ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНА ТЕХНІКА

УДК 004.054

Р. С. Чопей[1]
Д. В. Федасюк[1]

МЕТОД АНАЛІЗУ ТРИВАЛОСТІ ВИКОНАННЯ ПРОГРАМНОГО КОДУ З УРАХУВАННЯМ АРХІТЕКТУРИ МІКРОКОНТРОЛЕРА

[1]Національний університет «Львівська політехніка»

Розглянуто загальні проблеми, які можуть виникнути під час аналізу тривалості виконання програмного коду вбудованої системи. Огляд існуючих підходів, проведений авторами, показав, що методи, засновані на аналізі кеш-пам'яті та архітектури конвеєра команд мікроконтролера, неефективні для оцінки тривалості виконання програмного коду, що виконується на мікроконтролерних вбудованих системах. Один зі способів підвищення ефективності аналізу тривалості виконання програмного коду — це врахування внутрішньої архітектури мікроконтролера, включаючи затримки, що відбуваються під час обміну даними між внутрішніми модулями мікроконтролера та затримками при передачі даних через стандартні інтерфейси зв'язку.

Запропоновано метод аналізу тривалості виконання програмного коду з урахуванням швидкості обміну даними між внутрішніми модулями мікроконтролера та швидкості надсилання даних через інтерфейси зв'язку. Чисельними експериментами доведено підвищення точності отриманих результатів у порівнянні з існуючими методами.

Підвищення ефективності запропонованого методу, можливе шляхом врахування тривалості читання даних з оперативної пам'яті, а також за рахунок автоматизації процесу розрахунку тривалості виконання програми.

Ключові слова: вбудовані системи, архітектура мікроконтролера, середовище розробки Keil uVision, тривалість виконання програмного коду.

Вступ

Частина завдань, вирішуваних вбудованими системами реального часу, повинні виконуватись до настання їх крайнього терміну. Для гарантування завершення їх виконання застосовується аналіз тривалості виконання програмного коду, що передбачає три етапи:

- аналіз потоку керування, який визначає можливі шляхи виконання програми;
- аналіз функцій низького рівня, який враховує поведінку апаратних компонент (конвеєр процесору, аналіз кеш-пам'яті тощо);
- розрахунок найгіршої тривалості виконання завдань.

Найважливішим етапом аналізу тривалості виконання програмного коду вбудованих систем є аналіз функцій низького рівня. Для його виконання необхідно знати часові характеристики апаратного забезпечення, на якому буде виконуватись програма. Однак, дані, отримані про часові характеристики мікроконтролерів, є наближеними. Причиною цього є висока складність внутрішньої архітектури мікроконтролера, збереження архітектури ядра мікроконтролера у секреті через жорстку конкуренцію між виробниками, а також подані у документації прості формули, що дають розробнику наближене уявлення про часові характеристики мікроконтролерів.

Спрощення часових моделей мікроконтролерів, зумовлене відсутністю коректної інформації, призводить до заниження фактичної тривалості виконання програмного коду та некоректного використання ресурсів системи. Одним зі шляхів підвищення точності розрахунку тривалості виконання програмного коду є розроблення методу, що дає змогу врахувати архітектуру мікроконтролера, зокрема тривалість обміну даними між внутрішніми модулями та тривалість надсилання

даних через зовнішні інтерфейси зв'язку.

Метою роботи є розроблення методу аналізу тривалості виконання програмного коду з урахуванням втрати швидкості виконання програмного коду під час обміну даними через інтерфейси зв'язку.

Сучасні підходи до аналізу тривалості виконання функцій низького рівня

У роботах [1], [2] вважається, що тривалість виконання усіх функцій програми залежить лише від кількості обчислювальних операцій на секунду, які виконує мікроконтролер, тому ці методи аналізу суттєво занижують значення тривалості виконання програмного коду.

У інших роботах вважається, що тривалість виконання програмного коду залежить не лише від кількості обчислювальних операцій на секунду, що виконує мікроконтролер, а і від внутрішньої архітектури. У підходах, представлених у [3], [4], враховується вплив архітектури конвеєра команд на тривалість виконання програмного коду у мікропроцесорних вбудованих системах. У [5], [6] запропоновано метод розрахунку тривалості з урахуванням архітектури конвеєра команд та кеш-пам'яті. Однак, навіть у випадку використання цих методів розрахунку, отримані показники тривалості виконання будуть заниженими, оскільки не враховується швидкість обміну даними між внутрішніми модулями та швидкість обміну даними через інтерфейси зв'язку. Окрім того, використання цих методів є недоцільним для аналізу тривалості виконання програмного коду, що виконується на мікроконтролерних вбудованих системах, адже більшість мікроконтролерів мають просту архітектуру конвеєра команд, та відсутню кеш-пам'ять.

У [7] описано метод розрахунку тривалості виконання програмного коду, розроблений для мікроконтролерних вбудованих систем. Цей метод враховує втрати швидкості обміну даними через інтерфейси зв'язку. Тривалість надсилання даних через інтерфейс зв'язку розраховується за формулою

$$\text{bus-clock states} := \frac{f_{CPU}}{\gcd\left(f_{CPU}, f_{BUS}\right)}, \tag{1}$$

де f_{CPU} — частота, з якою працює мікроконтролер; f_{BUS} — частота, з якою працює інтерфейс зв'язку; gcd — найбільший спільний дільник.

Недоліком цього методу є відсутність можливості врахування швидкості обміну даними між внутрішніми модулями мікроконтролера.

Розробка методу аналізу тривалості виконання програмного коду з урахуванням швидкості обміну даними

Для розроблення методу використано мікроконтролер STM32F407VG, побудований на ядрі ARM Cortex-M4 та два різні джерела інформації: посібник користувача для мікроконтролерів серії STM32F4 [8], а також документація на ядро ARM Cortex-M4 [9]. Отримані дані про архітектуру мікроконтролера слугують вхідними даними для розроблення методу аналізу тривалості виконання програмного коду. В результаті аналізу інформаційних джерел [8], [9] встановлено, що усі внутрішні модулі під'єднані до однієї з трьох шин синхронізації: AHB, APB1, APB2 (рис. 1). Ці шини синхронізації забезпечують синхронний зв'язок між усіма внутрішніми модулями. Максимальна частота синхросигналу на шинах: AHB — 168 МГц, APB1 — 42 МГц, APB2 — 84 МГц.

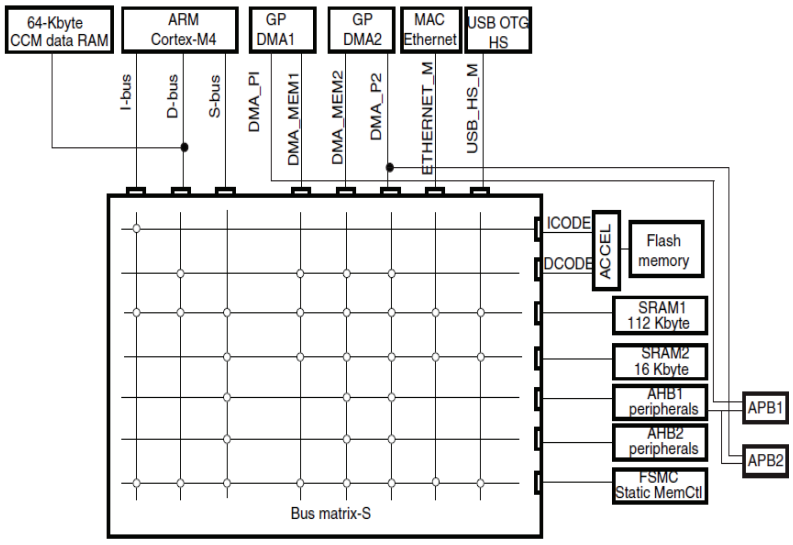

Рис. 1. Архітектура мікроконтролера STM32F407VG [8]

В процесі розроблення методу аналізу тривалості виконання програмного коду з урахуванням тривалості обміну даними між внут-

рішніми модулями та зовнішніми периферійними пристроями, передбачається послідовне вирішення таких задач: визначення налаштувань мікроконтролера на основі програмного коду, визначення взаємозв'язків між шинами синхронізації та внутрішніми модулями, визначення налаштувань інтерфейсів зв'язку з зовнішніми периферійними пристроями, розрахунок тривалості виконання програмного коду.

Особливості визначення налаштувань мікроконтролера на основі програмного коду

Кожний мікроконтролер після вмикання живлення налаштований за замовчуванням, відтак процес ввімкнення необхідних периферійних модулів, налаштування їхньої швидкості роботи, а також налаштування частоти роботи ядра мікроконтролера відбувається із запуском кожної програми. Для визначення частоти, на якій буде працювати мікроконтролер, аналізуємо текст функції, що виконує налаштування мікроконтролера. Розрахунок частоти, з якою працює мікроконтролер, проводиться за формулою:

$$f_{CPU} = f_O / PLL_M \cdot PLL_N / PLL_P, \tag{2}$$

де f_O — частота кварцового генератора; PLL_M, PLL_P — коефіцієнти ділення частоти кварцового генератора; PLL_N — коефіцієнт множення частоти кварцового генератора.

Розрахунок частоти синхросигналу на шинах APB проводиться за формулою

$$f_{APB} = \frac{f_{CPU}}{APB_P}, \tag{3}$$

де APB_P — коефіцієнт ділення частоти.

Особливості визначення взаємозв'язків між шинами синхронізації та внутрішніми модулями

Процес визначення взаємодії між шинами синхронізації та внутрішніми периферійними модулями передбачає проведення аналізу даних документації для обраного мікроконтролера, зокрема розділу «Reset and clock control», що містить інформацію про можливі варіанти налаштування шин синхронізації та взаємозв'язок між шинами синхронізації та внутрішніми модулями. Аналіз цього розділу здійснюється розробниками програмного забезпечення у ході його створення. Відтак, для визначення взаємозв'язків між шинами синхронізації та внутрішніми модулями на етапі тестування, достатньо провести синтаксичний аналіз тексту програми, що відповідає за ввімкнення тактування внутрішнього модуля. Текст програми виглядає так:

RCC_APB1PeriphClockCmd(RCC_APB1Periph_PPPx, ENABLE),

де RCC_APB1PeriphClockCmd — функція ввімкнення тактування внутрішнього модуля, який під'єднаний до шини синхронізації APB1; PPPx — назва периферійного модуля.

Особливості визначення налаштувань інтерфейсів зв'язку із зовнішніми периферійними пристроями

Для визначення актуальних налаштувань кожного з інтерфейсів зв'язку, що застосовуються у програмі мікроконтролера, аналізуємо текст програми, який використовується для налаштування. Розрахунок тривалості надсилання даних проводиться згідно з одержаною інформацією про налаштування інтерфейсу, а також у відповідності до типу інтерфейсу. Тривалість надсилання одного байту даних через інтерфейс SPI оцінюється за формулою

$$t_{SPI} = t_{APB} + t_{bSPI} = \left(\frac{1}{f_{APB}} \cdot 8\right) + \left(\frac{1}{f_{SPI}} \cdot 8\right), \tag{5}$$

де t_{APB} — тривалість надсилання одного байту до внутрішнього модуля SPI; t_{bSPI} — тривалість надсилання одного байту через інтерфейс SPI; f_{SPI} — робоча частота інтерфейсу SPI.

Тривалість надсилання одного байту даних через інтерфейс I^2C проводиться за формулою

$$t_{\text{I2C}} = t_{APB} + t_{\text{bI2C}} = \left(\frac{1}{f_{APB}} \cdot 8\right) + \left(\frac{1}{f_{\text{I2C}}} \cdot 10\right), \quad (6)$$

де t_{bI2C} — тривалість надсилання одного байту через інтерфейс I^2C, що включає надсилання двох додаткових біт синхронізації; f_{I2C} — робоча частота інтерфейсу I^2C.

Тривалість надсилання одного байту даних через інтерфейс UART проводиться за формулою

$$t_{\text{UART}} = t_{APB} + t_{\text{bUART}} = \left(\frac{1}{f_{APB}} \cdot 8\right) + \left(\frac{1}{f_{\text{UART}}} \cdot N\right), \quad (7)$$

де t_{bUART} — тривалість надсилання одного байту через інтерфейс UART, що включає надсилання N додаткових бітів; N — кількість додаткових бітів, що визначається у конфігурації інтерфейсу, знаходиться в межах від 2 до 4; f_{UART} — робоча частота інтерфейсу UART.

Розрахунок тривалості виконання програмного коду

```
  int main(void)
b508            PUSH     {r3,lr}
  {
    HAL_Init();
f7fffffe        BL       HAL_Init

    /* Configure the system clock to 168 MHz */
    SystemClock_Config();
f7fffffe        BL       SystemClock_Config

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_GPIOD, ENABLE);
bf00            NOP
2000            MOVS     r0,#0
9000            STR      r0,[sp,#0]
4821            LDR      r0,|L3.152|
6800            LDR      r0,[r0,#0]
f0400008        ORR      r0,r0,#8
491f            LDR      r1,|L3.152|
6008            STR      r0,[r1,#0]
4608            MOV      r0,r1
6800            LDR      r0,[r0,#0]
f0000008        AND      r0,r0,#8
9000            STR      r0,[sp,#0]
bf00            NOP
bf00            NOP
```

Рис. 2. Фрагмент Listing файлу, що відображає відповідність між текстом програми мовою С та асемблерним кодом

Цей етап передбачає необхідність проведення аналізу тексту програми та асемблерного коду. Аналіз тексту програми дає змогу визначити відповідність обчислювальних операцій та апаратного модуля чи інтерфейсу. Аналіз асемблерного коду дає змогу визначити перелік асемблерних інструкцій, необхідних для виконання певної операції, а відтак і кількість тактів, які необхідні для її виконання. Вхідними даними для розрахунку тривалості виконання програмного коду є Listing-файл, що генерується у процесі компілювання програмного коду середовищем розробки Keil uVision (рис. 2).

Загальна тривалість виконання функції обчислюється за формулою:

$$t_f = \sum_{i=0}^{j} \left(n_c \cdot t_{op}\right) + t_r, \quad (8)$$

де j — кількість асемблерних інструкцій у функції; n_c — кількість тактів, які виконують асемблерну інструкцію; t_{op} — тривалість виконання однієї обчислювальної операції, або тривалість надсилання одного байту інформації через інтерфейс зв'язку; t_r — тривалість читання інструкції з пам'яті програм.

Експерименти

Дослідження запропонованого методу проводилось для одного з потоків програми, що виконується на вбудованій системі керування гідравлічним навантажувачем. Вбудована система складається з пульта керування та блоку керування гідравлічним навантажувачем, що з'єднуються через радіоканал. Потік програми, відповідальний за аварійну зупинку гідравлічної системи, запускається з натисканням клавіші аварійної зупинки на пульті керування. Цей потік оновлює структури даних, що відповідають за контроль гідравлічних клапанів, та надсилає їх через інтерфейс UART до Wi-Fi модуля, який передає отримане повідомлення до блока керування гідравлічним навантажувачем. Після надсилання повідомлення потік очікує відповідь про отримання відповідної команди.

Згідно з технічним завданням на пульт керування, критичним з точки зору безпечності є час формування повідомлення та його надсилання до Wi-Fi модуля. Для визначення тривалості виконання фрагмента програмного коду, що відповідає за виконання описаного функціоналу, використано три методи: метод аналізу тривалості виконання програмного коду, який описано у [7], запропонований авторами метод розрахунку, а також модифікований для локального

використання метод вимірювання тривалості виконання програмного коду [10]. Результати подано у таблиці.

Тривалість виконання фрагмента програми

Виміряне значення, с	Отримане згідно з запропонованим методом, с	Розраховане згідно з методом, описаним у [7]
$11{,}7426 \cdot 10^{-3}$	$11{,}5025 \cdot 10^{-3}$	$11{,}3984 \cdot 10^{-3}$

Відносна похибка в прогнозуванні тривалості виконання програмного коду із застосуванням запропонованого методу складає –2,045 %, що зумовлено відсутністю даних про тривалість операцій читання/запису даних з оперативної пам'яті. Відносна похибка в прогнозуванні тривалості виконання програмного коду із застосуванням методу, запропонованого у [7], складає –2,93 %, що зумовлено суттєвими відмінностями між архітектурою мікроконтролера, для якого розроблялася ця модель, та архітектурою мікроконтролера STM32. Також варто зазначити, що близько 1 % похибки внесено за рахунок наближеного розрахунку тривалості надсилання 40 байт даних через інтерфейс зв'язку UART.

Подальше зменшення похибки аналізу тривалості виконання програмного коду можливе зі збільшенням адекватності запропонованого методу, зокрема врахуванням тривалості читання та запису даних у оперативну пам'ять.

Висновки

Розв'язано актуальну задачу аналізу тривалості виконання програмного коду вбудованих систем з урахуванням архітектури мікроконтролера.

Наукова новизна роботи полягає в тому, що запропонований метод дає змогу оцінити тривалість виконання програмного коду з урахуванням швидкості роботи внутрішніх модулів та інтерфейсів зв'язку мікроконтролера, що підвищує точність отримуваних даних.

Перспективи подальших досліджень полягають у підвищенні точності розрахунку шляхом врахування тривалості читання даних з оперативної пам'яті, а також автоматизації процесу розрахунку тривалості виконання програми для подальшого використання як засобу верифікації методів вимірювання тривалості виконання програмного коду.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] T. Ringler, "Static Worst-Case Execution Time Analysis of Synchronous Programs," *Reliable Software Technologies Ada-Europe 2000*, vol. 1845, pp. 56-68, 2000.

[2] A. Tavares and C. Couto, "A machine independent WCET predictor for microcontrollers and DSPs [worst case execution time]," in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings* (Cat. No.01TH8570), 2001.

[3] C. Healy and D. Whaley, "Tighter timing predictions by automatic detection and exploitation of value-dependent constraints," *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium*, 1999.

[4] J. Engblom, "Processor pipelines and static worst-case execution time analysis." Uppsala: Acta Universitatis Upsaliensis, 2002.

[5] R. Wilhelm, D. Grund, J. Reineke, M. Schlickling, M. Pister and C. Ferdinand, "Memory Hierarchies, Pipelines, and Buses for Future Architectures in Time-Critical Embedded Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 7, pp. 966-978, 2009.

[6] Xianfeng Li, A. Roychoudhury and T. Mitra, "Modeling Out-of-Order Processors for Software Timing Analysis," in *25th IEEE International Real-Time Systems Symposium*, 2004.

[7] P. Atanassov, R. Kirner and P. Puschner, "Using real hardware to create an accurate timing model for execution-time analysis," in I*EEE Real-Time Embedded Systems Workshop held in conjunction with RTSS 2001*, 2001.

[8] *RM0090 Reference manual STM32F4, 2nd ed. STMicroelectronics*, 2017.

[9] "ARM Information Center," Infocenter.arm.com, 2017. [Online]. Available: http://infocenter.arm.com/help/index.jsp. [Accessed: 14- Dec- 2017].

[10] R. Chopey, B. Knysh and D. Fedasyuk, "The model of software execution time remote testing," in *IXth Intern. Conf. of Young Scientists «Computer Science and Engineering 2017» (CSE'2017)*, Lviv Polytechnic National University, 2017, pp. 398–402.

Рекомендована кафедрою захисту інформації ВНТУ

Стаття надійшла до редакції 20.12.2017

Чопей Ратібор Степанович — аспірант кафедри програмного забезпечення, e-mail: chopey.ratybor@gmail.com ;

Федасюк Дмитро Васильович — д-р техн. наук, професор, проректор, e-mail: dmytro.v.fedasyuk@lpnu.ua.

Національний університет «Львівська політехніка», Львів

R. S. Chopei[1]
D. V. Fedasiuk[1]

Execution Time Analysis Method Taking Into Account Architecture of Microcontroller

[1]Lviv Polytechnic National University

The paper deals with common problems that are likely to be encountered when analyzing the execution time of the firmware of any typical embedded system. The overview of the existing approaches performed by the authors has revealed that methods based on analysis of the cash memory and the instruction pipeline architecture of the embedded system being investigated are impractical for evaluating the execution time of the firmware running in microcontroller-based embedded systems. One of the ways of making the results of firmware execution time analysis more accurate is to take into consideration the internal architecture of the microcontroller including delays introduced by data exchange between internal blocks of the microcontroller via modules that implement standard communication interfaces.

We have proposed a method for firmware execution time analysis that assumes the following steps: determining the settings of the microcontroller on the basis of the firmware, determining interconnections between the synchronization buses and the internal blocks, analysis of the settings of communication interfaces for external peripheral devices, and calculation of the firmware execution time itself. Upon the performed multiple numerical experiments, one can conclude that the proposed method provides more accurate results in comparison with the existing methods.

The authors are planning to enhance the accuracy of calculating the firmware execution time due to making allowances for the duration of reading data from random access memory and due to the fact that the proposed method might be automated, in order to reduce the time of analysis for commercial projects. After being automated, the introduced method is applicable for verification of different methods for measurement of the firmware execution time.

Keywords: embedded systems, microcontroller architecture, Keil uVision, execution time.

Chopei Ratybor S. — Post-Graduate Student of the Chair of Software, e-mail: chopey.ratybor@gmail.com ;
Fedasiuk Dmytro V. — Dr. Sc. (Eng.), Professor, Vice-Rector, e-mail: dmytro.v.fedasyuk@lpnu.ua

Р. С. Чопей[1]
Д. В. Федасюк[1]

Метод анализа продолжительности выполнения программного кода с учетом архитектуры микроконтроллеров

[1]Национальный университет «Львовская политехника»

Рассмотрены общие проблемы, которые могут возникнуть при анализе продолжительности выполнения программного кода встроенной системы. Обзор существующих подходов, проведенный авторами, показал, что методы, основанные на анализе кэш-памяти и архитектуры конвейера команд микроконтроллера, неэффективны для оценки продолжительности выполнения программного кода, выполняемого на микроконтроллерных встроенных системах. Один из способов повышения эффективности анализа продолжительности выполнения программного кода — это учет внутренней архитектуры микроконтроллера, включая задержки, происходящие при обмене данными между внутренними модулями микроконтроллера и задержками при передаче данных через стандартные интерфейсы связи.

Предложен метод анализа продолжительности выполнения программного кода с учетом скорости обмена данными между внутренними модулями микроконтроллера и скорости передачи данных через интерфейсы связи. Численными экспериментами доказано повышение точности полученных результатов по сравнению с существующими методами.

Повышение эффективности предложенного метода, возможно путем учета продолжительности чтения данных из оперативной памяти, а также за счет автоматизации процесса расчета продолжительности выполнения программы.

Ключевые слова: встроенные системы, архитектура микроконтроллеров, Keil uVision, продолжительность выполнения программного кода.

Чопей Ратибор Степанович — аспирант кафедры программного обеспечения, e-mail: chopey.ratybor@gmail.com ;
Федасюк Дмитрий Васильевич — д-р техн. наук, профессор, проректор, e-mail: dmytro.v.fedasyuk@lpnu.ua