УДК 004.272.26, 004.274

Клименко І.А., к.т.н.,
Опанасюк О.В.

АЛГОРИТМ ЗАВЧАСНОЇ РЕКОНФІГУРАЦІЇ ДЛЯ РЕКОНФІГУРОВНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ НА ПЛІС

**Національний технічний університет України
«Київський політехнічний інститут»**

iklymenko@yandex.ua

Запропонований новий підхід до організації процесу управління реконфігурацією та відображенням задач в реконфігуровних обчислювальних системах на базі моделі обчислень керованих потоком даних. Запропонована модифікація відомого методу автоматичного розподілу задач з врахуванням механізму завчасної реконфігурації

Ключові слова: реконфігуровані обчислювальні системи, ПЛІС, накладні видатки реконфігурації, часткова динамічна реконфігурація, управління потоком даних

Вступ

Ефективне здійснення динамічної реконфігурації, в рамках актуальної на сьогодні парадигми реконфігуровних обчислень, пов'язано з проблемами, що обумовлені непродуктивними витратами часу, продуктивності й енергоресурсів під час реконфігурації обчислювальної структури. Важливою є проблема обмеження апаратних ресурсів кристалів ПЛІС. Обмеження логічного ресурсу мікросхеми впливають на ефективність просторових рішень під час розміщення конфігурацій апаратних задач на поверхні ПЛІС. Подоланняї просторових обмежень обумовлює додаткові непродуктивні витрати. Критичними показниками з точки зору часу здійснення реконфігурації є об'єм внутрішньої пам'яті та швидкодія інтерфейсів ПЛІС.

Таким чином, основною і найбільш актуальною проблемою в області реконфігурованих обчислень є непродуктивні витрати, рішенню якої присвячена стаття. Для вирішення поставленої проблеми не ефективно використовувати існуючі технології паралельного програмування, а так само традиційні методи і засоби підвищення ефективності паралельних обчислень. Слід зауважити, що традиційні технології паралельних обчислень, на відміну від реконфігурованих обчислень орієнтовані на фіксоване обчислювальне середовище, без будь-яких просторових і функціональних обмежень.

Огляд літературних даних

Відомо багато методів та технологій зменшення накладних витрат реконфігурації. Найбільш характерні серед них: повторне використання ресурсів [1], кешування конфігураційних даних [2], апаратні засоби прискорення процесів вводу/виводу [3], випереджуюча реконфігурація [4], оптимізація віртуальної структури пам'яті конфігураційних даних [5] . Всі вони засновані на максимально можливому прискоренні процесу реконфігурації типу «*Best Effort*», без будь-якої оптимізації просторових рішень та врахування апаратних та технологічних обмежень ПЛІС. Подолання просторових обмежень ПЛІС здійснюється стандартними засобами, наприклад дефрагментацією обчислювальної поверхні ПЛІС [1, 2], вивантаженням некритичних конфігурацій [1, 4, 5], відмовою виконання задач [1, 2, 4], що вносить додаткові накладні витрати в процес реконфігуровних обчислень. Розглянуті механізми зменшення непродуктивних витрат реконфігурації зазвичай вбудовані в логіку алгоритмів управління реконфігуровними ресурсами та відображенням задач на реконфігуровну структуру. Такі алгоритми в відомих системах реалізовані на рівні апаратного забезпечення надбудови операційної системи і

мають високу часову складність, що складає значну частину непродуктивного часу реконфігурації.

Постановка задачі

Час виконання задачі в реконфігуровній системі визначається, як сума часу виконання та часу реконфігурації, позначимо його, як T_{Sum}, і обчислимо наступним чином:

$$T_{Sum} = T_{Count} + T_{Rconf} ,$$

де T_{Count} – час виконання задачі апаратними засобами, T_{Rconf} – час реконфігурації апаратної задачі на ПЛІС. Складова T_{Count} визначає позитивний ефект від традиційних способів зменшення часу виконання задач, шляхом поглиблення паралелізму та апаратного прискорення. Таке прискорення очікуване від застосування реконфігурованих обчислень. Складова T_{Rconf} є непродуктивною складовою, яка визначає час витрачений саме на процес реконфігурації апаратури. Ця складова обумовлює витрати продуктивності під час реконфігурованих обчислень і вимірюється накладними видатками. При цьому в деяких випадках час апаратного прискорення задач і час їх конфігурації на апаратурі можуть бути порівняні у часі [1]. Час реконфігурації в свою чергу складається наступних компонент: часу виконання логічної та фізичної послідовності реконфігураційного процесу, що забезпечується на програмному рівні алгоритмами планування та розподілу задач; часу передавання конфігураційних даних, які можуть завантажуватись із загальної системної бібліотеки конфігураційних файлів, із зовнішньої локальної пам'яті обчислювального модуля або із внутрішньої пам'яті ПЛІС, часу прошивання мікросхеми.

З огляду на вищесказане, до найбільш вагомих компонент накладних видатків відносяться витрати продуктивності на забезпечення обчислювальної складності алгоритмів планування та розміщення задач, та час витрачений на передавання і завантаження конфігураційних даних. В статі вирішується задача зменшення накладних видатків реконфігурації за рахунок зменшення часової та обчислювальної складності алгоритмів управління реконфігурацією і зменшення комунікаційного часу реконфігурації.

В якості реконфігурованого обчислювального середовища розглядається динамічна область кристалу ПЛІС [6], яку визначимо як реконфігуровану область. Реконфігурована область має розмір і структуру. Розмір залежить від можливостей використовуваного сімейства ПЛІС і архітектури реконфігурованої системи. Структура визначається використовуваною моделлю розміщення задач [6].

В якості абстрактної моделі обчислювальної задачі розглядаємо функцією або так зване функціональне ядро, синтезоване в цифрову схему з метою розміщення на реконфігуровній області. Визначимо її, як апаратну задачу. Застосуємо традиційну модель задачі абстрактного рівня [1], коли задача характеризується розміром, формою відображення і часом виконання. Розмір визначає просторові вимоги до реконфігурованої області. Форма відображення залежить від моделі розміщення задач в реконфігурованій області. Для всіх моделей розміщення узагальнимо, що задача має форму прямокутника. Тоді задача визначається як вектор

$$T_i = \{S_i, T_{Sum\,i} \,|\, (T_{Count\,i} + T_{\mathrm{R}config\,i}), I_i, N_i\},$$

де S_i – площина прямокутника, що містить задачу T_i, а час реконфігурації $T_{\mathrm{R}config\,i}$ це частина часу її виконання $T_{Sum\,i}$, I_i – тип операції, що виконує задача, і N_i – унікальний ідентифікатор задачі у вихідному графі алгоритму, $i = \overline{1, N_{\max}}$, $N_{\max}$ – кількість вершин графу. Вважаємо що задачі розміщуються на поверхні реконфігуровної області згідно деякому алгоритму планування та розміщення задач з урахуванням їх просторових параметрів. Рішення просторових задач, та їх оптимізація в роботі не розглядається.

Таким чином, основною ціллю досліджень є загальне прискорення обчислень. Для забезпечення очікуваного ефекту від застосування реконфігуровних, має бути вирішена задача зменшення компонент часу реконфігурації $T_{\mathrm{R}config}$, принай-

мні в степені порівняній з прискоренням обчислень, що зводитиме до нуля накладні витрати реконфігурації згідно критерію ефективності:

$$r = \frac{T_{SW}}{[T_{Control} + T_{Rconf}] + T_{HW}}, \qquad (2)$$

де T_{SW} – час обчислення задачі на процесорному ядрі, T_{HW} – час обчислення задачі апаратними засобами, визначений та обґрунтований авторами в попередніх роботах [4, 7].

Оцінка ефективності застосування моделі обчислень керованих потоком даних в реконфігуровних обчислювальних системах

Реалізація моделі управління потоками даних, на відміну від традиційних послідовно-паралельних або паралельно-конвеєрних обчислень, дозволяє підвищити ефективність розпаралелювання на рівні макрокоманд за рахунок реалізації прихованого паралелізму. При цьому потокова модель управління дозволяє реалізувати ефективні апаратні засоби управління реконфігурацією на структурному рівні обчислювального модуля.

Ефективність парадигми апаратної реалізації засобів управління реконфігурацією, що описана та досліджена в попередній роботі авторів [8], надає значне збільшення ефективності в безпосередній близькості до фізичного рівня реконфігурованого модуля. На абстрактному рівні логічної послідовності реконфігурації апаратна реалізація зменшує навантаження на локальний процесор і прискорює реалізацію логічних алгоритмів управління. Абстрагування логічної послідовності управління реконфігурацією від системних задач обумовлює звуження їх специфічності, що в певній мірі спрощує їх розробку і застосування. В такому контексті перенос на апаратну структуру управляючих засобів принесе максимальний ефект від їх застосування.

З іншого боку децентралізація й локалізація засобів управління реконфігурованими ресурсами на структурному рівні однотипних обчислювальних модулів дозволяє прискорити процес реконфігурації, розвантажити операційну систему від рішення неспецифічних задач, зменшити рух даних на міжмодульному рівні, забезпечити мобільність технологій та спрощення масштабування обчислювальної потужності

В роботі досліджується ефективність застосування відомого методу автоматичного розподілу задач, описаного в роботі [7], в контексті зменшення часової складності алгоритмів управління реконфігурацію та відображенням задач, згідно з критерієм (2) за рахунок апаратної реалізації цього методу на рівні обчислювального модуля. Модифікація відомого методу стосується втілення в модель управління потоками даних механізмів зменшення непродуктивних витрат часу реконфігурації, а саме механізму завчасної реконфігурації.

Узагальнена концепція побудови архітектури реконфігуровних обчислювальних систем з можливістю застосування запропонованих засобів детально описана в попередній роботі авторів [8].

Розробка алгоритму завчасної реконфігурації

Програма задається у вигляді графу потоків даних $G_D = (W, D)$, де W – множина вершин, що відповідають задачам (завдання), а D – множина ребер, що визначають відношення між задачами (дані).

Завдання описується множиною елементів, яка визначає слово команди/завдання (актор). Для і-ої вершини графа актор має вигляд:

$$W_i = \{N_i, n_i, I_i, Q_i, M_i, R_i\},$$

де Ni - ім'я актора; ni – ознака актора; Ii - ідентифікатор завдання (виконувана в вершині графа функція); Qi - сумарна кількість потоків даних для завдання (вхідних дуг графа); Mi – множина імен акторів, для яких і-е завдання готує дані; Ri – множина імен потоків даних, які формуються при виконанні і-го завдання.

Потік даних, який відповідає дузі графа, що з'єднує d-у вершину з і-ю вершиною, характеризується дескриптором, що є множиною елементів:

$$D_{id} = \{N_i, n_{id}, Q_i, A_{id}\},$$

де n_{id} – ім'я (номер) потоку даних (дуги, що входить до і-у вершину); A_{id} – елемент адресації даних.

Всі вершини графа мають унікальні імена $N_i (i = \overline{1,m})$ Елемент адресації визначає місце розташування даних у пам'яті системи, доступної для кожного процесора. Ім'я потоку одночасно є ознакою, що відрізняє дескриптор даних від актора.

Для реалізації завчасної реконфігурації стає задача визначення критичного шляху потокового графу алгоритму. В роботі запропонований алгоритм формування критичного шляху, який заснований на використанні матриці зв'язності відповідного графу обчислювального алгоритму.

Опишемо граф програми в вигляді матриці зв'язності M_C розмірності $(v \times v)$. Введемо наступні позначення: w – кількість ярусів графу C; $W_j \mid (j = \overline{1,w})$ – поточний ярус; n_j – кількість вузлів на ярусі W_j. Рядки $M_C[g] \Big| g = \overline{1,v}$ відповідають виконуваним задачам. В стовпцях $M_C[k] \Big| k = \overline{1,v}$ кожного рядку $M_C[g]$ розміщується одиниця, якщо задача $Task_g$ готує дані для задач $Task_k$. Отримати множину всіх задач які готують дані для задачі $Task_k$, можна проаналізувавши елементи стовбця $M_C[k]$, в якому номери g елементів стовбця $M_C[g,k]$ вказують на номери задач $Task_g$, якщо $M_C[g,k]=1$.

Кожна задача асоціюється з параметром «Тип операції» (I_i). Тип операції відповідає функції, яку виконує задача, що синтезована в цифрову схему і завантажується в функціональний блок під час реконфігурованих обчислень. Апаратні реалізації задач створюються заздалегідь і зберігається в централізованій бібліотеці конфігурацій у вигляді конфігураційних файлів, кількість яких дорівнює M_I – по одній для кожного типу операції. Таким чином тип операції $I_j \mid j = \overline{1, I_{\max}}$, де $I_{\max}$ – кількість апаратно реалізованих типів операцій, однозначно визначає конфігураційний файл, що завантажується в функціональний блок і використовується як основний параметр для пошуку його конфігурації в пам'яті, або на мікросхемі ПЛІС. Формальна модель алгоритму завчасної реконфігурації приведена на рис. 1.

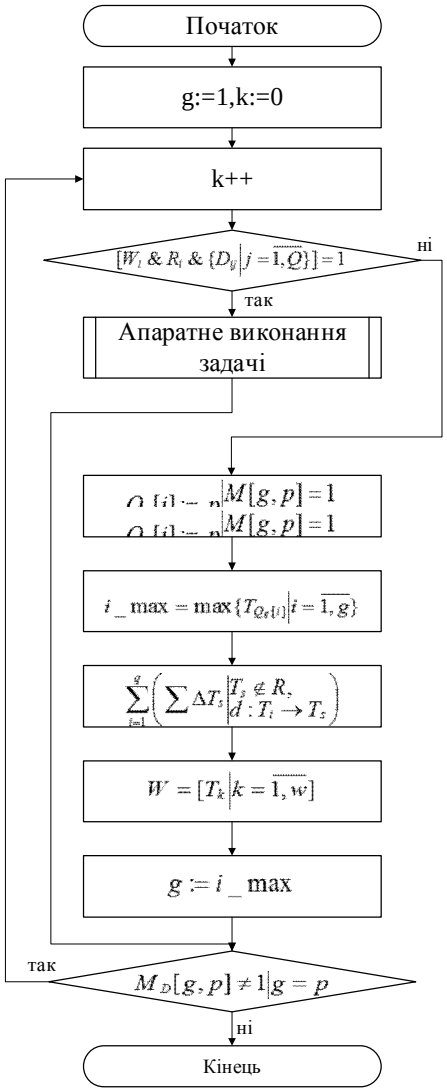

Рис. 1. Формальна модель алгоритму завчасної реконфігурації

складає в середньому 5 %. Але за рахунок використання принципу керування потоком даних, зрозуміло, що не потребує ніяких додаткових дій до початку обчислень, а також зберігання додаткової інформації про виконувану задачу, що і дає перевагу над методом статичного планування.

Висновки

Запропоновано та досліджено застосування моделі керування потоками даних для рішення проблеми відображення задач в реконфігуровних обчислювальних систем, що базуються на модифікації відомого методу автоматичного динамічного розпаралелювання для моделі обчислень під управлінням потоку даними. Модифікація методу стосується втілення в модель управління потоками даних механізму завчасної реконфігурації.

За результатами досліджень реалізація запропонованого алгоритму управління реконфігурацією з підтримкою механізму завчасної реконфігурації забезпечує прискорення обчислювального процесу в середньому на 5% за рахунок видалення непродуктивних витрат часу реконфігурації.

Запропоновані засоби зменшують обчислювальну складність алгоритмів управління логічною послідовністю реконфігурації та відображенням задач, за рахунок апаратної реалізації на рівні обчислювального модуля локалізовано від програмного прошарку операційної системи.

Список літератури

1. Bassiri M.M. Mitigating Reconfiguration Overhead In On-Line Task Scheduling For Reconfigurable Computing Systems / M.M. Bassiri, S.H. Shahriar // Proceeding of 2nd International Conference on Computer Engineering and Technology (ICCET), (Chengdu, 16 – 18 April 2010). – 2010. – Vol. 4. – P. V4 – 397. – V4 – 402.

2. Al-Wattar A. Efficient On-line Hardware/Software Task Scheduling for Dynamic Run-time Reconfigurable Systems / A. Al-Wattar, S. Areibi, F. Saffih // Proceeding in 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW). – IEEE, 2012. – P 401 – 406.

3. Liu S. Achieving Energy Efficiency through Runtime Partial Reconfiguration on Reconfigurable Systems / S. Liu, R.N. Pittman, A. Forin, J.-L. Gaudiot // Transactions on Embedded Computing Systems (TECS). – USA, NY, New York, ACM, 2013. – Vol. 12. – P.72:1 – 72:21.

4. Кулаков Ю.О. Розробка методу прискорення реконфігурації в динамічно реконфігурованих обчислювальних системах // Ю. О. Кулаков, І. А. Клименко, М. В. Рудницький 2015 // Східно-Європейський журнал передових технологій. – Харків.: УДА залізничного транспорту. – 2015. – №4/4 (76). – С. 25 – 29.

5. Кулаков Ю.О. Організація багаторівневої пам'яті в реконфігурованих обчислювальних системах // Ю.О.Кулаков, І.А.Клименко / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. Наук. Пр. – К.: Век+, 2014. – №61. – С. 18 – 26.

6. Клименко І.А. Класифікація реконфігурованих обчислювальних систем / І.А. Клименко, М.В. Рудницький // Вісник Вінницького політехнічного інституту. – Вінниця: ВНТУ, 2014. - №5 (116). – С. 120 – 128.

7. Жабин В.И. Реализация вычислений под управлением потока дескрипторов данных в мультипроцессорных системах / В.И. Жабин // Электронное моделирование. – К.: ИПМЕ, 2003. – Т.25, №1. – С. 35 – 47.

8. Клименко І.А. Аналіз ефективності управління ресурсами та додатками в реконфігурованих обчислювальних системах / І.А. Клименко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2015. – № 62. – С. 11 – 21.

Статтю подано до редакції 01.06.2016