

# VERIFICATION TEST PLAN

Fundamentals of Pre-Silicon Validation Spring-2024

Implementation and Verification of UART using both Class-based and UVM methodologies

Date:04-24-2024

## Contents

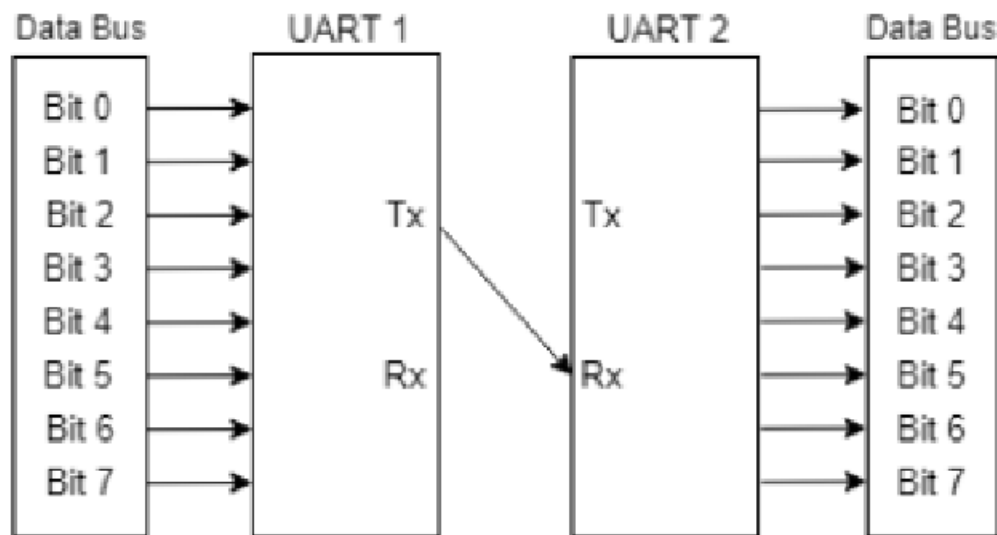
1.Introduction.....	2
2.Verification Requirements .....	3
3.Required Tools .....	4
4.Risks and Dependencies .....	4
5.Functions to be verified .....	5
6.Tests and Methods .....	5
7.Resources Requirements .....	6
8.Schedule .....	6
9.References.....	7

## **1.Introduction**

### **1.1 Objective of the Verification Plan**

This document establishes the verification plan for the UART design specified in the requirement specification. It identifies the features to be tested, the test cases, the expected responses, and the methods of test case application and verification.

### **1.2 Block diagram**

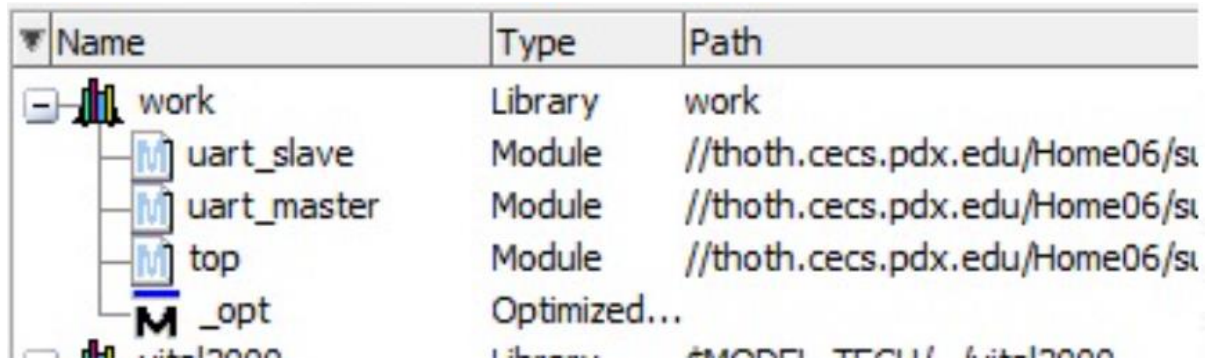


### **1.3 Specifications**

- Programmable baud rate: 1200, 2400, 4800, 9600, and 115200 bps.
- Implementation of parity bit for error.
- Stop bits considered are one.

## 2.Verification Requirements

### 2.1 Module Hierarchy



Name	Type	Path
work	Library	work
uart_slave	Module	//thoth.cecs.pdx.edu/Home06/si
uart_master	Module	//thoth.cecs.pdx.edu/Home06/si
top	Module	//thoth.cecs.pdx.edu/Home06/si
_opt	Optimized...	

Each module and its specifications

Transmitter-Specification:

- clk\_tx: transmitter clock in the uart with 50Mhz frequency.
- data: input data consists of 8 bits of data lines ,1 bit of start signal,1 bit of stop.
- en\_tx: when enabled, transmission of data to transmitter, 1 bit signal.
- u\_tx: output data to be transmitted by the transmitter, it consists of input data streamed and it also consists of 8 bits of input data.
- u\_tx\_done: flag to confirm the data is transmitted by the transmitter. Its one-bit signal

Receiver-Specification:

- clk\_rx: receiver clock in the uart with 50Mhz frequency
- data: output data from receiver consists of 8 bits of data lines,1 bit of start signal,1 bit of stop.
- en\_rx: enable receiving of data to receiver,1-bit signal
- u\_rx: input data to receiver from the transmitter,it consists of output data streamed and it also consists of 8 bits of output data.
- u\_rx\_done: flag to confirm the data is received completely. It's one-bit signal

### **3.Required Tools**

#### **3.1 Software and Hardware tools**

Tool used for design and Verification: Questa Sim

Revision control Tool: GITHUB

([https://github.com/ismavayrus/team\\_07\\_UART/tree/team\\_07\\_UART/M1](https://github.com/ismavayrus/team_07_UART/tree/team_07_UART/M1))

#### **3.2 Directory structure of your runs**

The directory structure for the verification runs will be organized for clarity and ease of access. Below is a proposed directory structure:

Verification\_plan

/testbench

uart\_tb.sv

/modules

uart\_master.sv

uart\_slave.sv

### **4. Risks and Dependencies**

There are several risks and dependencies that can impact the project. Here are some potential risks and dependencies to consider:

#### **a)Technical Risks:**

- Complexity of UART Design: The UART design might be more complex than anticipated, leading to difficulties in implementation and verification.
- Baud Rate Accuracy: Achieving accurate baud rates across different configurations might pose a challenge, especially in FPGA implementations.
- Parity Bit Handling: Implementing parity bit functionality could introduce additional complexity and potential bugs.

#### **b) Schedule Risks:**

- Design Iterations: Iterative design changes and debugging might extend the project timeline, delaying milestone completions.
- Verification Time: Comprehensive verification may take longer than estimated, potentially delaying project milestones.

### **c)Dependencies:**

- Tool Dependencies:Dependency on specific EDA tools for synthesis, simulation, and verification may introduce compatibility issues and workflow constraints.
- External Interfaces: Dependencies on external components or interfaces, such as clock sources or test equipment will impact design and verification activities.

### **d)Quality and Compliance Risks:**

- Functional Correctness: Ensuring the design meets all functional requirements, including baud rate accuracy, error handling, and compliance with UART standards.
- Compliance Verification: Verifying compliance with industry standards and protocols,

## **5.Functions to be Verified**

Several functionalities of the UART design can be tested using the provided testbench. Here are some key functionalities that can be verified:

- The test includes to check the data is transmitted from the 1<sup>st</sup> UART and is received by the second UART.
- To check input data is streamed by the transmitter.
- To check the transmitted data is streamed by the receiver.

## **6.Tests and Methods**

### **6.1 Testing methods to be used: Black/White/Gray Box.**

**Black Box Testing:** Functional verification based on specifications.

**White Box Testing:** Code coverage analysis and assertion-based verification.

**Gray Box Testing:** Scenario-based testing for corner cases.

### **6.2 Checking methodology**

a) **Self-Checking Testbench:** Automates the process of stimulus generation and response verification without manual intervention, using a testbench that compares the DUT's outputs against expected results.

b) **Assertion-Based Checking:** Employs conditional checks within the simulation to ensure the DUT's behavior aligns with specified design properties and constraints.

c) **Score boarding:** Utilizes a reference model to predict outputs, facilitating comparison against actual DUT responses for validation of correct behavior.

d) **Coverage Analysis:** Measures the extent to which the DUT's state space is exercised by the testbench, aiming to identify untested parts of the design.

e) **Randomized Testing:** Generates random inputs to stress-test the DUT across a broad range of scenarios, often uncovering edge cases not considered in directed testing.

f) **Directed Testing:** Focuses on specific and critical DUT functionalities by crafting targeted test cases, ensuring that behaviors are correctly implemented.

g) **Simulation Control:** Manages the execution flow of the simulation environment, such as clock cycles and reset sequences, to emulate realistic operating conditions.

h) **Logging and Error Reporting:** Captures and outputs simulation data and error messages to track test progress and facilitate debugging of failed checks.

## **7.Resources Requirements**

<b>No</b>	<b>Task</b>	<b>Actual Start Date</b>	<b>Responsible person</b>	<b>Target date &amp; Status</b>	<b>Completion</b>	<b>Comments</b>
1	<b>High Level Design Specification (HLDS)</b>	12/4/2024	PRASANNA KUMAR	16/4/24	Completed	Understanding and finding Design specifications documentation
1.1	<b>Design spec calculation and Implementation</b>	17/4/2024	SURYA VAMSI	19/4/24	Completed	Comprehending modules and devising a plan for their design, followed by the division of modules.
1.2	<b>Transmitter &amp; Receiver Design</b>	20/4/4	SUGGU SAI TAGORE	23/4/24	Completed	Crafted the code for both the transmitter and receiver UART modules ensuring functionality for both sending and receiving data via UART
1.3	<b>Baud Rate Calculation, Test Bench Plan</b>	21/4/24	SUSRITHA	23/4/24	Completed	Calculation for required Baud Rate along with planning

						testing Scenarios, for Testbench.
--	--	--	--	--	--	-----------------------------------

## **8.Schedule**

- Week 1-2 (until 04-24-2024): High-level design Specification, testing strategy, and drafting verification plan. Designing the basic UART.
- Week 3-4 (until milestone 2): Converting this basic testbench to a class-based testbench. Verifying with a randomized burst of data, updating the verification plan accordingly
- Week 5-6 (until milestone 3): Complete class-based verification to the updated RTL if any and maximize the code coverage and functional coverage. Inclusion of specific corner cases in the testbench
- Week 7-8 (until milestone 4): Complete implementation of UVM architecture and creating log files.
- Week 9-10 (until milestone 5): Finalize documentation, prepare all the deliverables, and submit.

## **9.References**

1. Open AI, "Chat GPT," [Online].
2. Verification academy for detailed coverage requirements of UART (<https://verificationacademy.com/cookbook/coverage/uart-example-test-plan/> )
3. Online learning video System Verilog for Verification Part2: Projects from Udemy ( <https://www.udemy.com/course/systemverilog-for-verification-part-2-projects/> )
4. Spear, Chris, Tumbush, Greg, "SystemVerilog for Verification" Book.
5. Yamini R, Ramya M V, "Design And Verification Of Uart Using System verilog ", IJEAT,2020.
6. Online learning website, chip verify