SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V13I5P111

Volume 13 Issue 5, 127-142, May 2026

Original Article

MIRA: Memory-Centric Intelligent Reconfigurable Architecture for LSTM

Sagar Vijay Mhatre[1], Vinitkumar Jayaprakash Dongre[2], Sudhakar Mande[3]

[1,2]*Department of Electronics and Telecommunication, Thakur College of Engineering, Mumbai, India.*
[1]*K J Somaiya Institute of Technology, Mumbai, India.*
[3]*Department of Electronics and Telecommunication, Don Bosco Institute of Technology, Mumbai, India*

[1]*Corresponding Author : sagar.mhatre@somaiya.edu*

Received: 19 February 2026 Revised: 18 March 2026 Accepted: 17 April 2026 Published: 30 May 2026

Abstract *- Accurate and real-time state-of-charge (SoC) estimation is essential for safe and efficient operation of lithium-ion batteries in electric vehicles. Although Long Short-Term Memory (LSTM) networks provide high prediction accuracy, their deployment in embedded battery management systems is limited by computational and resource constraints. This paper presents Memory-Centric Intelligent Reconfigurable Architecture (MIRA), a hardware–software co-designed FPGA-based accelerator for real-time SoC prediction using an optimized LSTM model. The proposed framework integrates feature reduction, sequence modeling, and quantization-aware training with hardware-aware optimizations, including fixed-point representation, activation function approximation, and parallelized matrix multiplication. The accelerator is implemented on a low-cost PYNQ-Z2 platform using near-memory computation and tiling to improve performance and energy efficiency. Experimental results show that the proposed design achieves an inference latency of 0.9536 ms per sample with a throughput of 1048.92 samples/s, significantly outperforming a CPU-based implementation. The system consumes 136 mW and achieves 4.28 GOPS/W. A cost-aware metric, GOPS/W/$, is introduced, with the proposed design achieving 0.0331, outperforming existing accelerators. These results demonstrate an effective balance between accuracy, efficiency, and cost for edge deployment.*

Keywords *- Edge AI, FPGA Acceleration, LSTM, Memory-Centric Architecture, State-of-Charge Estimation*

1. Introduction

The rapid adoption of Electric Vehicles (EVs) has intensified the demand for reliable and efficient Battery Management Systems (BMS). EVs widely use Lithium-ion Batteries (LIBs) because of their high energy density, long cycle life, and higher efficiency. But accurate monitoring of key internal parameters is important in order to ensure safe and optimum operation of the batteries. State-of-Charge (SoC) is one of the most critical parameters, as it represents the remaining capacity of the battery. It determines the driving range estimation, charging strategies, and battery safety [1].

Inaccurate SoC estimation misleads the BMS, reducing the range accuracy, compromising battery safety, and shortening the battery life [36]. Accurate SoC estimation remains a challenging task due to the inherently nonlinear and time-varying electrochemical behavior of Lithium-ion Batteries. It is affected by numerous factors, including temperature, aging of the battery, and variable operating conditions [2]. Conventional SoC estimation methods, such as Coulomb Counting, Open-Circuit Voltage (OCV), and equivalent circuit models, are widely used in commercial BMS implementations. These methods often suffer from problems of cumulative errors, parameter variation sensitivity, and low accuracy in variable load conditions.

To address these limitations, data-driven approaches, including machine learning and deep learning, have gained increasing attention for battery state estimation. These methods can learn the nonlinear complex relation between the battery state and other parameters, directly from data. They don't have to rely on detailed electrochemical models of the battery. Among the other Deep Neural Networks (DNNs), Long Short-Term Memory (LSTM) Networks have shown better results, as they can model sequential data and capture long-term temporal dependencies of battery parameters such as voltage, current, and temperature [3]. Several studies have demonstrated that LSTM-based models achieve high accuracy in predicting SoC and State-of-Health (SoH), outperforming traditional model-based methods in dynamic operating conditions [2]. Although LSTM models have shown encouraging performance, they often need substantial computational resources and memory requirements. Such requirements restrict their usage in the embedded BMS hardware. Typical BMS platforms use resource-constrained microcontrollers, where strict requirements of power

consumption, real-time response, and reliability must be satisfied. Running complex DNN models on traditional computational platforms at the edge can lead to large inference latency and energy consumption. While GPUs can accelerate the performance, they are often unsuitable for automotive embedded systems due to high power consumption and cost constraints. In order to address these issues, hardware accelerators for neural network inference have been widely investigated. Field Programmable Gate Arrays (FPGAs) provide a very useful solution that can provide high performance and low power consumption at a lower cost. The FPGAs can exploit the inherent parallelism of the DNN models for the purpose of real-time response.

The reconfigurable customized hardware DNN pipelines can be efficiently used for real-time edge-AI applications [4]. Recent studies have explored the integration of deep neural networks (DNNs) with FPGA-based hardware for battery management systems (BMSs), showing that hardware-accelerated battery state estimation is feasible. However, most existing FPGA-based LSTM accelerators are designed for general sequence-processing applications rather than battery State-of-Charge (SoC) estimation.

Similarly, SoC-focused deep-learning studies largely emphasize prediction accuracy, with limited attention to the practical challenges of deploying LSTM models on low-cost, resource-constrained FPGAs. Existing hardware implementations are also commonly evaluated on higher-capacity devices, which may not reflect the limited DSP, BRAM, LUT, and power budgets of practical edge-based BMS platforms. Therefore, the microarchitectural design space of LSTM accelerators for SoC prediction remains insufficiently explored, particularly in terms of balancing accuracy, latency, energy efficiency, resource utilization, and platform cost. This motivates the development of a hardware accelerator for LSTM-based SoC prediction in EV battery management systems. The main challenge is to achieve accurate SoC estimation on a low-cost FPGA while meeting strict constraints on latency, energy consumption, memory, and hardware resources.

To address this problem, the proposed accelerator implements a multilayer LSTM network using precision-sensitive fixed-point arithmetic on the Xilinx PYNQ-Z2 board. In addition, a comprehensive design space exploration is performed to study how architectural choices, such as matrix multiplication parallelism, gate-computation microarchitecture, and activation-function implementation, affect prediction accuracy, performance, and resource utilization. The main contribution of this work is summarized as follows:

1. Hardware-Optimized SoC Prediction Model: A compact two-layer LSTM model is developed using the NASA EV battery dataset and adapted for FPGA deployment through selected input features, sequence length, and 8-bit fixed-point representation.
2. Efficient FPGA Accelerator Architecture: A memory-centric accelerator is designed for the low-cost PYNQ-Z2 board, using local BRAM-based weight reuse, tiled matrix multiplication, pipelined gate computation, and optimized activation functions.
3. Design Space Exploration: Different precision formats, matrix multiplication parallelism levels, and tanh and sigmoid implementations are evaluated together to identify a balanced design in terms of accuracy, speed, power, and hardware resources.
4. Practical Hardware Evaluation: The implemented accelerator achieves 0.9536 ms/sample, 1048.92 samples/s, 136 mW power consumption, and 4.28 GOPS/W, demonstrating its suitability for real-time edge-based SoC prediction.
5. Cost-Aware Assessment: The work also considers GOPS/W/$ to assess the practical value of using a low-cost FPGA platform for EV battery management applications.

The remainder of the paper is organized as follows, Section 2 reviews related work on battery SoC estimation and neural network hardware acceleration. Section 3 describes the LSTM-based SoC prediction model, and Section 4 presents the proposed hardware accelerator architecture. Section 5 discusses the design space exploration methodology. Section 6 presents experimental results and performance evaluation. Finally, section 7 concludes the paper, outlining future research directions.

2. Background and Motivation

Accurate estimation of the State-of-Charge (SoC) of Lithium-Ion Batteries in Electric Vehicles (EVs) is a fundamental requirement of an effective Battery Management System (BMS) used in these vehicles. Over the years, various approaches for SoC estimation have been explored by researchers.

With the rapid development of artificial intelligence, data-driven approaches have gained increasing attention for battery state estimation. Neural networks can capture non-linear relationships between current, voltage, temperature, aging, and SoC without detailed electrochemical modelling.

Among these methods, Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are well-suited for battery behavior modelling because they can learn how past operating conditions influence the present battery state. Use of the LSTM network in both its pure or vanilla form and hybrid form, where it is Combined with other DNN models, it is popular for SoC estimation. A comparative summary of representative studies on SoC estimation using LSTM is presented in Table 1.

Table 1. Comparative analysis of representative machine learning and deep learning models for Lithium-ion Batteries SoC estimation

Method / Model	Key Features	Accuracy	Limitations	Ref.
LSTM-RNN	Direct mapping of voltage, current, and temperature to SoC using deep recurrent networks	MAE ≈ 0.57%	High computational cost and large training data requirement	[13]
Stacked LSTM	Multi-layer LSTM model trained on dynamic driving cycles (DST, US06, FUDS)	RMSE < 2%, MAE < 1%	Requires extensive training data and long training time	[12]
CNN–LSTM	Combines CNN for spatial feature extraction and LSTM for temporal modeling	MAE < 1%, RMSE < 2%	Increased model complexity and computational overhead	[17]
LSTM Neural Network	Data-driven SoC estimation using voltage, current, and temperature from the PV battery system	MSE < 0.62%	Performance depends heavily on the quality of the training dataset	[14]
LSTM / BiLSTM + FPGA	AI-based BMS with FPGA implementation for SoC and SoH prediction	RMSE ≈ 0.947%	Limited exploration of hardware accelerator architecture	[11]
ML comparative study	Compares SVM, decision tree, ANN, and LSTM models for battery prediction	Deep learning models show superior performance	Requires large datasets and complex tuning	[15]
Multiple ML & DL models	Evaluates ANN, CNN, LSTM, SVR, and ensemble methods for BEV SoC prediction	Deep learning methods achieve higher accuracy	Generalization across battery chemistries remains challenging	[16]

As shown in Table 1, DNN networks, especially LSTM, outperform conventional model-based and ML approaches in estimating SoC. Furthermore, hybrid architectures such as CNN-LSTM improve the accuracy by combining the spatial feature extraction with the temporal feature extraction. However, complex LSTM networks face a major challenge of large computational and memory requirements, which can limit their usage in resource-constrained edge devices used in real-time BMS. These challenges highlight the importance of developing a hardware accelerator for real-time SoC prediction in edge devices. FPGA-based neural network accelerators employ optimization strategies at various levels to improve computational efficiency, reduce memory footprint, reduce memory bandwidth requirement, reduce power consumption, increase scalability, and achieve real-time inference. Researchers have attempted numerous techniques to optimize the designs at different level. The previous efforts to accelerate LSTM computations for different applications are outlined in Table 2. The optimization levels can be categorized into algorithm-level optimization, computational-level optimization, activation-function optimization, and dataflow optimization [37].

Figure 1 summarizes different types of optimizations used for LSTM at different levels. Algorithm-level techniques try to reduce the memory and the hardware by quantization and weight pruning, i.e., by removing nonsignificant weights or by compressing the model. Computation-level optimization techniques increase performance by exploiting the different computational patterns.

These include unrolling of loops, pipelining, parallel processing, or fusing different computations together. Memory-level optimization techniques improve performance by reducing the memory access time, primarily by utilizing the memory access patterns and localization. Localization helps in reducing the power consumption, as it avoids the frequent data transport, which is the major contributor to the power consumption. Activation function optimization focuses on utilizing different approximation techniques to shrink the memory footprint, reduce hardware, and minimize latency. Dataflow optimizations manage the flow of the data in order to improve the performance by minimizing the wait time of computational units and reducing the hardware by obviating the intermediate buffers.

Table 2. Comparative summary of optimization techniques used in FPGA-based LSTM accelerators

Techniques	Features	Pros	Cons	References
Loop unrolling	Replicates the loop body to process multiple iterations in parallel in HLS kernels	Increases parallelism and throughput; good for inner MAC loops	Higher LUT/DSP/BRAM use; long routes can hurt timing on large unroll factors	[22, 24]
Loop pipelining	Starts a new loop iteration every cycle (or a few cycles)	High throughput with limited hardware;	Pipeline depth and dependencies can limit the	[18, 21, 22, 24]

	via HLS PIPELINE	raises clock frequency; hides memory latency	initiation interval; harder timing closure	
Quantization / fixed-point arithmetic	Use fixed-point or mixed precision instead of FP32; sometimes, trainable/normalized schemes.	Cuts DSP and BRAM use; improves energy efficiency; enables full on-chip storage	Needs quantization-aware training; very low bits risk accuracy loss/gradient issues.	[18-21]
Weight pruning/sparsity exploitation	Structured or balanced sparsity in LSTM weights (row-balanced, block-diagonal, CBTD)	Fewer MACs and memory accesses; can reach >90% sparsity with small accuracy loss	Must redesign dataflow/indexing; irregular sparsity hurts PE utilization if not structured	[20, 23]
Matrix multiplication tiling/blocking	Partition matrices across time/feature dimensions; multi-level tiling to fit on-chip buffers	Fits large LSTMs on small FPGAs; improves data reuse and reduces DDR traffic	More complex control; sub-optimal tile sizes can underutilize PEs	[21-23]
Parallel PEs / systolic or MAC arrays	Spatial arrays for matrix–vector/matrix–matrix ops in LSTM gates	High parallelism and regular routing; amortizes memory cost over many MACs	Area and bandwidth scale with array size; may need careful mapping to hidden size	[18, 23, 24]
Memory partitioning/ banking	BRAM/URAM banking, array partitioning, ping-pong buffers	Increases memory bandwidth to PEs; reduces access conflicts and stalls	Consumes more BRAMs; bank conflicts are still possible with poor mapping	[21, 22, 24]
Activation function approximation	LUT / piecewise-linear tanh & sigmoid tuned to quantization	Saves DSPs and latency; better fits low-bit datapaths	Approximation error; must match numeric range	[18, 21]
Gate fusion optimization	Fuse i/f/o gates and part of cell update in shared matmul/post-processing hardware.	Reuses inputs and weights; reduces memory traffic and control overhead	Larger fused kernels complicate scheduling; they may reduce flexibility	[18, 23]
Streaming/weight- or output-stationary dataflow	Streaming dataflow across PEs with local buffers; weight/output stationary styles	High data reuse, low off-chip bandwidth, good for batch-1 LSTM	Requires careful design of on-chip buffers and interconnect	[18, 21, 23, 24]

Although LSTM-based models, especially hybrid models, achieve good accuracy in SoC prediction. But due to the computational complexity and dependencies across time instances, it limits its deployment at the edge. Existing FPGA-based implementations primarily focus on algorithm mapping on high-end FPGA hardware, rather than optimizing hardware microarchitecture for low-cost FPGAs. In particular, limited work has been done to predict the SoC of LIB in EVs, though it is essential for ensuring safe operation, accurate energy management, optimal performance, and lifespan control under dynamic and nonlinear operating conditions [25]. Also, limited efforts are made towards the deployment of LSTM hardware accelerators on resource-constrained low-cost FPGA platforms, which makes it more practical.

3. Hardware-Software Design Framework for LSTM-Based Prediction

Authors have proposed a hardware-software design framework for LSTM-based SoC prediction of LIB in EVs. The framework integrates data-driven modelling, hardware-friendly quantization-aware learning, and FPGA-based hardware acceleration to enable efficient and low-latency deployment in BMS at the edge. Unlike conventional approaches, the proposed framework jointly optimizes both aspects to achieve a balance between prediction accuracy, computational efficiency, and hardware resource utilization.

The overall workflow of the end-to-end pipeline that combines software-level model development with hardware-aware optimization and hardware implementation is outlined in Figure 2. The key objective of the framework is to allot tasks to the software and hardware components to ensure flexibility and performance.

3.1. Dataset Preparation

3.1.1. Dataset Selection

We have selected the NASA PCoE dataset as it is publicly available, well structured, rich in aging time-series, and heavily benchmarked, which makes it ideal for algorithm

development and comparison. These advantages far outweigh the limitations, such as limited operating diversity, specific chemistries, and lab-centric test profiles. The dataset consists of 2815 charge and discharge cycles and over 5 million data samples. Each sample has voltage, current, temperature, and time.

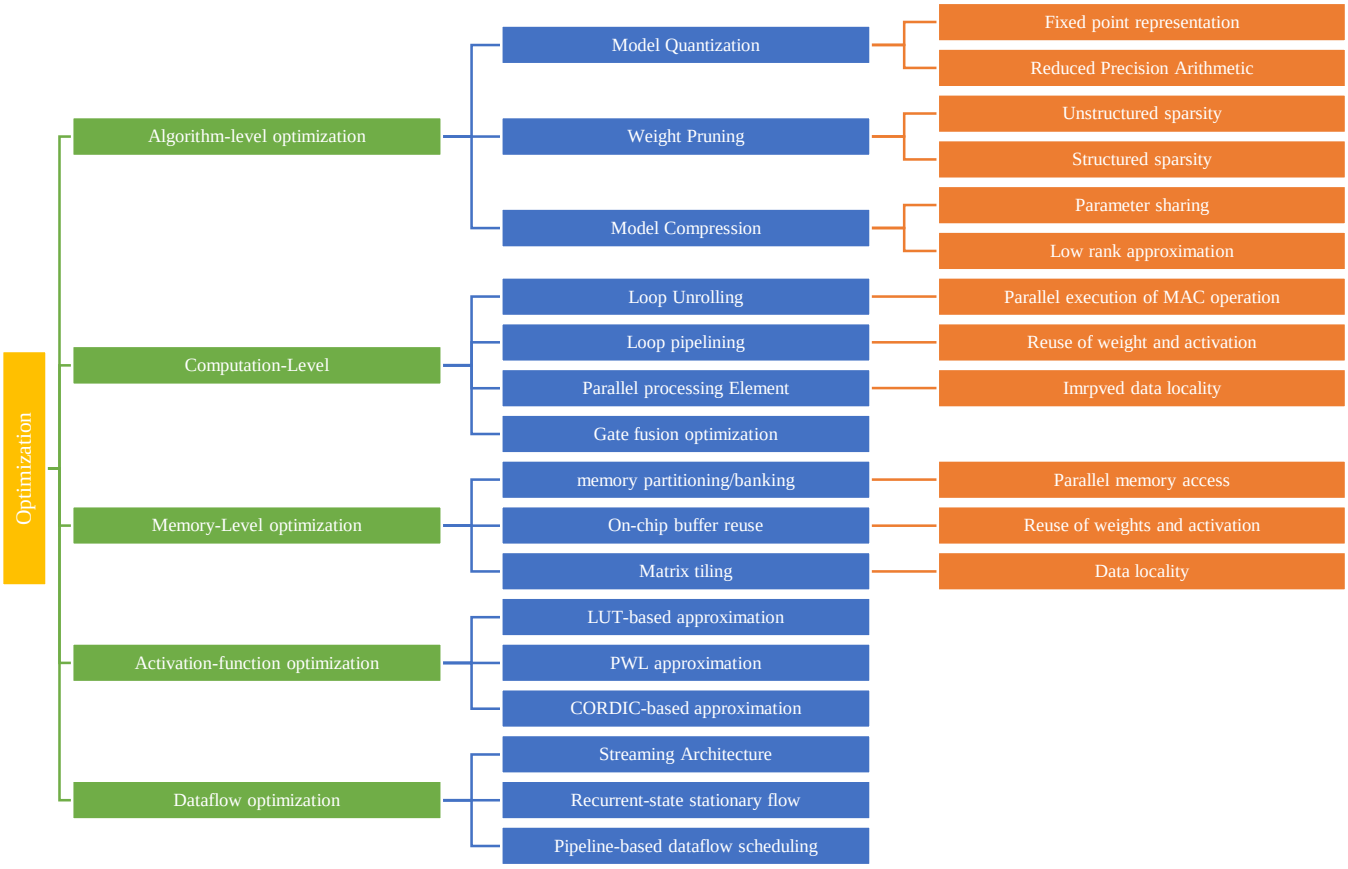


Fig. 1 Taxonomy of optimization techniques for FPGA-based LSTM accelerators, categorized into algorithm-level, computation-level, memory-level, activation-function, and dataflow optimizations

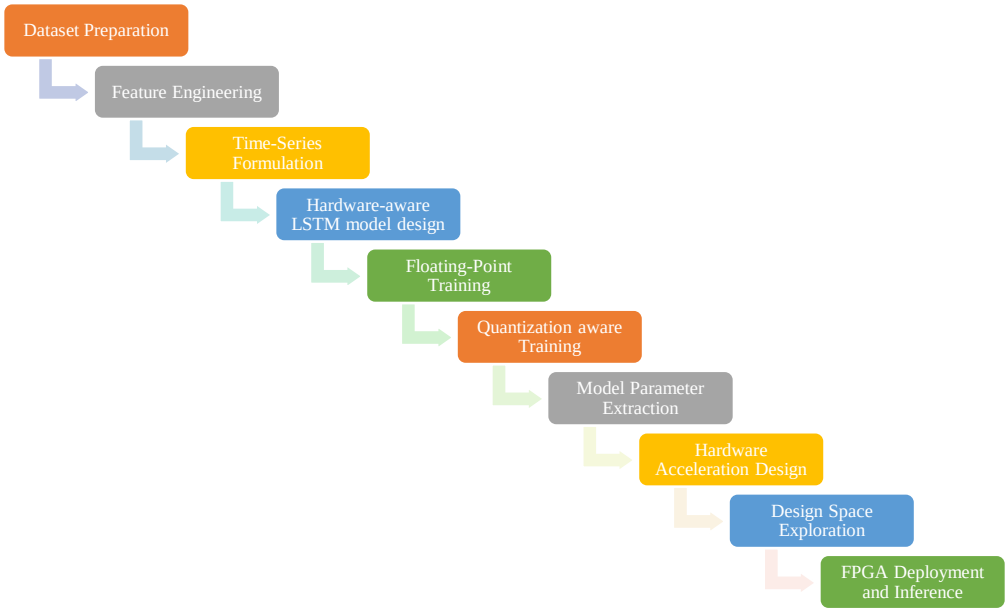


Fig. 2 End-to-end hardware–software co-design workflow for LSTM-based state-of-charge (SoC) prediction

3.1.2. Data Preprocessing

Raw data from the dataset, such as voltage, current, temperature, and time, are processed to compute the ground truth SoC using coulomb counting (Equation 1).

$$SOC = \frac{\int I(t)\, dt}{Qtotal} \quad (1)$$

To ensure the physical consistency, the compute SoC is forced to be monotonically increasing during charging and decreasing during discharging. Samples with very low current (0.05 A) are removed to filter out the noise during the idle state. The SoC estimation problem is formulated as a supervised sequence learning task using a sliding window approach.

$$X_t = f(x_{t-n}, x_{t-n+1}, \ldots, x_t) \quad (2)$$

Where n denotes the length of the sequence, each input sequence is mapped to the SoC value at the final timestep. This enables the model to capture the temporal dependencies in battery parameters. The choice of sequence length (n) is an important parameter, as it is pivotal in balancing temporal context capturing, model accuracy, and computational load.

3.2. Hardware-Aware LSTM model design

A hardware-aware architecture is designed to balance the prediction accuracy and computational efficiency on the FPGA. As shown in Figure 3, the model consists of two stacked LSTM layers (with 32 and 16 units), followed by a dense layer, and a sigmoid layer at the output for SoC prediction. The dimensions were fixed with the hyper parameter tuning with the aim of balancing between the accuracy and the computational cost. Hyperparameter tuning of the model is done to determine optimal network parameters, including the size of the feature set, sequence length, layer dimensions, hardware-friendly activation function, and data precision.

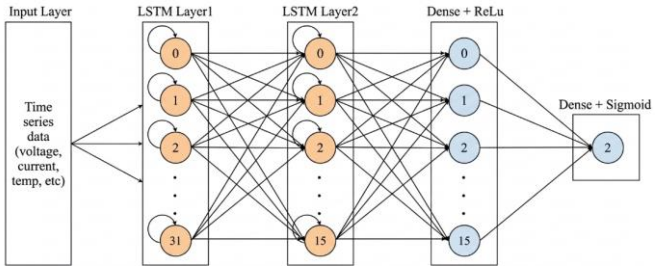


Fig. 3 The proposed hardware-aware LSTM network to predict the SoC of LIB

To enable the hardware-aware deployment, a reduced set of features is selected. The compact model (student) is evaluated against the comprehensive (Teacher) model. A comprehensive set of 43 features was created initially, including raw signals, first-order differentials, mean and standard deviation over a rolling window of size 10, lag features, charging phase indicator, charging progression, and battery identity code. As shown in Table 3, reducing the input feature space from 43 to 12 causes only a marginal degradation in MAE, RMSE, and R^2. The error distribution in Figure 4 further supports the observation. Both models exhibit highly overlapping absolute error distributions with no significant variation in the central tendency of the model. At the same time, it reduces the computational complexity significantly (~73%), as illustrated in equation 4.

For the LSTM layer,

Contribution to computation $\approx MAC_{input} \propto 4 \times F \times H$ (3)

Relative reduction =

$$\approx \frac{MAC-student}{MAC-teacher} = \frac{Fstudent}{Fteacher} = \frac{12}{43} \approx \mathbf{0.279} \quad (4)$$

Table 3. Performance comparison between the teacher (full-feature) and student (reduced-feature) LSTM models for SoC prediction

Model	No of features	MAE	RMSE	R2
Teacher	43	3.78%	5.79%	0.9551
Student	12	3.92%	5.82%	0.9546
Δ	31	0.14%	0.03%	0.0005

Sequence length is another significant parameter in deciding the accuracy and computational cost of the DNN model. The experiments were conducted with lengths n = 30, 60, 90, and 120 to evaluate R2 score, Mean Absolute Error (MAE), and inference per sample on Google Colab GPU, in order to decide the length of the window. The results are tabulated in Table 4. The table reveals the impact of the sequence length on the performance. As can be seen, reducing the sequence length has a marginal impact on the accuracy, but reduces the inference time per sample. Consequently, the sequence length of 30 has been selected for the design.

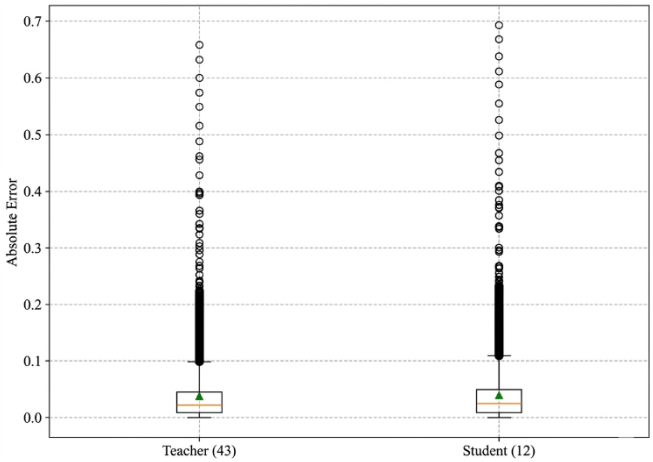


Fig. 4 Boxplot comparison of absolute prediction error distributions for teacher (43 features) and student (12 features) LSTM models

Precision of the data directly impacts the accuracy, memory footprint, hardware resources, power consumption,

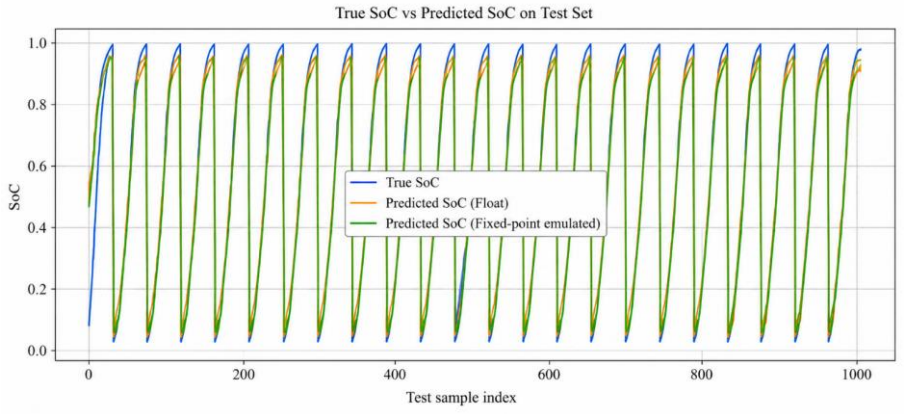
and the inference time of the system; it is one of the most important design parameters. A systematic precision impact analysis was conducted using divide-and-method. Starting from higher precision fixed point data type, ap_fixed<16,8>, progressively reduced to ap_fixed<8,4>, and ap_fixed<4,2>, were evaluated. MAE, RMSE, and R^2 score were used as the evaluation metrics, along with the graphical comparison of the actual SoC and predicted SoC. Results indicate that aggressive quantization (e.g., ap_fixed<4,2>) leads to significant accuracy degradation (MAE ≈ 7.7%), while higher precision formats (ap_fixed<16,8>) provide marginal improvement at the cost of increased hardware complexity. After the analysis, the 8-bit data type was finalized, based on the marginal compromise on accuracy and a potential large gain in terms of accuracy. Further, the resolution was determined by evaluating the impact of a change in the number of fractional bits in the fixed data type. Starting with the ap_fixed<8,4>, the MAE, RMSE and R^2 score were evaluated for ap_fixed<8,3> and ap_fixed<8,2>. The result demonstrated that ap_fixed<8,3> gives the best balance between accuracy and the resources (Figure 5). The results tabulated in Table 5 confirm the trend discussed above. The proposed hardware-software design framework systematically evaluates different parameters of the LSTM network at different stages of development. This leads to finalizing the different parameters, such as the size of the feature set, the number of layers, the size of layers, the precision configuration, etc., with the aim of balancing model accuracy while keeping the computation cost within limits. This framework facilitates the development of a DNN model that is sensitive to the resource-constrained hardware platform at the edge for SoC estimation in EVs.

Table 4. Impact of sequence length (n) on LSTM model performance, comparing prediction accuracy (R^2 score and MAE) and inference time per sample

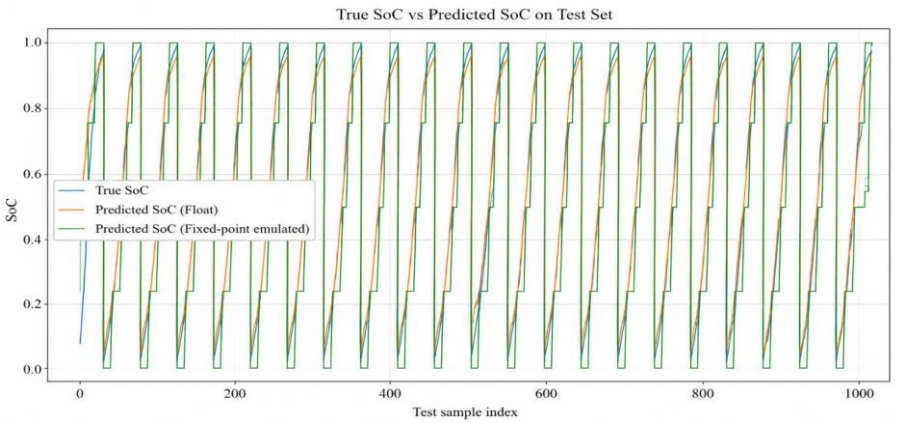
n	30	60	90	120
R^2 score	0.9592	0.954	0.957	0.919
MAE	0.0378	0.042	0.039	0.06
Inference/ sample (ms)	296.48	583.21	819.33	1092.182

Table 5. Impact of fixed-point data precision on LSTM model performance, comparing different ap_fixed configurations in terms of prediction accuracy (R^2, MAE, and RMSE)

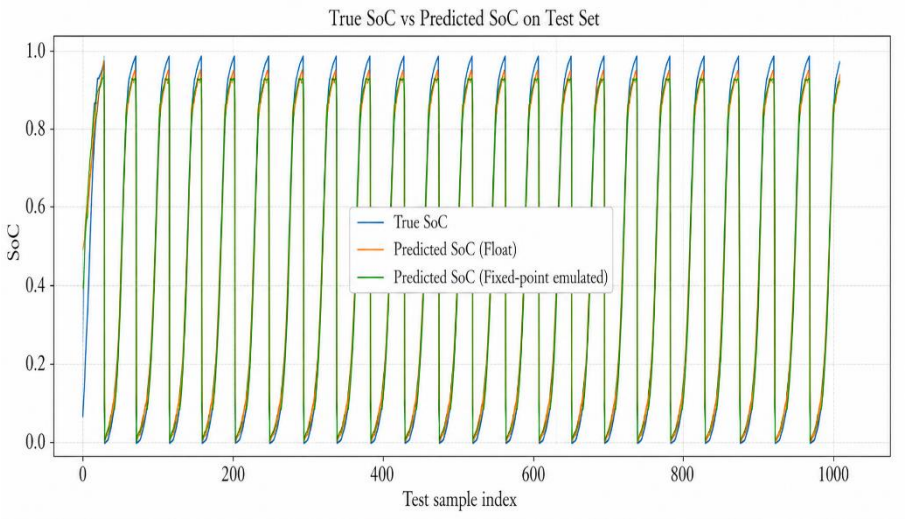
Data Type	ap_fixed< 16,8>	ap_fixed <8,4>	ap_fixed <4,2>	ap_fixed <8,3>	ap_fixed <8,2>
R^2	0.9566	0.9448	0.881	0. 9816	0.8887
MAE	0.0408	0.0514	0.0772	0.0361	0.0754
RMSE	0.0602	0.0679	0.0998	0. 0468	0.0964



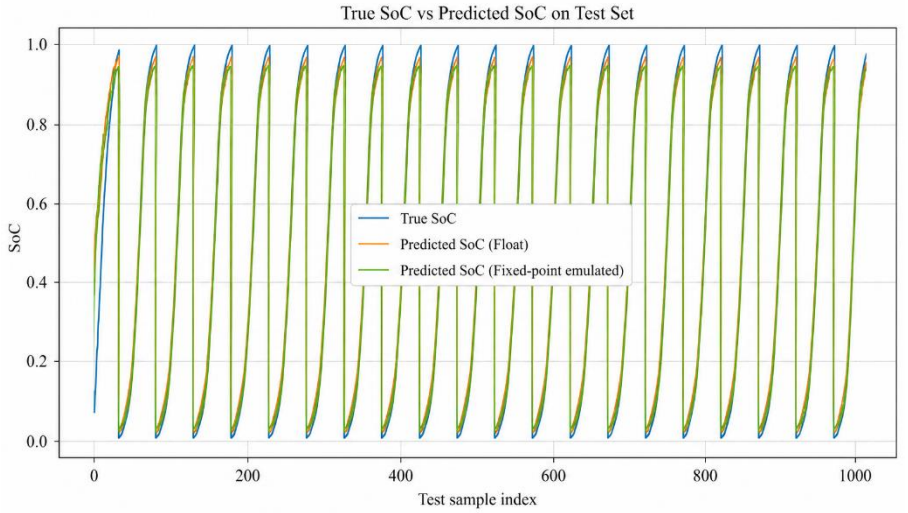
(a) SoC for ap_fixed<16,8>



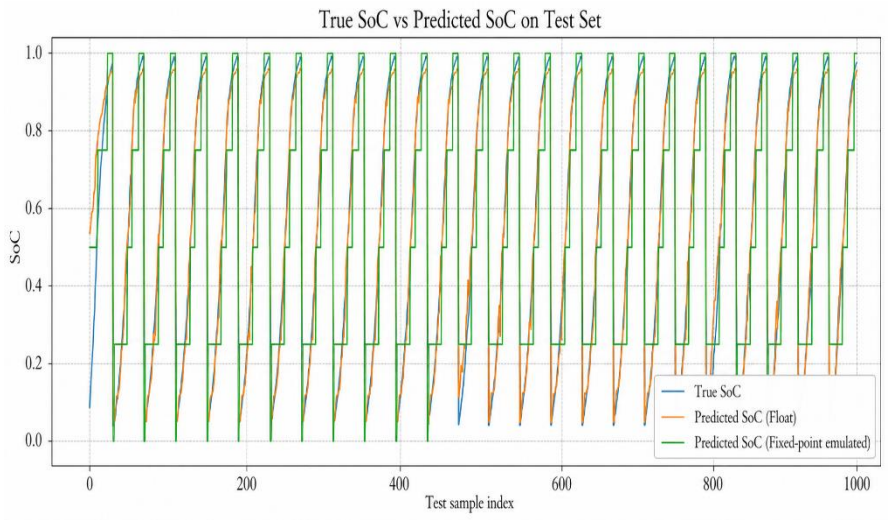
(b) SoC for ap_fixed<8,4>



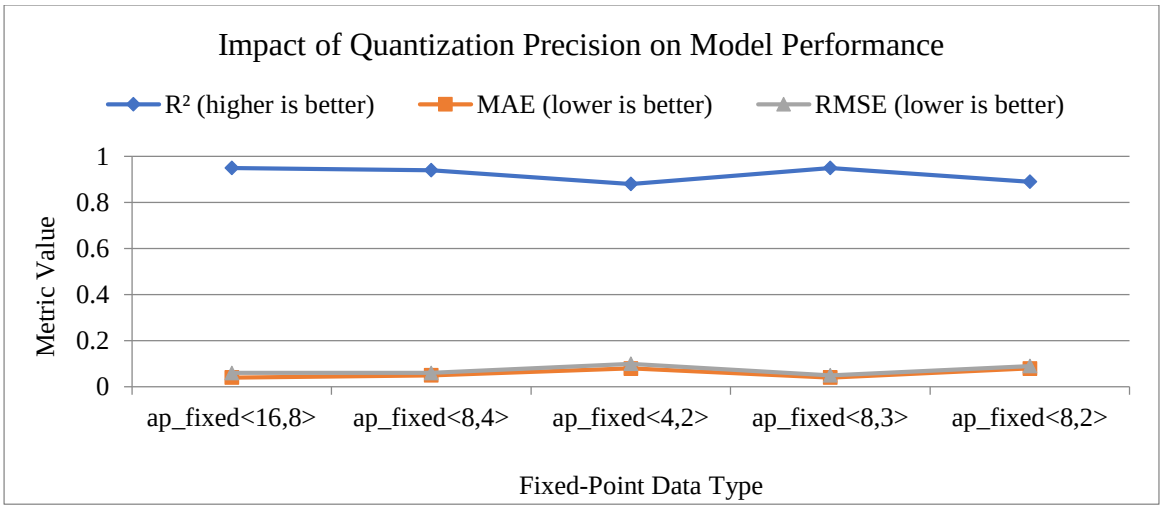
(c) SoC for ap_fixed<4,2>



(d) SoC for ap_fixed<8,4>



(e) SoC for ap_fixed<8,5>



(f) Performance comparison of the accuracy of different precision formats

Fig. 5 Impact of fixed-point quantization on LSTM-based SoC prediction performance. Subfigures (a)-(e) compare true and predicted SoC under different precision configurations. Subfigure, (f) summarizes the corresponding performance metrics.

4. Proposed FPGA-based Hardware Accelerator Architecture

A domain-specific FPGA-based hardware accelerator design, Memory-Centric Intelligent Reconfigurable Architecture (MIRA), is proposed to facilitate real-time deployment of the LSTM-based SoC prediction on resource-constrained BMS. The accelerator is designed following the hardware-software co-design methodology, optimizing the DNN model for efficient hardware execution while maintaining the prediction accuracy. In the first phase of the design framework, a hardware-aware DNN model is designed, and in the latter part, modularized hardware is designed in line with the DNN.

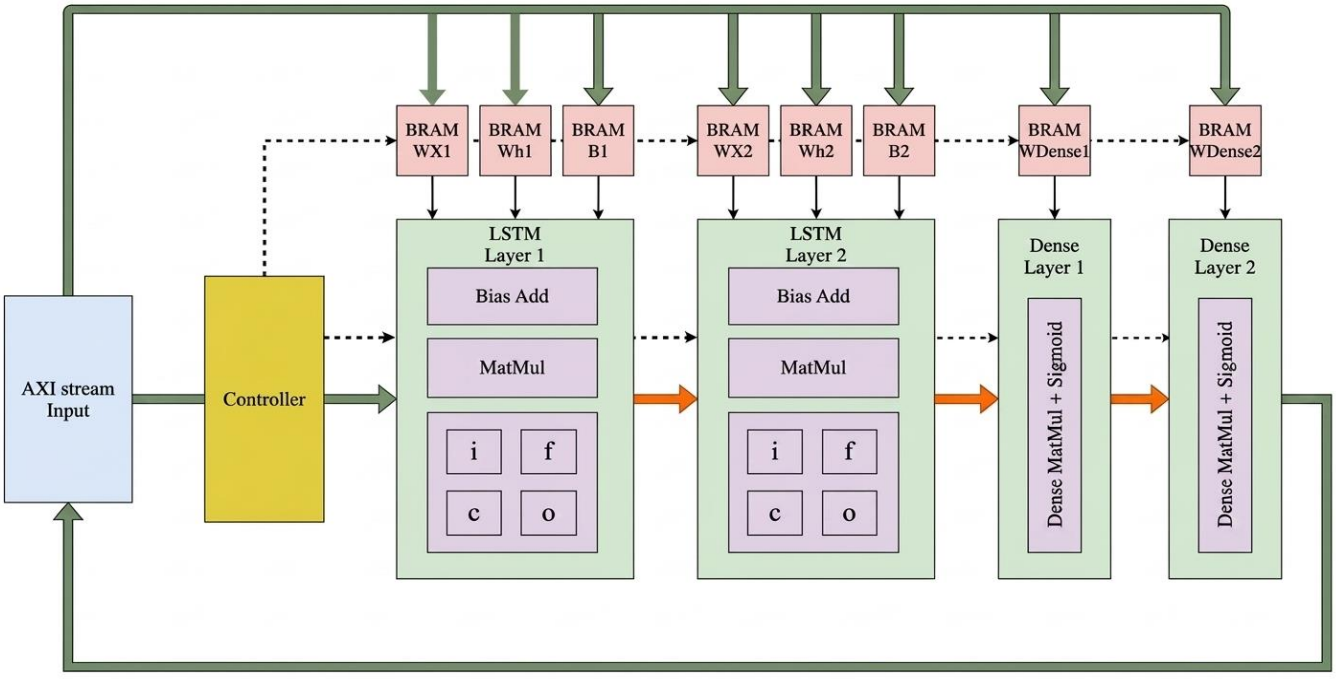


Fig. 6 Proposed hardware architecture of MIRA, where the yellow block denotes the control unit, green blocks represent computational modules, and purple blocks indicate compute engines. Olive green paths illustrate data movement, red paths indicate buffer-based data sharing, and dotted lines correspond to control signals.

The model is designed. Modularized design of the hardware provides an opportunity for Design-Space Exploration in order to optimize different modules for low latency, smaller memory requirements, low power

consumption, and low resource requirements. The accelerator targets the computational bottlenecks of LSTM inference, primarily matrix multiplication and the nonlinear activation function, by exploiting fine-grained parallelism and fixed-point arithmetic. To enhance the performance, fixed-point arithmetic is adopted, as it results in a significant reduction in latency, logic, and memory footprint [27]. Based on the analysis done in the first phase of the software-hardware design framework, the ap_fixed<8,3> format is selected for optimal precision, as it sufficiently captures the data range. The overall architecture consists of different modules: an input buffer and a preprocessing unit, an LSTM computer engine, a Dense layer engine, an Activation function Unit, an on-chip memory, and a control and scheduling unit. Figure 6 depicts the architecture of MIRA.

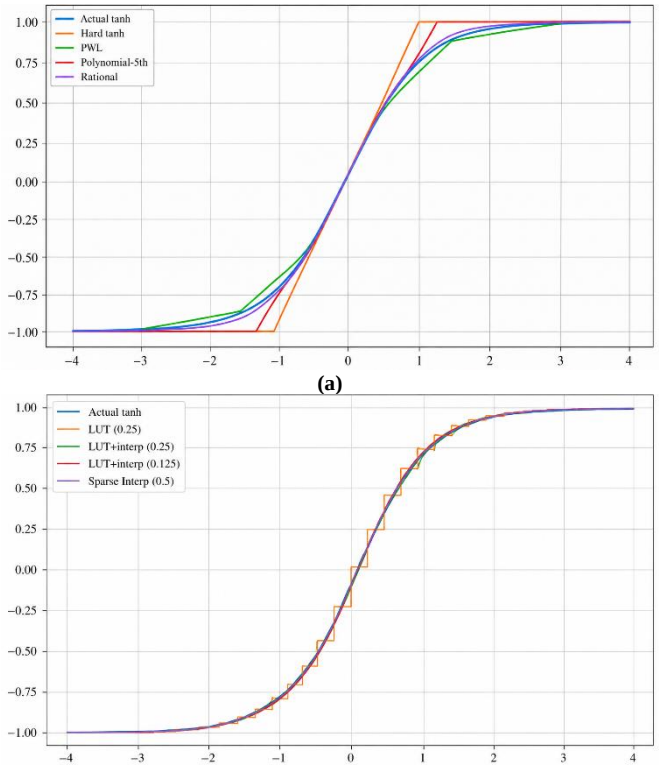


Fig. 7 True tanh vs tanh approximation techniques (a) Analytical approximation (b) LUT-based approximation

Considering the inherent computational complexity and interdependency of modules in LSTM, the hardware accelerator is designed using a hierarchical and iterative design methodology. Here, an initial system-level architecture is first established, followed by optimization through systematic design space exploration of individual modules. Each computational module is independently optimized, evaluating the impact on the performance and resource utilization of the entire architecture. The best performing configuration is selected before proceeding to the subsequent modules. The design space exploration focuses on three computational modules: tanh, sigmoid, and matrix multiplication. As activation functions have a direct impact on both prediction accuracy and hardware metrics, they are optimized prior to matrix multiplication.

4.1. Tanh Kernel Optimization

The hyperbolic function plays a critical role in LSTM cell update and significantly affects accuracy, latency, and hardware complexity. Multiple approximation techniques, including hard tanh, Piecewise Linear (PWL) approximation, LUT-interpolation hybrid methods, and rational function approximation, were selected based on the Python-based software evaluation (Figure 7). This ensures the balance between accuracy and computational simplicity.

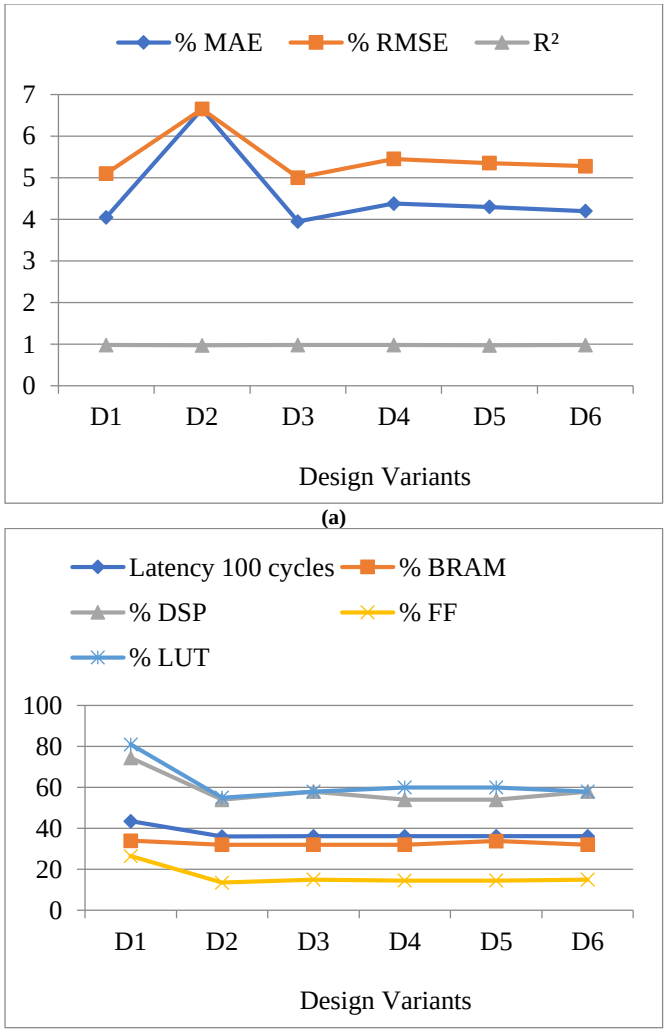


Fig. 8 Design space exploration for tanh computation, (a) Accuracy metrics, (b) Hardware cost.

A comprehensive design space exploration was conducted, synthesizing HLS implementation to evaluate these approaches across accuracy metrics (MAE, RMSE, and R^2 score), latency, and hardware cost (LUTs, FFs, DSPs, and BRAM). The results demonstrated the PWL approximation.

Table 6. Evaluation result of the design space exploration of tanh

Design	Optimization technique	Latency	BRAM	DSP	FF	LUT	MAE	RMSE	R^2 score
		Cycles	% utilization						
Dtan1	Baseline Library function	43447	33.93	75	26.519	80.73	0.0405	0.0512	0.9780
Dtan2	Hard tanh	35827	32.14	54	13.643	55.29	0.0665	0.0665	0.9628
Dtan3	**PWL**	**36127**	**32.14**	**58**	**15.025**	**58.44**	**0.0397**	**0.0503**	**0.9787**
Dtan4	LUT+interpolation (0.25 stepsize)	36187	32.14	54	14.517	59.52	0.0437	0.0547	0.9749
Dtan5	LUT+interpolation (0.125 stepsize)	36187	33.57	54	14.517	59.52	0.0429	0.0534	0.9760
Dtan6	Rational	36127	32.14	58	15.025	58.44	0.0420	0.0526	0.9767

Table 7. Evaluation result of the design space exploration of the sigmoid

Design	Optimization technique	Latency	BRAM	DSP	FF	LUT	MAE	RMSE	R^2 score
		Cycles	% utilization						
DSig1	Baseline	36127	32.14	58	15.02	58.44	0.0397	0.0503	0.9787
DSig2	**Hard Sigmoid**	**34110**	**32.14**	**45**	**9.03**	**31.05**	**0.0361**	**0.0468**	**0.9816**
DSig3	PWL	34112	32.14	45	9.77	35.42	0.0406	0.0512	0.9779
DSig4	LUT+interpolation 0.25 stepsize	34113	32.14	49	10.08	39.01	0.0398	0.0502	0.9788
DSig5	polynomial	34298	32.14	57	12.10	37.05	0.0443	0.0541	0.9754

Achieves the most balanced trade-off (Table 6), demonstrating the best MAE, while maintaining the hardware cost near the average (Figure 8). It was observed that the model exhibits a transient warm-up phase due to zero-initialized hidden states, resulting in higher prediction error in initial timesteps. To ensure fair evaluation, a burn-in period of 20 samples is excluded from metric computation.

4.2. Sigmoid Kernel Optimization

Following the selection of tanh implementation, a similar design space exploration was carried out for the activation function. Approximation techniques such as hard sigmoid, PWL, LUT interpolation, and polynomial are selected based on Python-based software evolution to ensure the balance between accuracy and computational simplicity (Figure 9).

Based on the comparative evaluation of the HLS implementation of the sigmoid kernel (Table 7), the hard sigmoid is selected as it offers a simpler realization with minimum hardware cost, yet provides the best MAE and RMSE values (Figure 10).

4.3. Matrix Multiplication Kernel Optimization

Matrix multiplication is the major computational bottleneck in the DNN accelerator, hence it is the main target of optimization. The majority of the delay and power consumption is attributed to the data transport. To address the memory-bound nature of the operations, a near-memory computation strategy is adopted, where weight matrices are stored in local BRAM and reused across multiple computation cycles. This weight-stationary dataflow is particularly favorable in recurrent networks like LSTM [22]. Parallelism in matrix multiplication is enhanced using a tiling strategy by further exploiting the data reuse. This not only does the load balancing but also reduces latency and energy consumption [29]. Within each tile, parallelism is further enhanced by processing multiple columns in a cycle. The extent of parallelism through tiling and the number of columns processed per cycle are evaluated and finalized by iteratively evaluating the performance and the hardware cost. Table 8 shows the impact of these parameters on the accuracy and the latency. As the requirement exceeds the available resource count, the optimization parameters are fixed.

Table 8. Design space exploration of matrix multiplication kernel configurations

Design	col_per_cycle_H	col_per_cycle_X1	col_per_cycle_X2	Latency	BRAM	DSP	FF	LUT
				cycles	% utilization			
DMatMul1	16	1	1	34110	32.14	45.45	9.03	31.05
DMatMul2		**2**	**2**	**31170**	**48.57**	**65.45**	**9.23**	**31.18**
DMatMul3		4	4	29730	80	*105.45*	9.48	36.08
DMatMul4		8	8	28890	*141.42*	*185.45*	25.29	59.00

Similarly, the dense layer matrix multiplication kernel is optimized using an iterative evaluation to identify the optimal tile size and the number of columns processed in parallel. Further, batch processing is employed to improve performance by enabling the simultaneous SoC prediction for multiple LIB cells, thereby increasing the throughput and reducing the average power consumption.

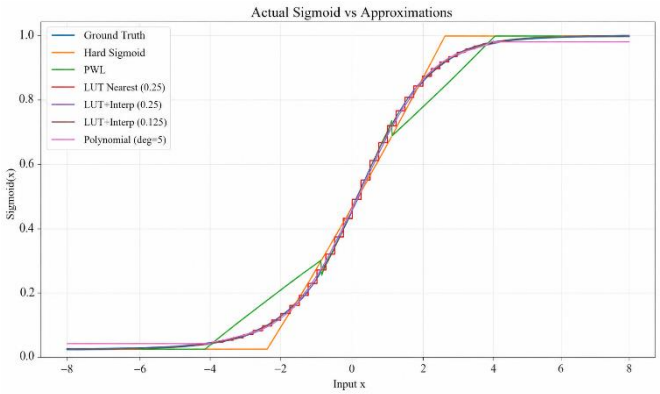


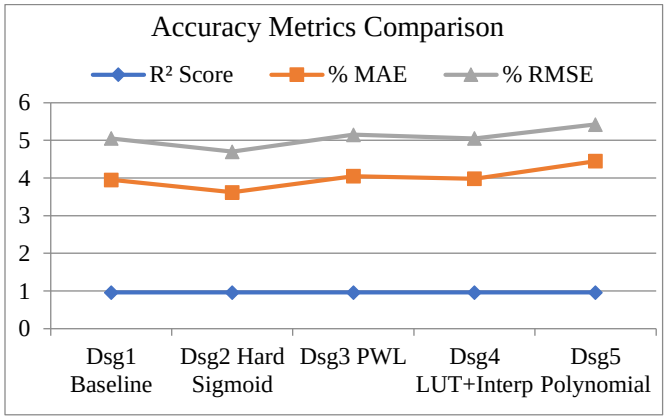
Fig. 9 True tanh Vs tanh approximation techniques

5. Experimental Evaluation and Analysis

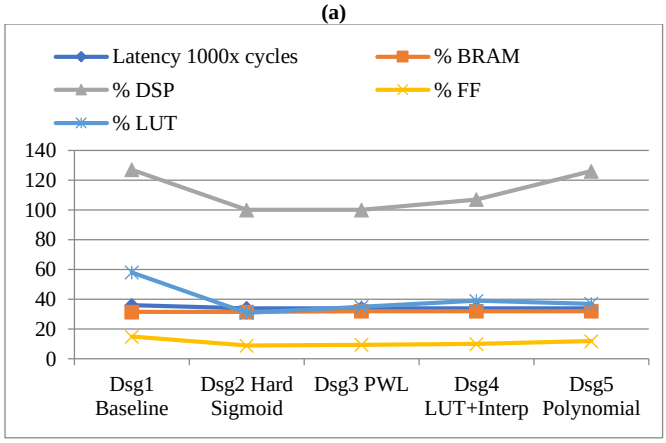
The proposed accelerator, MIRA, is evaluated thoroughly in terms of prediction accuracy, performance, and energy efficiency. The architecture is designed particularly for resource-constrained edge devices such as Pynq Z2.

5.1. Experimental Setup

The proposed architecture is implemented on the Xilinx Pynq Z2 platform, which features a Zynq- 7020 System-On-Chip (SOC). It consists of Programmable Logic (PL) and ARM-based Processing System (PS). The platform is selected due to its low cost, limited hardware resources, and suitability for edge deployment. This makes it a good candidate for evaluating the MIRA accelerator targeting resource-constrained low-cost FPGAs. The system design follows the software-hardware co-design to achieve maximum performance from the limited resources. The LSTM model configuration follows the hardware-aware design described in section 3. It consists of two stacked LSTM layers followed by dense layers and a sequence length of 30. The LSTM-based DNN models are trained using the NASA LIB dataset. Fixed-point arithmetic is determined in the software phase of the framework and employed in designing the hardware to balance the numerical accuracy and hardware efficiency. The architecture follows the weight-stationary dataflow to minimize the data traffic with the purpose of minimizing the latency and power consumption.



(a)



(b)

Fig. 10 Design space exploration for sigmoid computation, (a) Accuracy metrics, (b) Hardware cost.

The PS of the Zynq-7000 acts as a central processor, which is responsible for data collection from the sensors.

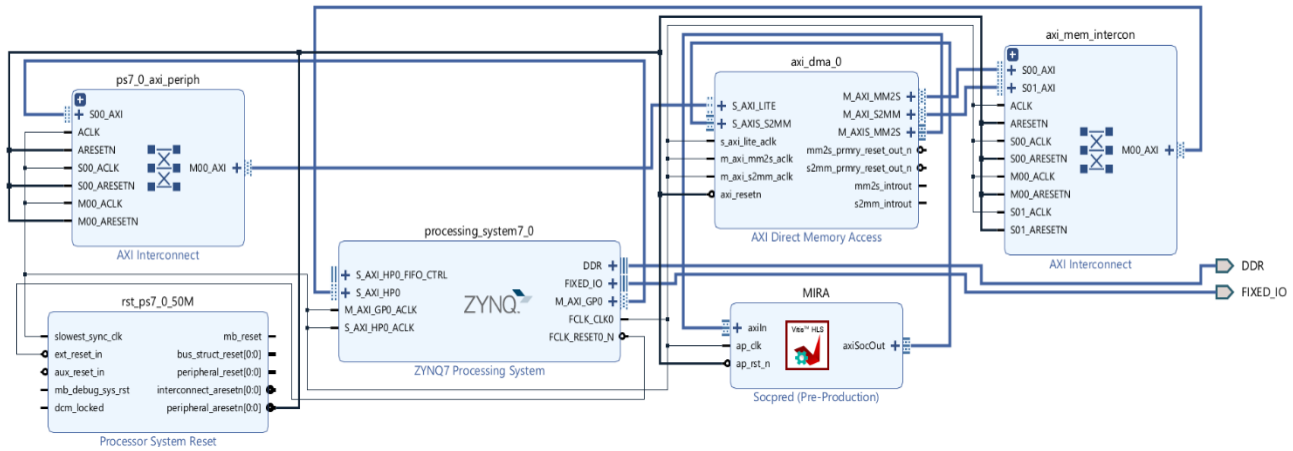


Fig. 11 The hardware-software co-design for SoC prediction. The PS of Zynq-7000 acts as a central processor while the hardware accelerator MIRA is deployed on the PL. PS and PL communicate with each other over AXI-DMA communication

Preprocessing of the data, and controlling the overall operation of the system (Figure 11). The compute-intensive SoC prediction is offloaded to the MIRA implemented on PL, which communicates to PS over AXI-DMA streams.

The system is evaluated using both algorithmic and hardware metrics. Prediction performance is quantified using Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and coefficient of determination (R^2). Hardware performance is evaluated in terms of latency (number of clock cycles), resource utilization, and throughput. Additionally, the energy efficiency metric GOPS/W is also used to compare with the existing system. A novel metric, GOPS/W/$, is devised to enable fair and meaningful comparison across the hardware platforms. The metric facilitates the benchmarking of resource-constrained, low-cost systems against the high-end, expensive devices. This metric also depicts the practicality of using the device at the edge.

5.2. Prediction Accuracy Analysis

The prediction accuracy of the proposed hardware accelerator is appraised by comparing it against the software baseline (Table 9). The result indicates that there is marginal degradation in comparison with the floating-point model. The selection of the activation function plays an important role in maintaining accuracy. It is observed that tanh and sigmoid activation functions provide superior performance compared to hardware-friendly counterparts. Also, the design space exploration done has facilitated the selection of implementation approaches that can balance accuracy with hardware efficiency. Overall, the hardware accelerator achieves high correlation with the ground truth. The results approve the effectiveness of the hardware-software design framework.

5.3. Hardware Resource Utilization Analysis

Resource utilization of the hardware design synthesized in Vivado is presented in Figure 12. The results demonstrate the successful deployment on a low-cost platform such as Pynq Z2. The adoption of fixed-point arithmetic and approximation techniques used for the realization of the activation functions significantly reduces the hardware resource consumption. As can be seen from Figure 12, BRAM and DSP have dominant resource usage, mainly because of the parallelization in the matrix multiplication engine. Despite this careful tuning of tiling and column-level parallelism, the design ensures that it balances performance and resource usage. Further parallelization is restricted due to the limited resources on the platform, which consequently hinders performance.

Table 9. Comparative analysis of predictive accuracy of MIRA with software baseline

MAE	RMSE	R^2 score
Software baseline	0.0346	0.0429
MIRA result	0.0361	0.0468
Percentage change	4.34	9.09

5.4. Performance and Energy-Efficiency Analysis

The performance of the proposed architecture, MIRA, is evaluated in terms of latency, throughput, and energy efficiency with a unified analysis using GOPS and GOPS/W. Latency is the critical aspect of real-time edge devices. The average latency for MIRA is observed for the computation of a single sample, which is 0.9536 ms, giving the throughput of 1048.92 samples/second. In contrast, the average inference time of the CPU is 180 ms per sample, highlighting the significant improvement in performance. The rise in performance is achieved through fixed-point arithmetic, near-memory computation, efficient tiling, parallel column-wise processing, and hardware-friendly approximation of the activation functions. Energy efficiency is another critical factor in stand-alone edge devices operating under a limited power budget. MIRA consumes 136 mW of power, accounting for only 7.36% of the total power consumed by the platform. Energy efficiency is evaluated using GOPS/W. The proposed accelerator design achieves an energy efficiency of 4.28, owing to the restricted

Table 10. Comparison of FPGA-based LSTM accelerator architectures across different platforms

System	Platform	GOPS/W	$*	GOPS/W/$
Stall-free blocking and batching [30]	VX690T	20.8	8094	0.0025
POLAR [31]	XC7Z045	22.90	8486	0.0026
E-LSTM [32]	SX660	17.74	5845	0.0030
Balanced Sparsity [20]	XCKU115	59.33	11996	0.0049
Stall free Blocking and batching Accel – II [30]	7Z045	13.48	2325	0.0057
BRDS [35]	XCKU9P	22.22	3679	0.0060
Streaming Overlay Accel [34]	XCZU7EV	65.53	5200	0.0126
MIRA (This work)	**Pynq Z2**	**4.28**	**129**	**0.0331**

*All prices are referred from AMD websites or their official distributors' websites

Power consumption due to fixed-point arithmetic and near-memory computation. Furthermore, to enable fair comparison across different hardware platforms, a cost-aware metric, GOPS/W/$, is introduced, incorporating cost alongside performance and energy efficiency. While the performance can be improved by bringing in more parallelization, it is constrained in the low-cost platforms due to the limited resources. In contrast, high-end platforms can

exploit the maximum parallelization in the algorithm to increase the performance, due to the abundance of resources. Restrictions in the low-cost devices make the metric GOPS/W/$ just for gauging the design aptness. To contextualize the MIRA, Table 10 compares it with the FPGA-based LSTM accelerator. While high-end platforms can achieve superior GOPS/W due to abundant hardware resources, they incur a higher cost. The proposed design demonstrates competitive GOPS/W/$ of 0.0331, outperforming all compared works (Figure 13). These results demonstrate that the proposed design is particularly well-suited for the resource-constrained edge BMS applications, where cost is a critical consideration. This further validates the importance of incorporating cost-aware metrics in accelerator evaluation, particularly for real-world embedded and automotive applications.

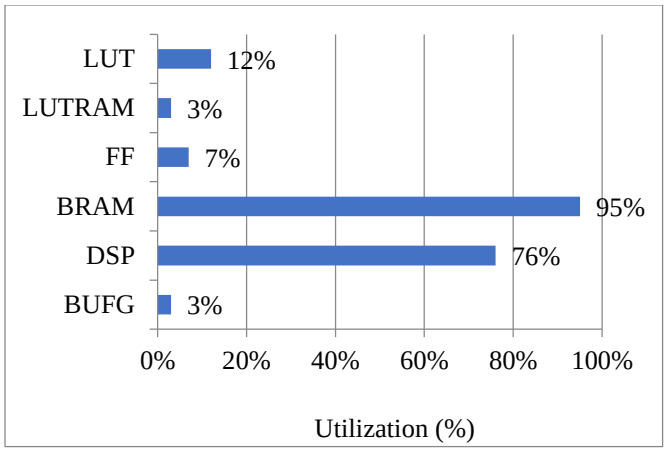


Fig. 12 Hardware resources utilization of the design synthesized in Vivado

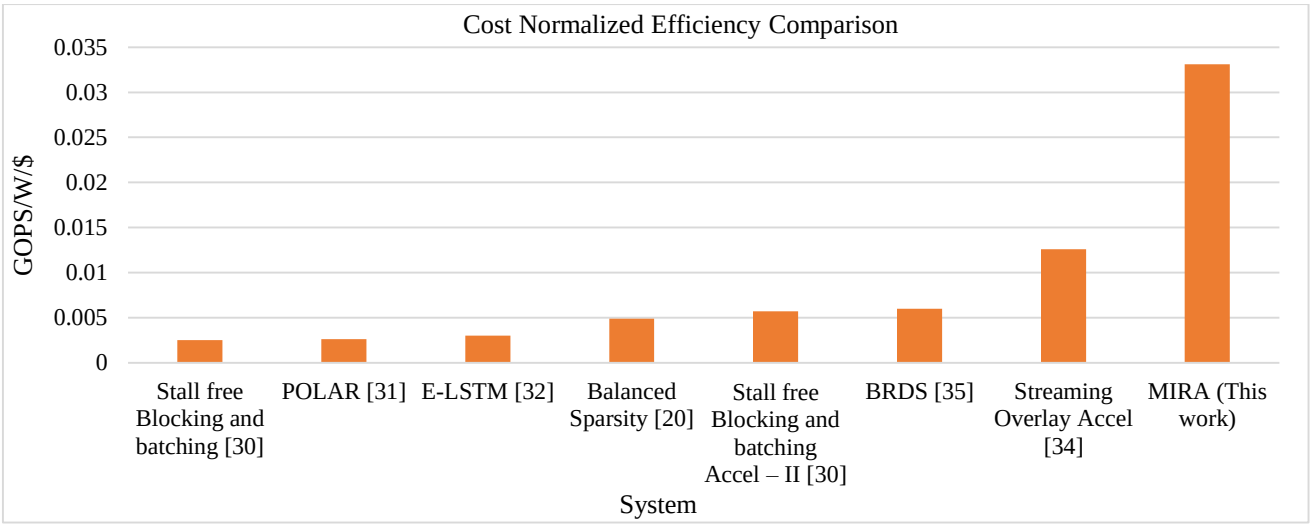


Fig. 13 Comparison of MIRA with the other hardware architectures

The experimental results provide several important insights. A hardware-friendly approximation of the activation function is crucial in balancing accuracy and the hardware cost. Memory-centric optimization is more impactful than compute scaling for energy efficiency. The degree of parallelism can be increased due to fixed-point arithmetic by reducing the hardware cost. Overall, the proposed FPGA-based LSTM accelerator achieves an effective balance between latency, resource utilization, accuracy, and practicability to use at the edge due to low cost.

6. Conclusion

This paper presents MIRA, a hardware–software co-designed FPGA-based accelerator for real-time state-of-charge (SoC) prediction of Lithium-ion Batteries in electric vehicles. The proposed framework integrates data-driven LSTM modeling with hardware-aware optimizations, including feature reduction, sequence tuning, fixed-point quantization, activation function approximation, and parallelized matrix multiplication.

A systematic design space exploration is conducted to identify optimal configurations that balance prediction accuracy, latency, energy efficiency, and hardware resource utilization.

The accelerator is implemented on a low-cost PYNQ-Z2 platform, demonstrating its suitability for resource-constrained edge deployment. Experimental results show that the proposed design achieves a latency of 0.9536 ms per sample and a throughput of 1048.92 samples/s, significantly outperforming CPU-based implementations. The system operates at a low power consumption of 136 mW, achieving an energy efficiency of 4.28 GOPS/W.

Furthermore, the introduction of the GOPS/W/$ metric enables fair comparison across heterogeneous platforms, where the proposed design achieves superior cost-normalized efficiency compared to existing FPGA-based accelerators. These results validate that the proposed approach effectively balances performance, energy efficiency, and cost, making it

well-suited for real-time embedded battery management systems. The study also highlights key insights: (i) hardware-friendly activation function approximation plays a crucial role in balancing accuracy and resource utilization, (ii) memory-centric optimization is more impactful than compute scaling for energy efficiency, and (iii) fixed-point quantization enables higher parallelism within constrained resources.

Future work will focus on extending the proposed framework to support multi-cell battery pack-level estimation, incorporating state-of-health (SoH) prediction, and exploring adaptive precision techniques for further optimization. Additionally, deployment on newer FPGA platforms and integration with real-world EV datasets will be investigated to enhance scalability and robustness.

References

[1] Raja Rajendra Timilsina et al., "Global Drive toward Net-Zero Emissions and Sustainability via Electric Vehicles: An Integrative Critical Review," *Energy, Ecology and Environment*, vol. 10, pp. 125-144, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Erdi Tosun et al., "Evaluation of Lithium-ion Batteries in Electric Vehicles," *International Journal of Automotive Science and Technology*, vol. 8, no. 3, pp. 332-340, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ingvild B. Espedal et al., "Current Trends for State-of-Charge (SoC) Estimation in Lithium-Ion Battery Electric Vehicles," *Energies*, vol. 14, no. 11, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Anith Khairunnisa Ghazali, Nor Azlina Ab. Aziz, and Mohd Khair Hassan, "Advanced Algorithms in Battery Management Systems for Electric Vehicles: A Comprehensive Review," *Symmetry*, vol. 17, no. 3, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Zuolu Wang et al., "A Review on Online State of Charge and State of Health Estimation for Lithium-Ion Batteries in Electric Vehicles," *Energy Reports*, vol. 7, pp. 5141-5161, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Dickson N. T. How et al., "State of Charge Estimation for Lithium-Ion Batteries Using Model-Based and Data-Driven Methods: A Review," *IEEE Access*, vol. 7, pp. 136116-136136, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Adolfo Dannier et al., "Li-Ion Batteries for Electric Vehicle Applications: An Overview of Accurate State of Charge/State of Health Estimation Methods," *Energies*, vol. 18, no. 4, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[8] Ruifeng Zhang et al., "State of the Art of Lithium-Ion Battery SOC Estimation for Electrical Vehicles," *Energies*, vol. 11, no. 7, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Xinyue Liu et al., "Advances in the Study of Techniques to Determine the Lithium-Ion Battery's State of Charge," *Energies*, vol. 17, no. 7, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ming Zhang, Kai Wang, and Y. Zhou, "Online State of Charge Estimation of Lithium-Ion Cells Using Particle Filter-Based Hybrid Filtering Approach," *Complexity*, vol. 2020, no. 1, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Satyashil D. Nagarale, and B. P. Patil, "Accelerating AI-Based Battery Management System's SOC and SOH on FPGA," *Applied Computational Intelligence and Soft Computing*, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Fangfang Yang et al., "State-of-Charge Estimation of Lithium-Ion Batteries via Long Short-Term Memory Network," *IEEE Access*, vol. 7, pp. 53792-53799, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ephrem Chemali et al., "Long Short-Term Memory Networks for Accurate State-Of-Charge Estimation of Li-Ion Batteries," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6730-6739, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[14] Eyad Almaita et al., "State of Charge Estimation for a Group of Lithium-Ion Batteries Using Long Short-Term Memory Neural Network," *Journal of Energy Storage*, vol. 52, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Saadin Oyucu et al., "Comparative Analysis of Commonly Used Machine Learning Approaches for Li-Ion Battery Performance Prediction and Management in Electric Vehicles," *Applied Sciences*, vol. 14, no. 6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ehab Issa El-Sayed, Salah K. ElSayed, and Mohammad Alsharef, "Data-Driven Approaches for State-of-Charge Estimation in Battery Electric Vehicles Using Machine and Deep Learning Techniques," *Sustainability*, vol. 16, no. 21, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] Xiangbao Song et al., "Combined CNN-LSTM Network for State-of-Charge Estimation of Lithium-Ion Batteries," *IEEE Access*, vol. 7, pp. 88894-88902, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[18] Dazhong He et al., "An FPGA-Based LSTM Acceleration Engine for Deep Learning Frameworks," *Electronics*, vol. 10, no. 6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Shashwat Khandelwal et al., "FINN-GL: Generalized Mixed-Precision Extensions for FPGA-Accelerated LSTMs," *arXiv*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Nianyi Wang et al., "A Compression Strategy to Accelerate LSTM Meta-Learning on FPGA," *ICT Express*, vol. 8, no. 3, pp. 322-327, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Chao Qian, Tianheng Ling, and Gregor Schiele, "Exploring Energy Efficiency of LSTM Accelerators: A Parameterized Architecture Design for Embedded FPGAs," *Journal of Systems Architecture*, vol. 152, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[22] Richie Li, "Dataflow & Tiling Strategies in Edge-AI FPGA Accelerators: A Comprehensive Literature Review," *arXiv*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[23] Chang Gao, Tobi Delbruck, and Shih-Chii Liu, "Spartus: A 9.4 TOp/s FPGA-Based LSTM Accelerator Exploiting Spatio-Temporal Sparsity," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 1098-1112, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[24] Anouar Nechi et al., "FPGA-Based Deep Learning Inference Accelerators: Where Are We Standing?," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 4, pp. 1-32, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Rui Xiong et al., "Critical Review on the Battery State of Charge Estimation Methods for Electric Vehicles," *IEEE Access*, vol. 6, pp. 1832-1843, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ming Zhang et al., "A Review of SOH Prediction of Li-Ion Batteries Based on Data-Driven Algorithms," *Energies*, vol. 16, no. 7, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[27] Don Lahiru Nirmal Hettiarachchi, Venkata Salini Priyamvada Davuluru, and Eric J. Balster, "Integer vs. Floating-Point Processing on Modern FPGA Technology," *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, pp. 606-612, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[28] Fabian Schuiki et al., "A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 484-497, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[29] Gordon Euhyun Moon et al., "Evaluating Spatial Accelerator Architectures with Tiled Matrix-Matrix Multiplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 1002-1014, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[30] Zhiqiang Que et al., "Mapping Large LSTMs to FPGAs with Weight Reuse," *Journal of Signal Processing Systems*, vol. 92, no. 9 pp. 965-979, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[31] B Erfan Bank-Tavakoli et al., "Polar: A Pipelined/Overlapped FPGA-Based LSTM Accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 3, pp. 838-842, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[32] Meiqi Wang et al., "E-LSTM: An Efficient Hardware Architecture for Long Short-Term Memory," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 280-291, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[33] Shaorun Wang et al., "Acceleration of LSTM with Structured Pruning Method on FPGA," *IEEE Access*, vol. 7, pp. 62930-62937, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[34] Lenos Ioannou, and Suhaib A. Fahmy, "Streaming Overlay Architecture for Lightweight LSTM Computation on FPGA SoCs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 1, pp. 1-26, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[35] Seyed Abolfazl Ghasemzadeh et al., "BRDS: An FPGA-Based LSTM Accelerator with Row-Balanced Dual-Ratio Sparsification," *arXiv*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[36] Maheshwari Adaikkappan, and Nageswari Sathiyamoorthy, "Modeling, State of Charge Estimation, and Charging of Lithium-Ion Battery in Electric Vehicle: A Review," *International Journal of Energy Research*, vol. 46, no. 3, pp. 2141-2165, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[37] Li Gao, Zhongqiang Luo, and Lin Wang, "Convolutional Neural Network Acceleration Techniques Based on FPGA Platforms: Principles, Methods, and Challenges," *Information*, vol. 16, no. 10, 2025. [CrossRef] [Google Scholar] [Publisher Link]