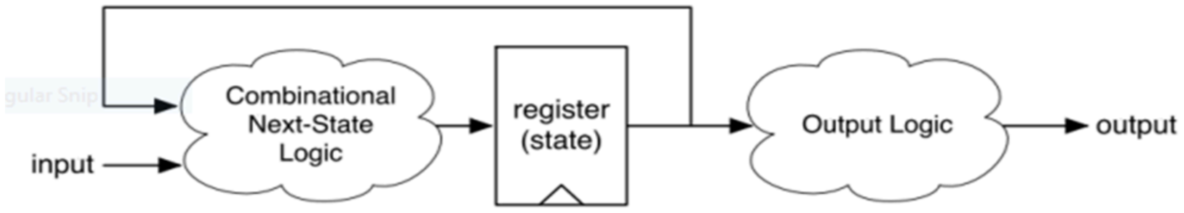
Section B:

HDL Fault Detection FSM

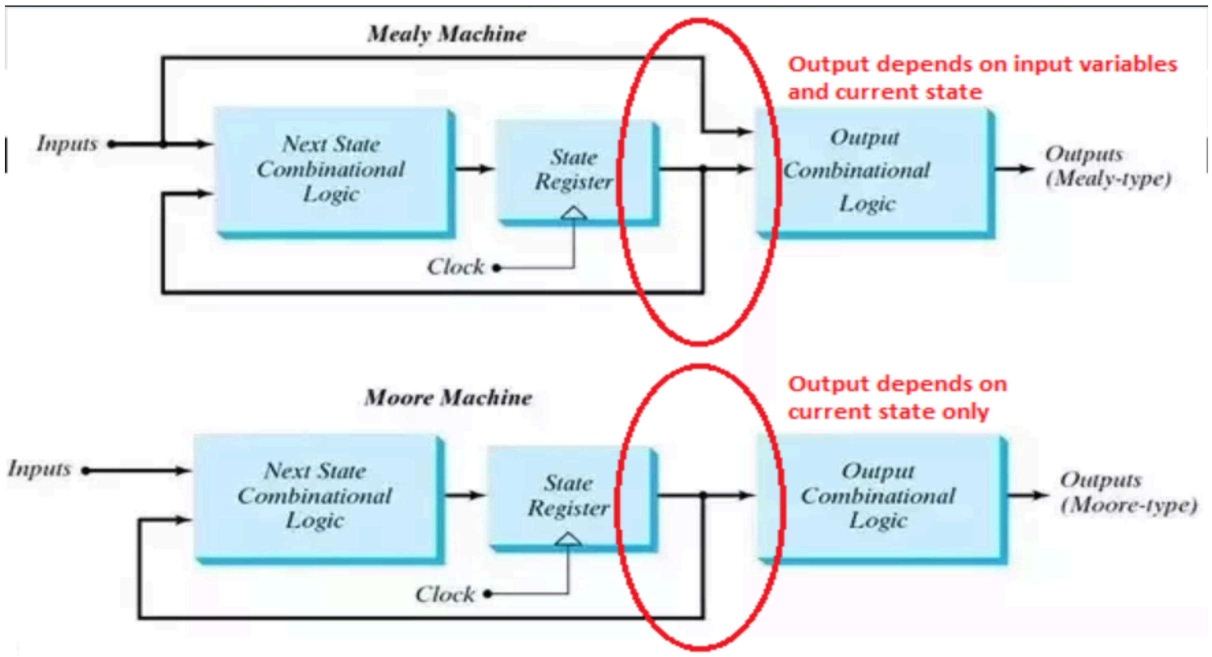
Varshini D V
USN:1MS22EC141
M S RAMAIAH INSTITUTE OF TECHNOLOGY

Abstract

A Finite State Machine (FSM) is a theoretical computer that can be used to design systems which change between a finite number of states depending on both inputs and internal logic. In other words, a synchronous sequential circuit is also called a Finite State Machine (FSM), if it has a finite number of states. FSMs are popular in digital design, which includes synchronous logic for state machines and asynchronous logic as microprograms.



There are two types of finite state machines namely,Mealy State Machine and Moore State Machine. A Moore FSM outputs are a function of current state only and such an architecture is more stable or predictable. Instead, a Mealy FSM produces outputs not only as result of the current state but also as consequence of an input, with faster and more responsive function.


Mealy Machine
Output depends on input variables and current state
Inputs
Next State Combinational Logic
State Register
Clock
Output Combinational Logic
Outputs (Mealy-type)
Moore Machine
Output depends on current state only
Inputs
Next State Combinational Logic
State Register
Clock
Output Combinational Logic
Outputs (Moore-type)

This project implements a **fault detection Finite State Machine (FSM)** in SystemVerilog using both Mealy and Moore models to ensure safe operation of a

battery monitoring system. The FSM monitors cell voltages, current, and temperature flags, transitioning through four states - **NORMAL**, **WARNING**, **FAULT**, and **SHUTDOWN.** The design features include debounce filtering to discard transient spikes, persistence checks to verify that faults remain constant over time, fault masking for excluding some fault checks and fault priority logic for prioritizing the shutdown or not based on criticality. This dual-model approach provides trade-offs between responsiveness and robustness in safety-critical digital systems.The state transitions in normal, transient, and permanent fault conditions are validated by simulation on EDA Playground, thereby verifying the FSM design both reliability-wise and safety-criticalness.

Tools and AI usage

- **EDA Playground:** It was utilized for simulation and formation of wave form.
- **Copilot:** Contributed to FSM design brainstorming, writing skeleton of initial code structure.
- **ChatGPT:** Helped me in understanding the idea on different terminologies used like debounce, persistence, fault priority, masking etc.
- **GTKWave:** To visualise signal transitions and verify the FSM operation.

Design and Methodology

The finite state machine (FSM) runs around critical battery data including cell voltages, current and temperature alert signals. Given these inputs, the FSM transitions between four distinct states: Normal, Warning, Fault and Shutdown. This approach involves fault filtering (debounce and persistence), masking, and prioritization.

Inputs:

- clk: Provides the clock signal. For every positive edge of the clk, the FSM transits to the next state from the current state, depending on the conditions met.
- reset: Trigger the FSM to a known state such as NORMAL. If reset=1, initiates reset.
- cell_voltage[4]: The voltage reading from 4 battery cells. It is used for detecting overvoltage and undervoltage faults.

- current: Provides the system current and to detect overcurrent faults.
- temp_flag[4]: It is the temperature flag(reading) per cell and also for detection of overtemperature faults.
- mask[3:0]: It is used to mask the fault for a particular cell.

Outputs:

- state[1:0]: Provides the current FSM state.

  00: Normal

  01:Warning

  10: Fault

  11: Shutdown

- shutdown_signal: It is set only when the FSM transitions into the Shutdown state. In mealy it depends on the fault as well(input) along with the state.

Faults types:

- Overcurrent
- Overtemperature
- Overvoltage
- Undervoltage
- Imbalance

FSM States:

1. **Normal state:**
- All readings observed are in safe operating range.
- The system is in steady state, and there are no fault-triggered actions.

2. **Warning State:**
- Entered when the fault condition has been detected, but it hasn't kept its state long enough to be declared critical.
- The stage acts as an ‘intermediate’ on noise or transient spikes.
- When the condition clears, FSM returns to Normal.

3. **Fault State:**
- When a fault condition is present for more than some defined debounce/persistence threshold (e.g. 5 cycles).

- Indicates a verified problem that needs to be addressed by the system.
- But the system stays on to be monitored, or attended.

4. **Shutdown State:**

- The-off-state, which is only reached in the event of a critical fault condition (e.g., severe overcurrent or thermal runaway).
- When the shutdown_signal is triggered, the higher level system (i.e., battery management or power electronics) can disconnect the load/charger for safety.

Design Features:

- **Debounce Logic:** It is a brief, single cycle anomalies (e.g., caused by sensor-noise or switching transients) shall not cause a state-transition. A debounce counter will not allow a fault to be considered valid until it is present for N cycles.
- **Persistence Check:** Warnings only become Faults if they persist beyond a short persistence threshold. This avoids an overreaction to brief transients.
- **Masking:** Some of these fault checks can be turned off with a mask input. For instance, if undervoltage is anticipated during deep discharge testing then this can be masked to prevent false cutoffs.
- **Fault Priority:** When simultaneous fault causes appear, the FSM uses a hierarchy that determine which must dominate:
    1.Overcurrent (highest priority, immediate shutdown)
    2.Overtemperature
    3.Overvoltage
    4. Undervoltage
    5.Imbalance (lowest priority, tolerable for a while)

FSM models:

This project implements two types of finite state machines (FSMs): **Mealy** and **Moore**, each with distinct output behavior.

- **Moore FSM:** In a Moore machine, output is a function only of the present state. This makes the system more stable and predictable, because outputs change only when FSM moves from one state to another. In this design the shutdown_signal is active only when the FSM goes to SHUTDOWN not because inputs changed.
- **Mealy FSM:** In the Mealy model, outputs are determined by the current state and the inputs. This model allows for a more immediate reaction to a critical state. For instance, because the fault score is above the threshold, in the FAULT

state, we can assert shutdown_signal immediately instead of waiting to move states to shut down in the FAULT state.

FSM Diagram:

1. **Normal → Warning**
   Condition: A fault is texted for the first time.
   Example: Cell voltage has crossed the undervoltage limit for 1 cycle.
   Reason: We do not issue a fault right away because it could just be a glitch (electrical noise, etc.). We transition the system to a Warning state.
   Purpose: The condition gives the system a chance to "double-check" to ensure that the condition is real.
2. **Warning → Normal**
   Condition: The fault condition very soon disappears within the debounce period.
   Example: The voltage fell below threshold for 2 cycles, but then rose again.
   Action: The FSM terminates the warning and is able to operate normally and safely.
   Purpose: This avoids unneeded alarm/shutdown actions from non-risky momentary conditions.
3. **Warning → Fault**
   Condition: The fault persists longer than a persistence threshold (example: 5 cycles).
   Example: The voltage is overvoltage, the current is over-limit, the temperature is above threshold, etc.
   Action: The FSM escalates to Fault status because the problem is now confirmed.
   Purpose: Only to classify faults compared to risk-free / short-lived conditions.
4. **Fault → Shutdown**
   Condition: A serious fault has occurred or is persistent long enough that operating the system is unsafe.
   Examples: An overcurrent exists → shutdown immediately.
   A severe overtemperature exists → shutdown.
   Action: The FSM sets the shutdown_signal to engage the user and remove power from the battery or halt operations, most likely both.
   Purpose: Protects the hardware and helps the users from catastrophic injury.
5. **Fault → Fault (Remain in Fault)**
   Condition: Fault exists, but the fault detected is not serious enough to require shutdown.
   Example: Slight overtemperature fault detected (does not pass absolute limits/ is within safe operation).

Action: The FSM will stay in a fault state and continue to monitor the system in its current running state.
Purpose: Gives the system an opportunity to mitigate the fault (ex: cool) without going completely offline.

6. **Shutdown → (Terminal State)**
Condition: Once shut down. The FSM will not reboot to a running operational state.
Example: Overcurrent
Reason: Safety-critical systems must require a manual reset in order to reboot for safety. Users of the system should be trained to reboot the system in safe operational conditions.
Purpose: A fail-safe hard-coded behavior.

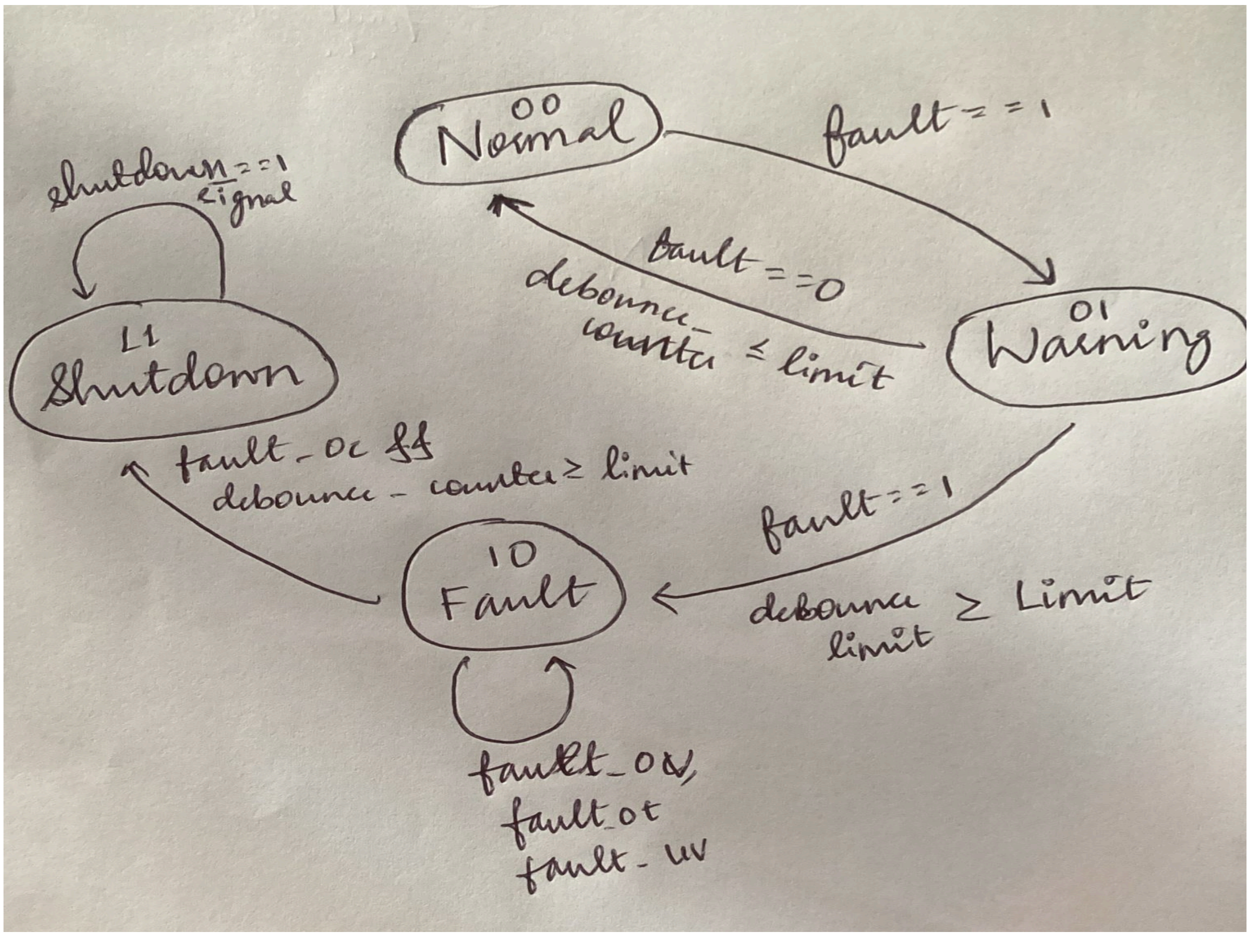


Fig: FSM state diagram

Implementation

Language: SystemVerilog

FSM Models used: Mealy and Moore

Platform: EDA Playground

The FSM was implemented using a synchronous state-machine design pattern. The transitions between states of the state machine will solely depend on clocked **always** blocks for that timing predictability and for avoiding glitches when transitioning states.

FSM Module:

- **Mealy Module**: fault_fsm_mealy;
    1. **State updates:** The FSM always utilizes the always_ff blocks in regards to clock (clk) and reset (reset) rising edges for synchronous state transitions and debounce counter updates.
    2. **Fault Scoring Logic:** One always_comb block checks for faults, for all the active (unmasked) cells. Each fault type (overvoltage, undervoltage, overtemperature, imbalance, and overcurrent) has a corresponding configurable priority weight. An overall fault score is determined in real-time, and the fault score is a determining factor in shutdown eligibility.
    3. **Shutdown Signal:** The shutdown_signal is asserted immediately in the FAULT state if the fault score is above the shutdown_threshold we have defined. The Mealy-style output makes sure we don't have to be in the SHUTDOWN state to respond as quickly to severe faults like overcurrent.

- **Moore Module:** fault_fsm_moore;
    1. **State updates:** Similar to the Mealy model, the FSM shifts states and manages debouncing logic in always_ff blocks in response to clock and reset.
    2. **Fault Scoring Logic:** An always_comb block checks fault conditions over unmasked cells, and with weighted fault types calculates a cumulative fault score. This score is used to determine if the FSM transitions to the SHUTDOWN state.

3. **Shutdown Signal:** The shutdown_signal is asserted only when the FSM transitions to the SHUTDOWN state. This output is a Moore-style output, controlled and driven solely by the state and does not change with transitory causes in the input.

Debounce and Persistence Counter:

- A debounce counter has been added for each fault type being monitored, voltage, current, and temperature.
  If a fault flag was active for one or two clock cycles, the debounce counter would have already reset, so the state will not transit.
- The persistence counter will ensure that faults that persist will escalate from the Warning state to a Fault state.
  As an example, the fault state must persist for N = 5 clock cycles for the state to change to Fault.
- This allows for a little false positive due to electrical noise, measurement error/miscalibration, and switching chatter/transient.

Testbench Implementation:

Mealy Module: tb_fault_fsm_mealy;
Moore Module: tb_fault_fsm_moore;

Scenarios:

1. **Normal Case (No Faults):**
   - Setup: The **set_normal()** task applies safe operating conditions.
   - Expected Behavior: The FSM should stay in the Normal state. shutdown_signal = 0.
   - Purpose: To confirm if it can work stably under ideal circumstances and has been not falsely triggered.

2. **Transient Fault Case:**
   - Setup: The **set_transient_spike()** causes a brief increase or decrease in the value for one cycle.
   - Expected Behavior: The FSM should transition Normal → Warning → Normal after the spike clears. No shutdown occurs.
   - Purpose: To validate the debounce logic, which should prevent short transient noise from causing permanent state changes.

3. **Persistent Fault Case**
   - Setup: The **set_persistent_fault()** injects multiple faults simultaneously.

- Expected Behavior: The FSM transitions from Normal to Warning to Fault after persistence check. Overcurrent being the highest priority causes the FSM to upscale further to Shutdown. shutdown_signal = 1 in Shutdown.
- Purpose: To demonstrate the correct fault escalation, its persistence check, and shutting down under extreme fault conditions.

4. **Imbalance Fault**
   - Setup: The **set_imbalance_fault()** establishes the imbalance.
   - Expected Behavior: FSM should detect the imbalance → transition to Warning (if the fault has persisted past the warning threshold). Depending on fault aggravation, it may optionally escalate to Fault, never to override any critical faults.
   - Purpose: Validates imbalance detection which hinders long-term pack degradation.

5. **Masking Test**
   - Setup**:** The **set_masked_fault()** induces a fault and masks it.
   - Expected Behavior: FSM should remain Normal even in the presence of a fault. shutdown_signal = 0.
   - Purpose: Confirms fault masking, so that during special tests, it is possible to productively disable checks selectively.

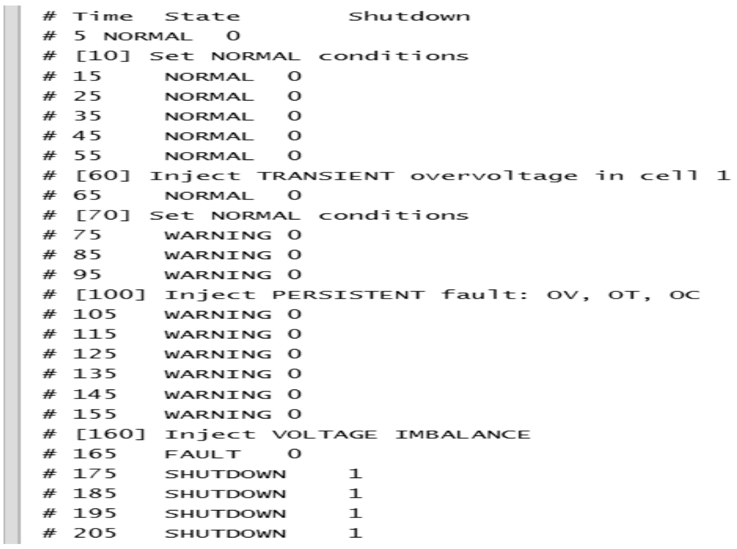
6. **Fault Priority Test**
   - Setup: The **set_persistent_fault()** injects multiple simultaneous faults (overvoltage, overtemperature, and overcurrent).
   - Expected Behavior: FSM should consider all faults and apply priority rules: Overcurrent is considered highest priority → FSM immediately transitions to Shutdown even if other faults exist.
   - Purpose: Ensures fault prioritization is adhered to so that less urgent faults do not stand in the way of an immediate shutdown.

Waveform Output:
The testbench generates a wave.vcd file containing all relevant signal transitions. GTKWave is used to visualize:

- FSM state progression (state, next_state)
- Fault signals (fault_oc, fault_ot, etc.)
- Debounce counter behavior
- Input changes (current, cell_voltage, temp_flag)
- Assertion timing of shutdown_signal

Result

Moore model:



```
# Time   State        Shutdown
# 5 NORMAL   0
# [10] Set NORMAL conditions
# 15     NORMAL   0
# 25     NORMAL   0
# 35     NORMAL   0
# 45     NORMAL   0
# 55     NORMAL   0
# [60] Inject TRANSIENT overvoltage in cell 1
# 65     NORMAL   0
# [70] Set NORMAL conditions
# 75     WARNING 0
# 85     WARNING 0
# 95     WARNING 0
# [100] Inject PERSISTENT fault: OV, OT, OC
# 105    WARNING 0
# 115    WARNING 0
# 125    WARNING 0
# 135    WARNING 0
# 145    WARNING 0
# 155    WARNING 0
# [160] Inject VOLTAGE IMBALANCE
# 165    FAULT    0
# 175    SHUTDOWN     1
# 185    SHUTDOWN     1
# 195    SHUTDOWN     1
# 205    SHUTDOWN     1
```

Fig: Command section of Moore model

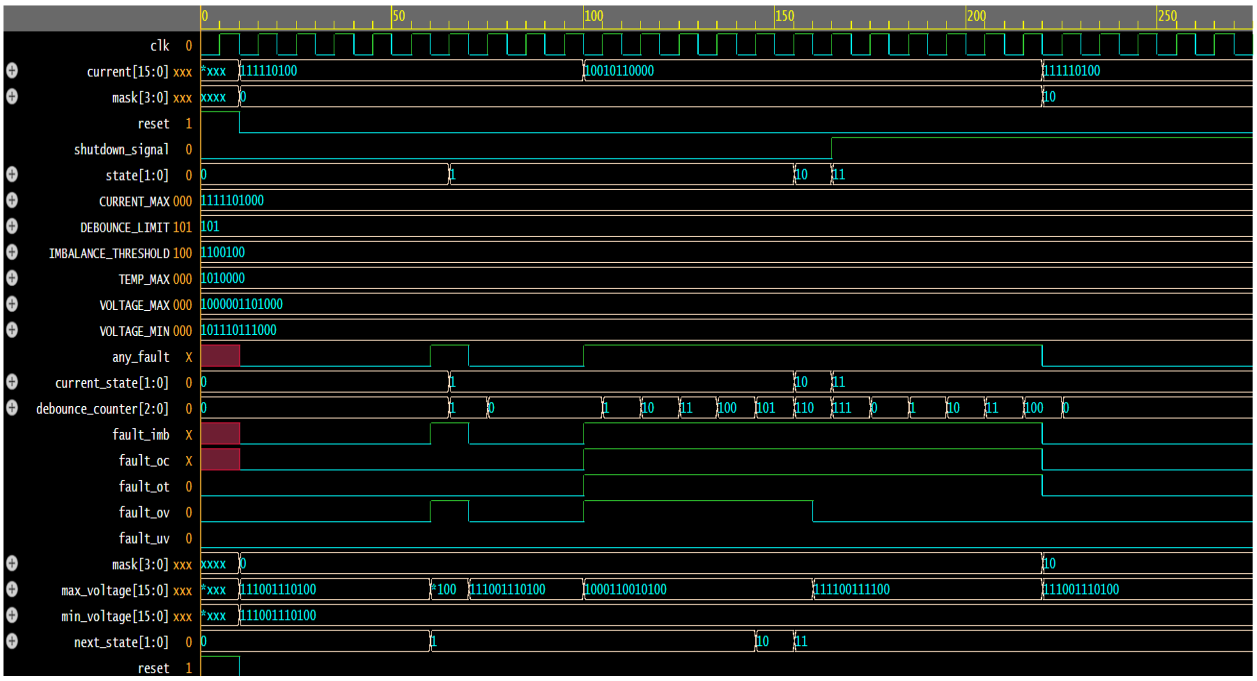
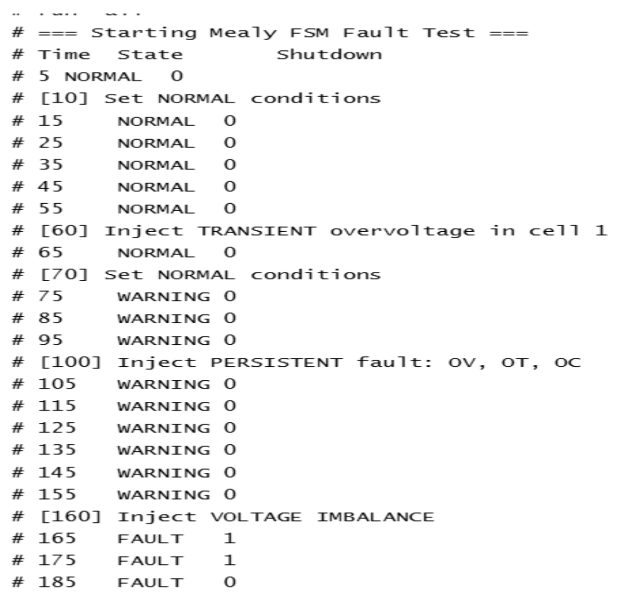


Fig: Waveform generated by Moore model

Mealy Model:



```
# === Starting Mealy FSM Fault Test ===
# Time  State        Shutdown
# 5 NORMAL   0
# [10] Set NORMAL conditions
# 15     NORMAL   0
# 25     NORMAL   0
# 35     NORMAL   0
# 45     NORMAL   0
# 55     NORMAL   0
# [60] Inject TRANSIENT overvoltage in cell 1
# 65     NORMAL   0
# [70] Set NORMAL conditions
# 75     WARNING 0
# 85     WARNING 0
# 95     WARNING 0
# [100] Inject PERSISTENT fault: OV, OT, OC
# 105    WARNING 0
# 115    WARNING 0
# 125    WARNING 0
# 135    WARNING 0
# 145    WARNING 0
# 155    WARNING 0
# [160] Inject VOLTAGE IMBALANCE
# 165    FAULT    1
# 175    FAULT    1
# 185    FAULT    0
```

Fig: Command section of Mealy model

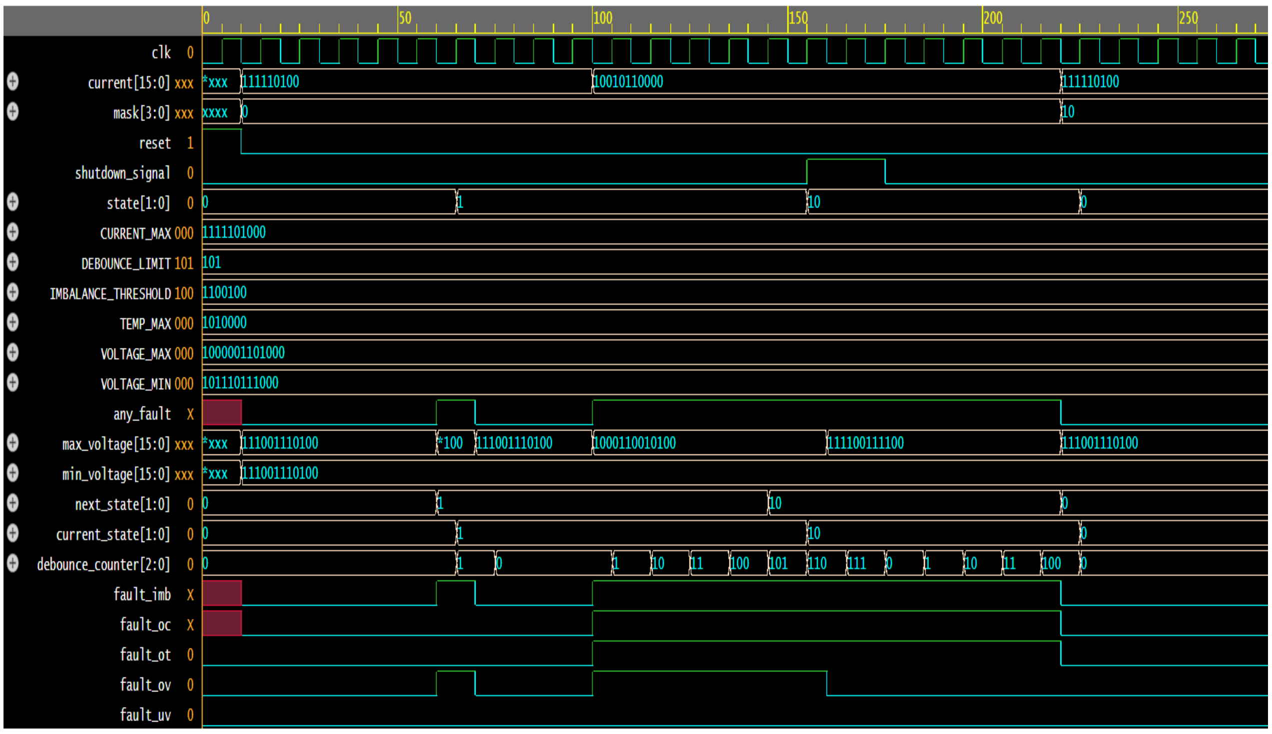


Fig:Waveform generated by Mealy Model

Following outcomes were observed from the project:

- FSM transitions correctly through all states.
- shutdown_signal asserted only when fault score exceeds threshold.
- Masking successfully excludes selected cells from fault checks.
- Waveforms confirm debounce behavior and fault prioritization.

Comparison between Mealy and Moore Models:

Features	Mealy model	Moore model
shutdown_signal	Depends on state + inputs	Depends only on state
FAULT State	shutdown_signal drops immediately	FSM remains in SHUTDOWN until reset or recovery logic
Timing	Immediate	Delayed
Used in	Real-time fault response	Safety-critical, stable control
Persistent Fault	FSM transits to FAULT, then shutdown_signal remains low	FSM escalates to FAULT, no transition to SHUTDOWN

Challenges and Limitations

Challenges:

- Fault response must be fast (Mealy) while output behavior must be stable (Moore) and also had to make sure that the fault-scoring and debounce logic remained consistent in both models.
- Debounce and persistence thresholds needed to be finely tuned to distinguish between short transients and lasting conditions, avoiding false alarms and missed detections.
- Interpretation from GTKWave was difficult as there were clutters due to multiple signals in EDA Playground. Signal grouping and standard naming were necessary for a pointed interpretation of results.

- Fault priority handling becomes a great challenge to design and verify correctly when more than one faults are prioritized together.

Limitations:

- The FSM has only been tested with 4 cells. A 100+ cell-scale (as is typical in EV packs) would necessitate some optimization, hierarchical fault aggregation, and efficient masking logic.
- The FSM would only recover from Shutdown via a manual reset. While this is safe, it detracts from system autonomy. It would be preferable to have a controlled auto-recovery method.
- The design has only been simulated in ideal conditions. Real hardware latency, ADC resolution, and clock-domain crossing may affect debounce/persistence timing accuracy.
- There is no retention of fault history and frequency of fault occurred, which could be very useful for applications such as predictive maintenance or severity escalation.

Conclusion

HDL-implemented fault detection FSM was designed, implemented, and successfully verified through simulation. It also compares two FSM models-Moore and Mealy-for fault detection purposes in battery systems. Mealy FSM supports speedy exit-to-input response path and so immediately assert shutdown signal once critical fault like overcurrent occurs; the Moore FSM provides a stateful-mode control that only asserts shutdown upon entering the SHUTDOWN state.

The system reliably classifies normal operation, transient anomalies, faults, and shutdowns, guaranteeing silence on short glitches in conjunction with debounce filtering and elevating only persistent events to fault status via persistence counters. Fault masking can be fine-tuned during the test phase, while a fault-priority scheme guarantees that the most hazardous conditions like overcurrent are promptly addressed.

Henceforth, the project confirms that real-time battery fault monitoring is feasible by applying a compact FSM solution and can be configured for larger packs and practical BMS integration. Some future directions may include auto-recovery mechanisms, advanced thermal modeling, and FPGA/ASIC prototyping to study timing effects on real hardware.

References

1. EDA Playground: https://edaplayground.com/
2. Engineersgarage:https://www.engineersgarage.com/melay-machine-fsm-vhdl-code/
3. Tutorialspoint:https://www.tutorialspoint.com/digital-electronics/digital-electronics-finite-state-machines.htm

Appendix

Github log: https://github.com/Varshini031004/arys-hdlfsm-varshinidv.git

Run Instructions:

- Click on the link given below.
- Choose Siemens Questa from tools and simulation
- Click on Open EPWave after run

Mealy FSM Model: https://edaplayground.com/x/cQLu

Moore FSM Model: https://edaplayground.com/x/jNPU