УДК 004.4

С.Д. ПОГОРІЛИЙ, М.С. СЛИНЬКО

МЕТОДИ МОДЕЛЮВАННЯ І ВЕРИФІКАЦІЇ ДЛЯ ПРОЄКТУВАННЯ ЗАСТОСУНКІВ У ГЕТЕРОГЕННИХ АРХІТЕКТУРАХ

Анотація. Запропоновано методологію проєктування застосунків для систем із масовим паралелізмом на прикладі GPGPU-систем, орієнтовану на алгоритмічний етап проєктування. Розглянуто два етапи проєктування: створення формальної специфікації та її дослідження і верифікація. Для першого етапу запропоновано використання математичних апаратів системи алгоритмічних алгебр/модифікованої системи алгоритмічних алгебр та транзиційних систем. Для другого етапу проаналізовано використання мережевих та автоматних моделей і наведено переваги кожної моделі. Зокрема, проведено дослідження моделі обчислень в архітектурі NVIDIA CUDA за допомогою мереж Петрі, а також формул лінійно-темпоральної логіки та автоматної моделі.

Ключові слова: транзиційні системи, системи алгоритмічних алгебр, GPGPU-системи, мережі Петрі.

ВСТУП

Складність програмного забезпечення наближається до межі людського розуміння, а отже, до межі керованості. Сучасні тенденції в галузі високопродуктивних обчислень зміщуються від застосування кластерів, що складаються з модулів загального призначення, до використання більш спеціалізованих компонентів-акселераторів (тобто від універсальних центральних процесорів (CPU) до інших блоків — відеоадаптерів на основі графічних процесорів (GPU), програмованих вентильних матриць (FPGA) тощо), іншими словами — до менш функціональних і менш енерговитратних модулів. Зокрема, у роботах [1, 2] розглянуто використання GPU-акселераторів для обчислення неграфічних завдань.

Створення систем високого рівня складності з масовим паралелізмом на основі відеоадаптерів потребує розроблення новітніх наукових методів обґрунтування як архітектур таких систем, так і застосунків для них. Оскільки сучасні багатопотокові застосунки налічують сотні тисяч і мільйони потоків, розв'язання задач паралелізації для такого класу систем унеможливлює інженерне проєктування і вимагає наявності формальних методів, використання математичного апарата і способів обґрунтування прийнятих рішень.

Розробник, який працює у такому середовищі, має ров'язати низку складних задач. З одного боку, доменну логіку застосунку не варто жорстко прив'язувати до обчислювальної архітектури, оскільки це знизить ступінь гнучкості отриманої системи. З іншого боку, під час експериментальної реалізації системи потрібно конкретизувати застосунок для того, щоб максимально скористатися перевагами та особливостями конкретної архітектури. У цій роботі запропоновано методологію проєктування систем, яка надає змогу розв'язати ці задачі на алгоритмічному етапі проєктування, тобто ще до етапу експериментальної реалізації. В основу цієї методології покладено систему алгоритмічних алгебр Глушкова, транзиційні системи, метод перевірки на моделі (model checking) та правила лінійно-темпоральної логіки, а також мережі Петрі.

СТВОРЕННЯ ФОРМАЛЬНОЇ СПЕЦИФІКАЦІЇ ПРОЄКТОВАНОЇ СИСТЕМИ

Для формалізованого представлення алгоритмів функціонування абстрактної моделі електронно-обчислювальної машини (ЕОМ) В.М. Глушков запропонував математичний апарат системи алгоритмічних (мікропрограмних) алгебр (САА). Перевагою САА (порівняно з класичним математичним апаратом) є її орієнтованість на проєктування алгоритмів і подання алгоритму у вигляді формули, що надає змогу зосередитися на концепціях та методах трансформації паралельних схем алгоритмів, абстрагуючись від конкретних деталей програмної реалізації. Використання САА дає можливість формалізувати логіку роботи алгоритму у такий спосіб, що можна абстрагуватися від конкретної обчислювальної архітектури та розглядати їх як окремі випадки більш загальної абстрактної схеми. До того ж, отримані схеми слугують документацією та теоретичною базою для відповідного програмного забезпечення. Фіксована САА являє собою двоосновну алгебраїчну систему, в основу якої покладено множину операторів і множину умов [3]. Використання САА як формалізованого методу проєктування ґрунтується на теоремі Глушкова, яка стверджує, що для довільного алгоритму існує (у загальному випадку не єдина) САА, в якій цей алгоритм може бути представлений регулярною схемою. Іншими словами, якщо визначити основи САА для конкретного алгоритму, можна представити цей алгоритм у вигляді схеми і здійснювати подальші трансформації та оптимізацію вже не з алгоритмом, а з його представленням у вигляді САА-схеми. У роботах [4–6] проілюстровано здійснення ланцюжка еквівалентних трансформацій у схемах алгоритмів Данцига та Джонсона на основі модифікованих систем алгоритмічних алгебр (САА-М).

Використання схем алгоритмів надає змогу отримати модель системи першого рівня. Втім, застосунок буде реалізовано на основі конкретної обчислювальної архітектури, тому потрібно обґрунтувати коректність та надійність моделі саме в межах цієї архітектури. Розв’язання задачі щодо обґрунтування коректності є нетривіальним, особливо якщо обчислювальна архітектура є гетерогенною з масовим паралелізмом (наприклад, як у технології General Purpose GPU (GPGPU)). У разі використання традиційних методів розроблення програм основним методом забезпечення коректності є тестування. Проте, за словами Едсгера Дейкстри, тестування може показати наявність помилок, але не може гарантувати їхню відсутність [7]. До того ж, під час тестування не можна виявити типові помилки синхронізації паралельних програм: помилки в них можуть зберігатися роками і виявлятися після тривалої експлуатації як реакція на специфічну комбінацію багатьох факторів, наприклад, непередбачуваних швидкостей виконання окремих паралельних процесів. Навіть якщо механізм функціонування кожного компонента системи, спроєктованої на основі моделі масового паралелізму, є абсолютно зрозумілим, людині важко осягнути роботу всієї системи, процеси якої є взаємопов’язаними.

Зручним засобом моделювання проєктованої системи є апарат транзиційних систем (ТС). Хоча САА і надають змогу осягнути структуру застосунку та здійснювати еквівалентні перетворення схеми застосунку без прив’язування до апаратної архітектури, проте вони не забезпечують можливості темпорального моделювання та глибокого аналізу. Зате моделі, створені за допомогою апарату ТС, можна легко конвертувати в інші формати, які підтримують темпоральну логіку, наприклад, мережі Петрі. Крім того, перевагою ТС є можливість створення моделей різного рівня складності залежно від розв’язуваної задачі.

Означення 1. Простою транзиційною системою [8] називається четвірка $A = (S, R, \alpha, \beta)$, де S — скінченна або нескінченна множина станів; R — скінченна або нескінченна множина переходів; α, β — два відображення із R в S, що ставлять у відповідність кожному переходу $t \in R$ два стани $\alpha(t)$ та $\beta(t)$, які називаються відповідно початком і кінцем переходу t.

Означення 2. Загальною транзиційною системою (ЗТС) називається ТС $A = (S, X, R, s_0, AP, L)$, де S — множина станів; X — множина дій, асоційованих з переходами; $R \subseteq S \times X \times S$ — відношення переходів; s_0 — початковий стан ЗТС; AP — множина пропозиційних формул, асоційованих зі станами; $L : S \to B(AP)$ — функція позначок станів, де $B(AP)$ — булеан множини AP.

Інтеграція ТС у цілісну систему, яка організовує роботу всіх ТС, виконується залежно від характеру потрібної взаємодії складових (синхронна, асинхронна, паралельна, послідовна). Ці способи взаємодії узагальнюють операцією синхронного добутку ТС. Визначення синхронного добутку ТС та приклад його побудови наведено в [1]. Застосування ТС, які описують складові компоненти проєктованої системи, а також визначення протоколу їхньої взаємодії (обмеження синхронізації добутку ТС) надає змогу отримати високорівневу специфікацію системи (модель загального типу). Маючи вищезазначену модель, можна приступати до процесу її верифікації. Основними моделями такого процесу є автоматні та мережеві моделі, а шляхи аналізу розглянуто в наступних розділах роботи.

ВИКОРИСТАННЯ МЕРЕЖЕВИХ МОДЕЛЕЙ ДЛЯ ВЕРИФІКАЦІЇ СПЕЦИФІКАЦІЇ

Використання ТС як моделі системи високого рівня складності надає змогу створювати абстракції різних рівнів (за рахунок рівня деталізації, вибору формальної логічної мови для опису специфікацій тощо), композиції яких можна транслювати в мережі Петрі (МП) і досліджувати за допомогою засобів МП. Нагадаємо, що сучасні гетерогенні обчислювальні архітектури з використанням відеоадаптерів оперують багатьма групами потоків одночасно, тому виконання застосунку має синхронно-асинхронний характер, оскільки різні інструкції можуть мати різну тривалість часу виконання і не повернуться одночасно в місце синхронізації. Тому доцільним є використання саме апарату мереж Петрі.

У [8] показано, що семантика добутку ТС і семантика МП, яка його моделює, є узгодженими в тому сенсі, що послідовність глобальних переходів $t_1, \ldots, t_k$ являє собою глобальну історію добутку транзиційних систем тоді й тільки тоді, коли вона є допустимою послідовністю спрацьовувань переходів у МП. Відповідно елементи множини T стають переходами МП, а глобальні стани ТС (набір станів кожної з ТС, що беруть участь в синхронному добутку до або після глобального переходу) стають місцями отриманої мережі.

Мережі Петрі та інші мережеві моделі доцільно використовувати для визначення коректності моделі на високому рівні абстракції, без прив'язування до семантики позначок переходів, тобто для перевірки надлишковості системи, наявності дедлоків тощо. Прикладами аналізу моделей за допомогою МП є роботи [1] (дослідження дедлоків та пасток) та [2] (структурна живучість на обмеженість).

ВИКОРИСТАННЯ АВТОМАТНИХ МОДЕЛЕЙ ДЛЯ ВЕРИФІКАЦІЇ СПЕЦИФІКАЦІЙ

Використання лише мереж Петрі не завжди є достатнім: навіть якщо система відповідає всім критеріям надійності на високому рівні абстракції, вона може не відповідати власне функціональним вимогам. Застосування автоматних моделей та формул лінійно-темпоральної логіки надає змогу знайти фрагменти шляхів ви-

конання (префікси), що можуть вивести систему із глобального стану семантичної коректності (тобто система перестає відповідати певним інваріантам).

Як видно з означення 2, для довільної ТС $A = (S, X, R, s_0, AP, L)$ множина функцій $L_i : s_i \in S$ надає змогу визначити формули семантично коректної роботи системи для кожного переходу та в кожному стані. Формалізуємо процес аналізу відповідності системи формулам, що описують функціональні вимоги.

Означення 3. Підмножину P називають лінійно-темпоральною (LT) формулою/характеристикою над множиною атомарних пропозиційних формул AP, якщо $P \subseteq (B(AP))^*$, і $B(AP)$ — булеан множини AP.

Означення 4. Нехай дано ЗТС $A = (S, X, R, I, AP, L)$ та послідовність станів $\pi = s_0 s_1 \dots s_n$. Трасою $trace(\pi)$ скінченної послідовності π називають слово $L(s_0)L(s_1)\dots L(s_n)$. Отже, множина трас — це множина скінченних слів над алфавітом пропозиційних формул $B(AP)$ таких, які виконуються у станах цієї послідовності. Позначимо $trace(\Pi) = \{trace(\pi) \mid \pi \in \Pi\}$, $trace(s) = trace(Path(s))$, $Traces(A) = \bigcup\limits_{s \in I} trace(s)$, де $Path(s)$ — максимальний фрагмент шляху π, що починається зі стану s. «Поганими» префіксами $L(s_0)L(s_1)\dots L(s_n)$ будемо називати префікси, які порушують істинність $L(s_i)$, тобто слова такого вигляду:

$$p'_1 = \overline{L(s_0)} \dots L(s_n),$$

$$\dots$$

$$p'_n = L(s_0) \dots \overline{L(s_n)}.$$

Звідси випливає можливість визначення регулярної мови «поганих» префіксів $BadPref(A) = (Traces(A))^* \left(\bigcup\limits_i p_i \right)$. Така мова акцептується скінченним автоматом $B = (Q, AP, f, Q_0, F)$, де $Q_0 \cap F = \varnothing$ (рис. 1).

Якщо P_A — властивість, виконання якої гарантує правильність функціонування ЗТС A, тоді семантична коректність ЗТС A визначається умовою

$$Traces(A) \cap BadPref(P_A) = Traces(A) \cap L(B) = \varnothing,$$

де $L(B)$ — мова, яку акцептує автомат B. Отже, алгоритм перевірки лінійно-темпоральної формули P містить такі кроки.

1. Створити автомат Бюхі [9], що акцептує слова, які заперечують P.

2. Побудувати добуток автомата і ЗТС, що моделює вихідну систему.

3. Знайти переріз шляхів, які генерує ЗТС, і шляхів, які акцептує автомат (так званих «поганих префіксів»). Якщо переріз є пустою множиною, формула P є істинною. Інакше було знайдено контрприклад, який доводить невиконання формули P.

Розглянемо процес використання автоматних моделей для аналізу семантичної коректності ЗТС, що описує модель виконання застосунку в архітектурі NVIDIA CUDA (деталі формування ЗТС описано в [2]). Зображення ЗТС $Z = (S, T, R, s_1, AP, L)$ наведено на рис. 2. Ця система відповідає критеріям надійності на високому рівні абстракції (її верифіковано за допомогою мережевих моделей у [1, 9]).

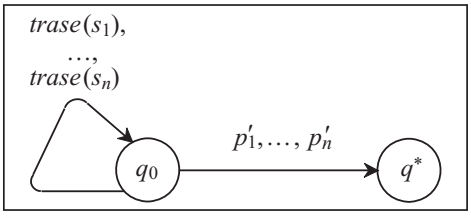

Рис. 1. Автомат B, який акцептує «погані» префікси, $q_0 \in Q_0$, $q_0 \notin F$, $q^* \in F$

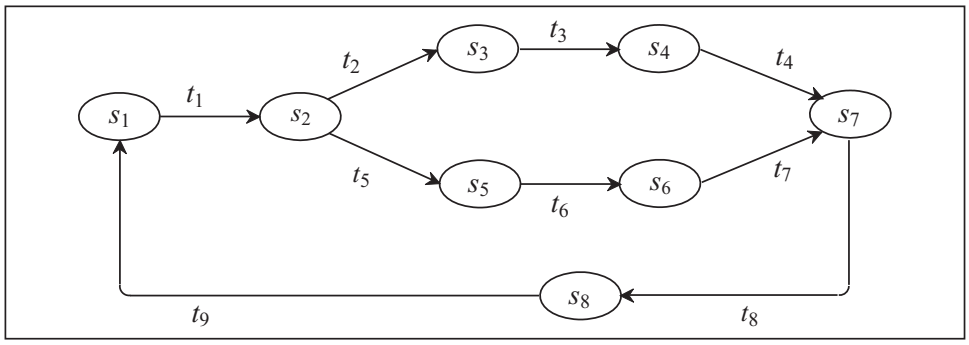

Рис. 2. Схематичне зображення ЗТС Z

У ЗТС Z існують такі шляхи виконання:

$$\pi_1 = s_1 s_2 s_3 s_4 s_7 s_8 s_1,$$

$$\pi_2 = s_1 s_2 s_5 s_6 s_7 s_8 s_1.$$

Траси, що відповідають шляхам виконання π_i, є такими:

$$p_1 = trace(\pi_1) = L(s_1)L(s_2)L(s_3)L(s_4)L(s_7)L(s_8)L(s_1),$$

$$p_2 = trace(\pi_2) = L(s_1)L(s_2)L(s_5)L(s_6)L(s_7)L(s_8)L(s_1),$$

$$Traces(Z) = trace(\pi_1) \vee trace(\pi_2) =$$

$$= L(s_1)L(s_2)(L(s_3)L(s_4) \vee L(s_5)L(s_6))L(s_7)L(s_8).$$

Однією з характеристик, які визначають семантичну коректність роботи системи, є властивість взаємного виключення: у межах одного варпа в будь-який момент часу інструкція може бути або арифметичною, або інструкцією доступу до пам'яті (або інструкція відсутня взагалі). Цю властивість представлено у вигляді лінійно-темпоральної формули $P = G(\overline{f_{\text{ar}}} \vee \overline{f_{\text{mem}}})$, де f_{ar} — пропозиційна формула, яка вказує на арифметичний тип інструкції; f_{mem} — формула, що вказує на тип інструкції доступу до пам'яті; G — розширення булевої алгебри семантикою LT логіки, що означає «завжди» [9].

Перевіримо формулу P на істинність, використовуючи алгоритм, наведений вище. Автомат, який акцептує слова, що заперечують формулу P, має вигляд, зображений на рис. 3.

Щоб перевірити істинність формули P, необхідно описати добуток ЗТС Z та B_P, в якому потрібно перевірити наявність шляху-циклу такого, що є доступним із початкового стану, та який включає в себе стан із F.

Шляхи, що задовольняють зазначені вище правила, відсутні в системі, зображеній на рис. 4. Це означає, що $Traces(Z) \cap BadPref(P) = \varnothing$, тобто можна стверджувати, що формула P є істинною.

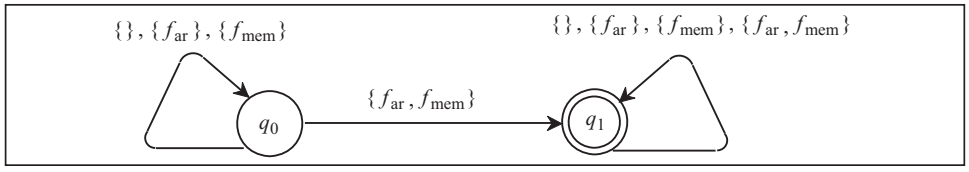

Рис. 3. Автомат $B_P = (Q, AP, f, Q_0, F)$, що акцептує заперечення P, $q_0 \notin F$, $q_1 \in F$

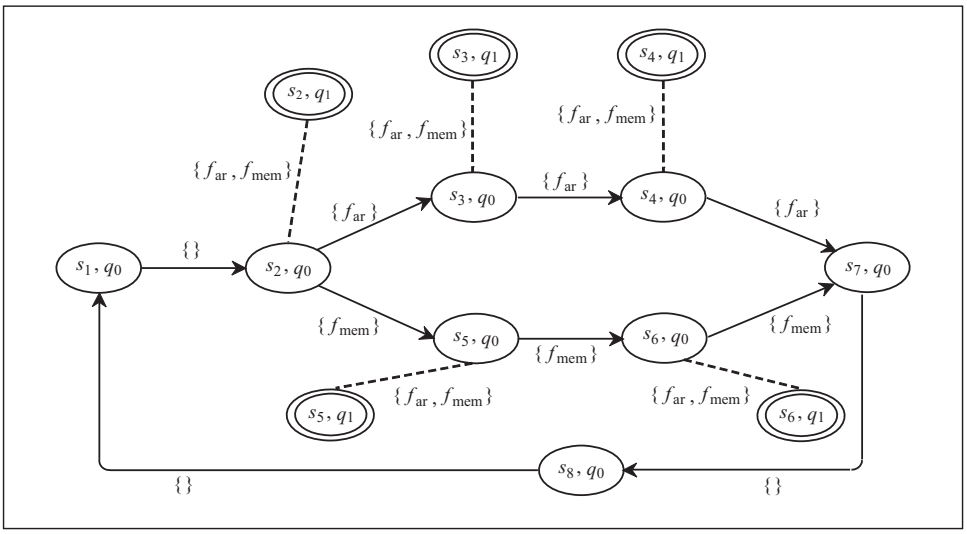

Рис. 4. Добуток автомата B_P і ЗТС Z

ВИСНОВКИ

У роботі запропоновано методологію проєктування застосунків для систем із масовим паралелізмом на прикладі GPGPU-систем. Методологія є орієнтованою на алгоритмічний етап проєктування та містить два основні кроки: створення специфікації системи та її верифікація і аналіз. Також запропоновано і описано використання САА-М Глушкова та апарату ТС для отримання формалізованої специфікації проєктованої системи. Розглянуто автоматні та мережеві моделі для застосування на етапі верифікації та аналізу на прикладі автоматів, що акцептують «погані» префікси, та мереж Петрі відповідно. Показано, що перевагою моделей, створених за допомогою апарату ТС є їхня конвертованість в інші формати, що підтримують темпоральну логіку (зокрема, мережі Петрі). Наведено приклад представлення властивості взаємного виключення в моделі виконання застосунку в архітектурі NVIDIA CUDA як лінійно-темпоральної формули та її дослідження за допомогою автоматних моделей.

СПИСОК ЛІТЕРАТУРИ

1. Погорілий С.Д., Кривий С.Л., Слинько М.С. Проектування та модельне обгрунтування застосувань на основі відеоадаптерів. *Управляющие системы и машины.* 2018. № 4. С. 46–56. https://doi.org/10.15407/usim.2018.04.0046.
2. Kryvyi S., Porogilyy S., Slynko M. Transition systems as method of designing applications in GPGPU technology. *Problems in Programming.* 2018. N 2–3. P. 12–20.
3. Anisimov A.V., Pogorilyy S.D., Vitel D.Yu. About the issue of algorithms formalized design for parallel computer architectures. *Applied and Computational Mathematics*. 2013. Vol. 12, N 2. P. 140–151.
4. Погорілий С.Д., Мар'яновський В.А., Бойко Ю.В., Верещинський О.А. Дослідження паралельних схем алгоритму Данцига для обчислювальних систем зі спільною пам'яттю. *Математичні машини і системи.* 2009. № 4. С. 27–37.
5. Pogorilyy S.D., Slynko M.S. Research and development of Johnson's algorithm parallel schemes in GPGPU technology. *Problems in Programming.* 2016. N 2–3. P. 105–112.
6. Pogorilyy S.D., Shkulipa I.Yu. A conception for creating a system of parametric design of parallel algorithms and their software implementations. *Cybernetics and Systems Analysis.* 2009. Vol. 45, N 6. P. 952–958.
7. Dijkstra E.W., Buxton J.N., Randell B. Software engineering techniques. *Report on a Conference Sponsored by the NATO Science Committee*. (27–31 October 1969, Rome, Italy). Rome, Italy, 1969. P. 16.

8. Бойко Ю.В., Кривий С.Л., Погорілий С.Д. та ін. Методи та новітні підходи до проектування, управління і застосування високопродуктивних IT-інфраструктур. Київ: ВПЦ «Київський університет», 2016. 447 с.
9. Albert E., Lanese I. Formal techniques for distributed objects, components, and systems. *Proc. 36th IFIP WG 6.1 International Conference, FORTE 2016, held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016,* (June 6–9, 2016, Heraklion, Crete, Greece). Heraklion, Crete, Greece, 2016. 275 p.

Надійшла до редакції 20.02.2020

С.Д. Погорелый, М.С. Слинько

МЕТОДЫ МОДЕЛИРОВАНИЯ И ВЕРИФИКАЦИИ ДЛЯ ПРОЕКТИРОВАНИЯ ПРИЛОЖЕНИЙ В ГЕТЕРОГЕННЫХ АРХІТЕКТУРАХ

Аннотация. Предложена методология проектирования приложений для систем с массовым параллелизмом на примере GPGPU-систем, ориентированная на алгоритмический этап проектирования. Рассмотрены две фазы проектирования: создание формальной спецификации и ее исследование и верификация. Для первого этапа предложено использование математических аппаратов системы алгоритмических алгебр/модифицированной системы алгоритмических алгебр и транзиционных систем. Для второго этапа проанализировано использование сетевых и автоматных моделей и приведены преимущества каждой из них. В частности, проведено исследование модели вычислений в архитектуре NVIDIA CUDA при помощи сетей Петри, а также формул линейно-темпоральной логики и автоматных моделей.

Ключевые слова: транзиционные системы, системы алгоритмических алгебр, GPGPU-системы, сети Петри.

S. Pogorilyy, M. Slynko

MODELING AND VERIFICATION METHODS FOR APPLICATION DESIGN IN HETEROGENEOUS ARCHITECTURES

Abstract. An application design methodology for massively parallelized systems (GPGPU systems) focused on the algorithmic design phase is proposed. Two design sub-steps are considered: creation of the formal specification of the system; and its research and verification. The use of mathematical apparatuses SAA/SAA-M as well as transition systems is proposed for the first step. Advantages and specifics of using network and automaton models for the second step are given. NVIDIA CUDA architecture computation model analysis is proposed using Petri nets (network model) and using linear-temporal logic formulas and Buchi automaton (automaton model).

Keywords: transition systems, systems of algorithmic algebras, GPGPU-systems, Petri nets.

Погорілий Сергій Дем'янович,
доктор техн. наук, професор, завідувач кафедри Київського національного університету імені Тараса Шевченка, e-mail: sdp@univ.net.ua.

Слинько Максим Сергійович,
аспірант Київського національного університету імені Тараса Шевченка, e-mail: maxim.slinko@gmail.com.