УДК 004.383

DOI: 10.18372/2073-4751.74.17885

Молчанов О.А.,
orcid.org/0000-0001-8384-0918,
Сергієнко А.М., д.т.н.,
orcid.org/0000-0001-5965-1789,
Романкевич В.О., д.т.н.,
orcid.org/0000-0003-4696-5935

МІКРОКОНТРОЛЕРИ ЗІ СТЕКОВОЮ АРХІТЕКТУРОЮ

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

aser@comsys.kpi.ua

Вступ

Програмовні логічні інтегральні схеми (ПЛІС) стають широко використованою елементною базою в транспорті, телекомунікаціях, авіакосмічній та військовій галузях, робототехніці, медицині. Широка номенклатура ПЛІС розділена на дві множини: ПЛІС великого та малого об'єму. Межа між цими множинами приблизно знаходиться на ємності у 20000 логічних таблиць (ЛТ). Причому ПЛІС великого об'єму часто у своєму складі мають апаратні ядра високопродуктивних мікропроцесорів з архітектурою ARM.

На відміну від них ПЛІС малого об'єму таких ядер не мають. Зате вони є значно дешевшими, меншими за розмірами і тому є більш використаними у різноманітних застосунках. При цьому при необхідності застосувати процесорне ядро розробник пристрою має змогу лише використати програмовне ядро (softcore). Найчастіше це 32-розрядне ядро з архітектурою MicroBlaze, Nios, OpenRISC або ARM. Причому таке ядро займає велику частку пам'яті ПЛІС і її ресурсів ЛТ, яка може досягати половини і більше, а також споживає суттєву частку енергії [1].

ПЛІС малого об'єму використовується у таких застосунках, в яких не потрібна велика функціональність вищеперелічених архітектур. Наприклад, це Інтернет речей (IoT). Натомість, для виконання алгоритмів керування, прийняття нескладних рішень, реалізації нескладних комунікаційних протоколів достатньо восьми- чи 16-розрядної архітектури. Але доступний для цього восьмирозрядний мікроконтролер Xilinx PicoBlaze є доволі примітивним і має невисоку швидкодію (одна 18-розрядна команда за два такти) [2]. Деякі клони звичайних мікроконтролерів, таких як i8051, ефективно використовуються для функцій IoT, але вони мають збільшений об'єм апаратного забезпечення, оскільки їх властивості архітектури не відповідають функціям ПЛІС [3, 4].

Архітектура Nanoblaze, запропонована в [4], за характеристиками швидкості та апаратного обсягу займає середнє місце між архітектурами Picoblaze та Microblaze. Ймовірно, вона добре підходить для цілей IoT. Але вона не досягла належного поширення через те, що її назва потрапляє в порушення авторських прав фірми AMD-Xilinx.

У даній статті розглядається розробка архітектур мікроконтролерів, які здатні вирішити проблему використання програмовних ядер у ПЛІС малого об'єму.

Аналіз проблеми

Еволюція мікроархітектури центрального процесора (ЦП) протягом десятиліть була спрямована на підвищення швидкості звичайних обчислень у різних областях. З цією метою використовується паралелізм рівня команд в напрямках конвеєрної обробки, суперскалярних обчислень, кешування даних і команд, передбачення розгалужень, динамічного планування, спекулятивних обчислень тощо. Як результат, один процесор може виконувати, в середньому, до двох або трохи більше команд за такт синхросигналу з частотою

до кількох гігагерц. Ці досягнення є результатом збільшення апаратних витрат на кілька порядків і енергоспоживання до десятків Ватт. Але в даний час удосконалення процесора припинилося, в основному, через обмеження закону Мура і закону масштабування Деннарда [5].

Тепер очікується еволюція мікроархітектури ЦП у напрямку її вдосконалення в конкретних областях застосування. Один із успішних підходів базується на додаванні складних прикладних команд, реалізованих у ПЛІС, яка знаходиться поблизу ЦП. У сучасних мікропроцесорах неефективно реалізовані логічні алгоритми, що потребують прийняття рішень. Це пов'язано з фактом частих зупинок конвеєра ЦП в результаті неправильного передбачення розгалужень [5]. Звідси виходить ідея взагалі відмовитись від конвеєра команд.

Для впровадження системи IoT у ПЛІС важливо мати настроюваний мікроконтролер з мінімізованим апаратним і програмним забезпеченням. Останнє продиктовано тим, що блоки пам'яті, які вбудовані в ПЛІС, мають істотно обмежений обсяг. Бажано мати такий мікроконтролер, набір команд якого може бути налаштований програмістом вручну під потреби проекту, щоб спростити програмування, дозволити повторне використання підпрограм і, як наслідок, мінімізувати довжину програми. Його набір команд повинен бути адаптований до передачі даних через мікроконтролерні інтерфейси.

Мета

Метою статті є створення програмовних ядер мікроконтролерів, які є найбільш придатними для реалізації у ПЛІС малого об'єму, що використовуються, наприклад, для виконання задач IoT.

Стекова архітектура

Серед усіх мікропроцесорних архітектур можна виділити стекову архітектуру процесора. Її набір команд відрізняється тим, що операнди мають неявну адресацію, оскільки вони зазвичай розміщуються в кількох фіксованих регістрах стека. Такі команди мають коротку довжину, оскільки у них неявна адресація регістрів. Оскільки ці команди підтримують алгоритми, які активно використовують стекову адресацію, то програми, складені для цього процесора, займають дуже малий об'єм пам'яті [6].

Різними авторами розроблено кілька проєктів стекових процесорів, які реалізовані в ПЛІС і які доступні для відтворення [7–9]. Усі вони мають 16-розрядні команди та обробляють 16-розрядні дані. У [9] показано, що стековий процесор має приблизно в 2,5 рази меншу довжину програми, ніж програма для процесора Xilinx MicroBlaze при розробці проекту IoT. Крім того, усі стекові процесори дають змогу розробнику розширювати систему команд. Для цього необхідно внести відповідні зміни в опис процесора на рівні регістрових передач.

Отже, архітектура стекового процесора забезпечує мінімізацію як довжини програми, так і апаратних витрат. Крім того, для такої архітектури легко розробити компілятор, оскільки, як правило, його система команд є підмножиною операторів мови Forth. Мова асемблера стекового процесора також має той самий синтаксис, що й мова Forth [6]. Тому привабливо розвивати стекову архітектуру процесора, яка дає не тільки мінімізовані витрати на апаратне забезпечення, але й спрощену реалізацію команд користувача.

Мікроконтролер SM8

Розроблено мікроконтролер SM8, який має стекову архітектуру. Його структура наведена на рис. 1. Цей 8-розрядний мікроконтролер має лічильник програм PC, оперативний запам'ятовуючий пристрій (ОЗП) даних DRAM, який суміщений з постійною пам'яттю програм PROM, регістр команди IR, блок кодування команд користувача UIE, стек адрес повернення RSstack, стек даних DStack, арифметико-логічний пристрій ALU і периферійні регістри R0 ,..., Rn, n < 32. Регістри T, N є крайніми регістрами DStack і призначені для зберігання операндів і результатів ALU. PROM має обсяг до 7936 байт, а DRAM – до 256 байт.

Система команд ядра має 23 базові команди. Усі команди мікроконтролера, крім 16-розрядних команд виклику підпрограми CALL, читання константи LIT і умовного переходу IF, мають довжину 8 біт. Команди CALL, LIT, IF та команди введення-виведення виконуються за два такти, а решта команд – за один такт. Завдяки частому використанню команд CALL, LIT і IF, середня тривалість виконання однієї команди приблизно дорівнює 1,5 тактам.

Визначені користувачем команди реалізуються наступним чином. По-перше, код команди відображається у зазначену адресу в бібліотеці підпрограм, де розташована підпрограма, визначена користувачем. По-друге, коли керування програми попадає на цю команду, її код записується в регістр IR. Потім він перекодується в UIE у адресу відповідної підпрограми. Причому адреса повернення (адреса наступної команди) зберігається у вершині стеку R. Після цього потік керування передається до першої команди підпрограми.

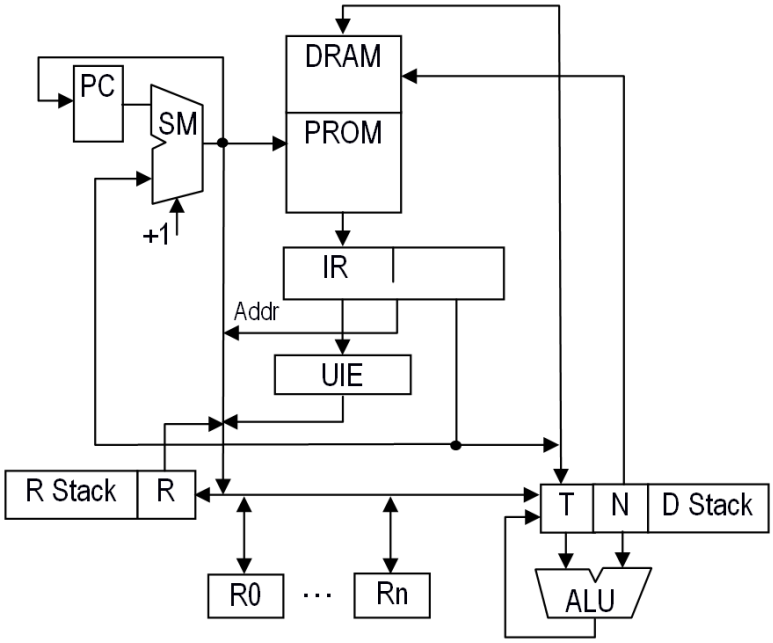

Рис. 1. Структура мікроконтролера SM8

Повернення потоку керування від підпрограми виконання команди користувача до наступної команди виконується командою RET.

Команда користувача закодована одним байтом на відміну від двобайтової команди CALL. Таким чином, такі команди можуть додатково заощадити обсяг пам'яті програмного забезпечення порівняно з еквівалентною реалізацією програми за допомогою команд CALL.

Команди користувача можна зберігати як у PROM, так і в DRAM. Таким чином, мікроконтролер може зберігати певний сценарій динамічної обробки даних, який формується такими командами. Наприклад, він може виконувати розбір командних рядків, таких як ввід з клавіатури.

Мікроконтролер SM16

16-розрядне мікроконтролерне ядро SM16 є наступником ядра SM8. Його структура на рис. 2 схожа на структуру на рис. 1 за винятком того, що DRAM і PROM є відокремленими і відсутній блок UIE. Крім того, додані індексні регістри A і B, які спрощують обробку масивів даних.

Система команд SM16 має 43 команди з модифікаціями. Команди процесора виконуються за один такт за винятком команд розгалуження, які виконуються за два такти. Усі команди мають розрядність 16. Команда містить від одного до трьох полів коду операції F1, F2, F3 і поле D змінної довжини, яке зберігає константу або адресу переходу (рис. 3).

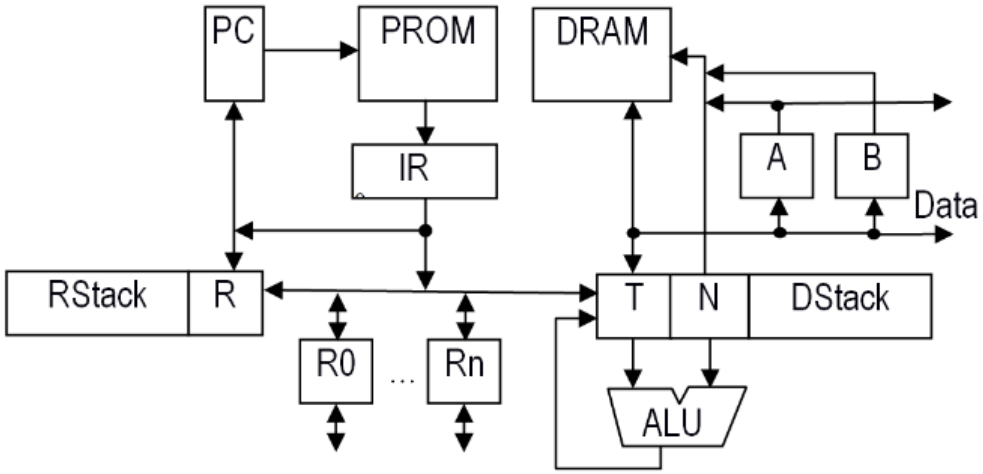

Рис. 2. Структура мікроконтролера SM16

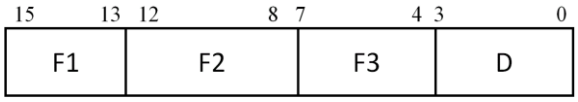

Рис. 3. Формат команди SM16

Процесор може виконувати до трьох операцій F1, F2, F3 одночасно. Наприклад, команда мовою асемблера

:L1 HASH @ab+ L1 DJNZ

обчислює хеш-код з байтом у регістрі T і кодом у регістрі N. Якщо [R] ≠ 0, то потік керування передається до мітки L1, а стан регістра R, що служить лічильником циклів, зменшується. В іншому випадку відбувається вихід із циклу. При цьому наступний байт за адресою у регістрі A зчитується в регістр T, потім ця адреса збільшується. Таким чином, за однією командою розраховується хеш-код цілого рядка, який зберігається у DRAM. Розрахований код використовується як адреса доступу до хеш-таблиці.

Програмування стекового процесора зазвичай виконується в стилі "шитого" коду, коли програма виглядає як послідовність викликів підпрограм. Це дає можливість отримати програми мінімальної довжини, що важливо для реалізації у ПЛІС. Можливість вставити операцію повернення RET в полі F1 в більшість команд і поєднати її з умовним розгалуженням зменшує як довжину підпрограм, так і їх тривалість. Процесор також має систему переривань. Оскільки контекст стекового процесора має мінімальний обсяг і зберігається апаратним чином, то накладні витрати на переривання також є незначними.

Експериментальні результати

Ядро мікроконтролера SM8 описано мовою VHDL. У табл. 1 наведено результати його конфігурування у ПЛІС різних серій, а саме, апаратні витрати у кількості ЛТ, тригерів і максимальна тактова частота F_{MAX}. При цьому встановлена глибина стеків RStack та DStack рівною 16, а загальний об'єм пам'яті DRAM і PROM – 1024.

Таблиця 1. Результати конфігурування ядра SM8 у ПЛІС

Серія ПЛІС	Апаратні витрати,		F_{MAX}, МГц
	ЛТ	тригерів	
Xilinx Spartan6	201	55	153
Xilinx Artix7	197	58	121
Altera Cyclone V	303	379	104
Lattice MachXO3	291	90	42

У табл. 2 наведене порівняння ядра SM8 з іншими ядрами мікропроцесорів різних архітектур при їх конфігуруванні у ПЛІС Spartan6. Там же приведене посилання на джерело. Її аналіз показує, що

мікроконтролер SM8 має суттєво менші апаратні витрати і більшу продуктивність ніж мікроконтролер з архітектурою i8085. Хоча мікроконтролер PicoBlaze має менші апаратні витрати, він має менші швидкодію та функціональність.

Таблиця 2. Результати конфігурування восьмирозрядних ядер у ПЛІС Spartan6

Ядро	Апаратні витрати, ЛТ	F_{MAX}, МГц	Продуктив-ність, команд/сек
SM8	201	153	100
FS8051 [10]	1293	89	30
PicoBlaze [2]	87	140	70

Ядро SM8 було успішно випробуване як мікроконтролер в проєкті цифрової FM-радіостанції для забезпечення управління та передачі даних через послідовні інтерфейси, такі як I2C, SPI, RS232 зі спеціалізованими протоколами [11].

Розроблене ядро мікроконтролера SM16 було застосовано для виконання завдання аналізу XML-пакетів, що поступають через Інтернет. Для цього до ядра ПЕ було приєднано три додаткові стеки та хеш-таблицю, а також команди, які підтримують процес аналізу. Серед них команда HASH обчислює хеш-функцію, обробляючи ключові слова XML зі швидкістю один символ за 2 такти.

Для розробки матзабезпечення був розроблений симулятор архітектури разом з компілятором. При цьому компілятор генерує як блок постійної пам'яті програми разом з програмним кодом, так і хеш-таблицю, які описані мовою VHDL і які відповідають заданій граматиці XML-запитів. В результаті, мікроконтролер SM16 може обробляти XML-запити зі швидкістю приблизно 7,5 мегабайт на секунду.

У табл. 3 наведено характеристики ядра SM16 при його конфігуруванні у ПЛІС різних серій, а в табл. 4 – його порівняння з іншими подібними 16-розрядними ядрами ПЕ при конфігуруванні у ПЛІС Xilinx Spartan6.

Таблиця 3. Результати конфігурування ядра SM16 у ПЛІС

Серія ПЛІС	Апаратні витрати,		F_{MAX}, МГц
	ЛТ	тригерів	
Xilinx Spartan6	721	116	102
Xilinx Artix7	767	114	135
Altera Cyclone V	1020	1074	116
Lattice MachXO3	1436	90	48

Таблиця 4. Результати конфігурування 16-розрядних ядер у ПЛІС Spartan6

Ядро	Апаратні витрати, ЛТ	F_{MAX}, МГц	Продуктив-ність, команд/сек
SM16	721	102	68
MSL16 [7]	235	100	67
b16-small [8]	280	100	50
J1 [9]	342	106	70
MSP430 [13]	1240	65	25

Аналіз таблиць 3 і 4 показує, що запропоноване ядро SM16 займає невеликий обсяг апаратних ресурсів ПЛІС. У порівнянні з аналогічними ядрами стекових процесорів MSL16, b16-small, J1 дане ядро має дещо збільшені апаратні витрати за рахунок реалізації більш широкої та ефективної системи команд, а також наявності системи переривань. Але у порівнянні з ядром поширеної архітектури MSP430 воно виграє у 1.7 разів у апаратних витратах та у 2.7 разів у продуктивності, що доводить ефективність стекової архітектури процесорних ядер при їх реалізації у ПЛІС.

Висновки

Для інтеграції системи Інтернету речей у ПЛІС невеликого об'єму необхідно мати настроюваний мікроконтролер з обмеженим апаратним та програмним забезпеченням. Для спрощення програмування і зменшення обсягу програми потрібно мати мікроконтролер з настоюваним набором команд, які можуть бути налаштовані програмістом під конкретні вимоги проекту. Для пришвидшення виконання алгоритмів слід оптимізувати виконання команд умовних переходів.

Встановлено, що для таких цілей найкраще підходить стекова архітектура процесора, у якій мінімізовані як розмір зкомпільованого програмного коду, так і апаратні витрати. Крім того, в такій архітектурі швидко виконуються програми з великою кількістю умовних переходів.

Розроблені ядра мікроконтролерів SM8 та SM16 використовують стекову архітектуру і були успішно використані кількох проєктах. Зокрема, ядро SM16, дозволяє ефективно обробляти XML-запити та забезпечує невеликі апаратні витрати у ПЛІС. Загальне порівняння розроблених ядер з ядрами традиційних архітектур показує їх менші апаратні витрати і більшу продуктивність.

Ці ядра можуть бути ефективно використані у різноманітних пристроях, які реалізовані на ПЛІС з невеликим об'ємом.

Література

1. Meyer-Baese U. Embedded Microprocessor System Design using FPGAs. Springer, 2021. 509 p.

2. Chapman K. PicoBlaze for Spartan-6, Virtex-6, and 7-Series (KCPSM6). Xilinx, Inc., 2012. 118 p.

3. Processor Design. System-on-Chip Computing for ASICs and FPGAs. Nurmi, J. (ed.) Springer, 2007. 525 p.

4. Meyer-Baese U. Digital Signal Processing with Field Programmable Gate Arrays, 4th Ed. Springer, 2005. 930 p.

5. Hennessy J.L., Patterson D.A. Computer Organization and Design. The Hardware Software Interface. 2nd Ed. Elsevier Inc., 2021. 1131 p.

6. Koopman P. Stack computers: the new wave. Ellis Horwood, Mountain View Press, CA,1989. 234 p.

7. Leong P.H.W., Tsang P.K., Lee T.K. A FPGA Based Forth Microprocessor. *Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines* / Napa Valley, CA, USA, 1998. P. 254–255.

8. Paysan B. b16-small – Less is More. *EuroForth 2004 Proceedings* / Saarland, Germany, 2004. 8 p.

9. Bowman J., Garage W. J1: a small Forth CPU Core for FPGAs. *EuroForth 2010 proceedings* / Hamburg, Germany, 2010. 4 p.

10. Maslennikov O., Shevtshenko J., Sergyienko A. Configurable microcontroller array. *Proceedings. International Conference on Parallel Computing in Electrical Engineering* / Warsaw, Poland, 2003. P. 47–49.

11. Sergiyenko A., Molchanov O., Orlova M. Software/Hardware Co-design of the Microprocessor for the Serial Port Communications. *Advances in Intelligent Systems and Computing*. 2020. V. 938. P. 238–246.

12. Sergiyenko A., Molchanov O., Orlova M. Microcontroller for the Logic Tasks. *Information, Computing and Intelligent Systems*. 2021. No. 2. P. 1–9.

13. Girard O. OpenMSP430. OpenCores, Rev. 1.13. 2013.

Молчанов О.А., Сергієнко А.М., Романкевич В.О.

МІКРОКОНТРОЛЕРИ ЗІ СТЕКОВОЮ АРХІТЕКТУРОЮ

Для впровадження систем Інтернету речей, спеціалізованих систем на базі програмовних логічних інтегральних схем (ПЛІС) невеликого об'єму необхідна розробка ядер мікроконтролерів, які мають як невеликі апаратні витрати, так і мінімізовану довжину скомпільованого програмного коду, який виконується з необхідною продуктивністю.

Основне завдання дослідження полягає в розробці ефективної архітектури ядра мікроконтролера, яке конфігурується у ПЛІС невеликого об'єму.

У результаті дослідження встановлено, що таке ядро повинно мати стекову архітектуру. Така архітектура має мінімізовані як розмір зкомпільованого програмного коду, так і апаратні витрати. Крім того, в такій архітектурі швидко виконуються програми з великою кількістю умовних переходів, викликів підпрограм і частими перериваннями.

Запропоновано восьми- та шістнадцятирозрядні ядра мікроконтролерів з назвами SM8 та SM16, відповідно, які мають стекову архітектуру. До цієї архітектури додаються спеціалізовані команди користувача, які налаштовані на пришвидшення виконання алгоритму, що програмується. Так, є можливість додавати до ядра SM8 до десятків команд користувача, які виконуються за підпрограмами. За однією командою ядра SM16 виконується обчислення хеш-функції ключових слів зі швидкістю два такти на символ. Ядра відрізняються невеликими апаратними витратами і конфігуруються у ПЛІС різних серій та виробників. Зокрема, ядро SM16 має у 1.7 разів менші апаратні витрати та у 2.7 разів вищу продуктивність ніж ядро з архітектурою MSP430.

***Ключові слова**: FPGA, процесор, система команд, стекова архітектура, VHDL.*

Molchanov O.A., Sergienko A.M., Romankevich V.O.

MICROCONTROLLERS WITH STACK ARCHITECTURE

The development of microcontroller cores that have both low hardware costs and minimized length of compiled software code that is executed with the required performance is necessary for the implementation of Internet of Things systems, and application specific systems based on field programable gate arrays (FPGA) of small volume.

The main task of the research is to develop an efficient architecture of the microcontroller core, which is configured in a small-volume FPGA.

As a result of the research, it was established that such a kernel should have a stack architecture. Such an architecture minimizes both the size of the compiled software code and hardware costs. In addition, in such an architecture, programs with a large number of conditional instructions, subroutine calls and frequent interruptions are quickly executed.

Eight- and sixteen-bit microcontroller cores with the names SM8 and SM16, respectively, which have a stack architecture, are proposed. Application specific user instructions can be added to the instruction set that are configured to speed up the execution of the executed algorithm. So, it is possible to add to the SM8 core up to dozens of user instructions that are executed by subroutines. A single instruction of the SM16 core calculates the hash function of the input keywords at a rate of two clocks per symbol.

The cores are characterized by low hardware costs and can be configured in FPGAs of various series and manufacturers. In particular, the SM16 core has 1.7 times lower hardware costs and 2.7 times higher performance than the MSP430 architecture core.

***Keywords:** FPGA, processor, instruction set, stack architecture, VHDL.*