Hardware-in-the-Loop Simulation of FPGA-based State Estimators for Electric Vehicle Batteries

R. Morello*, F. Baronti*, X. Tian§, T. Chau§, R. Di Rienzo*, R. Roncella*,
B. Jeppesen§, W. H. Lin§, T. Ikushima§ and R. Saletti*
* Dip. di Ingegneria dell'Informazione, Università di Pisa, Italy
§ Altera (now part of Intel), USA

***Abstract*—This paper describes a hardware-in-the-loop (HiL) simulation platform specifically designed to test state estimators for Li-ion batteries in electric vehicle applications. Two promising estimators, the Mix algorithm combined with the moving window least squares and the dual extended Kalman filter, are implemented in hardware on a field-programmable gate array (FPGA) and evaluated using the developed HiL platform. The simulation results show the effectiveness of using FPGAs for hardware acceleration of battery state estimators and the importance of their assessment under different operating conditions, *i.e.*, driving schedules, which can be simulated by the HiL platform.**

I. Introduction

In the last few years, plug-in hybrid electric vehicles (PHEVs) and electric vehicles (EVs) have gained popularity due to the ever more stringent emission standards and the increasing consumer awareness of environmental issues. The energy storage system (ESS) is a key component of these vehicles and the enabler of the transition towards e-mobility. Li-ion battery technology is considered the most suitable choice for implementing the on-board ESS (*i.e.*, the traction battery), because of its high power and energy densities and long lifetime. An effective battery management system (BMS) is used to ensure a safe and reliable operation of a Li-ion battery, by monitoring and controlling its charging and discharging processes. This requires the knowledge of the internal state of each battery cell, which is usually expressed by means of the state of charge (SOC) and the state of health (SOH) variables. SOC indicates the remaining amount of charge stored in the battery and SOH is an index of the battery performance degradation compared to the fresh status, which accounts for the capacity fading and the increase of the internal resistance [1].

These variables cannot directly be acquired and need to be inferred from the voltage, current and temperature measurement. The most straightforward method for SOC estimation is the integration of the battery current over time and is named Coulomb Counting (CC). It may provide an accurate SOC estimate assuming that the initial SOC value is known and the current is acquired with a high precision sensor. However, unavoidable errors in the current measurement cause the CC estimate to become unreliable over time. This problem can be tackled by also using the voltage information in a model-based algorithm, such as the popular extended Kalman filter (EKF) [2] and the Mix algorithm [3], among many others. The main open issue is reaching the desired estimation accuracy with a complexity suitable for real-time implementation in the BMS hardware. A model is used to predict the cell voltage in these techniques. The predicted cell voltage is compared with the measured one and the resulting error is used to correct the estimate of the model state variables. The SOC estimation accuracy thus depends on the model capability to reproduce the cell behaviour reliably. An equivalent circuit model (ECM) is often adopted, because it offers a good trade-off between complexity and accuracy. The ECM parameters change with the cell operating conditions (*i.e.*, SOC and temperature) and ageing. An effective approach to track these variations in a BMS is to identify the ECM parameters online. This leads to a joint state and parameter estimation problem.

Although a great deal of research has been conducted on developing new algorithms, just a few works focus on the algorithm assessment under realistic operating conditions [4]–[6]. They exploit the concept of hardware-in-the-loop (HiL) simulation framework, in which the BMS or just the battery state estimator is tested in a simulation environment that reproduces the conditions under which the battery will operate. In more detail, cell level HiL testing platforms, which includes a real cell to which an application-specific current profile is applied under controlled conditions, are described in [4], [5]. In particular, the performance of the battery estimators are assessed with a current profile based on the electric power measured on an EV diving the Federal Test Procedure (FTP) driving schedule in [4] and a current profile representative of a smartphone use in [5]. A mathematical model of the traction battery is used in [6]. The simulation results are only limited to constant current charge/discharge cycles.

The aim of this work is to provide an HiL simulation platform that allows a battery state estimator to be tested under a wide range of operating conditions representative of the EV usage. The developed platform is used to evaluate two battery state algorithms, the Adaptive Mix Algorithm (AMA) and Dual EKF (DEKF), which have proved to be a promising solution for SOC and parameters co-estimation [7], [8]. The AMA and DEKF estimators have been implemented on a Altera MAX® 10 field-programmable gate array (FPGA), which targets low-cost applications and includes non-volatile memory and integrated ADCs. This allows us to assess also the computational complexity of these algorithms and their suitability to be executed in real time on a hardware platform attractive for industrial BMS implementation.

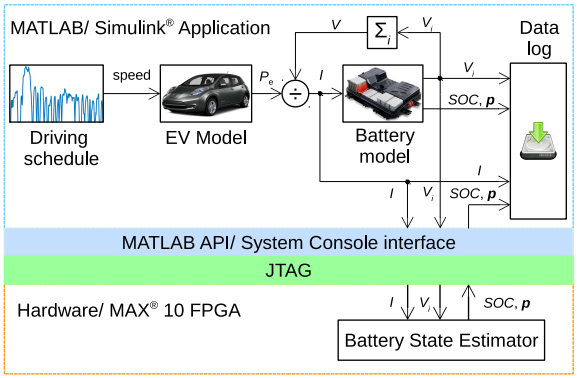


Fig. 1. Block diagram of the developed hardware-in-the-loop simulation platform.

This paper is organized as follows. The next Section describes the HiL simulation platform, including the battery and the electric propulsion models. Section III presents the AMA and DEKF algorithms, while their FPGA implementation is discussed in Section IV. The simulation results are discussed in Section V and finally some conclusions are drawn in Section VI.

II. Hardware-in-the-loop Simulation Platform

In the framework of this paper, the developed HiL simulation platform aims at testing a battery state estimator implemented in an FPGA device in a simulation environment that reproduces its usage in an EV. The traction battery and the electric propulsion system are represented by mathematical models, implemented in a MATLAB/Simulink® application, as shown in Fig. 1. The latter is executed on a PC with a $100\,\mathrm{ms}$ integration time step, which is suitable for capturing the system dynamics of interest. The model outputs consist of the battery current I and the cell voltages V_i. They form the input of the battery state estimator, which in turn computes the SOC estimation as well as the ECM parameter vector p.

The traction battery, simulated by the MATLAB/Simulink® application, and the battery state estimator, implemented in a MAX® 10 FPGA, interact by using digital signals only. Consequently, the interface between the HiL model and the hardware can be implemented as a digital communication layer mapped over the JTAG link, without the need of reproducing the power interface of the battery, as instead required for validating other BMS functions as cell balancing [9]. A brief description of the electric propulsion system (EV model) and the traction battery models are reported below.

A. EV Model

The EV model computes the electric power at the battery's terminals, so that the vehicle speed follows a driving schedule. The latter can be selected among 11 standard driving cycles. The Urban Dynamometer Driving Schedule (UDDS), the Highway Fuel Economy Test (HWFET) and the Federal Test Procedure (FTP) are defined by the U.S. Environmental Protection Agency [10]. The New European Driving Cycle (NEDC), the Extra-Urban Driving Cycle (EUDC) and the Economic Commission for Europe urban driving cycle (ECE R15) are maintained by the United Nations Economic Commission for Europe (UNECE) [11]. The Common Artemis Driving Cycles consist of the Urban cycle (ArtUrban), the Rural road cycle (ArtRoad) and the Motorway cycles (ArtMw130 and ArtMw150, with a maximum speed of 130 and 150 km/h, respectively). The Worldwide harmonized Light vehicles Test Procedures (WLTP) Class 3 are developed following the guidelines of UNECE World Forum for Harmonization of Vehicle Regulations. The duration, distance and average speed of each cycle are reported in Table I. The various driving schedules differ a lot in the average speed and, thus, in the electric power required from the traction battery.

TABLE I
Driving schedules details

Driving schedule	Duration (min)	Distance (km)	Average speed (km/h)
UDDS	23	12.0	31.5
HWFET	13	16.5	77.5
FTP	31	17.8	34.1
EUDC	7	6.5	58.6
NEDC	20	8.3	25.4
ECE R15	3	0.9	16.5
WLTP class 3	30	23.2	46.5
ArtUrban	17	4.9	17.6
ArtRoad	18	17.3	57.4
ArtMw130	18	28.7	96.8
ArtMw150	18	29.5	99.5

A dynamic model has been implemented to simulate the behaviour of an EV on a zero grade road, as in [7]. The mechanical power P_m is calculated as the sum of three contributions: one linked to the acceleration, one to the air resistance and the other to the rolling resistance (1).

$$P_\mathrm{m} = Fv = \left(M\dot{v} + \frac{1}{2}\rho_\mathrm{air} S C_\mathrm{X} v^2 + \alpha_\mathrm{R} M g \right) v \qquad (1)$$

In this equation, F is the traction force, v is the speed, M is the kerb weight, S is the frontal area, C_X is the drag coefficient, α_R is the rolling resistance, ρ_air is the air density and g is the gravity acceleration.

The electric power P_e is obtained from P_m by using the equation (2), in which η_wheel is the efficiency from the battery to the wheels and η_reg is the efficiency in the opposite direction, *i.e.*, during the regenerative braking.

$$P_\mathrm{e} = \left(\frac{1}{\eta_\mathrm{wheel}} \frac{1 + sgn(P_\mathrm{m})}{2} + \eta_\mathrm{reg} \frac{1 - sgn(P_\mathrm{m})}{2} \right) P_\mathrm{m} \qquad (2)$$

In order to obtain the battery current, the electric power is divided by the sum of the cell voltages calculated by the battery model, as shown in Fig. 1.

B. Battery Model

The battery model is able to simulate a given number of series-connected cells. The only input is the battery current which is the same for all the series-connected cells. At each

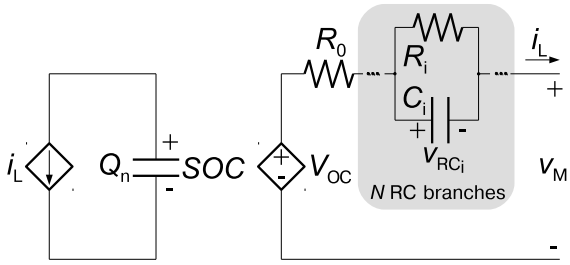


Fig. 2. Electric circuit model.

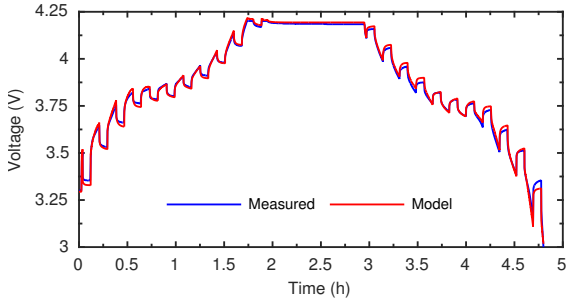


Fig. 3. Model and measured voltages.

time step, the model generates the arrays of the cell voltages V_i, SOC, as well as the current values of the model parameters. The model adopted is the ECM shown in Fig. 2 with 2 RC branches. This is a very common choice to simulate a Li-ion battery with high accuracy in an HiL platform [9]. The left hand side models the cell capacity Q_n and evaluates the SOC as the voltage across a linear capacitor with a capacity equal to Q_n (expressed in Coulomb) divided by 1 V (this is equivalent to the CC method). The cell voltage v_M is obtained by the sum of the open-circuit voltage V_OC and a dynamic term, which accounts for the internal ohmic resistance R_0 and the double layer (V_{RC_1}) and diffusion (V_{RC_2}) effects of the Li-ion battery during charging and discharging (2 RC branches).

The model parameters change with manufacturing variations, ageing and operating conditions, such as temperature and state of charge. In order to model the dependency of the parameters on temperature and SOC, their values are stored in 2D LUTs. The variability of the cell behaviour is considered by setting the initial SOC, the model parameters, the temperature and the capacity of each cell individually.

In this work, the LUTs have been populated with the values extracted from pulsed current tests performed at different temperatures and with different pulse amplitudes on a 1.5 A h NMC cell [12]. The model is then generalized to simulate a cell with the same technology but different capacity by proportionally scaling the LUT values with the capacity, directly for the resistive elements and inversely for the capacitive ones. As an example of the model capability of reproducing the cell voltage, Fig. 3 shows the comparison between the cell voltage predicted by the model and the measured one, during a pulsed current test. We note that the predicted cell voltage agrees very well with the measured one, as the maximum and rms errors are 132 mV and 13.6 mV, respectively.

The simulated traction battery consists of 96 series-connected NMC cells with a capacity of 66.2 A h. The battery nominal voltage is 355.2 V. The EV model has been parameterized to resemble a commercial electric car. The model parameters are reported in [7]. Fig. 4 shows the results of the ArtMw150 driving cycle simulation. Together with the speed profile, the electric power (calculated by the EV model), the battery current and voltage (computed by the battery model) are reported.

III. Battery State Estimators

This Section briefly describes the AMA and DEKF battery state estimators [8]. They are both based on the ECM shown in Fig. 2, but only one RC branch is used. This reduces the computational complexity, while preserving a good accuracy, especially in applications with fast transients. The ECM parameters are identified online in both approaches to track their variations with the operating conditions and the ageing of the battery.

AMA is a technique based on the Mix Algorithm for SOC estimation [3] and the Moving Window Least Squares (MWLS) method, applied to the AutoRegressive eXogenous (ARX) representation of the ECM for online parameter identification of the ECM [13], [14]. The Mix Algorithm acts as an observer by comparing the model output voltage to the measured cell voltage. The resulting error is amplified and used to correct the estimation of the SOC state variable, computed using the CC method. The procedure to determine the observer gain is discussed in [15]. The ARX model is obtained by firstly linearising the OCV-SOC non-linear

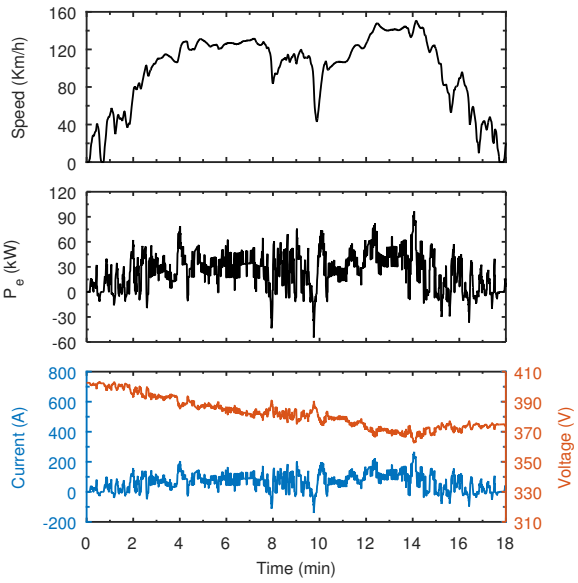


Fig. 4. Speed, electric power, battery current and voltage during an ArtMw150 driving cycle.

relationship of the ECM around the time-varying cell operating point and then by calculating the discrete-time relationship (3) between the input samples $u(k)$ and output samples $y(k)$, *i.e.*, between the current input and the cell terminal voltage output.

$$\begin{aligned} y(k) = &- a_1 y(k-1) - a_2 y(k-2) + \alpha_0(1 + a_1 + a_2) \\ &+ b_0 u(k) + b_1 u(k-1) + b_2 u(k-2) \end{aligned} \tag{3}$$

The parameters $[a_1, a_2, b_0, b_1, b_2]$ are identified by applying the LS method to a set of current and voltage samples in a given time window, which is periodically shifted in time. The ECM parameters $[R_0, R_1, C_1]$ are then extracted from the coefficients of the ARX model [7].

In the DEKF technique, two cooperating Kalman Filters for non linear systems are executed simultaneously: one for the state and the other for the parameter estimation. The use of the dual estimation, instead of a joint estimation (in which only one Kalman Filter is used) reduces the state matrix dimensions and may improve the estimation robustness [16]. The parameter evolution is described by the process equation (4), which is used in combination with the measurement equation (6), in order to build the first EKF. The state evolution is instead represented by (5), which is again combined to the measurement equation (6) to form the second EKF.

$$p(k+1) = p(k) + \chi(k), \tag{4}$$

$$x(k+1) = \mathcal{F}(x(k), i_{\mathrm{L}}(k), p(k)) + \xi(k), \tag{5}$$

$$v_{\mathrm{T}}(k) = \mathcal{G}(x(k), i_{\mathrm{L}}(k), p(k)) + \psi(k). \tag{6}$$

The measurement equation (6) is the same for both filters. In the above equations, k is the discrete time, p is the parameter vector, $x = [SOC, V_{\mathrm{RC}_1}]$ is the battery state vector. χ, ξ and ψ are the parameters, the state and the measurement noise with zero mean and covariance matrices Σ^{χ}, Σ^{ξ} and Σ^{ψ}, respectively.

IV. FPGA Implementation of the AMA and DEKF Battery Estimators

FPGAs have proven their effectiveness in many industrial applications. They are capable of high throughput, low latency processing through parallelism and optimized data paths. The flexibility of user-defined circuits enables the combination of different data types and precisions, which improves performance and reduces cost. An FPGA is also highly scalable for design upgrade and system expansion [17].

The AMA and DEKF battery estimators are implemented using the Altera design flow as illustrated in Fig. 5. Hardware design starts in DSP Builder where the algorithm is described in Simulink models and synthesized to low-level hardware description. The design is optimized for performance and resource by applying pipelining, time-division multiplexing/folding and customizing precision. In the Qsys system integration tool, the generated hardware components, such as the AMA module and matrix processor in Fig. 6, are connected to other components in the system, including a Nios II 32-bit soft processor, JTAG and memory. A complete design is synthesized and programmed for the target FPGA using

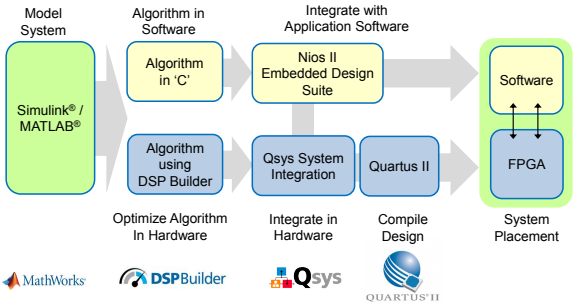
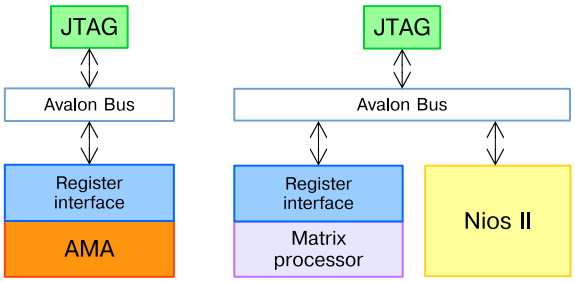


Fig. 5. Altera FPGA design flow.



(a) AMA (b) DEKF

Fig. 6. Block diagram of the implemented estimators.

Quartus II design software. For the DEKF implementation, which includes application software running on Nios II, Embedded Design Suite compiles the C software and runs the compiled application on the FPGA. To support HiL simulation, DSP Builder provides an interface to the System Console system debugging tool. Through this interface, the Simulink application can perform memory-mapped access to the design running on the FPGA.

The AMA estimator is entirely built in hardware and is provided with a memory mapped interface (Fig. 6(a)), which can be used to integrate the module in a system on chip. This interface consists of input and output registers, to write the algorithm input values (*i.e.*, cell voltage and current) and to read the computed cell state (*i.e.*, SOC and ECM parameters).

The DEKF is built on an architecture with a Nios II embedded processor and a dedicated matrix processor, as shown in Fig. 6(b). Nios II is a 32-bit soft-core processor which is implemented in FPGA logic and is customizable for specific application requirements. In this application, a floating-point custom instruction IP component is included and supported by the C compiler to accelerate standard floating point operations. In simulation, Nios II uses JTAG to read from and write to the Simulink application. The prediction phase of SOC estimation and parameter identification are also performed on Nios II. To improve performance, the correction phase is offloaded to the matrix processor, which is a generic matrix processing engine able to perform various matrix calculations using Fadeev and matrix multiply-accumulate cores [18]. The matrix size is programmable at run-time and a number of matrix calculations can be scheduled in sequence.

TABLE II
ESTIMATOR RESOURCE USAGE

Resource	AMA	DEKF
Logic Elements	38 k/50 k (76 %)	23 k/50 k (46 %)
9-bit Multiplier	219/288 (76 %)	39/288 (14 %)
Memory bits	170 Kb/1638 Kb (10 %)	230 Kb/1638 Kb (26 %)
Execution time	34 µs (@100 MHz)	33 µs (@100 MHz)

The AMA and the DEKF hardware implementations have been implemented in a low cost Altera MAX® 10 FPGA (10M50DAF484C6GES device). A comparison of the FPGA resource usage is shown in Table II. Both estimators fit in the chosen device, but the DEKF uses fewer resources than the AMA. They need a similar execution time to update both the state and the parameters (the value in the table is obtained with a clock frequency of 100 MHz). Such a very short execution time allows the same module to be used for estimating a large number of cells in a time multiplexing fashion. The number of cells affects the required memory inside the AMA and DEKF modules (the memory bits figure reported in the table refers to 12 cells).

V. SIMULATION RESULTS

The developed HiL platform has been used to assess the performance of the hardware implementations of the AMA and DEKF estimators. The sampling time is equal to 100 ms, the length of the moving window in the AMA is set to 90 s and the noise covariance matrixes in the DEKF have been empirically determined. The simulation selects one driving schedule from Table I and repeats it until the battery becomes fully discharged (*i.e.*, $SOC = 0$). Even if the parameters, as well as SOC initialisation and temperature, of each cell of the battery can be set independently, the simulations described below have been carried out with identical cells all starting from the full charge state (*i.e.*, $SOC = 100\,\%$). Moreover, the temperature of all the cells has been kept constant at 25 °C throughout all the simulation. Thus, all the battery cells behave in exactly the same way and consequently so do the estimators, which are capable of handling up to 12 cells.

As an example, Fig. 7 shows the simulation results for the UDDS and ArtMw150 cycles, which are representative of urban and motorway driving, respectively. The ArtMw150 electric power is on average significantly higher than the UDDS one, leading to a much shorter driving time. The driving range is 93.3 km for the ArtMw150 cycle and 166.4 km for the UDDS cycle. The SOC estimated by both algorithms is in good agreement with the reference one evaluated by the HiL battery model, apart from the SOC range 50 % down to 25 %, in which the SOC is poorly observable from the cell voltage, as discussed in previous works [8], [15]. In this range, AMA provides a better SOC estimation than DEKF for the UDDS cycle, whereas the opposite behaviour can be observed for the ArtMw150 cycle.

Fig. 7 shows also the comparison between the identified Ohmic resistance R_0 and the time constant $\tau_1 = R_1 C_1$ of the single RC branch of the ECM used in the estimator and the corresponding values used in the HiL battery model (regarding its time constants, the fastest one is considered). It is worth noticing that R_0 is well identified by both estimators, especially during the UDDS cycle. This is an important result, as this parameter affects the accuracy of the model and provides a good indication of the battery ageing. The identification of the time constant seems to be more noisy.

TABLE III
SOC ESTIMATION ERROR

Driving Schedule	AMA		DEKF	
	Max (%)	rms (%)	Max (%)	rms (%)
UDDS	3.3	1.3	5.7	1.9
NEDC	3.7	1.4	5.4	1.8
HWFET	6.2	2.6	5.9	2.3
FTP	3.8	1.4	6.0	2.0
EUDC	5.4	2.3	7.6	2.7
ECE R15	1.2	0.4	3.8	1.3
WLTP class 3	4.6	2.0	5.6	2.2
ArtUrban	2.5	1.1	5.1	2.0
ArtRoad	5.5	2.2	7.1	2.5
ArtMw130	9.0	4.5	5.4	2.8
ArtMw150	9.5	4.5	3.5	1.9

The maximum and rms SOC errors for all driving schedules are reported in Table III. We note that both estimators provide a good SOC estimation for all the driving cycles, as the rms error is always below the 4.5 % and 2.8 % for AMA and DEKF, respectively. As a comparison, the SOC errors reported in [4] are 4.1 % and 9.2 % for two different Li-ion batteries subject to a current profile based on the electric power measured on an EV driving the FTP cycle. Finally, we observe that the characteristics of the driving cycle and thus of the related battery current have a remarkable impact on the performance of the state estimator. In more detail, AMA provides better results for urban driving schedules, whereas DEKF is more reliable for motorway driving schedules, such as the HWFET, the ArtMw130, and ArtMw150.

VI. CONCLUSIONS

This paper has discussed the development of a HiL platform for testing battery state estimators under realistic operating conditions found in EV applications. The HiL platform consists of a dynamic model of an EV and an ECM of the traction battery. It is implemented in a MATLAB/Simulink® application, which interacts with the estimator implemented on an Altera MAX® 10, using a highly automated design flow, which starts from describing the algorithm in a MATLAB/Symulink® model. Two promising model-based estimators, the AMA and the DEKF algorithm, have been implemented and tested using the developed HiL platform. Simulation results show that both estimators are suitable for battery state estimation in EVs, providing good SOC estimation accuracy and reliable identification of the ECM parameter embedded in the estimator.

This work has demonstrated that FPGAs can be an effective solution for hardware acceleration of battery state estimators, so that a single low cost device can be used to estimate all the cells of a battery module (typically consisting of 12 cells) or

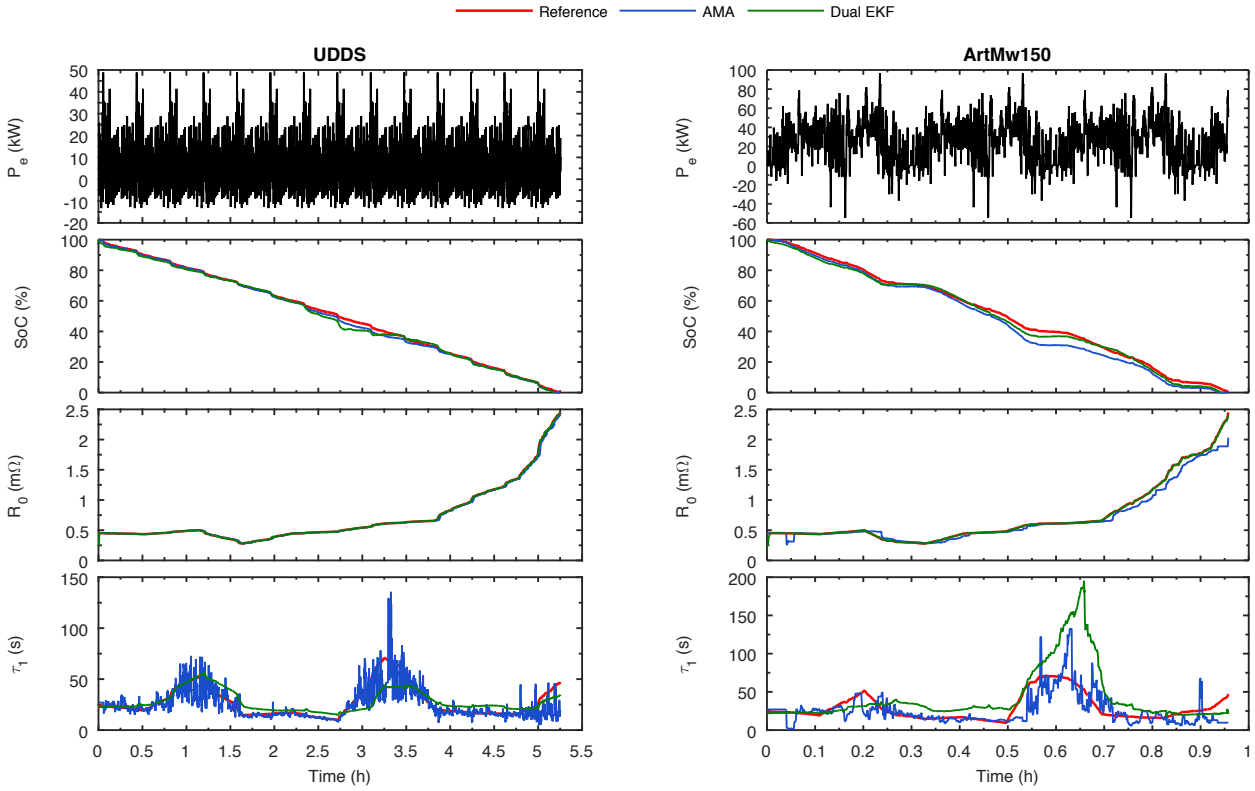


Fig. 7. Behaviour of SOC and ECM parameters during an UDDS test (left-hand side) and an ArtMw150 test (right-hand side).

even all the cells in the traction battery (typically consisting of 8 modules). Moreover, it has highlighted the importance of assessing a battery state estimator for an EV battery under a wide range of driving schedules, as its performance may change with the current load profile of the battery in a remarkable way. To the best of our knowledge, this is the first time that this result is clearly shown.

REFERENCES

[1] H. Rahimi-Eichi, U. Ojha, F. Baronti, and M.-Y. Chow, "Battery Management System: An Overview of Its Application in the Smart Grid and Electric Vehicles," *IEEE Ind. Electron. Mag.*, vol. 7, no. 2, pp. 4–16, jun 2013.

[2] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 3. State and parameter estimation," *J. Power Sources*, vol. 134, no. 2, pp. 277–292, aug 2004.

[3] F. Codeca, S. M. Savaresi, and G. Rizzoni, "On battery State of Charge estimation: A new mixed algorithm," in *2008 IEEE Int. Conf. Control Appl.* IEEE, sep 2008, pp. 102–107.

[4] Y. He, W. Liu, and B. J. Koch, "Battery algorithm verification and development using hardware-in-the-loop testing," *J. Power Sources*, vol. 195, no. 9, pp. 2969–2974, may 2010.

[5] G. Avvari, B. Pattipati, B. Balasingam, K. Pattipati, and Y. Bar-Shalom, "Experimental set-up and procedures to test and validate battery fuel gauge algorithms," *Appl. Energy*, vol. 160, pp. 404–418, dec 2015.

[6] H. Wu, "Hardware-in-loop verification of battery management system," in *2011 4th Int. Conf. Power Electron. Syst. Appl.* IEEE, jun 2011, pp. 1–3.

[7] F. Baronti, W. Zamboni, N. Femia, H. Rahimi-Eichi, R. Roncella, S. Rosi, R. Saletti, and M.-Y. Chow, "Parameter identification of Li-Po batteries in electric vehicles: A comparative study," in *2013 IEEE Int. Symp. Ind. Electron.* IEEE, may 2013, pp. 1–7.

[8] R. Morello, W. Zamboni, F. Baronti, R. D. Rienzo, R. Roncella, G. Spagnuolo, and R. Saletti, "Comparison of State and Parameter Estimators for Electric Vehicle Batteries," in *IECON 2015 - 41st Annu. Conf. IEEE Ind. Electron. Soc.*, 2015, pp. 5433–5438.

[9] H. Dai, X. Zhang, X. Wei, Z. Sun, J. Wang, and F. Hu, "Cell-BMS validation with a hardware-in-the-loop simulation of lithium-ion battery cells for electric vehicles," *Int. J. Electr. Power Energy Syst.*, vol. 52, pp. 174–184, nov 2013.

[10] "United States Environmental Protection Agency (U.S. EPA)." [Online]. Available: http://www3.epa.gov/

[11] "United Nations Economic Commission for Europe (UNECE)." [Online]. Available: http://www.unece.org/info/ece-homepage.html

[12] F. Baronti, G. Fantechi, E. Leonardi, R. Roncella, and R. Saletti, "Enhanced model for Lithium-Polymer cells including temperature effects," in *IECON 2010 - 36th Annu. Conf. IEEE Ind. Electron. Soc.* IEEE, nov 2010, pp. 2329–2333.

[13] H. Rahimi-Eichi, F. Baronti, and M.-Y. Chow, "Modeling and online parameter identification of Li-Polymer battery cells for SOC estimation," in *2012 IEEE Int. Symp. Ind. Electron.* IEEE, may 2012, pp. 1336–1341.

[14] ——, "Online Adaptive Parameter Identification and State-of-Charge Coestimation for Lithium-Polymer Battery Cells," *IEEE Trans. Ind. Electron.*, vol. 61, no. 4, pp. 2053–2061, apr 2014.

[15] F. Baronti, R. Roncella, R. Saletti, and W. Zamboni, "FPGA Implementation of the Mix Algorithm for State-of-Charge Estimation of Lithium-Ion Batteries," in *IECON 2014 - 40th Annu. Conf. IEEE Ind. Electron. Soc.*, 2014, pp. 5641–5646.

[16] R. Restaino and W. Zamboni, "Rao-blackwellised particle filter for battery state-of-charge and parameters estimation," in *IECON 2013 - 39th Annu. Conf. IEEE Ind. Electron. Soc.* IEEE, nov 2013, pp. 6783–6788.

[17] J. J. Rodriguez-Andina, M. D. Valdes-Pena, and M. J. Moure, "Advanced Features and Industrial Applications of FPGAsA Review," *IEEE Trans. Ind. Informatics*, vol. 11, no. 4, pp. 853–864, aug 2015.

[18] D. Pritsker, "Hybrid implementation of Extended Kalman Filter on an FPGA," in *2015 IEEE Radar Conf.* IEEE, may 2015, pp. 0077–0082.