RESEARCH ARTICLE

MARTE-Based Modeling and Analysis for Real-Time Neuromorphic Computing in Embedded Systems

Liangshun Wu[1,*] **and Tao Tao**[2]

[1] Shanghai Key Laboratory of Trustworthy Computing,East China Normal University, Shanghai 200062, China
[2] School of Integrated Circuits(School of Information Science and Electronic Engineering), Shanghai Jiao Tong University, Shanghai 200240, China

Abstract

With the rapid advancement of deep learning, Spiking Neural Networks (SNNs) have attracted growing interest due to their low power consumption, sensitivity to temporal information, and biological plausibility. However, deploying SNNs in resource-constrained, real-time embedded environments presents significant challenges—chiefly their complex training processes, limited hardware acceleration support, and the difficulty of performing scheduling analysis. This paper presents an integrated modeling and scheduling analysis framework for SNNs based on the MARTE (Modeling and Analysis of Real-Time and Embedded Systems) standard defined by the OMG. Key SNN components—such as neurons, synapses, and spike events—are mapped to schedulable tasks and communication resources within the MARTE profile. Leveraging the Papyrus MARTE tool, we conduct simulation and verification on a heterogeneous embedded platform comprising multi-core ARM and DSP processors. Experimental results show that the proposed framework effectively satisfies end-to-end latency and power constraints, while significantly reducing system integration risks and enhancing design efficiency. Finally, we discuss future research directions, including support for more complex SNN architectures, advanced scheduling strategies, deployment on heterogeneous and distributed platforms, and formal verification for safety-critical applications.

Keywords: spiking neural network (SNN), MARTE, embedded system, real-time scheduling, model-driven engineering (MDE), energy optimization, papyrus MARTE, system modeling and verification.

1 Introduction

With the rise of deep learning, neural networks have achieved remarkable breakthroughs in fields such as image recognition, natural language processing, and autonomous driving. However, traditional artificial neural networks (ANNs) are computationally intensive and typically depend on GPUs or specialized accelerators to perform large-scale processing efficiently. In contrast, spiking neural networks (SNNs)—inspired by the behavior of biological neurons—employ event-driven processing to minimize redundant computations [1]. This leads

Submitted: 16 June 2025
Accepted: 11 July 2025
Published: pub-date

Vol. 1, **No.** 1, 2025.
10.62762/JSE.2025.495949

***Corresponding author:**
✉ Liangshun Wu
wuliangshun@sjtu.edu.cn

Citation
Wu, L., & Tao, T. (2025). MARTE-Based Modeling and Analysis for Real-Time Neuromorphic Computing in Embedded Systems. *ICCK Journal of Software Engineering*, 1(1), 9–16.

to lower power consumption, greater sensitivity to temporal dynamics, and enhanced biological plausibility, making SNNs an increasingly promising choice for energy-efficient intelligent systems.

The increasing adoption of embedded systems in real-time applications—such as industrial automation, UAVs, and robotics—calls for the reliable deployment of neural networks under stringent timing and resource constraints. The MARTE (Modeling and Analysis of Real-Time and Embedded Systems) standard, developed by the OMG, provides a unified framework for modeling, analyzing, and verifying such systems. By incorporating SNN architectures, execution timing, and hardware mappings into the MARTE framework, designers can achieve more systematic support for system design along with enhanced verification capabilities.

This paper reviews the foundational concepts of Spiking Neural Networks (SNNs) and the MARTE standard, identifies key challenges in deploying SNNs within real-time embedded environments, and introduces a MARTE-based modeling and analysis framework. Preliminary experimental results are presented, followed by a discussion of future research directions.

2 Preliminaries and Related Work

2.1 Spiking Neural Network (SNN)

SNNs are a class of neural models that transmit and process information via discrete-time spikes. Representative models include the Leaky Integrate-and-Fire (LIF) and Izhikevich models [2]. Unlike traditional deep neural networks (DNNs), SNN neurons fire only when their membrane potential exceeds a threshold, introducing intrinsic temporal dynamics. This event-driven mechanism enables low power consumption, real-time responsiveness, and greater biological plausibility.

Despite these advantages, SNNs face several limitations in terms of hardware and algorithmic maturity:

1. Traditional backpropagation is not directly applicable due to the non-differentiable nature of spike generation, necessitating the use of surrogate gradients or alternative training methods.
2. SNNs often require specialized neuromorphic hardware to support spike-based computation, which is difficult to optimize for resource-constrained embedded platforms.
3. Ensuring that SNN inference meets stringent timing requirements remains a key challenge in time-critical applications.

2.2 MARTE

MARTE (Modeling and Analysis of Real-Time and Embedded Systems), developed by the Object Management Group (OMG), is a UML-based standard that enhances modeling capabilities for real-time system properties such as timing, scheduling, and resource management. Its key features include:

1. Specification and annotation of timing constraints, such as Worst-Case Execution Time (WCET) [3], and support for periodic scheduling.
2. Modeling of hardware components (e.g., CPUs, memory), facilitating hardware-software co-analysis.
3. Application of real-time scheduling theory for evaluating task feasibility and timing conformance.
4. Early-stage architectural and scheduling evaluation through high-level system models.

MARTE incorporates profiles such as Generic Resource Modeling (GRM), Software Resource Modeling (SRM), and Hardware Resource Modeling (HRM), enabling a unified, system-level description of applications, platform resources, and scheduling strategies. These features support early trade-off analysis for both functional and non-functional system requirements [4].

2.3 Research Progress

Recent advances in integrating machine learning and neural networks into real-time systems have primarily focused on two directions:

1. Lightweight deployment: Implementing neural networks on resource-constrained hardware such as microcontrollers or FPGAs to enable efficient processing of image, audio, or sensor data [5].
2. Model-driven optimization: Utilizing MARTE or other UML-based approaches to model timing and resource constraints, enabling scheduling analysis and early validation of system design [6].

UML models help capture key architectural decisions, their interdependencies, and influencing factors, thereby promoting structured system design [7]. Extensions through domain-specific profiles allow quality attributes—such as performance and reliability—to be integrated early in the design process [8]. Combined with platforms like Apache Hadoop, these models support simulation, bottleneck detection, and resource optimization before deployment [9]. Moreover, UML-based decision models aid in navigating architectural trade-offs based on empirical evidence and system objectives [10].

While significant progress has been made with deep neural networks, the integration of SNNs with MARTE for timing and resource-aware modeling in embedded contexts remains underexplored. This paper introduces a preliminary framework to address this gap and foster further research in this promising domain.

3 Methodology

This section presents a MARTE-based framework for the real-time and resource-aware analysis of SNNs, comprising three core components:

1. Mapping SNN modeling elements to MARTE constructs;
2. Real-time scheduling and execution simulation of SNNs;
3. Model-driven verification to ensure system correctness and performance compliance.

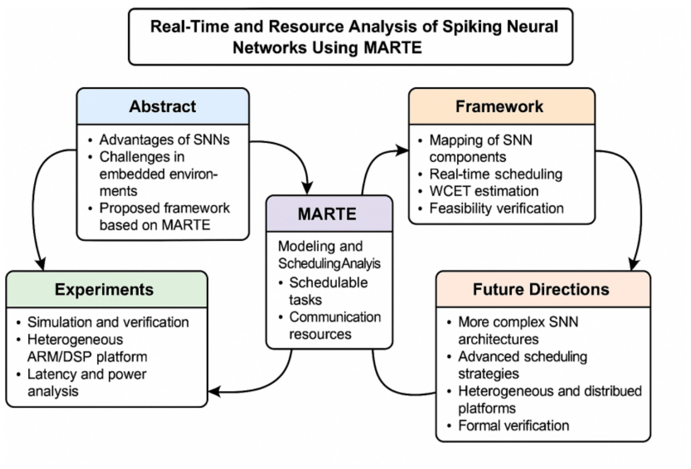

Real-Time and Resource Analysis of Spiking Neural Networks Using MARTE
Abstract
• Advantages of SNNs
• Challenges in embedded environments
• Proposed framework based on MARTE
Framework
• Mapping of SNN components
• Real-time scheduling
• WCET estimation
• Feasibility verification
MARTE
Modeling and Scheduling Analyis
• Schedulable tasks
• Communication resources
Experiments
• Simulation and verification
• Heterogeneous ARM/DSP platform
• Latency and power analysis
Future Directions
• More complex SNN architectures
• Advanced scheduling strategies
• Heterogeneous and distribued platforms
• Formal verification

Figure 1. Methodology overview.

3.1 Mapping of SNN Modeling Elements to MARTE

To represent the structural and behavioral aspects of SNNs within the MARTE framework (see Figure 1), common SNN components—such as neurons, synapses, and spike events—are mapped to corresponding elements in MARTE's meta-model. This enables the modeling of these components as time-sensitive tasks and interacting entities. Examples are outlined below (also illustrated in Figure 2):

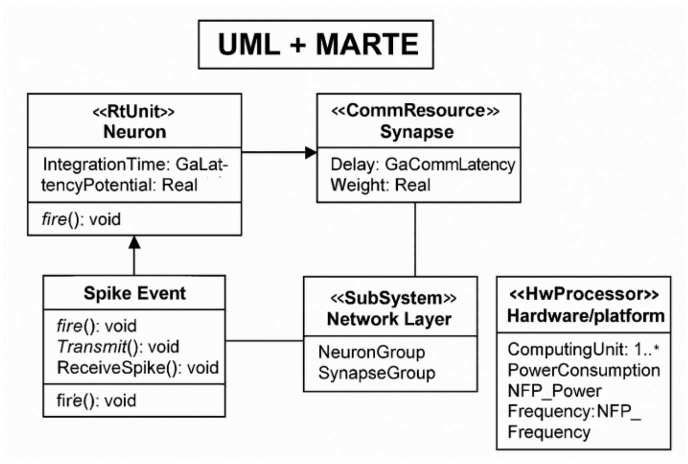


Figure 2. MARTE modeling framework.

1. **Neuron**: A neuron can be modeled as a real-time object using MARTE's ≪RtUnit≫ stereotype. Its attributes represent processing characteristics such as activation and integration times, as well as internal states like membrane potential.
   - IntegrationTime defines the temporal constraint for membrane potential accumulation.
   - Fire() represents the event of spike emission.
2. **Synapse**: A synapse is mapped to a ≪CommResource≫, representing a communication resource in MARTE. This includes attributes like transmission delay and synaptic weight updates.
   - Delay captures the spike transmission latency.
   - Weight represents the synaptic strength between neurons.
3. **Network Layer**: The abstract structure of an SNN layer can be modeled using the ≪SubSystem≫ stereotype in MARTE, enabling hierarchical organization and execution semantics.
   - NeuronGroup and SynapseGroup encapsulate groups of neurons and synapses, respectively.
   - ExecuteCycle() defines the periodic computation cycle of the SNN.
4. **Hardware/platform**: The deployment platform for SNNs is modeled using MARTE's Hardware

Resource Modeling (HRM), which includes processor cores, accelerators, caches, and on-chip interconnects. Extensions can be included for analyzing power consumption and thermal characteristics.

- ≪HwProcessor≫ represents a processor or computational unit.
- Attributes may include computing capability, frequency, and power usage.

5. **Spike Event**: A spike event can be represented as a Message or Trigger in MARTE, capturing timing constraints between sender and receiver tasks.
   - Fire(): Neuron A emits a spike.
   - Transmit(): The spike travels through a synapse, potentially incurring delay.
   - ReceiveSpike(): Neuron B receives the spike and updates its membrane potential.

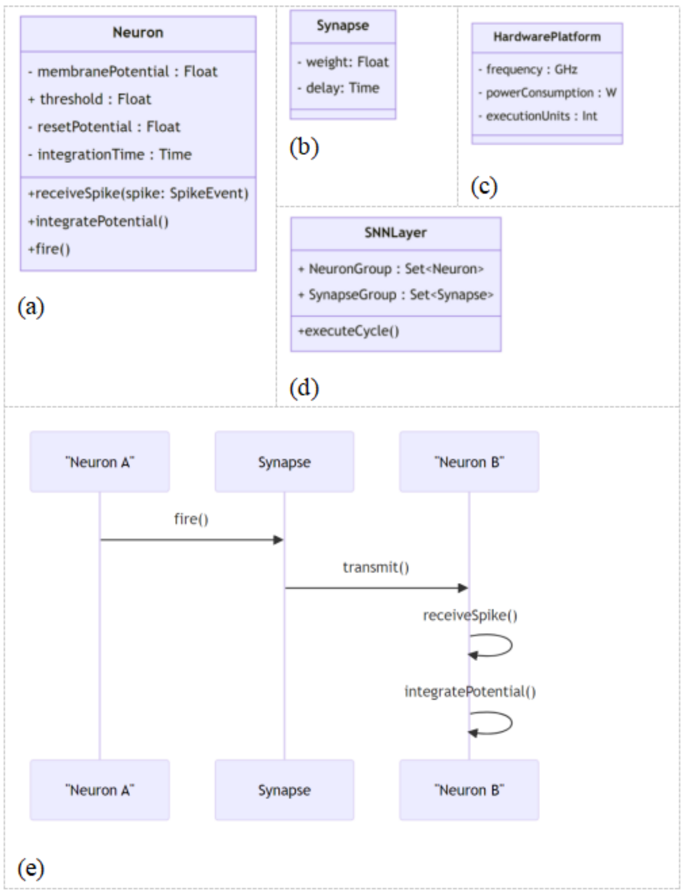


Figure 3. UML class diagram of (a)Neuron; (b)Synapse; (c)Network Layer; (d) Hardware/platform; and (e) Spike Event.

3.2 Real-Time Scheduling and Execution Analysis

During SNN execution, neurons perform operations such as membrane potential updates and threshold detection in response to incoming spikes. In real-time applications—such as robotics and sensor systems—meeting strict end-to-end latency requirements necessitates systematic analysis of execution time, task scheduling, and resource allocation. This is facilitated through MARTE-based modeling, using tools like GQAM (Generic Quantitative Analysis Model) and schedulability analysis, following a structured and model-driven workflow.

3.2.1 Task Division and Priority Allocation

- **Task Division**: Schedulable Resources are used to define different functional levels of the SNN:
  - SR_PerceptionLayer (perception layer task)
  - SR_HiddenLayer (middle layer task)
  - SR_OutputLayer (output layer task)

  Neuron clusters are modeled as schedulable tasks (e.g., Task_MembraneUpdate), while online learning operations are represented as schedulable services (e.g., Service_WeightUpdate).
- **Priority Assignment**: Using either partitioned or global scheduling, task dependencies are derived from a Dependency Graph to ensure correct execution order—for example, giving precedence to SR_PerceptionLayer. Priority policies are defined using MARTE's SchedParameters, which support both fixed priority (FP) and dynamic priority (DP) schemes.

3.2.2 Worst Case Execution Time (WCET) Estimation

The WCET of each SNN layer or neuron cluster is estimated using MARTE's performance modeling constructs. For example, the WCET of Task_MembraneUpdate on a given hardware resource can be expressed as:

```
<<ResourceUsage>> {
  executionTime = { min = 5us, max = 20us },
  accessCost = { memory = 3us, NoC = 5us }
}
```

This information is incorporated into the GQAM::GaAnalysisContext, which models key parameters such as executionTime, accessCost, and switchCost. Additional delays, such as NoC communication latency and cache miss penalties, are also included in the analysis.

3.2.3 Feasibility Verification of Scheduling

Schedulability analysis is performed using established algorithms such as Rate Monotonic (RM) or Earliest Deadline First (EDF):

1. Define scheduling parameters, including task cycle time t and deadline d.
2. Compute CPU utilization $u = \text{WCET}/t$ for each task.
3. Assess total utilization to determine the feasibility of the scheduling plan.

This analysis ensures that real-time constraints are met across all layers of the SNN.

3.2.4 Resource Allocation and Optimization

- **Resource Allocation**: Schedulable resources are mapped to hardware platforms (CPU, GPU, or NPU) using MARTE's ≪HwComputingResource≫. For example:

  ```
  <<HwComputingResource>> {
    type = "NPU",
    processingUnits = 128,
    frequency = 1GHz
  }
  ```

  Power management attributes are modeled to evaluate trade-offs between energy consumption and computational performance.

- **Dynamic Reconfiguration**: The model supports mode switching to accommodate different system loads. For instance, reconfiguration between high-load and low-load modes can be defined to optimize resource utilization dynamically.

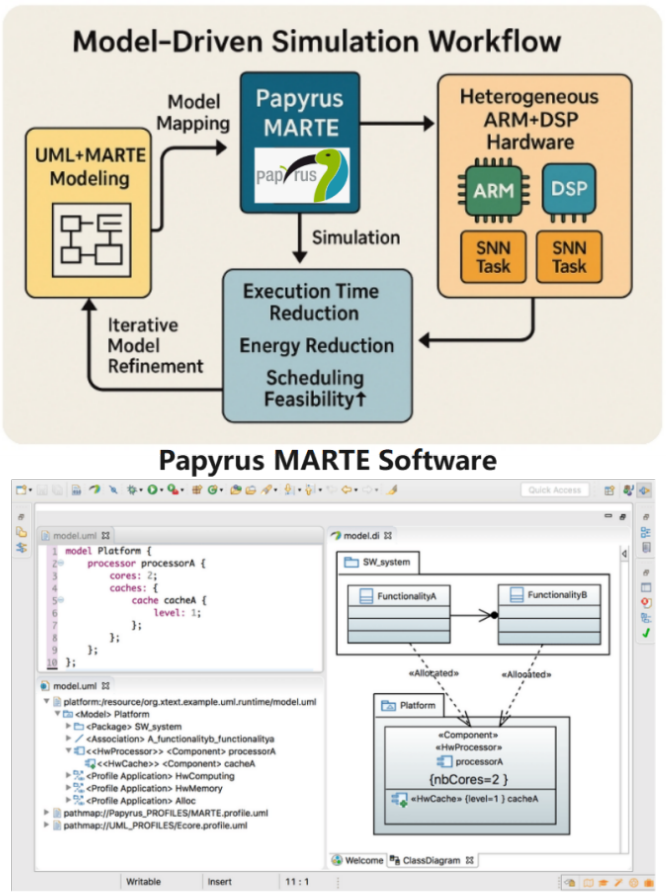
3.3 Model-Driven Verification and Iteration

Following the initial modeling phase, the behavior of the SNN in a real-time embedded environment is simulated using Model-Driven Engineering (MDE) tools such as Papyrus MARTE or MagicDraw. Key performance metrics—including execution time and power consumption—are validated against the constraints defined in the MARTE model.

If violations or inefficiencies are detected, iterative refinements can be made at the model level. This supports rapid re-analysis and optimization, enabling efficient tuning of SNN deployment strategies for embedded real-time systems.

4 Experiments

To validate the proposed framework, we conducted an experimental study using a lightweight SNN based on the Leaky Integrate-and-Fire (LIF) model, consisting of input, hidden, and output layers. The network was designed for classifying spike-coded sensor data and deployed on an embedded platform equipped with a multi-core ARM processor and a low-power DSP. The overall simulation workflow is illustrated in Figure 4.



```
model Platform {
    processor processorA {
        cores: 2;
        caches: {
            cache cacheA {
                level: 1;
            };
        };
    };
};
```

```
▼ platform:/resource/org.xtext.example.uml.runtime/model.uml
  ▼ <Model> Platform
    ▶ <Package> SW_system
    ▶ <Association> A_functionalityb_functionalitya
    ▼ <<HwProcessor>> <Component> processorA
        <<HwCache>> <Component> cacheA
    ▶ <Profile Application> HwComputing
    ▶ <Profile Application> HwMemory
    ▶ <Profile Application> Alloc
▶ pathmap://Papyrus_PROFILES/MARTE.profile.uml
▶ pathmap://UML_PROFILES/Ecore.profile.uml
```

Figure 4. A schematic diagram illustrating how Papyrus MARTE's UML-based modeling and simulation workflow.

4.1 Experimental Setup

- Hardware: ARM Cortex-A9 quad-core processor integrated with a low-power DSP module
- Software: Papyrus MARTE for modeling both the SNN and the hardware platform; Object Constraint Language (OCL) for specifying real-time and resource constraints
- Dataset: Spike-based input signals generated from multiple acceleration sensors, with temporal resolution aligned to the processing period of the SNN
- Evaluation Metrics: End-to-end latency (E2E),

Table 1. Comparison of end-to-end execution time under different hardware configurations

Hardware allocation scheme	Average E2E delay (ms)	Deadline compliance rate (5ms)	Resource utilization (ARM+DSP)
Full ARM core processing	6.2	62%	95% + 0%
ARM+DSP single-core auxiliary	4.8	88%	78% + 65%
ARM+Dual DSP Parallel (Experimental Scheme)	3.9	100%	65% + 85% × 2

Table 2. Performance Comparison of Scheduling Strategies

Scheduling strategy	Task set schedulability	Average waiting time	Worst response time	Applicable scenario
Fixed priority (RM)	100%	1.2ms	4.1ms	High load deterministic scenario
Dynamic priority (EDF)	100%	0.7ms	3.8ms	Variable load elastic scenario
Preemption without priority	72%	2.5ms	6.3ms	Non-real time reference baseline

Table 3. Comparison of Energy Consumption Optimization (Same Delay Constraint)

Calculation allocation scheme	Total power consumption	Power consumption reduction ratio	Key bottleneck module
Full ARM processing	420ms	–	ARM core (peak) 90°C
ARM++single DSP assisted	375ms	10.7%	Bus bandwidth
ARM++dual DSP parallel	357ms	15.0%	DSP unit load balancing

Table 4. Fault Prevention Effect of MARTE Model Constraint Analysis

Evaluation stage	Time-out times (reasoning every time)	Time-consuming troubleshooting (man-hours)	Constraint conflict early detection rate
Model-free analysis (post-integration)	23	12+	$<30\%$
MARTE model-driven verification	2	2	92%

schedulability, and energy consumption

The input, hidden, and output layers were modeled as three periodic tasks, with Spike Events governing their execution timing and inter-task sequencing. The processing capabilities of the ARM and DSP, as well as system parameters such as bus bandwidth and memory access latency, were modeled using MARTE's Software Resource Modeling (SRM) and Hardware Resource Modeling (HRM) profiles.

A strict end-to-end latency constraint of 5 ms was imposed to validate the system's real-time feasibility during scheduling analysis. Synaptic updates were modeled using shared memory, incorporating mutual exclusion and priority inheritance mechanisms to ensure data consistency and prevent race conditions within the MARTE framework.

4.2 Experimental Results

Model simulation using Papyrus MARTE yielded the following preliminary results:

Table 1 shows that parallelizing the hidden layer across dual DSPs significantly reduces end-to-end delay and meets hard real-time constraints.

Table 2 shows that EDF minimizes average waiting time, while RM offers greater stability under high load.

Table 3 shows that offloading to DSPs reduces ARM core temperature and overall power consumption, though bus contention and parallel overhead must be managed.

Table 4 shows that model-driven constraint verification cuts late integration risks and triples conflict detection efficiency.

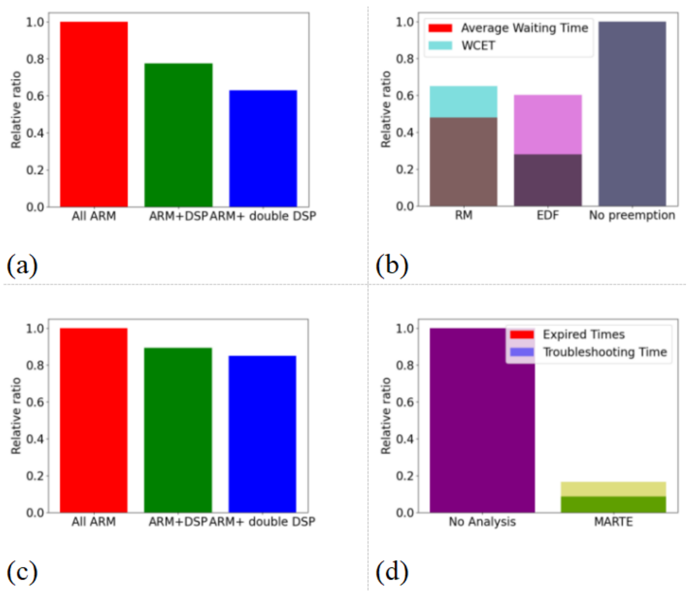


Figure 5. Numerical ratio comparison of (a)end-to-end execution time under different hardware configurations; (b) scheduling strategies; (c)energy consumption optimization (same delay constraint); and (d) fault prevention effect of marte model constraint analysis.

Figure 5 illustrate the comparative results from Tables 1–4. Without early model-level timing analysis, deadline misses and costly troubleshooting are likely. MARTE-based pre-analysis significantly reduces integration and testing risks.

5 Conclusion

Deploying Spiking Neural Networks (SNNs) in real-time embedded environments leverages their inherent advantages in low power consumption and temporal sensitivity. However, key challenges persist, including training complexity, limited support for hardware acceleration, and the difficulty of real-time scheduling. This paper presents a MARTE-based framework that integrates system modeling, scheduling analysis, and resource allocation tailored for SNNs. Preliminary experiments demonstrate the framework's feasibility in meeting stringent end-to-end latency and energy consumption requirements.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported by Shanghai Key Laboratory of Trustworthy Computing (East China Normal University) under Grant 24Z670103399, Key Laboratory of Embedded System and Service Computing (Tongji University), Ministry of Education under Grant ESSCKF2024-10, and Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence (Fudan University), Ministry of Education under Grant 25Z670102051, and Pre-research Fund of the School of Integrated Circuits (School of Information Science and Electronic Engineering), Shanghai Jiao Tong University under Grant JG0340001.

Conflicts of Interest

The authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Brette, R., & Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology, 94*(5), 3637-3642. [Crossref]

[2] Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on neural networks, 14*(6), 1569-1572. [Crossref]

[3] Selic, B., & Gérard, S. (2013). *Modeling and analysis of real-time and embedded systems with UML and MARTE: Developing cyber-physical systems*. Elsevier.

[4] Shailesh, T., Nayak, A., & Prasad, D. (2020). An UML based performance evaluation of real-time systems using timed petri net. *Computers, 9*(4), 94. [Crossref]

[5] Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., ... & Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro, 38*(1), 82-99. [Crossref]

[6] Di Alesio, S., & Sen, S. (2018). Using UML/MARTE to support performance tuning and stress testing in real-time systems. *Software & Systems Modeling, 17*(2), 479-508. [Crossref]

[7] Warnett, S. J., & Zdun, U. (2022, March). Architectural design decisions for machine learning deployment. In *2022 IEEE 19th International Conference on Software Architecture (ICSA)* (pp. 90-100). IEEE. [Crossref]

[8] Perez-Palacin, D., Merseguer, J., Requeno, J. I., Guerriero, M., Di Nitto, E., & Tamburri, D. A. (2019). A UML profile for the design, quality assessment and

deployment of data-intensive applications. *Software and Systems Modeling*, *18*(6), 3577-3614. [Crossref]

[9] Priyanka, E. B., Thangavel, S., Meenakshipriya, B., Prabu, D. V., & Sivakumar, N. S. (2021). Big data technologies with computational model computing using hadoop with scheduling challenges. In *Deep Learning and Big Data for Intelligent Transportation: Enabling Technologies and Future Trends* (pp. 3-19). Cham: Springer International Publishing. [Crossref]

[10] Díaz-Pace, J. A., Tommasel, A., & Capilla, R. (2024, September). Helping novice architects to make quality design decisions using an llm-based assistant. In *European Conference on Software Architecture* (pp. 324-332). Cham: Springer Nature Switzerland. [Crossref]

Liangshun Wu(Member, ICCK) received his B.S. degree from Central South University, Changsha, China, in 2014, and his M.S. and Ph.D. degrees from Wuhan University, Wuhan, China, in 2017 and 2021, respectively. He was a Visiting Scholar at the University of Electro-Communications, Tokyo, Japan, in 2024. He is currently a postdoctoral researcher at Shanghai Jiao Tong University, Shanghai, China. He serves as Editor-in-Chief of ICCK Transactions on Green Communications and Networking, and Associate Editor for IJSSN, JAIR, CST, and Blockchain. (Email: wuliangshun@sjtu.edu.cn)

Tao Tao(Member, ICCK) received his B.S. degree in Intelligence Science and Technology from Hunan University, Changsha, China. He then obtained his M.S. and Ph.D. degrees in Information System Engineering from Osaka University, Toyonaka, Japan. He is currently a postdoctoral researcher at Shanghai Jiao Tong University. (Email: tao.tao@lab.ime.cmc.osaka-u.ac.jp)