Lab 4B: Vending Machine Controller

Objective

The objective of this lab is to design and implement a **Vending Machine Controller** that accepts coins of **5¢, 10¢, and 25¢**, dispenses an item priced at **30¢**, returns change when necessary, and handles a **coin return request**. The design must operate synchronously with a clock and provide outputs to display the current balance and change returned.

Design Specifications

- **Inputs**:
    - `coin_5`, `coin_10`, `coin_25` → coin insert signals
    - `coin_return` → manual return request
    - `clk` → system clock
    - `rst_n` → active-low reset
- **Outputs**:
    - `dispense_item` → signal to dispense the product
    - `return_5`, `return_10`, `return_25` → signals to return coins as change
    - `amount_display` → shows the current balance in cents
- **Functional Requirements**:
    - Accept **5¢, 10¢, and 25¢ coins**.
    - Dispense the item when balance ≥ **30¢**.
    - Provide **change** if inserted amount exceeds 30¢.
    - Handle **coin return requests**.
    - Display the current balance on an LED display.

Design Methodology

1. **State Identification**
   The vending machine is modeled as a **finite state machine (FSM)** with the following states:
   - **IDLE**: Waiting for coins.
   - **ACCUMULATE**: Tracking inserted coins and updating balance.
   - **DISPENSE**: Item is dispensed when balance ≥ 30¢.
   - **RETURN_COINS**: Coins are returned either as change or due to coin return request.
2. **State Transitions**
   - From **IDLE** → Move to **ACCUMULATE** if a coin is inserted.
   - From **ACCUMULATE** →
     - If balance ≥ 30¢ → transition to **DISPENSE**.
     - If coin return is pressed → transition to **RETURN_COINS**.
   - From **DISPENSE** → If balance > 0 after purchase, return to **ACCUMULATE**; otherwise, go back to **IDLE**.
   - From **RETURN_COINS** → Sequentially return coins until balance = 0, then return to **IDLE**.
3. **Change-Making Logic**
   - If balance ≥ 25¢ → return a 25¢ coin.
   - Else if balance ≥ 10¢ → return a 10¢ coin.
   - Else if balance ≥ 5¢ → return a 5¢ coin.
   - Repeat until balance = 0.
4. **Error Handling**

- If invalid coin inputs or coin jams occur, the design ensures system stability by not updating balance incorrectly.
- The **coin return** feature ensures recovery from unexpected conditions.

Simulation and Verification

Test Scenarios

The design was simulated with the following cases:

1. **System Reset**
   - Machine resets to IDLE state with balance = 0.
2. **Exact Payment** (e.g., 10¢ + 25¢ → 35¢)
   - Item dispensed when balance reaches 30¢.
   - Extra 5¢ returned as change.
3. **Overpayment** (e.g., 25¢ + 10¢ = 35¢)
   - Item dispensed, and extra balance returned in coins.
4. **Coin Return Before Purchase**
   - User inserts coins but presses `coin_return` before reaching 30¢.
   - All inserted coins are returned appropriately.
5. **Continuous Purchases**
   - If balance > 30¢ after purchase, system dispenses item and deducts cost, allowing another item purchase without resetting balance.

Simulation Results

- The **LED display** showed the correct current balance throughout operation.
- The **dispense signal** was correctly asserted when the item was purchased.

- The **change logic** worked properly, returning correct coin denominations.
- The **coin return request** worked at any stage, refunding inserted coins.

Results and Discussion

- The FSM-based design successfully implemented the vending machine controller.
- Simulation confirmed proper handling of **exact payment, overpayment, and coin return requests**.
- The **change-making logic** ensured efficient coin returns using the highest possible denomination first.
- The system was robust against invalid inputs by ensuring synchronous updates.

Conclusion

The vending machine controller was designed, implemented, and tested using an FSM approach. It met all requirements, including coin acceptance, item dispensing, change return, and coin return handling. The design demonstrates how **finite state machines** can be effectively used to model real-world control systems.