УДК 519.6:512.972, 004.27

Ю.О. Кулаков, д-р. техн. наук, **І.А. Клименко**, канд. техн. наук
Національний технічний університет України «Київський політехнічний інститут»,
проспект Перемоги, 37, г. Київ, 03056, Україна.

Метод оптимізації ярусно-паралельної форми подання задачі для реконфігурованих обчислювальних систем

В роботі запропонований метод оптимізації структури графа задачі, поданого в ярусно-паралельній формі, що враховує вимоги мінімізації часу виконання задачі та обмеження апаратних ресурсів, та дозволяє підвищити продуктивність реконфігурованих обчислювальних систем. Бібл. 7, рис. 3, табл. 1.

Ключові слова: *ярусно-паралельна форма, реконфігуровані обчислювальні системи, змішаний паралелізм, М-програма, MDG, Macro Dataflow Graphs.*

Вступ

Розпаралелювання найбільш суттєвий спосіб досягнення максимальної продуктивності обчислювальних систем (ОС). Сучасні тенденції в галузі розробки високопродуктивних ОС спрямовані на універсалізацію їх архітектури та надання надвисокої продуктивності для широкого класу задач, за рахунок технічних можливостей. У той же час сьогодні актуальна парадигма реконфігурованих обчислень, коли архітектура обчислювальних систем реконфігурується або адаптується до вимог вирішуваних задач за рахунок використання перепрограмованої елементної бази. Для реалізації паралельних обчислень в реконфігурованих обчислювальних системах (РОС) найбільш природно використання структурно-процедурної організації, яка заснована на побудові інформаційного графа задачі [4]. Однак ефективна реалізація інформаційних графів задач в сучасних високопродуктивних РОС вимагає перетворення їх у паралельно-послідовну форму. Традиційно для цього використовується ярусно-паралельна форма (ЯПФ) подання задачі [4, 5].

Подання інформаційних графів задач у ЯПФ пов'язане з рядом проблем, які негативним чином впливають на продуктивність обчислень. Найзначніші з них випливають із визначення інформаційного графу – високої деталізації операцій, значної кількості інформаційних зв’язків між операційними вершинами та ациклічності структури. В результаті чого, відповідно зростанню складності і розмірності розв'язуваної задачі, відбувається збільшення розмірів ЯПФ і кількості інформаційних зв'язків. При цьому, навіть за паралельного виконання задач самого широкого рівня графу неможливо збільшити швидкість обчислень, яка фактично визначається кількістю рівнів ЯПФ. Інша проблема в реальних системах - обмеження обчислювальних ресурсів. Особливо значні ці проблеми в РОС, де час, що припадає на непродуктивні операції, додатково збільшується за рахунок часу реконфігурації структури обчислювального модуля, відповідно до вимог розв'язуваної задачі. Апаратні ресурси обчислювальних модулів обмежені кінцевими обсягами обчислювальної площі кристалів ПЛІС, внутримодульної пам'яті і комунікаційних каналів. Таким чином, необхідність вирішення окреслених проблем зумовлює високу актуальність і перспективність даного напрямку наукових досліджень.

У статті пропонується метод оптимізації структури графа задачі, поданої в ЯПФ, що дозволяє підвищити продуктивність РОС. В якості критеріїв оптимізації приймається мінімізація часу виконання задачі і обмеження апаратних ресурсів РОС.

Огляд існуючих рішень

Багато сучасних досліджень присвячені ефективній організації паралельних обчислень заснованих на ЯПФ подання задач. Традиційним підходом є модифікація структури вихідного графа задачі з метою задоволення вимог ефективного розпаралелювання та максимальної швидкодії.

Подання інформаційного графа у вигляді графа М-програми – макро- графа потоків даних (MDG, Macro Dataflow Graphs) з вершинами, що є крупно-модульними М-задачами, і ребрами, що вказують на залежності між цими вершинами [6, 7], дозволяє найбільш адекватно структурувати граф за подання його в ЯПФ. При цьому збільшується компактність ЯПФ, досягається мінімізація операцій міжпроцесорних обмінів, локалізація дрібнозернистих обчислень і циклічних ділянок графа на рівні обчислювальних вузлів. Таке перетворення є обґрунтованим згідно архітектурних особливостей сучасних ОС, обчислю-

вальні вузли яких побудовані на базі багатоядерних процесорів або реконфігурованих обчислювальних модулів [3, 7].

В роботі [1] запропоновано метод оптимізації обчислювального алгоритму ЯПФ за критерієм максимальної швидкодії. Метод спрямований на подолання проблеми обмеження обчислювальних ресурсів і зводиться до розрахунку найбільш ранніх строків виконання операторів алгоритму і перерозподілу операторів по ярусах ЯПФ, з урахуванням довжини критичного шляху, що обумовлює мінімально можливий час виконання алгоритму. Однак практична реалізація даного методу потребує великих витрат машинного часу. В роботі [2] пропонується інший метод рішення аналогічної задачі, заснований на обчисленні перевантажень і недовантажень ярусів з подальшим перерозподілом операторів на вільні місця ЯПФ. Цей метод дає суттєвий виграш у часі вирішення завдання в порівнянні з попередньою реалізацією.

У роботі [6] запропоновано метод планування М-програм, поданих в ЯПФ, заснований на класичному алгоритмі зменшення часу вирішення задач критичного шляху [6], за рахунок виділення їм більшої кількості процесорів. Цей алгоритм перевершує за швидкодією ряд аналогічних алгоритмів класу змішаного паралелізму [6], однак, вимагає перерахунку часу виконання на кожному кроці, що робить його обчислювально складним.

Автори роботи [4] пропонують подавати інформаційний граф в послідовно-конвеєрній формі, яка дозволяє використання зворотних зв'язків і є більш компактною порівняно з ЯПФ. Макрозадачі паралельно реалізуються на незалежних конвеєрах, проте внутрішня структура макрозадачі, являє собою ЯПФ інформаційного графа з усіма її негативними проявами. Економія часу досягається за рахунок попередньої реконфігурації всіх елементів конвеєра під час його ініціалізації для вирішення чергової макрозадачі.

Постановка задачі

Основна модель задачі подана графом М-програми $G_M = (V, E, Q)$, де V – множина вершин, які є М-задачами програми; E – множина дуг, при цьому дуга $e \in E$ поєднує М-задачі M_1 и M_2, якщо між ними є співвідношення за даними або за управління; Q – множина вершин графу, як є нащадками вершин із множини вершин V. Іншими словами дуга $e \in E$ поєднує вершини V_j и Q_q, якщо між ними є відношення $V_j \rightarrow Q_g \,|\, (j = \overline{1,v}; q = \overline{1,m})$, де v – кількість вершин графу G_M, m – кількість вершин-нащадків.

Введемо також наступні позначення: w – кількість ярусів графу G_M; $W_i \,|\, (i = \overline{1,w})$ – поточний ярус; N_i – кількість вузлів на ярусі W_i; T_i – час виконання задачі поточного ярусу, що визначається часом рішення найтривалішої задачі даного ярусу; B_j – множина вершин, що складають гілку графа з початком у вершині V_j; T – час обчислення М-програми, що визначається як $T = \sum_{i=1}^{w} T_i$.

В якості критерію оптимізації графа М-програми приймаємо граничну кількість часу виконання М-задачі $T_{прип}$, що визначається характером задачі, і максимально-припустима кількість вершин $N_{прип}$ на ярусі графа W_i, що визначається обмеженнями апаратних ресурсів системи. Тоді $T \le T_{прип}$, $N_i \le N_{прип}$.

Графічно процес модифікації графа можна зобразити у вигляді прямокутника, що покриває граф, висота якого дорівнює максимальній кількості вершин на ярусі графа, а ширина дорівнює кількості ярусів. Дотримуючись критеріїв оптимізації прямокутник трансформується за рахунок зменшення висоти і збільшення ширини. Основна мета оптимізації – отримання прямокутника з мінімальною площиною, але такою що не перевищує максимально припустиму.

Розглянемо приклад оптимізації структури графа, зображеного на рис. 1. В якості критеріїв оптимізації приймемо наступні значення: $T_{прип} = 8T_i$, $N_{прип} = 3$.

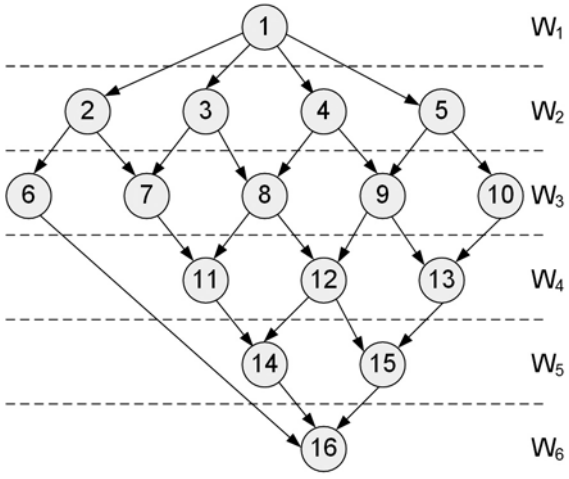

Рис. 1. Приклад графу М-програми

Графічна інтерпретація методу оптимізації, зображена на рис. 2 (а, б, в). Кількість вузлів на ярусах W_3 и W_4 вихідної структури (рис. 2, *а*) перевищує максимально-припустиме значення і, відповідно, площина вихідного покриваючого прямокутника більш ніж припустима. Перемещення вузла V_2 призведе до переміщення двох суміжних гілок $B_6 = \{V_6\}$ та $B_2 = \{V_2, V_7, V_{11}, V_{14}, V_{16}\}$ (рис. 1), які слід змістити на два рівні вниз (рис. 2, *а*, *б*) для досягнення вимог оптимізації.

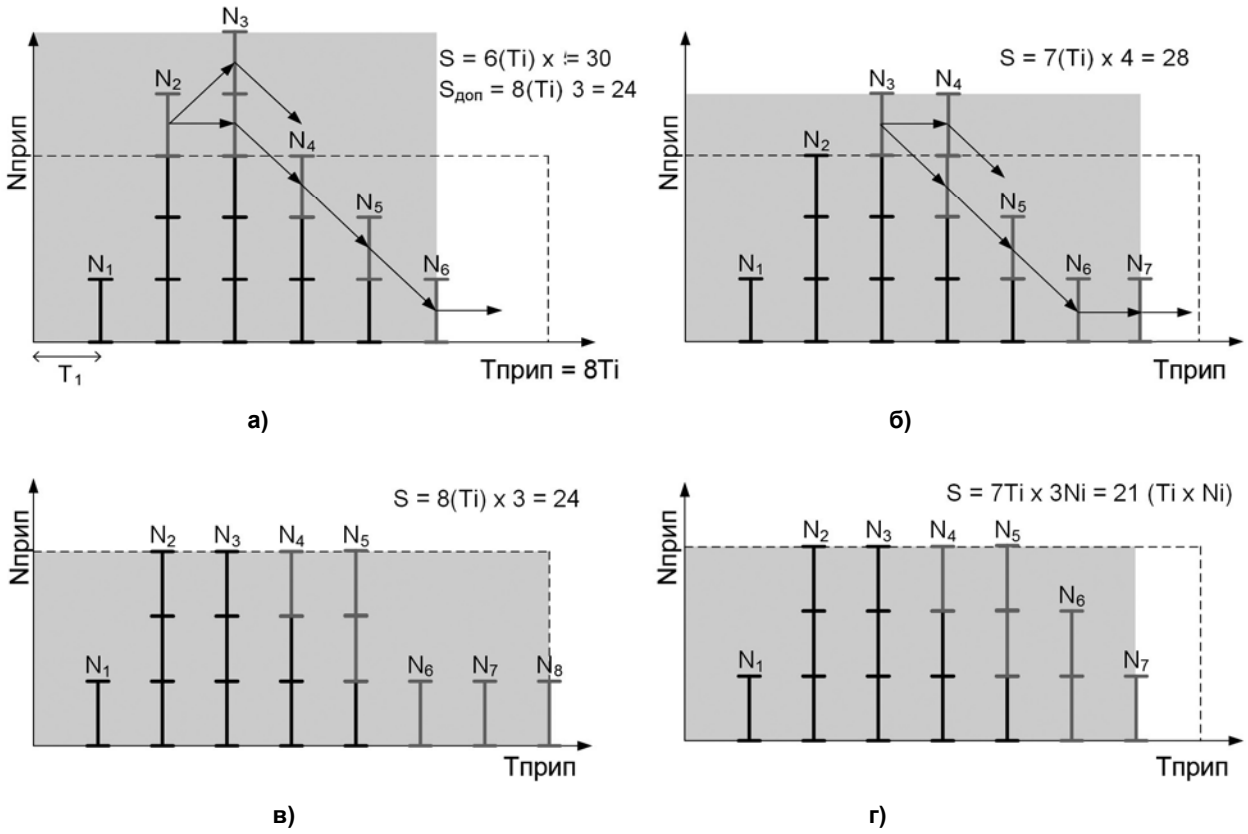

Рис. 2. Графічна інтерпретація методу оптимізації: *а*, *б*, *в* – етапи безпосереднього переміщення гілок; *г* – результат застосування алгоритму оптимізації

В результаті отримана структура вкладається в прямокутник припустимої площини (рис. 2, в) і подальша модифікація не потрібна. Таке безпосереднє переміщення зв'язаних гілок хоча і призвело до припустимої структури, однак в результаті (рис. 2, в), велика кількість ярусів недовантажена задачами, при тому, що кількість ярусів значно збільшилася, так як переміщувана гілка включає кінцевий вузол графа. Таким чином, необхідні додаткові засоби, які дозволять збільшити щільність розміщення переміщуваних вузлів на ярусах графа і забезпечити зменшення площини покриваючого прямокутника.

Ряд засобів, що забезпечують підвищення ефективності оптимізації запропоновані на рівні алгоритму реалізації методу: перегляд рівнів графа здійснюється знизу вгору, це дозволяє знизити кількість безглуздих переміщень гілок і суміжних з ними гілок, а пошук і переміщення найкоротшої гілки з мінімальною кількістю відгалужень призводить до зменшення кількості ярусів модифікованого графа. Графічна інтерпретація застосування цих засобів обґрунтовує зменшення площини покриваючого прямокутника (рис. 2, г) в порівнянні з фігурою, отриманою на рис. 2, в.

Алгоритм оптимізації

Запропонований алгоритм оптимізації має низьку обчислювальну складність, на відміну від відомого алгоритму [2], найбільш близького за своєю суттю. Ніякі розрахунки, в тому числі, щодо перевантаження або недовантаження ярусів відповідно наявним обчислювальним ресурсам не виконуються. Це пов'язано з особливостями реалізації РОС, де вирішальним фактором є необхідна кількість апаратних ресурсів для реалізації задач. У цьому контексті враховується лише припустима кількість задач на ярусі.

Приклад візуальної реалізації основних етапів алгоритму оптимізації зображений на рис. 3.

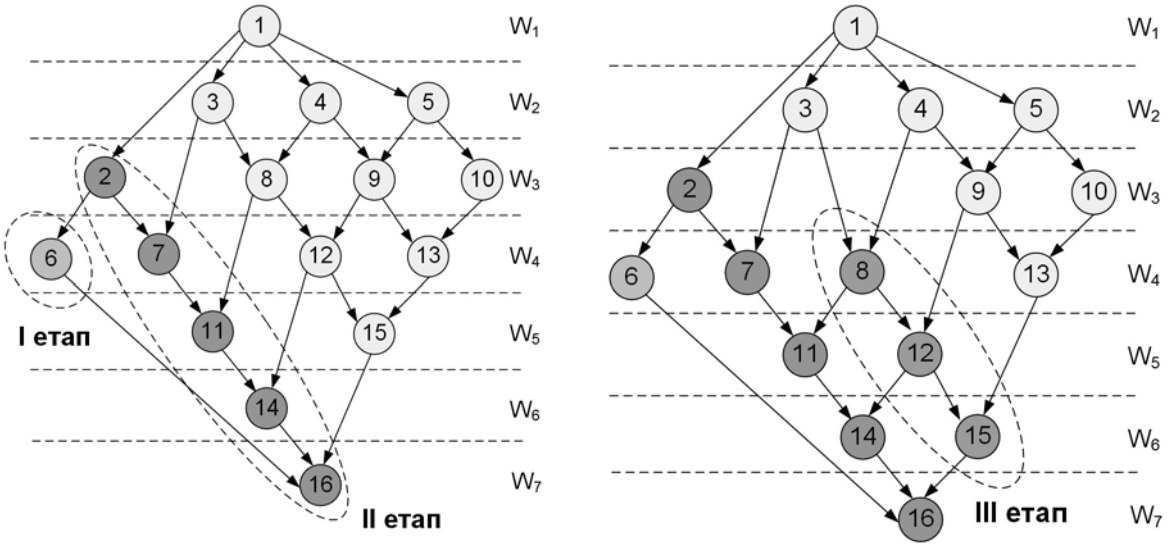

Рис. 3. Графічна ілюстрація етапів алгоритму оптимізації на прикладі графа М-задачі

Опишемо граф програми в вигляді матриці зв'язності M_G (табл. 1) розмірності $(v \times v)$. Елементи матриці $M_G[g,k] \mid (g = k)$ вказують на номер ярусу, на якому розміщений відповідний вузол $V_j \mid (j = g = k)$. На перетині рядка $M_G[g]$ і стовпця $M_G[k]$ розміщується інформація щодо потомків вузла $V_j \mid (j = g)$, де $Q_q \mid (q = k)$ – вузол-нащадок.

Таблиця 1. Матриця зв'язності

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	2	2	2											
2		2				3	3									
3			2				3	3								
4				2				3	3							
5					2				3	3						
6						3										6
7							3				4					
8								3			4	4				
9									3			4	4			
10										3			4			
11											4			5		
12												4		5	5	
13													4		5	
14														5		6
15															5	6
16																6

Основний цикл алгоритму полягає в перегляді знизу вгору ярусів графу і пошуку ярусу, на якому кількість вузлів перевищує припустиме значення. Основна процедура алгоритму `choose_move_node` – пошук такого вузла поточного рівня, переміщення якого забезпечує оптимальну, з точки зору прийнятих критеріїв, структуру графа. Щільність розміщення вузлів на ярусах графа забезпечується тим, що на кожній ітерації переміщення вузла/гілки здійснюється на один рівень вниз. Після чого виконується черговий перерахунок кількості вузлів на ярусах графа.

За основою процедурою пошуку, в першу чергу вибираються такі вузли, які мають найменшу кількість нащадків на наступному ярусі. Ідеальний варіант – відсутність нащадків, тоді вузол без наслідків переміщується на наступний ярус. В іншому випадку переміщення вузла тягне за собою переміщення всієї зв'язаної з ним гілки графа. Під гілкою в даному контексті ми маємо на увазі послідовність вузлів, розміщених на суміжних ярусах, причому останній вузол гілки – це вузол, який не має нащадків на наступному ярусі. За вибору гілки для переміщення перевага віддається найкоротшим гілкам, у склад яких не входять вузли останнього ярусу

графа. Кількість нащадків визначає кількість пов'язаних з вузлом гілок , які так само мають бути перенесені. Серед вузлів з рівною кількістю нащадків вибираються вузли з мінімальною довжиною зв'язаних гілок. Враховується подальше розгалуження гілки, перевага віддається гілкам що складається з вузлів з мінімальною кількістю нащадків, бо переміщення гілки тягне за собою переміщення пов'язаних з нею відгалужень. Далі наведений псевдокод основної процедури вибору вузла/гілки і переміщення вузлів.

```
   procedure choose_move_node;
   // вибір та переміщення вузла/гілки
на один ярус вниз
   begin
    procedure form_mass_level
    //  формування масиву вузлів Wi
ярусу, де Ni - ширина Wi ярусу
    begin
        procedure M_Graf_choice
        // перебирання елементів мат-
      рицi зв’язності,
        // g – рядок, k - стовбець
        begin
            for (j=1, j<=Ni, j++)
                if (g = = k)
                begin
                mass_Wi[j]:=M_Graph[g,k
      ]; // масив вузлів ярусу Wi
                    procedure count_Q //
      визначення кількості нащад-
      ків
                    begin
                    ...
                    return Qj;
                    end;
                mass_Qi[j]:=Qj;  //  ма-
      сив нащадків вузлів ярусу Wi
                    //   елемент   масиву
      mass_Wi[j]   має   mass_Qi[j]
      нащадків
                    // на наступом ярусі
      графа W(i+1)
                ...
                end;
      return mass_Wi, mass_Qi;
      end;

      procedure sort_mass_level
      // сортування масиву вузлів яру-
су Wi за зростанням кількості
      // нащадків, в результаті елеме-
нту масива mass_Wi[1] відповідає
      // мінімальне значення із масиву
нащадків mass_Qi[1]
      begin
      ...
      return mass_Wi;
      end;

         procedure short_branches
         // визначення найкоротшої гіл-
      ки з мінімальною кількістю відга-
      лужень
         // вузла mass_Wi[1] з міні-
      мальною   кількістю   нащадків   на
      W(i+1) ярусі
      begin
      ...
      return mass_Bi; // масив з еле-
ментами найкоротшої гілки
      end;

         procedure move_branches
         // переміщення гілки на один
      ярус вниз
         begin
          procedure M_Graf_choice
          // перебирання елементів мат-
      рицi зв’язності
          begin
          ...
          for (j=1, j<=length_mass_Bi,
          j++)
               for (g=1, g<=n, g++)
               if (M_Graph[g,j]!=0)
            M_Graph[g,j]:=
          M_Graph[g,j]+1;  //  перемі-
          щення вузла на W(i+1)
          ...
          end;
    end;

   end.
```

Висновки

Дослідження стосується реконфігурованих обчислювальних систем, які характеризуються гнучкою архітектурою, що перебудовується динамічно в процесі відображення задач із урахуванням їх вимог. Запропонований метод оптимізації структури графа поданого в ЯПФ, за критеріями мінімального часу обчислення й обмежень апаратних ресурсів. Даний метод дозволяє підвищити продуктивність РВС, за рахунок такого перерозподілу задач за рівнями ЯПФ, яке, с одного боку, дозволить ефективно й рівномірно використовувати доступні апаратні ресурси, вкладаючись в їх обмежену кількість, з іншого боку, отримати мінімально-можливий час виконання задачі. Алгоритм, реалізований на основі запропонованого методу має меншу обчислювальну складність у порівнянні з відомими алгоритмами модифікації структури графів ЯПФ [1, 2, 6], з причини відсутності розрахунків параметрів обчислювальних вузлів и задач – враховуються лише припустимі значення кількості вузлів на ярусі графа і кількості ярусів.

Список використаних джерел

1. Дронов В.В. Проблемы имитационного моделирования в режиме реального времени / В.В. Дронов // Вестник НовГУ. Серия: Естеств. и техн. науки. – №19. – 2001. – С.99 – 102.
2. Дронов В.В. Распараллеливание вычислительного алгоритма методом приоритетного спуска / В.В.Дронов, Ю.В.Иванов // Вестник НовГУ. Серия: Естеств. и техн. науки. – №22. – 2001. – С.46 – 49.
3. Клименко И.А. Метод отображения задач на реконфигурируемую архитектуру вычислительной системы / И.А. Клименко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2014. – № 59. – 159 с.
4. Левин И.И. Методы преобразования параллельных программ под конфигурацию вычислительной системы / И.И. Левин, И.М. Пономарев, А.В. Шматок // Штучний інтелект. - №3. – 2001. – С. 212 – 227.
5. Поспелов Д.А. Введение в теорию вычислительных систем / Д.А. Поспелов. – М.: Советское радио, 1972. – 280.
6. Bansal S. An improved two-step algorithm for task and data parallel scheduling in distributed memory machines / S. Bansal, P. Kumar, K. Singh // Parallel Computing. – Vol. 32, Issue 10. – 2006. – P. 759 – 774.
7. Dümmler J. Scalable computing with parallel tasks / J. Dümmler, T. Rauber, G. Rünger // Proc. on the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS '09 (Portland, Oregon, November 16) – ACM New York, NY, USA, 2009. – Article No. 9 – P. 1 – 10.

Поступила в редакцию 26 мая 2014 г.

УДК 519.6:512.972, 004.27

Ю.А. Кулаков, д-р. техн. наук., **И.А. Клименко**, канд. техн. наук

Национальный технический университет Украины «Киевский политехнический институт»,

проспект Победы, 37, г. Киев, 03056, Украина.

Метод оптимизации ярусно-параллельной формы представления задачи для реконфигурируемых вычислительных систем

В статье предложен метод оптимизации структуры графа задачи, представленного в ярусно-параллельной форме, который учитывает требования минимизации времени выполнения задачи, ограничения аппаратных ресурсов и позволяет повысить производительность реконфигурируемых вычислительных систем. Библ. 7, рис. 3, табл. 1.

Ключевые слова: *ярусно-параллельная форма, реконфигурируемые вычислительные системы, смешанный параллелизм, М-программа, MDG, Macro Dataflow Graphs.*

UDC 519.6:512.972, 004.27

Y.O. Kulakov, Dr.Sc., **I.A. Klymenko**, Ph.D.

National Technical University of Ukraine "Kyiv Polytechnic Institute",

37, Prospect Peremohy, 03056, Kyiv, Ukraine.

The optimization method of a macro dataflow graph for reconfigurable computing systems

The method of optimizing the structure of the task graph represented by a macro dataflow graph, is proposed. The method takes into account the requirement to minimize execution time and the limitations of hardware resources. The method improves the performance of reconfigurable computing systems. References 7, figures 3, table 1.

Keywords: *Macro Dataflow Graphs, MDG, reconfigurable computing systems, mixed parallelism, M-program.*

References

1. *Dronov V.V.* (2001), “Problems simulation in real time”. Vestnik NovSU. Seriya: Yestestvennye i tehnicheskie nauki. Vol. 19, pp. 99 – 102. (Rus)
2. *Dronov V.V., Ivanov Y.V.* (2001), “Parallelization of computational algorithm by lowering the priority”. Vestnik NovSU. Seriya: Yestestvennye i tehnicheskie nauki. Vol. 22, pp. 46 – 49. (Rus)
3. *Klymenko I.A.* (2014), “The method of mapping tasks to the reconfigurable architecture of the computer system”. Visnyk NTUU “KPI” Informatyka, upravlinnya ta obchyslyuvalna tehnika. Vol. 59, P. 159. (Ukr)
4. *Lyevin I.I., Ponomaryov I.M., Shmatok A.V.* (2001), “Methods for the conversion of parallel programs the computer system configuration”. Shtutchnuy intelekt. Vol. 3, pp. 212 – 227. (Rus)
5. *Pospyelov D.A.* (1972), “Introduction to Computer Systems”. M.: Sovetskoye radio, P. 280. (Rus)
6. *Bansal S., Kumar P., Singh K.* (2006), “An improved two-step algorithm for task and data parallel scheduling in distributed memory machines”. Parallel Computing. Vol. 32, Issue 10, P.p. 759 – 774. (Eng)
7. *Dümmler J., Rauber T., Rünger G.* (2009) “Scalable computing with parallel tasks”. Proc. on the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS '09 (Portland, Oregon, November 16). Article No. 9, P.p. 1 – 10. (Eng)