

## Working of the EVM machine

### Overview of the EVM FSM Design

The given Verilog module `EVM_FSM` models an **Electronic Voting Machine using a Finite State Machine (FSM)**. The FSM controls the entire voting process, starting from election initialization to vote recording and closure. The design supports **four candidates**, uses **clocked operation**, and includes **confirmation and cancellation mechanisms** to ensure correct voting.

---

### Inputs and Outputs

The module operates on a clock (`clk`) and asynchronous reset (`reset`). Administrative controls include `start` to begin or reopen voting and `close` to end the election. The voter selects a candidate using a 2-bit input `vote` and confirms or cancels using `confirm` and `cancel`.

The outputs include:

- `ready`: indicates the machine is ready for a voter,
  - `locked`: indicates a vote is being recorded,
  - `led_state`: a 4-bit debug indicator showing the current FSM state,
  - `count1` to `count4`: 8-bit registers storing vote counts for each candidate.
- 

### FSM States

The FSM uses **seven states**:

1. **IDLE** – Waiting for the administrator to start the election
2. **READY** – Machine ready to accept a voter
3. **VOTING** – Candidate selected by voter
4. **CONFIRM** – Confirmation stage
5. **LOCKED** – Vote is recorded and locked
6. **THANKS** – Delay before next voter
7. **CLOSED** – Election closed

State encoding is done using 3-bit parameters.

---

## State Transition Mechanism

A **sequential always block** updates the current state on the rising edge of the clock or resets it to **IDLE** when **reset** is asserted.

A second **clocked always block** controls:

- next state transitions (**next\_state**)
- output signals
- vote counting
- delay timer

At the beginning of each clock cycle, default values are assigned to outputs to prevent unintended latches.

---

# Detailed State Operation

## 1. IDLE State

The system remains in **IDLE** until the admin asserts **start**. No voting is allowed in this state. Once **start** is detected, the FSM transitions to **READY**.

---

## 2. READY State

The machine signals readiness by asserting readiness. If **close** is activated, the FSM moves to **CLOSED**. If a non-zero **vote** input is detected, the selected candidate is stored in the **candidate** register and the FSM moves to **VOTING**.

---

## 3. VOTING State

The voter has selected a candidate but has not finalized the vote. Pressing **cancel** returns the FSM to **READY**, allowing reselection. Pressing **confirm** moves the FSM to the **CONFIRM** state.

---

## 4. CONFIRM State

This is a safety stage requiring final confirmation. A **cancel** returns the system to **READY**. A second confirmation moves the FSM to the **LOCKED** state.

---

## 5. LOCKED State

The vote is permanently recorded. Based on the stored **candidate** value, the corresponding vote counter (**count1** to **count4**) is incremented. The **locked** signal is asserted to indicate vote recording. The FSM then initializes a timer and moves to **THANKS**.

---

## **6. THANKS State**

A short delay is created using the `timer` variable to prevent immediate re-voting. After the timer exceeds a fixed count, the FSM returns to the `READY` state for the next voter.

---

## **7. CLOSED State**

When the election is closed, voting is disabled. The FSM remains in this state until `start` is asserted again, which reopens the election.