ACCÉLÉRER LE DÉVELOPPEMENT ET LA MISE EN ŒUVRE DE FONCTIONNALITÉS BMS AUTOMOBILES

Lea.Pitault@nxp.com
AI/ML & SW Engineer

Vincent.Martinez@nxp.com
System Engineer

CTO Automotive System Innovations
Accelerating development and implementation of automotive BMS functionalities (e.g. SoC, SoH) on NXP Greenbox 3 with MATLAB and Simulink

**Introduction**

**Data Generation with Simulink BMS model**

**State-of-Charge Algorithms Investigation**

**Prototyping on NXP GreenBox with Simulink**
Introduction
NXP SEMICONDUCTORS: CORPORATE OVERVIEW

World leader in secure connectivity solutions for embedded applications, NXP is pushing boundaries in the automotive, industrial & IoT, mobile, and communication infrastructure markets.

OUR TARGET MARKETS
A POSITION OF STRENGTH TO BETTER SERVE OUR 26,000+ CUSTOMERS

We accelerate breakthroughs that advance the world through our semiconductor technology leadership

<table>
<thead>
<tr>
<th>EMPLOYEES IN</th>
<th>30+ COUNTRIES</th>
<th>~34,500 TEAM MEMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>France: Caen, Paris, Mougins, Toulouse, Grenoble</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| ~9,500 Patent Families | ~$13.21B Annual Revenue ¹ |
| ~60+ Year History | ~11,000 Engineers |

¹ Posted revenue for 2022 – Please refer to the Financial Information page of the Investor Relations section of our website at www.nxp.com/investor for additional information
GREENBOX 3 DEVELOPMENT PLATFORM FOR S32Z2 AND S32E2 REAL-TIME PROCESSORS

Main Function

- Quick evaluation for Hybrid & Electric Vehicle propulsion, motor-control and battery management use cases
- Complete NXP system solution includes PMICs, Ethernet switch & CAN transceivers

S32Z2/E2

- ASIL D
- 8x Arm Cortex-R52 cores operating up to 1 GHz with NEON™, powerful support for distributed computing, machine learning workloads and DSP/ML Processor offload
- Numerous automotive communication interfaces: Ethernet, CAN FD, LIN, UART, JTAG, SDHC, PSI5, SENT
- Easy to use out of the box experience with advanced control applications including example code

Compatible with S32Z/E Vehicle Integration Platform (GreenVIP)

- Provide a functional software platform that allows customers to begin developing an application on the S32ESZ family with minimal effort

www.NXP.Com/GreenBox3
AUTOMOTIVE BATTERY MANAGEMENT SYSTEM (BMS)

- Megatrend: electrification of vehicles
- Typical number of cells in a car battery pack: between 100 and 200
- Cells are monitored continuously (current, voltage, temperature)
- BMS purposes: avoids hazards, optimizes charging and discharging
- Today, BMS processing mostly based on signal processing techniques

**Challenge**

Accurate estimation of the State-of-Charge (SoC) of the battery cells required

→ Can we do it with Deep Learning?
MODEL-BASED DESIGN TOOLBOX (MBDT)
ADVANCED TOOLING ADD-ON FOR MATLAB® AND SIMULINK®

Developed as collaboration between NXP and MathWorks. MathWorks tools optimized for use with NXP HW.

“A collection of Tools & Libraries designed to assist customers with prototyping and accelerate algorithm development on NXP MCUs from MATLAB and Simulink”

- Provides an integrated development environment and toolchain support
- Used to configure, generate and deploy applications on the MCU
- “True” Model-Based Design approach by introducing the usage of an External Configuration tool for pins, clocks and peripherals
- Generates code on top of NXP’s Real-Time Drivers (RTD) for AUTOSAR® and non-AUTOSAR
Data Generation with Simulink BMS model
Very few public datasets available

Data generation based on MathWorks BMS Simulink model + light customization (thermal etc.)

→ 6 or 16x6 cells in series

Development of 54 profiles:
- 48 for training
- 6 for validation/test

- Current (of the pack)
- Temperature
- Voltage
- “True” SOC
- SOC calculated with Coulomb Counting and EKF

→ per cell

Input current profile

Simulink BMS model

Generated data
DATA GENERATION – BMS SIMULINK MODEL (2)

Module of 6 cells in series

Example of charge/discharge cycle

Model architecture of the cells

Ambient temperature

Cell 1
Cell 3
Cell 5
Cell 2
Cell 4
Cell 6

different temperature evolutions due to different thermal dissipation

Cell temperature

Cell_Temperatures

Cell_Temperatures: 1
Cell_Temperatures: 2
Cell_Temperatures: 3
Cell_Temperatures: 4
Cell_Temperatures: 5
Cell_Temperatures: 6
SoC Algorithms Investigation
ALGORITHMS INVESTIGATION

- Investigate the **features**
- Investigate the **input shape**
- Investigate the neural network architecture: **recurrent VS convolutional**
- Investigate the **hyperparameters**
ALGORITHMS INVESTIGATION

Flexibility in the workflow

MATLAB

Python

Conversion in MATLAB

Neural Network

MBDT (for hardware implementation)

Recurrent/Convolutional Neural Network

LSTM/GRU Architecture

CNN Architecture

MATLAB Deep Learning Network Analyzer
Prototyping on NXP GreenBox with Simulink
SETUP

- CodeWarrior Debug Probe
- Computer with Simulink and Serial connection
- GreenBox 3 Daughter Card with S32E278 Processor
PROTOTYPING – SIMULINK MODEL (1)

• Goal: assess if the ML based SoC algorithm can be run on embedded platform
• Use of the Model-Based Design ToolBox in Simulink to generate C code and run it on NXP GreenBox
• Adapt the models developed in Python environment to MATLAB
PROTOTYPING – SIMULINK MODEL (2)

SoC according to estimation method EKF/FNN

- **Target**
- **NN estimate**
- **EKF estimate**
PROTOTYPING – SIMULINK MODEL (3)

- The MBDT provides a Profiling Report
- Example generation of the profiling report for one sequence data (3*70) as input for LSTM model

### Code Execution Profiling Report for NN_SoC_predict_App

The code execution profiling report provides metrics based on data collected from a MBDT or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the MBDT or PIL test harness or inside the code generated for each component. See [Code Execution Profiling](#) for more information.

#### 1. Summary

<table>
<thead>
<tr>
<th>Total time</th>
<th>0.12 ms</th>
</tr>
</thead>
</table>

#### 2. Feasibility Checks

<table>
<thead>
<tr>
<th>Check Description</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall average CPU utilization lower than threshold (79%)</td>
<td>Passed</td>
</tr>
<tr>
<td>Overall maximum CPU utilization lower than threshold (79%)</td>
<td>Passed</td>
</tr>
<tr>
<td>All average execution times are shorter than their task period</td>
<td>Passed</td>
</tr>
<tr>
<td>All maximum execution times are shorter than their task period</td>
<td>Passed</td>
</tr>
</tbody>
</table>

#### 3. Profiled Sections of Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Maximum Execution Time (in ms)</th>
<th>Average Execution Time (in ms)</th>
<th>Maximum Self Time (in ms)</th>
<th>Average Self Time (in ms)</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] NN_SoC_predict_App_summary</td>
<td>390</td>
<td>292</td>
<td>292</td>
<td>292</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_App_warmup</td>
<td>67959</td>
<td>1493</td>
<td>278</td>
<td>101</td>
<td>71</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_App.utime1</td>
<td>532</td>
<td>106</td>
<td>732</td>
<td>296</td>
<td>71</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_App.utime2</td>
<td>84657</td>
<td>84657</td>
<td>184</td>
<td>100</td>
<td>53</td>
</tr>
<tr>
<td>[1] DeepLearningNetwork_predictors</td>
<td>67959</td>
<td>67959</td>
<td>6892</td>
<td>6902</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>10702</td>
<td>10702</td>
<td>698</td>
<td>698</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>27191</td>
<td>27191</td>
<td>560</td>
<td>560</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>486</td>
<td>486</td>
<td>486</td>
<td>486</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>10702</td>
<td>10702</td>
<td>698</td>
<td>698</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>84657</td>
<td>84657</td>
<td>184</td>
<td>100</td>
<td>53</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>6474</td>
<td>6474</td>
<td>520</td>
<td>520</td>
<td>1</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>6324</td>
<td>6324</td>
<td>574</td>
<td>574</td>
<td>10</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_Appmultiply1</td>
<td>624</td>
<td>624</td>
<td>574</td>
<td>574</td>
<td>10</td>
</tr>
<tr>
<td>[1] NN_SoC_predict_App_summary</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>1</td>
</tr>
</tbody>
</table>
Take aways
CONCLUSIONS

**Data Generation**

- Able to generate meaningful datasets with Simulink model

**Algorithm investigation**

- Able to explore variety of SoC algorithms with MATLAB/Simulink toolboxes
- AI-based SoC algorithms (FNN, LSTM...) show equal or better accuracy than ‘classical’ EKF
- Bridges with Python environments

**Prototyping**

- C code generation from Simulink
- Fast prototyping
- Detailed performance report
SECURE CONNECTIONS
FOR A SMARTER WORLD