УДК 004.07

Р.Б. ДУНЕЦЬ, Д.Я. ТИХАНСЬКИЙ

Національний університет «Львівська політехніка», Україна

ДОСЛІДЖЕННЯ ЧАСТКОВОЇ РЕКОНФІГУРАЦІЇ ПЛІС

Проведено аналітичний огляд методів часткової реконфігурації вибраних ділянок ПЛІС у будь-який час після початкової конфігурації, що може здійснюватись як в активному режимі, тобто в процесі її роботи, так і в зупиненому режимі. Особлива увага звернена на час реконфігурації. Показано, що динамічна реконфігурація необмежена тільки одним фіксованим модульним слотом, а й групи модулів з відповідною комунікаційною інфраструктурою.

Ключові слова: *реконфігурація, часткова реконфігурація, ПЛІС, частково реконфігурований модуль, макрос, апаратні ядра, програмні ядра.*

Вступ

Сьогодні цифрові системи різноманітного призначення все частіше і частіше реалізуються як спеціалізовані комп'ютерні системи на ПЛІС, оскільки вони мають краще співвідношення ціна/продуктивність, а також коротший термін розробки проекту. Разом з тим, до застосування таких систем в області мобільних комунікацій, проведення обчислень бортовими системами, побутової техніки, тощо, все більше ставиться вимога до збільшення їх гнучкої, шляхом розширення та модифікації функціональних можливостей після випуску, тобто в процесі експлуатації. Такі вимоги можна задовольнити шляхом побудови реконфігурованих систем.

В загальному випадку конфігурація системи завантажується у ПЛІС в кінці циклу розробки проекту і залишається незмінною протягом всього часу її роботи. Реконфігурація загалом вимагає зупинки роботи системи, реконфігурації та перезапуску. З розвитком технології стало можливим зменшення часу, необхідного для проведення реконфігурації, а, відтак, стало можливим реконфігурувати ПЛІС між різними стадіями чи режимами роботи. Цей процес отримав назву реконфігурація в реальному часі, а відповідні системи на ПЛІС – динамічно реконфігурованими.

Проте, більшість реальних систем є занадто великими, щоб розміститися на окремому кристалі. В такому випадку загальну задачу розділяють на множину менших задач, і кожну з них реалізують на окремому чіпі, який можна буде реконфігурувати в ході виконання різних задач. Подальший розвиток технології ПЛІС дав можливість реконфігурувати тільки деякі з логічних елементів сучасних чіпів. Така часткова реконфігурація, в загальному випадку, є набагато швидшою, оскільки невелика частина ПЛІС має бути зміненою. Очевидно, що застосування такої часткової динамічної конфігурації ПЛІС може розглядатися як спосіб підвищення ефективності архітектури спеціалізованих комп'ютерних систем.

Потрібно зазначити, що прикладів практичного застосування часткової динамічної конфігурації ПЛІС є не багато попри те, що теоретичні дослідження проводяться впродовж більше десятьох років. Головною причиною відсутності практичних рішень є те, що на сьогодні практично нема ефективних інструментальних засобів побудови частково реконфігурованих ПЛІС. Крім того, більшість реалізацій динамічно реконфігурованих систем використовують прості підходи, які базуються на фіксованих модульних слотах. Першим кроком в напрямку створення ефективних інструментальних засобів побудови частково реконфігурованих ПЛІС є проведення аналізу відомих рішень, що й покладено в основу даної роботи.

1. Реконфігуровані архітектури систем

Відомі реконфігуровані системи можна розглядати в різних площинах, зокрема виходячи з того, хто керує процесом реконфігурації, коли конфігурація може бути згенерованою та якою є ступінь деталізації реконфігурації.

Щодо процесу реконфігурації, то він поділяється на два етапи – зовнішню та внутрішню реконфігурацію. Зовнішньою реконфігурацією керує зовнішній арбітр, зазвичай персональний комп'ютер. Внутрішня реконфігурація виконується вже безпосередньо самою ПЛІС. Для того, щоб це було можливим, ПЛІС повинна мати спеціальний фізичний компонент, такий, як наприклад, компонент ICAP в ПЛІС фірми Xilinx.

Що стосується створення самої конфігураційної послідовності, то її можна створити безпосередньо під час розробки проекту, враховуючи всі можливі варіанти конфігурації системи. Кожний модуль має бути синтезовано, а всі можливі зв'язки між модулями системи в процесі реконфігурації треба переглядати. Є інший, більш гнучкий шлях, який полягає у створенні під час проектування системи пресинтизованих модулів, що потребують динамічного перенаправлення сигналів взаємодії або створенні повністю динамічно реконфігурованих модулів. Цей шлях наразі не реалізується, оскільки передбачає синтез модулів з VHDL коду, на що затрачається дуже багато часу.

Стосовно ступеня реконфігурації, то він може мати два рівні – локальний та модульний. Перший рівень заключається в проведенні зміни одиничного конфігураційного логічного блоку, а другий – в зміні більшої частини ПЛІС, використовуючи раніше створені компоненти (модулі), що можуть бути додані або вилучені з системи кожного разу під час реконфігурації.

В проекті Garp університету Берклі [1], ПЛІС розглядалася як обчислювальний пристрій, що налаштовується під потреби обчислювального середовища, що включає структуровані програми, бібліотеки, контекстне переключення, віртуальну пам'ять і багато користувачів.

В іншому проекті [2] пропонується PRISM (Processor Reconfiguration through Instruction-Set Metamorphosis) архітектура, в якій для пришвидшення виконання програм передбачено спеціальний реконфігурований елемент, в якому синтезуються нові процесорні інструкції.

У проекті WUGS (Washington University Gigabit Switch) [3] запропоновано інший підхід, який на базі елементів з частковою динамічною реконфігурацією, забезпечує швидкий обмін програмними модулями. Він отримав назву Dynamic Hardware Plugin (DHP). Проект був реалізований на ПЛІС Virtex-E фірми Xilinx.

Спеціальний інструмент PARBIT [4] перетворює конфігураційні послідовності ПЛІС, створюючи DHP модулі для запису їх у ПЛІС. На його вхід подається оригінальна бітова послідовність та параметри, які вводить користувач, а на виході отримується нова бітова послідовність, яка може завантажити модуль DHP у необхідну зону ПЛІС.

Гнучкість даного підходу обмежена наперед встановленими розмірами та місцезнаходженням зарезервованої площі ПЛІС, де DHP модулі можуть бути розташовані, зменшуючи тим самим використання логіки ПЛІС. Особливо це стосується ситуації, коли використовуються DHP модулі різного розміру та виникає необхідність у зовнішньому пристрої реконфігурації.

2. Частково реконфігуровані системи

Проект частково реконфігурованої системи потребує розділення ПЛІС на області, статичні або динамічні. На рис. 1 показано схему розділення ПЛІС на області.

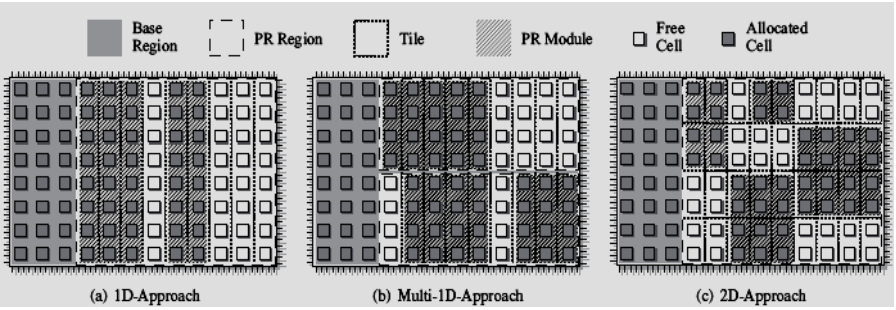

Рис. 1. Схеми поділу ПЛІС

Частково реконфігурована система складається зі статичних і динамічних компонентів. До статичних компонентів відносять частини системи, які завжди присутні, наприклад, менеджер реконфігурації або контролер пам'яті. Динамічні компоненти представлені апаратними модулями, наприклад, акселератори для криптографії, для комплексної арифметики чи обробки пакетів. Динамічні компоненти завантажуються під час роботи системи в процесі часткової реконфігурації.

Стосовно просторового розташування областей реконфігурації ПЛІС, то за основу можна прийняти підхід, що описаний в роботі [6], який передбачає наявність основної області та частково реконфігурованої області. Основна область описує ту частину ПЛІС, яка одноразово конфігурується під час ініціалізації всієї системи. Конфігурація основної області не змінюється під час роботи системи, а тому її можна розглядати як псевдостатичну. Всі статичні компоненти системи зосереджено в базовому регіоні.

Частково реконфігурована область (ЧР область) на відміну від основної реконфігурується під час ро-

боти системи. Всі динамічні компоненти системи зосереджені в цій області. Частково реконфігурована система може мати одну або декілька окремих ЧР областей. Крім того, частково реконфігурована область може складатися з одного або декількох окремих реконфігурованих вузлів, що є найменшими реконфігурованими одиницями, які в ПЛІС фізично реалізуються кількома логічними елементами.

Частково реконфігурований модуль, що реалізує динамічний компонент системи, може бути як розташований, так і видалений під час роботи системи відповідно до потреб програми на вільні вузли ПЛІС.

У найпростішому випадку один вузол покриває всю площу ЧР області і тоді відповідні модулі можуть реалізуватися лише на одному вузлі. В результаті максимальна кількість модулів, яка може бути розміщена і виконана в один і той самий проміжок часу, рівна кількості ЧР областей. Перевага цього підходу полягає в тому, що під час роботи системи, розміщення модуля проводиться простим вибором вільної ЧР області для розміщення модуля [7]. Однак, недоліком цього підходу є неефективне використання ресурсів, оскільки невеликі модулі можуть займати цілу ЧР область. Крім цього, розмір модуля обмежений розміром самого вузла.

Щоб зменшити цей недолік ЧР область може бути розділена на багато вузлів так, як показано на рис. 1. Модуль в цьому випадку – це група суміжних вузлів. Процес розташування модуля еквівалентний пошуку площі з необхідною кількістю суміжних вузлів. Кількість і розмір вузлів, а також зв'язки в ЧР області можуть бути реалізовані трьома способами.

Перший спосіб – це 1D-розбиття (рис. 1, a). Висота вузлів відповідає висоті ЧР області, а ширина вибирається згідно конкретної потреби для даної системи. Так, наприклад, для ПЛІС типу Virtex, зазвичай вибирають ширину, яка складає 4 CLB стовпчика. Вузли розташовані пліч-о-пліч, а тому необхідний модуль може бути розташований в окремому місці (позиції) з достатньою кількістю вільних вузлів.

Якщо ЧР область є великою, а модулі є достатньо малими в 1D-розбитті, то це може призвести до неефективного розміщення внутрішніх зв'язків між модулями. У цьому випадку застосовують другий спосіб – мульти-1D-розбиття (рис. 1, b), у якому ЧР область поділяють на одинакові підобласті, кожна з яких знову розбивається на багато вузлів, аналогічно 1D-розбитті. Висота вузла відповідає висоті підобласті. Модуль може бути розміщений в одну з підобластей у будь-яку позицію з достатньою кількістю вільних суміжних вузлів. Таке розбиття найбільш підходить для ПЛІС із стовпцево орієнтованою реконфігурацією. Так, наприклад, для Xilinx Virtex-4/5 найменшою частково реконфігурованою одиницю є конфігураційний фрейм, який складається з декількох вертикально розміщених логічних елементів. Висота підобласті є кратною висоті конфігураційного фрейму.

Третій спосіб – це 2D-розбиття (рис. 1, c). Він передбачає, що кожен модуль реалізується прямокутними групами вузлів. На відміну від мульти-1D-розбиття, висота модуля тепер необмежена висотою ЧР області. При генеруванні модуля площу та співвідношення сторін можна оптимізувати відповідно до внутрішніх зв'язків модуля. Хоч 2D-розбиття пропонує найбільшу гнучкість розміщення, його практичне застосування, наприклад, на ПЛІС Virtex 4/5 на сьогодні гальмується відсутністю необхідних засобів, які забезпечують однорідність вузлів та встановлення зв'язків між ними.

І, на кінець, концепція часткової реконфігурації потребує належної комунікаційної інфраструктури для з'єднання ЧР модулів і базової області. Комунікаційна інфраструктура не повинна створювати неоднорідність у системі, щоб не зменшувати гнучкість розміщення, зберігаючи при цьому число придатних позицій для розміщення модулів.

3. Комунікаційні інфраструктури

Шинні комунікаційні інфраструктури включають загальні сигнали і внутрішні сигнали. Загальні сигнали зазвичай використовуються для передачі даних і адресування інформації, а для контролю і арбітражу – внутрішні сигнали. В контексті динамічно реконфігурованих систем загальні і внутрішні сигнали, від базової області до модулів, реалізуються як статичні комунікаційні лінії.

В роботі [7] описано комунікацію фірми Xilinx, згідно якої для організації зв'язків між модулями і основною областю застосовуються шинні макроси. Шинний макрос визначає фіксований набір точок, які використовуються для передачі сигналів між модулями і базовою областю. В межах кожного модуля точки з'єднання розміщені в позиції, як показано на рис. 2.

Власне такий тип підключення називають макросом підключення. Макрос підключення може бути використаний для реалізації міжмодульної комунікації в простих схемах розбиття з декількома вузлами. Якщо модуль розташовується в одному вузлі, тоді кожний вузол потребує макроса з'єднання з базовою областю, щоб була змога розмістити модуль в кожному вузлі. Вузли, які несуміжні з базовою областю, потребують адаптованого макроса з'єднання, де маршрути проходять через суміжні вузли для з'єднання з базовою областю.

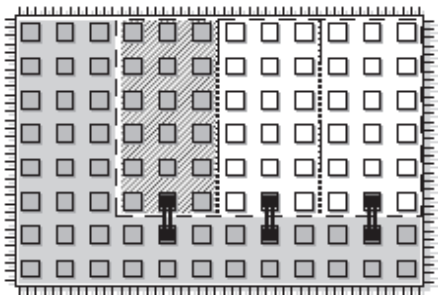

Рис. 2. Приклад реалізації макроса з'єднання

Макрос з'єднання забезпечує пряме з'єднання „точка-точка", а тому він добре підходить для спеціалізованих сигналів. Проте він не може бути використаний для генерування користувацької комунікаційної інфраструктури для загальних сигналів. В розширеній схемі розбиття, такій як зображено на рис. 1, комунікація кожного вузла з базовою областю через макрос з'єднання може ввести додаткову неоднорідність або взагалі бути неможливою. Неоднорідність викликана комунікаційною інфраструктурою може обмежити кількість виконуваних модулів. У найгіршому випадку кожний модуль може бути розміщений тільки на одній виконуваній позиції. Для збільшення кількості виконуваних позицій, можна згенерувати додаткові модульні варіанти з різними виконуваними позиціями. Однак, кожний модульний варіант базується на власних конфігураційних даних, котрі потребують збереження їх в пам'яті системи. Відповідно, таке збільшення гнучкості в розміщенні модулів досягнуто за рахунок додаткової пам'яті. Крім того, модулі з одною виконуваною позицією не можуть бути перерозміщені під час виконання, що не дає змоги робити дефрагментацію під час роботи системи.

4. Аналіз часу реконфігурації

Для дослідженя часових характеристик процесу часткової реконфігурації була використана платформа з ПЛІС XC2V1000 фірми Xilinx. Час реконфігурації T_r складений зі суми часу ініціалізації і часу передачі

$$T_r = T_i + T_t . \quad (1)$$

Час ініціалізації T_i – це час читання і опрацювання першого конфігураційного слова з пам'яті. Час передачі T_t – це час, витрачений на пересилання всіх інших конфігураційних слів через конфігураційний інтерфейс пристрою. В свою чергу T_t як

$$T_t = N_{KC} \times T_W , \quad (2)$$

де N_{KC} – кількість конфігураційних слів на бітову послідовність; T_W – час пересилання кожного слова через конфігураційний інтерфейс.

В результаті час реконфігурації визначиться як

$$T_r = T_i + N_{KC} \times T_W . \quad (3)$$

Для порівняння часу часткової реконфігурації і часу повної реконфігурації був використаний модуль контролю конфігурації з часом реконфігурації, наведеним в [4], та USB кабель Xilinx для конфігурації з часом реконфігурації, наведеним в [5]. Для реконфігурації через кабель використовувався програмний продукт Impact фірми Xilinx. Загальний розмір конфігураційної послідовності ПЛІС XC2V1000 складає 127 581 32-х бітових слова.

$$T_r = 884\text{ns} + N_{KC} \times 748\text{ns} \quad (4)$$

$$T_r = 160\text{ms} + N_{KC} \times 8,44\mu\text{s} \quad (5)$$

На рис. 3 зображено графік порівняння реконфігураційних часів відповідно до формул 4 і 5.

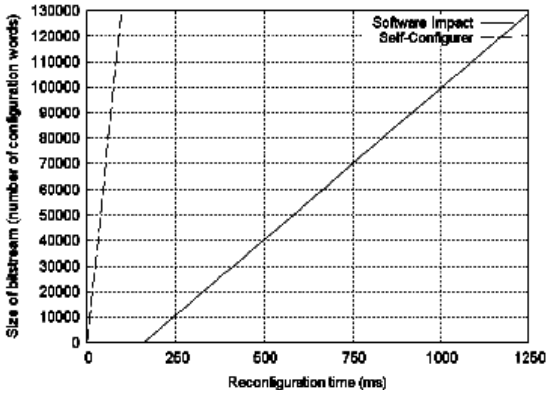

Рис. 3. Порівняння часу реконфігурації

Порівняння часів виконання програмної і апаратної реалізації виконання операцій зображено на рис. 4. З суфіксом *_hw* показано час виконання апаратної реалізації, а з суфіксом *_sw* показано час виконання програмної реалізації операцій. Всі апаратні сопроцесори були синтезовані в одній реконфігурованій області, відповідно бітові послідовності для них однакові. Кожна бітова послідовність конфігурувалась приблизно 10ms.

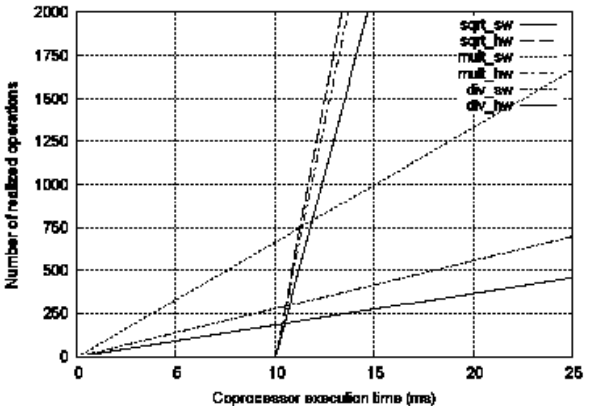

Рис. 4. Час виконання операцій

Аналіз показує, що при виконанні більш ніж 750 послідовних множень, апаратні засоби їх реалізують швидше, навіть враховуючи час реконфігурації. Для операцій ділення і добування квадратного кореня ефе-

ктивність реконфігурованих засобів досягається вже при послідовності 260 і 200 операцій відповідно.

Висновок

Таким чином, спеціалізовані комп'ютерні системи з частковою динамічною реконфігурацією не обмежуються застосуванням лише простих підходів до розташування модулів. У них доцільно застосовувати для організації з'єднань в комунікаційних інфраструктурах вбудовані макроси.

Особлива увага повинна бути приділена зменшенню часу реконфігурації, оскільки він є складовою загального часу роботи системи. На сьогодні такі системи найкраще застосовувати у тих випадках, коли розв'язок задач пов'язаний з великою кількістю довгих операцій множення ділення тощо. Прикладом можуть бути задачі цифрової обробки сигналів, зокрема фільтрації, спектрального аналізу, мультимедіа.

Література

1. Hauser J.R. Garp: A MIPS Processor with a Reconfigurable Coprocessor. / J.R. Hauser, J. Wawrzynek // IEEE Symposium on FPGAs for Custom Computing Machines, 1997.

2. Silverman H.F. Processor reconfiguration through instruction–set metamorphosis. / H.F. Silverman // IEEE Computer, 1993.

3. Edson H. Dynamic hardware plugins in an fpga with partial run–time reconfiguration. / H. Edson, J.W. Lockwood, D. Parlour. – 1993. – P. 844-848.

4. Edson H. Parbit: A tool to transform bitfiles to implement partial reconfiguration of field programmable gate arrays. / H. Edson, J.W. Lockwood. - Washington University, Department of Computer Science, Technical Report WUCS–01–13, July 2001.

5. Guccione S. Run–time parameterizable cores. / S. Guccione, D. Levi // IEEE Symposium on Filed Programmable Logic and Application. – 1999. – P. 215-222.

6. Xilinx. Early access partial reconfiguration user guide // UG208, 2006.

7. Xilinx Inc. Application notes 290. Two flows for partial re-configuration: Module based or small bit manipulations, 2002.

8. Ewerson C. Reconfiguration Control for Dynamically Reconfigurable Systems / C. Ewerson, N. Calazans, F. Moraes, D. Mesquita, 2004.

9. Hagemeyer J. Design of homogeneous communication infrastructures for partially reconfigurable fpgas. / J. Hagemeyer, B. Kettelhoit, M. Koester // 2007 Int. Conf. on Eng. of Rec. Systems and Algorithms (ERSA'07). - Las Vegas, USA. - June 25-28, 2007.

10. Butel P. Managing partial dynamic reconfiguration in Virtex-II Pro FPGAs. /P. Butel, G. Habay, A. Rachet // Xcell Journal, 2004.

11. Doraiaj N. PlanAhead software as a platform for partial reconfiguration. / N. Doraiaj, E. Shiflet, M. Goosman // Xcell Journal. - 2005.

Надійшла до редакції 11.02.2009

Рецензент: д-р техн. наук, проф., проф. кафедри інформаційних технологій та телекомунікаційних систем Ю.П. Рак, Львівський державний університет безпеки життєдіяльності, Львів, Україна.

ИССЛЕДОВАНИЯ ЧАСТИЧНОЙ РЕКОНФИГУРАЦИИ ПЛИС

Р.Б. Дунец, Д.Я. Тыханский

Проведен аналитический обзор методов частичной реконфигурации отдельных участков ПЛИС в произвольный момент времени после изначальной конфигурации, которые могут осуществляться как в активном режиме, то есть в процессе ее работы, так и в остановленном состоянии. Особое внимание уделено времени осуществления реконфигурации. Показано, что динамическая реконфигурация не ограничена только одним фиксированным модульным слотом, а и группой модулей с соответствующей коммуникационной инфраструктурой.

Ключевые слова: реконфигурация, частичная реконфигурация, ПЛИС, частично реконфигурированный модуль, макрос, аппаратные ядра, программные ядра.

RESEARCHING OF THE PARTIAL RECONFIGURATION FPGA

R.B. Dunets, D.Y. Tykhanskyi

Analytic review of advanced methods for partial FPGA reconfiguration for selected areas at anytime after initial configuration has been made in the paper. The review covers FPGA reconfiguration methods which allow changing FPGA configuration on real time or in idle time. Key point of the analytic review is reconfiguration time. In the paper are shown, what dynamic reconfiguration are not limited by one fixed slots unit. It covers other units with corresponding communication architecture also.

Key words: reconfiguration, partial reconfiguration, FPGA, partial reconfigurable unit, macro, IP Cores, programmable IP cores.

Дунець Роман Богданович – д-р техн. наук, доцент, завідувач кафедри спеціалізованих комп'ютерних систем Національного університету „Львівська політехніка", Львів, Україна, e-mail: dunets@polynet.lviv.ua.

Тиханський Дмитро Ярославович – аспірант кафедри спеціалізованих комп'ютерних систем Національного університету „Львівська політехніка", Львів, Україна, e-mail: tyxdima1@gmail.com.