УДК 004.713, 004.27

Д.Я. Тиханський, Р.Б. Дунець, Р.В. Грица
Національний університет “Львівська політехніка”,
кафедра спеціалізованих комп’ютерних систем

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЧАСТКОВО РЕКОНФІГУРОВАНОЇ СИСТЕМИ

Описано метод підвищення ефективності частково реконфігурованої системи на кристалі у разі зміни конфігурації, який ґрунтується на використанні тільки можливостей кристала. Також порівняні розмір конфігурації і час реконфігурації під час використання запропонованої системи.

Ключові слова: ефективність, частково реконфігурована система.

Described a method of improving the efficiency of partially Reconfigurable systems on chip by changing the configuration, which is based on using only features. Also done comparing the size of configuration and reconfiguration time by using the proposed system.

Key words: efficiency, partially reconfigurable systems.

Постановка проблеми

Більшість сучасних, поданих на ринку, реконфігурованих комп’ютерів засновані на Програмованих логічних інтегральних схемах (ПЛІС). Обіцяна особливість ПЛІС – це можливість використовувати спільні апаратні ресурси для різних задач, на різних стадіях виконання програми. Більше того, задачі можуть перемикатись “на льоту” доки решта апаратури продовжує працювати. Ця особливість має назву: динамічна реконфігурація, і оцінка її швидкодії надасть цікавий шлях для подальших досліджень. У роботі досліджено швидкість реконфігурації і запропоновано систему для реконфігурації, котра не потребує жодних додаткових ресурсів, окрім ресурсів котрі перебувають на ПЛІС. Це дослідження буде корисним у разі використання його результатів і підходів у розробці майбутніх реконфігурованих систем.

Вступ

ПЛІС – це інтегральні схеми, котрі складаються з великої нейтральної матриці програмованих логічних елементів і зв’язків, що може бути сконфігурована для виконання цифрової схеми. Більшість ПЛІС базуються на технології SRAM, це означає, що біти SRAM під’єднані до конфігураційних точок на кристалі, і програмування SRAM бітів конфігурує ПЛІС. ПЛІС можуть бути налаштовані під поточну задачу і вихідна продуктивність задачі на ПЛІС буде вища, ніж програмна продуктивність, зберігаючи при цьому адаптивність до змін. Різні сфери задач мають свою реалізацію на ПЛІС, починаючи з задач біоінформатики на найсучасніших системах [1] і закінчуючи детекторами руху на найдешевших системах [2].

Однією з дуже цікавих можливостей деяких ПЛІС є динамічна реконфігурація [3], і це дослідження показує, що в частковій реконфігурації є багато цікавих моментів [4], [5]. Ця властивість дає змогу змінювати модуль, котрий розташований на деякій частині пристрою, доки решта пристрою продовжує виконувати свою задачу. Різні апаратні модулі можуть поділяти в часі спільні фізичні ресурси, і апаратура може адаптуватись під програмні задачі “на льоту”, або ж до частини задачі. Це дозволяє зменшувати пристрої шляхом зменшення їх вартості, розмірів, споживаної потужності, і ефективнішого використання місця на платі.

Багато задач отримують перевагу, якщо їх реалізовувати динамічно. У мережевому домені Качріс та ін. [6] запропонували динамічно реконфігурований процесор для досягнення вимог мережевого навантаження. Модулі подібні до шифрування, стиску, виявлення проникнень, котрі

використовуються в сучасних маршрутизаторах обробки інформації, є динамічно змінними відповідно до розпоподілу мережевого потоку. До прикладу для програмного радіозв'язку випущений набір моделювання, котрий використовує часткову реконфігурацію для підтримки різних хвиль зв'язку і протоколів, в одному пристрої [4]. Це дає змогу еластично і ефективно зв'язуватись з пристроями, котрі відрізняються виробником, радіочастотою або протоколом обміну. Також динамічна реконфігурація використовується для підтримки високоенергетичних фізичних досліджень у Великому адронному колайдері Європейського центру ядерних досліджень[5]. Існують деякі системи, котрі використовують часткову реконфігурацію для розробки комерційних програм. Але в таких системах може погіршуватись час виконання, через час, котрий необхідний для завантаження конфігураційних даних перед тим, коли система стане готовою до виконання. Цей час називається: час реконфігурації, і кількісний аналіз, необхідний для визначення коли динамічна реконфігурація є необхідною для поточної задачі. Також цей аналіз можна використовувати для оцінювання запропонованих механізмів для зменшення часу реконфігурації, завдяки кешуванню, стисненню і передбаченню [7, 8].

У [9] вже було проведено поверхневий експеримент з часткової реконфігурації, ця робота є своєрідним продовженням попередньої, але з використанням новітньої архітектури ПЛІС і розширеними задачами.

Мета дослідження

Основні завдання,які закладені в цю роботу, такі:

- методологія для оцінки динамічної реконфігурації ПЛІС з перспективи системи;
- структура для швидкого налаштування системних параметрів і опрацювання експериментальних даних;
- формування всіх часових компонентів, які становлять час реконфігурації;
- результати, котрі показують різницю в часі реконфігурації через зміну системних параметрів

Аналіз останніх досліджень та публікацій і оцінка реконфігурації

МкГрегор і Лісахт оцінили самоконтролюючу динамічно реконфігуровану систему, використовуючи логічний аналізатор, і зробили висновок, що процес реконфігурації значно повільніший, ніж швидкість роботи логіки на ПЛІС [15]. У [16], МкКей і Сінгх розробили інструменти і техніки для налагодження динамічно реконфігурованої системи. Логічний аналізатор використовується для оцінки покращення спеціалізованих схем, таких як константний коефіцієнт помножувача над відповідними загальними схемами. Тен та ін. [17] порівняли продуктивність двох інтерфейсів для часткової реконфігурації ПЛІС, для оцінки компромісів між степенем інтеграції, додаткових ресурсів, реконфігураційною еластичністю і затримкою реконфігурації. Вони використовували інструмент Xilinx Chipscope Pro, як внутрішній логічний аналізатор [18]. У [19] Гаймел та ін. вивчали покращення ефективності в часі і використанні ресурсів в новому методі розробки частково реконфігурованих проектів на Xilinx Virtex-4 через віддалене оновлення. Зазначені вище праці показують, що оцінка ефективності динамічної реконфігурації є цікавим полем для досліджень. Також наявні інструменти не підтримують симуляції динамічної реконфігурації, через відсутність поведінкових і апаратних моделей. Вищезгадані роботи використовують логічний аналізатор для вимірювання часу реконфігурації для кількох бітів, або для оцінювання спеціалізованих схем із загальними схемами, або для порівняння інтерфейсів для часткової реконфігурації. Тому метою цієї роботи є розробка і перевірка моделі для реконфігурування ПЛІС з найбільш можливою швидкодією. Ця робота дозволить при майбутніх розробках реконфігурованих систем, рахувати максимальну теоретично можливу швидкодію, на яку спроможна майбутня система. Адже однією з найбільших проблем реконфігурованих систем є саме час реконфігурації. І система доти не може отримати нові можливості, котрі надає завантажуваний модуль, доки не пройде повний час реконфігурації для цього модуля. Другою метою цієї роботи було створення і дослідження "незалежної" системи реконфігурації. Тобто зробити ПЛІС максимально незалежною від зовнішніх пристроїв і інтерфейсів. По-перше, система

на ПЛІС має сама визначати, коли необхідно проводити реконфігурацію, по-друге, за можливості, система на ПЛІС має використовувати якнайменше зовнішніх, по відношенню до неї, пристроїв та інтерфейсів. Використання цих підходів дозволить знижувати вартість і час розробки майбутніх реконфігурованих систем на ПЛІС. Звісно, "незалежна" динамічно реконфігурована система, за можливості, має працювати з максимально можливою швидкодією. Відповідно, у цій роботі поставлена мета зробити і дослідити "незалежну" реконфігуровану систему на ПЛІС, котра працюватиме з найбільшою ефективністю.

Virtex-5 FX і реконфігурація

Загальна структура ПЛІС Virtex-5 FX показана на рис. 1. Вона складається з апаратного процесора Power PC, високопродуктивної шини PLB(Processor Local Bus), повільнішої шини OPB (On-Chip Peripheral Bus) [11] для комунікації з матрицею. Модулі, котрі розташовані в матриці логічних елементів, можуть виступати як периферія процесора. Матриця являє собою двовимірну дрібнозернисту структуру, котра переважно складається з конфігураційних логічних блоків CLB (configurable logic blocks) , апаратних блоків пам'яті (BRAM) і апаратних перемножувачів. Кожний конфігураційний блок (CLB) складається з таблиць істинності LUT (look up table), регістрів, мультиплексорів і вентилів, які конфігуруються для виконання логіки проекту. Матриця може бути сконфігурована апаратним процесором, або спеціалізованим автоматом, використовуючи спеціальні інструкції за допомогою внутрішнього конфігураційного порта ICAP (Internal Configuration Access Port). Цей порт може бути сконфігурований на ширину шини 8, 16, 32-біт, і опрацьовувати дані на максимальній частоті 100МГц.

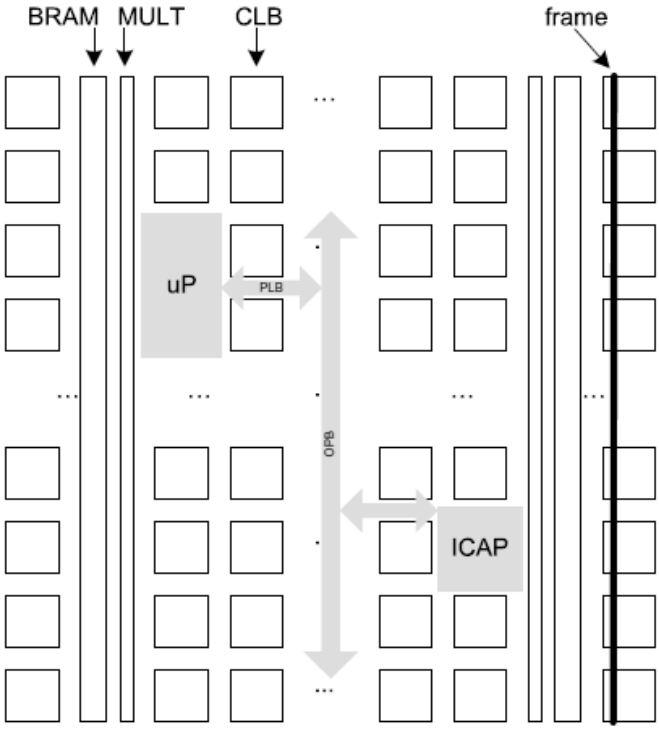

Рис. 1. Загальна структура ПЛІС Virtex-5 FX

Конфігураційна пам'ять ПЛІС Virtex-5 побудована з так званих фреймів, які покривають весь пристрій. Ці фрейми є найменшими адресованими сегментами в конфігураційній пам'яті Virtex-5. Відповідно всі операції конфігурування мають проходити над всіма конфігураційними фреймами. Не можна безпосередньо звертатись до одного елемента апаратури для конфігурування [10]. Додатковий фрейм скиду необхідно додавати в кінці конфігураційних даних, який скидає реконфігураційний конвеєр для наступних фреймів, котрі можуть бути завантажені. Тому для успішного завантаження одного фрейму в пристрій, необхідно завантажити два фрейми, фрейм даних і фрейм скиду.

Конфігураційні дані створені для програмування ПЛІС називаються бітовим потоком (bitstream). Коли створюються дані для конфігурування порції ПЛІС, тоді такий набір даних називається частковим бітовим потоком (partial bitstream). У старших ПЛІС від Xilinx і САПР, для створення частково реконфігурованих модулів використовували техніку, яка базувалась на малих відмінностях в модулях. У проект вносились незначні зміни і тоді створювались бітові потоки, базовані на відмінностях в проекті [12]. Але сучасні ПЛІС і САПР підтримують тільки модульну часткову реконфігурацію [13]. Ця робота виконана в ідеології модульної часткової реконфігурації, оскільки метод часткової реконфігурації, заснований на відмінностях, вважають застарілим.

Налаштування експерименту платформа ML507

Система для експериментів, показана на рис. 2, складається з платформи ML507 з ПЛІС Virtex-5 FX70 на борту [21]. Платформа під'єднана до комп'ютера через USB-JTAG кабель. JTAG інтерфейсу цілком достатньо для завантаження повної і часткової прошивки ПЛІС і для перевірки внутрішніх сигналів через логічний аналізатор. На рис. 2 показано тільки ті частини платформи ML507, котрі використовуються для експерименту. Використання кнопки на платформі дозволяє користувачу виконати реконфігурацію у будь-який момент часу. Два світлових індикатори показують роботу статичної частини ПЛІС. По роботі решти двох індикаторів можна побачити роботу реконфігурованих модулів.

Система на ПЛІС

Внутрішня система на ПЛІС показана на рис. 3. У цій системі основним модулем є модуль позначений як Reconfigure Control, він контролює весь процес реконфігурації. Модуль під назвою Stactic Module є статичним і виводить результат роботи дворозрядного лічильника на світлові індикатори. Присутність статичного модуля показує, що ПЛІС продовжує працювати під час реконфігурації. Пристрій під назвою reconfigurable module є частково реконфігурованим лічильником, частота якого змінюється залежно від конфігурації. Натиск на відповідну кнопку на платі ML507 змушує модуль Reconfigure control вибирати часткову конфігурацію з пам'яті Reconfigurable memory module і безпосередньо передати її на на модуль ICAP. Розмір пам'яті має бути не менший, ніж розмір бітової послідовності, котра створена відповідним програмним забезпеченням від Xilinx. Вибірка з пам'яті відбувається словами, і процес вибірки завершується тоді, коли буде вибране останнє слово часткової прошивки з пам'яті. Також у системі на ПЛІС присутня додаткова логіка для Логічного аналізатора, котра дозволяє зчитувати необхідні сигнали ПЛІС і передавати їх через JTAG на комп'ютер.

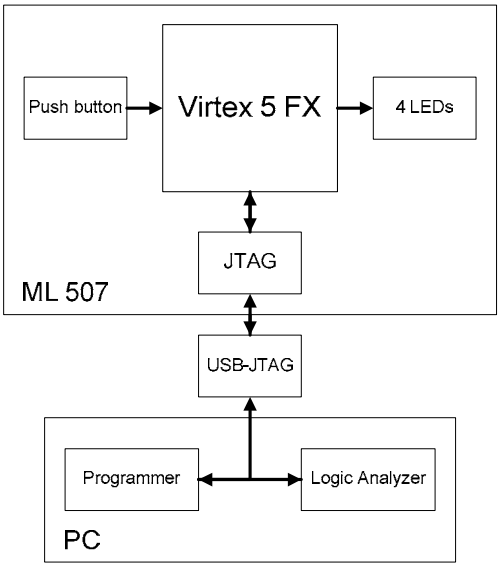

Рис. 2. Загальна структура системи

Створення часткових послідовностей

У загальному випадку під час створення часткових послідовностей ми користувались модульним підходом до проектування проектів з частково реконфігурованими модулями [13]. Ми реалізували логічну функцію, котра відповідає роботі простого лічильника, як частково реконфігурований модуль, котрий використовує таблиці істинності LUT з одного стовпця на ПЛІС. Для цього одного реконфігурованого лічильника зроблено два лічильника різної частоти. Тобто, можна під час роботи ПЛІС завантажувати другий реконфігурований лічильник з пам'яті Reconfigurable memory module. Перший реконфігурований лічильник завантажується разом зі всією прошивкою на ПЛІС. При розводці ПЛІС можна вибрати, який фізичний розмір на ПЛІС займатиме реконфігурований модуль. Але який би ми розмір не вибрали, реконфігурований модуль все одно займатиме розмір, кратний одному стовпцю на ПЛІС, також будучи кратним одному конфігураційному фрейму. У таблиці показана відповідність розміру часткової прошивки до часу реконфігурації.

Зрозуміло, що переваги часткової реконфігурації ідуть разом з ціною пам'яті де часткові конфігурації необхідно зберігати. Часткові конфігурації можна зберігати в зовнішній енергонезалежній пам'яті або у внутрішній пам'яті ПЛІС. Якщо враховувати співвідношення внутрішньої площі кристала до ціни, то кращим виходом буде зберігати неактивні бітові послідовності в зовнішній енергонезалежній пам'яті [22]. Також можна використовувати внутрішню пам'ять ПЛІС BRAM, для зберігання часткових послідовностей [20], але це накладає обмеження на розмір і кількість послідовностей, котрі можуть бути збережені. У нашому експерименті ми використали внутрішню пам'ять ПЛІС, оскільки вона забезпечує необхідну швидкодію вибірки даних, і для її реалізації необхідно затрачати менше часу, оскільки її інтерфейс з рештою ПЛІС є простіший, ніж у зовнішньої пам'яті.

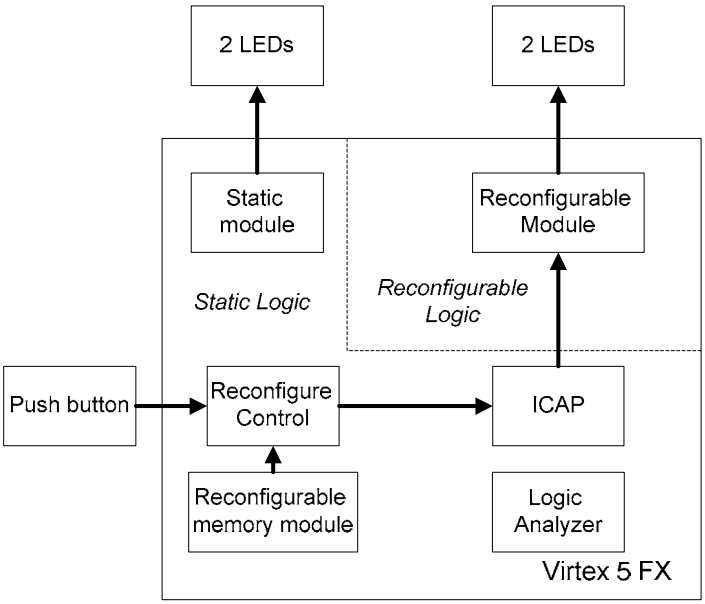

Рис. 3. Внутрішня система на ПЛІС

Демонстрація часткової реконфігурації

На одну область ПЛІС припадає два реконфігурованих модулі. Ці два модулі являють собою два лічильники, котрі відрізняються довжиною лічильного регістра. Виводи кожного з реконфігурованих модулів під'єднанні до найстарших виводів лічильного регістра. Оскільки довжина лічильного регістра в кожному з двох модулів різна, тоді світлодіоди, котрі під'єднанні до виводів реконфігурованої області, будуть блимати з різною частотою, залежно від конфігурації.

Під час всієї роботи системи, коли натиснути кнопку реконфігурації відбудеться завантаження часткової послідовності в ПЛІС. Під час завантаження часткової послідовності реконфігурований лічильник поміняє свою логіку роботи "на льоту", доки решта системи залишається

виконувати свою роботу. Ця операція виглядає простою, але в ній ми досягнули поставленої мети, а саме: зробити систему на ПЛІС “самореконфігурованою”, незалежною, і найбільш швидкодіючою. Реальні дані, котрі ми отримали з експерименту, зображені в табиці.

Результати експерименту

Результати експеременту наведені в таблиці. Як видно з таблиці, було проведено 10 експериментів, з різними розмірами прошивок. У таблиці присутні такі поля:

- Розмір прошивки в Plan Ahead – це попередній розмір прошивки, який обчислює САПР наперед.
- Реальний розмір прошивки – це розмір бінарного файла, який було отримано після розводки проекту, цей файл безпосередньо використовується для часткової реконфігурації, як файл реконфігурованого модуля.
- Час прошивання – час від початку реконфігурації до її завершення, тобто реконфігурований модуль почав виконувати корисну роботу.

Розміри часткових прошивок і час їх прошивання

Розмір прошивки в Plan Ahead (байт)	Реальний розмір прошивки (байт)	Час прошивання (такт)	Час прошивання (мс)
5904	6632	1661	0.01661
11808	12540	3138	0.03128
17712	18444	4614	0.04614
23616	24348	6090	0.06090
47232	48148	12040	0.12040
70848	71948	17990	0.17990
94464	95748	23940	0.23940
118080	119548	29890	0.29890
141696	143348	35840	0.35840
165312	167148	41790	0.41790

Як видно з даних таблиці реальний розмір прошивки, який ми отримуємо після трасування проекту, відрізняється від розміру прошивки, який обчислює система наперед. Також не було виявлено пропорційної залежності між реальним і передбаченим розміром прошивки. Чим більший розмір передбаченої прошивки, тим на більшу кількість байт відрізняється розмір реальної прошивки. Час прошивання можна чітко визначити з розміру прошивки, як зображено у формулі

$$Час\ реконфігурації\,(мс) = \left(\frac{Розмір\ файла\ (байт)}{4} + 3 \right) \cdot 10^{-5}.$$

Як видно з формули, час реконфігурації залежить майже цілком від розміру прошивки реконфігурованого модуля. Тобто час реконфігурації залежить тільки від швидкодії інтерфейсу, через який виконується реконфігурація. Для налаштування інтерфейсу реконфігурування необхідно всього три такти. Як показали експерименти, після реконфігурування система починає працювати одразу, тобто на наступний такт після відсилання останнього слова з прошивки реконфігурованого модуля.

Висновки

Метою роботи було проведення незалежної внутрішньої часткової реконфігурації. І також оцінка часових меж реконфігурації. Поставлені завдання були виконані. У поставленому експерименті було проведено реконфігурування на максимальній швидкодії 3.2 Гбіт/с для ПЛІС сімейства Virtex-5[13]. Також було розроблено механізм для незалежної реконфігурації ПЛІС, тобто ПЛІС сама визначає коли провести реконфігурацію, і сама виконує реконфігурацію без допомоги зовнішніх пристроїв. Було оцінено час проведення реконфігурації на максимальній швидкодії з мінімальною кількістю затримок. Встановлено, що мінімальний час реконфігурації для ПЛІС сімейства Virtex-5 не може бути нижчим, ніж 0.016 мс для найменшого модуля, котрий займає 0.36

% Virtex-5 FX70 і складається з 160 логічних елементів. Час реконфігурації для великого модуля, який займає 10 % логічних елементів ПЛІС Virtex-5 FX70, становить 0.42 мс. Наведені часові затримки є ідеальними, тобто показують час реконфігурації, зменшити який неможливо, натомість час реконфігурації в реальних системах може бути збільшений за рахунок необхідності додаткових перевірок. Відповідно, під час розроблення майбутніх реконфігурованих систем, результати цієї роботи можуть істотно допомогти при виборі методу реконфігурації і дати часову оцінку реконфігурації у майбутніх системах.

1. Afratis P., Sotiriades E., Chrysos G., Fytraki S. and Pnevmatikatos D. A Rate-based Prefiltering Approach to BLAST Acceleration, in Proceedings of the International Conference on Field Programmable Logic and Applications , 2008. 2. Papademetriou K., Dollas A. and Sotiropoulos S. Low Cost Real-Time 2-D Motion Detection based on Reconfigurable Computing // IEEE Transactions on Instrumentation and Measurement, vol. 55, no. 6, pp. 2234–2243, Dec. 2006. 3. Compton K. and Hauck S. Reconfigurable Computing: A Survey of Systems and Software // ACM Computing Surveys ,vol. 34, no. 2, pp. 171–210, June 2002. 4. Xilinx Inc. Press Release: ISR and Xilinx Roll Out Ready-to-Wear SDR. Xilinx Inc., San Jose, CA., 2006, www.fpga journal.com. 5. Programmable Logic Design Line. Xilinx honored for enabling technology in the ALICE experiment at CERN. Xilinx Inc., San Jose, CA., 2008, http://www.pldesignline.com/news/207101017. 6. Kachris C. and Vassiliadis S. Performance Evaluation of an Adaptive FPGA for Network applications in Proceedings of the 17th IEEE International Workshop on Rapid System Prototyping, 2006. 7. Li Z. Configuration Management Techniques for Reconfigurable Computing // PhD thesis, Northwestern University, 2002. 8. Papadimitriou K. and Dollas A. Performance Evaluation of a Preloading Model in Dynamically Reconfigurable Processors // in Proceedings of the International Conference on Field Programmable Logic and Applications , August 2006, pp. 901–904. 9. Дунець Р.Б. Проблеми побудови частково реконфігурованих систем на ПЛІС / Р.Б. Дунець, Д.Я. Тиханський // Радіоелектронні і комп'ютерні системи. – № 7(48). – 2010, С. 200–204. [10] Xilinx Inc. Virtex-5 FPGA Configuration User Guide. Datasheet. UG191 (v3.9.1) August 20, 2010" 2010. 11. IBM, http://www.chips.ibm.com/products/coreconnect, 2007. 12. Xilinx Inc. Difference Based Partial Reconfiguration. XAPP290 (v2.0) December 3, 2007" 2007. 13. Xilinx Inc. Partial Reconfiguration User Guide Datasheet. UG702 (v13.1) March 1, 2011" 2011. 14. Дунець Р.Б. Дослідження часткової реконфігурації ПЛІС / Р.Б. Дунець, Д.Я. Тиханський // Радіоелектронні і комп'ютерні системи. – № 6(40). – 2009, С. 24–-244. 15. McGregor G. and Lysaght P. Self Controlling Dynamic Reconfiguration: A Case Study, in Proceedings of the International Conference on Field Programmable Logic and Applications, 1999. 16. McKay N. and Singh S. Debugging Techniques for Dynamically Reconfigurable Hardware, in Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, 1999. 17. Tan H., DeMara R.F., Thakkar A.J., Ejnioui A. and Sattler J.D. Complexity and Performance Evaluation of Two Partial Reconfiguration Interfaces on FPGAs: a Case Study, in Proceedings of the ERSA, 2006. 18. Xilinx Inc., http://www.xilinx.com/tools/cspro.htm, 2011. 19. Hymel R., George A.F. and Lam H. Evaluating Partial Reconfiguration for Embedded FPGA Applications Interfaces, in Proceedings of the High Performance Embedded Computing Workshop, 2007. 20. Blodget B., James-Roxby P., Keller E., McMillan S. and Sundararajan P. A Self-reconfiguring Platform, in Proceedings of the International Conference on Field Programmable Logic and Applications, 2003, pp. 565–574. 21. Xilinx Inc., http://www.xilinx.com/products/boards-and-kits/HW-V5-ML507-UNI-G.htm, 2011. 22. MacBeth J. and Lysaght P. Dynamically Reconfigurable Cores, in Proceedings of the International Workshop on Field Programmable Logic and Applications, 2001.