Internship Project Plan: Joint SoC and SoH Estimation Architecture on FPGA

Project Title: Joint SoC and SoH Estimation Architecture on FPGA using Extended Kalman Filter and Coulomb Counting for BMS.

1. Abstract

The natural aging process of batteries leads to capacity degradation and internal resistance deviation, which significantly reduces the accuracy of traditional State-of-Charge (SoC) estimation models. This project proposes a unified FPGA-based VLSI architecture for the joint estimation of State-of-Charge (SoC) and State-of-Health (SoH). The SoC estimation utilizes a triple-hybrid methodology: an Extended Kalman Filter (EKF) for recursive error correction, Coulomb Counting (CC) for real-time tracking, and Open Circuit Voltage (OCV) for periodic calibration. Concurrently, the SoH is evaluated by monitoring capacity fade and internal resistance alterations. Integrating both estimators on a single FPGA fabric enables real-time parameter updates, allowing the SoH block to dynamically adjust the SoC algorithm as the battery ages. The system is developed in MATLAB/Simulink and implemented in Verilog HDL using fixed-point arithmetic, optimized for low-latency automotive BMS applications.

2. State of the Art & Gap Analysis

2.1 Literature Review

Recent advancements in Battery Management Systems (BMS) have shifted from simple microcontroller-based Coulomb Counting to complex FPGA-based estimators. A review of current research reveals three dominant architectures:

1. **Dual Extended Kalman Filter (DEKF):** Researchers have proposed using two separate Kalman Filters running in parallel—one for SoC and one for battery parameters (SoH). While accurate, this approach is computationally expensive, often consuming significant DSP resources on mid-range FPGAs.
2. **Neural Network / AI Estimators:** Recent studies explore implementing Bi-LSTM or Neural Networks on FPGA. While these offer high accuracy without a physical model, they suffer from "black-box" unpredictability and require massive training datasets, making them less suitable for strict automotive safety standards (ISO 26262).
3. **Static EKF Implementations:** Many standard industry solutions use a single EKF with fixed battery parameters (Resistance, Capacity). These work well for fresh batteries but fail as the battery ages, leading to drift errors >5% after partial cycling.

2.2 Comparative Analysis

The table below compares existing architectures against the solution proposed in this project.

Feature	Standard Coulomb Counting	Dual EKF (State of the Art)	AI/Neural Network on FPGA	Proposed Joint Architecture
Methodology	Current Integration	Two parallel Kalman Filters	Data-driven (Black Box)	**EKF + SoH Feedback Loop**
Computational Cost	Very Low	High (2x Matrix Inversion)	Very High (MAC Operations)	**Moderate (1x Matrix Inv + Feedback)**
Aging Adaptation	None (Fails over time)	Excellent	Good (if trained well)	**Excellent (Real-time updates)**
Resource Usage	<1% FPGA Fabric	High DSP Usage	High BRAM Usage	**Optimized (Shared Resources)**
Determinism	High	High	Low (Unpredictable)	**High (Safety Critical)**

2.3 Innovation: The Universal Feedback Architecture

Most low-cost FPGA implementations assume a specific battery chemistry (typically Li-ion) and a "fresh" state of health. This project bridges the gap by implementing a **"Universal Feedback Architecture."** Instead of hard-coding battery parameters, we separate the estimation engine (math) from the battery personality (data). A single, fast EKF runs generic matrix operations, while a lightweight SoH estimator dynamically updates the model parameters (Resistance, Capacity) in real-time. Additionally, a reconfigurable "Efficiency Factor" (η) allows the same core to adapt to high-efficiency Li-ion or lower-efficiency Lead-Acid batteries. This ensures the system remains accurate across **multiple chemistries** and throughout the battery's lifecycle without the hardware penalty of rewriting code.

3. System-Level Objectives

3.1 Core Problem Statement

Standard Coulomb Counting relies on fixed parameters and chemistry assumptions. However:

1. **Chemistries differ:** Li-ion has a flat voltage curve and high efficiency; Lead-Acid has a sloped curve and significant charging loss (Peukert effect).

2. Aging Drifts: Capacity fades and internal resistance increases over time.
   Without updating these parameters or adapting to the specific chemistry, SoC estimation drifts, leading to critical errors in universal BMS applications.

3.2 Solution Approach

A **Universal Joint Estimation Framework** is required where:

1. **SoC** is estimated using a generic Extended Kalman Filter (EKF) that fetches parameters from a selectable memory block.
2. **SoH** (Capacity and Resistance) is estimated to update the specific battery model currently in use.
3. **FPGA Implementation** ensures deterministic, low-latency processing (<10ms) via parallel hardware acceleration and chemistry-agnostic fixed-point arithmetic.

3.3 Expected Outcomes

- **Universal Simulink Model:** A "Twin-Plant" Golden Reference verifying algorithm convergence for both Li-Ion and Lead-Acid models.
- **RTL Design:** Synthesizable, parameter-driven Verilog code for EKF and SoH blocks.
- **Accuracy:** SoC estimation error < 3% under dynamic load profiles across different chemistries.
- **Hardware Metrics:** Resource utilization report (LUTs, DSPs) verifying fit on standard FPGAs.

4. System Architecture

The design follows a top-down hierarchical approach:

1. Input Interface Captures Battery Voltage (V_batt), Load Current (I_load), Temperature, and a **Chemistry Selector**.

2. Adaptive Coulomb Counter (CC) Performs high-frequency integration with programmable efficiency.

- **Equation:** $SoC_{cc} = SoC_{prev} + \left(\int (\eta * I)\, dt \right) / C_{total}$
- *Note:* η is variable (e.g., 0.85 for Pb-Acid charge, 0.99 for Li-Ion).

3. Battery Model (Predictor) A generic digital twin (1-RC Thevenin Model) predicting terminal voltage.

- **OCV Lookup:** Retrieved from a **Multi-Bank ROM** based on the selected chemistry.

4. EKF Engine (Corrector)

- Compares Measured Voltage vs. Predicted Voltage.
- Computes Kalman Gain (K) using generic matrix math to correct the SoC estimate.

5. SoH Estimator (Feedback Loop)

- **Resistance SoH:** Calculated via voltage drop analysis. $V_{drop} = I * R_{internal}$
- **Capacity SoH:** Calculated via charge throughput vs. voltage change. $Capacity = \Delta Q / \Delta V$
- *Note:* The SoH block updates the R0 and C_total parameters used by the CC and EKF.

---

5. Modeling Strategy (MATLAB / Simulink)

Before hardware implementation, a **"Universal Golden Reference Model"** will be created.

5.1 Plant Modeling

- **Strategy:** "Twin-Plant" Architecture.
- **Models:** Two distinct 1-RC Thevenin models:
    1. **Plant A (Li-Ion):** Flat OCV curve, low resistance.
    2. **Plant B (Pb-Acid):** Sloped OCV curve, higher resistance.
- **Model Equation:** Vt = Vocv(SoC) - V_polarization - (I * R0)
- **Validation:** The estimator will be tested against distinct drive cycles (DST for Li-Ion, Discharge/Rest for Pb-Acid) to ensure robustness.

5.2 Algorithm Development

- **EKF Logic:** Implemented as a generic MATLAB function block that accepts "Model Parameters" as inputs.
- **Floating-Point Validation:** Verify that the estimated SoC converges to the true SoC for *both* plants when parameters are swapped.

---

6. Hardware Design Strategy

6.1 Fixed-Point Conversion

FPGAs require fixed-point arithmetic for efficiency.

- **Format: Q8.8** (Range 0-255V) to support higher voltage Lead-Acid modules while maintaining precision for Li-Ion cells.
- **Scaling:** All constants will be scaled to fit this unified format.
- **Quantization Analysis:** MATLAB fi() objects will be used to ensure the precision is sufficient for the flat region of Li-Ion curves.

6.2 Verilog HDL Implementation Plan

- **Top Module:** universal_bms_top.v
- **Submodules:**
    - lut_manager.v: **(New)** Multi-bank ROM containing OCV curves for Li-Ion and Pb-Acid, with a selector mux.
    - matrix_math.v: **(New)** Optimized algebraic expansion for generic matrix operations.
    - adaptive_coulomb_cnt.v: Accumulator with efficient factor (η) logic inputs.
    - divider_core.v: CORDIC or restoring division logic.
    - fsm_controller.v: State machine managing the "Predict-Measure-Update" sequence.

---

7. Verification & Integration Flow

7.1 Test Vector Generation

- **Stimulus:** MATLAB will generate two sets of files: stim_li_ion.hex and stim_pb_acid.hex.
- **Golden Output:** MATLAB will generate corresponding golden_results_li.txt and golden_results_pb.txt.

7.2 Co-Simulation

1. Verilog testbench reads the stimulus and the **Chemistry ID**.
2. DUT (Device Under Test) reconfigures its parameters.
3. Data is fed cycle-by-cycle.
4. DUT output is logged and compared to the specific golden reference to calculate RMSE for each scenario.

---

9. Final Deliverables

- **MATLAB/Simulink Files:** Twin-Plant models and generic scripts.
- **RTL Source Code:** Parametric, well-commented Verilog files.
- **Verification Environment:** Testbenches capable of "Hot-Swapping" battery profiles.
- **Final Project Report:** Detailed documentation including block diagrams, adaptability analysis, and synthesis reports.
- **Performance Analysis:** Error metrics (RMSE) and FPGA resource utilization table.