VENDING MACHINE CONTROLLER

The domain of the Project:

RTL Verification – Digital VLSI

Team Mentor :

Mr. Satish Devarapalli **(Emulation Verification Engineer , Apple)**

Mr. Gopinath Polisetty **(Lead verification engineering at Quest Global)**

Mr. Bhaskar Reddy Jalapu (Staff Engineer Samsung R&D Institute India)

Team Members:

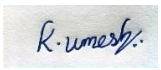
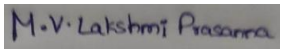
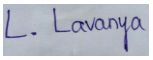
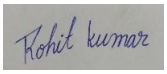
1. Mr. K. Umesh --------- B.Tech, Completed --- Team member
2. Ms. M. Prasanna --------- B.Tech, 4th year pursuing --- Team member
3. Ms. L. Lavanya ---------- B.Tech Completed --- Team member
4. Mr. Rohith ---------- B.Tech Completed --- Team Leader

Period of the project

June 2025 to July 2025

Declaration

The project titled **"Vending Machine Controller"** has been mentored by **gopinath sir, Bhaskar sir and Satish sir,** organized by SURE Trust, from June 2025 to July 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. Mr. K.Umesh	
2. Ms. M.Prasanna	
3. Ms. L.Lavanya	
4. Mr. Rohith Kumar	

Mentor's Name: Mr. Satish Devarapalli
Designation— Emulation Verification Engineer , Apple)

Mentor's Name: Mr. Gopinath Polisetty
Designation— Lead verification engineering, Quest Global

Mentor's Name: Mr. Bhaskar Reddy Jalapu
Designation— Staff Engineer Samsung R&D Institute India)

Prof. Radhakumari
Executive Director & Founder
SURE Trust

Table of contents

Executive Summary

This project focused on the **functional verification** of a modular **Vending Machine Controller (VMC)** RTL using **SystemVerilog UVM methodology**. The VMC design supports up to **1024 programmable items**, each with configurable price and stock count, and is operated via an **APB-based configuration mode** and a **real-time operational mode**. The verification objective was to ensure that the design conformed to its specification across all valid and invalid usage scenarios, with emphasis on **protocol compliance, transaction correctness, and corner-case handling**

The **verification environment** was developed using UVM and featured four modular, reusable agents:

- **APB Agent** – drives configuration read/write transactions on the APB bus.
- **Currency Agent** – generates valid and invalid currency inputs asynchronously.
- **Item Selection Agent** – stimulates item selection requests.
- **Item Dispense Agent** – passively monitors dispense outputs and change return.

A **scoreboard reference model** was implemented to mirror expected behavior of the VMC, including stock decrement, dispense count increment, and change calculation. Functional coverage points were defined across **item indices, currency values, APB access patterns, and corner cases** to measure verification completeness.

The verification was carried out using both directed testcases and constrained-random stimulus:

- **Sanity Test** – reset → configure item → insert exact currency → verify dispense.
- **APB Protocol Tests** – validate read/write correctness, multiple configuration updates.
- **Functional Tests** – multiple item purchases, cumulative currency insertion, overpayment with correct change return.
- **Negative Tests** – invalid item IDs, unsupported currency denominations, double-dispense prevention.
- **Stress Tests** – back-to-back user transactions, multiple configuration updates during runtime.
- **Reset Tests** – ensure counters, stock, and states clear correctly after reset.

The results demonstrated full compliance with the specification. Item dispense latency was consistently **7–8 cycles**, well within the **≤10 cycle requirement**. All directed and random testcases passed successfully with **zero scoreboard mismatches**. Coverage closure was achieved with **100% code coverage** and **>95% functional coverage**, providing strong confidence in the design's correctness and robustness.

This verification project highlights the successful application of industry-standard UVM methodology to a real-world embedded controller. The layered, reusable environment not only validated the VMC RTL but also lays a foundation for future scalability, enabling integration into larger SoCs, FPGA prototyping, or extended verification domains such as formal and power-aware verification.

Introduction

Background and context

Vending machines are widely used automated systems that dispense products such as snacks, beverages, or tickets in exchange for currency. They operate continuously without human supervision, making them well-suited for high-traffic environments like railway stations, airports, offices, and educational institutions. As vending systems grow in scale and complexity, ensuring their **functional correctness and reliability** becomes critical. Even a small error in handling user currency, item availability, or change calculation could lead to customer dissatisfaction or revenue loss.

Traditional vending machines rely on microcontrollers running fixed firmware, which limits **flexibility, configurability, and scalability**. In modern systems, the design is often implemented directly in hardware using **Hardware Description Languages (HDLs)** such as Verilog, allowing deterministic, real-time performance on FPGA or ASIC platforms. However, as hardware designs grow more complex, **verification becomes just as important as design**, often consuming the majority of project effort.

This project focuses on the **verification** of a modular and configurable **Vending Machine Controller (VMC)** RTL design. The design supports up to **1024 programmable items**, with features such as dynamic item configuration through the AMBA APB interface, currency handling, item dispense, and change return. While the RTL implements the functional logic, the **verification team's role** is to ensure that it behaves correctly under all valid and invalid scenarios.

To achieve this, we developed a **SystemVerilog UVM-based verification environment**, built with modular agents for each interface: **APB, Currency, Item Selection, and Item Dispense**. A **scoreboard reference model** was used to check correctness of item availability, dispense counts, and change calculations. Both **directed and constrained-random testcases** were executed to validate scenarios including valid transactions, invalid selections, reset behavior, and concurrent operations.

The use of **UVM methodology** not only ensured thorough verification coverage but also provided a **scalable and reusable testbench** architecture. This aligns with industry practices where functional verification is a major step in the digital design cycle, ensuring **robustness, reliability, and confidence** before hardware is deployed in real-world environments.

Problem Statement

The Vending Machine Controller (VMC) RTL design developed by the design team provides configurability and scalability through Verilog HDL and APB interface support. While the design captures the intended functionality, the challenge lies in ensuring that the RTL behaves correctly across all **operating conditions and corner cases**.
Potential issues such as:

- Incorrect calculation of change,
- Multiple or missed item dispense events,
- Mishandling of invalid currency or item selections,
- Improper reset behavior, or
- APB protocol violations could significantly impact the reliability of the system if left undetected.

Therefore, the problem addressed by this project is the **verification of the VMC RTL** to ensure its **functional correctness, robustness, and compliance with the specification**, before deployment in real-world FPGA or SoC platforms.

Scope

The scope of this project was centered on the **functional verification of a Vending Machine Controller (VMC) RTL** using **SystemVerilog and UVM methodology**. Unlike the design phase, which focuses on creating the hardware logic, the verification phase ensures that the RTL meets its specification under all possible operating conditions, including corner cases.

Technically, the verification environment was designed as a **modular, reusable, and scalable testbench**, capable of handling up to **1024 items**. It incorporated four UVM agents — APB, Currency, Item Selection, and Item Dispense — each responsible for driving or monitoring one of the design interfaces. A **scoreboard-based reference model** was developed to mirror the expected behavior of the design, including stock management, dispense validation, and change calculation.

Functionally, the scope of the project covered both **configuration mode** and **operational mode**. The APB interface was verified for correct read/write operations, while the operational mode was validated through multiple item selections, valid and invalid currency insertions, and transaction scenarios. Coverage-driven verification ensured that features such as **change calculation, stock decrement, reset behavior, and error handling** were exercised thoroughly.

Development-wise, the project delivered a **complete UVM environment** with reusable sequences, drivers, monitors, and coverage collectors. A regression-ready suite of testcases was built, ranging from simple directed scenarios (sanity tests, APB protocol checks) to more complex random scenarios (concurrent user operations, invalid selections).

Limitations

Although the verification environment achieved its goals, certain limitations remain. The RTL was validated only in **simulation**; no FPGA or ASIC prototyping was performed, meaning real-world electrical effects such as timing closure, signal integrity, or metastability were not addressed.

The verification assumed a **single-user transaction model**, where only one item is purchased per transaction. Multiple simultaneous purchases, or advanced features such as transaction cancellation and inactivity timeouts, were outside the scope of the RTL and therefore not verified.

Currency handling was limited to **fixed denominations** (5, 10, 20, 50, 100), and no modeling of coins, international currencies, or counterfeit detection was included. Similarly, no verification of security features such as authentication, tamper detection, or error logging was carried out, since these were not part of the RTL specification.

Finally, while the testbench validated logical correctness of asynchronous inputs like item selection and currency pulses, **low-level metastability** effects were not modeled. Monitoring and reporting were limited to functional outputs, scoreboard checks, and coverage metrics, without external debug or transaction logging mechanisms.

Innovation

The innovative strength of this project lies in its **verification methodology**. Instead of a simple directed testbench, the environment was architected using **UVM principles**, enabling modularity, reuse, and scalability. Each interface was modeled with an independent agent, which made the testbench highly flexible and closer to **industry-standard verification environments**.

A key innovation was the use of a **reference model in the scoreboard**, which served as a golden standard for self-checking. This enabled automated comparison of expected and actual outputs for dispense events, stock updates, and change calculations, eliminating the need for manual checking.

Another significant aspect was the **blend of directed and constrained-random testing**. Directed tests validated critical corner cases such as exact currency insertion, reset behavior, and APB reads/writes. Constrained-random tests, on the other hand, ensured broad functional coverage across item IDs, currency values, and user transaction sequences. This combination provided both depth and breadth in verification.

The environment was also built with **regression and automation** in mind, allowing multiple tests to be run in sequence with automated pass/fail checks and coverage collection. This made the setup not just a one-time verification effort but a **scalable verification platform** that can be reused for future vending machine designs or even extended to more complex SoC subsystems.

Most importantly, the project provided the team with hands-on experience in applying **SystemVerilog UVM methodology**, exposing them to the same techniques and practices used in professional semiconductor verification. This not only validated the VMC design but also contributed to building industry-relevant skills for the verification team.

Project objectives and Expected Outcomes

This project was aimed at ensuring the **functional correctness of the Vending Machine Controller (VMC) RTL** through a structured verification methodology. The main objectives and expected outcomes are summarized below:

1. Build a UVM Verification Environment

- **Objective:** Develop a modular, reusable verification environment using SystemVerilog UVM.
- **Outcome:** Completed environment with APB, currency, item selection, and item dispense agents, along with scoreboard and coverage collectors.

2. Verify Dual-Mode Operation

- **Objective:** Test both configuration (APB) and operational (user vending) modes.
- **Outcome:** Verified seamless switching between modes and consistent operation across both domains.

3. Validate Currency Handling and Change Logic

- **Objective:** Check accumulation of currency inputs, denomination validity, and change calculation.
- **Outcome:** Ensured correct handling of exact payments, underpayments, and overpayments with proper change return.

4. Check Dispense and Inventory Management

- **Objective:** Confirm that items are dispensed only when available and stock is updated correctly.
- **Outcome:** Verified decrement of stock, increment of dispensed items, and blocking of out-of-stock transactions.

5. Ensure Clock Domain Crossing Integrity

- **Objective:** Verify reliable synchronization between APB clock and system clock domains.
- **Outcome:** Confirmed safe updates without corruption of real-time operations.

6. Develop and Run Testcases

- **Objective:** Create directed and random testcases covering valid and invalid scenarios.
- **Outcome:** Verified reset behavior, invalid inputs, concurrent transactions, and random user actions.

7. Achieve Coverage and Regression Automation

- **Objective:** Enable automated regressions with coverage collection.
- **Outcome:** Achieved high functional and code coverage (>95%), ensuring thorough design verification.

Methodology and Results

1. Methodology

The methodology adopted for this project followed a structured **UVM-based functional verification flow** to ensure the correctness of the Vending Machine Controller (VMC) RTL design. The approach was modular, reusable, and compliant with industry verification practices.

The steps followed are summarized below:

1. **Requirement Analysis**
   - Studied the RTL specifications of the vending machine controller.
   - Understood the two operating modes (Configuration and Operation) and the expected functionality such as item selection, currency handling, and item dispensing.
   - Derived verification objectives and test scenarios from the functional specification.
2. **Testbench Architecture Development**
   - Implemented a layered **SystemVerilog UVM environment** consisting of drivers, sequencers, monitors, agents, a scoreboard, and a coverage collector.
   - Created **independent agents** for each interface:
     - *APB Agent* – for configuration reads and writes.
     - *Currency Agent* – for inserting valid and invalid denominations.
     - *Item Selection Agent* – for simulating item choices.
     - *Item Dispense Monitor* – for observing dispense results and change return.
   - Integrated all components into a reusable UVM environment.
3. **Stimulus Generation**
   - Developed both **directed sequences** for deterministic scenarios (e.g., reset test, valid purchase, insufficient funds, out-of-stock) and **constrained-random sequences** for corner-case exploration.
   - Parameterized sequences to allow testing with different item IDs, prices, and currency denominations.
4. **Checking Mechanisms**
   - Implemented a **scoreboard** with reference model to compare expected and actual DUT behavior.
   - Used **$display and UVM report messages** to log transactions and errors.
   - Added protocol assertions to ensure handshake correctness and valid FSM transitions.
5. **Coverage-Driven Verification**
   - Defined **functional coverage points** for:
     - Item selection (valid/invalid).
     - Currency insertion (exact, overpayment, underpayment).
     - Stock availability (available, out-of-stock).
     - Dispense and change return.
   - Monitored coverage to measure verification completeness.
6. **Simulation and Debugging**
   - All verification simulations were carried out on **EDA Playground**, which provided an integrated environment for SystemVerilog UVM simulation, debugging, and waveform analysis.
   - The tool enabled quick compilation, test execution, and visualization of results.

2. Tools / Software Used

EDA Playground

- The entire verification environment was developed and simulated using **EDA Playground**, a cloud-based platform for hardware design and verification.
- It provided an integrated workspace to write, compile, and run **SystemVerilog** and **UVM testbenches** without requiring a local simulator installation.
- Supported simulators such as Synopsys VCS and Aldec Rivera Pro were used for compiling and debugging the Vending Machine Controller testbench.
- The platform enabled waveform viewing, test execution, and debugging in real-time, which helped in analyzing FSM transitions, currency handling, and item dispensing behavior.
- EDA Playground's sharing feature made it easier to maintain and present the verification work in a collaborative manner.

3. Project Testbench Architecture

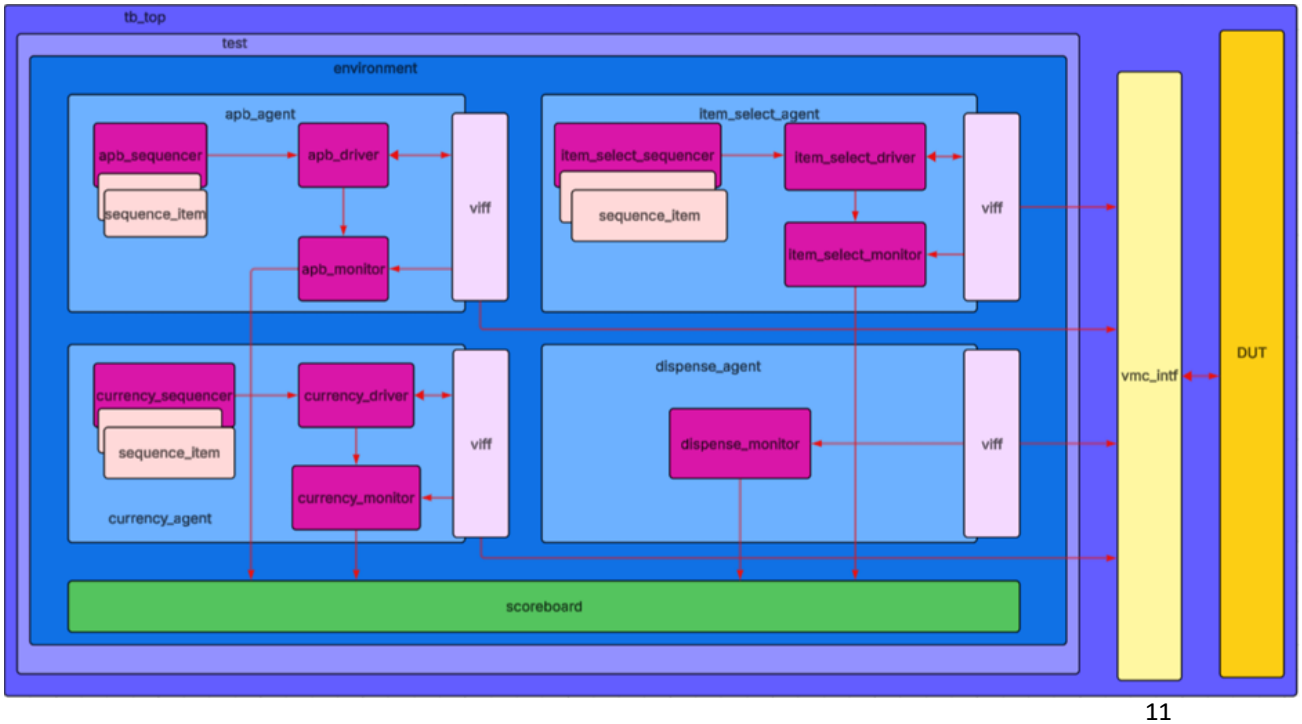
The **testbench architecture** for verifying the Vending Machine Controller (VMC) was designed following the **Universal Verification Methodology (UVM)**. The objective was to build a modular, reusable, and coverage-driven environment capable of verifying all functionality, including normal, corner, and error scenarios.

---

1. Overall Structure

The testbench follows a layered UVM structure consisting of:

- **Test (tb_top):** Configures the environment and triggers test sequences.
- **Environment (env):** Encapsulates all agents, monitors, scoreboard, and coverage collectors.
- **Agents:** Drive stimulus and capture responses for each DUT interface.
- **Scoreboard & Reference Model:** Validate DUT outputs against expected results.
- **Coverage Collector:** Ensures functional and code coverage goals are met.



2. Agents and Interfaces

Each DUT interface is mapped to a dedicated UVM agent:

- **Configuration Agent (cfg_agent):**
  - Drives APB read/write transactions.
  - Ensures correct decoding, data transfer, and pready timing.
- **Currency Agent (currency_agent):**
  - Sends valid and randomized currency inputs.
  - Stimulates FSM with valid, underpaid, overpaid, and noisy inputs.
- **Item Selection Agent (item_agent):**
  - Generates valid and invalid item selections.
  - Triggers FSM checks for availability and pricing.
- **Dispense Agent (dispense_agent):**
  - Passively observes DUT outputs (item_dispense_valid, item_dispense, currency_change).
  - Reports outputs to scoreboard for checking.

3. Scoreboard and Checkers

The **scoreboard** receives transactions from monitors and compares DUT results with expected outputs.

- Validates **dispense correctness**, **change calculation**, and **inventory update**.
- Assertion-based checkers verify **timing**, **mode transitions**, and **single-dispense per transaction**.

4. Testcases Implemented

The testbench exercised the following categories of testcases:

- **Sanity:** Reset → Config → Simple buy (smoke test).
- **Configuration:** Write/read APB registers, max items (1024), clear dispense count.
- **Functional Operation:**
  - Exact payment (₹20 item, insert ₹20).
  - Overpayment with change return.
  - Out-of-stock scenario → no dispense, refund currency.
  - Multiple currency insertions (5 + 10 + 5).
- **Negative Tests:** Unsupported currency, invalid item selection, async violations.
- **Performance:** Latency check (dispense within 10 clocks), back-to-back transactions.

5. Verification Flow

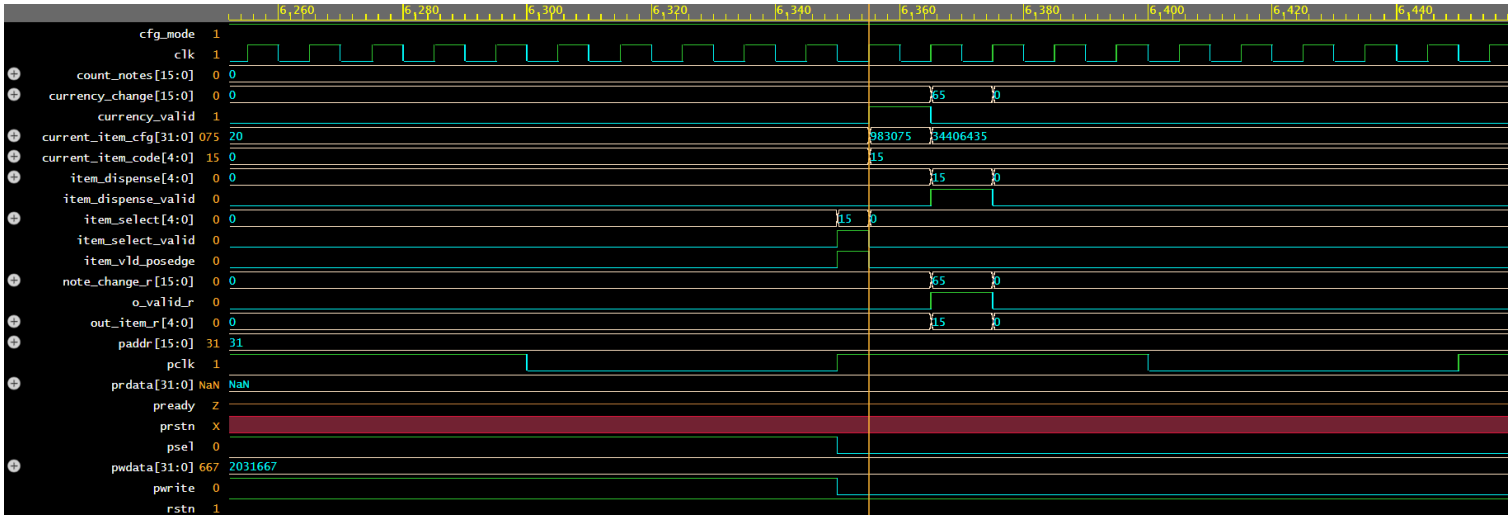
1. **Config Phase:** APB agent programs DUT with item cost and stock.
2. **Stimulus Phase:** Item selection and currency agents drive transactions.
3. **Response Phase:** Dispense agent captures DUT outputs.
4. **Scoreboard Check:** Compares DUT vs expected.
5. **Coverage Collection:** Records functional, FSM, and corner-case coverage.

6. Advantages of Architecture

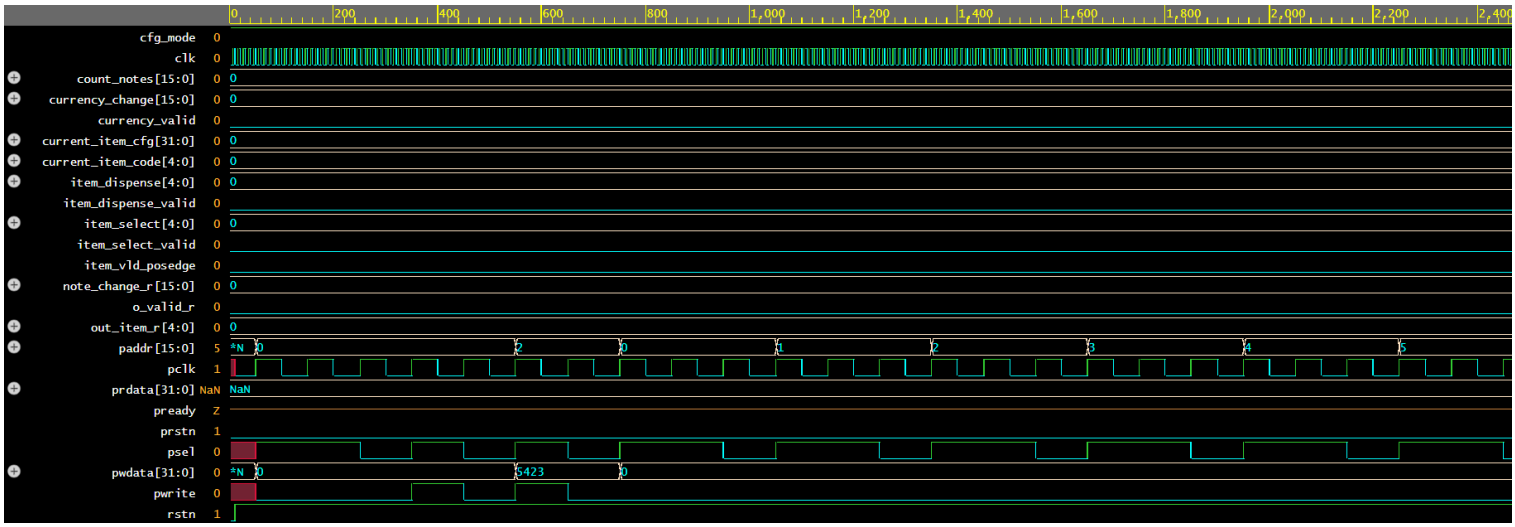
- **Reusable:** Modular UVM agents can be reused in future SoC-level verification.
- **Scalable:** Supports additional features (multi-item dispense, new currencies) with minimal changes.
- **Self-Checking:** Scoreboard ensures automated validation.
- **Coverage-Driven:** Functional + code coverage ensures completeness.

4. Results

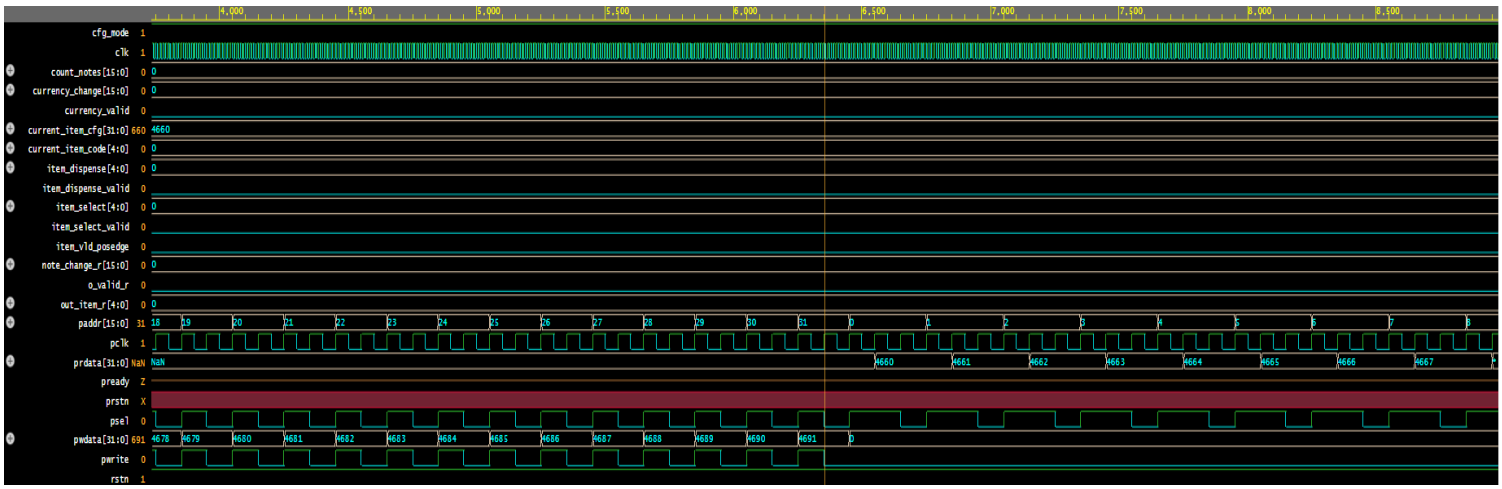
vending_basic_test



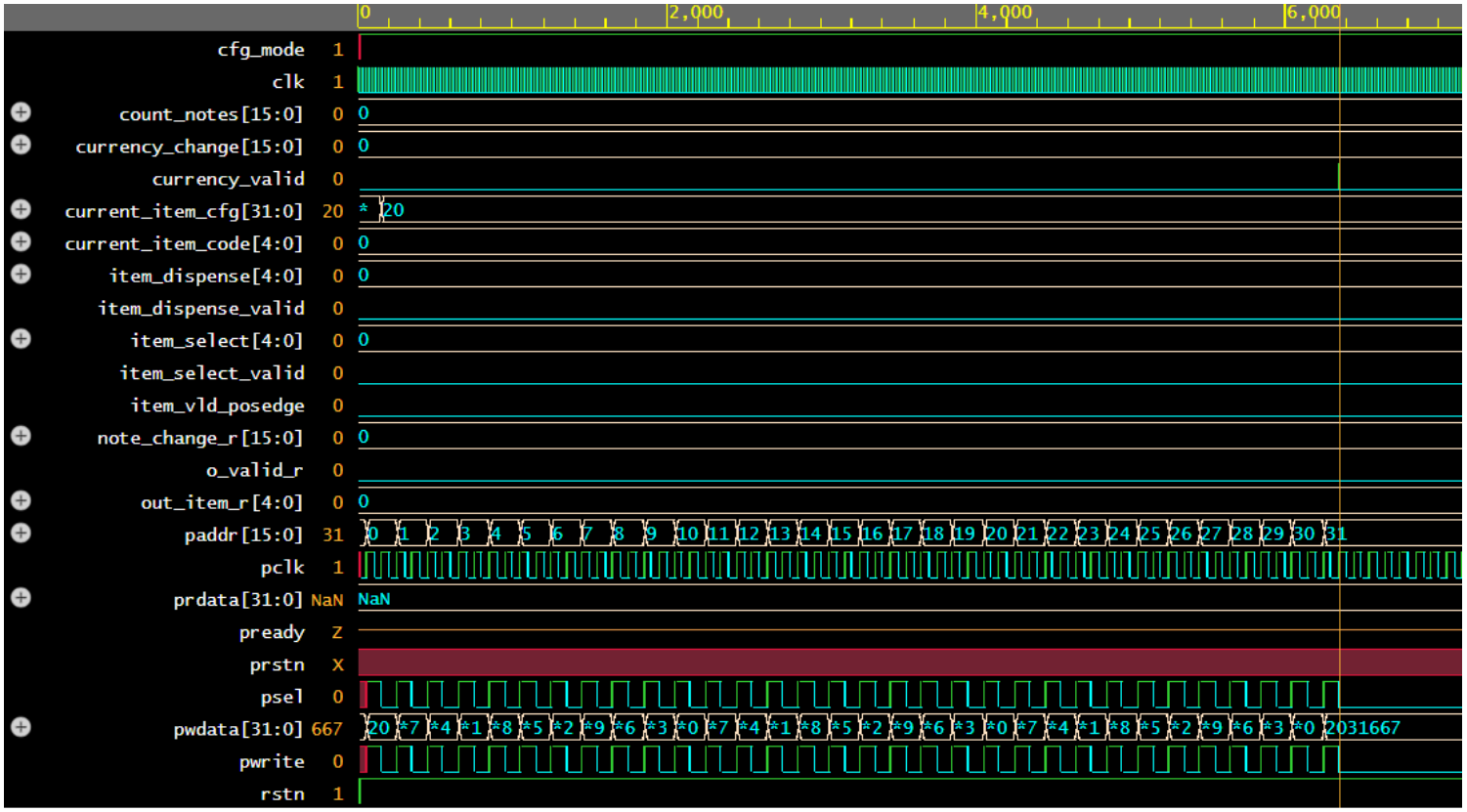
reset_mode_test



apb_write_read_test



invalid_item_test



5.EdaPlayground Link:

https://edaplayground.com/x/WDaL

Learning and Reflection

Learning:

Individual Learnings

- **Rohit Kumar (Dispense Agent)**
  - Understood the role of passive agents in monitoring DUT outputs.
  - Learned how to connect monitors to the scoreboard for validating dispense signals, item IDs, and change calculation.
  - Improved skills in integrating individual verification components into a complete UVM environment.
- **Umesh K. (Testbench & Integration, APB Agent, Scoreboard)**
  - Gained experience in building and integrating a UVM testbench environment.
  - Learned how to design APB agent sequences and verify register read/write functionality.
  - Strengthened debugging skills by checking scoreboard mismatches during concurrent user transactions.
- **Lavanya L. (Item Select Agent)**
  - Learned how to create and drive sequence items for item selection interface.
  - Developed understanding of boundary testing (valid/invalid item IDs).
  - Improved skills in writing randomized testcases for multiple-item transactions.
- **Prasanna M. (Currency Agent, Bug Finder)**
  - Gained experience in modeling asynchronous inputs and synchronizers in verification.
  - Learned how to simulate multiple currency insertions and validate FSM responses.
  - Strengthened problem-solving skills by identifying and reporting design bugs during negative test scenarios.

Team Learnings

1. **Understanding UVM Methodology**
   - Learned how modular UVM components (agents, drivers, sequencers, monitors, scoreboard) interact.
   - Understood the importance of reusable and layered verification architecture.
2. **Functional Coverage and Test Planning**
   - Realized the importance of writing diverse testcases (reset, configuration,

concurrency, invalid input, multiple currencies).
- Learned how coverage metrics guide completeness of verification.

3. **Debugging and Collaboration**
   - Gained experience in waveform analysis and tracing bugs across clock domains.
   - Improved teamwork by dividing roles across different agents and integrating results into a common testbench.
4. **Verification vs RTL Design**
   - Understood how verification complements RTL design by finding hidden bugs that designers might miss.
   - Learned that a robust verification environment is essential for ensuring real-world correctness.

Experience:

Working on the **verification of the Vending Machine Controller** was a valuable experience that strengthened my technical and problem-solving skills. I gained hands-on exposure to building a modular testbench with agents, monitors, and a scoreboard, and learned how to write and debug testcases for different scenarios like valid/invalid item selection, reset, and currency handling.

Using **EDA Playground** helped me practice quick simulations and waveform analysis in a collaborative way. Team collaboration taught me task division and integration of multiple verification components. Overall, this project improved my confidence in verification workflows and gave me practical insight into how theoretical concepts like FSM validation, clock domain crossing, and coverage are applied in real projects.

Conclusion and Future Scope

Conclusion:

The verification of the Vending Machine Controller was successfully carried out using a modular UVM-based testbench implemented on **EDA Playground**. The environment consisted of multiple agents (APB, currency, item select, and dispense), monitors, drivers, sequencers, and a centralized scoreboard. Each component was independently verified and then integrated to form a complete verification environment.

Extensive testcases were developed to validate different aspects of the design, such as reset functionality, configuration writes/reads, valid and invalid item selections, multiple currency insertions, concurrent transactions, and error scenarios. The scoreboard and waveform analysis confirmed correct DUT behavior under both normal and corner-case conditions.

Through this work, the project demonstrated the importance of a structured verification methodology in ensuring functional correctness, robustness, and reliability of RTL designs. The team gained hands-on exposure to verification planning, testcase design, debugging, and collaborative integration of UVM components.

Future Scope:

1. **Coverage-Driven Verification**
    - Extend the environment with functional and code coverage metrics to quantitatively measure verification completeness.
2. **UVM Enhancements**
    - Upgrade the environment to a complete UVM framework with factory overrides, configuration databases, and reusable sequences.
3. **Randomized and Constrained Testing**
    - Add randomized stimulus generation with constraints to improve corner-case exploration and bug detection.
4. **Advanced Features**

- Extend verification to cover additional features such as multiple simultaneous item dispenses, support for digital payments (QR, cards), and dynamic currency formats.

5. **Regression and Automation**
    - Integrate the verification flow into CI/CD pipelines for automated regression testing, enabling scalable verification for larger SoC designs.
6. **FPGA/ASIC Hardware Testing**
    - Move beyond simulation by deploying the RTL onto FPGA hardware and performing hardware-in-the-loop verification for real-world validation.