014	0	0	pass
10	APB_wr_rd_test6.log	2320250918	0	0	pass
11	APB_wr_rd_test7.log	2154574558	0	0	pass
12	APB_wr_test1.log	2160878058	0	0	pass
13	APB_wr_test2.log	2808896050	0	0	pass
14	APB_wr_test3.log	3232518558	0	0	pass
15	APB_wr_test4.log	2576114254	0	0	pass
16		Total file		14	
17		error		1	
18		Pass_percentage		92.85714	
19		Fall_percentage		7.142857	

Fig. 5*Python Final Report with Bar Chart*

The above Fig 5 has complete information of 15 AMBA-APB log files. The Figure also shows the Pass percentage and Fail percentage of the 15 log files.

Applications: The Universal Verification Methodology is a standard verification methodology used in the semiconductor industry to verify the complex digital designs. In this Project there are mainly two different way of applications - one is Chip manufacturing and other is Data Processing and analysis.

- In manufacture of APB based SOC chip.
- For interface with other Protocol like AHB, AXI and even PCIe Protocol.
- UVM and even System Verilog log file analysis using script.
- Helpful for other Protocol framework design.

V. CONCLUSION AND FUTURE SCOPE

The proposed UVM is a robust methodology for verifying complex digital designs, and it can be effectively used to verify the correct operation of the APB protocol in a larger system context. UVM's structured approach and support for reusable components make it a valuable tool for ensuring the correctness and reliability of designs that incorporate the APB protocol.

Python provides a versatile and powerful environment for performing regression analysis on various types of data. Depending on the nature of the problem and the data, you can choose from a range of regression techniques and libraries to build accurate models and draw meaningful insights from your data. Proper data preprocessing, model selection, and interpretation of results are key to successful regression analysis in Python.

Finally, This Project helpful for future updated version of APB protocol and even other protocol like AHB Protocol (Advanced High-performance Bus), AXI Protocol (Advanced eXtensible Interface) and other AMBA protocols family.

VI. ACKNOWLEDGEMENT

I take this opportunity to convey my gratitude to all those who have been kind enough to offer their advice and provide assistance when needed which has led to the successful completion of the Project work.

I consider my cardinal duty to express the deepest sense of gratitude to my project guide **Dr. BHAGYA P**, Associate Professor, Department of Electronics and Communication Engineering, DBIT, for his skillful guidance, constant supervision, timely suggestion and constructive criticism in successful completion of my project in time.

I would also thank my parents for all support and encouragement and blessings they gave me all the way that kept me going, without which nothing would have been possible. I also thank all the other faculty members of E&C Department of DBIT and my friends for their help and support

REFERENCES

[1] **Research and Implementation of an automatic simulation tool** - By Liu Tang, You Li, Hongwei Wang, Huan Zhao, Luze Ma, Yuming Sun from Beijing Sunwise Information Technology Ltd, China Published Paper on 2020 7Th international Conference on Dependable Systems and Their application (DSA).

[2] **Development of a Generic and a Reconfigurable UVM-Based Verification Environment for SoC Buses** - By Alaa Hussien, Samar Mohamed, Mohamed Soliman, Hager Mostafa, Khaled Salah, Mohamed Dessouky, Hassan Mostafa Department of electronics and communications, faculty of engineering, Ain Shams University, Cairo, Egypt. Mentor Graphics, Cairo, Egypt. Electronics and Communications Engineering Department, Cairo University, Giza.

[3] **Design of Generic Verification Procedure for IIC Protocol in UVM** – By Jaideep Varier EV, Prabakar.V, Karthigha Balamurugan from Department of Electronics and Communication Engineering Amrita School of Engineering, Coimbatore Amrita Vishwa Vidyapeetham, India

[4] **Extraction of information from log files** – By Filipe Rigueira, Jorge Bernardino, Isabel Pedrosa from Coimbra Business School | ISCAC, Portugal

[5] **Predicting Student's Final Graduation CGPA Using Data Mining and Regression Methods**: A Case Study of Kano Informatics Institute – by Salim Jibrin Danbatta and Asaf Varol