*Innovation & Entrepreneurship Hub for Educated Rural Youth (SURE Trust – IERY)*

# *VENDING MACHINE CONTROLLER*

**The domain of the Project: RTL Design Verification**

**COURSE NAME: Integrated VLSI**

## Team Mentor:

**Mr. Satish Devarapalli**

**(Emulation Verification Engineer, Apple)**

## Team Members:

1. Ms.Thippi Reddy Sasikala ------- B.Tech, 4 th year pursuing -------------- Team Leader
2. Mr.Veldi Rahul --------------------- B.Tech, 4 th year pursuing ------------ Team Member
3. Ms.Bathala Anusha --------------- B.Tech Graduate ----------------------- Team Member

## Period of the project

**October 2025 to November 2025**

# Declaration:

The project titled “**VENDING MACHINE CONTROLLER” Verification using UVM** has been mentored by **Satish Sir**, organised by SURE Trust, from November 2025 to December 2025, for the benefit of the educated unemployed rural youth for gaining hands-on experience in working on industry relevant projects that would take them closer to the prospective employer. I declare that to the best of my knowledge the members of the team mentioned below, have worked on it successfully and enhanced their practical knowledge in the domain.

| **Team Members**: | **Signatures:** |
|---|---|
| 1. Ms.T.Sasikala | T.Sasikala |
| 2. Mr.V.Rahul | V. Rahul |
| 3. Ms.B.Anusha | B. Anusha |

**Mentor’s Name:**
Mr. Satish Devarapalli
Emulation Verification Engineer , Apple

Prof. Radhakumari
Executive Director & Founder
SURE Trust

## Table of Contents

## *Executive Summary*

This project presents the **functional verification** of a **parameterized Vending Machine Controller IP** using the **Universal Verification Methodology (UVM)**. The primary objective was to validate that the design strictly adheres to the given **functional specification** across all operating modes, namely **Reset Mode**, **Configuration Mode**, and **Operation Mode**. Special emphasis was placed on verifying critical functionalities such as **APB-based configuration**, **item selection**, **multi-item transactions**, **currency insertion**, **dispense decision logic**, **change calculation**, and **exception handling** for conditions like **insufficient balance**, **out-of-stock items**, and **invalid transactions**, all within the specified **latency constraints**.

A **modular and reusable UVM testbench architecture** was developed, consisting of key verification components including **UVM sequences**, **driver**, **monitor**, **scoreboard**, **reference model**, and **SystemVerilog assertions**. Both **directed tests** and **constrained-random stimulus** were employed to achieve comprehensive coverage of functional scenarios. **Functional coverage models** were defined to track item indices, currency values, item quantities, and configuration parameters, ensuring validation of all supported combinations and corner cases. The verification environment was designed to support **parameter scalability**, allowing validation for varying numbers of items and currency ranges as defined by the specification.

The verification results demonstrated that the **Design Under Test (DUT)** consistently produced correct **dispense outputs**, **item counts**, and **currency change values**, while maintaining proper synchronization across **multiple clock domains** (system clock and configuration clock). All identified mismatches between the DUT and the **golden reference model** were debugged and resolved through systematic **waveform analysis** and **scoreboard comparison**. The project successfully achieved high **functional coverage closure**, confirming the robustness and correctness of the design implementation.

Based on the outcomes, it is recommended to enhance the verification framework further by incorporating **stress testing**, **negative testing**, and **formal assertions** for protocol compliance. Additionally, integrating **low-power checks**, **performance monitoring**, and **regression automation** would improve verification completeness and make the environment suitable for **industrial-scale VLSI verification workflows**.

# *Introduction*

## ● Background and Context of the project:

With the increasing complexity of **digital system-on-chip (SoC) designs**, ensuring functional correctness before fabrication has become a critical requirement in the **VLSI design flow**. Modern embedded controllers, such as those used in **automated vending machines**, must reliably handle multiple inputs, configuration parameters, and operational scenarios while meeting strict **performance and latency constraints**. Any functional error in such systems can lead to incorrect transactions, revenue loss, or poor user experience, making **pre-silicon functional verification** an essential phase of development.

The **Vending Machine Controller IP** addressed in this project is a **parameterized and configurable digital controller** designed to support a large number of items, variable pricing, multi-item transactions, and multiple operating modes including **Reset**, **Configuration**, and **Operation** modes. The design interacts with **asynchronous currency inputs**, a **slow human-driven interface**, and a **high-speed system clock**, introducing challenges such as **clock-domain interactions**, **transaction sequencing**, and **state management**. Additionally, configuration of item values and availability is performed through an **APB (Advanced Peripheral Bus) interface**, which is widely used in industry-standard SoC designs.

Given these complexities, a **structured and scalable verification methodology** is required to validate the design thoroughly. **Universal Verification Methodology (UVM)**, built on **SystemVerilog**, provides a standardized, reusable, and coverage-driven approach to verify complex digital IPs. UVM enables separation of testbench and design logic, promotes **reusability**, and supports **constrained-random stimulus**, **functional coverage**, and **self-checking mechanisms**, making it well suited for industrial verification environments.

In this context, the project was undertaken to apply **UVM-based functional verification techniques** to the Vending Machine Controller IP, with the goal of ensuring specification compliance, identifying corner-case failures, and gaining practical exposure to **industry-relevant verification practices**. The project bridges the gap between **theoretical verification concepts** and **real-world IP validation**, providing hands-on experience in developing reusable testbenches, debugging functional mismatches, and achieving coverage closure for a realistic digital controller.

- **Problem statement or goals of the project**

  The increasing functionality and configurability of modern digital controller IPs demand a rigorous and systematic functional verification process to ensure correctness across all operating conditions. The Vending Machine Controller IP under consideration supports parameterized item counts, multiple currency values, multi-item transactions, and operates across different modes such as Reset, Configuration, and Operation. Additionally, the design must correctly handle asynchronous currency inputs, APB-based configuration transactions, and latency constraints for dispense decisions. Verifying such a design using ad-hoc or directed testing alone is insufficient, as it may fail to uncover corner-case bugs, race conditions, and state-transition errors. Therefore, there is a need for a robust, reusable, and coverage-driven verification framework that ensures the design complies fully with the given specification.

**The primary goals of this project are:**

- To develop a UVM-based functional verification environment for the Vending Machine Controller IP using SystemVerilog and UVM.
- To verify correct behavior across all operational modes, including reset initialization, configuration via APB, and normal operation transactions.
- To validate critical functionalities such as item selection, currency accumulation, dispense logic, change calculation, and error handling for out-of-stock and insufficient balance conditions.
- To implement a self-checking testbench using a reference model and scoreboard for automatic result comparison.
- To achieve high functional coverage, ensuring that all supported items, currency values, quantities, and configuration combinations are exercised.
- To identify, debug, and resolve functional mismatches through waveform analysis and systematic verification techniques.
- To gain hands-on experience with industry-standard verification methodology and best practices used in VLSI design verification roles.

- **Scope of the Project**

The scope of this project is focused on the RTL-level functional verification of a Vending Machine Controller IP using SystemVerilog and the Universal Verification Methodology (UVM). The project aims to validate the correctness, robustness, and specification compliance of the design under various operating conditions through a structured and reusable verification framework.

### Technical Scope

The technical scope includes the development of a UVM-compliant verification environment comprising sequence items, sequences, drivers, monitors, agents, environment, scoreboard, reference model, and assertions. The verification is performed at the simulation level, validating behavior across multiple clock domains and ensuring adherence to APB protocol requirements. Functional coverage models are implemented to measure verification completeness, while debugging is carried out using waveform analysis and UVM reporting mechanisms.

### Functional Scope

The functional scope covers verification of all major design functionalities, including Reset, Configuration, and Operation modes. It validates item configuration, item selection, multi-item transactions, currency accumulation, dispense decision logic, and change calculation. Exceptional scenarios such as insufficient balance, out-of-stock item selection, and invalid transaction conditions are also verified. The scope ensures correctness of inventory tracking, dispensed item counts, and compliance with specified latency constraints.

### Design and Development Scope

The design and development scope focuses on translating the design specification into a verification test plan and implementing a layered and reusable UVM testbench architecture. This includes development of directed and constrained-random sequences, a self-checking scoreboard, and a golden reference model for automated result comparison. Functional coverage points are defined and refined to achieve coverage closure, while maintaining scalability and reusability of the verification environment.

### Verification Scope and Constraints

The verification scope is limited to simulation-based functional verification at the RTL level. The project does not include RTL design modifications, formal verification, hardware emulation, synthesis, or post-silicon validation. Power analysis, physical timing closure, and integration with real-world mechanical components are outside the scope. Despite these constraints, the project follows industry-standard verification practices, providing strong practical exposure to professional Design Verification workflows.

- ## Limitations of the project

### 1. Simulation-Based Verification Limitation

The Vending Machine Controller IP has been verified only using RTL simulation with Universal Verification Methodology (UVM). The design has not been implemented or tested on FPGA or ASIC hardware platforms. Hence, real-world physical effects such as signal integrity issues, clock skew, voltage variations, and thermal conditions are not considered.

### 2. Functional Verification Scope Only

The verification effort is limited to functional correctness based on the provided specification. Advanced verification techniques such as formal verification, hardware emulation, and post-silicon validation are outside the scope of this project.

### 3. Fixed Currency Denominations Support

The controller supports only predefined currency denominations as specified (such as 5, 10, 15, 20, 50, and 100 INR). Support for additional denominations, coins, or international currencies would require RTL and verification environment modifications.

### 4. No Power and Physical Timing Analysis

This project does not include power analysis, low-power verification, or detailed physical timing closure. Latency verification is performed at a functional level only, without accounting for synthesis or layout-related timing effects.

### 5. Limited Stress and Regression Testing

Due to simulation time constraints, extensive stress testing, long-duration transaction verification, and large-scale regression testing are limited. Some rare corner-case scenarios may not be fully exercised.

### 6. No Real-World Hardware Integration

The verification environment does not integrate with real-world vending machine components such as coin validators, note acceptors, display units, or mechanical dispensing mechanisms. User interactions are modeled through simulated inputs only.

### 7. Specification-Dependent Verification Environment

The UVM testbench is developed strictly based on the given specification. Any future feature enhancements or protocol changes would require corresponding updates to the testbench architecture, sequences, and coverage models.

- ## Innovation

### 1. UVM-Based Industry-Oriented Verification Framework

The project introduces an industry-standard Universal Verification Methodology (UVM) framework for verifying the Vending Machine Controller IP. Instead of relying on basic directed testing, a layered, reusable, and scalable verification architecture is implemented, closely reflecting real-world semiconductor verification practices.

### 2. Self-Checking Verification Using Reference Model and Scoreboard

An innovative aspect of the project is the development of a self-checking verification mechanism using a golden reference model and scoreboard. This approach enables automatic comparison between expected and actual DUT outputs, significantly reducing manual checking effort and improving verification accuracy.

### 3. Constrained-Random Stimulus Generation

The project applies constrained-random testing techniques to generate a wide variety of transaction scenarios, including different item selections, currency combinations, and quantities. This enhances the ability to uncover corner-case bugs that are difficult to detect using only directed test cases.

### 4. Functional Coverage-Driven Verification

The verification environment incorporates functional coverage models to measure verification completeness. Coverage points are defined for item indices, currency denominations, quantities, and operational modes, enabling coverage-driven closure and systematic validation of the design functionality.

### 5. Multi-Mode and Multi-Clock Verification Handling

The project innovatively verifies multi-mode operation (Reset, Configuration, and Operation modes) and manages interactions across multiple clock domains (system clock and configuration clock). This adds robustness to the verification process and closely mirrors real-world IP verification challenges.

### 6. Scalable and Reusable Verification Environment

The UVM testbench is designed to be parameterized and reusable, allowing easy scalability for different numbers of items and currency ranges without major restructuring. This makes the verification environment adaptable for future enhancements or similar controller designs.

## *Project Objectives*

### 1. Develop a UVM-Based Verification Environment for Vending Machine Controller IP

**Objective:**

To design and implement a structured, modular, and reusable **UVM-based verification environment** using **SystemVerilog** for verifying the functionality of the Vending Machine Controller IP.

**Expected Outcome:**

A complete UVM testbench consisting of sequences, drivers, monitors, scoreboard, and reference model that follows industry-standard verification practices.

### 2. Verify Reset and Initialization Behavior

**Objective:**

To validate that all internal registers, counters, memories, and output signals are properly initialized during **Reset Mode** as per the specification.

**Expected Outcome:**

Correct reset behavior confirmed, ensuring the DUT starts from a known and stable state before configuration or operation.

### 3. Verify Configuration Mode Functionality Using APB Interface

**Objective:**

To verify correct read and write operations of configuration registers through the **APB interface**, including item values, available item count, and dispensed item count.

**Expected Outcome:**

Successful validation of APB transactions with correct programming and retrieval of configuration data.

### 4. Validate Mode Switching Between Configuration and Operation Modes

**Objective:**

To ensure seamless and correct transition between **Configuration Mode** and **Operation Mode** without data corruption or functional errors.

**Expected Outcome:**

Reliable mode switching with proper isolation of configuration updates from real-time vending operations.

### 5. Verify Item Selection and Quantity Handling Logic

**Objective:**

To validate correct processing of item selection signals and the number of items selected during vending transactions.

**Expected Outcome:**

Accurate identification of selected items and quantities for each transaction in operation mode.

### 6. Verify Currency Handling and Accumulation Logic

**Objective:**

To verify correct handling, accumulation, and validation of supported currency denominations inserted by the user.

**Expected Outcome:**

Proper accumulation of currency values and correct comparison against the total cost of selected items.

### 7. Verify Dispense Decision and Change Calculation Logic

**Objective:**

To ensure that the controller generates correct dispense signals, item numbers, item quantities, and currency change when sufficient balance is provided.

**Expected Outcome:**

Accurate dispense outputs and correct change calculation aligned with the functional specification.

### 8. Verify Exceptional and Corner-Case Scenarios

**Objective:**

To validate the controller’s behavior under exceptional conditions such as **insufficient balance**, **out-of-stock items**, and **invalid item selection**.

**Expected Outcome:**

Robust handling of error scenarios with correct responses and no functional instability.

### 9. Achieve Functional Coverage Closure

**Objective:**

To define and collect **functional coverage** across item indices, currency denominations, quantities, and operational modes.

**Expected Outcome:**

High functional coverage achieved, indicating thorough verification of all critical functional scenarios.

### 10. Gain Practical Experience in Industry-Level RTL Verification

**Objective:**

To gain hands-on experience with **UVM methodology**, verification planning, debugging, and coverage-driven verification techniques.

**Expected Outcome:**

Enhanced practical knowledge and readiness for professional roles in **VLSI Design Verification**.

***Methodology and Results***

- ## Methods/Technology used

### 1. RTL Design Verification Using UVM

The Vending Machine Controller IP was verified using the **Universal Verification Methodology (UVM)**, an industry-standard verification framework based on SystemVerilog. UVM enabled the creation of a structured, reusable, and scalable verification environment to validate the functional correctness of the design.

### 2. SystemVerilog-Based Verification Environment

SystemVerilog was used as the primary verification language to develop UVM components such as interfaces, sequence items, sequences, drivers, monitors, scoreboards, and assertions. Its object-oriented features helped in building modular and maintainable verification code.

### 3. Modular UVM Testbench Architecture

The verification environment was designed using a modular UVM architecture consisting of agents, environment, scoreboard, and test classes. This modular structure allowed independent development, easier debugging, and reusability of components across different test scenarios.

### 4. Directed and Constrained-Random Testing

Both directed testing and constrained-random stimulus generation techniques were used. Directed tests verified specific functionalities such as reset behavior and configuration operations, while constrained-random testing helped uncover corner cases related to item selection, currency handling, and transaction sequences.

### 5. APB Interface Verification

The **APB (Advanced Peripheral Bus)** protocol was verified to ensure correct read and write operations during configuration mode. Register programming for item values, available stock, and dispensed item count was validated using APB transactions.

### 6. Functional Coverage-Driven Verification

Functional coverage models were implemented to measure verification completeness. Coverage points were defined for item numbers, currency denominations, quantities, operational modes, and transaction scenarios, enabling systematic coverage closure.

### 7. Scoreboard and Reference Model

A golden reference model was developed to predict the expected behavior of the DUT. The scoreboard compared DUT outputs with reference outputs automatically, enabling self-checking verification and faster identification of functional mismatches.

### 8. Assertion-Based Verification

SystemVerilog assertions were used to validate protocol behavior, signal validity, and timing relationships. Assertions helped detect functional violations early in the simulation process.

### 9. Simulation-Based Debugging and Analysis

Verification was carried out using RTL simulation. Functional mismatches were debugged using waveform analysis and UVM log messages, enabling detailed observation of signal behavior and transaction flow.

## ● Tools/Software used

### 1. SystemVerilog

SystemVerilog was used as the primary hardware verification language for developing the UVM testbench. It enabled implementation of interfaces, sequence items, assertions, and functional coverage, supporting object-oriented and constrained-random verification features.

### 2. Universal Verification Methodology (UVM)

UVM was used as the verification framework to build a structured, modular, and reusable testbench. It provided standardized components such as drivers, monitors, agents, scoreboards, and tests, enabling systematic verification of the Vending Machine Controller IP.

### 3. RTL Simulator

An RTL simulation tool was used to execute the design and UVM testbench together. The simulator enabled functional verification of the DUT, execution of test cases, waveform generation, and observation of signal behavior across different operating modes.

### 4. Waveform Viewer

A waveform viewer was used to analyze simulation results and debug functional mismatches. It helped in tracking signal transitions, validating timing behavior, and understanding interactions between DUT and testbench components.

### 5. EDA Playground

EDA Playground was used as an online verification platform for writing, compiling, and simulating SystemVerilog and UVM code. It provided quick access to simulators and simplified testing without the need for local tool installation.

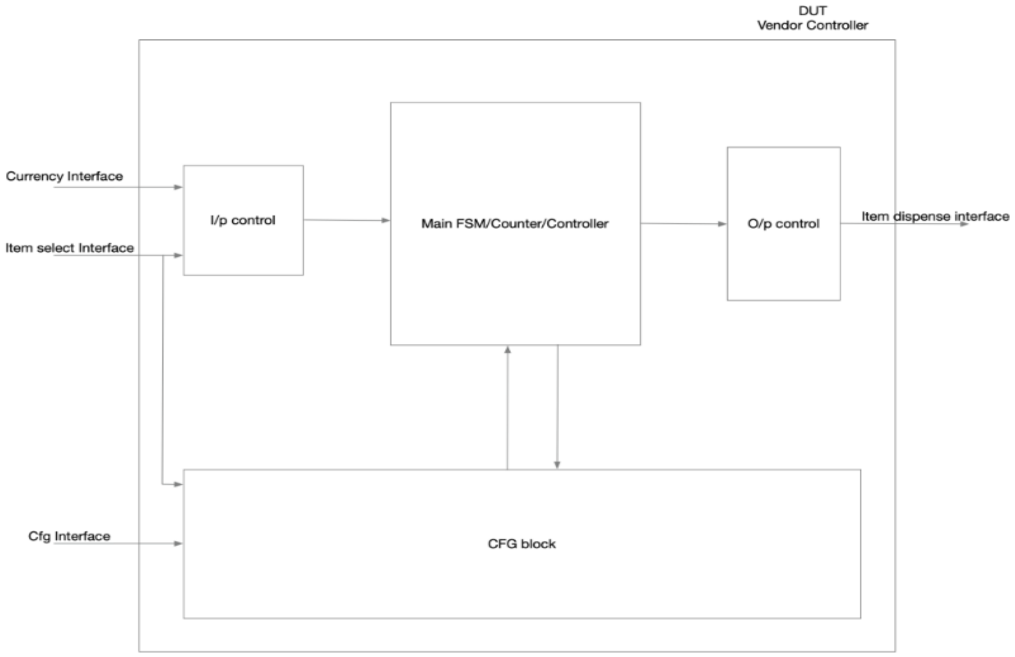
### 6. Version Control System (GitHub)

GitHub was used for version control and project management. It helped in maintaining different versions of the verification code, tracking changes, and sharing the project repository for review and collaboration.

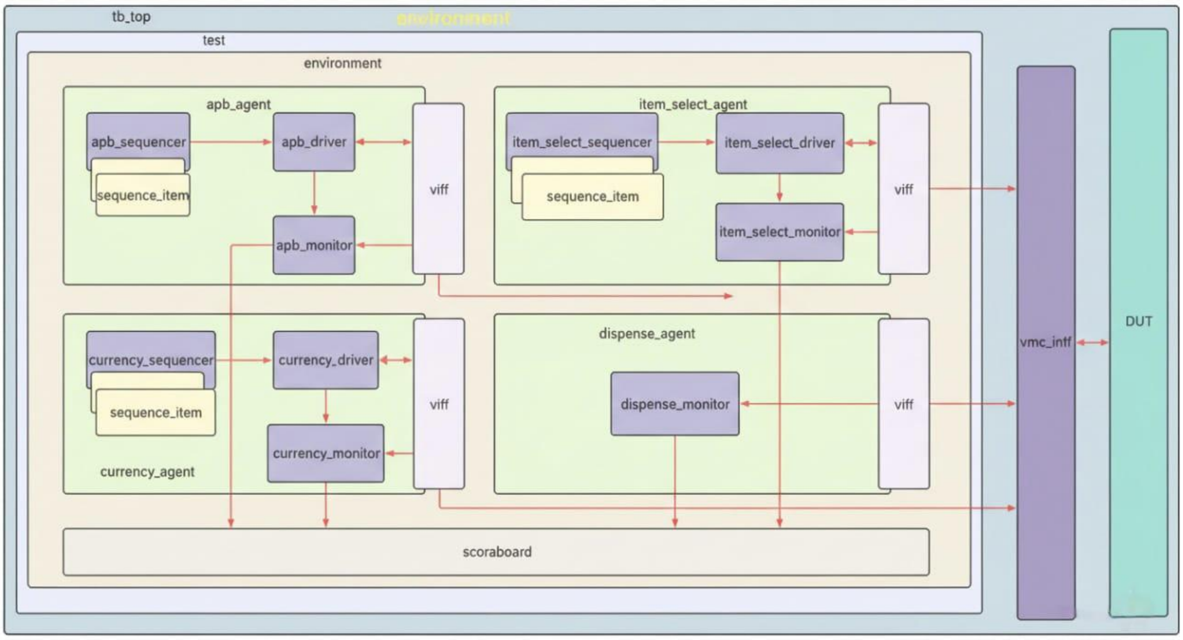
### 7. Documentation Tools

Word processors and documentation tools were used to prepare the project report, verification documentation, and test descriptions. These tools supported clear presentation of methodology, results, and learnings.

- **Project Block Diagram**



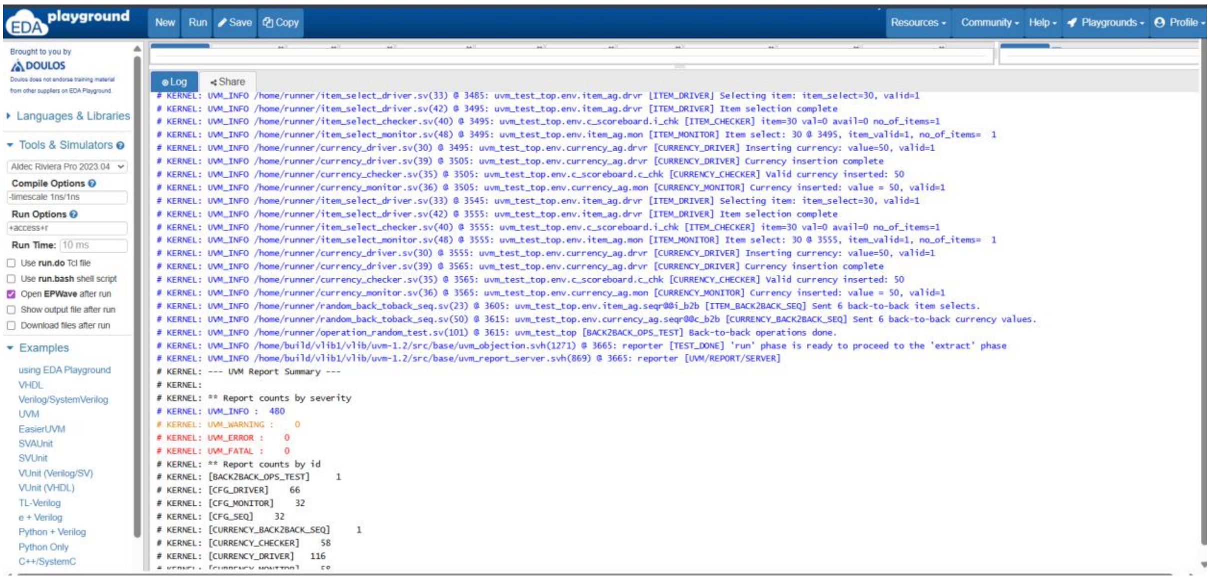
- **Project Architecture**



- This architecture represents a **UVM-based verification environment** for the Vending Machine Controller IP.
- Separate **UVM agents** (APB, currency, item selection, and dispense) are used to verify each DUT interface independently.
- Each agent contains a **sequencer, driver, and monitor** to generate stimulus and capture DUT responses via virtual interfaces.
- All monitor outputs are connected to a centralized **scoreboard** for expected vs actual result comparison.
- The modular structure enables **reusability, scalability, and coverage-driven verification**, following industry standards.

## Results:

- **Output Log:**



Aldec Riviera Pro 2023.04

Compile Options: -timescale 1ns/1ns

Run Options: +access+r

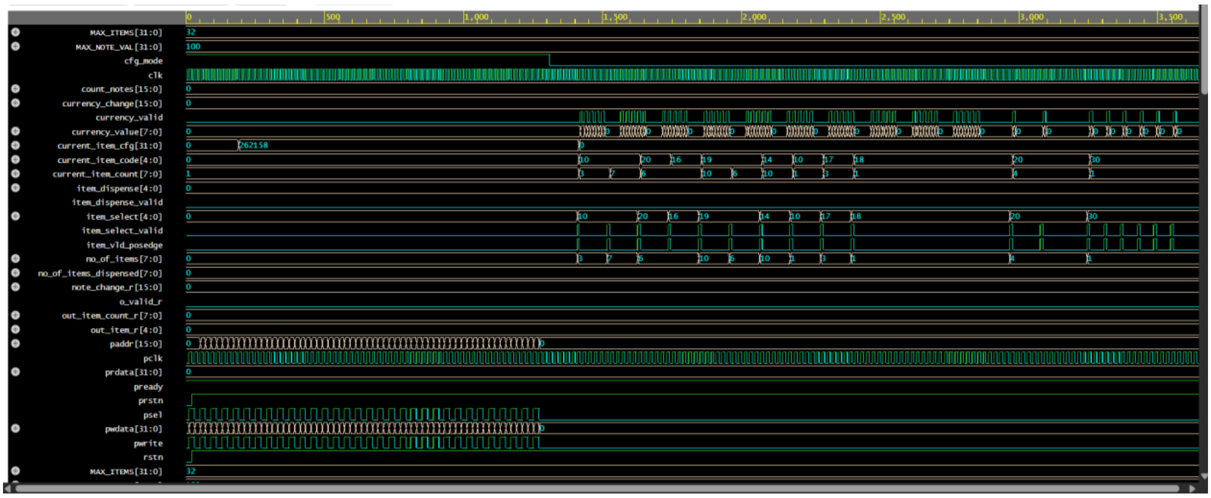
Run Time: 10 ms

☐ Use run.do Tcl file
☐ Use run.bash shell script
☑ Open EPWave after run
☐ Show output file after run
☐ Download files after run

```
# KERNEL: UVM_INFO /home/runner/item_select_driver.sv(33) @ 3485: uvm_test_top.env.item_ag.drvr [ITEM_DRIVER] Selecting item: item_select=30, valid=1
# KERNEL: UVM_INFO /home/runner/item_select_driver.sv(42) @ 3495: uvm_test_top.env.item_ag.drvr [ITEM_DRIVER] Item selection complete
# KERNEL: UVM_INFO /home/runner/item_select_checker.sv(40) @ 3495: uvm_test_top.env.c_scoreboard.i_chk [ITEM_CHECKER] item=30 val=0 avail=0 no_of_items=1
# KERNEL: UVM_INFO /home/runner/item_select_monitor.sv(48) @ 3495: uvm_test_top.env.item_ag.mon [ITEM_MONITOR] Item select: 30 @ 3495, item_valid=1, no_of_items=  1
# KERNEL: UVM_INFO /home/runner/currency_driver.sv(30) @ 3495: uvm_test_top.env.currency_ag.drvr [CURRENCY_DRIVER] Inserting currency: value=50, valid=1
# KERNEL: UVM_INFO /home/runner/currency_driver.sv(39) @ 3505: uvm_test_top.env.currency_ag.drvr [CURRENCY_DRIVER] Currency insertion complete
# KERNEL: UVM_INFO /home/runner/currency_checker.sv(35) @ 3505: uvm_test_top.env.c_scoreboard.c_chk [CURRENCY_CHECKER] Valid currency inserted: 50
# KERNEL: UVM_INFO /home/runner/currency_monitor.sv(36) @ 3505: uvm_test_top.env.currency_ag.mon [CURRENCY_MONITOR] Currency inserted: value = 50, valid=1
# KERNEL: UVM_INFO /home/runner/item_select_driver.sv(33) @ 3545: uvm_test_top.env.item_ag.drvr [ITEM_DRIVER] Selecting item: item_select=30, valid=1
# KERNEL: UVM_INFO /home/runner/item_select_driver.sv(42) @ 3555: uvm_test_top.env.item_ag.drvr [ITEM_DRIVER] Item selection complete
# KERNEL: UVM_INFO /home/runner/item_select_checker.sv(40) @ 3555: uvm_test_top.env.c_scoreboard.i_chk [ITEM_CHECKER] item=30 val=0 avail=0 no_of_items=1
# KERNEL: UVM_INFO /home/runner/item_select_monitor.sv(48) @ 3555: uvm_test_top.env.item_ag.mon [ITEM_MONITOR] Item select: 30 @ 3555, item_valid=1, no_of_items=  1
# KERNEL: UVM_INFO /home/runner/currency_driver.sv(30) @ 3555: uvm_test_top.env.currency_ag.drvr [CURRENCY_DRIVER] Inserting currency: value=50, valid=1
# KERNEL: UVM_INFO /home/runner/currency_driver.sv(39) @ 3565: uvm_test_top.env.currency_ag.drvr [CURRENCY_DRIVER] Currency insertion complete
# KERNEL: UVM_INFO /home/runner/currency_checker.sv(35) @ 3565: uvm_test_top.env.c_scoreboard.c_chk [CURRENCY_CHECKER] Valid currency inserted: 50
# KERNEL: UVM_INFO /home/runner/currency_monitor.sv(36) @ 3565: uvm_test_top.env.currency_ag.mon [CURRENCY_MONITOR] Currency inserted: value = 50, valid=1
# KERNEL: UVM_INFO /home/runner/random_back_toback_seq.sv(23) @ 3605: uvm_test_top.env.item_ag.seqr@@i_b2b [ITEM_BACK2BACK_SEQ] Sent 6 back-to-back item selects.
# KERNEL: UVM_INFO /home/runner/random_back_toback_seq.sv(50) @ 3615: uvm_test_top.env.currency_ag.seqr@@c_b2b [CURRENCY_BACK2BACK_SEQ] Sent 6 back-to-back currency values.
# KERNEL: UVM_INFO /home/runner/operation_random_test.sv(101) @ 3615: uvm_test_top [BACK2BACK_OPS_TEST] Back-to-back operations done.
# KERNEL: UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_objection.svh(1271) @ 3665: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# KERNEL: UVM_INFO /home/build/vlib1/vlib/uvm-1.2/src/base/uvm_report_server.svh(869) @ 3665: reporter [UVM/REPORT/SERVER]
# KERNEL: --- UVM Report Summary ---
# KERNEL:
# KERNEL: ** Report counts by severity
# KERNEL: UVM_INFO :  480
# KERNEL: UVM_WARNING :    0
# KERNEL: UVM_ERROR :    0
# KERNEL: UVM_FATAL :    0
# KERNEL: ** Report counts by id
# KERNEL: [BACK2BACK_OPS_TEST]     1
# KERNEL: [CFG_DRIVER]    66
# KERNEL: [CFG_MONITOR]    32
# KERNEL: [CFG_SEQ]    32
# KERNEL: [CURRENCY_BACK2BACK_SEQ]     1
# KERNEL: [CURRENCY_CHECKER]    58
# KERNEL: [CURRENCY_DRIVER]   116
# KERNEL: [CURRENCY_MONITOR]    58
```

- This architecture represents a **UVM-based verification environment** for the Vending Machine Controller IP.
- Separate **UVM agents** (APB, currency, item selection, and dispense) are used to verify each DUT interface independently.
- Each agent contains a **sequencer, driver, and monitor** to generate stimulus and capture DUT responses via virtual interfaces.
- All monitor outputs are connected to a centralized **scoreboard** for expected vs actual result comparison.
- The modular structure enables **reusability, scalability, and coverage-driven verification**, following industry standards.

- **Output Waveforms:**



- The waveform illustrates the **end-to-end functional operation** of the Vending Machine Controller during simulation.
- **Item selection and currency input signals** toggle in synchronization with the clock, indicating valid user transactions.
- Internal registers such as **current item code, item count, and currency value** update correctly after each valid input.
- The **item_dispense and item_dispense_valid signals** assert when sufficient currency is detected and stock is available.
- APB signals (PSEL, PWRITE, PWDATA, PRDATA) show proper **configuration and control transactions** without protocol violations.

- **Project GitHub Link:**

  https://github.com/sure-trust/THIPPI-REDDY-SASIKALA-g2-25-integrated-vlsi-design

## Team Experience:

Completing the verification of the Vending Machine Controller using a **UVM-based SystemVerilog environment** was a highly enriching experience for our verification team. Throughout the phases of testbench architecture development, sequence creation, simulation, debugging, and coverage analysis, we gained strong practical exposure to **industry-standard verification methodologies**. This project strengthened our understanding of UVM components such as **agents, drivers, monitors, sequencers, scoreboards, and virtual interfaces**, and how they interact to verify complex digital designs.

By developing constrained-random test scenarios, back-to-back transactions, and corner-case checks, we learned how to ensure functional correctness and robustness of the design. Verifying asynchronous events like currency insertion and item selection enhanced our **debugging and analysis skills** using waveforms and UVM logs. Beyond technical expertise, this project improved our **team coordination, test planning, documentation, and time management skills**. Overall, the verification experience bridged the gap between theoretical knowledge and real-world VLSI verification practices, preparing us for future roles in Design Verification.

## Learning:

1. Thippi Reddy Sasikala (Team Leader – Verification: Currency Handling & Integration)

   a) Worked extensively on the currency handling verification, focusing on validating currency insertion logic, accumulation of multiple currency inputs, supported denomination checks, and correct balance computation as per the specification.

   b) Led the verification effort by coordinating between different functional blocks and ensuring proper integration of currency logic with item selection and dispense decision flow.

   c) Actively contributed to debugging simulation issues related to currency mismatch, insufficient balance scenarios, and change calculation, using UVM logs and waveform analysis to identify and resolve DUT versus expected behavior mismatches.

d) Gained strong practical experience in UVM-based verification flow, transaction-level modeling, system-level debugging, and team coordination, while enhancing leadership, planning, and technical decision-making skills.

2. Rahul Veldi (Verification – Item Select Agent & Checking)

a) Worked on the Item Select Agent verification, focusing on validating item selection logic, control signaling (item_select, item_select_valid, and quantity handling), and correct interaction with the DUT under normal and corner-case scenarios.

b) Contributed to functional checking and scoreboard integration for item selection, ensuring accurate expected-versus-actual comparisons related to selected item ID, requested quantity, availability status, and selection validity.

c) Actively participated in debugging simulation issues by analyzing UVM logs and waveforms, identifying protocol violations, timing mismatches, and incorrect DUT responses during item selection flows.

d) Gained strong hands-on experience in UVM agent architecture, sequence-to-driver communication, transaction monitoring, and reusable verification component development, enhancing understanding of item-level and system-level verification.

3. Anusha Bathala (Verification – CFG Block & Scoreboard)

a) Worked on the Configuration (CFG) block verification, focusing on validating APB-based register read/write operations, address decoding, and proper initialization of configuration parameters.

b) Developed and maintained the UVM scoreboard, implementing expected-versus-actual comparisons for item selection, currency handling, dispense decisions, and change calculation.

c) Actively participated in debugging simulation failures by analyzing UVM logs and waveforms, identifying mismatches between DUT behavior and specification.

d) Gained strong practical experience in transaction-level modeling, functional checking, and reusable verification component design, improving overall understanding of system-level verification.

- ## Learning Experience:

1. Thippi Reddy Sasikala (Team Leader)

   As a Team Leader at SURE ProEd, I gained a well-rounded learning experience that combined technical upskilling, leadership development, community service, and soft skills training. The structured program and disciplined environment helped me stay focused, organized, and consistent throughout the internship while working on an industry-oriented Design Verification project.

   Through the technical training and project work, I developed strong hands-on experience in UVM-based verification, particularly in currency handling and integration logic for the Vending Machine Controller. The continuous guidance from mentors helped me understand real-world verification practices, debugging methodologies, and system-level thinking. As a team leader, I was also responsible for coordinating tasks, supporting team members, and ensuring smooth progress of the project.

   In addition to technical learning, participation in community service activities and Soft Skills Training (SST) sessions enhanced my communication skills, leadership qualities, sense of responsibility, and professional attitude. Overall, the experience at SURE Trust significantly strengthened my technical competence, teamwork abilities, and confidence, preparing me to take on future challenges in the VLSI Design Verification domain.

2. Rahul Veldi (Team Member)

   As a team member, I worked on the verification of the Item Select Agent and Dispense functionality for the Vending Machine Controller using the UVM methodology. My role primarily involved designing and implementing item selection and dispense-related sequences, validating correct handling of user inputs such as item ID, quantity, and selection validity under various operating conditions.

   I developed and executed multiple UVM sequences covering normal operations, invalid selections, boundary cases, and error scenarios, ensuring proper interaction between the item select logic, dispense control, and system outputs. I also verified the correctness of dispense behavior, including item count updates and change handling, by analyzing simulation results and DUT responses.

   Through detailed simulation runs, UVM log inspection, and waveform debugging, I identified and resolved functional mismatches between expected behavior and DUT implementation. This work strengthened my understanding of UVM agents, sequencer-driver coordination, transaction flow, and system-level verification, and provided hands-on experience with real-world Design Verification practices.

3. Anusha Bathala (Team Member)

As a team member, I worked on the verification of the Configuration (CFG) block and the development of the scoreboard for the Vending Machine Controller using UVM methodology. This responsibility helped me gain a clear understanding of how configuration registers control item pricing, availability, and system behavior through the APB interface.

I contributed to validating APB read and write transactions, ensuring correct address decoding, data integrity, and proper synchronization between clock domains. In addition, I developed the scoreboard logic to compare expected and actual outputs for item selection, currency processing, item dispense, and change return operations. Through extensive simulation, log analysis, and waveform debugging, I verified the correctness of system behavior across normal and corner-case scenarios.

This experience enhanced my skills in transaction-level modeling, functional verification, debugging, and system-level analysis, and provided strong practical exposure to industry-oriented Design Verification workflows.

## ● <u>Objectives:</u>

The primary objective of the Vending Machine Controller project was to design and implement a digital IP core that defines the architecture, interfaces, and operational behavior of a smart vending machine controller. The key objectives of the project are as follows:

1. To develop a **reusable and scalable UVM-based verification environment** capable of validating different configurations of the vending machine controller IP..
2. To verify correct functionality for **all supported item configurations**, including up to 1024 products, and proper handling of multiple currency denominations such as ₹5, ₹10, ₹15, ₹20, ₹50, and ₹100.
3. To ensure that the controller meets **performance requirements**, validating that the item dispense decision occurs within the specified number of system clock cycles after valid currency insertion.
4. To verify correct operation of all **three operating modes**:
   - **Reset Mode**: Validation of proper initialization of registers and memories
   - **Configuration Mode**: Verification of APB-based item price and availability loading
   - **Operation Mode**: Verification of item selection, currency processing, dispensing, and change return behavior

5. To validate proper functionality and protocol compliance of **all system interfaces**, including:
   - Clock and reset behavior
   - Currency and item selection inputs
   - Item dispense and change return outputs
   - APB interface for configuration and status monitoring

6. To verify **accurate inventory management**, ensuring correct updates of available item counts and total dispensed counts across transactions.

7. To ensure **robust error handling**, including correct currency return for unavailable items, in-valid selections, and insufficient balance conditions.

8. To achieve **maximum functional coverage** by verifying normal, boundary, and corner-case scenarios using constrained-random testing and scoreboard-based checking.

## ● Achievements:

The verification phase of the **Vending Machine Controller project** was successfully completed, and all verification objectives were achieved using an industry-standard methodology. The major achievements of the verification effort are listed below:

### 1. Complete UVM-Based Verification Environment

A **modular, reusable, and scalable UVM verification environment** was successfully developed using SystemVerilog. The environment was designed to support multiple configurations of the vending machine controller, enabling efficient verification across different item counts, pricing setups, and operational modes.

### 2. Comprehensive Mode Verification

All three operating modes—**Reset, Configuration, and Operation**—were thoroughly verified. Reset behavior was validated for proper initialization, APB-based configuration was verified for correct register access and data loading, and real-time operation mode was verified for accurate item selection, currency processing, dispensing, and change return.

### 3. Robust APB Interface Verification

The **APB configuration interface** was rigorously verified for correct read and write transactions, address decoding, data integrity, and protocol compliance. This ensured reliable configuration of item prices, availability, and system parameters.

### 4. Accurate Functional Checking Using Scoreboard

A **UVM scoreboard** was successfully implemented to perform expected-versus-actual comparisons for all major transactions. This enabled precise verification of item dispense decisions, currency handling, inventory updates, and error responses under normal and corner-case scenarios.

### 5. Verification of Asynchronous and Corner-Case Scenarios

The verification environment effectively validated **asynchronous currency inputs**, back-to-back transactions, insufficient balance cases, and out-of-stock conditions. This ensured robust system behavior under real-world usage patterns.

### 6. High Functional Coverage and Clean Simulation Results

Through constrained-random testing and directed test scenarios, **high functional coverage** was achieved. All simulations completed with **zero UVM errors or fatal messages**, confirming the functional correctness and stability of the design.

## ● Conclusion

The verification of the **Vending Machine Controller IP** was successfully completed using an industry-standard **UVM-based verification methodology**, ensuring functional correctness, reliability, and robustness of the design. A structured and reusable verification environment was developed to validate the controller's behavior across all operating modes, including Reset, Configuration, and Operation modes. Through systematic verification, the design was proven to meet its functional and timing requirements under both normal and corner-case scenarios.

The verification process effectively validated **real-time transaction handling**, including item selection, currency insertion, dispensing decisions, and change return logic. Special focus was placed on verifying **asynchronous inputs and clock domain interactions**, ensuring stable and error-free operation under realistic user-driven conditions. The use of a centralized scoreboard enabled accurate expected-versus-actual comparisons, strengthening confidence in the design's correctness.

The **APB configuration interface** was thoroughly verified to ensure reliable register access, correct data loading, and safe configuration of item prices and inventory counts. Constrained-random stimulus generation and coverage-driven testing helped uncover potential edge cases and improved overall test completeness. As a result, the verification environment confirmed that the vending machine controller is stable, scalable, and suitable for integration into larger SoC-based systems.

Overall, the verification effort ensured that the design not only meets the specification but also demonstrates **high reliability, reusability, and maintainability**, making it well-prepared for real-world deployment and future enhancements.

## ● Future Scope:

As digital systems continue to evolve, verification methodologies must also advance to address growing system complexity and emerging technologies. The following enhancements can further strengthen the verification of the vending machine controller:

### 1. Assertion-Based Verification (ABV)

SystemVerilog Assertions (SVA) can be added to continuously monitor critical protocol rules, timing constraints, and functional properties. Assertions improve bug detection speed and enhance design reliability.

### 2. Formal Verification Integration

Formal verification techniques can be applied to mathematically prove correctness of key control logic, such as state transitions, deadlock freedom, and safety properties, complementing simulation-based verification.

### 3. Power-Aware Verification

Future verification can include low-power scenarios by validating clock gating, power-down modes, and wake-up behavior, ensuring energy-efficient operation for next-generation vending systems.

### 4. SoC-Level and Regression Verification

The verification environment can be extended to **SoC-level integration**, validating interactions between the vending controller and other system components. Automated regression testing can further improve long-term stability.

### 5. Enhanced Coverage Metrics and Debug Automation

Advanced coverage models and automated debug tools can be incorporated to improve test completeness and reduce verification cycle time, increasing productivity and confidence in design quality.