

Performance Evaluation of an FPGA-Based Intelligent Battery Management System on Zynq-7000

Kuldeep Singh¹, Vinod Kumar¹, Ajay Kumar¹, Manoj², Prem Kumar³

¹Department of Electrical and Electronics Engineering,

Guru Jambheshwar University of Science and Technology, Hisar 125001, India

²Department of Computer Science and Engineering,

Guru Jambheshwar University of Science and Technology, Hisar 125001, India

³Dept. of Electronics and Communication Engineering, JB Institute of Technology, Dehradun, UK, India, 248197

kuldeepsingh@gjust.org; Vinodspec@yahoo.co.in; ajaypunia506@gmail.com; manojkoslia91@gmail.com; premhawal@gmail.com

Abstract—A battery management system (BMS) estimator has been designed and analyzed on the Avnet ZedBoard (Zynq-7000 XC7Z020-CLG484-1). The starting point was a neuro-fuzzy and regression method, which we proceed into fixed-point programmable logic. The evolved design has three parts: a 5–11–11–1 artificial neural network (ANN) that approximates specific gravity, a 5×7 fuzzy rule engine that classed as state of charge (SoC), and a Q16.16 multivariate regression datapath that estimates used and expected life. Verification is split into four levels so that floating-point model behavior, quantization effects, RTL behavior, and on-board execution can be scrutinized disparately. We then instrumented the design in Vivado 2019.2 at target clock frequencies between 15 and 100 MHz. At 100 MHz the worst negative slack (WNS) is –9.564 ns, which puts the critical-path delay at about 19.564 ns, nearly twice the 10-ns period. At 50 MHz the WNS shrinks to –0.073 ns, still a failure, and the first positive margin appears at 45.455 MHz (WNS = 0.475 ns). Total on-chip power falls only slightly across the sweep, from 1.698 W at 100 MHz to 1.671 W at 15 MHz, and the DSP, flip-flop, and LUTRAM counts do not change. For the present non pipelined design, roughly 45 MHz is therefore the highest frequency at which timing closure has been shown perfect; going faster will need the critical path to be restructured or pipelined.

Index Terms—Battery management system, FPGA, Zynq-7000, ZedBoard, state of charge, state of health, artificial neural network, fuzzy logic, fixed-point arithmetic, static timing analysis.

I. INTRODUCTION

Rechargeable batteries rest at the core of electric vehicles, renewable-energy storage, uninterruptible power supplies, and most portable electronics. Yet the quantities which a user actually desires about, such as state of charge (SoC) and state of health (SoH), cannot be read directly from the terminals. They have to be worked out from voltage, current, temperature, impedance-related measurements, the cell's operating history, or some mix of these. That inference is not easy. Batteries behave nonlinearly, respond strongly to temperature, age over time, and diverged by chemistry and load profile, and the sensors themselves add uncertainty. This is why a practical BMS is more than a single measurement channel: it packages of data acquisition, state estimation, protection, thermal supervision, cell balancing, communication, and diagnostics.

Several families of methods exist for estimating battery state. The coulomb counting is an economic method to measure, however the current-sensor offsets and the unpredicted initial value cause its error to elevate with time. Second, the open-circuit-voltage methods to measure is good approach once the cell has been installed or placed, that is very big issue in the case of dynamic loads. Third the equivalent-circuit and electrochemical simulations, when paired with a Kalman filter or an observer in the model, provides estimates with a transparent significance, but it costly for parameter identification and, a highly resource demanded load. Recent surveys sites to a steady shift toward machine-learning and deep-learning estimators for SoC and SoH, since a data-driven model can capture nonlinear behavior without solving the full electrochemical problem [2]–[5]. These models bring their own difficulties, though. The training data may not

represent field conditions, training is not always reproducible, extrapolation is risky, and the models can be expensive to run on small embedded processors.

When latency must be predictable and the arithmetic can be parallelized, dedicated hardware is an attractive option. Kumar, Khare, and Chaturvedi built an FPGA-based BMS that estimated SoC and SoH with neuro-fuzzy and statistical techniques [1]. Their work exhibited that battery-state estimation can be mapped onto reconfigurable logic, and it reported high clock rates on a different and much larger device. Others have followed a similar direction. VLSI and FPGA designs for multicell SoC/SoH estimation have been reported, including architectures that cut algorithmic complexity and were surveyed in real time on FPGA test beds [6]. FPGA versions of model-based estimators, for example the extended Kalman filter, have also been published [7], so interest in deterministic hardware execution is clearly still there.

In this paper, the contributions are multiforld. (i) We define a fixed-point architecture for Zynq that brings together a sequential 5–11–11–1 ANN, a 35-rule fuzzy SoC engine suited to hardware, and a Q16.16 regression datapath. (ii) We report a controlled frequency sweep from 15 to 100 MHz on the XC7Z020 using Vivado 2019.2, and use it to examine WNS, critical-path delay, estimated Fmax, power, and resource use.

II. BMS ESTIMATION MODEL

Five inputs, either normalized or derived, that drives the estimator. It takes in the consumption time, current drawn, terminal voltage, temperature, and internal resistance. Then model is feed a multilayer ANN with two hidden layers of eleven neurons each and a single output neuron. For the lead-acid case study considered here, that output is the specific gravity (SG), or a normalized form of it.

The SG evaluated by the ANN is feed to the fuzzy stage with normalized temperature. The specific gravity (SG) is completed in 7-regions and the temperature in 5-regions, which provide a 35 rule base combinations. We give the processing system (PS) the tasks of acquisition, calibration, filtering, logging, communication, and parameter management. The ANN, fuzzy, and regression computations, which need deterministic timing, run in the programmable logic (PL). AXI4-Lite is the natural choice for control and register access between the two sides. Splitting the system this way also keeps the wide internal arithmetic buses of the BMS off the physical FPGA pins, and it lets software drive the start, status, and data exchanges.

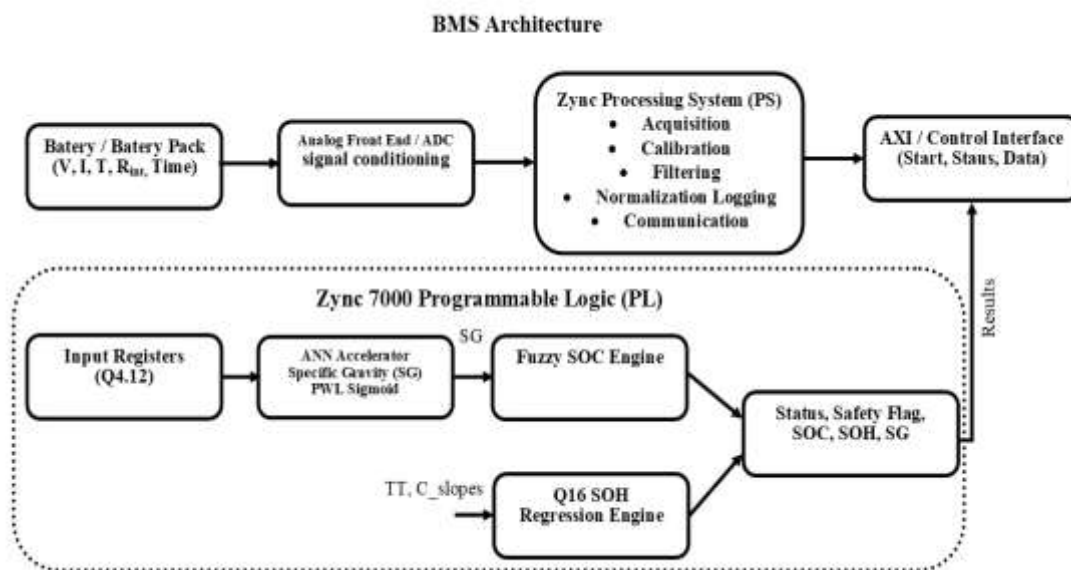


Fig. 1. BMS architecture and PS/PL partitioning.

Three compute blocks sit under the PL top level. The ANN takes five 16-bit Q4.12 inputs, performs inference sequentially, and raises a done signal when it finishes. The fuzzy engine reads SG and temperature and returns a linguistic code together with an 8-bit nominal SoC percentage. The regression engine works on 32-bit Q16.16 inputs and outputs

the used-life and expected-life values. ANN weights and biases are held in coefficient memories, so they can be replaced once the model has been trained. Before deployment, these memories and the fuzzy thresholds should be loaded from a frozen battery dataset that has been validated independently.

III. FPGA EXPERIMENTAL IMPLEMENTATION

Every hardware result in this paper was produced for the XC7Z020-CLG484-1 with Xilinx Vivado 2019.2. We kept the BMS architecture and the implementation settings fixed and changed only the target PL clock. The sweep covers 100, 75, 50, 45.455, 40, 30, 25, 22.727, 20, and 15 MHz. At each point we logged the target period, WNS, the critical-path delay derived from it, the timing-derived Fmax, total and dynamic on-chip power, and the LUT, LUTRAM, flip-flop, and DSP counts. Synthesizing at one nominal frequency would not show where setup timing flips from failing to passing, and that is exactly what the sweep is meant to reveal.

RTL simulation in XSim checks bit-level behavior and cycle latency. Finally, the ZedBoard itself confirms the physical implementation. Accuracy for SG, SoC, and the SoH-related output is to be measured with MAE, RMSE, R^2 , and maximum absolute error. We have left these numbers out of this paper on purpose: the distributed ANN memories currently hold placeholder values and the fuzzy thresholds are provisional.

IV. RESULTS AND DISCUSSION

TABLE I. POST-IMPLEMENTATION TIMING RESULTS FOR THE FREQUENCY SWEEP

Target f (MHz)	Ttarget (ns)	WNS (ns)	Tcritical (ns)	Fmax (MHz)	Setup
100	10.000	-9.564	19.564	51.114	Fail
75	13.330	-6.527	19.857	50.360	Fail
50	20.000	-0.073	20.073	49.818	Fail
45.455	22.500	0.475	22.025	45.403	Pass
40	25.000	3.475	21.525	46.458	Pass
30	33.333	9.764	23.569	42.428	Pass
25	40.000	16.513	23.487	42.577	Pass
22.727	44.001	21.462	22.539	44.368	Pass
20	50.000	25.134	24.866	40.216	Pass
15	66.667	40.911	25.756	38.826	Pass

Table I shows the design moving from failed timing at the higher target frequencies to positive setup margin at the lower ones. At 100 MHz, the 10-ns requirement is missed by 9.564 ns. The derived critical-path delay is 19.564 ns, and the timing-derived Fmax is 51.114 MHz. Dropping the target to 75 MHz brings WNS to -6.527 ns, which is better but still well short of closure. At 50 MHz the WNS is -0.073 ns. That is 9.491 ns better than at 100 MHz, but the slack is still negative, so 50 MHz cannot be called timing closed.

Positive setup margin first appears at 45.455 MHz, with WNS = 0.475 ns, and the 40-MHz run has a wider margin of 3.475 ns. Of the configurations we tested, 45.455 MHz is therefore the highest target frequency that meets setup timing. The results also show why an Fmax computed from a single post-route critical path should be read with some care. As the constraint changes, the tools may place, route, and optimize the design differently, which is why the derived Fmax does not move monotonically across the sweep. We suggest choosing the operating frequency from an implementation that has actually closed timing, with a sensible engineering margin, and not from one extrapolated critical-path value.

TABLE II. POWER AND RESOURCE RESULTS ACROSS TARGET FREQUENCIES

f (MHz)	P _{tot} (W)	P _{dyn} (W)	LUT	LUTRAM	FF	DSP	ADP
100	1.698	1.566	1398	60	945	6	27350.5
75	1.684	1.543	1398	60	945	6	27760.1
50	1.678	1.537	1303	60	945	6	26155.1
45.455	1.677	1.536	1283	60	945	6	28258.1
40	1.676	1.535	1284	60	945	6	27638.1
30	1.674	1.533	1284	60	945	6	30263.0
25	1.673	1.532	1283	60	945	6	30133.8
22.727	1.672	1.531	1285	60	945	6	28962.0
20	1.672	1.531	1283	60	945	6	31903.1
15	1.671	1.530	1283	60	945	6	33044.5

Table-II lists resource use changes very little across the sweep. Every point uses six DSP blocks, 60 LUTRAMs, and 945 flip-flops. The LUT count is 1398 at 100 and 75 MHz, falls to 1303 at 50 MHz, and stays between 1283 and 1285 at the lower frequencies. This is what one would expect from timing-driven synthesis and implementation. Under a tight constraint the tools may keep or replicate extra logic to shorten critical paths; when the constraint is relaxed, the mapped netlist can be a little smaller. Compared with the 100-MHz run, the 50-MHz run uses 6.80% fewer LUTs.

Power moves only a little. Total on-chip power goes from 1.698 W at 100 MHz to 1.671 W at 15 MHz, and dynamic power goes from 1.566 W to 1.530 W. At 50 MHz the total and dynamic figures are 1.678 W and 1.537 W, roughly 1.18% and 1.85% lower than at 100 MHz. It means the static power and the other clock independent parameters resulted as the large component in for EDA estimations. It simple meaning is that it just provide timing margins not saving the power. The area–delay product (ADP) is a relative indicator for how area and delay are normalized. The main conclusions defined with that the timing, resource, and power values reported directly by EDA, and ADP is calculated value.

TABLE III. DIRECT COMPARISON OF THE 50-MHz AND 100-MHz IMPLEMENTATIONS

Metric	50 MHz	100 MHz
Target clock frequency (MHz)	50	100
Clock period (ns)	20.000	10.000
WNS (ns)	−0.073	−9.564
Critical-path delay (ns)	20.073	19.564
Timing-derived F _{max} (MHz)	49.818	51.114
Setup timing status	Fail (marginal)	Fail (marginal)
Total on-chip power (W)	1.678	1.698
Dynamic power (W)	1.537	1.566
LUTs	1303	1398
LUTRAM	60	60
Flip-flops	945	945
DSP blocks	6	6

Table III deserves attention because 50 MHz lies right next to the observed timing boundary. The 50-MHz implementation almost closes, yet its WNS is still −0.073 ns, so we report it as a marginal failure and not as a pass. The 100-MHz run, by contrast, fails by a wide margin. The positive WNS at 45.455 MHz shows that the architecture, with its function unchanged, can be implemented successfully once the clock requirement is relaxed enough. For a real deployment some extra slack is preferable. With 3.475 ns of positive slack, 40 MHz is probably the safer operating point if the architecture is left as it is.

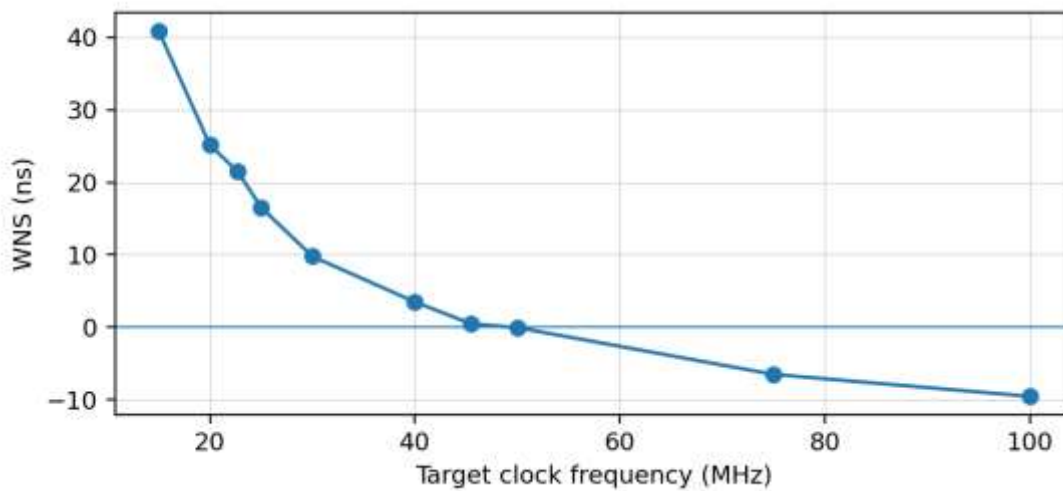


Fig. 2. Post-implementation WNS versus target clock frequency. The measured transition to positive setup slack occurs between 50 and 45.455 MHz.

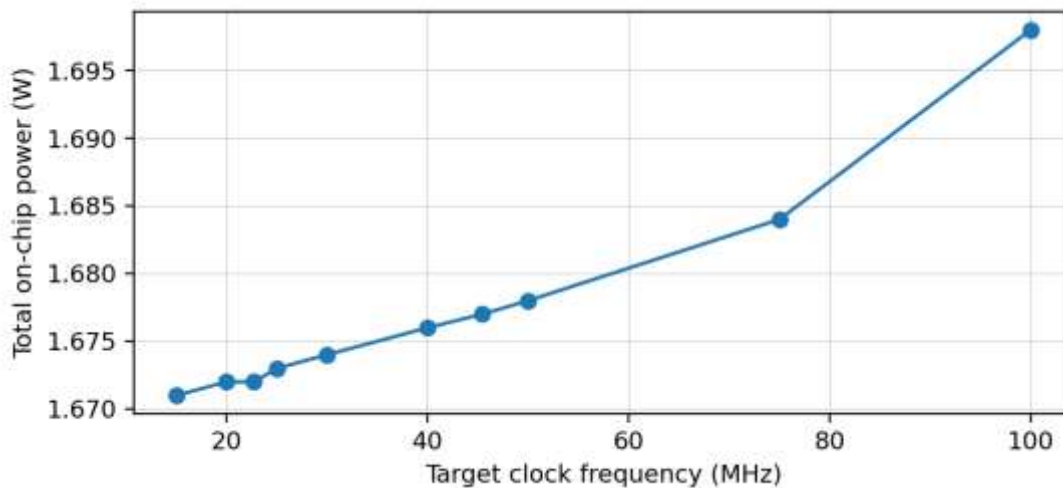


Fig. 3. Estimated total on-chip power versus target clock frequency.

V. CONCLUSION

This study expressed and evaluated the performance for an intelligent BMS estimator on a Zynq-7000 FPGA. The complete model is paired a fixed-point 5-11-11-1 ANN algorithm for a specific-gravity (SG) estimation with a SoC engine with fuzzy 35-rule and in the end a regression block predict expected battery life. The results expressed mainly for a controlled frequency sweep simulation and synthesis. The obtained results shows that timing closure fails at 100 MHz and 50 MHz (WNS = -9.564 & -0.073 ns). The highest operating point that meets the timing is 45.455 MHz with WNS = 0.475 ns, and 40 MHz provides a large margin of 3.475 ns. The total on-chip power decrease a little as the frequency starts elevating down, and the utilization for resource moves a little bit: six DSP blocks, 60 LUTRAMs, and 945 flip-flops at every point.

REFERENCES

- [1] B. Kumar, N. Khare, and P. K. Chaturvedi, "FPGA-based design of advanced BMS implementing SoC/SoH estimators," *Microelectronics Reliability*, vol. 84, pp. 66–74, 2018, doi: 10.1016/j.microrel.2018.03.015.
- [2] C. Vidal, P. Malysz, P. Kollmeyer, and A. Emadi, "Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art," *IEEE Access*, vol. 8, pp. 52796–52814, 2020.
- [3] A. Haraz, K. Abualsaud, and A. Massoud, "State-of-health and state-of-charge estimation in electric vehicles batteries: A survey on machine learning approaches," *IEEE Access*, vol. 12, pp. 158110–158139, 2024, doi: 10.1109/ACCESS.2024.3486989.
- [4] H. M. Hussein et al., "A review of battery state of charge estimation and management systems: Models and future prospective," *WIREs Energy and Environment*, vol. 13, no. 1, e507, 2024, doi: 10.1002/wene.507.
- [5] A. P. Renold and N. S. Kathayat, "Comprehensive review of machine learning, deep learning, and digital twin data-driven approaches in battery health prediction of electric vehicles," *IEEE Access*, vol. 12, pp. 43984–43999, 2024, doi: 10.1109/ACCESS.2024.3380452.
- [6] "VLSI design and FPGA implementation of state-of-charge and state-of-health estimation for electric vehicle battery management systems," *Journal of Energy Storage*, 2023. [Online]. Available via ScienceDirect, article PII S2352152X23022739.
- [7] "FPGA implementation of battery state-of-charge estimation using extended Kalman filter and dynamic sampling," *World Electric Vehicle Journal*, vol. 16, no. 10, art. 587, 2025, doi: 10.3390/wevj16100587.
- [8] AMD/Xilinx, Zynq-7000 SoC Technical Reference Manual and Vivado Design Suite documentation, Vivado 2019.2.
- [9] Avnet, ZedBoard Hardware User Guide and board reference documentation.
- [10] B. Kumar, N. Khare, and P. K. Chaturvedi, "Advanced Battery Management System using MATLAB/Simulink," project source study.