УДК 519.6:512.972, 004.27

Ю.О. Кулаков, І.А. Клименко, м. Київ

ФОРМАЛІЗАЦІЯ МЕТОДУ ПОВТОРНОГО ВИКОРИСТАННЯ АПАРАТНИХ РЕСУРСІВ ФУНКЦІОНАЛЬНИХ БЛОКІВ РЕКОНФІГУРОВАНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Abstract. The formalization of the reusing hardware resources method of function blocks of Reconfigurable Computing Systems is developed, which provides a formal assessment of the proposed method intensive performance reconfiguration by removing overhead part-time reconfiguration.

Актуальність

Актуальні тенденції підвищення продуктивності реконфігурованих суперкомп'ютерів (*High-Performance Reconfigurable Computers, HPRC*) базуються на застосуванні технології часткової динамічної реконфігурації [1, 2, 3]. Це сучасна технологія, що надає широкий ряд нових можливостей підвищення гнучкості архітектури обчислювальних систем, адаптуючи її до вимог вирішуваних задач в динамічному режимі, тобто без зупинення роботи системи (*RunTime*) [4, 5]. При цьому виникає негативний фактор, що впливає на загальну продуктивність обчислень, яким є накладні витрати реконфігурації. Найбільш критичною їх складовою є час, витрачений на здійснення реконфігурації. Сучасні технології провідних виробників ПЛІС пропонують убудовані механізми зниження затримок реконфігурації за рахунок часткової динамічної реконфігурації [5] та засобів внутрішнього конфігурування мікросхем [5, 6], що базуються на застосуванні внутрішньої пам’яті кристалу ПЛІС, об’єм якої є доволі обмеженим в контексті реалізації масштабних реконфігурованих обчислень. Залучення же зовнішньої пам’яті для збереження конфігураційних даних спричинює додаткові часові витрати на здійснення реконфігурації. Вирішення означених проблем є важливим аспектом в сфері високопродуктивних реконфігурованих обчислень, що обумовлює актуальність та доцільність проведених досліджень.

Огляд відомих рішень

Найбільш розповсюдженим методом зменшення накладних витрат реконфігурації в реконфігурованих суперкомп’ютерах є повторне використання обчислювальних ресурсів [2, 3, 4]. Більшість сучасних досліджень в галузі підвищення продуктивності реконфігурованих обчислень спрямовані на розробки алгоритмів планування і розміщення задач, що водночас реалізують різні методи зменшення витрат реконфігурації. Підтримка засобів зменшення накладних витрат алгоритмами планування та розміщення задачпідвищують їх обчислювальну складність та витрачають продуктивність програмно-апаратної надбудови операційної системи [3], спричинюючи додаткові накладні витрати реконфігурації. Окрім того,

переважна кількість оглянутих розробок базуються на програмному або експериментальному моделюванні процесу реконфігурації, часто абстраговано від фізичної сторони реалізації [2, 3, 4, 7]. Інші розглядають доволі загальну архітектурну модель обчислювальної системи, ігноруючи її реальні структурні особливості [3].

Постановка задачі

Процес реконфігурації обчислювальної структури містить наступні складові: управління логічною та фізичною послідовностями реконфігураційного процесу, час, витрачений на забезпечення ($T_{control}$) якої обумовлює втрати продуктивності; передавання конфігураційних даних – витрачений час (T_{comm}) обумовлює комунікаційні витрати, прошивання мікросхеми – витрачений час (T_{mapp}) залежить від технологічних особливостей фізичного рівня реконфігурованої структури. Тоді час реконфігурації оцінюється наступним виразом:

$$T_{reconf} = T_{control} + T_{comm} + T_{mapp}.$$

Для зменшення часу реконфігурації природньо стає задача зменшення часу виконання її складових, які більшим чином належать до непродуктивних складових часу. Для зменшення комунікаційних витрат ефективним методом є повторне використання апаратних ресурсів [2], вже колись зконфігурованих на поверхні реконфігурованої області ПЛІС. Найбільш ефективною парадигмою відомих реалізації цього методу є вбудовування його в алгоритми планування та розподілу задач, які зазвичай реалізуються на рівні програмного шару операційної системи. Це, в свою чергу, призводить до додаткових втрат продуктивності, що зменшує ефективність відомих реалізацій, та обґрунтовує необхідність розробки нових спеціальних методів управління реконфігурацією з підтримкою механізмів зменшення накладних витрат.

Таким чином, основною формальною задачею є виконання наступного положення

$$T_{reconf} = \min(T_{control}) + \min(T_{comm}) + T_{mapp},$$

рішення якої забезпечить прискорення реконфігурації, при цьому інтенсивність прискорення реконфігурації знаходиться в прямій залежності від кількості видалених непродуктивних витрат.

Розробка формальної моделі методу повторного використання апаратних ресурсів функціональних блоків

Нехай виконувана програма у вихідному стані подана у вигляді ациклічного графу *G*, в вершинах якого розміщуються обчислювальні задачі. Конфігурація та виконання чергової обчислювальної задачі може відбуватися за різними послідовностями дій в залежності від місця знаходження конфігураційних даних її апаратної реалізації (апаратної задачі) і тривати відповідний до цього час:

Послідовність I. Апаратна задача $j \mid j = \overline{1, M}$ (де M – кількість апаратних задач в БК) знаходиться у заздалегідь зконфігурованому вигляді на поверхні реконфігурованої області. Час виконання задачі дорівнює

$$T^{\mathrm{I}}_{sum\ j} = T_j\,, \tag{1}$$

де T_j – час виконання апаратної задачі на поверхні реконфігурованої області, включаючи процеси вводу вихідних даних та виводу результатів. За цією послідовністю $T_{reconf\ j} = 0$.

Послідовність II. Конфігураційні дані апаратної задачі *j* знаходиться в ЗПК. Час виконання задачі дорівнює

$$T^{\mathrm{II}}_{sum\ j} = T_{reconf\ j} + T_j = T_{comm\ j} + T_j\,, \tag{2}$$

де $T_{comm\ j}$ – час пошуку й передавання конфігураційних даних із локальної ЗПК та прошивання ПЛІС, який вимірюється пропускною здатністю інтерфейсів вводу/виводу ПЛІС та швидкодією ЗПК і внутримодульного комунікаційного середовища.

Послідовність III. Конфігураційні дані апаратної задачі *j* знаходиться в БК. Час виконання задачі дорівнює

$$T^{\mathrm{III}}_{sum\ j} = T_{reconf\ j} + T_j = (T_{comm\_net\ j}) + T_j\,, \tag{3}$$

де T_{comm_net} – час пошуку й передавання конфігураційних даних із БК на рівень обчислювального модуля мережевими засобами зв'язку та прошивання ПЛІС. При цьому $T_{comm\_net\ j} >> T_{comm\ j}$.

Загальний час виконання деякої послідовної гілки обчислень без повторного використання ресурсів апаратних задач дорівнює

$$T^b = \sum_{j=1}^{K} (P_j \cdot T^{\mathrm{III}}_{sum\ j})\ . \tag{4}$$

Запропонований метод повторного використання апаратних ресурсів функціональних блоків (МПВ) забезпечує наступний час виконання послідовної гілки алгоритму:

$$T^b_{МПВ} = \sum_{j=1}^{K} (T^{\mathrm{III}}_{sum\ j} + (P_j - 1) T^{\mathrm{I}}_{sum\ j})\ . \tag{5}$$

В виразах (4) і (5) *b* – кількість вузлів послідовної гілки алгоритму, *K* – кількість наборів однотипних апаратних задач I_j та P_j – кількість екземплярів кожної *j*-ї з них в межах даної гілки обчислень.

Зменшення часу обчислення за застосування запропонованого методу можна виміряти абсолютним приростом швидкодії $\Delta T_{\mathrm{МПВ}}$ і усередненим коефіцієнтом прискорення $\rho_{\mathrm{МПВ}}$. Перетворимо вирази (4) і (5):

$$T^b = \sum_{j=1}^{K} (P_j (T_j + \Delta T_j)) \ ,$$

$$T^b_{\text{МПВ}} = \sum_{j=1}^{K} ((T_j + \Delta T_j) + (P_j - 1)T_j) \ = \sum_{j=1}^{K} (P_j T_j + \Delta T_j) \ , \qquad (6)$$

де ΔT_j – час завантаження та конфігурування j-ї апаратної задачі з набору однотипних задач I_j. Отримаємо значення абсолютного приросщення:

$$\Delta T_{\text{МПВ}} = T^b - T^b_{\text{МПП}} = \sum_{j=1}^{K} (P_j T_j + P_j \Delta T_j) - \sum_{j=1}^{K} (P_j T_j + \Delta T_j) = \sum_{j=1}^{K} (P_j - 1)\Delta T_j$$

Складова $P_j T_j$ в виразі (6) відповідає продуктивному сумарному часу виконання всіх екземплярів апаратної задачі I_j на поверхні реконфігурованої області. Складова ΔT_j відповідає часу одноразового завантаження її конфігураційних даних. Отриманий показник $\Delta T_{\text{МПВ}}$ зображує, що метод повторного використання ресурсів апаратних задач видаляє значну частину непродуктивного часу обчислення алгоритму.

Реалізовані архітектурні вдосконалення обчислювальної системи на базі застосування багаторівневої пам'яті [8] дозволяють прискорення обчислень за рахунок зменшення складової ΔT_j в виразі (6).

На підставі визначень (1) та (3) отримаємо

$$T_j < (T^{\text{II}}_{sum\ j} = T_j + \Delta T_j^{\ -}) < (T^{\text{III}}_{sum\ j} = T_j + \Delta T_j^{\ \max}) \, . \qquad (7)$$

Зменшимо комунікаційні витрати за рахунок збереження конфігураційних файлів в локальній пам'яті обчислювального модуля:

$$\Delta T_j = \Delta T_j^{\ -} < \Delta T_j^{\ \max} \, ,$$

отримаємо наступний вираз для запропонованого методу:

$$T^b_{\text{МПВ}} = \sum_{j=1}^{K} (P_j T_j + \Delta T_j^{\ +}) \, . \qquad (8)$$

На діаграмі (рис. 1) видно збільшення швидкодії обчислень за застосування запропонованого методу відносно кількості повторів однотипних задач.

Надалі виконаємо розрахунок усередненого коефіцієнта прискорення. На підставі виразів (4) і (5) отримаємо:

$$\rho_{\text{МПВ}} = \frac{T^{\text{III}}_{sum\ j}}{T_j} = \frac{\sum_{j=1}^{K} (P_j T_j + P_j \Delta T_j)}{\sum_{j=1}^{K} (P_j T_j + \Delta T_j)} = \frac{\sum_{j=1}^{K} (T_j + \Delta T_j)}{\sum_{j=1}^{K} (T_j + \frac{\Delta T_j}{P_j})} \, .$$

Для отримання усередненого показника прискорення застосуємо сумарні

значення часу і усереднені значення кількості повторів кожної задачі:

$$\rho_{\text{МПВ}}^{Av} = \frac{\sum_{j=1}^{K}(T_j) + \sum_{j=1}^{K}(\Delta T_j)}{\sum_{j=1}^{K}(T_j) + \frac{1}{P_{Av}}\sum_{j=1}^{K}(\Delta T_j)} = \frac{T + \Delta T}{T + \frac{\Delta T}{P^{Av}}}.$$

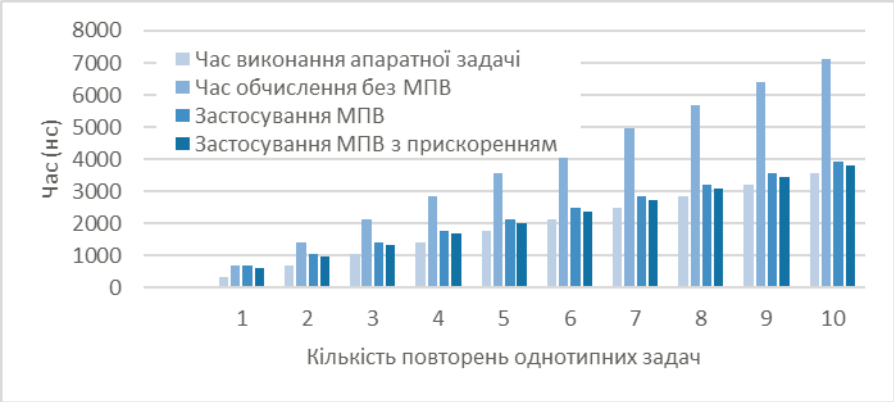

Рис. 1. Залежність швидкодії обчислень від кількості повторів однотипних задач

На діаграмі (Рис. 2) представлені результати теоретичного дослідження залежності значення коефіцієнта прискорення від кількості повторів однотипних задач. Коефіцієнт прискорення збільшується зі збільшенням кількості екземплярів задач кожного набору, при цьому ефективність використання запропонованого методу збільшується в алгоритмах, що мають високу щільність надходження нових задач.

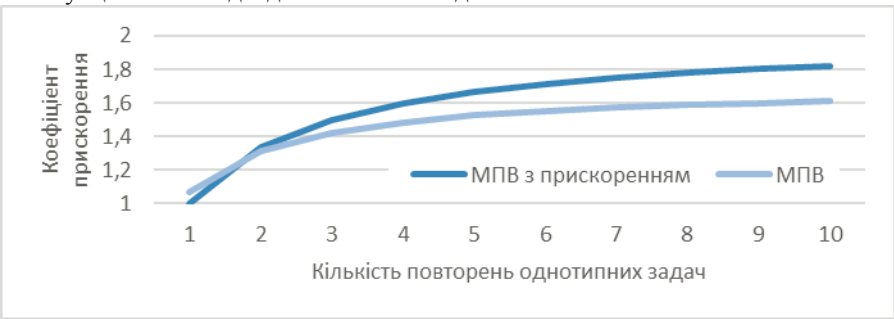

Рис. 2. Залежність коефіцієнту прискорення від кількості повторів однотипних задач

Отримані формальні визначення (5), (6), (8) методу повторного застосування ресурсів апаратних задач справедливі для будь-якого послідовного алгоритму обчислень A:

$$T_{\text{МПВ}}^{A} = \sum_{j=1}^{K} \Delta T_j + \sum_{j=1}^{K} (P_j T_j) . \qquad (9)$$

За реалізації паралельного обчислювального алгоритму вважаємо, що обчислювальна частина виразу (9) розгалужується ідеально на F функціональних блоків, а сумарний час реконфігурації є послідовною частиною, розгалуження якої неможливо зважаючи на послідовні засоби реконфігурації і інтерфейс конфігурації. Отримаємо аналітичний вираз для запису методу повторного застосування ресурсів апаратних задач для паралельного алгоритму обчислень:

$$T_{\text{МПВ}}^{\|A} = \sum_{j=1}^{K} \Delta T_j + \frac{\sum_{j=1}^{K} (P_j T_j)}{F} . \qquad (10)$$

Застосування методу повторного використання ресурсів апаратних задач максимально зменшує послідовну компоненту паралельного обчислювального процесу, яка за законом Амдала обмежує потенційну можливість зменшення часу виконання паралельного алгоритму.

Надалі врахуємо архітектурні та апаратні рішення, що базуються на архітектурі багаторівневої пам'яті, описаної в роботі [8], та запроваджують, відповідно до виразів (7) та (8), додаткові механізми зменшення послідовної компоненти в виразі (10). На рис. 3, а, б зображені відповідні часові діаграми. Застосування контролера реконфігурації [8] на локальному рівні обчислювального модуля запроваджує подальше зменшення непродуктивних витрат часу реконфігурації, що досягається за рахунок розгалуження обчислювального і реконфігураційного процесу. Відповідно до чого вираз (10) приймає наступний вигляд:

$$T_{\text{МПВ}}^{\|A} = \sum T_{comm} + \frac{\sum_{j=1}^{K} \Delta T_j + \sum_{j=1}^{K} (P_j T_{sum\ j}^{\text{I}})}{F} . \qquad (11)$$

Послідовна компонента визначається послідовністю процесів, які не можуть бути розділені у часі. Це процеси, що відбуваються під управлінням локального процесора обчислювального модуля, якими є ініціалізація контролера реконфігурації і комунікаційні процеси вводу/виводу даних апаратних задач.

За застосування ярусно-паралельної форми алгоритму G, виконання принципу реконфігурації як можна раніше, призводить практично до повного знехтування часом реконфігурації, що наближає до нуля непродуктивні реконфігураційної витрати. Пояснимо це наступним чином. Кожна готова до виконання задача, на підставі інформації щодо своїх нащадків, запускає процедури їх реконфігурації. Це переміщує всі процедури реконфігурації на попередній ярус. Виключеннями є процедури реконфігурації першого ярусу.

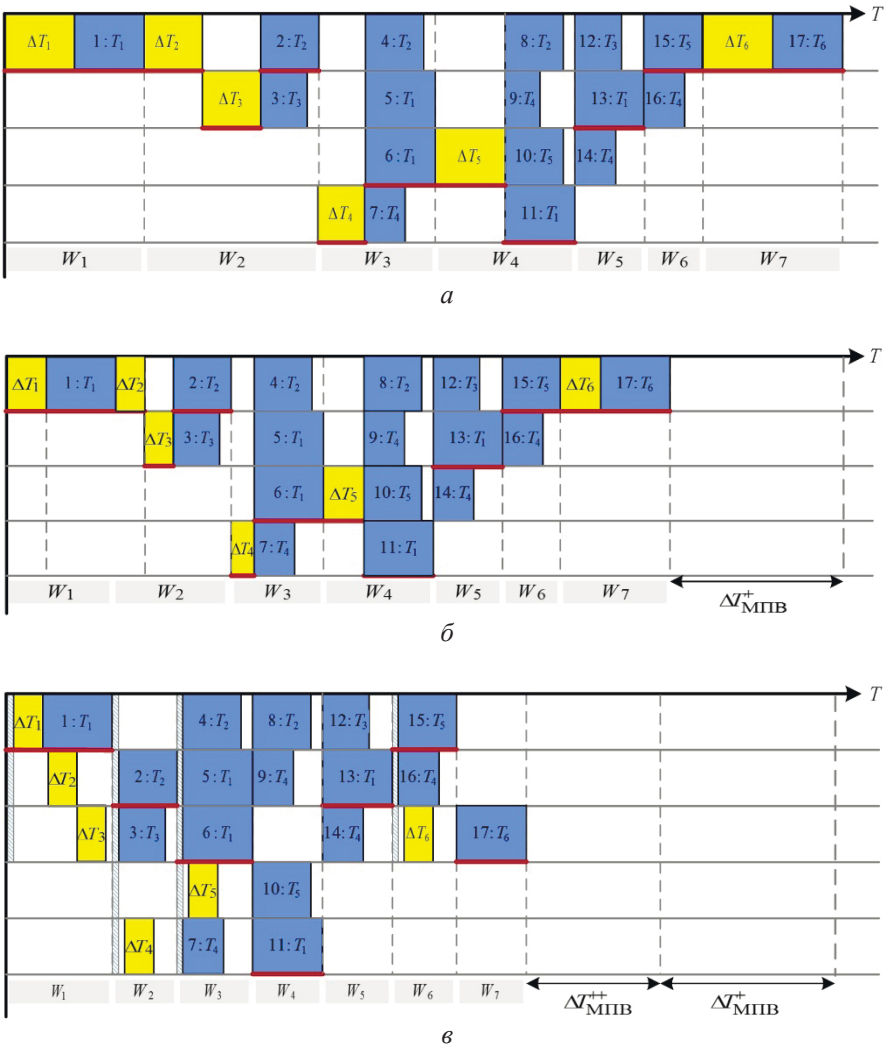

Рис. 3. Порівняльна часова оцінка методу повторного використання апаратних ресурсів функціональних блоків: *а* – без мінімізації часу передавання конфігураційних даних, *б* – з прискоренням за рахунок застосування локальної пам'яті для збереження конфігураційних даних; *в* – застосування контролера фізичної послідовності реконфігурації

Враховуючи вищесказане, а також те, що час виконання кожного ярусу визначається часом виконання його найтривалішої задачі або сумарним часом всіх процедур реконфігурацій що виконуються на ньому, якщо він більше ніш

час виконання найтривалішої задачі, із виразу (11) отримаємо наступне визначення часу обчислення ЯПФ алгоритму G за застосування методу повторного використання ресурсів апаратних задач:

$$T_{МПВ}^{G} = \sum_{k=1}^{w} T_{comm\_k} + \sum_{k=1}^{w} \max\left(\sum_{v=1}^{H_k} \Delta T_v, \{T_v \,|\, v = \overline{1, H_k}\}\right),$$

де $k = \overline{1, w}$ – номер ярусу, w – кількість ярусів обчислювального алгоритму, $v = \overline{1, H_k}$ – номер вузла на ярусі k, H_k – кількість вузлів на ярусі k. Відповідна часова діаграма зображена на рис. 3, *в*. Із діаграми видно, що метод практично видаляє непродуктивні витрати на реконфігурацію, за виключенням першого ярусу, де виконувані задачі мають бути заздалегідь зконфігурованими.

Основні аспекти практичної реалізації методу повторного використання апаратних ресурсів функціональних блоків

Метод повторного використання апаратних ресурсів функціональних блоків спрямований на ефективне управління розкладом розміщення та підтримки конфігурацій апаратних задач з врахуванням проблеми зменшення накладних витрат реконфігурації. Практична реалізації методу представлена авторами статті в роботі [8]. Для зменшення непродуктивних комунікаційних витрат часу в роботі [8] запропоновано новий спосіб збереження конфігураційних даних апаратних задач, за якого найчастіше затребувані з них зберігаються в заздалегідь зконфігурованому вигляді на поверхні реконфігурованої області ПЛІС. Окрім часового прискорення цій спосіб забезпечує зберігання обмежених ресурсів внутрішньої пам'яті ПЛІС. Апаратні задачі, що затребувані з найменшою частотою, зберігаються у вигляді конфігураційних файлів в додатковій швидкодіючій пам'яті. Для підтримки методу повторного використання апаратних ресурсів функціональних блоків запропонована багаторівнева швидкодіюча пам'ять конфігурацій, що реалізована засобами декількох прошарків кеш-пам'яті між центральною бібліотекою конфігурацій (БК) і реконфігурованою структурою мікросхеми ПЛІС [8]. Кеш-прошарок першого рівня, реалізований на борту мікросхеми ПЛІС поряд з реконфігурованою структурою, застосовується для збереження слів стану функціональних блоків апаратних задач. Другий рівень розміщується в безпосередній близькості до мікросхеми ПЛІС, що дозволяє максимально мінімізувати комунікаційні витрати, наприклад у локальній зовнішній пам'яті конфігурацій (ЗПК) обчислювального модуля. Кеш-пам'ять другого рівня застосовується для збереження слів стану і прискорення пошуку конфігураційних даних.

Процес підтримки функціонування методу розкладу зберігання конфігурацій апаратних задач здійснюється прозоро для програмного шару управління реконфігурованими ресурсами. Для зменшення навантаження на локальний процесор у склад обчислювального модуля введено контролер

реконфігурації, на який покладається реалізація фізичної послідовності реконфігурації.

Висновки

Розроблена формалізація запропонованого методу повторного використання ресурсів апаратних задач для реконфігурованих обчислювальних систем, яка досліджує кількісні оцінки видалених непродуктивних часових витрат та обґрунтовує пряму залежить інтенсивності прискорення реконфігурації від їх кількості.

Формальна модель доводить, що запропонований метод повторного використання апаратних ресурсів функціональних блоків забезпечує інтенсивне прискорення реконфігурації, за рахунок видалення всієї непродуктивної складової часу реконфігурації, за виключенням задач першого ярусу ЯПФ графу алгоритму. Це, в результаті, обґрунтовує ефективність запропонованого методу та засобів його реалізації.

1. *Huang M.* Reconfiguration and Communication-Aware Task Scheduling for High-Performance Reconfigurable Computing / M. Huang, V.K. Narayana, H. Simmler, O. Serres, T. El-Ghazawi // Transactions on Reconfigurable Technology and Systems (TRETS). – USA, NY, New York, ACM, 2010. – Vol. 3, Issue 4, Article № 20.
2. *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.
3. *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). – 2012. - P 401 – 406.
4. *Panella A.A* Design Workflow for Dynamically Reconfigurable Multi-FPGA Systems / A. Panella, F. Redaelli, F. Cancare, D. Sciuto // Published in 18th IEEE/IFIP VLSI System on Chip Conference (VLSI-SoC), (Spain, Madrid, 27-29 Sept. 2010). – 2010. – P. 114 – 119.
5. What's New in Xilinx ISE Design Suite 12 [Електронний ресурс]. – Xilinx, 2014. – http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/whatsnew.htm.
6. *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. – Volume 12, Issue 3, Article № 72. – 21 p.
7. *Pellizzoni R.* Adaptive allocation of software and hardware real-time tasks for FPGA-based embedded systems / R. Pellizzoni, M. Caccamo // Proceeding of the 12th Real-Time and Embedded Technology and Applications Symposium, 2006. – 2006 – P. 208 – 220.
8. *Кулаков Ю.О.* Організація багаторівневої пам'яті в реконфігурованих обчислювальних системах // Ю.О.Кулаков, І.А.Клименко / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна технік: Зб. Наук. Пр. – К.: Век+, 2014. - №61. – С. 18 – 26.

Поступила 28.9.2015р.