УДК 004.383, 004. 386

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

КОНЦЕПТУАЛЬНІ ОСНОВИ ПОБУДОВИ СКЛАДОВИХ ЧАСТИН САМОКОНФІГУРОВНОЇ КОМП’ЮТЕРНОЇ СИСТЕМИ

Розроблено концептуальні основи побудови базових компонентів самоконфігуровної комп’ютерної системи – програмних засобів розподілу обчислювального навантаження між універсальним комп’ютером та реконфігуровним середовищем і програмних засобів генерування програмних моделей спеціалізованих процесорів, а також сформовано вимоги до інших її компонентів – реконфігуровного середовища, його драйвера і програмних засобів його конфігурування і логічного синтезу спеціалізованих процесорів.

Ключові слова: самоконфігуровна комп’ютерна система, система розподілу обчислювального навантаження, система генерування програмних моделей спеціалізованих процесорів.

In a paper a conceptual basis for development of basic components of the self-configurable computer system – software for computational load balancing between the general-purpose computer and reconfigurable environment and software for application-specific processors program models generation, is developed. Requirements to the other components of that system – reconfigurable environment, its driver and software for its configuring and application-specific processors logic synthesis, are formulated.

Key words: self-configurable computer system, software for computational load balancing, software for application-specific processors program models generation.

Вступ

Досягнутий рівень як засобів проектування, так і прискорювачів на основі мікросхем програмованих логічних інтегральних середовищ (ПЛІС), дає змогу зробити висновок про потребу створення нового класу високопродуктивних комп’ютерних засобів на основі пристроїв реконфігуровної логіки – самоконфігуровних комп’ютерних систем, які будуть здатні повною мірою використовувати властивість їх реконфігуровності та автоматично здійснювати апаратне налаштування своєї внутрішньої структури під заданий алгоритм з метою найшвидшого його виконання. Для такого розвитку вже склалися необхідні передумови, адже сьогодні існує значна кількість програмних засобів для генерування програмних моделей спеціалізованих процесорів, типів ПЛІС та друкованих плат реконфігуровних прискорювачів, високошвидкісних інтерфейсів, за допомогою яких під’єднують реконфігуровні прискорювачі до універсальних комп’ютерів та інтегрованих середовищ автоматизованого проектування електронних пристроїв у ПЛІС.

Розробка самоконфігуровних високопродуктивних комп’ютерних систем на основі пристроїв реконфігуровної логіки з відповідним вдосконаленням їх інфраструктури є природним процесом розвитку моделей, методів і засобів проектування комп’ютерних систем та їх компонентів у напрямку розпаралелювання та оперативності виконання обчислень. Тому цілком виправданою є необхідність розроблення принципів побудови та організації функціонування самоконфігуровних комп’ютерних систем, їх реалізація, а також дослідження їх характеристик та оцінювання ефективності їх застосування.

Аналіз досліджень та публікацій

В [1] запропоновано концепцію побудови самоконфігуровних прискорювачів, а в [2] – метод їх самоконфігурування, згідно з якими усі кроки з налаштування до виконання обчислювальних завдань у комп'ютерній системі з реконфігуровною логікою виконуються цією комп'ютерною системою автоматично.

Запропоновану в [1] концепцію взято за основу під час побудови самоконфігуровної комп'ютерної системи (СККС) – 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. Проблема інертності, притаманна реконфігуровним комп'ютерним системам (РККС) і викликана витратами часу на їх конфігурування, в СККС знімається завдяки зміні в ній способу опрацювання інформації. Концепцію побудови СККС, спосіб опрацювання в ній інформації та її структуру запропоновано в [3].

Як зазначено вище, сьогодні існують програмні засоби, які можна використати для побудови СККС. Зокрема, в [4] запропоновано підхід до побудови системи автоматичного розподілу обчислювального навантаження між універсальним комп'ютером та прискорювачем. На ринку наявні засоби автоматичної генерації програмних моделей [5, 6], бібліотеки програмних моделей [7], а також системи високорівневого проектування програмних моделей спеціалізованих процесорів на основі опису алгоритмів їх роботи мовою високого рівня [8–10].

Постановка завдання

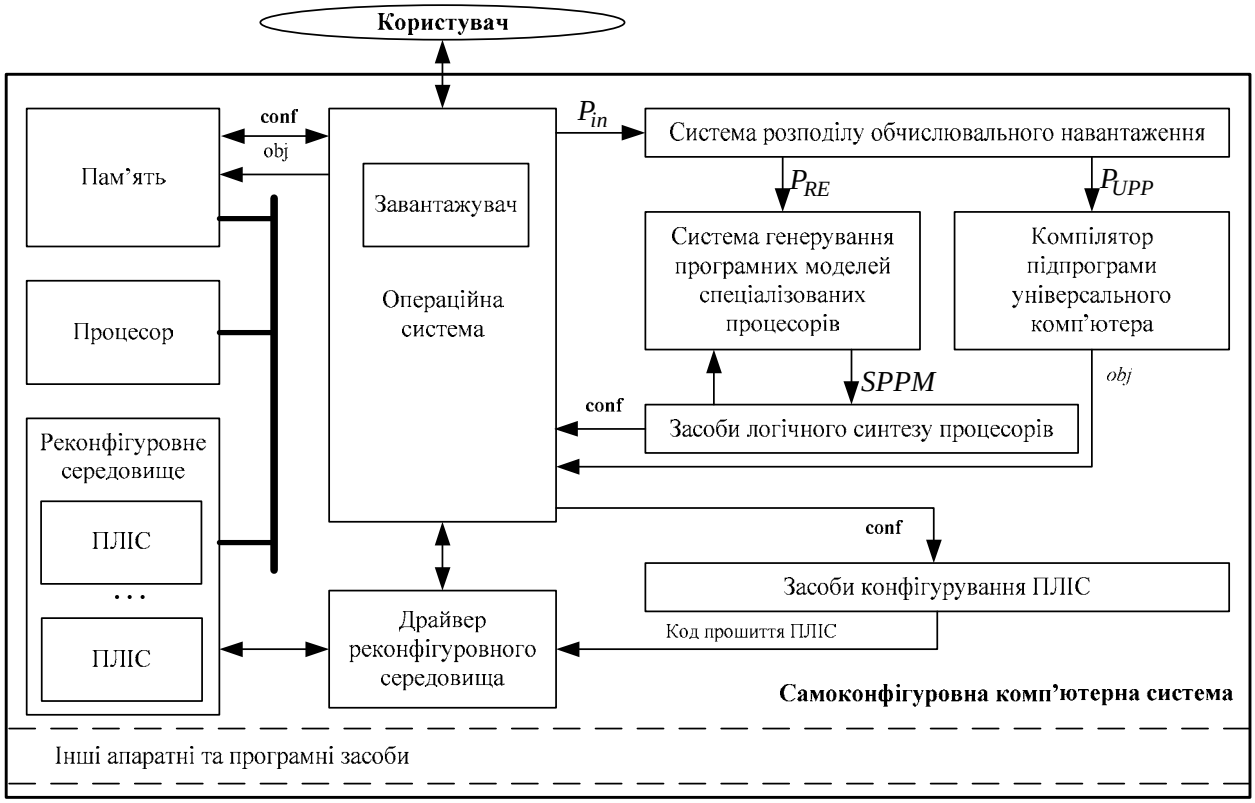
З метою створення основ проектування та реалізації складових частин СККС і організації ефективного функціонування її програмних та апаратних засобів, а також ефективного використання ресурсів універсального комп'ютера та забезпечення високих показників продуктивності постає завдання розроблення концептуальних основ побудови та формування вимог до складових частин СККС, що і є предметом наукового дослідження, висвітленого у цій роботі.

1. Складові частини самоконфігуровної комп'ютерної системи

Структуру самоконфігуровної комп'ютерної системи, яка реалізує запропонований у [3] спосіб опрацювання інформації, показано на рисунку.

До складу СККС належать такі програмні засоби:

- система розподілу обчислювального навантаження між універсальним комп'ютером та реконфігуровним середовищем, яка автоматично знаходить у програмі P_{in} такі фрагменти, виконання яких в реконфігуровному середовищі прискорює роботу комп'ютерної системи, і розподіляти програму P_{in} на підпрограму P_{UPP} універсального комп'ютера, заміняючи в ній виокремлені фрагменти на команди роботи з реконфігуровним середовищем, та сформовану із цих фрагментів підпрограму P_{RE} реконфігуровного середовища. Приклад такої системи висвітлено в [4];
- система генерування, яка автоматично генерує програмну модель $SPPM$ спеціалізованого процесора з підпрограми P_{RE}, наприклад, Хамелеон від компанії Інтрон [9], *Agility Compiler* [11] та *DK4 Design Suite* [12] – від компанії *Celoxica*, *CoDeveloper* [13] – від компанії *Impulse Accelerated Technologies*;
- компілятор з мови, на якій подається підпрограма універсального комп'ютера, в об'єктний код, що безпосередньо може ним виконуватись, для компіляції підпрограми універсального комп'ютера і формування її виконавчого файла;
- програмні засоби виконання логічного синтезу програмних моделей спеціалізованих процесорів та конфігурування реконфігуровного середовища, наприклад: *ISE*, *Alliance*, *Foundation* від компанії *Xilinx*, *Quartus* II, *Max*+II – від компанії *Altera* (цим компаніям належить левова частка світового ринку ПЛІС).



Структура самоконфігуровної комп'ютерної системи

Проведемо дослідження і розробимо концептуальні основи побудови базових компонентів СККС – системи розподілу обчислювального навантаження між універсальним комп'ютером та реконфігуровним середовищем і системи генерування програмних моделей спеціалізованих процесорів, а також сформуємо вимоги до інших компонентів СККС – засобів логічного синтезу спеціалізованих процесорів та конфігурування ПЛІС, реконфігуровного середовища і його драйвера.

2. Концептуальні основи побудови системи розподілу обчислювального навантаження між комп'ютером і реконфігуровним середовищем

Система розподілу обчислювального навантаження виконує функцію розподілу заданого програмою обчислювального завдання між комп'ютером та реконфігуровним середовищем. Її можна описати так:

$$\{P_{UPP}, P_{RE}\} = CLBS\,(P_{in}), \tag{1}$$

де $CLBS$ – система розподілу обчислювального навантаження (англ. *Computational Load Balancing System*).

Ефективність $CLBE$ розподілу обчислювального навантаження є найвищою за такого варіанта розподілу програми P_{in}, за якого час $T_{SCCS}^{EXE}(P_{in})$ виконання цієї програми в СККС є найменшим порівняно з часом її виконання за усіх інших варіантів розподілу, тобто

$$CLBE_{max} = \frac{\max\limits_{r}\left(T_{SCCS}^{EXE}(P_{in})\right)}{\min\limits_{r}\left(T_{SCCS}^{EXE}(P_{in})\right)}, r = \overline{1...R}, \tag{2}$$

де R – кількість варіантів розподілу програми P_{in}.

Для того, щоб забезпечити найвищу ефективність розподілу, система розподілу обчислювального навантаження повинна:

1) виділяти з програми P_{in} такі фрагменти, які становлять найбільше обчислювальне навантаження (характеризуються найвищою обчислювальною складністю) і виконання яких вимагає найбільшої кількості ресурсів універсального процесора, і з цих фрагментів формувати підпрограму P_{RE};

2) розподіляти програму P_{in} на підпрограми P_{UPP} і P_{RE} так, щоб кількість даних, що пересилаються між комп'ютером та реконфігуровним середовищем, а також кількість сеансів взаємодії комп'ютера з реконфігуровним середовищем в процесі виконання цієї програми, були якомога меншими;

3) розподіляти програму P_{in} на підпрограми P_{UPP} і P_{RE} так, щоб вони могли виконуватися паралельно.

Як показали дослідження [4], виконання зазначених умов залежить від ефективності вибору типу фрагмента програмного коду, над яким виконуються оптимізаційні процедури під час здійснення розподілу обчислювального навантаження. Загалом, що менше часу витрачається на обмін даними з фрагментом програмного коду по відношенню до часу його виконання за збереження частки виконуваних універсальним комп'ютером та реконфігуровним середовищем об'ємів обчислень в програмі, то ефективнішим є вибір типу фрагмента програмного коду за розподілу обчислювального навантаження. Якщо позначити множину типів фрагментів програмного коду як $\mathbf{FC}$, причому $\mathbf{FC} = \left\{FC_i, FC_j; i = \overline{1...n}; j = \overline{1...n}\right\}$, де FC_i, FC_j – i-й та j-й типи фрагмента програмного коду, серед яких виконується вибір, n – кількість типів фрагментів програмного коду, то вибір типу фрагмента програмного коду FC_i за розподілу обчислювального навантаження є ефективним, якщо для $FC_i \in \{FC\}, FC_j \in \{FC\}$, то виконуються співвідношення:

$$\begin{cases} \dfrac{\sum\limits_{n} T_{DT}\left(FC_i\right)_k}{\sum\limits_{n} T_{FCE}\left(FC_i\right)_k} < \dfrac{\sum\limits_{m} T_{DT}\left(FC_j\right)_l}{\sum\limits_{m} T_{FCE}\left(FC_j\right)_l}, \\ Q_{RE}\left(FC_i\right) = Q_{RE}\left(FC_j\right) \end{cases} \tag{3}$$

де $T_{DT}\left(FC_i\right)_k$ – час обміну даними з k-м, $k = \overline{1...n}$ фрагментом програмного коду за i-му типу фрагмента програмного коду; $T_{FCE}\left(FC_i\right)_k$ – час виконання k-го фрагмента програмного коду за i-го типу фрагмента програмного коду; $T_{DT}\left(FC_j\right)_l$ – час обміну даними з l-м, $l = \overline{1...m}$ фрагментом програмного коду за j-го типу фрагмента програмного коду; $T_{FCE}\left(FC_j\right)_l$ – час виконання l-го фрагмента програмного коду за j-го типу фрагмента програмного коду; n – кількість фрагментів програмного коду в об'ємі обчислень, що припадають на підпрограму реконфігуровного середовища, за i-го типу фрагмента програмного коду; m – кількість фрагментів програмного коду в об'ємі обчислень, що припадають на підпрограму реконфігуровного середовища за j-го типу фрагмента програмного коду; Q_{RE} – частка виконуваного реконфігуровним середовищем об'єму обчислень в програмі P_{in}, яка визначається з такого виразу:

$$Q_{RE} = \frac{V_{RE}}{V_{UPP} + V_{RE}} \cdot 100\ \%, \tag{4}$$

де V_{RE} – об'єм обчислень, що припадають на реконфігуровне середовище; V_{UPP} – об'єм обчислень, що припадають на універсальний процесор.

3. Концептуальні основи побудови системи генерування програмних моделей спеціалізованих процесорів

Концептуальні основи побудови системи генерування сформовано та детально досліджено у [14], тому зупинимось лише на основних висновках цього дослідження.

Функцією системи генерування програмних моделей процесорів в СККС є автоматична генерація програмної моделі процесора для виконання обчислювальних алгоритмів, отриманих від системи розподілу обчислювального навантаження. Функцію системи генерування можна описати так:

$$PMSP\left(A_i, i=\overline{1...z}\right)=GS\left(P_{RE}\left(A_i, i=\overline{1...z}\right)\right), \tag{5}$$

де GS – система генерування; P_{RE} – програма реконфігуровного середовища, яка описує обчислювальні алгоритми $A_i, i=\overline{1...z}$; $PMSP$ – програмна модель спеціалізованого процесора для виконання обчислювальних алгоритмів $A_i, i=\overline{1...z}$; z – кількість обчислювальних алгоритмів.

Відповідно до цього завдання, система генерування повинна відповідати вимогам, які можна поділити на такі три групи [14]:

1. Вимоги в частині її функціональної повноти:

система генерування повинна бути функціонально повною, тобто такою, що для довільного обчислювального алгоритму може згенерувати модель спеціалізованого процесора.

2. Вимоги в частині технічних характеристик створюваних системою генерування моделей процесорів:

a) система генерування повинна генерувати такі моделі процесорів, які забезпечують максимально можливу продуктивність, яка може бути досягнута на виділеному для їх реалізації обладнанні ПЛІС;

b) система генерування повинна генерувати такі моделі процесорів, які займатимуть максимум з виділеного в ПЛІС прискорювача місця під їх реалізацію з тим, щоб найефективніше використати обладнання. Наприклад, якщо під реалізацію процесора виділено увесь кристал ПЛІС, то його схема повинна займати увесь кристал.

3. Вимоги в частині архітектури створюваних системою генерування моделей процесорів:

a) система генерування повинна генерувати такі моделі процесорів, архітектура яких враховує особливості архітектури ПЛІС реконфігуровного середовища;

b) система генерування повинна генерувати такі моделі процесорів, архітектура яких враховує можливості інтерфейсу між універсальним комп'ютером та реконфігуровним середовищем.

Продуктивність спеціалізованого процесора є найвищою за такого варіанта його реалізації, за якого значення відношення часу $T_{SP}\left(P_{RE}\right)$ виконання ним заданих підпрограмою P_{RE} обчислювальних алгоритмів до часу $T_{UPP}\left(P_{RE}\right)$ виконання підпрограми P_{RE} універсальним процесором UPP є найменшим порівняно зі значенням такого відношення за усіх інших варіантів його реалізації, тобто

$$P_{SP}^{max}=\frac{1}{min\left(\frac{T_{SP_j}\left(P_{RE}\right)}{T_{UPP}\left(P_{RE}\right)}\right)}=max\left(\frac{T_{UPP}\left(P_{RE}\right)}{T_{SP_j}\left(P_{RE}\right)}\right),\ j=\overline{1...m}, \tag{6}$$

де m – кількість варіантів реалізації спеціалізованого процесора.

Продуктивність спеціалізованого процесора, відповідно до [15], розраховується так.

Нехай у процесор поступають N елементів вхідних даних і у ному має бути реалізований алгоритм з обчислювальною складністю $R=R(N)$, тобто R – це кількість операцій, які потрібно виконати для реалізації алгоритму. Тоді необхідна для забезпечення виконання алгоритму за заданий час $T_{SP}\left(P_{RE}\right)$ продуктивність процесора визначається з такого виразу:

$$P_{SP}=R/N.$$

Для кожного алгоритму, який реалізується в процесорі, наперед відомими є кількість N вхідних даних та обчислювальна складність R алгоритму. Тому для кожного алгоритму можна наперед визначити необхідну продуктивність спеціалізованого процесора під час його реалізації в реконфігуровному середовищі з тим, щоб він забезпечував обробку даних за заданий час.

Постає питання, чи ця продуктивність може бути досягнута, та як можна забезпечити вищеназвані вимоги. Для того, щоб виконати це завдання, виходитимемо з такого. Нехай кількість обладнання реконфігуровного середовища дорівнює Q_{RE}, а кількість необхідного для реалізації спеціалізованого процесора обладнання дорівнює Q_{SP}. Можливі чотири випадки у співвідношенні між цими величинами. Якщо Q_{SP} незначно менше або дорівнює Q_{RE}, то мета досягнута. Якщо ж Q_{SP} незначно більше Q_{RE}, то потрібно дещо зменшити вимоги до продуктивності спеціалізованого процесора, що нескладно зробити, оскільки сьогоднішні засоби генерування програмних моделей спеціалізованих процесорів дають змогу задавати на етапі генерування значення продуктивності або обмеження на затрати обладнання. Якщо виконується нерівність $Q_{RE} < Q_{SP}$, причому співвідношення між затратами обладнання більше від одиниці, то виникає потреба в скороченні затрат обладнання на спеціалізований процесор, що досягається спрощенням його структури та відповідно зниженням продуктивності. Якщо ж виконується нерівність $Q_{RE} > Q_{SP}$, причому співвідношення між затратами обладнання більше від одиниці, то для ефективнішого використання ресурсів реконфігуровного середовища доцільно включити паралельно $m = int\left(Q_{SP} \middle/ Q_{RE}\right)$ спеціалізованих процесорів.

4. Вимоги до засобів логічного синтезу процесорів та конфігурування ПЛІС

Завданням засобів логічного синтезу процесорів та конфігурування ПЛІС реконфігуровного середовища є виконання логічного синтезу згенерованої системою генерування програмної моделі спеціалізованого процесора і створення файла конфігурації ПЛІС на етапі компіляції програми P_{in}, а також завантаження цієї конфігурації до ПЛІС реконфігуровного середовища на етапі її виконання.

Хоча робота засобів логічного синтезу процесорів та конфігурування ПЛІС розподілена по двох етапах, значна частина виробників постачають вказані засоби в єдиному інтегрованому середовищі, тому доцільно формувати вимоги до цих засобів як до комплексної системи.

Відповідно до поставлених завдань, можна виділити такі вимоги до засобів логічного синтезу процесорів та конфігурування ПЛІС:

1. Підтримка того типу, серії і типу корпуса ПЛІС, на основі яких побудовано реконфігуровне середовище СККС. На практиці це означає, що виробником засобів логічного синтезу та конфігурування повинна бути та сама компанія, яка виготовляє ПЛІС, і особливості цієї ПЛІС повинні бути відображені в бібліотеці цих засобів. З метою економії пам'яті для зберігання програмних засобів СККС та для прискорення функціонування бібліотеку засобів логічного синтезу та конфігурування доцільно обмежити типом, серією і типом корпусу ПЛІС, на основі яких побудовано реконфігуровне середовище.

2. Виконання логічного синтезу спеціалізованого процесора за прийнятний для компіляції програми час необхідно забезпечити відповідними налаштуваннями засобів логічного синтезу та їх оптимізацією.

3. Мінімальний час виконання реконфігурування ПЛІС. Ця вимога є особливо важливою, оскільки тривалість конфігурування ПЛІС є складовою частиною часу завантаження програми до виконання в СККС.

5. Вимоги до драйвера реконфігуровного середовища

Завданнями драйвера реконфігуровного середовища є надання доступу до нього програмними засобами СККС для виконання ініціалізації, конфігурування, завантаження до нього вхідних даних для виконання обчислювальних завдань та отримання від нього результатів. Відповідно до цих завдань, можна виділити такі вимоги до драйвера реконфігуровного середовища:

1. Виконання початкової ініціалізації реконфігуровного середовища.
2. Виконання динамічного реконфігурування реконфігуровного середовища на запит операційної системи.
3. Передача даних до спеціалізованого процесора в реконфігуровному середовищі.
4. Отримання даних від спеціалізованого процесора в реконфігуровному середовищі і надсилання відповідного інформаційного повідомлення операційній системі.

6. Вимоги до реконфігуровного середовища

Завданнями реконфігуровного середовища є налаштування ПЛІС відповідно до отриманої від засобів конфігурування ПЛІС-конфігурації, в такий спосіб утворюючи у ній синтезований засобами логічного синтезу спеціалізований процесор, сприйняття від комп'ютера вхідних даних для опрацювання у цьому процесорі, та повернення результатів роботи до комп'ютера. Відповідно до цих завдань, можна виділити такі вимоги до реконфігуровного середовища:

1. Універсальність, тобто здатність налаштовувати внутрішню структуру до виконання довільного обчислювального алгоритму. Зазначимо, що сучасні ПЛІС повністю відповідають цій вимозі, хоча продуктивність виконання тих чи інших алгоритмів є різною в ПЛІС різних виробників внаслідок розмаїття структури логічних комірок.
2. Мінімальний час завантаження кодів конфігурації до ПЛІС.
3. Наявність інтерфейсної частини, яка на апаратному рівні виконуватиме взаємодію між реалізованим в реконфігуровному середовищі спеціалізованим процесором та комп’ютером.

Висновки

Розроблено концептуальні основи побудови складових частин СККС – системи розподілу обчислювального навантаження та системи генерування програмних моделей спеціалізованих процесорів. Встановлено, що для того, щоб забезпечити найвищу ефективність розподілу, система розподілу обчислювального навантаження повинна формувати підпрограму реконфігуровного середовища з таких фрагментів, які становлять найбільше обчислювальне навантаження, і розподіляти навантаження так, щоб кількість даних, що пересилаються між комп'ютером та реконфігуровним середовищем, а також кількість сеансів взаємодії комп’ютера з реконфігуровним середовищем в процесі виконання цієї програми були якомога меншими, а відповідні підпрограми могли виконуватись паралельно. Показано, що система генерування програмних моделей спеціалізованих процесорів повинна відповідати трьом групам вимог: в частині її функціональної повноти, в частині технічних характеристик, створюваних нею програмних моделей спеціалізованих процесорів, та в частині архітектури цих процесорів. Також показано, як розраховується продуктивність реалізованого в реконфігуровному середовищі СККС спеціалізованого процесора. Сформовано вимоги до інших складових частин СККС – засобів логічного синтезу спеціалізованих процесорів та конфігурування ПЛІС, реконфігуровного середовища та його драйвера. Розроблені концептуальні основи побудови та вимоги є основою проектування та реалізації складових частин СККС.

1. Мельник В.А. Самоконфігуровні апаратні прискорювачі обчислень в комп’ютерах / В.А. Мельник, З.Т. Сарайрех // Вісник Національного університету „Львівська політехніка” “Комп’ютерні системи та мережі”. – 2010. – №688. – С.163–171. 2. Мельник В.А. Метод самоконфігурування апаратного прискорювача / В.А. Мельник, З.Т. Сарайрех // Матер. V Міжнар.

конф. молодих вчених "Комп'ютерні науки та інженерія 2011" (CSE-2011), Україна, Львів, 2011. – С. 126–127. 3. Melnyk A., Melnyk V. "Self-Configurable FPGA-Based Computer Systems," Advances in Electrical and Computer Engineering. – Vol. 13, № 2. – P. 33-38, 2013, doi: 10.4316/AECE.2013.02005. 4. Мельник В. Система розподілу обчислювального навантаження між хост-комп'ютером та самоконфігуровним прискорювачем / В. Мельник, В. Степанов, З. Сарайрех // Науковий вісник Чернівецького університету "Комп'ютерні системи та компоненти". – Чернівці: Чернівецький національний університет імені Юрія Федьковича, 2012. – Т. 3, Вип. 1. – С.6–16. 5. A Proven EDA Solutions Provider makes all the difference. – [Електронний ресурс]. – Режим доступу: http://www.aldec.com/en. 6. Xilinx Core Generator. Xilinx Inc. – [Електронний ресурс]. – Режим доступу: http://www.xilinx.com/ise/products/coregen_overview.pdf – 2005. 7. Мельник А. Організація бібліотек ядер стандартизованих та замовних комп'ютерних пристроїв для високопродуктивних реконфігуровних прискорювачів / А. Мельник, В. Мельник // IV Всеукр. наук.-практ. конф. «Комп'ютерні технології: наука і освіта», Україна, м. Луцьк, 9–11 жовтня 2009 р.. – Луцьк: Луцький інститут розвитку людини університету «Україна». – С. 113–117. 8. Genest G. Programming an FPGA-based Super Computer Using a C-to-VHDL Compiler: DIME-C / G. Genest, R. Chamberlain, R. Bruce // Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference, 5–8 Aug. 2007. – P. 280–286. 9. Chameleon – the System-Level Design Solution. – [Електронний ресурс]. – Режим доступу: http://intron-innovations.com/?p=sld_chame. 10. ANSI-C to VHDL Compiler. – [Електронний ресурс]. – Режим доступу: http://www.nallatech.com/FPGA-Development-Tools/dimetalk.html. 11. Agility Compiler for SystemC. Electronic System Level Behavioral Design & Synthesis Datasheet. – 2005. – [Електронний ресурс]. – Режим доступу: http://www.europractice.rl.ac.uk/vendors/agility_compiler.pdf. 12. Handel-C Synthesis Methodology – Mentor Graphics. – [Електронний ресурс]. – Режим доступу: http://www.mentor.com/products/fpga/handel-c/. 13. C-to-FPGA Tools form Impulse Accelerated Technologies. Impulse CoDeveloper C-to-FPGA Tools. – [Електронний ресурс]. – Режим доступу: http://www.impulseaccelerated.com/products_universal.htm. 14. Мельник В.А. Вимоги до системи генерування моделей процесорів самоконфігуровного апаратного прискорювача / В.А. Мельник, З.Т. Сарайрех // Матер. 5-ї Міжнар. наук.-техн. конф. «Сучасні комп'ютерні системи та мережі: розробка та використання» (ASCN-2011). – Львів, 2011. – С.255–258. 15. Мельник А.О. Спеціалізовані комп'ютерні системи реального часу / А.О. Мельник. – Львів: Державний університет "Львівська політехніка", 1996. – 54 с.