Energy Consumption Prediction Framework in Model-based Development for Edge Devices

Yue Hou and Takuya Azumi
Graduate School of Science and Engineering
Saitama University

***Abstract*—In the digital era, edge devices are widely used in many fields such as industrial automation, autonomous driving vehicles, and healthcare. Model-based development (MBD), as a fast and efficient development method, has been gradually applied to embedded software development for edge devices in recent years. However, energy consumption prediction is often neglected in the traditional MBD process. This study proposes an energy consumption prediction framework that aims to integrate energy consumption prediction into the MBD process. In addition, this paper proposes an energy consumption prediction framework for edge devices. The development of such a prediction tool implies that energy consumption can be evaluated during the design phase so that the design can be optimized to reduce energy consumption, which is particularly important for power-sensitive applications.**

***Index Terms*—Edge devices, embedded software, energy consumption prediction, model-based development**

I. Introduction

Edge devices are becoming increasingly important in modern life, from industrial control systems to medical devices, and autonomous systems [1]. As technology evolves, the systems that these edge devices carry become larger and more complex, and traditional development methods are gradually revealing their shortcomings of slow development processes and poor reusability. On the other hand, due to hardware limitations, embedded systems targeting edge devices must control energy consumption when designing software. This limitation highlights the importance of predicting the energy consumption of embedded software. Such prediction allows programs to be modified or completely redesigned to optimize energy consumption. To address the above issues, Model-based Development (MBD) [2] [3] has been proposed to efficiently develop embedded software for edge devices.

MBD is an advanced development methodology for software and systems engineering that focuses on using graphical models to guide design and implementation throughout the development cycle. MBD simplifies the process of designing and verifying complex systems by providing a higher level of abstraction than traditional hand-coding methods. At the same time, the developed generic model can be reused in different projects, which greatly improves the development efficiency. Embedded system development for edge devices typically involves applications with stringent requirements for real-time, reliability, and energy consumption. With MBD, engineers can utilize simulation and verification tools to assess the impact of design choices on system performance, including key metrics such as response time, energy consumption, and resource utilization, significantly reducing development time. In addition, MBD using MATLAB/Simulink supports automatic code generation using Embedded Coder, i.e., generating executable code directly from the model, thus reducing manual coding errors and development time.

MBD has a wide range of application prospects, but the simulation software used in MBD can only verify its function and performance. In practical embedded software design, due to the many limitations of the deployment environment and usage conditions of edge devices, the stability of their actual operation depends not only on the quality of the code, but also on the hardware platform and environmental factors and other aspects. Especially for battery-driven edge devices, their energy consumption directly determines their working life. Therefore, it is necessary to consider tools that can predict the energy consumption of software at the model design stage.

This paper proposes an energy consumption prediction framework based on Low Level Virtual Machine Intermediate Representation (LLVM-IR), which is expected to solve the problem of model's energy consumption prediction under cross-platform and cross-architecture with the help of the powerful expansion capability through LLVM-IR.

Contributions: The contributions of this work are stated as follows.

- A schema is proposed to describe the energy consumption of instructions, facilitating the prediction of model energy consumption.
- A method is introduced for extracting the working portion from automatically generated code and transforming it into LLVM-IR code for the purpose of energy consumption prediction. This method ensures that the accuracy error is controlled within 4%.
- A software tool is designed to predict model energy consumption without requiring real-world deployment, significantly reducing development costs and time.

This paper is organized as follows. The system model is described in Section II. The schema design and underlying data acquisition methods are introduced in Section III. The results of the evaluation by schema are discussed in Section IV. Furthermore, the studies related to this paper are discussed in Section V. The conclusions and future work scope of this research are summarized in Section VI.

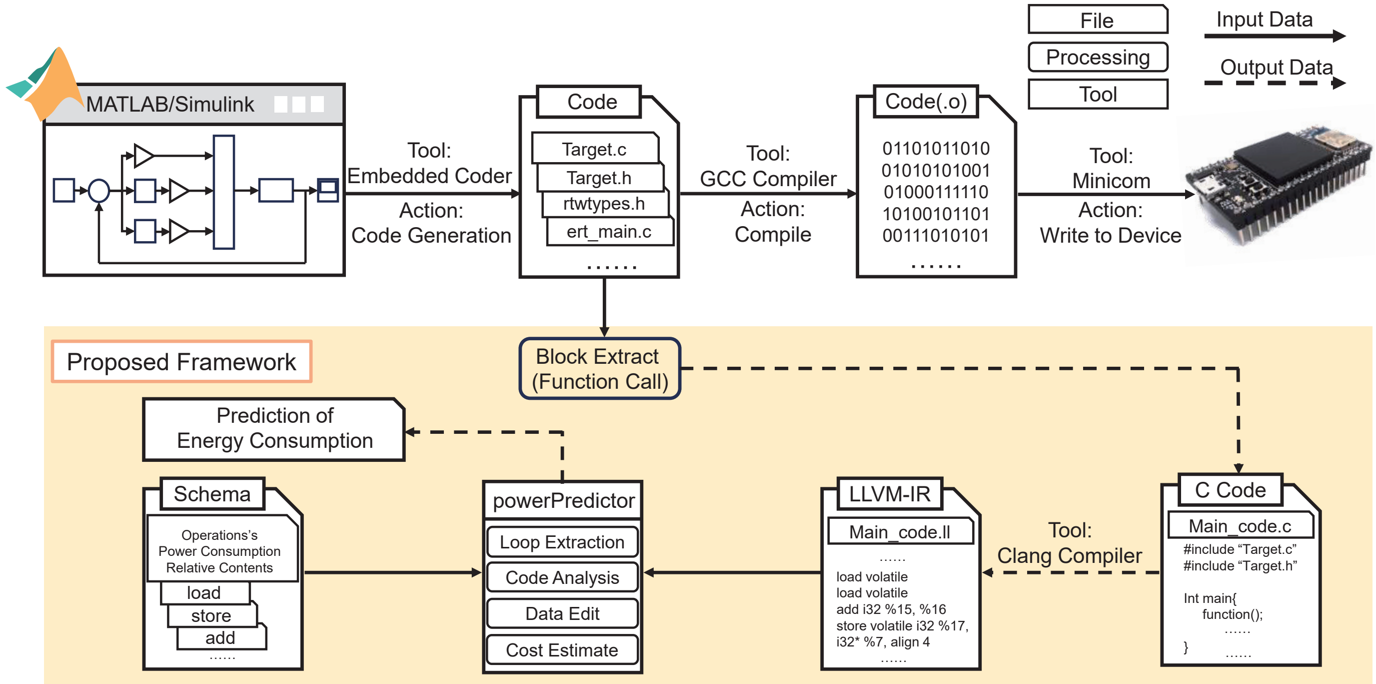

Fig. 1. Proposed Framework for Energy Consumption Prediction.

II. System Model

The proposed framework shown in Fig. 1 is described in this section. The proposed framework is divided into three main parts. The first is to propose a schema that describes the energy consumption of different instructions.

The second part is to propose a methodology to extract the actual operation part from the code generated by Embedded Coder by means of function calls and construct a new C file for constructing the LLVM-IR code.

The last part is to develop a prediction software that reads energy consumption pattern files and generated C code as input. The C code is converted to LLVM-IR code by calling the Clang compiler, and the LLVM-IR code is parsed to extract the loop section, which is used to calculate the number of occurrences of each instruction in the loop section. Finally, the actual number of times each instruction is performed is returned by combining the loop counts, and then the energy consumption is predicted in conjunction with the schema file.

This section introduces MATLAB/Simulink and LLVM-IR.

A. MATLAB/Simulink

Simulink [4] is a widely recognized simulation tool that is extensively used for modeling and simulating dynamic systems across various fields, including control systems, signal processing, and embedded systems development. One of the most attractive features of Simulink is its ability to simplify the process of translating a simulation model into actual code that can run on embedded hardware. Simulink can automatically generate C code through extensions, making Simulink an incredibly beneficial tool for those working on embedded systems.

The popularity of Simulink in the field of embedded systems has grown significantly. With Simulink's help, professionals can streamline their workflow and improve productivity by modeling, simulating, and generating code for their embedded systems more efficiently. This allows them to stay ahead of their projects and adapt to the ever-evolving technology landscape with ease.

B. LLVM-IR

LLVM-IR is an intermediate representation language used in the LLVM compiler framework. It acts as a bridge between source code and machine code, providing a hardware-independent, unified programming model. LLVM-IR also provides a unified, hardware-independent set of base instructions that are not dependent on any particular programming language or target hardware architecture. This means that regardless of whether the source code is written in C, C++, Rust, or any other LLVM-enabled language, after conversion to LLVM-IR, all instructions will use this common instruction set representation. This also provides certain cross-platform and cross-architecture advantages for the energy prediction framework proposed in this paper.

III. Proposed Method

The detailed process of predicting the energy consumption of embedded software is described in this section.

A. Methods of Constructing a Energy Consumption Description Schema

The schema used to describe energy consumption plays an important role in the overall framework. As one of the benchmarks for prediction, it is desirable to design a schema that is easy to understand and has the ability to be extended and compatible. One of the main goals of this schema is to provide a generalized framework for describing energy consumption information for any instruction. In order to achieve this goal, the schema of the proposal is designed as a three-tier structure.

1) Top layer: CommonInstructionSet: This layer serves as the root element of the entire description schema and is intended to contain energy consumption information for one or more instructions.

2) Middle Layer: Instruction: Below the top layer, each *Instruction* element represents the energy consumption description of a single instruction. This layer identifies the specific instruction name through the name attribute, allowing the energy consumption information of each instruction to be represented and considered individually.

3) Bottom layer: PowerConsumption, Cost, and Impact: This layer is the most specific information layer that directly describes the cost of the energy consumption associated with each instruction *Cost* and all the possible impacts of resource competition between instructions in a multicore system *Impact*. *Impact* is considered as an outlook for future energy predictions made in multi-core systems. There is no practical application in this paper.

B. Methods to Obtain the Execution Time and Energy Consumption of Instructions

Code execution time is closely related to energy consumption, and in order to predict energy consumption, predictions of code time can be synchronized and used as validation. Therefore, the measurement of execution time and energy consumption for individual instructions is extremely important, and the error in the measurement of individual instructions will be directly reflected in the overall prediction. For individual instruction execution time and energy consumption measurements, a method similar to the cyclic execution method is often used to obtain more accurate predictions.

In order to accurately obtain the execution time, the measurement of choosing a suitable timing method is also essential. Although the base library provides timing functions, the accuracy may need to be improved. The SONY Spresense board used in this experiment is equipped with the DWT (Data Watchpoint and Trace Unit), which is able to enable the timer directly by manipulating the specified registers, but also carries the same risk of clock overruns.

C. Methods to Extract the Operation Part from Code Generated by Embedded Coder

The most important aspect of the code extraction process is the need to identify the entry point of the code. For embedded systems, the entry point is usually the startup code or the main function. Therefore, it is sufficient to call the main function section in the generated file by means of a function call in the new c-file.

IV. Evaluation

This section focuses on evaluating the energy consumption prediction for embedded systems. The accuracy, as well as the practicality of the proposed scheme, is evaluated by comparing the predicted energy consumption with the actual energy consumption. The elements related to the evaluation are divided into the following sections.

First, the energy consumption and actual execution time of basic instructions (LLVM-IR) of the target machine are measured and computed using the method described in Section III. The target device is a SONY Spresense [5], which incorporates a CXD5602 microprocessor. This microprocessor features six ARM Cortex-M4F cores, each capable of operating at up to 156 MHz. It also includes 1.5 MB of SRAM and 8 MB of flash memory. Importantly, it should be noted that this device does not have a cache. In order to measure the power consumption of the device and considering the universality of the framework, only a simple USB tester was used in this study. The device used is the AVHzY CT-3 [6], a measurement device capable of achieving a measurement frequency of up to 1,000 measurements per second.

Further, a test script was created to acquire the actual execution time and energy consumption in a single-core environment. Based on the measured data, predictions were made at the LLVM-IR instruction level and compared with the measured values to analyze the prediction accuracy.

Secondly, a Simulink model is constructed and code is generated using Embedded Coder and then deployed to the target device to obtain its real runtime as well as energy consumption data through measurements.

Finally, the generated code is directly imported into the prediction software of the proposed framework and combined with the verified schema content to make predictions, and finally compared and analyzed with the real data.

A. Prediction Based on Single Command Execution Time and Energy Consumption

Since the prediction function of the proposed framework is calculated based on the number of command executions at the LLVM-IR level, the proposal's model, as one of the inputs, needs to provide the prediction software with the energy consumption per command. This data is obtained through actual measurements as described in Section III.

We measured the four basic instructions as well as the instructions to read data from memory and write data to memory. The "for" serves as a benchmark reflecting the time and energy consumption required for a single iteration of the empty loop body. Since the execution time and energy consumption measurements for the other commands are based on the execution of the loop body, the data was processed to accurately reflect the execution time and energy consumption of individual commands. Specific results are shown in Fig. 2.

In order to better analyze the most time consuming and energy consuming parts of the code, focus was initially placed on loop bodies. To confirm that the loop body significantly impacts the overall time and energy consumption, we assumed

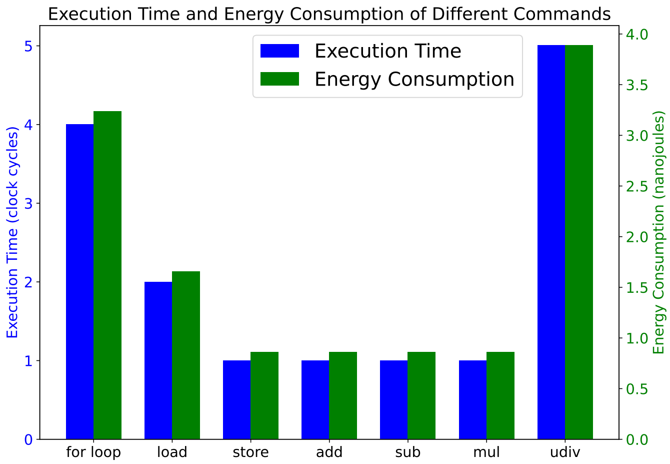

Fig. 2. Execution Time and Energy Consumption of Commands.

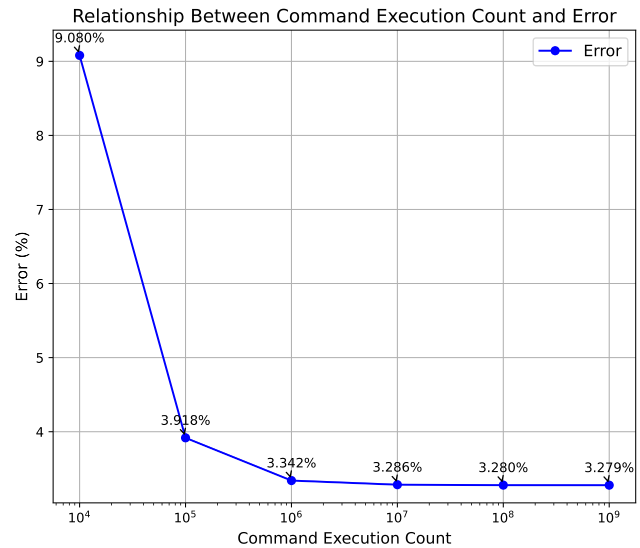

Fig. 3. Energy Consumption Prediction Error.

TABLE I
COMPARISON OF THIS THESIS WITH EXISTING STUDIES.

	Number of Execution	Total Time (clock cycles)	Total Energy (nanojoules)
load	8	16.018	13.248
store	4	4.005	3.456
add	1	1.001	0.864
sub	1	1.001	0.864
mul	1	1.001	0.864
udiv	1	5.006	3.888
for	1	4.007	3.24
Total		32.039	26.424

that when the number of loop bodies in the test code surpasses a certain magnitude, the running time and energy consumption of these loop bodies could somewhat represent the code's total running time and energy usage. The test code was designed to execute the four basic arithmetic instructions—addition, subtraction, multiplication, and division—within a loop.

As previously discussed, our primary focus is on the loop body. Therefore, in the LLVM-IR code, only the loop body portion needs to be extracted. This process resulted in identifying the frequency of each code's execution within the loop body, as depicted in Table I. The term *for*, while not an LLVM-IR command, serves here to offer a more intuitive and succinct depiction of the loop body.

The *Total Time* column in Table I calculates the time consumed by the execution of command, and the *Total Energy* column calculates the energy consumed on the command. The data used in this calculation are the actual measurement data with the target device. Finally, the total time and energy consumed by the commands of the loop body are summed up to get the total time and energy required to execute a loop body, which are 32.039 clock cycles and 26.424 nanojoules.

Experiments were conducted on an actual machine, with the number of loop body executions as the control variable. These experiments started with the loop body being executed 10,000 times, increasing incrementally until reaching a maximum of 1,000,000,000 executions.

Analyzing Fig. 3, reveals a gradual decrease in the discrepancy between predicted and actual values as the number of loops increases. This observation verifies our previous assumption that the running time and energy consumption of the loop body can be regarded as that of the whole code as long as the number of loops exceeds a certain order of magnitude. This phenomenon is particularly evident in embedded devices. Embedded devices are typically used in a variety of sensing applications, where environmental data is constantly collected and tasks such as processing and transmission are performed. Therefore, the design of the loop body is particularly important. Experimental results show that the running time and energy consumption of the loop body can reflect the overall state of the embedded system to a certain extent.

B. Prediction of the Energy Consumption of the Generated Code with Proposed Framework

In the previous subsection, we verified the feasibility of predicting execution time and energy consumption based on a single LLVM-IR command. In this section, we model and generate the corresponding code through Simulink and perform energy prediction through the proposed framework. The generated code is also deployed to the object device to actually test the energy consumption. Finally, the predicted and actual values are compared, and an analysis is performed.

1) Building Simulink models: As shown in Figs. 4 and 5, the main body of the model consists of a For Iterator Subsystem, where the loop control variables of the For are defined externally by i. The storage class of i is *ImportedExtern*, which is used to set the size of i externally.

Inside the For Iterator Subsystem is a simple arithmetic procedure with four inputs and three outputs. The storage class of each input variable is set to *ImportedExtern* to specify the input externally, while the storage class of the output variables is set to *ExportedGlobal* to read the output externally. The output variables are set to *ExportedGlobal* to read the

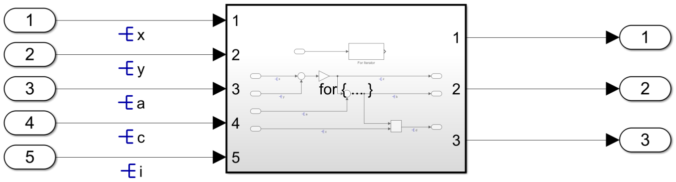

Fig. 4. Overall Model Construction.

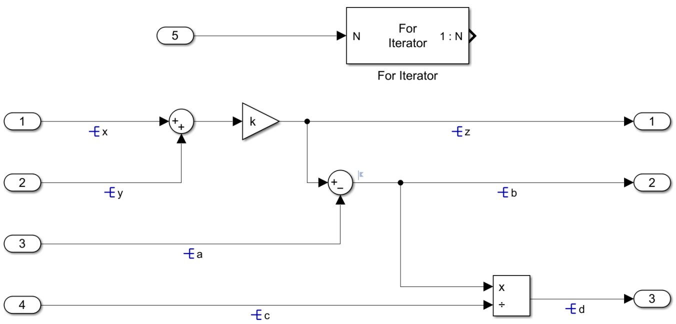

Fig. 5. For Iterator Subsystem Construction.

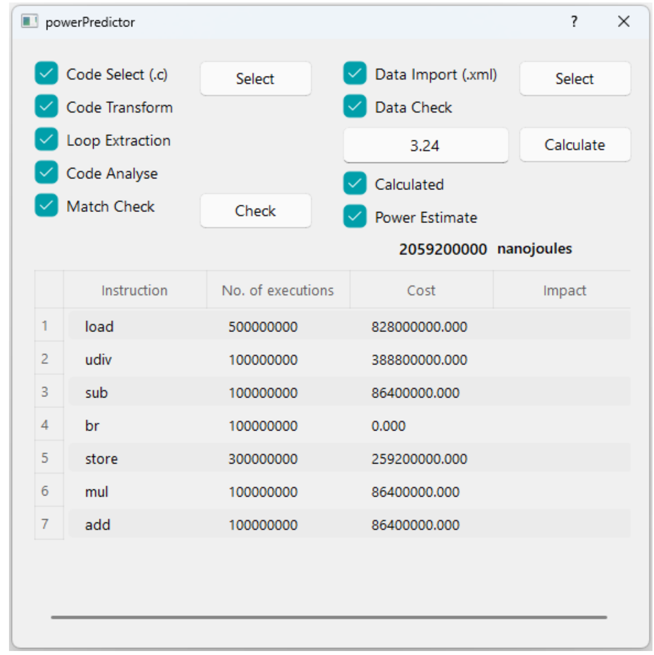

Fig. 6. Code Analysis and Energy Consumption Prediction Result.

outputs externally. k is set to a constant with the storage class *ConstVolatile*.

2) Energy Consumption Prediction through Proposed Framework: In order to facilitate data entry and modification, the framework provides a self-contained editor named *xmlEditor*, in which users can directly inspect and edit individual data to facilitate the construction of a schema for predicting energy consumption.

Once the schema is constructed, the next thing to do is to open the prediction software called *powerPredictor*. The first step is to click on the Select button and select a C file as input, in this case to predict the energy consumption of the model, we directly select the .c file generated by the corresponding model. Next, by clicking the Check button, the software analyzes the running part of the matching code, extracts the loop body part, and performs LLVM-IR instruction counting.

After completing the above two parts, the analysis of the generated code is finished, and the next step is to select the schema file generated by the user as the data input. Since there is no for in the schema, we need to manually input the energy consumption of a single for loop. After that, click the Calculate button, and the software will calculate the predicted energy consumption as shown in Fig. 6.

3) Comparison and Analysis of Prediction Result: The prediction obtained from the prediction software was 2,059,200,000 nJ, while the energy consumption of the generated code was measured to be 1,980,000,000 nJ by the actual deployment, with an error of 4%.

The primary cause of this error is the limited precision of the measurement equipment. The maximum measurement frequency is only 1,000 times per second. For embedded devices, which typically have low energy consumption, this low measurement frequency results in inaccurate measurement outcomes. Consequently, these inaccuracies propagate through the system, affecting the schema and ultimately leading to errors in the final prediction.

V. Related Work

Energy consumption prediction for embedded software involves research in many areas, including modeling, automatic code generation, and energy consumption measurement and prediction methods. The comparison of this study with related studies is displayed in Table II.

First, in terms of modeling, a unified development framework for embedded software development has been proposed [7]. The framework provides an intermediate representation of the model (MIR) and its parser, which enables a unified representation and coordination between different design tools. On the other hand, to address the problem of existing code generation tools' disregard for the instruction pipeline, a code generation tool *Mercury* for Simulink models has been developed to achieve efficient and optimized code generation for Simulink models by improving the efficiency of generated code and the utilization of the instruction pipeline [8]. The generated code is also deployed to devices with different architectures to verify the efficiency of the code generated by the proposed framework.

Accurate energy consumption measurements are essential for assessing the efficiency of software and hardware, optimizing the energy efficiency of algorithms and codes, etc. For this paper, correctly measuring the energy consumption of a device is indispensable for predicting energy consumption as well as evaluating prediction results. Guo [9] et al. investigate methods and techniques for measuring energy consumption of embedded systems and compare different models and approaches. In particular, energy consumption measurement

TABLE II
COMPARISON OF THE PROPOSED METHOD AND OTHER METHODS

	Modeling	Code Generation	Energy Consumption Measurement	Energy Consumption Prediction
MDD [7]	✓	✓		
Mercury [8]	✓	✓		
IEEE Access. 2021 [9]			✓	
SDK4ED [10]			✓	✓
J4CS [11]			✓	✓
PARTSim [12]			✓	✓
This paper	✓	✓	✓	✓

methods using external and internal devices were compared and the advantages and disadvantages of the different methods were analyzed.

In order to predict the energy consumption of edge devices, an analysis approach that combines static analysis with data-driven regression modeling, aims to tackle the challenges of high performance and low energy consumption in edge devices has been proposed [10]. This method stands out for its ability to offer rapid prototyping and flexibility, circumventing the time-intensive dynamic measurement analysis, redundancy, complexity, and scalability issues found in previous strategies, but requires a significant amount of time to collect data for regression modeling. In contrast, the J4CS tool aims to facilitate early-stage performance analysis, enabling swift energy consumption estimation and comparison for embedded software on target processors [11]. This tool supports specific hardware/software co-design methods, enhancing early development stages.

In addition, as an extension to RTSim, the PARTSim simulator introduces a new mechanism to accommodate power-aware embedded platforms [12]. PARTSim advances the simulation methodology using a two-phase process: first, data collection and analysis are performed to build a linear model, and then a system simulation is performed to evaluate task execution time and energy consumption under various configurations.

VI. CONCLUSION

This paper proposed an energy consumption prediction framework designed for the prediction of software energy consumption within MBD process. The framework encompasses schema construction, module extraction, and the application of energy consumption prediction software. Employing this framework enables the accurate prediction of energy consumption for code generated from models without the necessity for live machine execution or deployment. The prediction accuracy is validated with an error margin of 4%, affirming the framework's efficacy in leveraging energy consumption schemata for predictions in MBD of embedded software. Furthermore, the study elucidates that the overall energy consumption of an embedded system is intricately linked to the frequency of code loop body executions.

Notably, the advent of multi-core processors in edge devices offers enhanced computational capabilities while maintaining low energy consumption. The prediction of energy consumption in edge devices equipped with multi-core processors emerges as a prospective area for future investigation. Moreover, the utilization of peripheral devices within edge devices significantly contributes to energy consumption, representing a crucial aspect in the prediction of total system energy consumption.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 23H00464.

REFERENCES

[1] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," in *Proc. of ACM/IEEE International Conference on Cyber-Physical Systems*, 2018, pp. 287–296.

[2] Y. Kobayashi, K. Honda, S. Kojima, H. Fujimoto, M. Edahiro, and T. Azumi, "Mapping Method Usable with Clustered Many-core Platforms for Simulink Model," *Journal of Information Processing*, vol. 30, pp. 141–150, 2022.

[3] R. Yoshinaka and T. Azumi, "Model-based development considering self-driving systems for many-core processors," in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2020, pp. 337–344.

[4] "MATLAB/Simulink: Design, Simulate, Deploy - Simulink is a block diagram environment used to design systems with multidomain models, simulate before moving to hardware, and deploy without writing code," Available: http://www.mathworks.com/products/simulink.

[5] "SONY Spresense: High-performance microcontroller board with hi-res audio, camera input, internal GPS and Edge AI support." Available: https://developer.sony.com/spresense.

[6] "AVHzY CT-3 USB 3.1 Power Meter - DC 26V 6A," Available: https://www.avhzy.com/html/product-detail/ct3.

[7] Z. Su, D. Wang, Y. Yang, Z. Yu, W. Chang, W. Li, A. Cui, Y. Jiang, and J. Sun, "MDD: A Unified Model-Driven Design Framework for Embedded Control Software," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 10, pp. 3252–3265, 2022.

[8] Z. Yu, Z. Su, Y. Yang, J. Liang, Y. Jiang, A. Cui, W. Chang, and R. Wang, "Mercury: Instruction Pipeline Aware Code Generation for Simulink Models," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4504–4515, 2022.

[9] C. Guo, S. Ci, Y. Zhou, and Y. Yang, "A Survey of Energy Consumption Measurement in Embedded Systems," *IEEE Access*, vol. 9, pp. 60 516–60 530, 2021.

[10] C. Marantos, K. Salapas, L. Papadopoulos, and D. J. Soudris, "A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis," *SN Computer Science*, vol. 2, 2021.

[11] V. Muttillo, P. Giammatteo, V. Stoico, and L. Pomante, "An early-stage statement-level metric for energy characterization of embedded processors," *Microprocessors and Microsystems*, vol. 77, p. 103200, 2020.

[12] G. Ara, T. Cucinotta, and A. Mascitti, "Simulating execution time and power consumption of real-time tasks on embedded platforms," in *Proc. of ACM/SIGAPP Symposium on Applied Computing*, 2022.