May. 13th 2024

RAGE

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

Yue Hou Azumi Takuya

Saitama Univ. Japan Saitama Univ. Japan

Saitama University

Outline

- Background
- Proposed Prediction Framework
- Evaluation
- Conclusion

Edge Device

■ Development of Edge Device

- The count of devices is increasing
- Systems is becoming complex

■ Development of Systems on Edge Devices

- The cost is becoming significant.
- The reusability of the code developed with traditional method is limited.

MATLAB/Simulink

■ What is MATLAB/Simulink

- MATLAB
  - A programming and numeric computing platform developed by MathWorks.
- Simulink
  - A block diagram environment for Model-based Development integrated with MATLAB.

■ Why Use MATLAB/Simulink

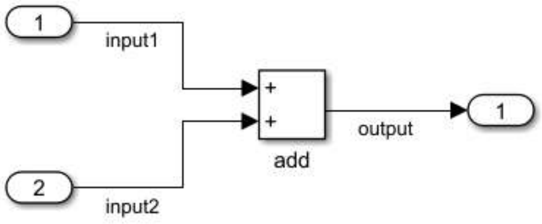
- Efficiency
  - Simplifies complex numerical calculations and visualization.
- Integration
  - Provides an easy-to-use environment.
- Versatility
  - Used in various industries, such as engineering and finance.

Model-based Development (MBD)

■ What is MBD

- uses graphical models to guide design and implementation

$$output = input1 + input2 \rightarrow$$



- allows engineers to verify system behavior and performance by simulation
- MBD with MATLAB/Simulink and Embedded Coder supports automatic code generation

```
void untitled_step(void)
{
  /* Outport: '<Root>/Outport' incorporates:
   *  Inport: '<Root>/Inport'
   *  Inport: '<Root>/Input'
   *  Sum: '<Root>/add'
   */
  rtY.Outport = rtU.input1 + rtU.input2;
}
```

Low Level Virtual Machine Intermediate Representation (LLVM-IR)

■ What is LLVM-IR

- A low-level programming language used as the primary IR within the LLVM compiler framework.
- Serves as a bridge between high-level languages and the machine code, enabling code analysis and transformation.

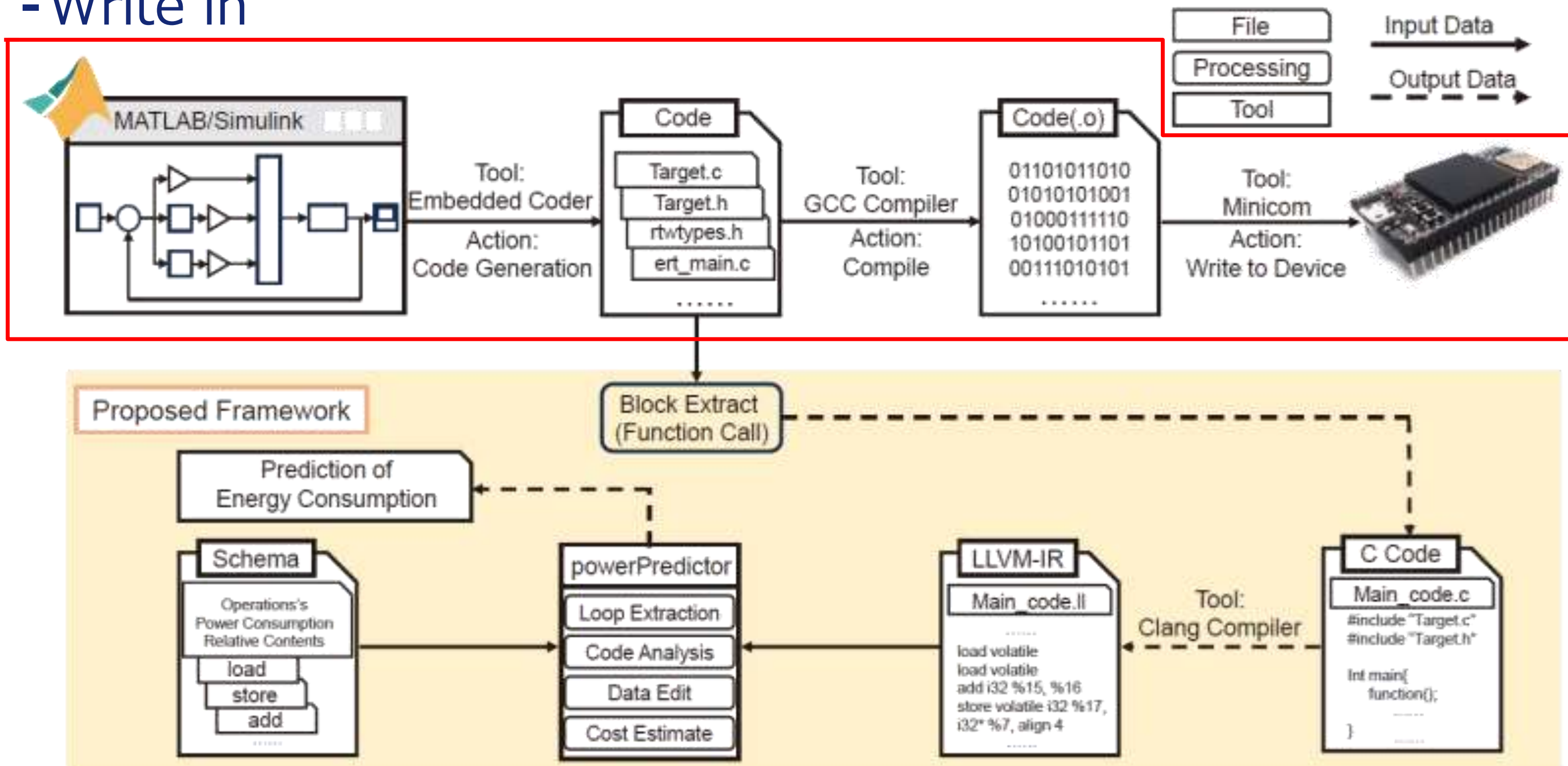
■ Why Use LLVM-IR

- Portability
  - Easy to support multiple target platforms.
- Optimization
  - Facilitates various optimizations both at the compile time and at run time.
- Flexibility
  - Allows developers to create compilers for new programming languages more easily, or extend the capabilities of existing languages.

System Model

■ MBD Sequence

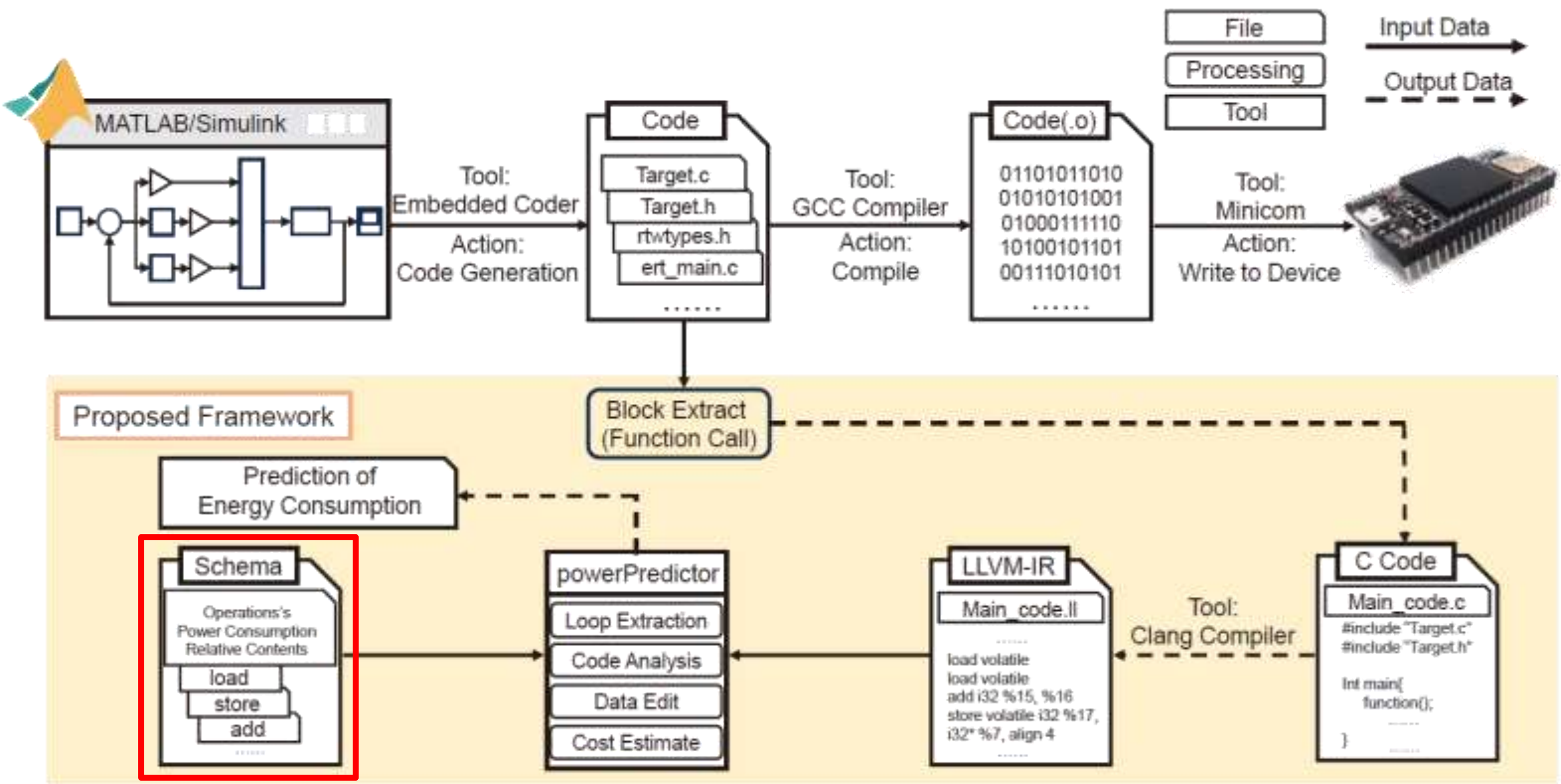
- Modeling
- Simulation
- Code Generation
- Compile
- Write in



Proposed Method

■ Energy Consumption Description Schema

- Objective
  - Provide a universal framework to describe energy consumption for any given instruction.



Proposed Method

■ Energy Consumption Description Schema

- Three-Tier Structure
  - Top Layer
    - CommonInstructionSet
  - Middle Layer
    - Instruction
  - Bottom Layer
    - PowerConsumption

```
<CommonInstructionSet>
        <Instruction name="example">
                <PowerConsumption>
                        <Cost>1.000</Cost>
                        <Impact>0.000</Impact>
                </PowerConsumption>
        </Instruction>
</CommonInstructionSet>
```

Proposed Method

■Time and Energy Consumption of Instructions

- Correlation of Time and Energy
  - Premise: Execution time of code is directly linked to energy consumption.
  - Validation: Accurate time predictions validate energy consumption estimates.

- Importance of Measurement
  - Accuracy: Precise measurement of individual instructions is critical—errors directly affect overall prediction reliability.
  - Technique: A cyclic execution approach is commonly used for better precision in predicting individual instructions' time and energy use.

Proposed Method

■Time and Energy Consumption of Instructions

- Execution Time Measurement
  - Challenge: Selecting an accurate timing method is crucial for exact execution time capture.

  - Tools: While standard libraries offer timing functions, their precision may be insufficient.

  - Advanced Method: The experiment utilizes the DWT (Data Watchpoint and Trace Unit) on the SONY Spresense board, enhancing accuracy by directly accessing specific registers.

  - Consideration: Using DWT presents risks such as clock overruns, which must be managed.

Proposed Method

■Extract the Operation Part from Generated Code

```c
#define ITERATION 1000000

real_T x = 0.0;
real_T y = 0.0;
real_T a = 0.0;
real_T c = 0.0;
real_T z = 0.0;
real_T b = 0.0;
real_T d = 0.0;
uint32_T i = 0;

int main() {
    untitled_initialize();

    x = 1.0;
    y = 2.0;
    a = 3.0;
    c = 4.0;
    i = ITERATION;

    untitled_step();

     printf("Output z: %f\n", z);
     printf("Output b: %f\n", b);
     printf("Output d: %f\n", d);

    return 0;
}
```

```c
/* Model step function */
void untitled_step(void)
{
  z = (x + y) * k;

  b = z - a;

  d = b / c;
}

/* Model initialize function */
void untitled_initialize(void)
{
  b = 20.0;
}
```

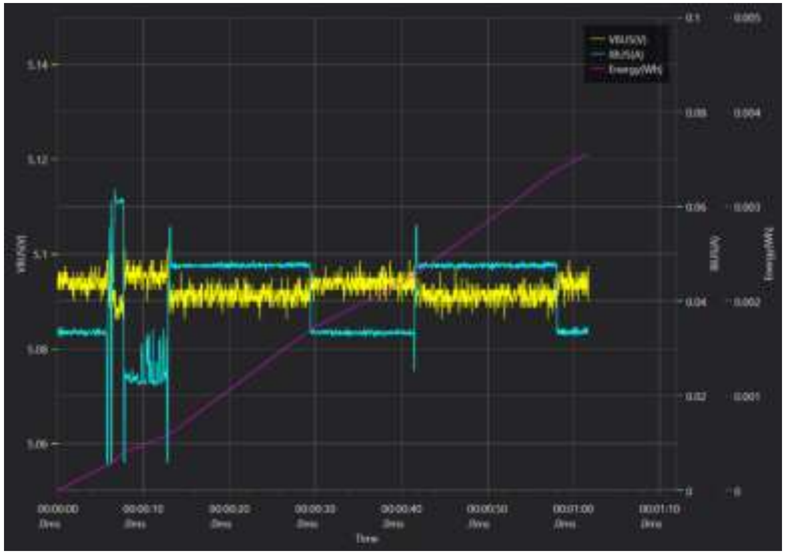
Experimental Environment

■Target device

-SONY Spresense (ARM Cortex M4F)

■Measurement device

-AVHzY CT-3 USB tester

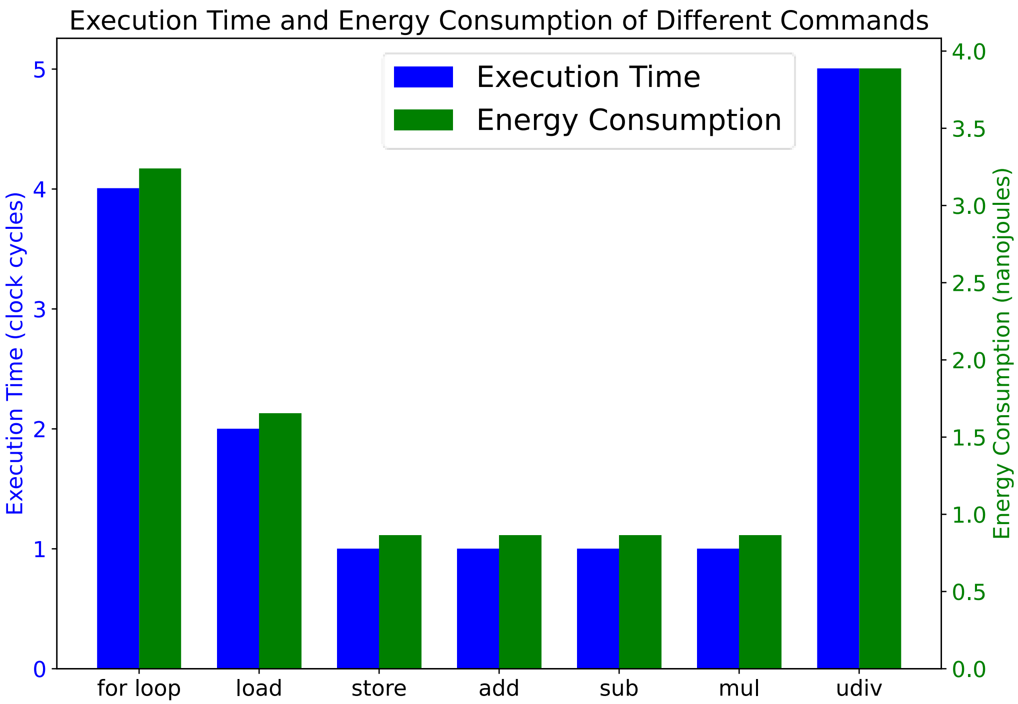


Experimental

■Basic Evaluation

- Measure the power consumption and execution time of basic instructions on the target device

- Create test scripts to obtain actual execution time and power consumption in a single core environment

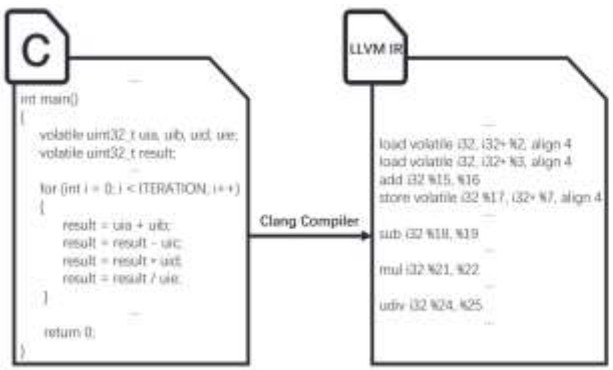


Experimental

■Basic Evaluation

- Test Scripts
  - Four arithmetic operations（add,sub,mul,div)

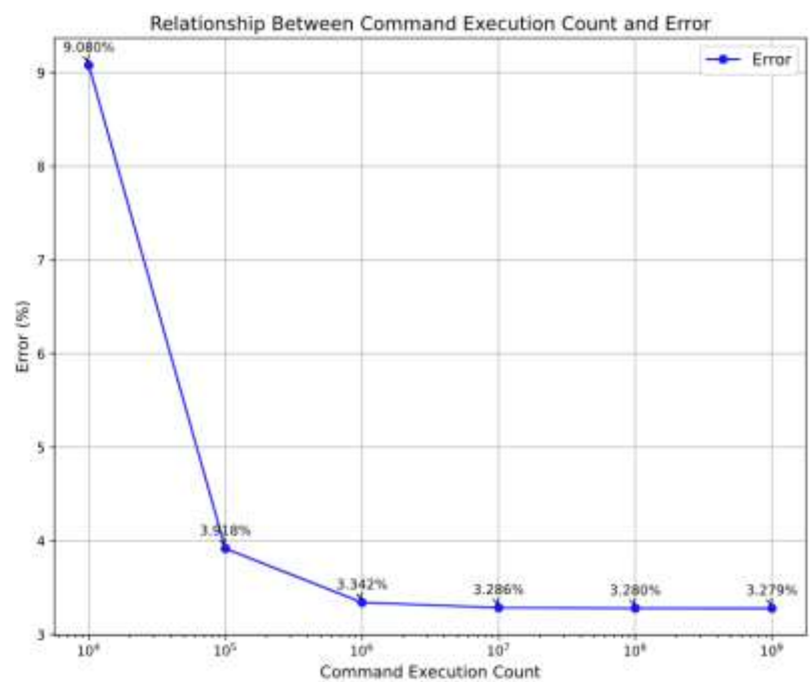
- Predicting power consumption
  - Focus on the “for” statement part



Test Scripts

	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

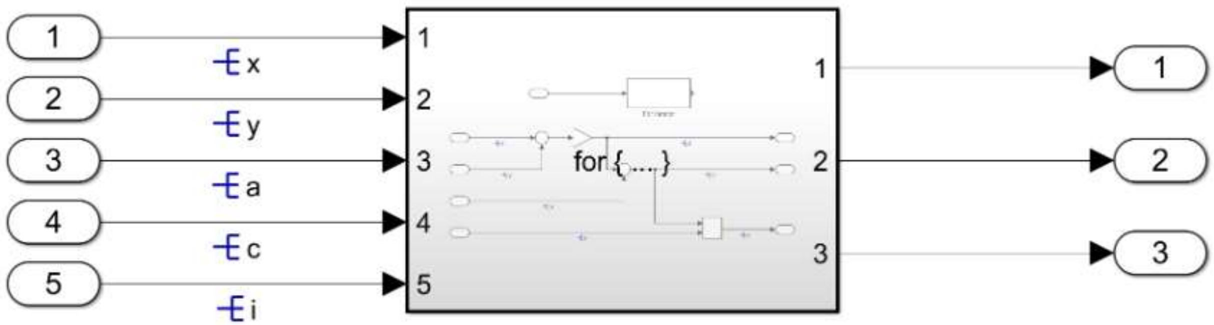
Time and Energy Consumption of Instructions



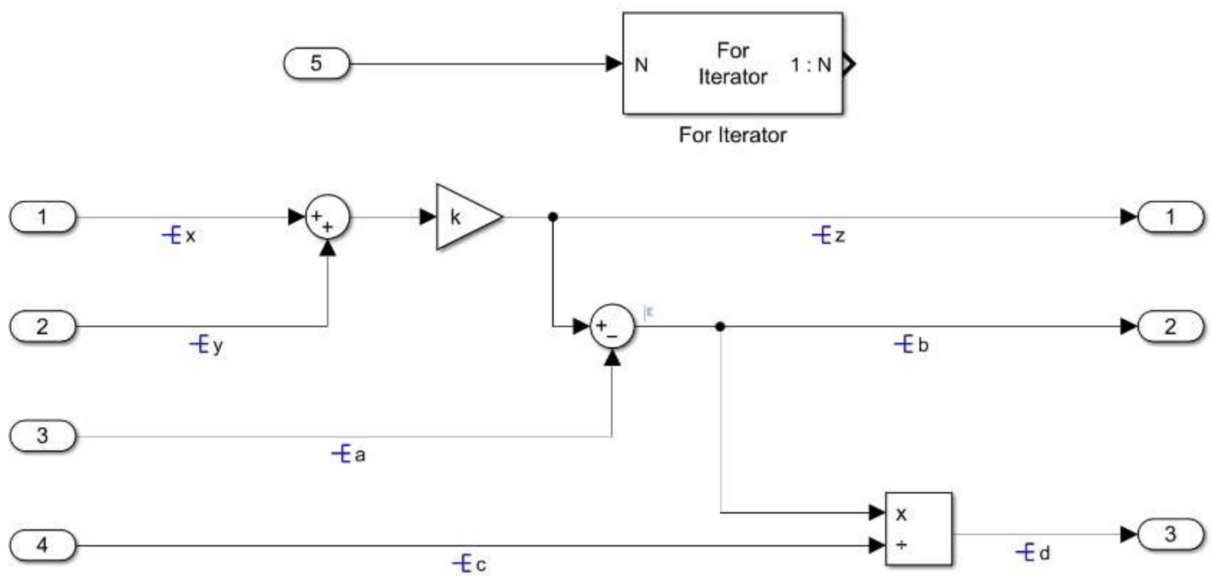
Power Consumption Prediction Focus on “for”

Experimental

■Evaluation with Models



Overall Model Construction



For Iterator Subsystem Construction

Experimental Environment

■Evaluation with Models

- Code generated by Embedded Coder
- Execute in user code with function calls
- Convert to LLVM-IR instructions
- Make predictions with a prediction tool

```
for (s1_iter = 1; s1_iter <= tmp; s1_iter++) {
  /* Gain: '<S1>/Gain' incorporates:
   *  Inport: '<Root>/Inport'
   *  Inport: '<Root>/Input'
   *  Sum: '<S1>/Sum'
   */
  z = (x + y) * k;

  /* Sum: '<S1>/Sum1' incorporates:
   *  Inport: '<Root>/Inport1'
   */
  b = z - a;

  /* Product: '<S1>/Divide' incorporates:
   *  Inport: '<Root>/Inport2'
   */
  d = b / c;
}
```

Experimental Environment

■Evaluation with Models

- Result
  - Prediction
    - 2,059,200,000 nJ
  - Measurement
    - 1,980,000,000 nJ
  - Error: 4%
- Error Analysis
  - Precision of the equipment
    - 1,000 times per second