УДК 004.274, 004.383

В. А. Мельник
Національний університет “Львівська політехніка”,
кафедра безпеки інформаційних технологій

ОСНОВИ ОРГАНІЗАЦІЇ ТА ЧАСОВІ ХАРАКТЕРИСТИКИ БАГАТОПРОЦЕСОРНИХ САМОКОНФІГУРОВНИХ КОМП’ЮТЕРНИХ СИСТЕМ

© Мельник В. А., 2016

Висвітлено принципи побудови і організації функціонування багатопроцесорних самоконфігуровних комп’ютерних систем. Розроблено спосіб опрацювання інформації в такій комп’ютерній системі та її структуру. Досліджено її часові характеристики. Визначено необхідні для досягнення високої продуктивності багатопроцесорної самоконфігуровної комп’ютерної системи умови та проаналізувано способи забезпечення їх виконання.

Ключові слова: багатопроцесорна самоконфігуровно комп’ютерна система, ПЛІС, високопродуктивні обчислення, реконфігуровні обчислення, реконфігуровна логіка.

ORGANIZATION BASICS AND TIMING CHARACTERISTICS OF THE MULTIPROCESSOR SELF-CONFIGURABLE COMPUTER SYSTEMS

© Melnyk V., 2016

The principles of design and operation of the multiprocessor self-configurabre FPGA-based computer systems are proposed in the article. The method of information processing and the structure of such system are developed. Its timing characteristics are explored. The conditions are determined necessary to achieve the high performance by the multiprocessor self-configurable computer system, and the approaches to implement these conditions are analyzed.

Key words: multiprocessor self-configurable computer system, field programmable gate arrays, high performance computing, reconfigurable computing, reconfigurable logic.

Вступ

Як відомо, в основу парадигми реконфігуровних обчислень покладено підхід поєднання універсальних програмовних процесорів та реконфігуровної логіки, в якій створюють спеціалізовані процесори з адаптованою до виконуваних алгоритмів архітектурою, та за потреби виконання інших алгоритмів замінюють ці процесори на інші, змінюючи конфігурацію реконфігуровної логіки. Такий підхід дозволяє забезпечити універсальність і необхідну продуктивність комп’ютерних систем. Впровадження парадигми реконфігуровних обчислень стало можливе з появою програмовних логічних інтегральних схем (ПЛІС) високого ступеня інтеграції та засобів їх конфігурування.

Комп’ютерну систему, до складу якої входять універсальний комп’ютер, реконфігуровне середовище на основі одного або більшої кількості кристалів ПЛІС для виконання обчислювально складних завдань, та програмне забезпечення, необхідне для зміни конфігурації реконфігуровного середовища, називають реконфігуровною. Об’єднання універсального комп’ютера з реконфігуровним середовищем в єдину систему дозволяє виконувати обчислювальні завдання значно швидше, ніж без нього, що забезпечується створенням в цьому середовищі спеціалізованих процесорів для апаратного виконання найбільш обчислювально складних частин цих завдань.

Поряд з високою продуктивністю, яка забезпечується реконфігуровними комп'ютерними системами (РККС), є й ряд проблем, пов'язаних з їх застосуванням. Це, зокрема, значні часові затрати на розподіл обчислювального навантаження, часта відсутність необхідних для реалізації в реконфігуровному середовищі РККС програмних моделей спеціалізованих процесорів, що викликає необхідність розробленя цих моделей від початку, та високі додаткові вимоги до кваліфікації користувачів РККС, оскільки вони, крім моделювання і програмування, повинні виконувати системний аналіз та розподіляти обчислювальне навантаження, розробляти архітектуру спеціалізованих процесорів, здійснювати їх синтез та реалізацію в реконфігуровному середовищі.

Вказаних проблем позбавлені самоконфігуровні комп'ютерні системи (СККС), в яких зазначені трудомісткі задачі з організації обчислювальних процесів повністю автоматизовано і перекладено з оператора на обчислювальну систему. Зважаючи на необхідність подальшого підвищення продуктивності комп'ютерних засобів, а також розширення застосування пристроїв реконфігуровної логіки в комп'ютерних системах, розвиток самоконфігуровних комп'ютерних систем сьогодні є актуальним завданням наукових досліджень та інженерних робіт.

Аналіз досліджень та публікацій

У роботі [1] запропоновано концепцію побудови самоконфігуровних комп'ютерних систем, яка позбавлена проблем, притаманних РККС. Самоконфігуровна комп'ютерна система (СККС) – це комп'ютерна система з реконфігуровною логікою, в якій компіляція програми включає автоматично виконувані дії з формування конфігурації, і яка набуває цієї конфігурації автоматично під час завантаження програми до виконання. Проблема інертності, притаманна РККС і викликана витратами часу на їх конфігурування, в СККС знімається завдяки зміні в ній способу опрацювання інформації. Також в роботі [1] запропоновано спосіб опрацювання інформації в СККС та її структуру.

Сьогодні існують програмні засоби, які можна використати для побудови СККС. Зокрема, в роботі [2] запропоновано підхід до побудови системи автоматичного розподілу обчислювального навантаження між універсальним комп'ютером та реконфігуровним прискорювачем. На ринку наявні засоби автоматичної генерації програмних моделей [3], [4], бібліотеки програмних моделей [5], а також системи високорівневого проектування програмних моделей спеціалізованих процесорів на основі опису алгоритмів їх роботи мовою високого рівня [6] – [8].

Постановка проблеми

З огляду на те, що СККС належать до класу високопродуктивних комп'ютерних систем, а останні, як правило, є багатопроцесорними, особливо важливим завданням є розроблення основ організації багатопроцесорних самоконфігуровних комп'ютерних систем. Ця робота є продовженням теоретичного дослідження, викладеного в роботі [1], і висвітлює загальні принципи організації функціонування та структурної організації багатопроцесорної самоконфігуровної комп'ютерної системи, а також показує її часові характеристики.

1. Багатопроцесорна СККС

Багатопроцесорність є традиційним підходом до підвищення продуктивності комп'ютерних систем і полягає у використанні двох або більшої кількості універсальних процесорів (процесорів загального призначення) в одній комп'ютерній системі. Високої продуктивності в багатопроцесорних комп'ютерних системах досягають завдяки паралельному виконанню процесорами різних обчислювальних процесів, причому процеси ці можуть належати як різним програмам, так і одній паралельній програмі. Створення паралельних програм є складним завданням і вимагає застосування методів і технологій паралельного програмування, відповідних мов програмування й інструментальних засобів.

Отже, сформулюємо визначення багатопроцесорної СККС. При цьому візьмемо за основу запропоноване в роботі [1] визначення СККС як "комп'ютерної системи з реконфігуровною логікою, в якій компіляція програми включає автоматично виконувані дії з формування конфігурації, і яка набуває цієї конфігурації автоматично під час завантаження програми до виконання".

В доповнення до цього визначення, під багатопроцесорною будемо розуміти СККС, до складу якої входять два або більше процесорів загального призначення, між якими розподіляється вхідна програма, і кожен з яких під час виконання своєї підпрограми взаємодіє з реконфігуровною логікою.

Також багатопроцесорною будемо вважати СККС, побудовану на основі одного чи більшої кількості багатоядерних процесорів загального призначення, де кожне ядро може виконувати окрему підпрограму.

2. Спосіб опрацювання інформації в багатопроцесорній СККС

Крім здійснення самоконфігурування згідно з описаним в роботі [9] методом, спосіб опрацювання інформації в багатопроцесорній СККС також повинен забезпечувати можливість паралельної роботи багатьох процесорів загального призначення так, щоб кожен з них під час виконання своєї підпрограми міг взаємодіяти із спеціалізованим процесором, синтезованим за реконфігуровною логікою.

Для забезпечення такої можливості візьмемо за основу описаний в роботі [1] спосіб опрацювання інформації в СККС і внесемо в нього необхідні доповнення і зміни, не змінюючи при цьому його суті і зберігаючи подання процесу опрацювання інформації трьома етапами: компіляції програми, її завантаження та виконання.

Спосіб опрацювання інформації в багатопроцесорній СККС полягає в наступному:

1. Користувач створює програму P_{in} мовою програмування високого рівня та подає її до СККС.

На першому етапі – компіляції програми, – багатопроцесорна СККС автоматично виконує такі дії:

- Розпаралелює вхідну програму P_{in} на k підпрограм, де значення k залежить від просторових властивостей описаного програмою P_{in} алгоритму та, можливо, від кількості універсальних процесорів в СККС. Отже, $P_{in} = P_{in_1} \,\&\, P_{in_2} \,\&\, ... \,\&\, P_{in_k}$.
- Розподіляє кожну підпрограму P_{in_i} на підпрограму P_{GPP_i} універсального процесора та підпрограму P_{RCE_i} реконфігуровного середовища і формує дві множини підпрограм – $\mathbf{P_{GPP}} = \left\{ P_{GPP_i} \right\}$ та $\mathbf{P_{RCE}} = \left\{ P_{RCE_i} \right\}$, $i = \overline{1....k}$, причому, якщо не враховувати залучені до підпрограми P_{GPP_i} команди звернення до підпрограми P_{RCE_i}, то можна записати, що $P_{in_i} = P_{GPP_i} \,\&\, P_{RCE_i}$.
- Компілює множину підпрограм $\mathbf{P_{GPP}}$ в об'єктний файл $\mathbf{obj} = \left\{ obj_i, i = \overline{1...k} \right\}$ паралельної програми для виконання на універсальних процесорах багатопроцесорної СККС.
- Генерує множину **ASPM** програмних моделей спеціалізованих процесорів $ASPM_i$, поданих мовою опису апаратних засобів, для виконання обчислювальних завдань, описаних підпрограмами P_{RCE_i}, $i = \overline{1....k}$.
- Виконує логічний синтез множини **ASPM** програмних моделей спеціалізованих процесорів і формує множину файлів конфігурації $\mathbf{conf} = \left\{ conf_q, q = \overline{1...K_{FPGA}} \right\}$ ПЛІС реконфігуровного середовища, де K_{FPGA} – кількість задіяних для реалізації **ASPM** кристалів ПЛІС.
- Формує з множини файлів **obj** і **conf** виконавчий файл **exe** програми P_{in} і зберігає його в пам'яті багатопроцесорної СККС (тут під пам'яттю маємо на увазі пам'ять для довготермінового зберігання інформації, яка зазвичай використовується для зберігання скомпільованих програм, наприклад, магнітний диск).

Отриманий в результаті компіляції програми P_{in} виконавчий файл **exe** містить дві секції: секцію об'єктного коду підпрограм P_{GPP_i}, $i = \overline{1..k}$ (значення k можна трактувати як кількість паралельних потоків у процесі), та секцію конфігурації ПЛІС реконфігуровного середовища яка містить набір кодів конфігурації цих ПЛІС. Формат виконавчого файла СККС запропоновано в

роботі [10]. Зазначимо, що задіяними можуть бути не всі кристали ПЛІС реконфігуровного середовища, а деяка їх частина, як і те, що кількість потоків може не обмежуватись кількістю процесорів K_{GPP}.

2. Після отримання від користувача команди ініціалізації виконання програми, на другому етапі – завантаження програми, багатопроцесорна СККС завантажує до виконання виконавчий файл **exe** програми P_{in}. При цьому до основної пам'яті завантажуються об'єктні коди **obj**, і одночасно до реконфігуровного середовища завантажуються коди конфігурації $\mathbf{conf} = \{conf_q, q = \overline{1...K_{FPGA}}\}$ та створюється в цьому середовищі множина спеціалізованих процесорів **ASP**.

3. Потім відбувається третій етап – виконання програми, на якому універсальні процесори GPP_i у взаємодії між собою виконують відповідні підпрограми P_{GPP_i}, а спеціалізовані процесори ASP_i у реконфігуровному середовищі виконують відповідні підпрограми P_{RCE_i}, $i = \overline{1....k}$, причому кожен з універсальних процесорів GPP_i взаємодіє з відповідним спеціалізованим процесором ASP_i відповідно до алгоритму роботи підпрограми P_{in_i} і з іншими універсальними процесорами відповідно до алгоритму роботи програми P_{in}.

4. У разі наявності виконавчого файла **exe**, а також в разі повторного виконання програми P_{in}, виконуються тільки другий і третій етапи.

Графічно спосіб опрацювання інформації в багатопроцесорній СККС зображено на рис. 1.

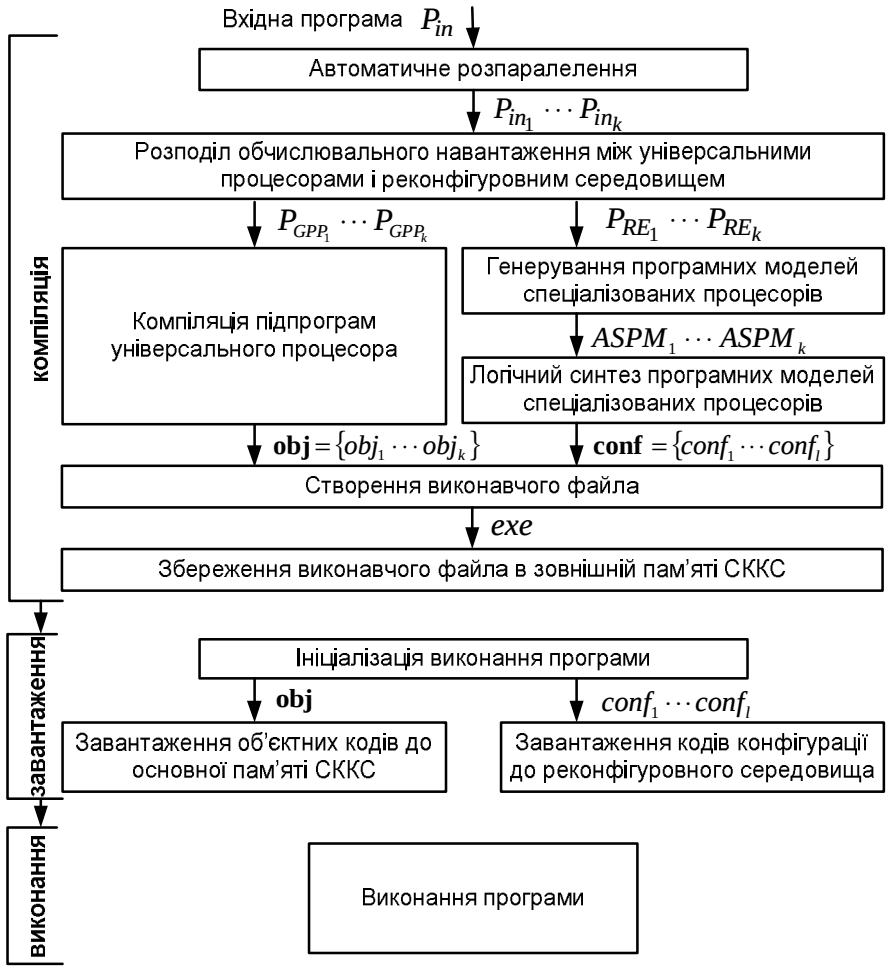

Рис. 1. Діаграма способу опрацювання інформації в багатопроцесорній СККС

Наведений вище спосіб опрацювання інформації в багатопроцесорній СККС передбачає застосування таких програмних засобів:

1. Засобів автоматичного розпаралелення, які виконують розподіл програми P_{in} на k паралельних підпрограм. Сьогодні доступна низка засобів автоматичного розпаралелення типу "*source-to-source*", які отримують на вході програми мовами *Fortran*, *C* та *C++*, та перетворюють їх на паралельний код, наприклад, трансформований багатопотоковий паралельний код з використанням функцій *pthreads* мови *C* і конструкцій *OpenMP*, або *OpenCL*, або навіть код платформи *CUDA*. Деякі з цих засобів розробляються як проекти з відкритим вихідним кодом (анг. *open source*), наприклад, *Par4All* [11], *PLUTO* [12], *The TRACO compiler* [13]. Серед інших – *Intel C++ Compiler* з автоматичним розпаралеленням, *Cetus* [14], розроблений в Університеті Пердью (*Purdue University*), *S2P Tool* [15], розроблений компанією *KPIT Cummins Infosystems Ltd*.

2. Системи розподілу обчислювального навантаження між універсальним процесором і реконфігуровним середовищем. Ця система автоматично знаходить у підпрограмах P_{in_i} такі фрагменти, виконання яких в реконфігуровному середовищі скорочує тривалість виконання цих підпрограм, і розподіляє кожну підпрограму P_{in_i} на підпрограму P_{GPP_i} універсального процесора, заміняючи в ній виокремлені фрагменти на команди роботи з реконфігуровним середовищем, та сформовану із цих фрагментів підпрограму P_{RCE_i} реконфігуровного середовища. Один з прикладів реалізації такої системи висвітлено в роботі [2]. Ця система формує підпрограму реконфігуровного середовища мовою асемблера *x86*, тому для застосування в СККС її необхідно доповнити засобами трансляції коду з мови асемблера на мову високого рівня. Засоби цього типу наявні на ринку, наприклад *Relogix Assembler-to-C Translator* [16] від компанії *MicroAPL*.

3. Компілятора для компіляції підпрограм P_{GPP_i} зі вхідної мови, на якій подаються ці підпрограми, в об'єктні коди obj_i, що безпосередньо можуть виконуватись універсальним процесором GPP_i.

4. Системи генерування програмних моделей комп'ютерних пристроїв, яка повинна автоматично генерувати програмні моделі спеціалізованих процесорів $ASPM_i$ з підпрограм P_{RCE_i}, наприклад, *Хамелеон* від компанії *Інтрон* [7, 17], *Agility Compiler* [18] та *DK4 Design Suite* [19] від компанії *Celoxica*, *CoDeveloper* від компанії *Impulse* [20].

5. Засобів логічного синтезу і конфігурування ПЛІС для логічного синтезу програмних моделей спеціалізованих процесорів $ASPM_i$ на етапі компіляції програми та конфігурування ПЛІС на етапі її завантаження до виконання. Ці засоби доступні від виробників ПЛІС, наприклад, *Vivado Design Suite*, *ISE*, *Alliance*, *Foundation* від фірми *Xilinx* [21], [22]; *Quartus II*, *Max + II* від фірми *Altera* [23], [24].

3. Принципи структурної організації багатопроцесорної СККС

Структуру багатопроцесорної самоконфігуровної комп'ютерної системи, яка реалізує запропонований вище спосіб опрацювання інформації, наведено на рис. 2. Як можна зауважити, основну частину базових програмних засобів СККС становить компілятор, який об'єднує описані вище засоби автоматичного розпаралелення вхідної програми, систему розподілу обчислювального навантаження, компілятор для компіляції підпрограм універсального процесора, систему генерування програмних моделей спеціалізованих процесорів, та засоби логічного синтезу програмних моделей спеціалізованих процесорів і конфігурування ПЛІС.

З отриманих в результаті компіляції програми множин об'єктних та бінарних кодів формується виконавчий файл програми (для цього використовуються засоби ство-

рення/розпаковування виконавчих файлів), який запам'ятовується в зовнішній пам'яті. Ці ж засоби використовують після ініціалізації виконання програми для розпаковування виконавчого файла і завантаження програми до виконання. Завантаження виконується завантажувачем ОС та засобами конфігурування ПЛІС (за посередництвом відповідного драйвера), про які йшлося вище.

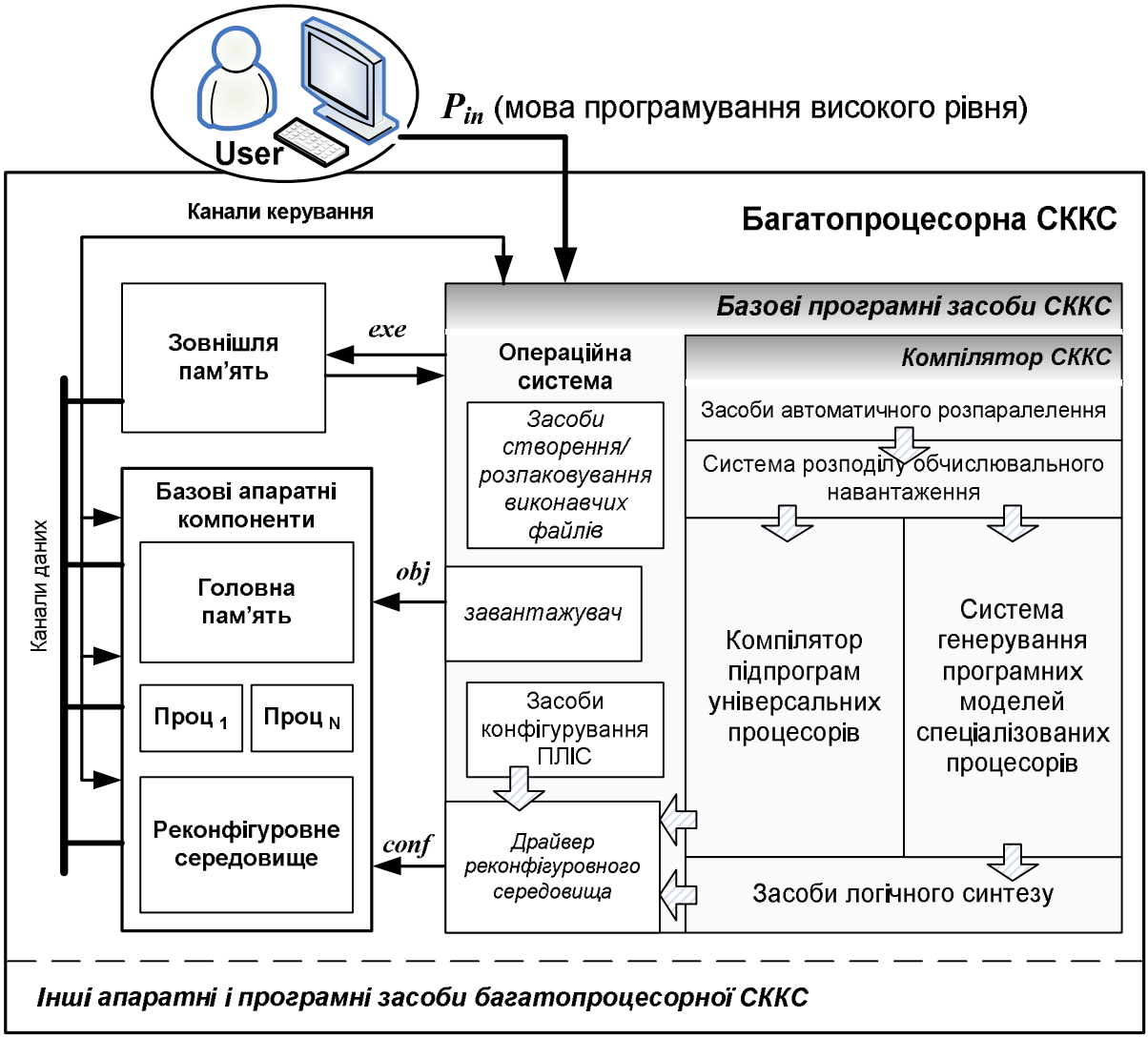

Рис. 2. Структура багатопроцесорної самоконфігуровної комп'ютерної системи

4. Часові характеристики опрацювання інформації в багатопроцесорній СККС

Нехай для виконання програми в СККС залучено K_{GPP} універсальних процесорів та K_{FPGA} кристалів ПЛІС, програма P_{in} після виконання розпаралелення представлена кількістю k паралельних підпрограм P_{in_i}, $i=\overline{1..k}$, а після розподілу навантаження визначено r спеціалізованих процесорів ASP_j, $j=\overline{1..r}$, які повинні бути синтезовані в реконфігуровному середовищі (важливо зазначити, що r не обов'язково дорівнює k, що буде описано нижче, але завжди $r \le k$). Тоді тривалість опрацювання інформації відповідно до програми P_{in} у багатопроцесорній СККС можна подати виразом:

$$T^{DP}_{MPSCCS}(P_{in}) = T^{COMPILE}_{MPSCCS}(P_{in}) + T^{LOAD}_{MPSCCS}(P_{in}) + T^{EXE}_{MPSCCS}(P_{in}), \quad (1)$$

причому:

$$T_{MPSCCS}^{COMPILE}\left(P_{in}\right)=t_{parallel}^{P}+\sum_{i=1}^{k}t_{balance}^{P}+\max\left(\sum_{i=1}^{k}t_{compile}^{GPP_i},\ \sum_{j=1}^{r}\left(t_{generate}^{ASPM_j}+t_{synth}^{ASPM_j}\right)\right)+t_{create}^{exe}+t_{store}^{exe} \quad (2)$$

де $t_{parallel}^{P}$ – тривалість розпаралелення програми P_{in}; $t_{balance}^{P}$ – тривалість розподілу підпрограми P_{in_i} на підпрограму P_{GPP_i} універсального процесора та підпрограму P_{RCE_i} реконфігуровного середовища; $t_{compile}^{GPP_i}$ – тривалість компіляції підпрограми P_{GPP_i} універсального процесора в об'єктний код; $t_{generate}^{ASPM_j}$ – тривалість генерування програмної моделі спеціалізованого процесора $ASPM_j$; $t_{synth}^{ASPM_j}$ – тривалість логічного синтезу програмної моделі спеціалізованого процесора $ASPM_j$; t_{create}^{exe} – тривалість створення виконавчого файла; t_{store}^{exe} – тривалість збереження виконавчого файла в зовнішній пам'яті (пам'яті для довготермінового зберігання даних).

Будемо виходити з того, що готова до виконання програма повинна бути попередньо скомпільованою і знаходитися в пам'яті СККС, а оскільки етапи компіляції та виконання програми логічно відокремлені і можуть бути рознесені в часі, тривалість виконання компіляції не визначає продуктивності СККС. Тому в двох випадках, а саме при повторному виконанні програми і при наявності її виконавчого файла тривалість опрацювання інформації відповідно до програми P_{in} у багатопроцесорній СККС можна подати виразом:

$$T_{MPSCCS}^{\prime DP}\left(P_{in}\right)=T_{MPSCCS}^{LOAD}\left(P_{in}\right)+T_{MPSCCS}^{EXE}\left(P_{in}\right). \quad (3)$$

Отже, продуктивність СККС залежить від того, як швидко програма завантажується до виконання і як швидко вона виконується. Тому надалі будемо розглядати характеристики опрацювання інформації в багатопроцесорній СККС саме на цих двох етапах.

Оскільки виконавчий файл програми містить секцію об'єктного коду та секцію конфігурації ПЛІС реконфігуровного середовища, тривалість етапу $T_{MPSCCS}^{LOAD}\left(P_{in}\right)$ завантаження програми P_{in} до виконання можна визначити за виразом:

$$T_{MPSCCS}^{LOAD}\left(P_{in}\right)=\max\left(t_{load}^{\mathbf{obj}},t_{load}^{conf_l}\right), \quad (4)$$

де $t_{load}^{\mathbf{obj}}$ – тривалість завантаження виконавчого файла **obj** багатопотокової програми універсальних процесорів до основної пам'яті; $t_{load}^{conf_l}$ – тривалість завантаження файлів конфігурації ПЛІС до реконфігуровного середовища; $l=\overline{1..K_{FPGA}}$.

Як показали дослідження, відображені в роботі [25], значення тривалості завантаження виконавчих файлів та значення тривалості завантаження файлів конфігурації є одного порядку, тобто $t_{load}^{conf}\approx t_{load}^{obj}$. Це підтверджує ефективність покладених в основу методу самоконфігурування та способу опрацювання інформації в СККС принципів, оскільки конфігурування ПЛІС під час завантаження програми в СККС не уповільнює етапу завантаження програми.

Тривалість $T_{MPSCCS}^{EXE}\left(P_{in}\right)$ етапу виконання програми P_{in} в багатопроцесорній СККС дорівнюватиме:

$$T_{MPSCCS}^{EXE}\left(P_{in}\right)=\sum_{i=1}^{k}T_{MPSCCS}^{EXE}\left(P_{in_i}\right)-T_{GPP}^{par}+\sum_{p=1}^{w}ttr_{GPP_w}-ttr_{GPP}^{par}, \quad (5)$$

де T_{GPP}^{par} – час, протягом якого дві або більше підпрограм (потоків) P_{in_i} виконуються відповідними процесорами паралельно; ttr_{GPP} – тривалість пересилання даних між

процесорами загального призначення під час виконання програми P_{in}; w – кількість сеансів пересилання даних; ttr_{GPP}^{par} – час, протягом якого два або більше процесів пересилання даних між універсальними процесорами здійснюються паралельно (тут прийнято, що час обміну даними між процесорами загального призначення під час виконання ними підпрограм P_{in_i} не входить до часу виконання підпрограм).

Позначивши тривалість виконання універсальним процесором GPP_i підпрограми P_{GPP_i} як $T_{GPP_i}\left(P_{GPP_i}\right)$, а тривалість виконання спеціалізованим процесором ASP_i поданого підпрограмою P_{RCE_i} алгоритму як $T_{ASP_i}\left(P_{RCE_i}\right)$, значення складової виразу (5) – тривалості $T_{MPSCCS}^{EXE}\left(P_{in_i}\right)$ виконання кожної з підпрограм (потоків) P_{in_i} можна описати таким виразом:

$$T_{MPSCCS}^{EXE}\left(P_{in_i}\right)=T_{GPP_i}\left(P_{GPP_i}\right)+T_{ASP_i}\left(P_{RCE_i}\right)-T_{GPP_i,\,ASP_i}^{par}+ttr_{GPP_i,\,ASP_i}, \quad (6)$$

де $ttr_{GPP_i,\,ASP_i}$ – тривалість пересилання даних між універсальним процесором GPP_i та спеціалізованим процесором ASP_i під час виконання підпрограми P_{in_i}; $T_{GPP_i,\,ASP_i}^{par}$ – час, протягом якого універсальний процесор GPP_i та спеціалізований процесор ASP_i виконують обчислення паралельно.

Як вже було зазначено, кількість r спеціалізованих процесорів ASP_j, $j=\overline{1..r}$ може бути меншою або дорівнювати кількості k паралельних підпрограм P_{in_i}, $i=\overline{1..k}$. Це пов'язано з тим, що під час виконання розподілу навантаження в деякій підпрограмі P_{in_q} може бути виявлено, що ця підпрограма не містить обчислювально складних фрагментів – таких, виконання яких у реконфігуровному середовищі прискорить її виконання. Це може мати місце, якщо величина прискорення, отриманого в результаті перенесення до реконфігуровного середовища найбільш обчислювально складних фрагментів підпрограми P_{in_q}, нівелюється затратами часу на пересилання інформації між процесорами GPP_q і ASP_q під час виконання ними відповідних підпрограм P_{GPP_q} і P_{RCE_q}, тобто не виконується основна мета розподілу навантаження, як це показано виразом (7):

$$ttr_{GPP_q,\,ASP_q} > T_{GPP_q}\left(P_{in_q}\right)-\left(T_{GPP_q}\left(P_{GPP_q}\right)+T_{ASP_q}\left(P_{RCE_q}\right)-T_{GPP_q,\,ASP_q}^{par}\right), \quad (7)$$

де $T_{GPP_q}\left(P_{in_q}\right)$ – тривалість виконання підпрограми P_{in_q} (отриманої перед розподілом навантаження) процесором GPP_q.

У цьому випадку немає змісту, а радше збитково переносити частину підпрограми P_{in_q} для виконання в реконфігуровному середовищі, тому вона повинна виконуватись в процесорі GPP_q повністю.

Враховуючи можливість існування деякої кількості підпрограм, для яких умова (7) є справедливою (ця кількість рівна $k-r$), вираз (5) варто записати в наступній, дещо зміненій формі:

$$T_{MPSCCS}^{EXE}\left(P_{in}\right)=\sum_{i=1}^{r}T_{MPSCCS}^{EXE}\left(P_{in_i}\right)+\sum_{q=1}^{k-r}T_{GPP_q}\left(P_{in_q}\right)-T_{GPP}^{par}+\sum_{p=1}^{w}ttr_{GPP_w}-ttr_{GPP}^{par}. \quad (8)$$

Зрозуміло, що у випадку, коли $r=k$, складова $\sum_{q=1}^{k-r}T_{GPP_q}\left(P_{in_q}\right)$ у виразі (8) буде відсутня, і вирази (5) та (8) будуть еквівалентними.

Отже, методика визначення тривалості виконання програми P_{in} у багатопроцесорній СККС полягає в наступному:

1. У випадку одноразового виконання програми:

a. визначення тривалості першого етапу опрацювання інформації в СККС – компіляції програми, за виразом (2);

b. визначення тривалості другого етапу опрацювання інформації в СККС – завантаження програми, за виразом (4);

c. визначення тривалості третього етапу опрацювання інформації в СККС – виконання програми, за виразами (6) і (8);

d. визначення суми тривалості всіх трьох етапів опрацювання інформації в СККС за виразом (1);

2. У випадку повторного виконання цієї самої програми виконуються кроки 2 і 3 і потім використовується вираз (3).

З виразів (8) і (6) можна зауважити, що для забезпечення якомога меншого значення тривалості $T^{EXE}_{MPSCCS}\left(P_{in}\right)$ етапу виконання програми P_{in} в багатопроцесорній СККС необхідне виконання системи умов:

$$\begin{cases} \dfrac{T_{GPP_i}\left(P_{GPP_i}\right)+T_{ASP_i}\left(P_{RCE_i}\right)}{T_{GPP}\left(P_{in_i}\right)} = jT \to \min; \\ \min\left(T_{GPP_i}\left(P_{GPP_i}\right), T_{ASP_i}\left(P_{RCE_i}\right)\right) - T^{par}_{GPP_i,\,ASP_i} = dT \to \min; \\ ttr_{GPP_i,\,ASP_i} \to \min; \\ \sum\limits_{p=1}^{w} ttr_{GPP_w} \to \min, \end{cases} \tag{9}$$

де $T_{GPP}\left(P_{in_i}\right)$ – тривалість виконання підпрограми P_{in_i} універсальним процесором.

Проаналізуємо ці умови.

Для того, щоб забезпечити якомога менше значення відношення jT, підпрограма P_{RCE_i} повинна містити такі частини підпрограми P_{in_i}, які складають найбільше обчислювальне навантаження (характеризуються найвищою обчислювальною складністю) і виконання яких вимагає найбільшої кількості ресурсів універсального процесора, оскільки виконання найбільш ресурсоємних частин програми в реконфігуровному середовищі дасть можливість досягти вищих показників її прискорення та повніше використати переваги підходу поєднання універсального процесора із спеціалізованим.

Значення різниці dT насамперед залежить від просторових характеристик описаного підпрограмою P_{in_i} алгоритму (наявності в ньому частин, які можна виконувати паралельно). Для забезпечення якомога меншого значення різниці dT ці характеристики повинні бути враховані під час розподілу підпрограми P_{in_i} на підпрограми P_{GPP_i} і P_{RCE_i}.

Тривалість $ttr_{GPP_i,\,ASP_i}$ пересилання даних між універсальним GPP_i та спеціалізованим ASP_i процесорами залежить від кількості N_{data} даних, що пересилаються, та від характеристик інтерфейсу: його пропускної здатності I_t, затримки I_d та часу реакції I_{rt}, тобто:

$$ttr_{GPP_i,\,ASP_i} = f\left(N_{data}, I_t, I_d, I_{rt}\right), \tag{10}$$

причому часові затримки I_d та I_{rt} виникають кожного разу під час ініціалізації сеансу обміну даними, тому обмін даними між підпрограмами P_{GPP_i} і P_{RCE_i} необхідно організувати так, щоб

кількість сеансів взаємодії та кількість даних, які потрібно передавати, були якомога меншими.

Виконання всіх наведених вище умов вирішується в СККС під час розподілу обчислювального навантаження між універсальним комп'ютером та реконфігуровним середовищем.

Подібно до складової $ttr_{GPP_i,\,ASP_j}$, значення складової ttr_{GPP} залежить від кількості даних, які передаються під час взаємодії між універсальними процесорами, та від пропускної здатності відповідних комунікаційних каналів. Отже, обмін даними між підпрограмами P_{in_i} і P_{in_j} необхідно організувати так, щоб кількість w сеансів взаємодії та кількість даних, які потрібно передавати, були якомога меншими. Завдання оптимізації за кількістю даних та сеансів взаємодії є традиційним для засобів автоматичного розпаралелення програм у багатопроцесорних комп'ютерних системах.

З виразу (9) можна зауважити, що тривалість $T_{SCCS}^{EXE}\left(P_{in}\right)$ етапу виконання програми P_{in} у багатопроцесорній СККС також залежить від значення $T_{ASP_i}\left(P_{RCE_i}\right)$, а отже, від продуктивності спеціалізованих процесорів ASP. Тому при реалізації цих процесорів потрібно застосовувати відповідні архітектурні підходи: апаратну орієнтацію та паралельне опрацювання даних, а також забезпечити максимальну ефективність використання виділеного під його реалізацію обладнання ПЛІС. Ці завдання вирішуються в СККС під час генерування програмних моделей спеціалізованих процесорів.

5. Оцінювання прискорення, забезпечуваного багатопроцесорною СККС

Під прискоренням будемо розуміти відношення тривалості виконання деякої програми P_{in} багатопроцесорною комп'ютерною системою на основі процесорів загального призначення до тривалості виконання цієї ж програми еквівалентною за кількістю і продуктивністю процесорів загального призначення багатопроцесорною СККС, при цьому етап компіляції програми не будемо враховувати з огляду на те, що компіляція програми є рознесена в часі з її виконанням. Також будемо вважати, що традиційній багатопроцесорній комп'ютерній системі відповідає й багатопроцесорна СККС за умови виконання нею обчислень без залучення реконфігуровного середовища.

Під час оцінювання забезпечуваного багатопроцесорною СККС прискорення враховуємо:

1. Тривалість завантаження програми P_{in} в багатопроцесорній СККС є того самого порядку, що й тривалість завантаження цієї програми в традиційній багатопроцесорній комп'ютерній системі на основі процесорів загального призначення (такий висновок можна зробити з досліджень, наведених в роботі [25]);

2. $P_{in_i} = \left(P_{GPP_i}\ \&\ P_{RCE_i}\right),\ i = \overline{1..r},\ r \le k$;

3. Виконання підпрограми P_{GPP_i} процесором GPP_i може закінчитись тільки після того, як закінчиться виконання підпрограми P_{RCE_i} процесором ASP_i;

4. Оскільки при виконанні підпрограм P_{in_i} кількість даних, які передаються під час взаємодії між процесорами загального призначення GPP_i в багатопроцесорній СККС не залежить від залучення реконфігуровного середовища до виконання обчислень, значення складових ttr_{GPP} в обох цих випадках буде однаковим.

Відповідно, значення прискорення виконання програми P_{in} у багатопроцесорній СККС можна визначити за виразом:

$$A_{MPSCCS}\left(P_{in}\right)=\frac{\sum_{i=1}^{k} T_{GPP_i}^{EXE}\left(P_{in_i}\right)-T_{GPP}^{par}}{\sum_{i=1}^{r}\left(T_{GPP_i}\left(P_{GPP_i}\right)+T_{ASP_i}\left(P_{RCE_i}\right)-T_{GPP_i,\,ASP_i}^{par}+ttr_{GPP_i,\,ASP_i}\right)+\sum_{q=1}^{k-r} T_{GPP_q}\left(P_{in_q}\right)-T'^{par}_{GPP}}, \quad (11)$$

де T_{GPP}^{par} – час, протягом якого дві або більше підпрограм (потоків) P_{in_i} виконуються процесорами GPP_i паралельно; T'^{par}_{GPP} – час, протягом якого дві або більше підпрограм (потоків) P_{GPP_i} виконуються відповідними процесорами паралельно.

Висновки

Стаття присвячена актуальному завданню розроблення та дослідження основ організації багатопроцесорних самоконфігуровних комп'ютерних систем. У ній вперше на рівні концепції висвітлено загальні принципи побудови і організації функціонування такої комп'ютерної системи, а також показано її часові характеристики.

Розроблено спосіб опрацювання інформації в багатопроцесорній СККС. Визначено перелік і описано порядок застосування необхідних для його реалізації комп'ютерних засобів. Показано принципи структурної організації багатопроцесорної СККС.

Подано вирази для розрахунку часових характеристик опрацювання інформації в багатопроцесорній СККС. Проаналізовано тривалість опрацювання інформації на всіх трьох етапах компіляції програми, її завантаження та виконання. Запропоновано методику визначення тривалості виконання програми у випадку її одноразового та повторного виконання.

Визначено необхідні для досягнення високої продуктивності багатопроцесорної СККС умови та проаналізовано способи забезпечення їх виконання. Запропоновано вираз для визначення прискорення, забезпечуваного багатопроцесорною СККС.

1. 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. 2. В. Мельник, В. Степанов, З. Сарайрех. Система розподілу обчислювального навантаження між хост-комп'ютером та самоконфігуровним прискорювачем // Науковий вісник Чернівецького університету. Комп'ютерні системи та компоненти. – Чернівці: Чернівецький національний університет імені Юрія Федьковича. – 2012. – Т. 3. – Вип. 1. – С. 6–16. 3. A Proven EDA Solutions Provider makes all the difference. [Online]. Available: http://www.aldec.com/en. 4. Xilinx Core Generator. Xilinx Inc. [Online]. Available: http://www.xilinx.com/ise/products/coregen_overview.pdf – 2005. 5. Мельник А., Мельник В. Організація бібліотек ядер стандартизованих та замовних комп'ютерних пристроїв для високопродуктивних реконфігуровних прискорювачів // IV Всеукраїнська науково-практична конференція "Комп'ютерні технології: наука і освіта", Україна, м.Луцьк, 9-11 жовтня 2009 р., Луцький інститут розвитку людини Університету "Україна", с.113-117. 6. Genest, G. "Programming an FPGA-based Super Computer Using a C-to-VHDL Compiler: DIME-C", Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference, 5-8 Aug. 2007. - P. 280 – 286. 7. Chameleon – the System-Level Design Solution. [Online]. Available: http://intron-innovations.com/?p=sld_chame. 8. ANSI-C to VHDL Compiler. [Online]. Available: http://www.nallatech.com/FPGA-Development-Tools/dimetalk.html. 9. Мельник А. О. Персональні суперкомп'ютери: архітектура, проектування, застосування: монографія / А. О. Мельник, В. А. Мельник. – Львів: Видавництво Львівської політехніки, 2013. – 516 с. 10. Мельник В. А. Розширений формат виконавчого файлу для самоконфігуровних комп'ютерних систем / В. А. Мельник, А. Ю. Кіт // Праці V Всеукраїнської науково-практичної

конференції "Проблеми інформатики та комп'ютерної техніки" (ПІКТ-2016). – Чернівці, 2016. – С. 186–187. 11. Par4All tool, http://www.par4all.org/ 12. Pluto tool, http://pluto-compiler.sourceforge.net/ 13. Anna Beletska, Wlodzimierz Bielecki, Albert Cohen, Marek Palkowski, Krzysztof Siedlecki. „Coarse-grained loop parallelization: Iteration Space Slicing vs affine transformations". Parallel Computing, Volume 37, Issue 8, August 2011, Pages 479–497. 14. Chirag Dave, Hansang Bae, Seung-Jai Min, Seyong Lee, Rudolf Eigenmann, and Samuel Midkiff, "Cetus: A Source-to-Source Compiler Infrastructure for Multicores". Computer, vol 42, no 12, pp. 36–42, Dec 2009. 15. P. Randive A. Athavale and A. Kambale. Automatic parallelization of sequential codes using s2p tool and benchmarking of the generated parallel codes. http://www.kpitcummins.com/downloads/research-papers/automatic-parallelization-sequential-codes.pdf. 16. Relogix Assembler-to-C translator. [Online]. Available: http://www.microapl.co.uk/asm2c/. 17. Мельник А. О. Хамелеон – система високорівневого синтезу спеціалізованих процесорів / А. О. Мельник, А. М. Сало, В. Клименко, Л. Цигилик, А. Юрчук // Науково-тухнічний журнал Національного аерокосмічного університету ім. М.Є. Жуковського "Харківський авіаційний інститут". – Харків. – 2009. – №5. – С. 189–195. 18. Agility Compiler for SystemC. Electronic System Level Behavioral Design & Synthesis Datasheet. 2005. [Online]. Available: http://www.europractice.rl.ac.uk/vendors/agility_compiler.pdf. 19. Handel-C Language Reference Manual For DK Version 4. Celoxica Limited, 2005. – 348 p. 20. C-to-FPGA Tools form Impulse Accelerated Technologies. Impulse CoDeveloper C-to-FPGA Tools. [Online]. Available: http://www.impulseaccelerated.com/products_universal.htm 21. Foundation Series ISE 3.1i User Guide" (PDF). www.xilinx.com. 22. Clive Maxfield, EE Times. "WebPACK edition of Xilinx Vivado Design Suite now available." Dec 20, 2012. 23. Clive Maxfield, "Latest and greatest Quartus II design software from Altera", EETimes, November 7, 2011. 24. Clive Maxfield, "Altera's Quartus II design software features Qsys System Integration Tool", EETimes, May 9, 2011. 25. V. Melnyk. Self-Configurable FPGA-Based Computer Systems: Basics and Proof of Concept. Scientific-Technical Journal "Advances in Cyber-Physical Systems". Vol. 1, No. 1, 2016. – pp. 37 – 47.