отражения Фонга. Модель отражения Блинна–Фонга (www.compgraphics.info). 5. Программирование шейдеров на HLSL. Модели освещения. (www.gamedev.ru) 6. Кубическая текстура. (www.ru.wikipedia.org). 7. Blinn K. Models of Light Reflection for Computer Sythesized Pictures, SIGGRAPH 77, Computer Graphics, v11, n4, cm273-536, 1977. 8. HDRCubeMap Direct3D Sample (www.msdn.microsoft.com).

УДК 004.272.2, 004.272.23

А.О. Мельник, А.М. Сало, В.А. Клименко, Л.О. Цигилик, А.В. Юрчук
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

РЕАЛІЗАЦІЯ ПРОГРАМНИХ СПЕЦІАЛІЗОВАНИХ ПРОЦЕСОРІВ У РЕКОНФІГУРОВНИХ ПРИСКОРЮВАЧАХ УНІВЕРСАЛЬНИХ КОМП'ЮТЕРІВ

Розглянуто історію появи прискорювачів універсальних комп'ютерів. Подано вимоги до реконфігурованого прискорювача щодо приймання, опрацювання та передавання даних. Представлено модель спеціалізованого процесора та показано методи оптимізації архітектури цієї моделі під об'єм ПЛІС. Визначено основні вимоги щодо оптимізації архітектури апаратних ресурсів ПЛІС. Розглянуто етапи проектування прискорювачів універсальних комп'ютерів за допомогою САПР ХАМЕЛЕОН.

In this article the requirements to the reconfigurable accelerator in relation to data reception, processing and data transfer are given. The model of the specialized processor is presented and the architecture optimization methods of this model in accordance to the FPGA capacity are described. The basic requirements concerning architecture optimization of FPGA hardware resources are defined.

Вступ. Потреба вирішення ресурсомістких задач та задач з інтенсивним використанням даних, таких як обробка мультимедійних даних в реальному часі, сучасне математичне моделювання і обробка вмісту спричинили формування нового типу програмних та апаратних засобів, які отримали назву реконфігуровних прискорювачів. Реконфігуровні прискорювачі все частіше використовуються для виконання складних задач обробки даних з метою зменшення навантаження на універсальні процесори та підвищення продуктивності комп'ютерних систем. При цьому можливість реконфігурування, тобто заміни вмісту реконфігуровного прискорювача новим спеціалізованим процесором, відкриває перспективи надання принципово нових властивостей та досягнення високих технічних характеристик комп'ютерної системи, до складу якої входить реконфігуровний прискорювач [6].

1. Поява прискорювачів універсальних комп'ютерів. Використання в комп'ютерних системах прискорювачів не є новим підходом. Перші прискорювачі універсальних комп'ютерів з'явились в середині 60-х років минулого століття, коли було набуто певного досвіду використання комп'ютерів у науковій та виробничій сферах, покращилась їх архітектура, але швидкодія була недостатньою для вирішення складних наукових проблем, насамперед завдань цифрової обробки сигналів. Для таких потреб почали створювати спеціалізовані процесори, які апаратно реалізували найскладніші алгоритми і під'єднувались до універсальних комп'ютерів як прискорювачі при виконанні цих алгоритмів. Зокрема, такі спеціалізовані процесори створювались для виконання

алгоритмів згортки під час обробки даних сейсморозвідки. З часом спеціалізовані процесори стали входити до складу більшості серій універсальних комп'ютерів. Наприклад, до складу серій IBM 360 і IBM 370 входили спеціалізовані процесори AP2938 та AP3838. При цьому універсальний комп'ютер здійснював загальний контроль над обчислювальним процесом та операціями введення–виведення.

Створення потужних реконфігуровних середовищ відновило інтерес до даного напряму. Можна впевнено стверджувати, що створення програмних апаратно-орієнтованих спеціалізованих процесорів сьогодні належить до найновіших та найважливіших напрямів розвитку високопродуктивних комп'ютерних систем [4]. У зв'язку з цим виникає потреба розроблення методів проектування та структурної організації програмних апаратно-орієнтованих спеціалізованих процесорів, методів їх інтеграції в систему, а також визначення вимог до них для ефективного синтезу в реконфігуровних прискорювачах [3].

2. Вимоги до реконфігуровного прискорювача в частині організації приймання даних, їх опрацювання та видавання. Продуктивність процесорного ядра прискорювача для виконання відповідного алгоритму можна обчислити так [1]:

$$P = K \cdot F \cdot \frac{R}{N} \qquad (2.1)$$

де P – продуктивність процесора; K – кількість вхідних каналів; F – частота процесора; R – складність алгоритму, що виконується на процесорі; N – кількість елементів вхідних даних.

Для кожного алгоритму, який реалізується в процесорному ядрі, наперед відомими є кількість N вхідних даних та обчислювальна складність R алгоритму. Значення K можне варіювати залежно від можливостей комп'ютера, до якого під'єднують прискорювач. Значення F визначається елементною базою, на якій будується прискорювач. Кількість каналів K вибирають відповідно до рівня продуктивності та апаратних витрат, яких треба досягти. При визначенні необхідного рівня продуктивності відповідно до апаратних затрат приймемо, що кількість обладнання реконфігуровного прискорювача дорівнює Q_{RA}, а кількість обладнання на реалізацію ядра прискорювача – Q_{SP} .

Можливі чотири випадки в співвідношенні між цими величинами. Якщо Q_{SP} незначно менше або дорівнює Q_{RA}, то мети досягнуто. Якщо ж Q_{SP} незначно більше Q_{RA}, то потрібно дещо зменшити вимоги до продуктивності ядра прискорювача. Якщо виконується нерівність $Q_{RA}<Q_{SP}$, причому співвідношення між затратами обладнання більше за одиницю, то виникає потреба в скороченні затрат обладнання на ядро прискорювача. Цього досягають спрощенням його структури та, відповідно, пониженням продуктивності. При цьому для того, щоб не простоювали ресурси комп'ютера, можна зменшити потоки інформації, скоротивши кількість вхідних каналів надходження даних на величину $m = Integer(Q_{RA}/ Q_{SP})$ аж до одного, а також за потреби понизити частоту надходження даних на спеціалізований процесор. Якщо ж виконується нерівність $Q_{RA}>Q_{SP}$, причому співвідношення між затратами обладнання більше за одиницю, то для ефективнішого використання ресурсів реконфігуровного прискорювача доцільно під'єднати паралельно $m = Integer(Q_{SP}/Q_{RA})$ ядер прискорювача.

Організація передачі даних між прискорювачем і персональним комп'ютером відіграє значну роль у продуктивності цілої системи. Можливі два варіанти побудови системи з прискорювачем:

1. Прискорювач взаємодіє з центральним процесором за допомогою периферійної шини, наприклад, шини PCI-Express.
2. Прискорювач взаємодіє з прискорювачем за допомогою системної шини.

Сьогодні існують готові платформи від лідерів виробництва універсальних процесорів фірм AMD та Intel для побудови системи з реконфігурованим прискорювачем. Такими платформами є AMD Torrenza від AMD та платформа Intel з технологією QuickAssist Accelerator Abstraction Layer (AAL). Ці платформи дають змогу під'єднати реконфігуровний прискорювач як до системної шини (у AMD – Hypertransport, у Intel – Front Side Bus), так і до периферійної шини PCI Express [9].

Використання шини PCI Express є більш універсальним рішенням – сьогодні вже багато персональних комп'ютерів оснащують шиною PCI Express. Для такого рішення канал взаємодії стає «вузьким місцем», адже частота периферійної шини значно менша від частоти системної шини.

При розміщенні прискорювача на периферійній шині для досягнення найбільшого рівня продуктивності було сформульовано такі вимоги до організації передавання даних.

1. Дані мають передаватися за допомогою каналів прямого доступу до пам'яті (Direct Memory Access – DMA). Використання DMA каналів розвантажує центральний процесор для виконання інших задач, що підвищує продуктивність системи.

2. Дані мають передаватися пакетами оптимального розміру. Розмір пакета залежить від швидкості каналу передачі даних, часу, витраченого на ініціалізацію однієї передачі, частоти шини даних та кількості вхідних даних. Оптимальний розмір пакета має забезпечити завантаження каналу передавання даних з мінімальними витратами на організацію передачі.

Необхідно вибрати розмір пакета такий, щоб:

$$t_{çàã} \to MIN$$

$t_{заг}$ – загальний час передавання даних.

За пакетного передавання даних загальний час становить

$$t_{çàã} = t_{ïàêåòà} \cdot N_{ïàêåò³â} = t_{ïàêåòà} \cdot \frac{V_{äàíèõ}}{V_{ïàêåòà}}$$

$t_{пакета}$ – час передачі пакета; $N_{пакетів}$ – кількість пакетів; $V_{даних}$ – обсяг переданих даних; $V_{пакета}$ – обсяг пакета.

Час передавання даних складається з часу ініціалізації та часу передавання. Отже, для пакета оптимального розміру має виконуватися така умова

$$\frac{t_i(V) + t(V)V_{äàíèõ}}{V_{ïàêåòà}} \to MIN \ , \qquad (2.2)$$

де t_i (V)– час ініціалізації пересилання при пересилці пакета обсягом V; t(V) – час пересилки пакета обсягом V.

Наприклад, для плати PLDA Xpress FX100 на ПЛІС Virtex 4 FX100 було досліджено оптимальний розмір пакета пересилки даних. У результаті отримано залежність швидкості передавання даних від розміру пакета (рис. 1). Розрядність шини даних становить 32 розряди. З результатів видно, що найоптимальнішим є розмір пакета в 1 Мб (рис. 1), при якому досягається максимальна швидкість передачі даних.

3. Швидкість каналу передачі даних вибирають відповідно до класу вирішуваних задач і частоти ядра конфігурованого прискорювача. Канал має забезпечити неперервний потік даних та результатів, щоб уникнути очікувань на отримання нових вхідних даних. Швидкість передавання даних має давати змогу повністю використати ресурси реконфігуровного прискорювача.

4. Вибір розрядності каналу передачі даних відповідно до вирішуваної задачі та архітектури конфігурованого прискорювача. Розрядність має забезпечити завантаження даних у всі функціональні частини конфігурованого прискорювача.

5. Організація передавання даних безпосередньо з локальної пам'яті комп'ютера у локальну пам'ять прискорювача. Це дасть змогу перенести обчислювальні витрати з організації передачі даних на контролер прямого доступу до пам'яті.

6. Опрацьовують дані під час пересилання (на фоні пересилання).

7. При виконанні однієї команди мають виконуватися два пересилання даних: пересилання до реконфігуровного прискорювача блоку вхідних даних та пересилання до центрального процесора блоку вихідних даних.

При розміщенні прискорювача на системній шині процесора не відбувається жодних простоїв системи, адже частота системної шини значно перевищує частоту сучасних ПЛІС. Необхідно забезпечити синхронізацію передачі даних за допомогою буферів.

На програмному рівні необхідно організувати бібліотеку функцій введення та виведення даних з прискорювача та конфігурування каналу передачі даних. Така бібліотека функцій дасть змогу користувачам прискорювачів організувати ефективний обмін даними.

Рис. 1. Графік росту продуктивності передачі даних шиною PCI-E

3. Вимоги до архітектури прискорювача. Оптимізація моделі спеціалізованого процесора під об'єм ПЛІС. Спеціалізований процесор призначений для реалізації обмеженого класу алгоритмів. Його внутрішня структура повинна містити тільки ті модулі та вузли системи, які безпосередньо використовуються для організації обчислень [6]. Модель спеціалізованого процесора можна описати так:

$$SC = \{I, O, MM, A, C\} \tag{3.1}$$

де SC – спеціалізований процесор;

$$I = \{i_p^r \mid r, p, (r, p) \in Z, (r, p) > 0\} \tag{3.2}$$

– множина портів введення i_p^r, де p – кількість портів введення, r – їх розрядність;

$$O = \{o_j^r \mid r, j, (r, j) \in Z, (r, j) > 0\} \tag{3.3}$$

– множина портів виведення o_j^r, де j – кількість портів виведення, r – їх розрядність;

$$MM = \{mm_e \mid e \in Z, e > 0\} \tag{3.4}$$

– множина комірок пам'яті mm_e, де e – об'єм пам'яті (кількість слів);

$$A = \{alu_{P(x,y)}^r \mid P(x, y) \in Z, P(x, y) > 0, x \in Z, y \in R\} \tag{3.5}$$

– множина арифметико-логічних пристроїв (АЛП) $alu_{P(x,y)}^r$, де $P(x, y)$ – кількість АЛП, x – максимальний рівень розпаралеленості алгоритму (кількість паралельних операцій), y – допустимий відсоток втрати продуктивності виконання алгоритму, r – розрядність;

$$C = \{c^{S(r,p,j,P(x,y))} \mid S(r, p, j, P(x, y)) \in R, S > 0\} \tag{3.6}$$

– керівний пристрій, де $S(r,p,j,P(x,y))$– складність керівного пристрою.

Ця модель спеціалізованого процесора (3.1) описує мінімальну конфігурацію системи, необхідну для організації обчислень. Змінюючи елементи кожної з множин в більший чи в менший бік, створюють новий спеціалізований процесор. Наприклад, змінивши один елемент r – розрядність, необхідно змінювати структуру в трьох модулях: множини $\{I, O, A\}$, що, своєю чергою збільшує не тільки апаратні затрати, а й складність обчислювальної системи $c^{S(r,p,j,P(x,y))} \uparrow$ (для $alu_{P(x,y)}^r$ – змінюється реалізація команди обчислення з рухомою комою).

Вимоги, які стоять перед розробниками спеціалізованих процесорів для реконфігурованих прискорювачів:

- *розробити високопродуктивну систему;*
- *оптимально використати апаратні ресурси.*

Під час розроблення спеціалізованого процесора як прискорювача універсальних комп'ютерів виникають такі фізичні обмеження: об'єм ПЛІС, інтерфейс зв'язку між комп'ютером та реконфігурованим прискорювачем тощо. Оскільки об'єм ПЛІС є обмежений, а задачі, які добре розпаралелюються та масштабуються, вимагають значних апаратних затрат, постає завдання: *оптимально підібрати кількість обчислювальних модулів (АЛП) для реалізації множини задач при наперед заданому об'ємі ПЛІС.*

Вирішенням цього завдання є розроблення способу, який би дав змогу розрахувати оптимальну кількість обчислювальних модулів для кожного окремого алгоритму, відповідно до об'єму кристала.

Сьогодні об'єм ПЛІС значно виріс, що дає можливість реалізовувати множину складних проектів на одному кристалі. Створивши модель спеціалізованого процесора та синтезувавши її під певну ПЛІС, визначимо, який відсоток кристала зайнятий проектом, а який – вільний. Відповідно, вільну область доцільно використати для реалізації інших задач та алгоритмів.

Як показано на рис. 2, у ПЛІС прошито моделі спеціалізованих процесорів для виконання двох задач (Рендерення та Фільтр), об'єм апаратних ресурсів для виконання третьої задачі (ДКП) перевищує допустимий об'єм ПЛІС. Це спонукає до оптимізації структури спеціалізованих процесорів цих задач з акцентом на економію апаратних ресурсів та незначну (<10%) втрату продуктивності. Такий підхід дає змогу ефективно використовувати не тільки об'єм ПЛІС, але й економить час для переходу від однієї задачі до іншої під час роботи системи.

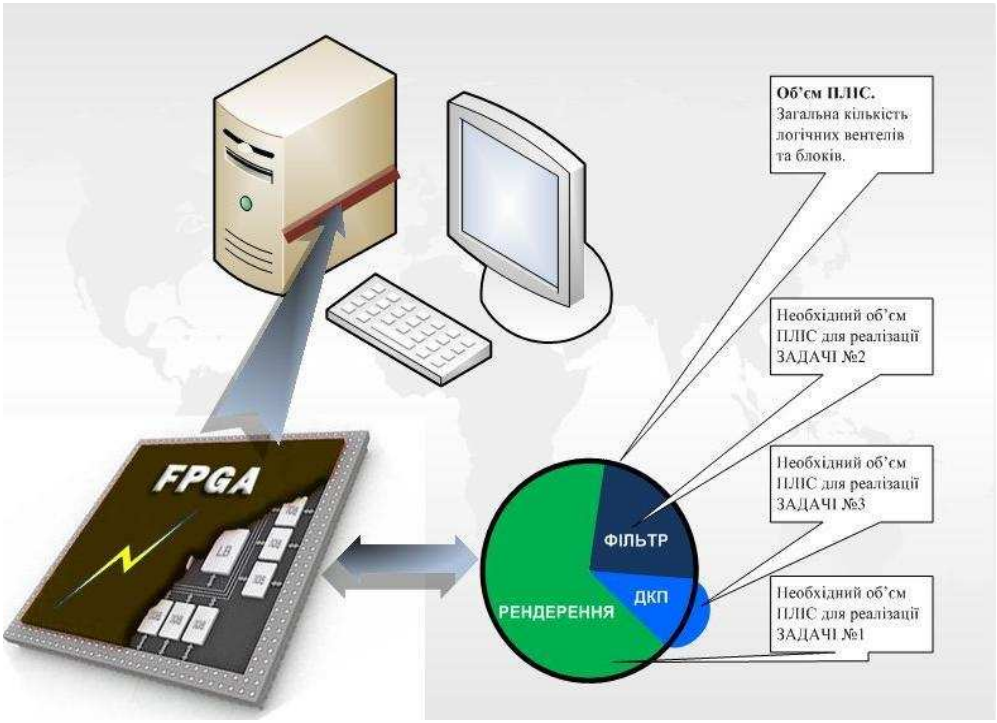

Рис. 2. Розподіл апаратних ресурсів ПЛІС для реалізації обчислень

Задачу, розв'язання якої необхідно прискорити, вибирають програмно. Драйвер реконфігуровного прискорювача подає спеціальну команду на ПЛІС, згідно з якою в певний момент часу буде розв'язано ту чи іншу задачу. Така ідея дає змогу не використовувати програматор ПЛІС як «перемикач» між задачами і відповідно економить апаратні витрати (мікросхему програматора ПЛІС) і час (необхідний для реконфігурування ПЛІС). З використанням цього підходу для створення прискорювача на ПЛІС можливі два варіанти:

- сумарний об'єм файлів конфігурації ПЛІС (для реалізації декількох задач, рис. 2) перевищує її загальний об'єм;

- з’являється невикористана область ресурсів ПЛІС.

Якщо виникає переповнення чи залишок незадіяних блоків ПЛІС, постає одна необхідність – *модифікація структури спеціалізованого процесора* із збереженням при цьому високої продуктивності системи.

Для отримання максимальної продуктивності виконання задачі спеціалізованим процесором необхідно розробити систему планування виконання команд, яка б відображала природний паралелізм задачі, тобто максимально розпаралелити незалежні вітки алгоритму (рис. 3) [5], після чого оптимізувати структуру процесорів під загальний об’єм ПЛІС.

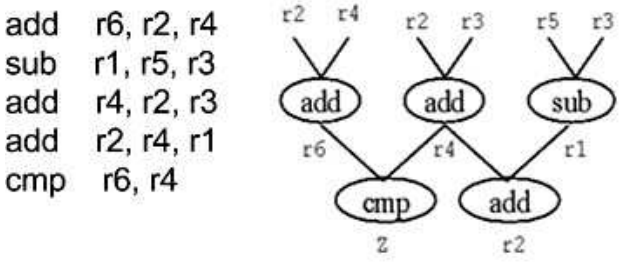

Рис. 3. Паралельне представлення коду програми

Процес проектування та оптимізації спеціалізованих процесорів під об’єм ПЛІС охоплює декілька етапів (для кожного алгоритму окремо):

1. Максимально розпаралеливши алгоритм, визначають значення x (множини A), та згідно з y (початково задається в конфігураційних параметрах програми) підбирають необхідну кількість АЛП. Враховуючи параметри алгоритму, формують множини $\{I,O,C\}$ та оптимально підбирають кількість елементів пам’яті $\{MM\}$; результатом об’єднання цих множин буде модель спеціалізованого процесора – *SC*. Тобто реалізація цієї задачі переходить з програмно-орієнтованого процесора в апаратно-орієнтований спеціалізований процесор, в якому досягається гранично висока продуктивність при мінімальних значеннях затрат обладнання та споживаної потужності.

2. Визначивши конфігурацію спеціалізованого процесора, формують множини $\{I,O,MM,A,C\}$, де їхні значення використовуватимуть для створення моделі процесора мовою опису апаратних засобів (VHDL, Verylog) [2].

3. Описавши структуру процесора, етапом синтезу HDL коду визначається необхідний відсоток використання модулів ПЛІС. Умовно це число приймається як модель спеціалізованого процесора *SC*. Відповідно математична модель реконфігурованого прискорювача складатиметься з множини моделей процесорів:

$$RA = \{SC_n\}, \tag{3.7}$$

де *RA* – реконфігурований прискорювач; *SC* – спеціалізований процесор; n – кількість задач.

Програмно перевіряють *RA* на використання усіх апаратних ресурсів ПЛІС. Якщо об’єм *RA* менший або дорівнює об’єму ПЛІС, то оптимізація задач пройшла успішно, інакше, коли об’єм *RA* перевищив загальний об’єм кристала (рис. 2.), необхідна модифікація множини A (для всіх задач, які виконує реконфігурований прискорювач) та повторення процесу розроблення структури спеціалізованого процесора *SC*. Параметри та кількість елементів множини A змінюють з огляду на зменшення апаратних ресурсів (оскільки виконано максимальне розпаралелення алгоритму, і їх збільшення є неможливим). Оптимальну кількість АЛП ($P(x,y)$) для будь-якої задачі підбирають після зміни вхідного параметра множини A – y. Значення – y є функцією, яка *обернено* залежить від двох параметрів: *використання апаратних ресурсів* (відсоток задіяних блоків ПЛІС) та *часу виконання задачі* (продуктивність). Зменшуючи один параметр, автоматично збільшується інший, і навпаки. Цей показник стає *основою* для розроблення спеціалізованого процесора. Змінивши y в більший бік (автоматично зменшуємо продуктивність), переходимо до пункту 1.

На рис. 4 показано приклад залежності кількості АЛП від кількості паралельних операцій для алгоритму восьмиточкового дискретного косинусного перетворення (ДКП – 8 т.).

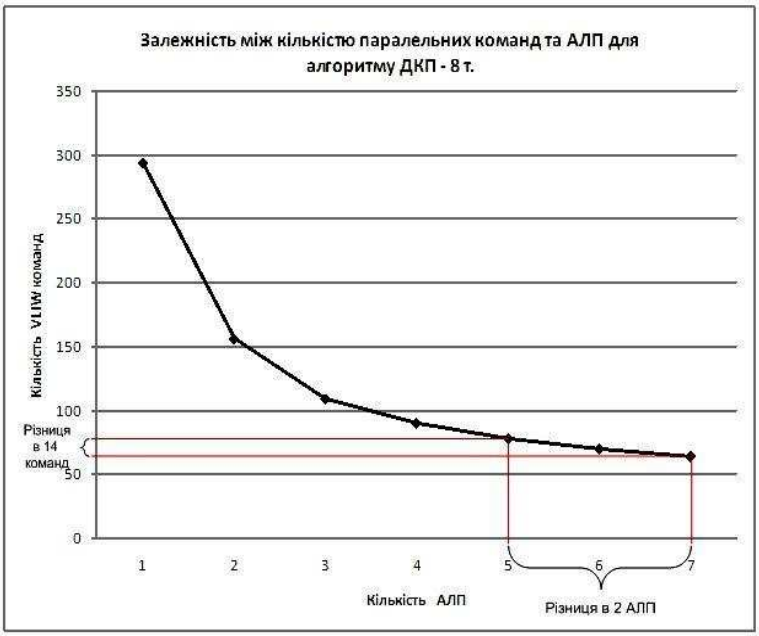

Рис. 4. Графік залежності кількості паралельних команд від кількості АЛП

Таблиця 1

Співвідношення VLIW команд до кількості АЛП для алгоритму ДКП – 8 т

Кількість АЛП	Кількість VLIW команд
1	294
2	156
3	109
4	90
5	78
6	70
7	64

За наперед заданого коефіцієнта *y* (у відсотках) визначається оптимальна кількість АЛП за процедурою 3.8, ідея якої полягає у зменшенні використання апаратних ресурсів ПЛІС:

$$
\begin{aligned}
&j=0;\\
&k=0;\\
&while\ (j < y)\\
&\{\\
&\ k=k+1;\\
&\ j=\frac{(Z_{(n-k)}-Z_n)\cdot 100}{Z_1};\\
&\}\\
&m=n-(k-1),
\end{aligned}
\qquad (3.8)
$$

де *j,y* – допустимий відсоток втрати продуктивності виконання алгоритму, Z_n – кількість VLIW команд для *n* АЛП, *n* – максимальна кількість АЛП, *k* – кількість незадіяних АЛП, без яких зберігається висока продуктивність, *m* – результуюча кількість АЛП.

Повторюючи етапи розроблення структури спеціалізованих процесорів для множини задач, досягають оптимального використання апаратних ресурсів ПЛІС. Цей метод можна використовувати для обчислень множини задач на одному кристалі. Оптимально підбираючи рівень розпаралелення кожної окремої задачі, так практично повністю використовують усі апаратні ресурси ПЛІС.

Наприклад, оптимізуємо алгоритм ДКП-8 т. з погляду апаратних витрат. Згідно з процедурою 3.8:

Наперед задамо допустимий відсоток втрати продуктивності системи – y = 8,2 % (порівняно з максимальним розпаралеленням алгориму, коли y = 0 %) та використаємо дані із табл. 1. Результатом виконання цієї процедури буде оптимальна кількість обчислювальних модулів (див. табл. 2), яка забезпечує високу продуктивність системи. Кількість VLIW команд при 5 арифметико-логічних пристроях дорівнює – 78, коли при максимальному розпаралеленні (для 7 АЛП) – 64. Різниця в 14 паралельних команд порівняно невелика відносно зменшення кількості АЛП на ≈ 29 % $\left(\frac{(k-1)\cdot 100}{n}\right)$.

Таблиця 2

Приклад оптимізації апаратних ресурсів для алгоритму ДКП-8 т

Проходи по циклу			
1	2	3	4
0<8,2 -?	2,04<8,2 -?	4,76<8,2 -?	8,84<8,2 -?
k=1;	k=2;	k=3;	
n=7;	n=7;	n=7;	
$j=\frac{(70-64)\cdot 100}{294}=2{,}04$	j=4,76	j=8,84	
			m=5;

Це число можна розглядати як відсоток економії апаратних ресурсів при втраті продуктивності системи на 4,76%. Згідно з рис. 2, коли при максимальному розпаралеленні алгоритму ДКП-8 т. для імпламентації в ПЛІС не вистачає апаратних ресурсів, то після оптимізації цієї задачі та зменшення апаратних витрат майже втричі, з'являється можливість прошити в ПЛІС алгоритм ДКП-8 т. із незначною втратою продуктивності.

Розроблені підходи покладеоі в основу побудови системи високорівневого проектування ХАМЕЛЕОН [5,6,7,8,10].

4. Етапи проектування прискорювача універсальних комп'ютерів за допомогою системи ХАМЕЛЕОН. В процесі проектування використовують такі засоби: САПР ХАМЕЛЕОН, засоби синтезу, логічної і часової симуляції та імплементації проекту в ПЛІС.

Етапи проектування:

1. Вибір основних характеристик системи;

1.1. Класу задач, які повинні виконуватися прискорювачем;

1.2. Типу побудови системи універсальний комп'ютер – реконфігуровний прискорювач;

1.3. Апаратної платформи та моделі ПЛІС;

1.4. Інтерфейсу зв'язку та розміру пакета передачі даних, від якого максимально залежить продуктивність системи.

2. Розроблення системи;Використовуючи САПР ХАМЕЛЕОН, описуємо задачу (задачі) мовою ANSI C. Вводимо початкові параметри системи (кількість АЛП – $\{A\}$, кількість портів вв./вив. – $\{I,O\}$, розрядність даних – r тощо).

2.2. Задавши *y* (відсоток втрати продуктивності виконання алгоритму), здійснюємо процес проектування (використовуючи програму ХАМЕЛЕОН), у результаті якого отримуємо VHDL модель спеціалізованого процесора –*SC*, оптимізовану з погляду використання апаратних ресурсів;

2.3. Використовуючи програмний комплекс, проводимо логічну та часову симуляцію проекту. У разі успішного виконання синтезуємо проект та визначаємо відсоток використаних модулів ПЛІС.

2.4. Якщо апаратні витрати для реалізації цього проекту перевищили загальний об'єм ПЛІС, збільшуємо *y* та переходимо до пункту 2.2, інакше виконуємо імпламентацію проекту в кристал.

3. Здійснення процесу верифікації продукту.

4. Розроблення бібліотек API функцій для організації взаємодії універсального комп'ютера з реконфігуровним прискорювачем;

5. Тестування роботи системи.

Результатом роботи буде прискорювач універсального комп'ютера, орієнтований на відповідий клас задач, який складається із ядер високопродуктивних спеціалізованих процесорів з оптимальним використанням апаратних ресурсів ПЛІС.

Висновки. У статті описано основні вимоги до реалізації програмних спеціалізованих процесорів в реконфігуровних прискорювачах. Наведено два методи організації системи "універсальний комп'ютер – реконфігуровний прискорювач" через системну та загальну шину. Досліджено переваги та недоліки цих методів. Подано рекомендації щодо організації обміну даними між прискорювачем та універсальним комп'ютером для отримання максимальної продуктивності. Наведено формули для обчислення розміру пакета передачі даних та підбору оптимальної кількості каналів. Описано ідею використання множини спеціалізованих процесорів на одному кристалі та процес вибору поточної задачі на реконфігуровному прискорювачі. Описано метод організації багатозадачної системи, який дає змогу для будь-якої задачі підібрати оптимальну кількість обчислювальних модулів, зберігаючи її продуктивність, що сприяє оптимальному використанню апаратних ресурсів ПЛІС. Наведено етапи проектування реконфігуровних прискорювачів. САПР ХАМЕЛЕОН дає змогу отримати спеціалізований процесор з наперед заданою продуктивністю виконання задачі мовою ANSI C.

1. Мельник А.А., Процессоры обработки сигналов. – Львов, 1989. – 63 с. 2. IEEE, Standard VHDL Language Reference Manual. Standard 1076-1993, NY: IEEE, New York, 1993. 3. A. Melnyk, V. Melnyk, Computer Devices Cores Design Methodology // Computer System and Networks, Lviv Polytechnic National University, Lviv, 2002. – P. 3-9. 4. J.A. Fisher, P. Faraboschi, Custom-Fit Processors: Letting Applications Define Architectures // In 29th Annual IEEE/ACM Symposium on Microarchitecture, Paris, December 1996. – 1996. – P. 324-335. 5. Henk Corporaal, Microprocessor Architectures: From VLIW to TTA, John Wiley & Sons, Inc., New York. – NY: 1997. – P. 324. 6. A. Melnyk, Newest Computer Devices Design Technology on a Base of Configurable Models // Advanced Computer System and Networks: Design and Application, Lviv, Ukraine, Sep. 24-26, ACSN-2003. – 2003.- P. 36-40 7. David Goodwin, Darin Petkov, Automatic Generation of Application Specific Processors // CASES'2003, San Jose, California, USA, Oct. 30- Nov.1. – 2003. – P. 55-62. 8. A. Melnyk, A. Salo, Automatic generation of ASICs // NASA-ISA Conference AHS-2007, Edinburgh. – 2007. – P. 96-101. 9. Шматок А. Аппаратные ускорители приложений на базе ПЛИС // Современная электроника. – 2007. – № 6. – С. 74–77. 10. Мельник А.О., Сало А.М., Клименко В., Цигилик Л., Юрчук А. Хамелеон – система високорівневого синтезу спеціалізованих процесорів // Радіоелектронні і комп'ютерні системи. – Харків: ХАІ, 2009. – С. 189–193.