УДК 004.274, 004.451

В. А. Мельник, А. Ю. Кіт
Національний університет "Львівська політехніка",
кафедра безпеки інформаційних технологій

ПІДХІД ДО ОРГАНІЗАЦІЇ ФУНКЦІОНУВАННЯ САМОКОНФІГУРОВНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ НА ОСНОВІ ОС GNU/LINUX

Запропоновано підхід до організації функціонування самоконфігуровної комп'ютерної системи на основі операційних систем GNU/Linux на етапі компіляції, завантаження та виконання програми. Запропоновано методи, що забезпечують автоматичне та прозоре для користувача виконання компіляції та завантаження підпрограми універсального процесора в оперативну пам'ять, а конфігурації в реконфігуровне середовище на основі програмовних логічних інтегральних схем. Продемонстровано можливості розширення та застосування сучасних операційних систем GNU/Linux для організації ефективної роботи із пристроями реконфігуровної логіки.

Ключові слова: реконфігуровна комп'ютерна система, самоконфігуровна комп'ютерна система, операційна система, завантажувач, виконавчий файл.

This article shows the approach to the operation of the self-configurable computer system at compilation, download and execution stage. There were proposed the methods that provide automatic and transparent compilation and loading subprogram of universal processor into the CPU as well as configuration file into reconfigurable unit based on programmable logic device. There were demonstrated the capabilities of expansion and the use of modern operating systems for effective work with reconfigurable devices.

Key words: reconfigurable computer system, self-configurable computer system, operating system, loader, executable file.

Вступ

Сьогодні у комп'ютерній техніці існують два основні методи здійснення обчислень [1]. Перший ґрунтується на використанні спеціалізованих для виконання конкретного алгоритму або класу алгоритмів інтегральних схем (СВІС – спеціалізована велика інтегральна схема). Другий – передбачає використання процесорів з визначеним набором команд, таких як центральні процесори сімейства *x86-64*, *IA32* або *EM64T*. СВІС ефективні лише на вузькому класі завдань, для виконання яких вони були спроектовані. Натомість універсальні процесори гнучкі та здатні виконувати багато завдань, хоча і не здатні досягти такої швидкодії, як СВІС. Окрім СВІС та універсальних процесорів, віднедавна існує новий клас комп'ютерних засобів – реконфігуровні апаратні прискорювачі. Комп'ютерні системи на основі реконфігуровних апаратних прискорювачів називаються реконфігуровними. Вони дають змогу поєднати притаманну універсальним процесорам гнучкість із швидкодією спеціалізованих інтегральних схем і позбавлені таких недоліків універсальних комп'ютерів, як низька реальна продуктивність, висока споживана потужність та низька ефективність використання обладнання. Реконфігуровні комп'ютерні системи (РККС) можуть бути перепрограмовані фактично у будь-який момент у процесі їх використання. Зміна топології з'єднань в процесі експлуатації дає змогу досягти більшої гнучкості та універсальності порівняно із СВІС, які конфігуруються один раз у процесі виробництва і не можуть бути перепрограмовані для виконання іншого алгоритму.

Наступним кроком у розвитку РККС та високопродуктивних обчислень загалом є самоконфігуровні комп'ютерні системи (СККС). У СККС виконання трудомістких та часомістких процесів, а саме – проектування та синтез спеціалізованих процесорів в реконфігуровному середовищі, зміна конфігурації реконфігуровного середовища, перекладена з користувача на програмні засоби комп'ютерної системи. Це, своєю чергою, дає змогу використати усі потенційні можливості, які надаються властивістю зміни конфігурації реконфігуровного середовища, і робить подальшу розробку СККС та забезпечення її функціонування одним із найперспективніших напрямків діяльності у галузі високопродуктивних обчислень.

Огляд досліджень та публікацій

Сьогодні існують операційні системи, методи та програмні засоби для організації роботи реконфігуровних комп'ютерних систем, що реалізують різноманітні підходи до організації взаємодії універсального процесора із реконфігуровним середовищем. Деякі з існуючих підходів забезпечують підтримку роботи із реконфігуровним середовищем за допомогою розширення операційної системи *Linux*. Прикладом реалізації такого підходу є операційна система *BORPH* [2]. *BORPH* визначає поняття апаратного процесу для програм, призначених для виконання у реконфігуровному середовищі. У *BORPH* запропоновано концепцію об'єднання об'єктного коду програми та конфігурації в єдиний виконавчий файл. Сьогодні також існують підходи, що розширюють існуючі концепції програмування. Прикладом таких програмних засобів є *ReconOS* [3], яка поширює концепцію багатопотокового програмування на системи з реконфігуровною логікою.

У [4 та 5] розглянуто принципи структурної організації та функціонування СККС. Запропоновано концепцію побудови СККС та метод самоконфігурування, який на відміну від методу конфігурування РККС передбачає автоматичне виконання розподілу програми між універсальним комп'ютером та реконфігуровним середовищем, створення файла конфігурації реконфігуровного середовища та автоматичне створення у цьому середовищі спеціалізованого процесора. Це дало змогу скоротити час та зменшити складність опрацювання інформації, а також зняти обмеження, які накладає спеціалізація процесора, і забезпечити ефективне використання реконфігуровної логіки для виконання довільних задач. У [6] розглянуто спосіб розширення стандартного завантажувача операційної системи для здійснення завантаження підпрограми універсального процесора у центральний процесор, а конфігурації – у реконфігуровне середовище.

Постановка завдання

Одним із пріоритетних завдань для організації функціонування СККС є реалізація підтримки обчислювальної моделі СККС на рівні операційної системи. Це передбачає розробку відповідних розширень операційної системи. У цій роботі ми пропонуємо рішення, яке дає змогу з використанням стандартного інтерфейсу ОС *Linux* і програмного забезпечення *Altera Quartus II* виконувати автоматичне завантаження програми для виконання у СККС та частково забезпечити організацію функціонування СККС на етапі компіляції програми.

Метод опрацювання інформації у СККС

Метод опрацювання інформації у СККС детально описаний у [3 і 4] і зображений на рис. 1. Він складається із трьох етапів: компіляції програми, її завантаження на виконання і виконання. Згідно з цим методом користувач створює програму P_{in} мовою високого рівня та подає її на компіляцію для подальшого виконання у СККС.

На етапі компіляції програми у СККС виконується автоматичне створення конфігурації ПЛІС, які формують її реконфігуровне середовище. При цьому автоматично здійснюється:

1. Розподіл вхідної програми на підпрограми відповідно універсального процесора (P_{GPP}) та реконфігуровного середовища (P_{RCE}) так, щоб частини програми, які становлять її основне обчислювальне навантаження, виконувались у реконфігуровному середовищі, що дасть можливість скоротити загальний час виконання програми комп'ютерною системою порівняно з її виконанням виключно на універсальному процесорі.

2. Компіляція підпрограми P_{GPP} та генерація об'єктного файла (*obj*) для виконання в універсальному процесорі.

3. Створення *VHDL*-моделі спеціалізованого процесора для виконання підпрограми P_{RCE}.

4. Логічний синтез *VHDL*-моделі спеціалізованого процесора і генерування файлів конфігурації (*conf*) ПЛІС.

5. Збереження отриманого об'єктного файла *obj* та файла конфігурації реконфігуровного середовища **conf** = $\{conf_q\}$, q = 1..K_{FPGA}, де K_{FPGA} – кількість ПЛІС, які входять до складу реконфігуровного середовища. Із цих файлів (*conf* та *obj*) формується єдиний виконавчий файл, який зберігається у пам'яті.

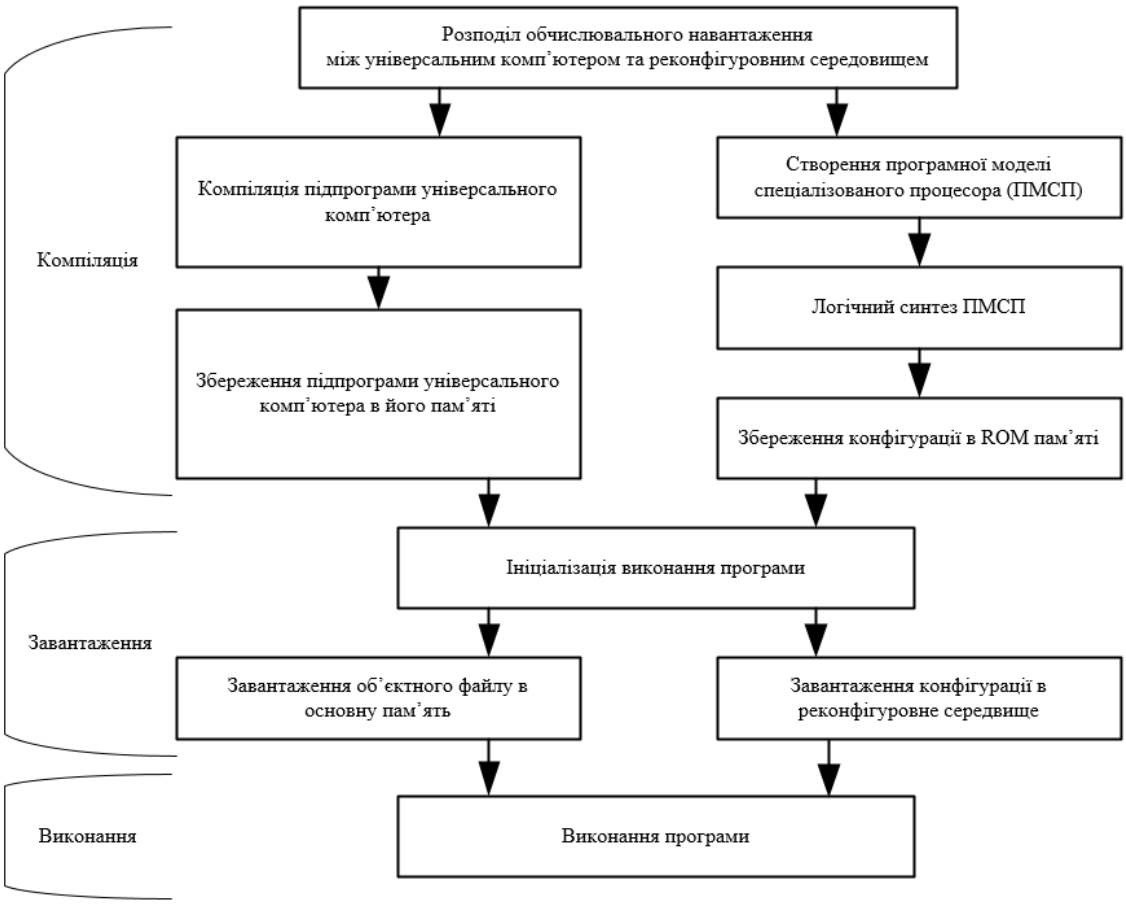

Рис. 1. Метод опрацювання інформації у самоконфігуровній комп'ютерній системі

На етапі завантаження програми одночасно із завантаженням об'єктного файла підпрограми універсального процесора в основну пам'ять СККС здійснюється завантаження файлів конфігурації ПЛІС у реконфігуровне середовище [7]. Після цього СККС виконує обидві підпрограми P_{GPP} і P_{RCE} та формує результат виконання програми P_{in}.

Деякі аспекти організації функціонування СККС на етапі компіляції програми

Завданням етапу компіляції програм у СККС є формування об'єктного коду та конфігурації ПЛІС. Формування об'єктного коду з програми, написаної мовою високого рівня, здійснюється компілятором. Наприклад, щоб отримати об'єктний код із С-програми, використовують компілятори *GCC*, *Clang* та ін. Сформувати файл конфігурації можна за допомогою засобів логічного синтезу. Одним із таких засобів є програмний продукт фірми *Altera – Quartus II*. За його допомогою ми формуємо із програмної моделі спеціалізованого процесора мовою *VHDL* файл конфігурації ПЛІС у форматі **.pof* (*Programmer Object Files*) або **.sof* (*SRAM Object Files*) (рис. 2). Важливою вимогою, яку ставлять до процесу формування файла конфігурації як складової частини

етапу компіляції програм в СККС, є його автоматичне виконання. Автоматичне формування файла конфігурації можна реалізувати за допомогою спеціального скрипту формування конфігурації. Такий скрипт повинен здійснювати налаштування системи та організовувати взаємодію із засобами логічного синтезу. Оскільки командний інтерпретатор *bash* є оболонкою за замовчуванням у багатьох дистрибутивах операційної системи *GNU/Linux* і уможливлює легко взаємодіяти з ОС, а мова скриптів *tcl* дає змогу легко взаємодіяти із середовищем *Quartus II*, для реалізації такого сценарію були обрані саме ці два інструментарії. Крім того, їх інструкції можна помістити в один файл сценарію.

Після формування файла конфігурації ПЛІС та об'єктного коду створюється виконавчий файл у форматі *SCCS*, призначений для виконання у СККС. Такий файл пов'язує між собою об'єктний модуль та відповідний йому файл конфігурації так, щоб їх паралельне завантаження в універсальний процесор та ПЛІС було прозорим для користувача.

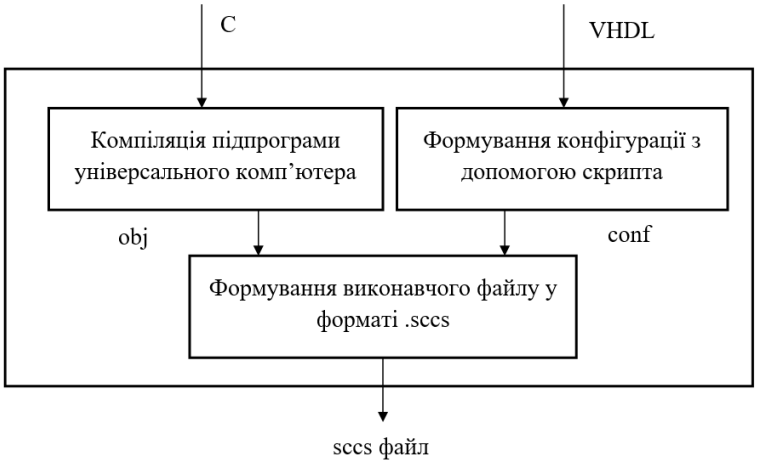

Рис. 2. Фрагмент етапу компіляції програми та формування виконавчого файла у самоконфігуровній комп'ютерній системі

Формат виконавчого файла СККС

Новий формат виконавчого файла для СККС реалізовано на основі формату *ELF*, що підтримується більшістю сучасних *UNIX*-подібних операційних систем. Виконавчий файл у форматі *ELF* містить такі компоненти: *ELF*-заголовок, в якому вказані загальні характеристики файла, таблиця програмних заголовків, яка використовується для описання сегментів файла, та таблиця заголовків секцій, яка характеризує секції файла (рис. 3, а). Формат *SCCS*, окрім стандартних для формату *ELF* розділів та таблиць заголовків, містить додатковий розділ *.conf* і є огорнутий новим заголовком СККС (рис. 3, б). Розділ *.conf* містить конфігурацію ПЛІС у закодованому згідно з *base64* вигляді.

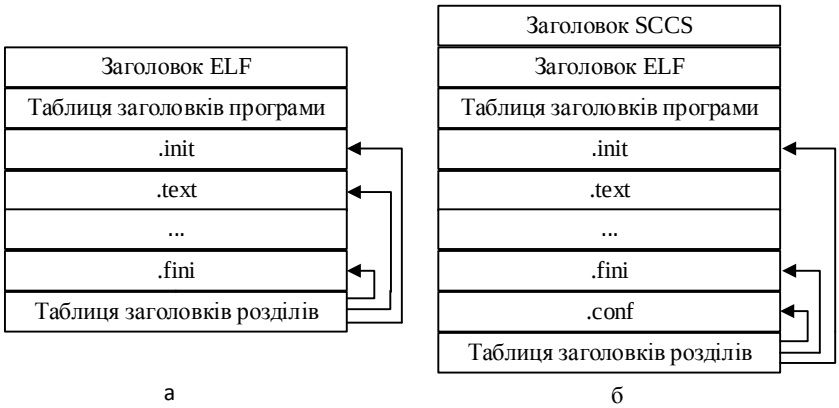

Рис. 3. Структури виконавчого файла ELF (а) і розширеного виконавчого файла SCCS (б)

Заголовок *SCCS* містить інформацію про використовуваний формат: його тип та формат файла конфігурації (*SRAM Object File, Programmer Object File*), який генерується з даних, що знаходяться у розділі *.conf*.

Підхід до організації функціонування СККС на етапі завантаження та виконання програми

В *UNIX*-подібних ОС, зокрема ОС *GNU/Linux*, завантаження програм виконує обробник системного виклику *execve*. Процес завантаження програм включає виконання системного виклику *execve*, виконання обробника системного виклику і виконання процедури ядра, яка викликає необхідний обробник. Для здійснення завантаження програм на виконання у СККС необхідно забезпечити можливість паралельного завантаження об'єктного коду програми на виконання в універсальному процесорі та конфігурації ПЛІС у реконфігуровне середовище. Для автоматичного виконання програм у СККС необхідно відповідно адаптувати та доповнити засоби завантаження програм в ОС *GNU/Linux*.

Завантаження програми і її виконання у СККС за допомогою розширеного завантажувача операційної системи включає такі кроки (рис. 4):

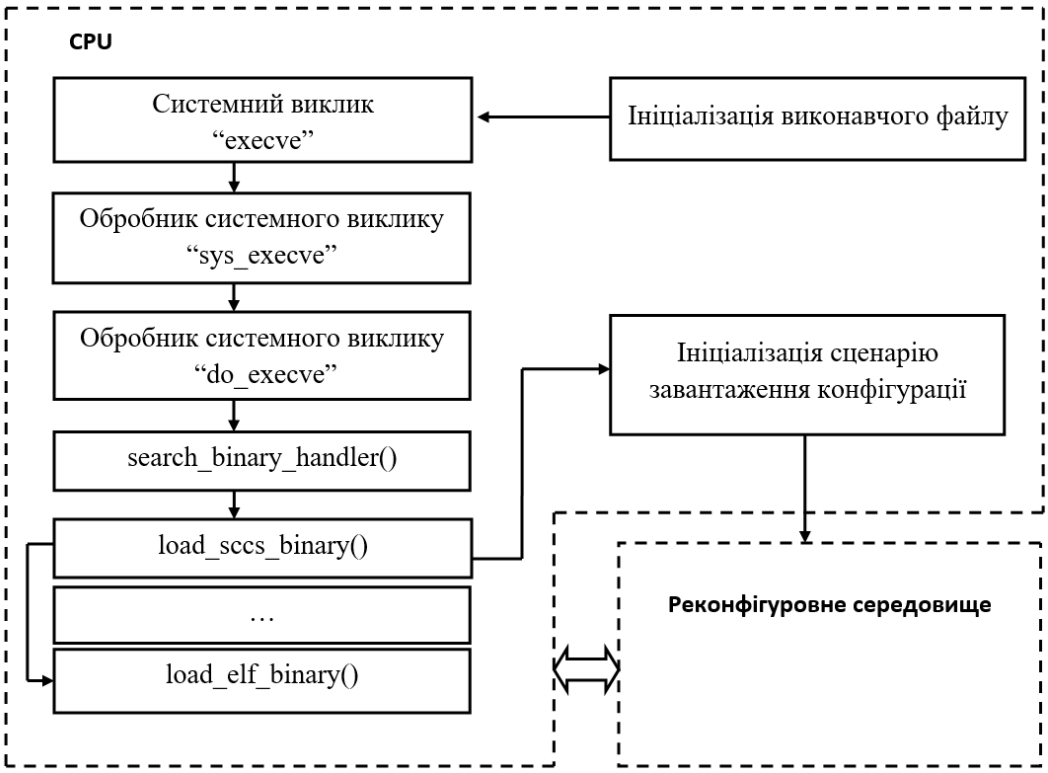

Рис. 4. Метод завантаження програми на виконання у самоконфігуровній комп'ютерній системі з використанням розширеного стандартного завантажувача ОС

1. Запуск на виконання виконавчого файла у форматі *SCCS*.
2. Здійснення командною оболонкою (*shell*) системного виклику "*execve*" з параметрами *argc / argv*.
3. Обробник системного виклику у ядрі отримує керування і починає його обробку. У ядрі обробник називається "*sys_execve*".
4. Універсальний обробник системного виклику у ядрі називається *do_execve*. Він створює і заповнює необхідні структури даних, копіює усю необхідну інформацію з простору користувача у простір ядра і викликає *search_binary_handler*(), який, своєю чергою, викликає відповідний

обробник. ОС *Linux* підтримує багато форматів виконуваних файлів, наприклад, *a.out* і *ELF*. Для забезпечення такої підтримки у ядрі є структура "*struct linux_binfmt*", яка містить вказівники на завантажувачі кожного з підтримуваних форматів. Отже, *search_binary_handler*() просто шукає потрібний завантажувач і викликає його. У нашому випадку – це *load_elf_binary*().

5. Своєю чергою, *load_elf_binary*() викликає *load_sccs_binary*(), завантажувач для розширеного формату виконавчого файла, призначеного для виконання у СККС.

6. Завантажувач розширеного формату виконавчого файла виділяє конфігурацію із структури виконавчого файла та запускає на виконання сценарій завантаження конфігурації.

7. Сценарій завантаження конфігурації викликає відповідні програмні засоби *Altera Quartus II* і завантажує конфігурацію у реконфігуровне середовище.

Цей метод є вдосконаленням методу, запропонованого у [6], проте завантажувач розширеного двійкового формату повністю реалізований на рівні ядра операційної системи, а на рівні користувача працюють лише конфігураційні скрипти. Це, своєю чергою, позбавляє користувача необхідності здійснювати будь-які додаткові налаштування системи чи встановлювати додаткові прикладні програми. Така операційна система одразу готова до роботи в умовах нової функціональності СККС.

Висновки

Одним із передових засобів у галузі високопродуктивних обчислень є самоконфігуровні комп'ютерні системи. У СККС виконання трудомістких та часомістких процесів, а саме: проектування та синтез спеціалізованих процесорів у реконфігуровному середовищі, зміну конфігурації реконфігуровного середовища перекладено з користувача на програмні засоби комп'ютерної системи.

У роботі, взявши за основу метод опрацювання інформації у СККС, запропоновано методи, що забезпечують автоматичне та прозоре для користувача виконання компіляції, формування виконавчого файла та завантаження на виконання програм в СККС на основі ОС GNU/Linux. Висвітлено деякі аспекти організації функціонування СККС на етапі компіляції програми. Розглянуто структуру розширеного виконавчого файла, використання якого дасть змогу пов'язати об'єктний модуль з модулем конфігурації ПЛІС та здійснити автоматичне завантаження програми на виконання у СККС. Наведено переваги удосконаленого методу завантаження програм на виконання порівняно із запропонованим у минулих дослідженнях методом завантаження програми на виконання у самоконфігуровній комп'ютерній системі з використанням розширеного стандартного завантажувача ОС.

1. Katherine Compton N., Hauck Scott. An Introduction to Reconfigurable Computing. 2. So H.K.-H., Brodersen R.: A unified hardware/software runtime environment for FPGA-based reconfigurable computers using BORPH. ACM Transactions on Embedded Computing Systems (TECS) 7 (2008). 3. E. L¨ubbers and M. Platzner, "Reconos: an RTOS supporting hard-and software threads,"in Proceedings of the International Conference on Field Programmable Logicand Applications (FPL '07), pp.441–446, August 2007. 4. Melnyk A., Melnyk V., "Self-Configurable FPGA-Based Computer Systems," Advances in Electrical and Computer Engineering, vol. 13, no. 2, pp. 33–38, 2013, doi:10.4316/AECE.2013.02005. [Online]. Available: http://www.aece.ro/abstractplus.php?year= 2013&number=2&article=5. 5. Melnyk V. "Self-Configurable FPGA-Based Computer Systems: Basics and Proof of Concept," Advances in Cyber-Physical Systems, vol. 1, no. 1, pp. 39–50, 2016. [Online]. Available: http://vlp.com.ua/files/special/ 9_117.pdf. 6. Viktor Melnyk, Andriy Kit "Program Loading and Execution in Self-Configurable Computer Systems using the Conventional Operating System Loader", Computer Science & Engineering (C5E), Lviv 2015 7. Melnyk V., Stepanov V., Sarajrech Z. System of load balancing between host computer and reconfigurable accelerator, Proceedings "Computer systems and components" of Tchernivtsi National University. – Tchernivtsi. 2012. – T. 3, Ed. 1. – P. 6–16.