Харківський національний університет імені В. Н. Каразіна
Міністерство освіти і науки України

Кваліфікаційна наукова праця
на правах рукопису

МОРОЗ ОЛЬГА ЮРІЇВНА

УДК 004.05

ДИСЕРТАЦІЯ

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВЕРИФІКАЦІЇ ПАРАЛЕЛЬНИХ ЧАСОПАРАМЕТРИЗОВАНИХ ПРОГРАМ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ

Спеціальність 122 – Комп’ютерні науки
Галузь знань 12 – Інформаційні технології

Подається на здобуття ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.
_____ **О. Ю. Мороз**

Науковий керівник: **Толстолузька Олена Геннадіївна**
доктор технічних наук, старший науковий співробітник

Харків – 2023

АНОТАЦІЯ

Мороз О. Ю. Інформаційна технологія верифікації паралельних часопараметризованих програм інформаційних управляючих систем. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки (Галузь знань 12 – Інформаційні технології). – Харківський національний університет імені В. Н. Каразіна, Міністерства освіти і науки України, Харків, 2023.

Дисертація присвячена верифікації паралельних програм для інформаційних управляючих систем, що є складною та важливою задачею з багатьма проблемами та викликами.

В першому розділі розглядаються особливості процесу верифікації паралельних програм інформаційних управляючих систем. Наводиться стислий огляд публікацій за темою дисертації, робиться аналіз проблемних питань з верифікації паралельних програм, опис методів верифікації паралельних програм інформаційних управляючих систем: експертизи програмного забезпечення, статичного аналізу, динамічних та формальних методи верифікації; наводяться переваги та недоліки цих методів. Описані найпоширеніші інструментальні засоби верифікації програм з підтримкою паралельних обчислень. Приділено увагу, що на сьогодні необхідність високої ефективності паралельного програмного забезпечення інформаційних управляючих систем вимагає суттєвого розширення складу факторів, що враховуються при формальній розробці часопараметризованих мультипаралельних програм інформаційних управляючих систем.

Акцентовано, що верифікація мультипаралельних часопараметризованих програм має три складові: компіляційна верифікація; декомпіляційна верифікація та семантична верифікація. Компіляційна верифікація забезпечує перевірку синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації,

декомпіляційна верифікація забезпечує перевірку логічної еквівалентності синтезованих мультипаралельних програм та текстів вхідних послідовних програм після завершення всіх етапів синтезу, а семантична верифікація полягає у перевірці збігу одиниць вимірювання фізичних величин, отриманих на основі формального синтезу часопараметризованих мультипаралельних програм інформаційних управляючих систем та одиниць вимірювання вхідних та вихідних даних, що задаються користувачами. Сформовано сучасні вимоги до суттєвого розширення складу факторів, що враховуються при формальній розробці часопараметризованих мультипаралельних програм інформаційних управляючих систем.

Формулюється задача дисертаційного дослідження, як розробка інформаційної технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення її ефективності за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації на основі семантико-числових специфікацій. Для вирішення поставленої науково-прикладної задачі вирішено розробити ряд методів, а саме: метод компіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем; метод декомпіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем; метод семантичної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем.

У другому розділі описано етапи розробки методу компіляційної верифікації паралельних програмних засобів інформаційних управляючих систем. Описані існуючі методи паралельної обробки інформації та їх сутність, показано доцільність використання сукупності методів паралельної обробки інформації при розробці паралельного програмного забезпечення інформаційних управляючих систем.

Дана загальна характеристика методу компіляційної верифікації паралельних програм інформаційних управляючих систем, представлені

основні етапи компіляційної верифікації процесів синтезу часопараметризованих мультипаралельних програм інформаційних управляючих систем. Наведено приклад компіляційної верифікації синтезу часопараметризованої паралельної програми задачі для ілюстрації її застосування. Для візуалізації результатів використовується гістограмне представлення верифікації розподілу множини часових операторів паралельної SMP-моделі по часовим ярусах з більшею деталізацією результатів верифікації, яке відображає результати перевірки коректності розташування часових операторів і множинних часових операторів на часових ярусах для часової моделі і структур семантико-числових специфікацій.

У третьому розділі описано сутність декомпіляційної верифікації часопараметризованих програмних засобів інформаційних управляючих систем. Починаючи з аналізу бінарного коду послідовної програми, проводиться його декомпіляція з метою відновлення вхідного коду. Отриманий вхідний код піддається формальній верифікації за допомогою математичних методів. Відбувається доведення коректності програми, перевірка властивостей безпеки, аналіз відповідності вимогам тощо. Аналізуючи вхідний код програми, можна виявити можливі вразливі місця, дефекти або помилки, що можуть впливати на її безпеку та надійність.

Також у розділі описані етапи постановки задачі декомпіляційної верифікації часопараметризованих паралельних програмних засобів інформаційних управляючих систем. Зокрема представлено структурну схему методу декомпіляційної верифікації часопараметризованих паралельних програмних засобів інформаційних управляючих систем та детальний опис процедур, які реалізують етапи методу.

Наведено приклади застосування методу декомпіляційної верифікації часопараметризованих паралельних програмних засобів інформаційних управляючих систем. Задля цього семантику основних етапів декомпіляційної верифікації паралельних програмних засобів інформаційних управляючих систем пояснено на прикладі паралельно-конвеєрної моделі задачі, яка

синтезована з використанням методу суміщення незалежних операцій і методу конвеєрної обробки інформації. З метою досягнення наочності зміст етапів декомпіляційної верифікації викладається та пояснюється за допомогою текстових і графічних специфікацій об’єктів і дій над ними. Зазначено, що додатковою перевагою методу є здатність перевіряти відповідність отриманого декомпільованого вихідного коду формальній специфікації та вимогам. Це дозволяє переконатися, що програма відповідає поставленим завданням та функціональним вимогам. Отже, метод декомпіляційної верифікації є потужним інструментом для забезпечення коректності та надійності програмного забезпечення. Його застосування може значно підвищити рівень довіри до програм та систем, особливо в умовах зростаючої складності та критичності інформаційних технологій.

У четвертому розділі описано метод семантичної верифікації часопараметризованих мультипаралельних програм, змістовно розглянуті основні етапи методу семантичної верифікації мультипаралельних програм та описано побудову графу, що здійснюється за допомогою засобів візуалізації паралельних апаратно-програмних об’єктів. Представлена концептуальна модель технології верифікації часопараметризованих паралельних програм інформаційних управляючих систем та змістовно описано основні компоненти архітектури технології верифікації. В більш деталізованому вигляді описана структура синтезаційного верифікатору графів та представлено структуру синтезаційного верифікатора текстів часопараметризованих мультипаралельних програм, синтезаційного верифікатора часопараметризованих мультипаралельних моделей процесів та саму структуру синтезаційного верифікатора текстів часопараметризованих мультипаралельних програм, що забезпечує декомпіляційно-семантичну верифікацію текстів синтезованих часових мультипаралельних програм з урахуванням типів та розмірностей даних, складу обчислювальних та керуючих операторів, засобів обміну даними та операторів часової синхронізації процесів.

Обрано показники ефективності технології семантико-числової верифікації інформаційних управляючих систем спрямовані на оцінку точності та надійності результатів верифікації. Оцінка показників ефективності розробленої технології проводилась по цілому ряду практичних прикладних задач. До них можна віднести: метод Гауса для рішення систем лінійних рівнянь великої розмірності, алгоритм адаптивного згладжування і екстраполяції траєкторії, алгоритм цілерозподілу, тощо.

Сукупність отриманих у дисертації нових наукових результатів, позитивна оцінка їхньої достовірності, наукової та практичної значущості дають змогу вважати сформульовану наукову задачу розробки технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення ефективності верифікації за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації на основі семантико-числових специфікацій, – розв'язаною, а поставлену мету – досягнутою.

Ключові слова: *технологія верифікації, методи верифікації, паралельні обчислювальні системи, часопараметризовані паралельні програми, мультипаралельні програми, розподілені системи, верифікація паралельних програм, інформаційні управляючі системи, компіляційна верифікація; декомпіляційна верифікація, семантична верифікація, методи семантико-числових специфікацій, методи формального синтезу, часопараметризована модель алгоритму Гауса.*

ABSTRACT

Moroz O. Yu. Information technology for verification of parallel time-parameterized programs of information control systems. – Qualification scholarly paper: a manuscript.

The dissertation submitted for obtaining the Doctor of Philosophy degree in Information Technology: Speciality 122 – Computer science. V. N Karazin Kharkiv National University, Ministry of Education and Science of Ukraine, Kharkiv, 2023.

The dissertation is devoted to the development and verification of parallel programs for information management systems, which is a complex and important task with many problems and challenges.

The first chapter examines the features of the process of verification of parallel programs of information management systems. It provides a brief overview of publications on the topic of the dissertation, analyzes problematic issues in the verification of parallel programs, describes methods of verification of parallel programs of information management systems: software expertise, static analysis, dynamic and formal verification methods; the advantages and disadvantages of these methods is given. The most common tools for verifying programs with parallel computing support are described. Attention is focused on the fact that today the need for high efficiency of parallel software of information management systems requires a significant expansion of the composition of factors that are taken into account during the formal development of time-parameterized multi-parallel programs of information management systems.

It is emphasized that the verification of multi-parallel time-parameterized programs has three components: compilation verification; decompilation verification and semantic verification. Compilation verification provides checking the syntactic and temporal correctness of the formal synthesis of semantic-numerical specification structures. Decompilation verification provides checking the logical equivalence of synthesized multiparallel programs and input sequential program texts after completion of all stages of synthesis. And semantic verification consists

in verifying the coincidence of the units of measurement of physical quantities obtained on based on the formal synthesis of time-parameterized multi-parallel information management systems programs and units of measurement of input and output data specified by users. Modern requirements for a significant expansion of the composition of factors taken into account during the formal development of time-parameterized multi-parallel information management systems programs have been formed.

The task of the dissertation research is formulated as the development of information technology for the verification of parallel time-parameterized programs of information management systems with the aim of increasing its efficiency due to the application of compilation, decompilation and semantic verification based on semantic-numerical specifications. To solve the scientific and applied problem, it was decided to develop a number of methods, namely: the method of compilation verification of parallel time-parameterized programs for information management systems; method of decompilation verification of parallel time-parameterized programs for information management systems; method of semantic verification of parallel time-parameterized programs for information management systems.

The second chapter describes the stages of development of the method of compilation verification of parallel software tools of information control systems. It describes the existing methods of parallel information processing and their essence, shows the expediency of using a set of methods of parallel information processing in the development of parallel information management systems software.

The general characteristics of the method of compilation verification of parallel programs of information management systems are given, the main stages of compilation verification of the synthesis processes of time-parameterized multiparallel programs of information control systems are presented. An example of compilation verification of the synthesis of a time-parameterized parallel program of the problem is given to illustrate its application.

It is used a histogram representation of the verification of the distribution of multiple time operators of the parallel SMP model on time tiers with greater detail

of the verification results to visualize the results, which displays the results of checking the correctness of the location of time operators and multiple time operators on time tiers for the time model and structures of semantic-numerical specifications.

The third chapter describes the essence of decompilation verification of time-parameterized software tools of information management systems. Starting with the analysis of the binary code of the C-program, it is decompiled in order to restore the input code. The received input code is subjected to formal verification using mathematical methods. Proving the correctness of the program, checking the security properties, analyzing compliance with the requirements, etc. takes place. Analyzing the program's input code can reveal possible vulnerabilities, defects, or errors that could affect its security and reliability.

The chapter also describes the stages of setting the task of decompilation verification of time-parameterized parallel software tools of information management systems. In particular, it presents the structural scheme of the method of decompilation verification of time-parameterized parallel software tools of information management systems and a detailed description of the procedures that implement the stages of the method.

In chapter, there are examples of the application of the method of decompilation verification of time-parameterized parallel software tools of information management systems. For this purpose, the semantics of the main stages of decompilation verification of parallel software tools of information management systems are explained on the example of a parallel-conveyor model of the problem, which is synthesized using the method of combining independent operations and the method of conveyor information processing. In order to achieve clarity, the content of the stages of decompilation verification is explained with the help of text and graphic specifications of objects and actions on them. It is noted that an additional advantage of the method is the ability to check the compliance of the received decompiled source code with the formal specification and requirements. This allows to make sure that the program meets the tasks and functional requirements.

Therefore, the decompilation verification method is a powerful tool for ensuring the correctness and reliability of software. Its application can significantly increase the level of trust in programs and systems, especially in conditions of increasing complexity and criticality of information technologies.

The fourth chapter describes the method of semantic verification of time-parameterized multiparallel programs, reviews the main stages of the method of semantic verification of multiparallel programs meaningfully, and describes the construction of the graph, which is carried out using means of visualization of parallel hardware and software objects. It presents the conceptual model of the verification technology of time-parameterized parallel information management systems programs and describes the main components of the verification technology architecture meaningfully. The structure of the synthesis verifier of graphs is described in more detail. Also, the chapter presents the structure of the synthesis verifier of texts of time-parameterized multiparallel programs, and the synthesis verifier of time-parameterized multiparallel process models, and the structure of the synthesis verifier of texts of time-parameterized multiparallel programs, which provides decompilation-semantic verification of texts of synthesized temporal multiparallel programs, taking into account the types and dimensions of data, composition of computing and control operators, means of data exchange and operators of time synchronization of processes.

To solve the task, it is chosen the indicators of the effectiveness of the technology of semantic-numerical verification of information management systems, aimed at assessing the accuracy and reliability of the verification results. The evaluation of the efficiency indicators of the developed technology was carried out on a number of practical applied problems. These include: the Gaussian method for solving systems of linear equations of large dimensions, the algorithm of adaptive smoothing and extrapolation of the trajectory, the algorithm of target distribution, etc.

The set of new scientific results obtained in the dissertation, a positive assessment of their reliability, scientific and practical significance make it possible

to consider the formulated scientific task of developing a technology for the verification of parallel time-parameterized programs of information management systems in order to increase the effectiveness of verification due to the application of compilation, decompilation and semantic verification based on semantic of numerical specifications, is solved, and the set goal is achieved.

Keywords: *verification technology, verification methods, parallel computing systems, time-parameterized multi-parallel programs, distributed systems, verification of parallel programs, information management systems, compilation verification; decompilation verification, semantic verification, methods of structures of semantic-numerical specifications, methods of formal synthesis, time-parameterized model of the Gaussian algorithm*.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях, що входять до міжнародних наукометричних баз

1. Dmytro Chumachenko, Ievgen Meniailov, Andrii Hrimov, Vladislav Lopatka, Olha Moroz, Olena Tolstoluzka. Simulation and forecasting of the influenza epidemic process using seasonal autoregressive integrated moving average model. Radioelectronic and Computer Systems, 2021, no. 4(100), Pp.22–35 DOI: https://doi.org/10.32620/reks.2021.4.02 **(*Scopus*).**

(Особистий внесок: розробка методу формальної верифікації та засобів його програмної реалізації в ході експериментальних результатів моделювання епідемічного процесу грипу, а також написання частини тексту та його переклад).

Статті у наукових фахових виданнях України

2. Olena Tolstoluzka, Dmitriy Tolstoluzkiy, Olga Moroz. Compilations method and semantic verification time parameterized of multiparallel programs. Computer Scienceand Cybersecurity, ISSNE 4(4), 2016 С. 26–34. DOI: https://periodicals.karazin.ua/cscs/article/view/8264

(Особистий внесок: участь у розробці методу семантичної верифікації часопараметризованих мультипаралельних програм, а також написання тексту та його переклад).

3. Parshentsev B., Tolstoluzka O., Moroz O. Parallel implementation of the method of gradient boosting. Advanced Information Systems. 2018. Vol. 2, No. 3 P. 19–23. DOI: https://doi.org/10.20998/2522-9052.2018.3.03

Особистий внесок: участь у процесі перевірки синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації, обробка результатів, а також написання частини тексту та його пеерклад.

4. Moroz O.Yu., Tolstoluzka O. G., Savchenko R.V. Analysis of existing technologies for verification of parallel programs. Bulletin of V. Karazin Kharkiv

National University series «Mathematical Modelling. Information Technology. Automated Control Systems». 2020. Issue 46. P. 76–81 DOI: 10.26565/2304-6201-2020-46-07 DOI: https://doi.org/10.26565/2304-6201-2020-46

(Особистий внесок: опис методів верифікації паралельних програм, експертизи програмного забезпечення, статичного аналізу, динамічних та формальних методів верифікації, а також написання частини тексту та його переклад).

5. Толстолузький Є. Д., Бердніков А. Г., Будько В. В., Толстолузька О. Г., Мороз О. Ю. Розробка та верифікація СЧС моделі мережевого планування. Вісник Харківського національного університету імені В. Н. Каразіна серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», випуск 51, 2021, С. 81–86.
DOI: https://doi.org/10.26565/2304-6201-2021-51

(Особистий внесок: участь у розробленні, підготовці та верифікації семантико-числової специфікації моделі мережевого планування, а також написанні частини тексту).

6. Мороз О. Ю. , Толстолузька О. Г. Використання методів формального синтезу та веріфікації паралельних часопараметризованих моделей для рішення системи лінійних рівнянь методом Гауса. Вісник Харківського національного університету імені В. Н. Каразіна серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», випуск 52, 2021 С. 55–71. DOI: https://doi.org/10.26565/2304-6201-2021-52-07

(Особистий внесок: обробка та перевірка синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації паралельних часопараметризованих моделей для рішення системи лінійних рівнянь методом Гауса, участь в обговоренні отриманих результатів та написанні тексту).

7. Мороз О. Ю. Технологія семантико-числової верифікації часопараметризованих паралельних програм для інформаційних і управляючих систем. *Вісник Харківського національного університету імені*

В. Н. Каразіна, сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». 2022. випуск. 55. С.43–48. DOI: https://doi.org/10.26565/2304-6201-2022-55-04

8. Мороз О. Ю. Компіляційно-семантична верифікація часопараметризованих мультипаралельних програм для інформаційних управляючих систем. *Вісник Харківського національного університету імені В.Н.Каразіна, серія. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2022. випуск 56. С.43–50. DOI: https://doi.org/10.26565/2304-6201-2022-56-04

Монографії

9. Мороз О. Ю., Толстолузька О. Г. Толстолузький Є.Д. Концептуальна модель технології семантико-числової верифікації часопараметризованих мультипаралельних програм. Moderní aspekty vědy: XXXI. Díl mezinárodní kolektivní monografie / Mezinárodní Ekonomický Institut s.r.o.. Česká republika: Mezinárodní Ekonomický Institut s.r.o., 2023. Oddíl 12. Počítačová vědy §12.1 str. 433-452. URL: http://perspectives.pp.ua/public/site/mono/mono-31.pdf

Особистий внесок: участь у побудові моделі технології семантико-числової верифікації часопараметризованих мультипаралельних програм, обговоренні результатів застосування методів формальної верифікації паралельних часопараметризованих моделей, участь в обговоренні та інтерпретації отриманих результатів, написання тексту.

Патенти

10. Корисна модель. Канал вимірювання кутових швидкостей літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Толстолузька О. Г., Звєрєв О.О., Садовий К. В. та ін., усього 10 осіб// Патент України на корисну модель №120560 від 10.11.2017 G01S 17/42, G01S 17/66.

11. Корисна модель. Канал вимірювання радіальної швидкості літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Толстолузька О. Г., Садовий К. В., Звєрєв О. О., та

ін., усього 10 осіб// Патент України на корисну модель №120559 від 10.11.2017 G01S 17/42, G01S 17/66

12. Корисна модель. Канал автоматичного супроводження літальних апаратів за напрямком оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Кузнєцов О. Л., Сачук І. І., Довбня О. В. та ін., усього 10 осіб // Патент України на корисну модель №120557 від 10.11.2017 G01S 17/66, G01S 17/42.

13. Корисна модель. Канал вимірювання похилої дальності до літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Садовий К. В., Толстолузька О. Г., Довбня О. В. та ін., усього 10 осіб //Патент України на корисну модель №121427 від 11.12.2017 G01S 17/66, G01S 17/42.

14. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання радіальної швидкості літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації, 05.01.2022, № 150146, Бюлетень № 1, 2022

15. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал автоматичного супроводження літальних апаратів за напрямком з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150147, 05.01.2022, бюл. № 1/2022

16. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання кутових швидкостей літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150148, 05.01.2022, бюл. № 1/2022

17. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання похилої дальності до літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150194, 12.01.2022, бюл. № 2/2022

18. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання радіальної швидкості літальних апаратів з використанням частот

міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150891, 04.05.2022, бюл. № 18/2022

19. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал автоматичного супроводження літальних апаратів за напрямком з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150892, 04.05.2022, бюл. № 18/2022

20. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання кутових швидкостей літальних апаратів з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150893, 04.05.2022, бюл. № 18/2022

Наукові праці, які засвідчують апробацію матеріалів дисертації:

21. Толстолужская В. В., Толстолужская Е. Г., Мороз О. Ю. Программная модель исследования эффективности трудозатрат на разработку параллельных программ. Труды международной НТК «Компьютерное моделирование в наукоемких технологиях», Харьков, 28 – 31 мая 2014р. С.378–381

22. Поляков Г. О., Мороз О.Ю., Толстолузький Д. О. Метод компиляционно-семантической верификации статических и времяпараметризованных мультипараллельных программ. Проблеми автоматизації, Черкаси, 12–13 листопада 2015 р. С. 61

23. Мороз О.Ю. Толстолузька О.Г. Анализ средств верификации параллельных программ. Проблеми інформатизації. Тези доповідей четвертої міжнародної науково-технічної конференції. – Черкаси – Баку – Бельсько-Бяла – Полтава, 3–4 листопада 2016р. С. 39–40

24. Мороз О.Ю., Синюк Б.В., Синюк Т.В., Уразов С.А. Методи верифікації паралельних програм. Сучасні напрями розвитку інформаційно-комунікацій-них технологій та засобів управління. Матеріали сьомої

міжнародної науково-технічної конференції, Полтава – Баку – Кіровоград – Харків, 2017р. С. 7.

25. Moroz O., Sinyuk T. The computer decompilation verification model of parallel programs for distributed systems. Проблеми інформатизації. Тези доповідей п’ятої міжнародної науково-технічної конференції 13–15 листопада 2017 р., Черкаси – Баку – Бельсько-Бяла – Полтава, 2017р. С. 29

26. Толстолузька О. Г., Мороз О. Ю., Паршенцев Б. В., Дослідження розпаралелювання градієнтного бустингу. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали восьмої міжнародної науково-технічної конференції. – Полтава –Баку– Харків–Жиліна, 26–27 квітня 2018р. С. 84.

27. Мороз О. Ю., Толстолузька О. Г. Аналіз існуючих технологій верифікації паралельних програм. Комп’ютерне моделювання в наукоємних технологіях: Збірник наукових праць міжнародної науково-технічної конференції (м. Харків, 22–24 квітня 2020 року) Харків: ХНУ ім. В.Н. Каразіна, 2020. №6 С.215–218

28. Мороз О. Ю., Толстолузька О. Г. Аналіз засобів технологій верифікації паралельних програм. Комп’ютерне моделювання в наукоємних технологіях: Збірник наукових праць міжнародної науково-технічної конференції (м. Харків, 21–23 квітня 2021 року) Харків: ХНУ ім. В.Н. Каразіна, 2021. С. 224–227.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DSL	–	Domain Specific Language, мова специфікації доменів
MAPLE	–	Комерційна система комп'ютерної алгебри від компанії Waterloo Maple
MATHCAD	–	Система комп'ютерної алгебри з класу систем автоматизованого проєктування
ModelChecking	–	Англ. Метод автоматичної формальної перевірки паралельних систем з конечним числом стану.
NASA	–	національне агентство з космічних досліджень
SDL, RSL	–	Алгебраїчні специфікації та мови
Spec#		Звичайна мова специфікації
UML	–	Unified Modeling Language, уніфікована мова моделювання
VDM	–	Мова формальної специфікацій
VDM, Z, RAISE	–	Універсальні мови формальної специфікацій
ІТ	–	Інформаційні технології
ІУС	–	Інформаційні управляючі системи
КМ	–	Концепторна мова
НМІ	–	людинно-машинний інтерфейс
ОС	–	Операційна система
ОУ	–	об'єкт управління
ПЗ	–	Програмне забезпечення
СППП	–	Системи проєктування паралельних програм
СЧС	–	Семантико-числова специфікація
ЧО	–	Часові оператори
ЧПГС	–	Часопараметризована графічна специфікація
ЧПМП	–	Часопараметризована мультипаралельна програма

ВСТУП

Обґрунтування вибору теми дослідження. Розробка та верифікація паралельних програм для інформаційних управляючих систем (ІУС) є складною та важливою задачею з багатьма проблемами та викликами. Сучасне суспільство неможливе без інформаційно-управляючих систем, які широко використовуються для автоматизації різних технологічних процесів та виконання важливих функцій. Складність задач, які вирішуються за допомогою ІУС, призводить до інтенсивного використання програмної реалізації для їхнього функціонування. З практики використання ІУС виокремлюється важливий факт: до 70% неполадок і відмов, що призводять до серйозних матеріальних і людських втрат, пов’язані з дефектами програмного забезпечення (ПЗ), що використовується в ІУС. Якість ПЗ є найважливішою складовою нормативного регулювання, одне з вимог якого – проведення незалежної верифікації ПЗ ІУС. Для незалежної верифікації ПЗ необхідно розробляти нові та різноманітні методи верифікації. Це важлива передумова для виявлення дефектів у програмному забезпеченні і підтвердження високої якості та надійності програмного забезпечення ІУС [1–5].

Паралельні програми складніше верифікувати, оскільки вони можуть мати незрозумілі взаємодії між різними потоками виконання. Потрібні спеціальні методи та інструменти для виявлення помилок (багів) та перевірки правильності паралельного коду. Процес верифікації повинен забезпечити ефективне використання паралельних обчислювальних можливостей, включаючи сучасні методи розробки та верифікації паралельних програм за допомогою спеціалізованих інструментів та методологій.

Теоретичні дослідження у галузі розробки та верифікації паралельних програм здобувають велику популярність та отримують значну увагу як в Україні, так і за її межами. Серед вітчизняних дослідників необхідно відзначити: Конорева Б. М., Дудкевича А. Т., Буя Д. Б., Харченка В. С., Жолткевича Г. М., Дорошенка А. Ю, Лістрового С. С., Семеренка В. П.,

Одарущенка О. М., Нікітченка Н. С., Полякова Г. О., Лосева Ю. І., Шматкова С. І., Толстолузьку О. Г., Глобу Л. С., Левченка Р. І., Качка Є. Г., Кузьму К.Т., Мельника О.В., серед дослідників зарубіжжя: Девід Берт-секаса, Камерон Хьюз, Трейси Хьюз, Столінгс Ст., Хембі С., Герберт Шилдт, Нік МакДональд, Леслі Лампорт, Луї Гуерін, Арвінд, Джек Донгарра, Річард Манчек, Меткалф М., Девід Б. Скіллікорн, Роберт Хокні, Кеннет Джесссхоуп, Джон Джонсон, Деніел Таліа, Кріс Вінскел, Мауро Каккамо, Ренцо Сімоне, Крістофер Андре [1–16].

Ці дослідники внесли свій вклад у різні аспекти інформаційних технологій, комп'ютерних наук, паралельного програмування та інших галузей і вони продовжують привертати увагу вчених та наукових груп як в Україні, так і у всьому світі.

Для забезпечення високої надійності та безпомилковості програмного забезпечення на додаток до традиційних методів тестування та налагодження все частіше використовуються формальні методи верифікації [13]. Знання формальних моделей і методів верифікації, вміння використовувати ці методи для забезпечення високої якості програмних систем, що розробляються, стають необхідними для професійних програмістів. Застосовувані на даний час різновиди тестування програмного забезпечення, такі як: доказ теорем, статичний аналіз, динамічна верифікація, ModelChecking не дозволяють проводити верифікацію паралельних програм, що представлені відповідними семантико-числовими специфікаціями. Верифікація за допомогою одиниць вимірювання фізичних величин є складним і неавтоматизованим процесом, а верифікація програмних засобів інформаційних управляючих систем керування підприємством – процесом на порядок складнішим. Таким чином виникає задача вдосконалення існуючих методів верифікації програмних засобів інформаційних управляючих систем [1–3, 17].

Актуальність проблеми розробки концепцій, принципів побудови інформаційних технологій (ІТ) та методів розробки інформаційних управляючих систем (ІУС) дійсно підтверджується рядом проєктів провідних

компаній у світі. Ось деякі з них:

– Програма ESPRIT (European Strategic Program for Research and Development in Information Technology): програма, що є частиною зусиль Європейського Союзу для розвитку IT-сектору, фінансує наукові дослідження та розробки у сфері IT.

– Прискорена стратегічна комп'ютерна ініціатива США (Accelerated Strategic Computing Initiative), що фокусується на розробці та застосуванні високопродуктивних обчислювальних систем для різних цілей, включаючи наукові дослідження та оборонні потреби.

– Проєкти фірм HP, IBM, Microsoft, які постійно працюють над розробкою нових інформаційних технологій та продуктів, що вимагає розвитку нових концепцій та методів.

– Спільний проєкт NASA з Центром передових технологій університету штату Вірджинія «Среда з інтелектуальним синтезом», ISE (Intelligent Synthesis Environment): Цей проєкт відзначається розробкою інтелектуальних систем, що вміють синтезувати та аналізувати інформацію.

Ці програми та проєкти свідчать про важливість розробки нових концепцій, принципів та методів у сфері IT та розвитку сучасних інформаційних технологій ІУС з використанням паралельних обчислень під час вирішення складних завдань у різних сферах життя, включаючи науку, технології та оборону [2, 6–9, 13].

Паралельні обчислення застосовуються в багатьох галузях, включаючи наукові дослідження, фінанси, медицину, виробництво та багато інших. До теперішнього часу методи паралельної обробки інформації є широко вивченими та дослідженими у літературі [5–7, 9, 10– 16, 18]. Вони включають в себе багато методів, технологій та підходів. Аналіз існуючих літературних джерел показує, що сучасні завдання в області обчислювальної науки та інженерії вимагають все більшого використання паралельних обчислень для вирішення складних завдань. Відзначимо, що Україна має активну спільноту дослідників і фахівців у галузі обчислювальної техніки та паралельних

обчислень, які проводять дослідження та впровадження паралельних обчислень у різні галузі.

Сучасні системи обробки інформації та управління визначаються необхідністю надавати результати обробки даних у реальному часі з високими вимогами до тактової частоти обробки. Ця проблема набуває особливого значення в контексті перспективних систем обробки радіолокаційної інформації, керування польотами літальних об'єктів, управління швидкоплинними технологічними процесами (наприклад, на виробництві матеріалів різного застосування, в тому числі високотемпературного), телекомунікаційними та робототехнічними системами. Тому існує великий потенціал для подальшого розвитку інформаційних технологій і забезпечення відповіді на зростаючі вимоги в цій галузі.

Семантична верифікація на основі семантико-числових специфікацій є одним з небагатьох методів, який враховує широкий склад факторів при верифікації паралельних програм. У взаємодії зі звичайним тестуванням, компіляційною та декомпіляційною верифікацією кількість помилок в розробляємих паралельних програмах для таких ІУС істотно знижується. Вочевидь, що з практичної точки зору доцільно продовження робіт в напрямку вдосконалення існуючих технологій верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення ефективності верифікації за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації на основі семантико-числових специфікацій [17, 19].

Викладене вище обумовлює актуальність рішення ***науково-прикладної задачі*** створення інформаційної технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення ефективності верифікації та тестування за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації семантико-числових специфікацій.

Зв’язок роботи з науковими програмами, планами, темами. Тематика дисертаційної роботи пов'язана з дослідженнями:

– Участь у НДР «Моделі інформаційних процесів та методи їх обробки» за 2016–2020 рр.(ДР № 0116U003141), у якості виконавця.

– Участь у НДР «Виконання завдань Перспективного плану розвитку наукового напряму «Технічні науки» Харківського національного університету імені В. Н. Каразіна» (ДР № 0121U11306883) за 2021–2022 рр.. НТЗ Виконання завдань (розділ 2) у якості виконавця.

– Участь у НДР «Моделювання інформаційних процесів у складних і розподілених системах». Науково-дослідницька робота без фінансування, термін виконання з 1.03.2021 р. до 31.12.2023 р. (ДР № 0121U109183) у якості виконавця.

Мета і задачі дослідження. Основною ***метою*** дисертаційної роботи є підвищення ефективності верифікації програмного забезпечення інформаційних управляючих систем на основі інформаційної технології верифікації паралельних часопараметризованих програм за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації семантико-числових специфікацій.

Для досягнення даної мети як рішення поставленого наукового завдання в цілому, були сформульовані ряд ***задач***. До їхнього числа належать.

1. Аналіз проблем розробки і верифікації паралельних програм інформаційних управляючих систем.
2. Удосконалення методу компіляційної верифікації паралельних програмних засобів ІУС.
3. Удосконалення методу декомпіляційної верифікації паралельних програмних засобів ІУС.
4. Удосконалення методу семантичної верифікації часопараметризованих мультипаралельних програм.
5. Розробка технології семантико-числової верифікації часопараметризованих мультипаралельних програм ІУС.

6. Практичне підтвердження працездатності та вірогідності розроблених методів та інформаційної технології семантико-числової верифікації часопараметризованих мультипаралельних програм ІУС.

Об'єкт дослідження – процеси верифікації верифікації програмного забезпечення інформаційних управляючих систем.

Предмет дослідження – моделі та методи верифікації паралельних програм в інформаційних управляючих системах, що застосовують методи компіляційної, декомпіляційної та семантичної верифікації.

Методи дослідження. Теоретичні основи роботи базуються на принципах і методах системного аналізу, математичного та імітаційного моделювання. Як математична основа для розробки методів компіляційної, декомпіляційної, семантичної верифікації та технології семантико-числової верифікації часопараметризованих мультипаралельних програм інформаційних управляючих систем, що задовольняє заданим вимогам й обмеженням, використовуються теорія складних систем, теорія множин, теорія графів та апарат часових паралельних граф-схем (під час розроблення методів верифікації та інформаційної технології); методи імітаційного моделювання (під час оцінювання коректності та достовірності часових паралельних моделей).

Наукова новизна отриманих результатів полягає в наступному.

1. ***Вперше розроблено*** технологію семантико-числової верифікації часопараметризованих мультипаралельних програм, особливістю якої є застосування компіляційної, декомпіляційної та семантичної верифікації структур семантико-числової специфікації, реалізація яких гарантує підвищення ефективності верифікації паралельних програм для інформаційних управляючих систем.

2. ***Дістав подальшого розвитку*** метод компіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, що містить етапи верифікації структур семантико-числової

специфікації послідовної програми, часової паралельної моделі програми з урахуванням особливостей обраної раціональної сукупності методів паралельної обробки інформації, часопараметризованої паралельної програми та перевірку відповідності показників ефективності заданим вимогам і обмеженням;

3. ***Дістав подальшого розвитку*** метод декомпіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, головними етапами якого є декомпіляція структур часової моделі та перевірка еквівалентності декомпіляційної структури та СЧС вхідної послідовної програми, синтез текстової і СЧС специфікації паралельної програми та перевірка еквівалентності текстової/графічної специфікації програми і тексту послідовної програми, для зменшення часу на верифікацію ураховуються особливості обраної сукупності методів паралельної обробки інформації;

4. ***Дістав подальшого розвитку*** метод семантичної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, який містить етапи перевірки семантичної коректності вхідної послідовної програми задачі, статичної мультипаралельної програми, логічної еквівалентності часової мультипаралельної програми та вхідної послідовної програми задачі.

Особистий внесок здобувача. Дисертаційне дослідження виконано здобувачкою самостійно, усі сформульовані в ньому положення та висновки з рекомендаціями обґрунтовані на основі особистих досліджень авторки. Для аргументації окремих положень використані праці інших науковців, на які зроблені посилання. В індивідуальних наукових працях застосовано лише авторські ідеї та розробки.

Дисертантка брала активну участь у наукових дискусіях, семінарах, підготовці наукових статей, опублікованих за темою дисертації, успішно доповідала результати досліджень на міжнародних конференціях.

У працях [1–3] здобувачці належить участь у процесі перевірки синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації, обробка результатів, опис методів верифікації паралельних програм, експертизи програмного забезпечення, статичного аналізу, динамічних та формальних методів верифікації, а також написання частини тексту. У публікаціях [4–6] здобувачка брала участь у розробленні, підготовці та верифікації СЧС. Дисертантка провела обробку та перевірку синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації, брала участь в обговоренні отриманих результатів та написанні тексту. В роботах [7–8] здобувачка описує технологію верифікації паралельних часопараметризованих програм інформаційних управляючих систем на основі семантико-числових специфікацій. У монографії [9] здобувачка брала участь у побудові моделі технології семантико-числової верифікації часопараметризованих мультипаралельних програм, обговоренні результатів застосування методів формальної верифікації паралельних часопараметризованих моделей, участь в обговоренні та інтерпретації отриманих результатів, написання тексту. У публікаціях [10–20] дисертантка брала участь у підготовці програмного забезпечення модернізації існуючих ІУС в верифікації програмного забезпечення супроводження літальних апаратів, обробці експериментальних даних, обговоренні результатів, написанні тексту. Публікації [21–28] здобувачка брала участь у проведенні верифікації, обробці даних, , написанні тексту матеріалів дисертації та доповіді на конференціяхї.

Практичне значення отриманих результатів.

При розробці та модернізації існуючих ІУС, в верифікації програмного забезпечення, яке реалізує вдосконалені комплекси алгоритмів (окремі алгоритми) управління та обробки інформації, в оцінці можливості їх реалізації в умовах заданого циклу (обмежень на час виконання), у виробленні рекомендацій щодо оптимізації конфігурації апаратних засобів їх паралельного виконання.

Результати дисертаційного дослідження були використані при верифікації програмного забезпечення обладнання, що використовується для виробництва графітових матеріалів фірмою «ЛЕГ» (ТОВ фірма «ЛЕГ», м. Харків), що дозволило скоротити час на виробництво графітових матеріалів різного застосування, в тому числі високотемпературного (акт впровадження від 06.12.2022 р.). В результаті збільшився ККД використання пресового обладнання, верстатів, що виготовляють вироби з вуглецевих матеріалів, нагрівачів в високотемпературних печах, які працюють у вакуумі або захисному середовищі, для спікання твердих сплавів, плавки кварцу. Також технологію семантико-числової верифікації часопараметризованих мультипаралельних програм було використано у ТОВ «НТЦ «Вуглець» для верифікації програмного забезпечення управління роботою теплових вузлів для вакуумних печей (типу ВВНК) (акт впровадження від 15.08.2023 р.), при використанні якого вдалося досягти вдосконалення технології виробництва виробів з вуглецевих матеріалів, зменшення часу на розробку окремого виробу та економії електроенергії.

Також за рахунок використання методів та моделей паралельної часопараметризованої обробки даних ІУС здійснюється підвищення швидкості обробки інформації обчислювальними системами в процесі керування аеродінамічними об’єктами різних класів та призначень, що засвідчують патенти на корисну модель [10–20].

Отримані в процесі дослідження теоретичні положення, моделі й методи можуть бути використані при поглибленому вивченні інформаційних комп'ютерних технологій, паралельних обчислювальних систем, методів розробки й оцінки ефективності паралельної обробки даних у паралельних обчислювальних системах, методів верифікації паралельних програм в існуючих, а також перспективних інформаційних технологіях. Отримані результати використовують у процесі викладання навчальних курсів «Паралельні системи й обчислення», «Засоби програмування багатопроцесорних систем», «Паралельне

програмування з OpenMP» у Харківському національному університеті імені В. Н. Каразіна.

Апробація результатів дисертації. Основні теоретичні положення, висновки і пропозиції, які містяться в дисертації, обговорювалися та були схвалені на засіданнях кафедри теоретичної та прикладної системотехніки Харківського національного університету імені В.Н. Каразіна. Ключові положення дослідження оприлюднені у доповідях на науково-технічних конференціях всеукраїнського та міжнародного рівнів (2016–2021 роки).

– Міжнародній науково-технічній конференції «Комп'ютерне моделювання в наукоємних технологіях» КМНТ–2014 (Україна, м. Харків, Харківський національний університет ім. В. Н. Каразіна, 2014).

– Міжнародній науково-технічній конференції «Проблеми автоматизації» (Україна, м. Черкаси, 12–13 листопада 2015 р.)

– Четвертій міжнародній науково-технічній конференції «Проблеми інформатизації». (Черкаси – Баку – Бельсько-Бяла – Полтава, 3–4 листопада 2016 р).

– Сьомій міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікацій-них технологій та засобів управління». (Полтава – Баку – Кіровоград – Харків, 2017р).

– П'ятій міжнародній науково-технічній конференції «Проблеми інформатизації». Проблеми інформатизації. Тези доповідей п'ятої міжнародної науково-технічної конференції ». (Черкаси – Баку – Бельсько-Бяла – Полтава, 13–15 листопада 2017 р).

– Восьмій міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління». (Полтава –Баку–Харків–Жиліна, 26–27 квітня 2018р).

– Міжнародній науково-технічній конференції «Комп'ютерне моделювання в наукоємних технологіях» КМНТ–2018 (Україна, м. Харків, Харківський національний університет ім. В. Н. Каразіна, 2018).

– Міжнародній науково-технічній конференції «Комп’ютерне моделювання в наукоємних технологіях» КМНТ–2020р (Україна, м. Харків, Харківський національний університет ім. В. Н. Каразіна, 2020р).

– Міжнародній науково-технічній конференції «Комп’ютерне моделювання в наукоємних технологіях» КМНТ–2021р (Україна, м. Харків, Харківський національний університет ім. В. Н. Каразіна, 2021р).

Публікації. Основні теоретичні положення і висновки дисертації викладені у 28 наукових працях, з яких 8 статті у наукових фахових виданнях України та ті, що входять до міжнародних наукометричних баз [1–8], 1 монографія [9], 11 патентів [10–20] та 8 тез наукових доповідей [21–28].

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і 3 додатків. Загальний обсяг дисертації становить 163 сторінки: у тому числі анотації на 10 сторінках, зміст на 2 сторінках, основний текст на 122 сторінках, список використаних джерел із 101 найменуваня на 11 сторінках та чотири додатки на 20 сторінках. Робота містить 15 таблиць та 37 рисунків, з яких 5 на окремих 6 сторінках.

РОЗДІЛ 1.
АНАЛІЗ ШЛЯХІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ТЕХНОЛОГІЙ ВЕРИФІКАЦІЇ ПАРАЛЕЛЬНИХ ПРОГРАМ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ.

1.1 Аналіз проблем розробки і верифікації паралельних програм інформаційних управляючих систем.

Наш час характеризується збільшенням кількості нових перспективних напрямків розвитку науки, техніки, виробництва й управління: дослідження космосу, атомної енергетики, телекомунікації, криптографії, розвитку медицини й генної інженерії, «інтелектуальної» робототехніки, управління складними системами в реальному часі та ін. Основою успішного рішення цих задач є застосування високоефективних інформаційних технологій (ІТ) та інформаційних і управляючих систем (ІУС), невід'ємними компонентами яких є програмне забезпечення, технічне забезпечення та бази даних. Однак, необхідність обробки великого обсягу даних, які надходять до ІУС в реальному часі, є однією з ключових проблем. Завданням таких систем є обробка цих даних вчасно та ефективно, щоб приймати рішення та виконувати відповідні дії.

Сучасні ІУС можуть мати складну архітектуру [20]. На Рис. 1.1. показана архітектура такої складної ІУС управління підприємством. Система має п'ять рівнів ієрархії 1…5.

Перший рівень називають «польовим» і він включає в себе датчики і виконавчі органи, що встановлені на об'єкті управління (ОУ).

Другий рівень – рівень промислових контролерів PLC, що організовують локальні замкнуті системи управління окремими елементами або процесами об'єкта управління.

Третій рівень називають операторським. На цьому рівні, реалізованих на основі комп'ютерів (в тому числі і промислових), організовується людино-машинний інтерфейс HMI оператора, обробка, зберігання і передача

інформації про стан і функціонування ОУ на вищі рівні ієрархії системи і отримання від них завдань керування ОУ. Рівні 4 і 5 є організаційними.

Четвертий MES – *рівень* керування виробничими процесами.

П'ятий ERP – *рівень* планування і організації виробництва.

Всі рівні об'єднані між собою відповідними інформаційними мережами, вибір яких визначається кількістю і часом обміну інформацією. Основною частиною такої складної ІУС (в принципі і ІУС любої складності) є перших три рівні, а власне підсистема, що керує окремими об'єктами і процесами, реалізується в основному першим і другим рівнями.

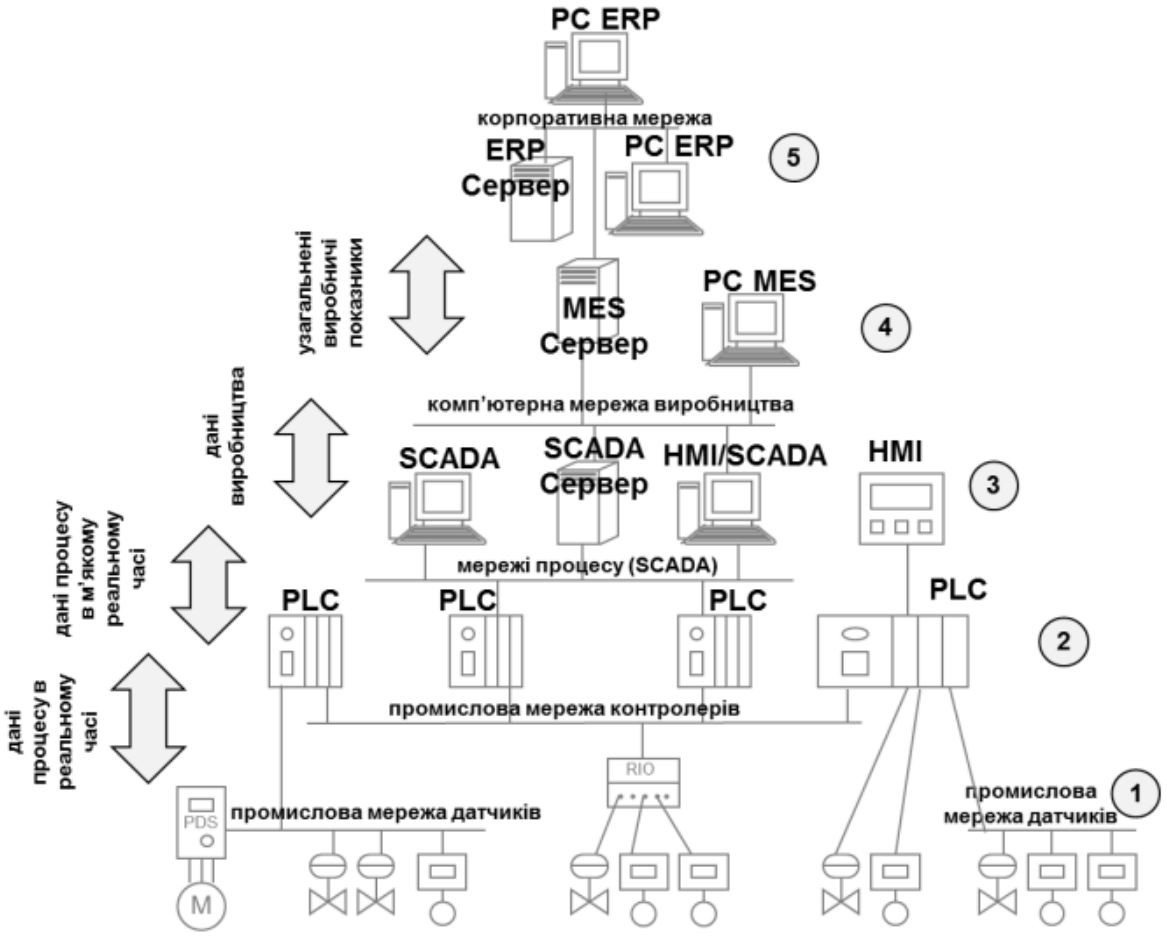


Рис. 1.1. Архітектура складної ІУС управління підприємством.

Таким чином, виникає необхідність в розробці нового програмного забезпечення ІУС на основі раціональної сукупності методів паралельної обробки інформації залежно від наявних вимог і обмежень. Це, в свою чергу,

приводить до потреби в верифікації правильності обробки вхідних даних. Важливо переконатися, що система працює коректно та надійно, інакше можуть виникнути серйозні наслідки, особливо в критичних для життя ситуаціях, таких як медична діагностика, автономні автомобілі або керування авіаційними системами.

На даний час основними способами отримання гарантій відповідності системи призначення є тестування та верифікація [1, 2, 4, 7, 9, 10, 17].

Тестування (*Testing*) визначається як «будь-який вид діяльності, в рамках якої шляхом реального виконання будь-яких завдань (тестових прогонів) перевіряється робота або системи в цілому, або її складової частини». Прийнято розрізняти два види тестування – статичне та динамічне тестування. Статичне тестування визначається як «тестова діяльність, пов'язана з аналізом результатів розробки програмного забезпечення шляхом перевірки програмних кодів, наскрізного контролю та перевірки програми без запуска машиною, тобто «перевірки за столом». Динамічне тестування «складається з прогону програми та порівняння її фактичної поведінки з очікуваною» поведінкою.

Під верифікацією (*verification*) деякого об’єкта/системи розуміється «повний набір перевірок, яким піддається система для отримання гарантій її відповідності своєму призначенню». До таких перевірок можуть входити жорсткий набір функціональних тестів, контроль пропускної спроможності, перевірка надійності тощо. Для підвищення конструктивності викладу надалі розумітимемо під верифікацією «повну об’єктивну перевірку відповідності об’єкта певної точно визначеної специфікації». Зазначимо, що, на відміну від верифікації під «підтвердженням правильності» (*validation*), розуміється «деяка суб’єктивна оцінка ймовірної відповідності призначенню в передбачуваних умовах функціонування» [17, 19]. Під «точно визначеною специфікацією», задоволення якій є критерієм коректності об’єкту проєктування, що верифікується, далі розуміється наступний склад даних:

– клас об’єкту проєктування – програмний, апаратний чи апаратно-програмний;

– функціональність – склад завдань, які вирішуються об’єктом проєктування та специфікованих текстами послідовних програм;

– склад вимог і/або обмежень, що пред’являються до об’єкту (наприклад, продуктивність, пропускна здатність, час виконання завдання, тактова частота обробки даних, складність/вартість, надійність/достовірність тощо).

Забезпечення верифікації інформаційних управляючих систем у реальному часі допомагає уникнути помилок, що можуть призвести до небезпеки або фінансових втрат. Також це сприяє покращенню якості ІУС, забезпечуючи високий рівень впевненості у їхній працездатності в будь-яких ситуаціях [5, 11, 20]. Важливо переконатися, що ПЗ працює коректно, інакше можуть виникнути серйозні наслідки, особливо в критичних для життя ситуаціях, таких як медична діагностика, автономні автомобілі або керування авіаційними системами.

Загальноприйнятий поділ методів верифікації можна представити в вигляді діаграми на Рис. 1.2 [10]:

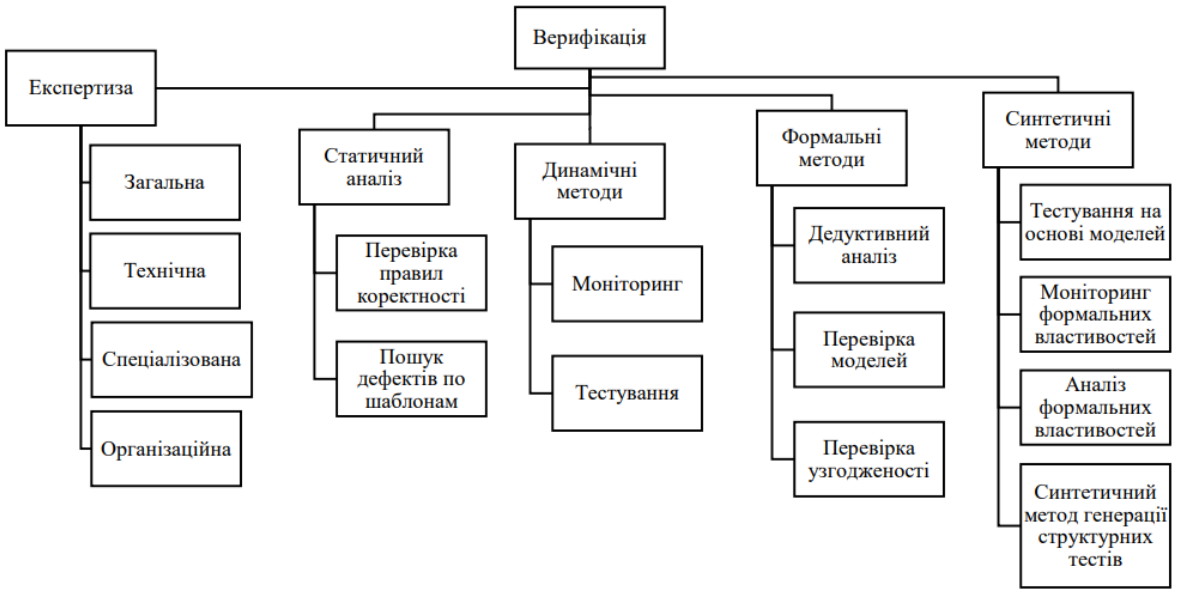


Рис. 1.2. Загальноприйнятий поділ методів верифікації.

Експертизою програмного забезпечення (ПЗ) називають всі методи верифікації, в яких оцінка артефактів життєвого циклу ПЗ виконується людьми. Перевагою даного методу є те, що при його використанні

виявляються в середньому 50–90% помилок [10, 19, 20]. Але цей метод має і недоліки. Пошук помилок, оцінка і аналіз властивостей ПЗ людиною (зазвичай це група 2–5 осіб). Потрібні справжні експерти, програмісти з досвідом роботи не менше 10 років.

Статичний аналіз– аналіз без виконання програми. Методи статистичного аналізу можна розділити на два види: контроль того, що всі формалізовані правила коректності побудови цих артефактів виконані, та пошук типових помилок і дефектів в них на основі деяких шаблонів [21]. Часто інструменти статичного аналізу використовують обидва типи перевірок. Статичний аналіз можна вважати найбільш широко застосовуваним методом верифікації. Перевірені на практиці правила коректності коду або шаблони типових помилок переносяться в середовища розробки.

Переваги статичного аналізу:

- Автоматичний аналіз багатьох шляхів виконання одночасно.
- Виявлення помилок, що проявляються лише на одиничних шляхах виконання або на незвичайних вхідних даних.
- Можливість аналізу на неповному наборі вхідних файлів.
- Відсутність накладних витрат під час виконання програми.

Недоліки даного методу:

- Велика кількість помилкових спрацювань.
- Необхідна ручна перевірка результатів роботи, що вимагає значних часових, людських та матеріальних ресурсів.

Динамічні методи верифікації використовують результати реальної роботи програмної системи або її прототипів, щоб перевіряти відповідність цих результатів вимогам і проєктним рішенням.

Існує два основних види динамічних методів верифікації: моніторинг, в рамках якого йде тільки спостереження, запис і оцінка результатів роботи ПЗ при його звичайному використанні, і тестування, при якому ПЗ виконується в рамках заздалегідь підготовлених сценаріїв. Перевагою даного методу є висока точність виявлення помилок. А недоліками – необхідно мати набір

вхідних даних та середовище виконання, а також високі вимоги до ресурсів [10, 19].

Формальні методи верифікації. Їх відмітною особливістю є можливість проведення пошуку помилок на математичній моделі, без звернення до фізичної реалізації, що в деяких випадках досить зручно і економічно. Для проведення аналізу формальних моделей застосовуються специфічні техніки, такі як дедуктивний аналіз, перевірка моделей, перевірка узгодженості. На жаль, для побудови таких моделей завжди необхідно виходити так само з коректності та адекватності моделі ПЗ. Лише після правильної побудови цієї моделі можна автоматично проаналізувати деякі з її властивостей. Проте, в більшості випадків для ефективного аналізу від фахівців будуть потрібні глибокі знання математичної логіки і алгебри і деякого набору навичок роботи з цим апаратом [21, 22].

В останні роки активно розробляються інструменти автоматичної генерації тестів на основі коду, які використовують додаткові джерела інформації. В якості таких джерел виступають статичний аналіз коду, формальний аналіз, моніторинг виконання раніше побудованих тестів і т.п. Оскільки в інструментах цього типу використовується зазвичай 3–4 техніки різних типів, методи, що лежать в їх основі, винесені в окремий різновид синтетичних методів верифікації. Синтетичні методи верифікації поєднують підходи декількох типів – статичний аналіз, формальний аналіз властивостей ПЗ, тестування. Деякі з таких методів породили в останні 10–15 років самостійні галузі досліджень, в першу чергу, тестування на основі моделей і моніторинг формальних властивостей. Переваги та недоліки синтетичних методів визначаються комбінацією методів верифікації, які входять до її складу [21, 23–25].

Методи формальної верифікації програмного забезпечення комп'ютерних систем дозволяють гарантувати перевірку виконання моделлю системи верифікованих властивостей. В даний час ці методи активно розвиваються в напрямку зниження загальної вартості формальної перевірки,

підтримки сучасних концепцій програмування і мінімізації «ручної» праці при переході від моделі системи до її реалізації. Кожен метод верифікації використовується в конкретному класі випадків в залежності від поставленої мети. Найактуальнішими, найбільш корисними та продуктивними можна вважати синтетичні методи верифікації ПЗ, оскільки вони так чи інакше намагаються поєднати переваги різних підходів до верифікації, зменшуючи їх недоліки. В даний час досягнуті значні успіхи в розробці таких методів і впровадженні їх у практику промислової розробки ПЗ.

На сьогодні сучасні напрями перевірки правильності програм пов’язані зі формальною специфікацією, методами доведення, верифікацією і тестуванням.

Формальна специфікація – це процес визначення поведінки програми за допомогою математичних моделей і формалізованих правил. Вона дозволяє точно визначити очікувані результати та поведінку програми для різних вхідних даних. Формальні специфікації допомагають уникнути двозначних інтерпретацій, що можуть виникнути при письмі звичайних текстових описів.

Методи доведення використовують математичні та логічні аргументи для переконання у правильності програми. Формальне доведення може включати математичні докази, індукцію, дедукцію та інші логічні методи для підтвердження того, що програма працює згідно зі специфікацією.

Комбінування цих методів дозволяє забезпечити більш високу надійність та правильність програм. Відповідно до конкретних потреб і обмежень проєкту, розробники можуть використовувати один або декілька з цих підходів для перевірки програм перед їхнім впровадженням.

Верифікацією комп'ютерних систем називається діяльність, спрямована на з'ясування їх правильності або помилковості. Діяльність ця може приймати різні форми: інспекція коду; статичний аналіз (пошук типових помилок); тестування (запуск програми на прикладах і перевірка коректності результатів); імітаційне моделювання (створення здійсненною моделі системи та її дослідження в спеціальному оточенні); формальна верифікація (побудова

логічної моделі системи та її аналіз засобами математичної логіки). Підходів багато, але жоден з них не здатний гарантувати коректність дійсно складних проєктів; кращі результати, як показує практика, досягаються при спільному використанні різних методів. Суть формальних методів полягає в створенні математичних моделей програм і вимог і в логічному аналізі відповідності між побудованими моделями. На сьогодні, формальні методи – це фундамент, на якому стоїть будівля програмної інженерії. Слід зазначити, що формальні методи давно вийшли за межі академічної спільноти і стали частиною вітчизняної індустрії, а так само частиною процесу розробки інформаційних управляючих систем.

1.2 Аналіз методів доведення специфікацій програм.

Відомо багато методів доведення специфікацій програм. Наведемо поширені [24–29].

Метод Флойда заснований на знаходженні умов для вхідних і вихідних даних і полягає у виборі контрольних точок у програмі, яка доводиться, таким чином, щоб шлях проходження перетинав хоча б одну контрольну точку. Для цих точок формулюються твердження про стан і значення змінних у них (для циклів ці твердження повинні бути істинними при кожному проходженні циклу – інваріанта).

Кожна точка розглядається для індуктивного твердження того, що формула залишається істинною при поверненні програми в цю точку, і залежить не тільки від вхідних і вихідних даних, а й від значень проміжних змінних. На основі індуктивних тверджень і умов на аргументи програми створюються твердження з умовами перевірки правильності цієї програми в окремих її точках. Для кожного шляху програми між двома точками встановлюється перевірка на відповідність умов правильності і визначається істинність цих умов при успішному завершенні програми на даних, що задовольняють вхідні умови.

Метод Хоара – це вдосконалений метод Флойда, заснований на аксіоматичному описі семантики мови програмування початкових програм.

Кожна аксіома описує зміну значень змінних за допомогою операторів цієї мови. Формалізація операторів переходу і викликів процедур забезпечується за допомогою правил виводу, що містять у собі індуктивні вислови для кожної точки і функції початкової програми.

Метод Маккарті полягає у структурній перевірці функцій, що працюють над структурними типами даних, структур даних і шляхів переходу під час символьного виконання програм. Ця техніка складається з моделювання виконання коду з використанням символів для змінних даних. Тестова програма має вхідний стан, дані і умови її виконання. Програма, що виконується, розглядається як серія змін станів. Саме останній стан програми вважається вихідним станом і, якщо його одержали, то програма вважається правильною. Даний метод забезпечує високу якість початкового коду.

Метод Дейкстри пропонує два підходи до доведення правильності програм. Перший підхід заснований на моделі обчислень, що оперує історіями результатів обчислень програми, аналізом шляхів проходження і правил оброблення великого об'єму інформації. Другий підхід базується на формальному дослідженні тексту програми за допомогою предикатів першого порядку. У процесі виконання програма одержує деякий стан, який запам'ятовується для подальших порівнянь.

Основу методу становить математична індукція, абстрактний опис програми і її обчислення.

Метод Р. Андерсена. Ґрунтується на припущенні, що правильність програм можна доводити як теореми в математиці. В основу методу доведення покладено апарат математичної індукції, суть якого полягає у неформальному доведенні правильності, унаслідок чого можуть допускатися помилки доведення, до того ж не всі помилки в програмі виявляються. Цей метод слід розглядати як системний для перевірки правильності програми по коду. Для доведення цими методами застосовується теоретикомножинний підхід, методи верифікації створюваних програм на етапах життєвого циклу, а також

тестування отриманих програм на множині тестових даних для встановлення їх правильності. Переваги і недоліки доведення правильності. Під час конструювання ручного або автоматичного доведення виявляються помилки в коді алгоритму. Техніка доведення додатково забезпечує формальне розуміння програми, оскільки перевіряється основна логічна структура. Регулярне використання цього підходу приводить до більш точного і строгого специфікування даних, структур даних і алгоритмічних правил. Проте точність досягається нелегко. Багато хто відмовляється від доведення, оскільки, наприклад, алгоритм пузиркового сортування набагато простіший, ніж його логічний опис і доведення. Крім того, великі і складні компоненти можуть включати логічні діаграми, перетворювачі та верифікацію великої кількості частин. Наприклад, програми для оброблення нечислових даних можуть бути складнішими для розуміння логіки, ніж для числових. Паралельне оброблення також важко перевіряти, структури даних дають результат лише після складного перетворення операторів для такого виконання. Техніка доведення ґрунтується лише на перевірці того, як вхідні твердження трансформуються у вихідні відповідно до логічних правил. Доведення коректності програми в логічному сенсі ще не означає, що в програмі немає помилок. Справді, ця техніка не розпізнає помилок у проєкті, в інтерфейсах з іншими компонентами, в інтерпретації специфікацій, у синтаксисі та семантиці мов програмування або в документації. Техніка логічного доведення ігнорує структуру і синтаксис мови програмування, у яких тестові програми виконуються. Імовірний випадок, коли ця техніка доводить, що спроєктовані компоненти є правильними, але вони не завжди виконуються. Інші техніки беруть до уваги характеристики мов.

1.3. Інструментальні засоби верифікації програм з підтримкою паралельних обчислень.

Універсальні мови специфікації – *VDM, Z, RAISE, Spec#* мають загально математичну основу з такими засобами [30–32]:

1) логіки першого порядку, включаючи квантори;

2) арифметичні операції;

3) множини і операції над множинами;

4) описи послідовностей (кортежів, списків) і операції над ними;

5) описи функцій і операцій над ними;

6) описи деревоподібних структур;

7) засоби побудови моделей областей;

8) процедурні засоби мов програмування (оператори присвоювання, циклу, вибору, виходу);

9) операції композиції, аргументами і результатами яких можуть бути функції, вирази, оператори;

10) механізм конструювання нових структур даних.

Мови специфікації предметних областей у програмуванні:

1) специфікації доменів;

2) описи взаємодій і паралельного виконання;

3) специфікації мов програмування і трансляторів;

4) специфікації баз даних і знань;

5) специфікації пакетів прикладних програм тощо.

Мови специфікації специфіки доменів DSL (*Domain Specific Language*) призначені для формалізованого опису задач в термінах предметної області, що підлягає моделюванню. Ці мови можна підрозділити на зовнішні і внутрішні. Зовнішні мови (типу *UML, OWL* та ін.) за рівнем вищі мов програмування і відповідають, наприклад, предметно-орієнтованої мові *DSL*, яка використовується для подання абстрактних понять і задач ПР. Їхній опис трансформується до понять деякої внутрішній мові або мови програмування спеціальними генераторами або текстовими редакторами. Внутрішні мови – мови опису специфічних задач обмеженим синтаксисом і семантикою потребують препроцесорів для перебудови цього опису до базової мови програмування.

Специфікації опису взаємодій і паралельного виконання окремих процесів систем ПЗ також добре подаються мовами DSL, наприклад, подібно діаграм UML.

Мови програмування предметної області, доповнені засобами і механізмами технологій.

Мета програмування є ефективним засобом автоматизації специфікацій розроблених програм і в даний час знаходять широке застосування у галузі інформаційних технологій.

Формальні мови специфікації мов програмування спочатку застосовувалися при розробленні трансляторів. Так зазвичай синтаксис мови програмування описувався КС-граматиками у формі Бекуса–Наура. Такого типу мови є метамовами. Для специфікації семантики мов програмування використовуються формалізми рівностей. Техніка опису мов програмування базується на атрибутних граматиках і абстрактних типах даних з використанням денотаційних, алгебраїчних і атрибутних підходів. Як мови специфікації трансляторів, а також систем реального часу, де правильність і точність виконання програм є головними, використовують мови *Z, VDM, RAISE* [30].

Мови специфікації з орієнтацією на засоби програмування базуються на рівностях і підстановках з операційною семантикою (Лісп, Рефал); логічні мови; мови операцій (APL) над послідовностями і матрицями; табличні мови; мережі, графи та ін. Мова логіки предикатів використовується для запису передумов і постумов, інваріантів і процесу верифікації (наприклад, Пролог).

Для визначення семантики рівності застосовують денотаційний, операційний і аксіоматичний опис. Операційна семантика пов’язана з підстановками (заміна, продукція) і визначається в термінах операцій, що призводять до обчислень алгоритмів. При цьому фіксується порядок і динаміка виконання операцій програми.

У денотаційному підході до семантики надається перевага статичному опису об’єктів у термінах математичних властивостей, а у аксіоматичному –

специфікації властивості об’єктів у рамках деякої логічної системи, що містить у собі правила виведення формул та/або інтерпретацій.

Окрім наведеної класифікації мов специфікацій, існують інші. Наприклад, можлива класифікація специфікацій за *способом виконання*:

– виконувана (executable);

– алгебраїчна (algebraic);

– сценарна (use case or scenarios);

– в обмеженнях (constraints).

Виконуючі специфікації припускають розробку прототипів систем для досягнення встановленої мети (*VDM, SDL, RSL*).

Алгебраїчні специфікації та мови *SDL, RSL* містять у собі механізми опису аксіом і тверджень, які призначені для доведення специфікованих програм.

Сценарні специфікації (*UML*) дозволяють описувати різні способи можливого застосування системи.

Програмування в обмеженнях використовують перед- і постумови для опису даних, операцій, інваріантів даних програм, що доводяться.

Мова формальної специфікацій – VDM

Мова специфікації VDM (*Vienna Development Method*) була розроблена у віденській лабораторії компанії IBM і призначалася для опису мов типу ПЛ/1, трансляторів і систем із складними структурами даних. У мові специфікується правильна програма і набір тверджень для її доведення [31, 32].

Мова формальної специфікації – RAISE.

Мова *RAISE і RSL*-специфікація (*RAISE Specification Language*) були розроблені в 80-роках XX ст. як результат попереднього дослідження формальних методів верифікації програм і поповнення їх новими можливостями щодо доведення. Метод містить у собі нотації, техніку і інструменти для формального опису (RSL, C++ і Паскаль) програм і доведення їх правильності [33–35].

До складу мови RSL входять абстрактні параметричні типи даних (специфікації, алгебри) і конкретні типи даних (модельно-орієнтовані), підтипи, операції для завдання послідовних і паралельних програм. Тобто ця мова надає аплікативний і імперативний стиль специфікації абстрактних програм, а також формальне конструювання окремих програм в інших мовах програмування і апарат доведення їх правильності. Синтаксис цієї мови близький до синтаксису мов С++ і Паскаль.

У мові RSL є абстрактні типи даних і конструктори складних типів даних, такі як добуток (*product*), множини (*sets*), списки (*list*), відображення (*map*), записи (*record*) і т.п.

Концепторна мова специфікації.

Для постановки складних математичних задач (підсумовування нескінченних рядів, теоретико-множинних операцій з нескінченними множинами тощо) і задач штучного інтелекту (ігри, розпізнавання образів тощо) з метою їх формального опису запропонована *загально математична процедурна мова,* а саме, *концепторна мова (КМ)* [32, 36]. У цій мові процес опису складного завдання проводиться шляхом обґрунтування розв’язку задачі з математичної точки зору, потім формального опису постановок задач і, нарешті, переходу до алгоритмічного опису. Концепторна мова містить у собі декларативні й імперативні засоби теорії множин Цермело–Френкеля. Ядро цієї мови містить набір елементів (типи, вирази, оператори) і засоби визначення нових типів, виразів і операторів.

Декларативні засоби КМ – це типізована логіко-математична мова для опису *виразів* і структуризації множин значень (денотат). Вирази складаються з термів і формул, терми позначають об’єкти ПЗ, а формули – твердження про об’єкти і відношення між ними.

Логіко-алгебраїчні специфікації КМ призначені для специфікації ПЗ, що задаються у вигляді алгебраїчної системи, за допомогою відповідних носіїв, сигнатури і трьох принципів. *Перший принцип* – логіко-алгебраїчна специфікація ПЗ і уточнення понять ПЗ, *другий принцип* – опис властивостей

ПЗ у вигляді аксіом, які формулюються мовою предикатів першого порядку і хорновських атомарних формул, і, нарешті, *третій принцип* – це визначення термальних моделей з основних термів специфікації. Засоби КМ використовуються при формальній специфікації *поведінки дискретних систем.* Для опису властивостей апаратно-програмних засобів динамічних систем застосовуються логіко-алгебраїчні специфікації.

Звичайна мова специфікації Spec#

Сучасна мова специфікація Spec# є розширенням об’єктно-орієнтованої мови C# засобами, що забезпечують верифікацію програм для платформи .Net [37]. Ці засоби подаються до програми в C# невеликими додатковими описами, а саме:

– ненульових посилань до параметрів викликів CALL;

– контрактів між викликами і реалізаціями;

– обробки виникаючих виключних ситуацій програми для інформування розробника;

– змінювання полів даних об’єктів тощо.

Ненульові типи даних помічаються типом T! і відповідають деяким змінним програми, які можуть використовуватися при специфікації полів даних, формальних параметрів і з обернених цьому типу значень, локальних змінних програми. Цей тип не належить до елементів масиву. Головне призначення ненульових типів – забезпечити посилання до інших елементів, опис патернів, перевірку умов виходу з виразів і циклів, обумовлених контрактом.

Контракт ставиться між тим, хто робить виклик, і хто – реалізацію. У передумові додається опис стану параметрів при виклику, в постумові визначається умова отримання результату об’єкта, що викликався, і передача цього результату протилежна. Spec# надає підтримку більш дисциплінованому використанню виключення, щоб забезпечити ясність і підтримку життєздатності програми [38]. У програмі можуть бути відмови і помилки. У даному випадку у методі використовується аналіз забороненої

умови, коли передумова була не задоволена. Більшість відмов у програмі – це, коли порушена умова контракту. Наприклад, отримання виходу з циклу при перевищенні значення параметра циклу, що не було визначено у передумові.

Обробка виключних ситуацій виконується при роботі з масивами, коли елемент не відповідає типу. Якщо в передумові специфікується індекс, що знаходиться всередині меж масиву, а при виконанні цього не відбувається, то компілятор відповідає клієнту про невиконання передумови в реальному часі.

Змінювання полів даних задається фреймовими умовами, що вміщується у контракт, і починаються modifies, за яким слідкує оператор частини програми методу реалізації, що дозволяє зміну.

Підхід до реалізації специфікації. Опис специфікації в Spec# є самостійною програмою, що містить у собі перед- і постумови, а також різні дії над фреймовими структурами щодо програми, яка перевіряється мовою C#. Ця специфікація подається в мово-незалежному форматі і перебудовується в мову *C#* за допомогою спеціального транслятора, що працює на платформі *.Net.* Цей транслятор має аналізатор і версифікатор для перевірки правильності опису специфікації. Транслятор виконує переклад у вигляді частини програми, для якої створювалася специфікація. Верифікація специфікації є статичною, вона орієнтована на перевірку правильності опису, а саме, меж масивів, явних значень змінних тощо. Прувер транслятора виконує перевірку деяких умов і операторів, а також значень змінних. Специфікації в *Spec#* накопичуються у репозитарії, і при застосуванні звертаються до *Base Class Library*, де накопичуються об’єкти, їхні інваріанти та контракти.

Контракти і аналогічні механізми верифікації програм з мов програмування реалізовані також в системах *Java, Eiffel і Spark*. У мові *Java* контракти вміщуються в опис програми як стилізовані коментарі. Середовище *Java* має такі засоби: перевірку контрактів у динаміці, виконання, виклики та об’єктні інваріанти. В об’єктно-орієнтованій системі *Eiffel* є бібліотека констрейнів, котрі вставляються у опис об’єкта і виконують верифікацію в динаміці виконання. Однак механізми модифікації не дозволяють для

callbacks об’єктних інваріантів і тому вони не вміщуються в модульні об’єкти. Система *Spark* підтримує *підмножину* мови *Ада* при вставці теорем для прувера, помічених як коментарі, але компілятор цієї мови їх не використовує. Засоби верифікації в *Spark* орієнтовані на окремий опис умов виконання Ада-програм для верифікації, аналогічно до методології *Spec#.* Вони окремо транслюються у вихідний код Ада-програми і виконуються разом з нею в режимі верифікації [37, 38].

1.4. Методи паралельної обробки інформації в часопараметризованих мультипаралельних програмах

Однією з головних відмінностей часопараметризованих мультипаралельних програм є урахування, при їх проєктуванні, раціональної сукупності методів паралельної обробки інформації залежно від вимог і обмежень замовника. На даний час в літературі знайшли опис п’ять методів паралельної обробки інформації [1–3, 7, 43, 52]:

1) метод суміщення незалежних операторів;
2) метод конвеєрної обробки;
3) метод декомпозиційної обробки;
4) кодово-матричний метод;
5) метод суміші алгоритмів.

Методи паралельної обробки інформації та особливості їхнього застосування показано на Рис. 1.3.

Сутність *методу суміщення* незалежних операторів полягає в одночасному початку виконання у кожний з дискретних моментів часу деякої кількості операторів алгоритму, для яких виконуються такі умови:

а) ці оператори не пов’язані інформаційними та/або керуючими зв’язками;

б) для кожного з таких операторів до моменту часу, що розглядається, є значення відповідних операндів.

Метод поєднання незалежних операцій застосовується у всіх відомих багатопроцесорних ОС як одночасне виконання множиною процесорів

сукупності «паралельних процесів», кожен із яких є фрагмент («нитка») операцій, які послідовно виконуються у межах загальної великої задачі.

Метод конвеєрної обробки – це метод паралельної обробки даних, в якому в якості об'єктів виступають:

- алгоритми виконання операторів – загальноприйнятих операцій/функцій відомих мов програмування;
- об'єкти алгоритмічного рівня – фрагменти алгоритмів або алгоритми, що розглядаються як неподільне ціле.

Рис. 1.3 Методи паралельної обробки інформації
та особливостей їх застосування.

Сутність методу полягає у виконанні для алгоритму наступних перетворень:

а) поділу часового алгоритму виконання операцій/функцій (на операційному рівні) або часового алгоритму розв'язання задачі (на алгоритмічному рівні) на "конвеєрні" фрагменти (*F*) рівної часової глибини *TK* (такт конвеєра), такі, що кожен попередній фрагмент формує вхідні дані для суміжного наступного фрагмента, а параметр $t^{n}(Fj)$ початку кожного

наступного фрагмента F_j визначається параметром $t^к(Fi)$ кінця попереднього фрагмента $F_i(i,j \in NF;NF= 0,1,...,nf\ 1;$ *де* $nf=|\ NF|$ – кількість конвеєрних фрагментів або глибина конвеєра);

б) послідовної реалізації у часі фрагментів F_0, F_1, ..., F_{nf-1} – у разі одиничної потужності ($sd = 1$) множини *SD* різних вхідних наборів даних алгоритму;

в) суміщення (при sd>1) інтервалів виконання "різнотипних" конвеєрних фрагментів алгоритму, що належать до різних наборів вхідних даних;

г) номери ρ поєднаних фрагментів $F\rho$, що відповідають вхідним наборам даних з номерами δ ($\delta=1,2,...,nf-1, nf, nf+1,...$), задовольняють (у режимі роботи конвеєра) наступному співвідношенню $\rho + \delta mod\ (nf +1) = nf$.

Метод конвеєрної обробки ґрунтується на розбитті кожного алгоритму на $n = TA/\Delta T$ фрагментів (ΔT – інтервал введення даних, що задається), постановці кожному фрагменту у взаємно однозначну відповідність пристрою («ступеня конвеєра»), послідовному з'єднанні зазначених частин відповідно до схеми з'єднання між собою відповідних фрагментів алгоритму, введення даних у конвеєрну структуру, що отримується, і виведення результатів рішення з тактовим інтервалом $\Delta T = TA/n$.

Сутність *методу паралельної суміші алгоритмів* полягає у створенні на операційному рівні суміші завдань, використанні такої суміші для досягнення 100% завантаження обладнання та забезпечення за рахунок цього потенційно можливого підвищення ефективності реалізації аналізованої сукупності алгоритмів. Суміш алгоритмів формально розглядається і виконується як єдине завдання.

Суміш завдань використовується в тих випадках, коли окремо виконувані завдання, що використовують усі або частину розглянутих вище методів паралельної обробки, не можуть забезпечити 100% завантаження обладнання та досягнення потенційного значення продуктивності.

Проведені дослідження показали доцільність використання сукупності методів паралельної обробки інформації при розробці часопараметризованих

мультипаралельних програм ІУС залежно від вимог і обмежень, що представлено в таблиці 1.1 [53].

Таблиця 1.1

Доцільний склад різних методів паралельної обробки для конкретних комбінацій вимог і обмежень

Вимоги/ обмеження		Методи паралельної обробки			
		Суміщення незалежних операцій	Метод конвеєрної обробки	Метод декомпозиційної обробки	Кодово-матричний метод
Час виконання алгоритму t_0 (алгоритмів t_v)	Z=1	+	-	-	+
	Z>1	+	+	+	+
Період оновлення вхідних даних T_{BX}	Z=1	-	+	+	+
	Z>1	-	+	+	+
Спільні вимоги $(t_0 / t_v) \& T_{BX}$	Z=1	+	+	+	+
	Z>1	+	+	+	+
Вимоги і обмеження $(t_0 / t_v) \& Q$	Z=1	+	-	-	+
	Z>1	+	+	+	+
Вимоги і обмеження $T_{BX} \& Q$	Z=1	-	+	+	+
	Z>1	-	+	+	+
Вимоги і обмеження $(t_0/t_v)\&T_{BX}\&Q$	Z=1	+	+	+	+
	Z>1	+	+	+	+

1.5. Постановка задачі розробки технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем.

Аналіз показує, що у відомих системах проєктування паралельних програм (СППП) широко використовуються методи статичної та динамічної верифікації [39–43]. У СППП основою верифікації є прогін та налагодження паралельних програм на конкретних значеннях числових даних. Відомі також спеціалізовані системи (наприклад, *MATHCAD, MAPLE*) верифікації з

урахуванням одиниць виміру фізичних величин [44, 45]. Засоби цих систем залежать від мови програмування та засновані на введенні до вихідного коду додаткових специфікацій фізичних величин та пов’язані з символьною обробкою, що ускладнює їхнє практичне використання, знижує ефективність результатів та обмежує сфери застосування [46, 47].

На сьогодні визнається, що кардинальним вирішенням цієї проблеми має бути створення інтелектуальних технологій автоматичного проєктування мультипаралельних програмних засобів [1, 6, 7, 12–16, 47].

Необхідність високої ефективності паралельного програмного забезпечення інформаційних управляючих систем вимагає суттєвого розширення складу факторів, що враховуються при формальній розробці часопараметризованих мультипаралельних програм [17, 39, 48]:

– особливостей алгоритмів задач, що розв’язуються, які описують у вигляді традиційних послідовних програм мовою високого рівня (Сі, С++, Фортран тощо);

– складу методів паралельної обробки даних, який може бути використаний при синтезі часових паралельних програм (суміщення незалежних операцій, конвеєрний метод, кодово-матричний метод, декомпозиційний метод, метод мультипаралельної суміші алгоритмів);

– особливостей архітектури та конфігурацій паралельних процесорів та багатопроцесорних обчислювальних систем, наприклад, клас паралельної обчислювальної системи (*SMP, MPP, NUMA*), тип процесорів, топології міжпроцесорних комунікацій (повнозв’язна система, загальна шина, кільце, гіперкуб тощо);

– ієрархія пам’яті, можливість та час паралельного доступу до пам’яті різних рівнів; організація міжпроцесорного обміну даними, спосіб синхронізації роботи процесорів та необхідні часові витрати тощо.);

– часових характеристик паралельних процесорів та багатопроцесорних обчислювальних систем: тривалість операцій, тактова частота, надійність, тощо;

– склад конкретної конфігурації паралельної обчислювальної системи (кількість функціональних блоків/процесорів та їх зв'язок тощо);

– системи вимог та обмежень (час вирішення, тактова частота, складність/вартість, надійність/достовірність), які висуваються користувачами до характеристик процесу паралельного вирішення задач;

– одиниць виміру фізичних величин, що надаються даними вихідних послідовних програм.

Формальний та автоматичний характер технологій проєктування часпараметризованих мультипаралельних програм робить винятково актуальною задачу розробки засобів автоматичної верифікації та візуалізації результатів усіх етапів проєктування з метою оперативної оцінки людиною коректності та достовірності синтезованих програмно-апаратних об’єктів [47].

Домовимося вважати, що поняття конструкція послідовних та паралельних програм визначається як сукупність специфікації даних, специфікації операцій\функцій, специфікації інформаційно-керівних зв’язків, специфікації моментів початку реалізації операцій\функцій та специфікації одиниць виміру фізичних величин. Для специфікації таких програм, як правило, використовується текстова (алгоритмічною мовою) та графічна (наприклад, мовою UML, Unified Modeling Language) форми подання. Необхідно відзначити, що UML-діаграми часто використовують при створенні багатопотокових, паралельних та розподілених програм [48, 49]. Це дозволяє скоротити час створення закінченої моделі програми (тобто деякого її спрощеного уявлення).

Основні побічні ефекти такого підходу:

– засобами UML неможливо поєднати розроблені ракурси докупи, тобто, створити специфікацію, придатну для отримання з неї машинних кодів;

– зміст моделі, як правило, неформалізовано [50].

На відміну від прийнятого в даний час трактування паралельних програм, часпараметризована мультипаралельна програма визначається як конструкція, яка містить у явному вигляді специфікації наступних категорій

інформації (Рис. 1. 4):

– множина об’єктів – даних, над якими повинні виконуватися дії (що задаються складом операцій/функцій алгоритмічної мови високого рівня);

– множина дій (операцій/функцій), які мають бути виконані над даними для вирішення задачі;

– множина статичних зв'язків, що задають відносини упорядкованості операцій/функцій за даними та управлінню;

– упорядкованість операцій/функцій у динаміці паралельного обчислювального процесу, що задається множиною моментів часу початку виконання операцій/функцій;

– поділ множини операцій/функцій на часові фрагменти (множинні часові оператори, МЧО), що включають сукупність операцій/функцій, виконання яких починається одночасно в конкретний момент дискретного часу;

– розподіл множини даних на фрагменти даних, що поставлені в однозначну відповідність множинним часовим операторам і використані у відповідні моменти дискретного часу;

– наявність інформації про розбиття множини команд різних фрагментів на підмножини (нитки), що виконуються відповідними модулями/процесорами;

– наявність інформації про одиниці виміру фізичних величин даних.

Об’єктами синтезу і верифікації мультипаралельних часопараметризованих програм є наступні:

– структури семантико-числової і графічної (в вигляді графів) специфікацій вхідних програм;

– структури семантико-числової і графічної (в вигляді часових паралельних граф-схем) специфікацій часопараметризованих паралельних моделей, які використовують різні методи паралельної обробки даних;

– структури семантико-числової, графічної та текстової специфікацій часопараметризованих паралельних програм.

Рис. 1. 4. Склад категорій інформації специфікованих часопараметризованих мультипаралельних програм.

Загалом верифікація мультипаралельних часопараметризованих програм має три складові:

– компіляційна верифікація;
– декомпіляційна верифікація;
– семантична верифікація.

Компіляційна верифікація забезпечує перевірку синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації перелічених вище об'єктів верифікації при проєктуванні цих об'єктів.

Декомпіляційна верифікація забезпечує перевірку логічної еквівалентності синтезованих мультипаралельних програм та текстів вхідних

послідовних програм після завершення всіх етапів синтезу (шляхом декомпіляції результату синтезу та порівняння з вхідним текстом).

Семантична верифікація полягає у перевірці (у динаміці або після завершення всіх етапів синтезу) збігу одиниць вимірювання фізичних величин, отриманих на основі формального синтезу мультипаралельних програм, та одиниць вимірювання вхідних та вихідних даних, що задаються користувачами.

Викладене вище обумовлює актуальність рішення *науково-прикладної задачі* розробки інформаційної технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення ефективності верифікації за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації на основі семантико-числових специфікацій.

Задля вирішення поставленої науково-прикладної задачі необхідно розробити цілий ряд методів, а саме:

– метод компіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем;

– метод декомпіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем;

– метод семантичної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем.

Вирішенню цих завдань присвячені наступні розділи дисертаційного дослідження.

Висновки до розділу 1

В першому розділі розглядаються особливості процесу розробки та верифікації паралельних програм інформаційних управляючих систем. Наводиться стислий огляд публікацій за темою дисертації, опис методів верифікації паралельних програм інформаційних управляючих систем: експертизи ПЗ, статичного аналізу, динамічних та формальних методів верифікації; наводяться переваги та недоліки цих методів. Описані

найпоширеніші інструментальні засоби верифікації програм з підтримкою паралельних обчислень. Зазначається, що на сьогодні необхідність високої ефективності паралельного програмного забезпечення інформаційних управляючих систем вимагає суттєвого розширення складу факторів, що враховуються при формальній розробці часопараметризованих мультипаралельних програм ІУС.

Формулюється задача дисертаційного дослідження, як розробка технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення ефективності верифікації за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації на основі семантико-числових специфікацій. Для вирішення поставленої науково-прикладної задачі вирішено розробити ряд методів, а саме: метод компіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем; метод декомпіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем; метод семантичної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем. Сформовано сучасні вимоги до суттєвого розширення складу факторів, що враховуються при формальній розробці часопараметризованих мультипаралельних програм ІУС.

Основні положення цього розділу викладені у публікаціях автора [1–8, 10, 24, 25, 27, 28].

РОЗДІЛ 2
МОДИФІКАЦІЯ МЕТОДУ КОМПІЛЯЦІЙНОЇ ВЕРИФІКАЦІЇ ПАРАЛЕЛЬНИХ ПРОГРАМНИХ ЗАСОБІВ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ.

Урахування сукупності методів паралельної обробки інформації, використаних при розробці часопараметризованих мультипаралельних програм, дозволило удосконалити метод компіляційної верифікації часопараметризованих паралельних програм для ІУС представлені в табл. 1.1.

2.1. Загальна характеристика методу компіляційної верифікації паралельних програм інформаційних управляючих систем

Однією зі складових верифікації паралельних програмних засобів ІУС є компіляційна верифікація [54]. Призначенням компіляційної верифікації є перевірка коректності статичних і динамічних результатів виконання основних етапів проєктування часопараметризованих мультипаралельних програмних засобів інформаційно управляючих систем. Основними етапами проєктування часопараметризованих паралельних програм є:

- синтез структур семантико-числової специфікації послідовної програми задачі;
- синтез графічної специфікації послідовної програми у вигляді графа;
- синтез часової паралельної моделі для заданої конфігурації програмних засобів і вимог;
- синтез текстових часових програм процесорів обчислювальної системи.

В основу перспективних методів формального синтезу паралельних програмних засобів покладені операції перетворення послідовних програм завдань в структури СЧС, виконання всіх операцій синтезу проєктованих об’єктів в термінах структур СЧС і зворотного переходу від СЧС-специфікацій синтезованих програмних об’єктів до традиційних за текстовим уявленням

програм [53]. Таким чином, виникає необхідність в верифікації наступних об’єктів:

− структури семантико-числової специфікації (СЧС) послідовних програм завдань;

− графічні специфікації послідовних програм у вигляді відповідних орієнтованих графів;

− структури семантико-числовий специфікації часопараметризованих мультипаралельних моделей завдань.

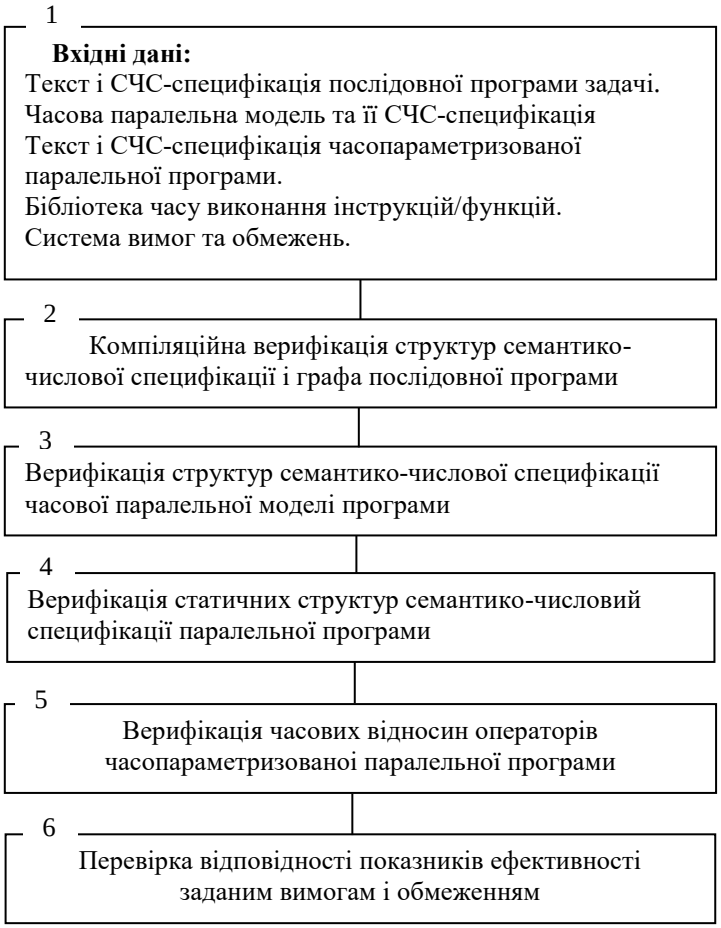


Рис. 2. 2. Етапи компіляційної верифікації формального синтезу часопараметризованих паралельних програм.

При компіляційній верифікації часопараметризованих паралельних програмних засобів забезпечуються наступні види перевірок:

а) рівність загальної кількості сполучених і зовнішніх зв’язків у часових паралельних моделях вхідних послідовних програм;

б) відповідність кількості входів, кількості виходів і значень їх розрядності для інструкцій або функцій у часових специфікаціях вхідних послідовних програм, значенням, заданим в текстах послідовних програм;

в) коректність типів входів і виходів, (типи: «дані», «управління», «сповіщальний») для інструкцій або функцій часових моделей послідовних програм;

г) коректність часової упорядкованості операторів – відсутність ситуацій початку виконання операторів - приймачів даних до завершення виконання операторів – джерел операндів. Основні етапи компіляційної верифікації процесів синтезу часопараметризованих мультипаралельних програм представлені на Рис. 2.2.

2.2. Метод компіляційної верифікації часопараметризованих паралельних програм інформаційних управляючих систем

В якості вхідних даних методу компіляційної верифікації паралельних часопараметризованих програм виступають:

– структури семантико-числової специфікації послідовної програми задачі;

– графічна специфікація послідовної програми задачі, що подається в вигляді орієнтованого графа;

– часові структури семантико-числової специфікації, що представляють часопараметризовані паралельні моделі програми задачі;

– графічні специфікації часопараметризованих паралельних моделей програм задачі;

– структури семантико-числової специфікації текстів часопараметризованих послідовних, моно- і мультипаралельних програм а також часових моделей їх виконання [13, 54–55]

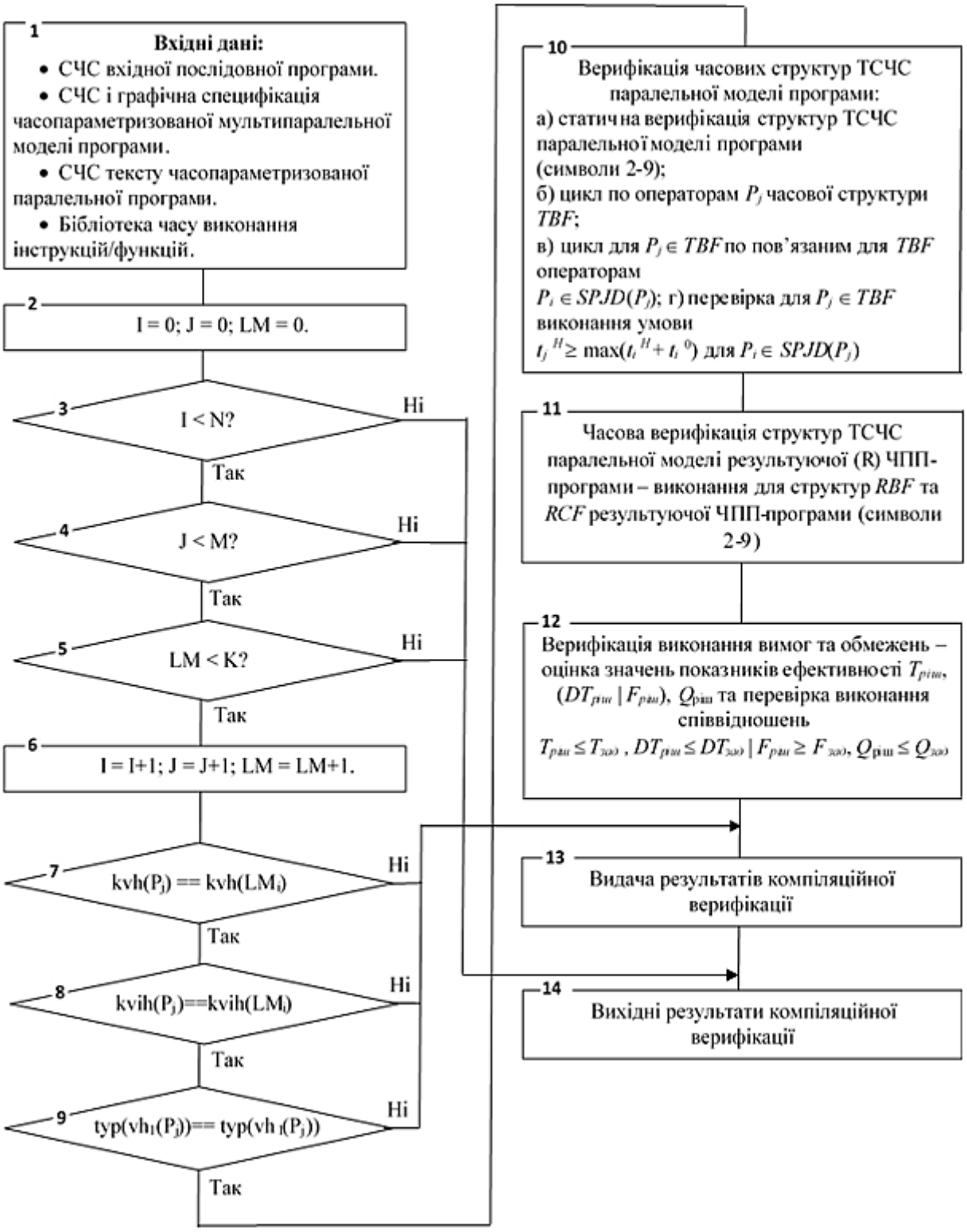


Рис. 2.3. Основні етапи компіляційної верифікації часопараметризованих мультипаралельних програм ІУС.

На Рис. 2.3 представлено основні етапи компіляційної верифікації часопараметризованих мультипаралельних програм ІУС.

Перший етап (символи 2–9, Рис. 2.3) виконує наступні дії, що проводять оцінку коректності статичних структур *BF* і *CF* семантико-числової специфікації (СЧС) і графа вихідної послідовної програми, а саме, забезпечує перевірку відповідності кількості входів, кількості виходів і значень їх типів для інструкцій у специфікаціях вхідних послідовних програм, значенням, заданим в текстах послідовних програм:

а) організуємо цикл по операторам P_j структури *BF*:

$$\forall i \in N(BF);\ i = P_j;\ i = \overline{1,n},$$

де *n* – кількість операторів задачі;

б) організуємо цикл зі зв'язків структури *CF* операторів P_j задачі: $\forall j \in NN(CF);\ \ j = \overline{1,m};$ де m кількість сполучених\ зовнішніх зв'язків операторів задачі;

в) організуємо цикл за елементами LM_i бібліотеки L вхідних і вихідних інтерфейсів:

$$\forall LM_i \in L;\ LM = \overline{1,k};$$

де *k* кількість елементів бібліотеки вхідних і вихідних інтерфейсів;

d) встановлення взаємно однозначної відповідності між операторами P_j ϵBF і операторами мови послідовного програмування, що мають однаковий тип:

$$\left(\forall P_j \in BF\right)\&(\forall LM_i \in L) : typ\left(vh_1\left(P_j\right)\right) = typ\left(vh_1(LM_i)\right), \quad (2.1)$$

$$kvh(P_j) = kvh(LM_i),$$

$$kvih(P_j) = kvih(LM_i).$$

Перевірка рівності загальної кількості сполучених і зовнішніх зв'язків у часових паралельних моделях вхідних послідовних програм забезпечується за допомогою «Тесту відповідності кількості сполучених і зовнішніх зв'язків». Коректне проходження тесту базується на необхідності виконання умови (2.2)

$$\sum_j n(sjd) = \sum_j n(wjd), \quad (2.2)$$

де *sjd* – множина потужностей сполучених операторів для всіх операторів задачі;

wjd – множина потужностей зовнішніх операторів для всіх операторів задачі;

n(sjd) – кількість сполучених операторів для *j*-го оператора задачі;

n(wjd) – кількість зовнішніх операторів для *j*-го оператора задачі.

При проходженні «Тесту числа зв'язків по сполученим елементам» виконуються наступні дії:

$$\forall P_j \in N(BF) : sjd = |SJD|,$$

а) вибираємо оператор P_j для якого виконується умова:

$$sjd(P_j) \neq 0.$$

Обираємо вказівник на початок ланцюжка сполучених операторів:

$$NN = NSJ(P_j);$$

б) SS_j-кількість сполучених операторів у масиві *SPJD*:

$$SS_j = 0;$$

в) по номеру *NN* рядка звертаємось до масиву *CF* структури *SPJD* :

$$P_l \in SPJD; l = \overline{1, sjd};$$

та фіксуємо 1-й оператор із множини сполучених для P_j;

г) збільшуємо значення лічильника поточної кількості сполучених операторів для P_j:

$$SS_j = SS_j + 1;$$

д) по вказівнику на продовження ланцюжка сполучених операторів (масив *JSD*) обираємо вказівник на рядок масиву *SPJD*, який містить черговий сполучений оператор для P_j:

$$njs_j = JSD(NN_j),$$

де njs_j– вказівник на продовження ланцюжка сполучених операторів для P_j,

$$\text{while } njs_j \neq -1;\ P_l = SPJD(njs_j);\ SS_j = SS_j; \qquad (2.3)$$

е) при виконанні умови $njs_j = -1$ ланцюжок номерів сполучених операторів закінчується і даний рядок є останнім у підмасиві $SPJD_j$ номерів сполучених операторів для P_j:

$$SS_j = SS_j + 1.$$

При коректному проходженні тесту отримуємо значення

$$SS_j = |SJD|. \quad (2.4)$$

Проходження «Тесту числа зв'язків за зовнішніми елементами» потребує виконання наступних дій:

$$\forall P_j \in N(BF) : wjd = |WJD| \quad (2.5)$$

а) організуємо цикл по операторам P_j структури *BF*:

обираємо оператор P_j для якого виконується умова

$$wjd(P_j) \neq \emptyset, \quad (2.6)$$

з масиву *NWJ* структури *BF* обираємо покажчик на початок ланцюгу зовнішніх операторів

$$NW = NWJ\ (Pj); \quad (2.7)$$

б) вводимо поточну змінну для оцінки значення кількості зовнішніх операторів в масиві *WPKD*:

$$WW_j = 0,$$

де WW_j – поточне значення кількості зовнішніх операторів в масиві *WPKD*;

в) за номером рядка *NW* зовнішніх операторів звертаємося до масиву *WPKD* структури *CF*

$$\forall P_k \in WPKD, k = \overline{1, wjd};$$

фіксуємо 1-ий оператор з множини зовнішніх для P_j;

г)збільшуємо значення лічільника поточної кількості зовнішніх операторів для P_j;

$$WW_j = WW_j + 1;$$

д) за покажчиком на подовження ланцюга зовнішніх операторів (масив *JWD*) обираємо показчик на рядок масиву *WPKD*, що містить черговий зовнішній елемент для P_j:

$$njw_j = JWD(WW_j), \tag{2.8}$$

де njw_j – покажчик на продовження ланцюга зовнішніх операторів для P_j;

$$while\ njw_j \neq -1 : Pk = WPKD(njw_j), \tag{2.9}$$

$$WW_j = WW_j + 1;$$

е) при $njwj=-1$ ланцюг номерів зовнішніх операторів для оператора P_j закінчується , цей рядок є останнім в підмножені $WPKD_j$ номерів зовнішніх операторів для P_j

$$WW_j = WW_j + 1.$$

За умови коректного проходження тесту WW_j приймає значення:

$$WW_j = |n|. \tag{2.10}$$

Тест числа зв’язків по сполученим елементам проводиться для всіх операторів задачі:

$$SS = \sum SS_j,\ \forall j \in N(BF). \tag{2.11}$$

Тест кількості зв’язків за зовнішніми елементами також проводиться для всіх операторів задачі:

$$WW = \sum_j WW_j,\ \forall j \in N(BF). \tag{2.12}$$

По закінченню проведення тестів відповідності кількості сполучених та зовнішніх елементів проводиться порівняння значень SS та WW.

За умови позитивного проходження тесту відповідності кількості сполучених та зовнішніх зв’язків структур семантико-числових специфікацій вхідної задачі значення SS та WW повинні співпадати:

$$SS = WW. \tag{2.13}$$

«Тест відповідності виводів по сполученим елементам» проводиться відповідно до наступних дій:

а) обираємо в масиві *N* структури *BF* оператор P_j є P , де P – множина номерів операторів задачі;

б) в відповідності до пунктів а, в, д «Тесту кількості зв'язків по сполученим елементам» обираємо сполучений оператор в рядку з номером njs_j.

В цьому ж рядку структури *CF* в масиві *SNWIH* знаходиться номер виходу оператора, сполученого для P_j, а в масиві *SNWHO* - номер входу оператора, пов'язаного з даними сполученим оператором:

$$nwih_j=SNWIH\ (njs_j); \qquad (2.14)$$
$$nwho_j=SNWHO\ (njs_j);$$

в) введемо поточне значення кількості вхідних операторів для P_j

$$KS=0\ ;$$
$$\forall P_j\ \in P\ , j=0, N-1: \text{якщо}\ nwho_j \neq -1,$$
$$KS=KS+1;$$

для кожного оператора задачі перевіряється номер входу та підсумовується поточне значення кількості входів по сполученим елеменам.

За умови позитивного проходження тесту $KS=|SJD(P_j)|$.

В іншому випадку видається інформація о похибці «Для елемента (P_j) тест пропущено» (невідповідність числа зв'язків)

Аналогічним чином організовано проходження Тесту відповідності виводів за зовнішніми елементами.

Другий етап (символ 10 Рис. 2.3). Виконує дії, що забезпечують оцінку коректності часових структур *TFM* СЧС (*BFM, CFM, TFM*) часопараметризованої паралельної моделі вихідної програми (з графічною специфікацією у вигляді часової паралельної граф-схеми (*ВПГС*):

а) перевірка коректності статичних структур *BFM, CFM* семантико-числової специфікації часової паралельної моделі вихідної послідовної програми шляхом виконання операцій 2–9 (Рис. 2.3);

б) реалізація циклу по операторам *Pj* часової структури *TF* параметрів початку виконання операторів;

в) реалізація циклу для кожного оператора $P_j \in TBF$ по зв'язаних для P_j операторам $Pi \in SPJD\ (Pj)$.

Врахування раціональної сукупності методів паралельної обробки інформації, застосованих при проєктуванні мультипаралельних програмних засобів ІУС, дозволяє значно скоротити час на компіляційну верифікацію структур *BFM, CFM* семантико-числової специфікації часової паралельної моделі.

Це відбувається за рахунок того, що оператори, які додані в структури СЧС при застосуванні тих, чи інших методів паралельної обробки розташовуються в структурах СЧС після вихідних операторів задачі і їх кількість не перевищує 30% від загальної кількості операторів задачі. Таким чином, не виникає необхідності оперувати зі всіма операторами задачі.

Третій етап (символ 11 Рис. 2.3). Забезпечує статико - часову верифікацію моделі результуючої (*R*) часопараметризованої паралельної програми шляхом виконання операцій символів 1–9 Рис. 2.3 для структур семантико-числової специфікації *RBF, RCF і RTF.*

Четвертий етап (символ 12 Рис. 2.3). Забезпечує оцінку значень показників ефективності синтезованої часопараметризованої паралельної програми – часу рішення задачі $T_{ріш}$, інтервалу введення даних $DT_{ріш}$ і перевірку задоволення вимог і обмежень

$$T_{ріш} \le T_з\ ,\ DT_{ріш} \le DT_з$$

Результати компіляційної верифікації відображаються в багатосторінковій панелі виводу і включають:

- узагальнені результати верифікації (сторінка *View of test results*);
- результати верифікації розподілу часових операторів по ярусах
- часової паралельної моделі (сторінка *View of tiers*);
- детальні результати верифікації статичних і часових параметрів операторів завдання (сторінка *View of connection of elements*).

2.3. Приклад компіляційної верифікації синтезу часопараметризованої паралельної програми задачі

Для ілюстрації застосування компіляційної верифікації паралельних програмних засобів ІУС використаємо послідовну програму задачі (послідовна програма якої представлена на Рис. 2.4):

```
#include <stdio.h>
void main(void)
{
int a,b,c;
int x,y,k,s;
 scanf("%d %d %d",&a,&b,&c);
 x = a * b;
 y = b * (a / b);
 if(x - y < 0) k = (x * y) * (a / c);
 else k = x + c * (b / a);
 s = k % 2;
 printf("%d",s);
}
```

2.4 Вхідна послідовна програма.

Вхідними даними для компіляційної верифікації часопараметризованої мультипаралельної програми даної задачі є:

- структури семантико-числової специфікації *BF, CF* послідовної програми задачі (представлені табл. 2.1 і табл. 2.2);
- графічна специфікація послідовної програми задачі у вигляді графа (Рис. 2.5);
- тривалості виконання інструкцій / функцій послідовної програми (табл. 2.3);
- графічна специфікації (ГСЧП) часопараметризованої паралельної моделі послідовної програми (Рис. 2.6);
- для даної задачі *BFM = BF, CFM = CF*, а моменти tj^H початку виконання операторів P_j задаються часової моделлю (Рис. 2.7).

Таблиця 2.1

Базова структура *BF* операторів вхідної послідовної програми

N	***MET***	***TYP***	***NSJ***	***SJD***	***BJ***	***NWJ***	***WJD***	***MP1***	***MP2***	***VH***	***VIH***	***RES***	***N***	***MET***	***TYP***	***NSJ***	***SJD***	***BJ***	***NWJ***	***WJD***	***MP1***	***MP2***	***VH***	***VIH***	***RES***
0	0	58	–1	0	0	0	1	0	0	0	1	a_in	35	0	12	42	2	2	63	2	0	0	2	1	=
1	0	58	–1	0	0	1	1	0	0	0	1	b_in	36	0	2	44	3	2	65	1	0	0	3	1	–
2	0	58	–1	0	0	2	1	0	0	0	1	c_in	37	0	12	47	2	2	66	2	0	0	2	2	=
3	0	47	–1	0	0	3	1	0	0	0	2	a	38	0	1	49	2	2	68	1	0	0	2	1	+
4	0	47	–1	0	0	4	1	0	0	0	2	b	39	0	12	51	2	2	69	2	0	0	2	2	=
5	0	47	–1	0	0	5	1	0	0	0	2	c	40	0	50	53	2	2	71	1	3	0	2	1	bp
6	0	47	–1	0	0	6	2	0	0	0	2	r	41	3	54	55	2	3	72	1	0	0	2	1	l.o
7	0	47	–1	0	0	8	1	0	0	0	2	k	42	0	53	57	2	3	73	1	0	0	2	1	a.o
8	0	47	–1	0	0	9	1	0	0	0	2	l	43	0	53	59	2	3	74	1	0	0	2	1	a.o
9	0	47	–1	0	0	10	2	0	0	0	2	m	44	0	53	61	2	3	75	1	0	0	2	1	a.o
10	0	47	–1	0	0	12	2	0	0	0	2	p	45	0	2	63	3	3	76	1	0	0	3	1	–
11	0	47	–1	0	0	14	1	0	0	0	2	s	46	0	12	66	2	3	77	3	0	0	2	1	=
12	0	47	–1	0	0	15	1	0	0	0	2	t	47	0	3	68	2	3	80	1	0	0	2	1	*
13	0	47	–1	0	0	16	2	0	0	0	2	x	48	0	4	70	2	3	81	1	0	0	2	1	/
14	0	47	–1	0	0	18	2	0	0	0	2	y	49	0	12	72	2	3	82	4	0	0	2	1	=
15	0	12	0	2	0	20	4	0	0	2	1	=	50	0	25	74	2	3	86	1	0	0	2	1	<
16	0	12	2	2	0	24	2	0	0	2	1	=	51	0	51	76	1	3	87	4	4	5	1	2	upl
17	0	12	4	2	0	26	2	0	0	2	1	=	52	4	3	77	3	4	91	1	0	0	3	1	*
18	0	3	6	2	0	28	1	0	0	2	1	*	53	0	12	80	2	4	92	2	0	0	2	2	=
19	0	12	8	2	0	29	4	0	0	2	1	=	54	0	4	82	3	4	94	1	0	0	3	1	/
20	0	1	10	2	0	33	1	0	0	2	1	+	55	0	12	85	2	4	95	2	0	0	2	2	=
21	0	12	12	2	0	34	5	0	0	2	1	=	56	0	50	87	2	4	97	1	6	0	2	1	bp
22	0	2	14	2	0	39	1	0	0	2	1	–	57	5	59	89	2	5	98	1	0	0	2	1	sin
23	0	25	16	2	0	40	1	0	0	2	1	<	58	0	12	91	2	5	99	2	0	0	2	2	=
24	0	51	18	1	0	41	5	1	2	1	2	upl	59	0	60	93	2	5	101	1	0	0	2	1	cos
25	0	57	–1	0	1	46	5	0	0	0	1	C2_	60	0	12	95	2	5	102	2	0	0	2	2	=
26	1	4	19	3	1	51	1	0	0	3	1	/	61	0	50	97	2	5	104	1	6	0	2	1	bp
27	0	3	22	2	1	52	1	0	0	2	1	*	62	6	54	99	2	6	105	1	0	0	2	1	l.o
28	0	12	24	2	1	53	2	0	0	2	2	=	63	0	53	101	2	6	106	1	0	0	2	1	a.o
29	0	3	26	3	1	55	1	0	0	3	1	*	64	0	53	103	2	6	107	1	0	0	2	1	a.o
30	0	12	29	2	1	56	2	0	0	2	2	=	65	0	50	105	1	6	108	1	980	0	1	1	bp
31	0	1	31	3	1	58	1	0	0	3	1	+	66	980	49	106	1	7	–1	0	0	0	1	0	stop
32	0	12	34	2	1	59	2	0	0	2	2	=	67	0	48	107	1	7	–1	0	0	0	1	0	x_out
33	0	50	36	3	1	61	1	3	0	3	1	bp	68	0	48	108	1	7	–1	0	0	0	1	0	y_out
34	2	3	39	3	2	62	1	0	0	3	1	*													

Базова структура *BF* (табл. 2.2) операторів вхідної послідовної програми містить у своєму складі такі масиви даних:

BF = (N MET TYP NSJ SJD BJ NWJ WJD VH VIH RES),

де *N* – масив номерів інструкцій програми(або вершин - під час роботи з графічною специфікацією програмиу вигляді графу); *MET* – масив міток інструкцій; *TYP* – масив типів інструкцій програми; *NSJ* – масив покажчиків початку ланцюжка номерів вхідних/«сполучених» (для конкретної інструкції) інструкцій программи; *SJD* – масив кількостей вхідних/«сполучених» (для аналізованої інструкції) інструкцій програми; *BJ* – масив номерів природних частин програми (нерозгалужувані фрагментів інструкцій програми); *NWJ* – масив покажчиків початку ланцюжка номерів вихідних/«зовнішніх інструкцій програми для аналізованої інструкції; *WJD* – масив кількостей вихідних/«зовнішніх» (для аналізованої інструкції) інструкцій програми; *VH і VIH* – кількість входів (операндів) та кількість виходів (вихідних даних) інструкцій/функцій програми; *RES* – масив імен даних та операцій/функцій.

Структура CF зв'язків (табл. 2.3) містить у своєму складі такі масиви даних :

JWD WPJD WNWHO CF = (NN JSD SPJD SNWIH SNWHO WNWIH),

де *N* – номер рядка у структурі *CF*; *SPJD* – масив пов'язаних множин різних інструкцій послідовної програми (різних вершин графу програми); *JSD* – покажчик на продовження ланцюжка номерів інструкцій програми (або вершин графа), що входять до пари *SPJD* конкретної інструкції програми (або вершини графа); *SNWIH* – номер виходу конкретної сполученої інструкції (вершини), пов'язаної з входом вершини, що розглядається; *SNWHO* – номер входу інструкції (вершини), що зв'язує цю вершину з виходом відповідної поєднаної інструкції (вершини); *WPJD* – зовнішня множина інструкцій (або вершин графа програми; *JWD*-покажчик на продовження ланцюжка номерів зовнішніх інструкцій (вершин графа), що утворюють зовнішню множину *WPJD* конкретної інструкції послідовної програми (або вершини графу);

WNWHO – номер входу інструкції (верш) , що є зовнішньою для інструкції (вершини), що розглядається, *WNWIH* – номер виходу аналізованої інструкції (вершини), що пов'язує її з відповідною зовнішньою інструкцією програми (зовнішньої вершиною графу).

Таблиця 2.2

Структура *CF* зв'язків операторів вхідної послідовної програми

N	***JSD***	***SPJD***	***SNWIH***	***SNWHO***	***TSS***	***JWD***	***WPJD***	***WNWHO***	***WNVIH***	***TVS***	***N***	***JSD***	***SPJD***	***SNWIH***	***SNWHO***	***TSS***	***JWD***	***WPJD***	***WNWHO***	***WNVIH***	***TVS***
0	1	0	0	0	0	–1	10	0	0	0	26	–1	21	0	2	0	–1	17	0	0	0
1	–1	3	1	1	1	–1	11	0	0	0	27	28	22	0	0	0	28	18	1	0	0
2	3	1	0	0	0	–1	12	0	0	0	28	–1	23	0	1	0	–1	22	1	0	0
3	–1	4	1	1	1	–1	10	1	1	1	29	30	8	1	1	1	–1	20	0	0	0
4	5	2	0	0	0	–1	11	1	1	1	30	–1	24	0	0	0	–1	20	1	0	0
5	–1	5	1	1	1	–1	12	1	1	1	31	32	10	0	0	0	–1	21	0	0	0
6	7	10	0	0	0	–1	14	1	1	1	32	33	11	0	1	0	33	22	2	0	0
7	–1	11	0	1	0	–1	17	1	1	1	33	–1	21	1	2	1	34	23	2	0	0
8	9	6	1	1	1	9	25	1	1	1	34	35	12	0	0	0	–1	26	2	1	1
9	–1	13	0	0	0	–1	29	1	1	1	35	–1	26	0	1	0	–1	24	0	0	0
10	11	10	0	0	0	–1	34	1	1	1	36	37	14	0	0	0	–1	24	1	0	0
11	–1	11	0	1	0	12	13	0	0	0	37	–1	27	0	1	0	–1	25	0	0	0
12	13	11	0	0	0	13	15	0	0	0	38	39	8	1	1	1	39	30	0	1	1
13	–1	15	0	1	0	14	23	0	0	0	39	–1	28	0	0	0	–1	31	0	0	0
14	15	7	1	1	1	–1	26	0	0	0	40	41	29	1	1	1	–1	27	1	0	0
15	–1	16	0	0	0	16	13	1	0	0	41	–1	25	1	0	1	–1	28	1	0	0
16	17	14	0	0	0	17	15	1	0	0	42	43	29	0	1	0	–1	29	0	0	0
17	–1	17	0	1	0	18	16	0	0	0	43	–1	25	0	0	0	44	30	1	1	1
18	19	18	0	0	0	19	23	1	0	0	44	45	31	0	0	0	–1	31	1	0	0
19	–1	19	0	1	0	–1	26	1	0	0	45	46	32	0	1	0	–1	33	2	0	0
20	–1	^0	0	0	0	–1	27	0	0	0	46	–1	30	0	2	0	–1	33	0	0	0
21	22	14	0	0	0	–1	14	0	0	0	47	48	9	1	1	1	–1	33	1	0	0
22	23	17	0	1	0	23	18	0	0	0	48	–1	33	0	0	0	–1	34	0	0	0
23	–1	21	0	2	0	24	22	0	0	0	49	–1	34	1	0	1	50	35	0	1	1
24	25	10	0	0	0	–1	28	0	0	0	50	–1	34	0	0	0	–1	36	0	0	0
25	26	11	0	1	0	–1	16	1	0	0											

Таблиця 2.3

Тривалості t_j^0 виконання операцій різних типів «*typ*» (в тактах), використані при синтезі паралельної *SMP*-моделі послідовної програми

typ	***vx***	**+, –**	**=**	*****	**/**	**>, <**	***upl***	***var***	**=**
t_j^0	1.00	1.00	2.0	10.00	35.00	1.00	1.00	1.00	4.00

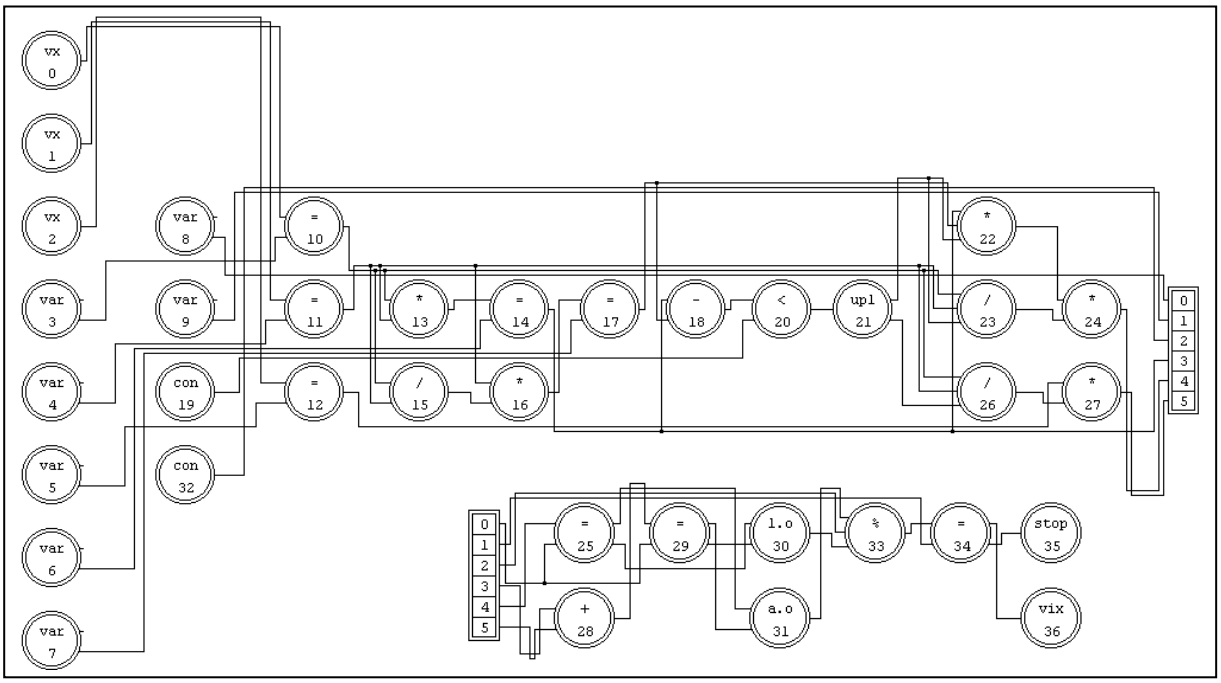


Рис. 2.5. Граф вхідної послідовної програми

При проходженні компіляційної верифікації можливі наступні варіанти отриманих результатів:

– варіант 1 - всі етапи формального синтезу виконані коректне;

– варіант 2 - має місце некоректне виконання формального синтезу статичних об'єктів;

– варіант 3 - має місце некоректне виконання формального синтезу часових об'єктів.

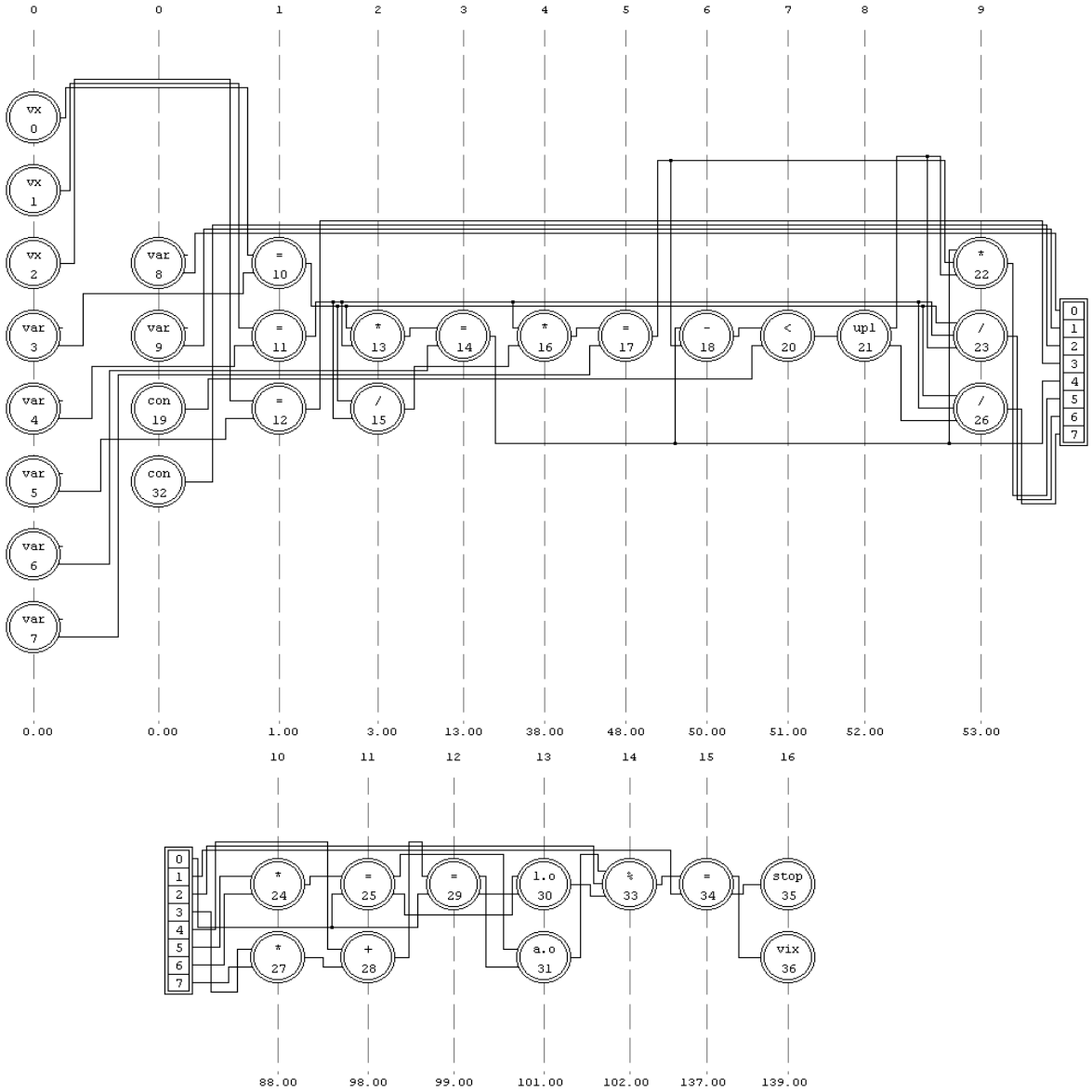


Рис. 2.6. Часова паралельна модель вхідної програми.

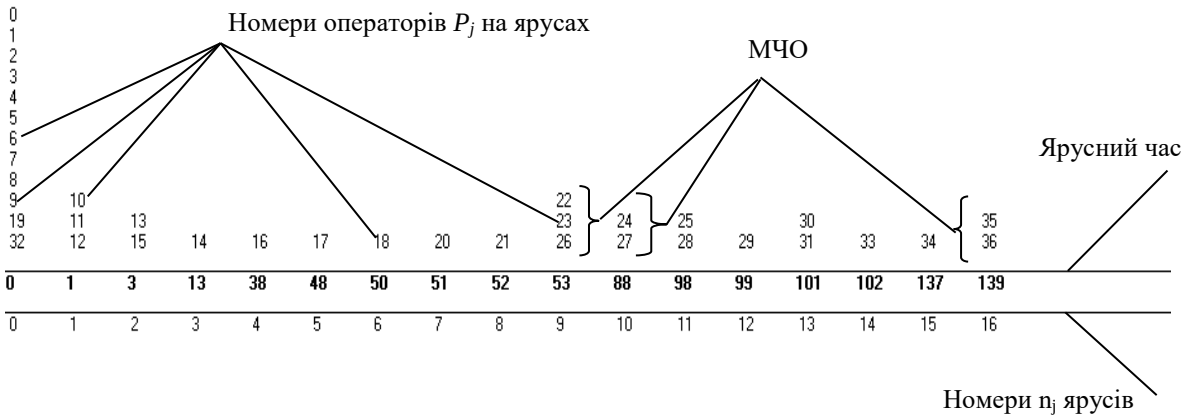
На Рис. 2.7 представлено результати компіляційної верифікації структур СЧС *BF* і *CF* графа задачі і часової моделі вхідної послідовної програми. Гістограму розподілу множинних часових операторів по ярусах паралельної моделі представляє Рис. 2.8.

Варіант 1 – етапи формального синтезу виконані коректно.

Результати перевірки статичної коректності графа програми (Рис. 2.5) і статичної (табл. 2.2, табл. 2.3) і часової коректності паралельної моделі (Рис. 2.6) ілюструє Рис. 2.7. Як видно, структури СЧС графа вхідної програми і часопараметризованої моделі паралельної програми коректні.

```
Файл елементів:                          C:\My_prog\C\Result\D_14_a\PRO\L_O_VIX1.TXT
Файл зв'язків елементів:                 C:\My_prog\C\Result\ D_14_a\PRO\L_O_VIX2.TXT
Файл розміщення по ярусам        C:\My_prog\C\Result\ D_14_a\PRO\NATA_LO.TXT

 ТЕСТ КОРЕКТНОСТІ ФАЙЛІВ:
 максимальна кількість елементів:  0-36
максимальна кількість зв'язків        0-50

 ТЕСТ ВІДПОВІДНОСТІ ЧИСЛА, СПОРУЖЕНИХ І ЗОВНІШНІХ ЗВ'ЯЗКІВ: ОК

 ТЕСТ ЧИСЛА ЗВ'ЯЗКІВ ПО СПОРУЖЕНИХ ЕЛЕМЕНТАХ: ОК

 ТЕСТ ЧИСЛА ЗВ'ЯЗКІВ ПО ЗОВНІШНІМ ЕЛЕМЕНТАМ: ОК

 ТЕСТ ВІДПОВІДНОСТІ ВИСНОВКІВ ПО СПОРУЖЕНИХ ЕЛЕМЕНТАХ: ОК

 ТЕСТ ВІДПОВІДНОСТІ ВИСНОВКІВ ПО ЗОВНІШНІМ ЕЛЕМЕНТАМ: ОК

 ТЕСТ ВІДПОВІДНОСТІ  ЯРУСНОГО ЧАСУ:
ТЕСТ ВІДПОВІДНОСТІ  ЯРУСНОГО ЧАСУ: ОК
ТЕСТ ВІДПОВІДНОСТІ ЧИСЛА ВХОДІВ ЕЛЕМЕНТУ І КІЛЬКОСТІ ЙОГО СПОЛУЖЕНИХ: ОК
```

Рис. 2.7. Результати компіляційної верифікації структур СЧС графа і часової паралельної моделі програми (*View of test results*) при коректному проходженні верифікації.



2.8. Гістограма верифікації розподілу множини часових операторів паралельної SMP-моделі по часовим ярусах (*View of tiers*).

Для візуалізації результатів компіляційної верифікації задач великої розмірності доцільно використовувати гістограмне представлення з більшею деталізацією результатів верифікації. На Рис. 2.8 відображено результати перевірки коректності розташування часових операторів (ЧО) і множинних

часових операторів (МЧО) на часових ярусах для часової моделі (Рис. 2.6) і структур СЧС (*BF CF* табл. 2.1 і табл. 2.2).

Варіант 2 – наявність помилок формування зв'язків операторів.

Наступним можливим варіантом результатів компіляційної верифікації є наявність помилок, які виникли при формальному синтезі графа і структур семантико-числовий специфікації *BF, CF* вихідної послідовної програми.

Припустимо, що склад виниклих помилок включає:

а) втрату повідомлюючого зв'язку від оператора типу «=» P_{34} до оператора P_{35} типу «*stop*»;

б) втрату зв'язку за даними від оператора типу «=» P_{34} до оператора P_{36} типу «*vix*».

Вид вихідних результатів компіляційної верифікації для прийнятих варіантів помилок формального синтезу представлений на Рис. 2.9.

```
Файл елементів:                          C:\My_prog\C\Result\D_14_a\VIXVIX1.TXT
Файл зв'язків елементів:                 C:\My_prog\C\Result\ D_14_a\VIXVIX2.TXT

 ТЕСТ КОРЕКТНОСТІ ФАЙЛІВ:
 максимальна кількість елементів:  0-36
максимальна кількість зв'язків     0-50

 ТЕСТ ВІДПОВІДНОСТІ ЧИСЛА, СПОРУЖЕНИХ І ЗОВНІШНІХ ЗВ'ЯЗКІВ:
          Різна кількість зв'язків!!!:
          Сполучених51   зовнішніх49


 ТЕСТ ЧИСЛА ЗВ'ЯЗКІВ ПО СПОРУЖЕНИХ ЕЛЕМЕНТАХ: ОК

 ТЕСТ ЧИСЛА ЗВ'ЯЗКІВ ПО ЗОВНІШНІМ ЕЛЕМЕНТАМ:
          Різна кількість зовнішніх для елемента [34]
          вказано 0   фактично зв'язків 2

 ТЕСТ ВІДПОВІДНОСТІ ВИСНОВКІВ ПО СПОРУЖЕНИХ ЕЛЕМЕНТАХ: ОК

 ТЕСТ ВІДПОВІДНОСТІ ВИСНОВКІВ ПО ЗОВНІШНІМ ЕЛЕМЕНТАМ: ОК
          Для елемента [34] тест пропущена (невідповідність числа зв'язків)
ОК

ТЕСТ ВІДПОВІДНОСТІ ЧИСЛА ВХОДІВ ЕЛЕМЕНТА ТА КІЛЬКОСТІ ЙОГО СПОЛУЧЕНИХ: ОК

                    Test(s) finished
```

Рис. 2.9. Результати компіляційної верифікації СЧС при наявності помилок зв'язків операторів.

До складу виявлених помилок входять:

– фіксація факту нерівності кількості пов'язаних і зовнішніх зв'язків операторів (кількість *sn* сполучених зв'язків повинно бути рівною кількості *wn* зовнішніх зв'язків);

– фіксація для оператора P_{34} факту відмінності зазначеного в структурі *BF* кількості *wjd* = 0 зовнішніх операторів від фактичного і коректного значення *wjd* = 2 (зовнішніми для оператора P_{34} є оператори P_{35} і P_{36}, Рис. 2.9).

Варіант 3 – наявність помилок розрахунку значень параметра початку виконання операторів.

При формальному синтезі часопараметризованих паралельних моделей задач можуть виникнути «часові» помилки, які спотворюють коректні часові відносини проходження операторів – джерел даних і операторів-споживачів.

Припустимо, що часовими помилками є:

– спотворення правильного значення параметра початку t_{20}^{H} = 51.00 оператора P_{20} на помилкове значення t_{20}^{H} = 53.00 (такти);
– спотворення правильного значення параметра початку t_{29}^{H} = 99.00 оператора P_{29} на помилкове значення t_{29}^{H} = 102.00 (такти).

Склад виявлених за допомогою компіляційної верифікації часових помилок представляє Рис. 2.10.

Як випливає з Рис. 2.10, склад операторів часової паралельної *SMP*-моделі, для яких зафіксована некоректність часових відносин, включає P_{20}, P_{21}, P_{29}, P_{30}, P_{31}. Виявлені помилки відносяться як до зв'язаних, так і зовнішніх зв'язків операторів кожної пари операторів з некоректними часовими відносинами: (P_{20}, P_{21}); (P_{29}, P_{30}); (P_{29}, P_{31}).

Семантика виявлених часових помилок для кожної пари операторів:

– виконання оператора P_{21}, що є зовнішнім для оператора P_{20}, починається раніше завершення виконання P_{20}, результати реалізації якого повинен використовувати оператор P_{21};

– виконання оператора P_{20}, що є зв'язаним для оператора P_{21}, починається пізніше початку виконання P_{21}, який повинен використовувати результати реалізації оператора P_{20};

– виконання оператора P_{30}, що є зовнішнім для оператора P_{29}, починається раніше завершення виконання P_{29}, результати реалізації якого повинен використовувати оператор P_{30};

– виконання оператора P_{29}, що є зв'язаним для оператора P_{30}, починається пізніше початку виконання P_{30}, який повинен використовувати результати реалізації оператора $P29;$

– виконання оператора P_{31}, що є зовнішнім для оператора P_{29}, починається раніше завершення виконання P_{29}, результати реалізації якого повинен використовувати оператор P_{31};

– виконання оператора P_{29}, що є зв'язаним для оператора P_{31}, починається пізніше початку виконання P_{31}, який повинен використовувати результати реалізації оператора P_{29}.

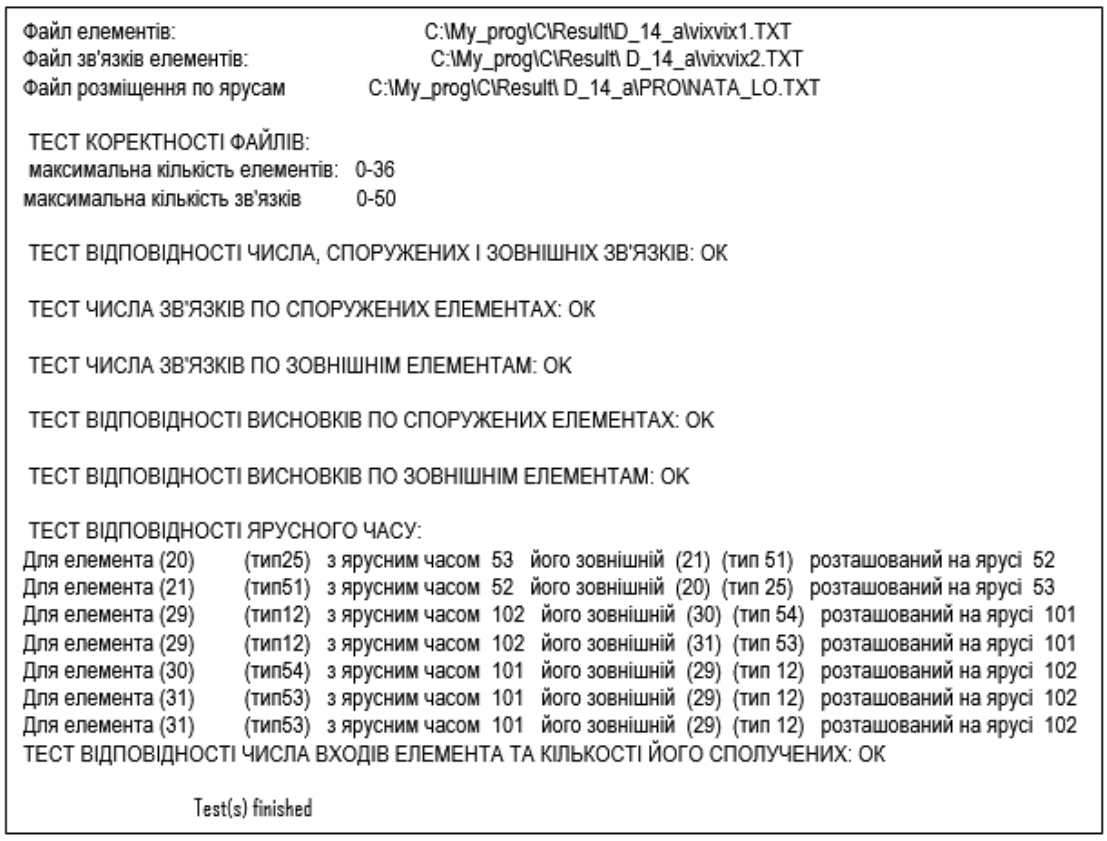


Рис. 2.10. Результати верифікації структур СЧС і часової паралельної моделі при наявності часових помилок (*View of test results*)

Висновок до розділу 2

У розділі описано етапи модифікації методу паралельних часопараметризованих програм для інформаційних управляючих систем, що містить етапи верифікації структур семантико-числової специфікації послідовної програми, часової паралельної моделі програми з урахуванням особливостей обраної раціональної сукупності методів паралельної обробки інформації, часопараметризованої паралельної програми та перевірку відповідності показників ефективності заданим вимогам і обмеженням.

Дана загальна характеристика методу, представлені основні етапи компіляційної верифікації процесів синтезу часопараметризованих мультипаралельних програм ІУС. Наведено приклад компіляційної верифікації синтезу часопараметризованої паралельної програми задачі для ілюстрації застосування компіляційної верифікації паралельних програмних засобів ІУС.

Для візуалізації результатів компіляційної верифікації використовується гістограмне представлення верифікації розподілу множини часових операторів паралельної моделі по часовим ярусам з більшею деталізацією результатів верифікації, яке відображає результати перевірки коректності розташування часових операторів і множинних часових операторів на часових ярусах для часової моделі і структур СЧС.

Основні положення цього розділу викладені у публікаціях автора [1, 2, 5, 9, 23, 24, 27, 29].

РОЗДІЛ 3
МОДИФІКАЦІЯ МЕТОДУ ДЕКОМПІЛЯЦІЙНОЇ ВЕРИФІКАЦІЇ ЧАСОПАРАМЕТРИЗОВАНИХ ПАРАЛЕЛЬНИХ ПРОГРАМНИХ ЗАСОБІВ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ

Зважаючи на обґрунтовану важливість забезпечення надійності та коректності програмного забезпечення в сучасних інформаційних управляючих системах, модифікація методу декомпіляційної верифікації виявляється дослідженням, що привертає значну увагу. Цей метод спрямований на поєднання аналізу бінарного виконуваного коду з подальшою формальною верифікацією декомпільованого вхідного коду, з метою впевненості в його коректності та властивостях.

3.1. Постановка задачі декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС

Основними етапами формального синтезу часопараметризованих паралельних програмних засобів інформаційних управляючих систем є:

– синтез структур семантико-числової специфікації послідовної програми задачі (наприклад, послідовної програми);

– синтез графічної специфікації послідовної програми у вигляді графа;

– синтез часової паралельної архітектурно-орієнтованої моделі для заданої конфігурації програмного забезпечення і вимог або обмежень;

– синтез текстових часових програм процесорів обчислювальних систем.

З формальної точки зору виконання вищезазначених етапів полягає в переході від початкового тексту послідовної програми задачі, що представлена структурами семантико-числової специфікації, до структур СЧС, які представляють специфікації часопараметризованих паралельних програм. Виконання дій, що забезпечують такий перехід, є розширенням складу елементів структур СЧС об’єктів кожного попереднього етапу шляхом

«додавання» операторів, що відображають додаткові (статичні та часові) перетворення даних в рамках того або іншого етапу, і введення нових або корекції наявних пов’язаних і зовнішніх зв’язків між операторами. Останнім етапом є перехід від структур семантико-числової специфікації, що представляють результат формального синтезу (часопараметризовану модель, часову паралельну програму), до візуалізації результату в текстовому або графічному форматі [13, 56–57].

Суть декомпіляціоної верифікації можна представити як процес, зворотній по відношенню до процесу компіляційної верифікації – видалення зі структур СЧС елементів введених при синтезі кожного з етапів формального синтезу програми і оцінених компіляційною верифікацією як «правильні». Підсумком процесу декомпіляціоної верифікації, в разі відсутності помилок розробки, має бути отримання автоматично декомпільованого тексту, що збігається з текстом вхідної послідовної програми задачі.

Вхідними даними для декомпіляціоної верифікації є наступні:

1. Текстова специфікація задачі у вигляді послідовної програми на мові програмування (наприклад Сі).

2. Структури СЧС послідовної програми задачі та відповідний граф.

3. Результати формального синтезу часопараметризованих паралельних об’єктів: синтезованих часових мультипаралельних моделей вирішення задачі (відповідних їх майбутньої програмної реалізації); часові мультипаралельні програми.

4. Структури семантико-числової специфікації перелічених об’єктів синтезу.

В якості критерію «правильності» використовується логічна еквівалентність текстової специфікації задачі, що отримана на основі декомпіляційної верифікації та вхідної текстової специфікації задачі.

Семантику основних етапів декомпіляційної верифікації представлено на Рис. 3.1.

Рис. 3.1. Основні етапи декомпіляційної верифікації паралельних програмних засобів ІУС.

Перший етап (символ 2 Рис. 3.1).

Змістом етапу є зворотний перехід від синтезованих в процесі компіляційної верифікації структур СЧС_*M* семантико-числової специфікації часопараметризованої мультипаралельної моделі *(P)* до структур СЧС вхідної послідовної програми задачі (структури *BF* та *CF*) та перевірка їх логічної еквівалентності за критерієм збігу кількості та «семантики» операторів та їх зв’язків.

Другий етап (символ 3 Рис. 3.1).

При коректності структур СЧС_*M* моделі виконується формальний синтез структур СЧС_*P* паралельної програми, виходячи з структур СЧС_*M*

мультипаралельної послідовної програми, а також формальний синтез відповідних текстових і графічних описів паралельної програми.

Третій етап (символ 4 Рис. 3.1).

Змістом етапу є перевірка коректності виконаного на другому етапі переходу від структур СЧС_*M* моделі до структур СЧС_*P* паралельної програми. Це досягається шляхом декомпіляції структур СЧС_*P* і порівняння результату декомпіляції структур СЧС_*PD* зі структурами СЧС послідовної програми задачі.

Четвертий етап (символ 5 Рис. 3.1).

Змістом етапу є перевірка збігу двох текстових специфікацій: текстової специфікації статичної паралельної програми синтезованого об’єкта (часопараметрізованої мультіпаралельної програми) і тексту вхідної послідовної програми. Збіг текстів є ознакою коректності всіх перетворень, що виконані при формальному і автоматичному синтезі проєктованих мультипаралельних програмних об’єктів.

3.2. Метод декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС

Декомпіляційна верифікація може бути проведена після компіляційної верифікації програмних засобів ІУС, але дослідження показали, що більш ефективним є застосування обох видів верифікації на кожному з етапів синтезу часопараметризованих мультипаралельних програм. Структурна схема методу декомпіляційної верифікації програмних засобів ІУС представлена на Рис. 3.2.

Розглянемо детальний опис процедур, які реалізують етапи методу декомпіляційної верифікації програмних засобів ІУС.

Виконання першого з етапів декомпіляційної верифікації програмних засобів, представлених на Рис. 3.1 забезпечується за рахунок виконання формальної декомпіляційної верифікації часопараметризованих мультипаралельних моделей задач.

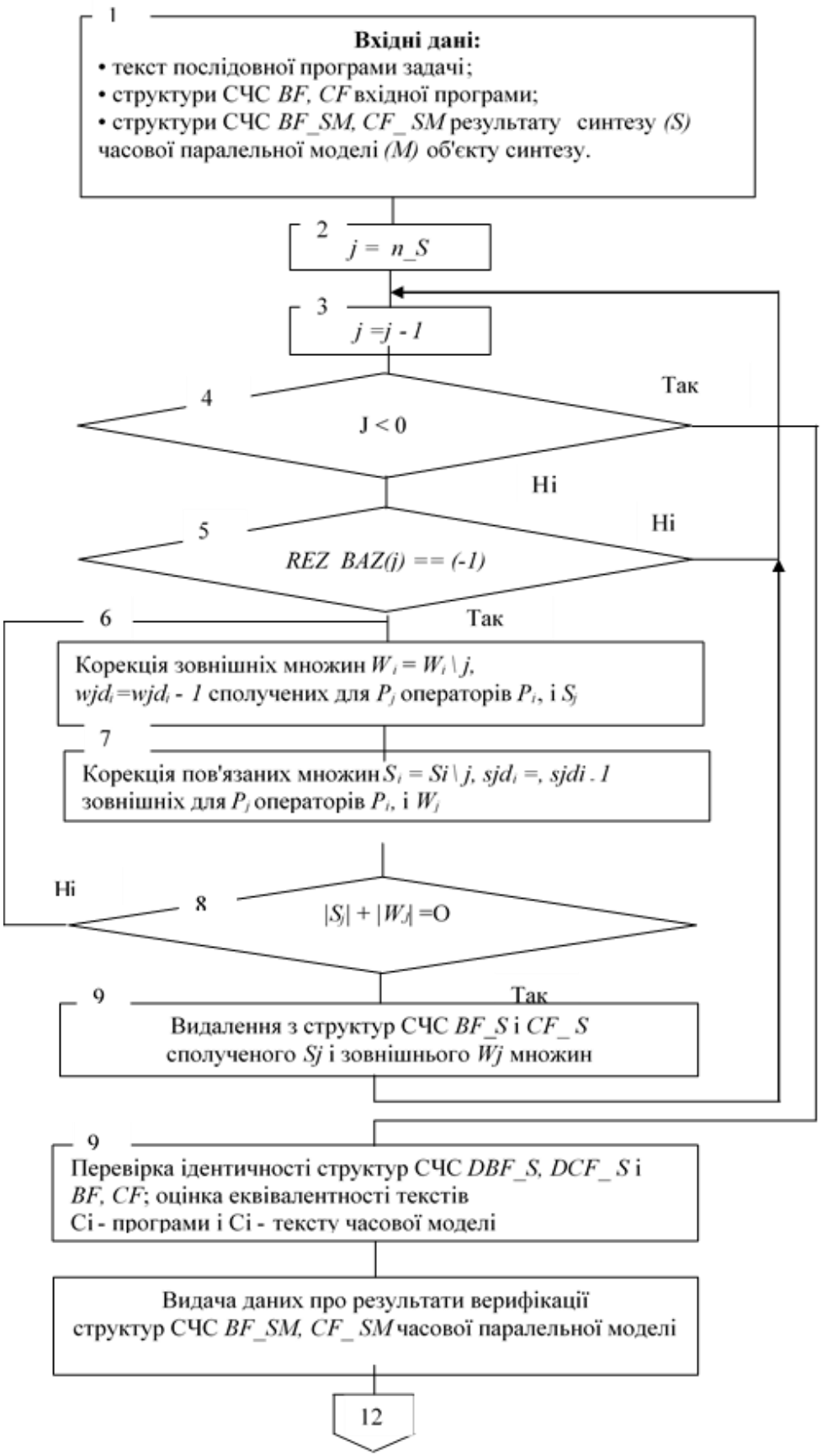


Рис. 3.2 Метод декомпіляційної верифікації часопараметризованих мультипаралельних програм *(початок)*

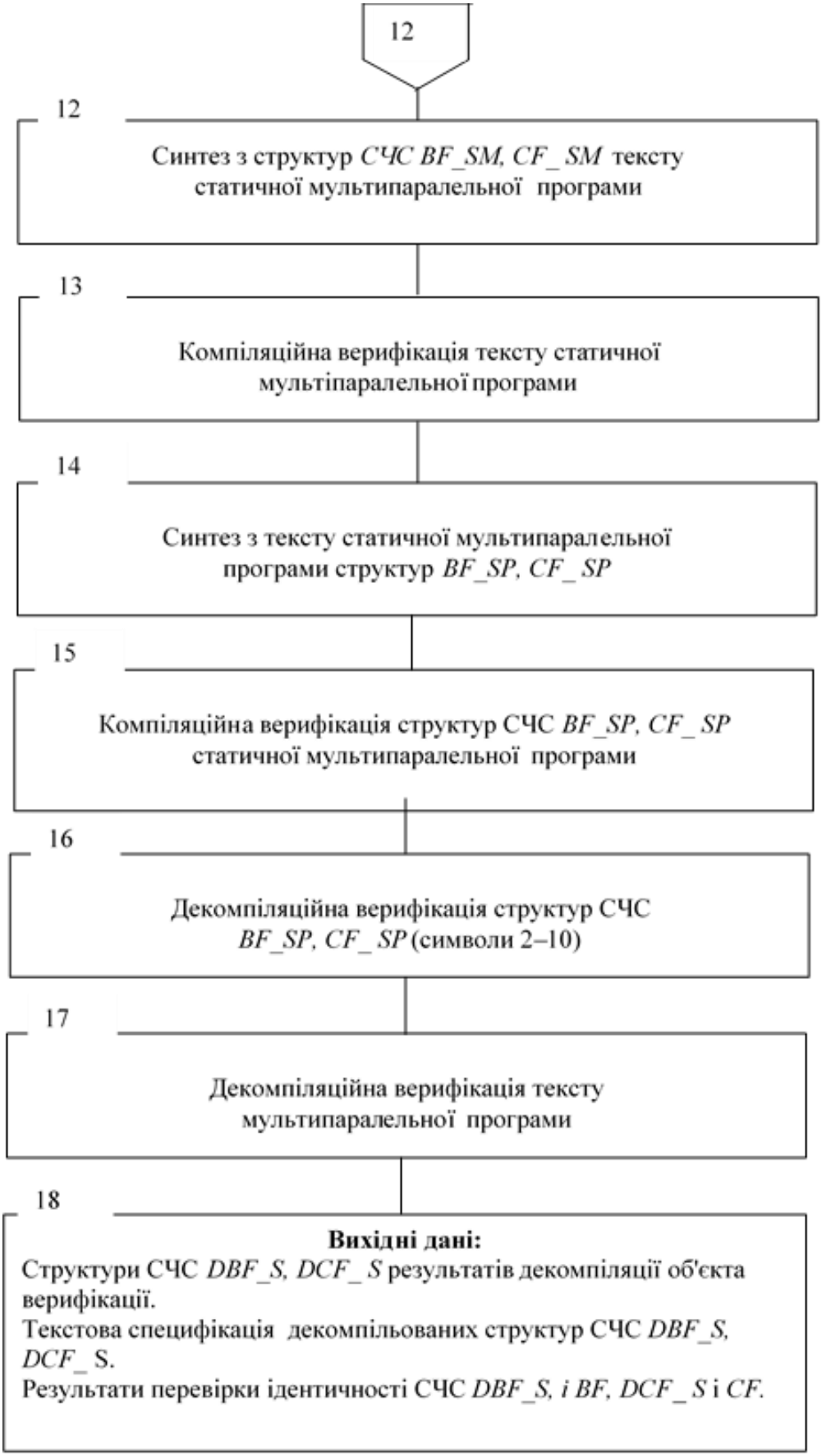


Рис. 3.2 Метод декомпіляційної верифікації часопараметризованих мультипаралельних програм *(закінчення)*

Склад і зв’язок основних етапів формальної декомпіляційної верифікації часових мультипаралельних моделей задач, представлений на Рис. 3.2.

Перший етап (символи 2 ... 5 Рис. 3.2). Змістом етапу є організація циклу по номерах операторів P_j структури *BF_CNW* (табл. 3.3) і виділення операторів, які відповідають умові (3.1)

$$RES\_BEG\ (j) = -1. \tag{3.1}$$

Другий етап (символ 6 Рис. 3.2). Змістом етапу є організація циклу по множині сполучених для P_j операторів $P_i\ \epsilon S_j$ і видалення з зовнішньої множини W_i кожного оператора P_i номера оператора P_j: $W_i = W_i \setminus j$, а також корекція потужностей зовнішніх множин (3.2)

$$w_j d_i = w_j d_i - 1. \tag{3.2}$$

Третій етап (символ 7 Рис. 3.2). Змістом етапу є організація циклу по множині зовнішніх для P_j операторів $P_i\ \epsilon W_j$ і видалення з сполученої множини S_i кожного зовнішнього оператора P_i номера оператора P_j:

$$S_i = S_i \setminus j, \tag{3.3}$$

а також корекція потужностей пов’язаних множин (3.4):

$$s_j d_i = s_j d_i - 1. \tag{3.4}$$

Четвертий етап (символ 8-10 Рис. 3.2). Забезпечує перевірку (3.5)

$$| S_j | + | W_J | = = 0, \tag{3.5}$$

що реалізована у вигляді

$$s_j d_j + w_j d_j == 0 \tag{3.6}$$

ізольованості поточного оператора P_j (відсутність зв'язків з іншими операторами) і, при виконанні умови, виключає рядок оператора P_j зі структури СЧС *BF_SM*.

Результатом виконання перелічених дій над структурами семантико-числової специфікації *BF_S і CF_ S* паралельної моделі є (в разі коректності цих структур) отримання декомпільованих структур *DBF_S і DCF_ S*, співпадаючих зі структурами СЧС *BF* і *CF* (наприклад, табл.3.1 і табл.3.2) вихідної послідовної програми (символ 11 Рис. 3.2).

Змістом другого етапу декомпіляційної верифікації паралельних програмних засобів ІУС (Рис. 3.1) є перевірка «правильності» текстової специфікації паралельної моделі (наприклад, Рис. 3.6). Шляхом реалізації цієї перевірки є видалення імен даних і операторів, що входять до складу «паралельно-конвеєрного розширення» або «паралельно-декомпозиційного розширення»– редукція тексту часопараметризованої мультипаралельної моделі.

Результатом виконання етапів 3–5 із представлених на Рис. 3.1 є перевірка логічної відповідності тексту часопараметризованої мультпаралельної програми, спроєктованої або заданої для вхідної послідовної програми задачі.

В якості вхідних даних для виконання з 3 по 5 етапів декомпіляційної верифікації мультипаралельних програм ІУС є наступні:

– вхідний текст послідовної програми;

– текстова специфікація часопараметризованої мультипаралельної програми задачі.

П'ятий етап (символ 12 Рис. 3.2). Змістом етапу є синтез з коректних структур *СЧС BF_SM CF_ SM* часопараметризованої паралельної моделі задачі тексту статичної мультипаралельної програми (мається на увазі програма, написана на мові послідовного програмування, наприклад Сі++, але з додаванням додаткових даних і операторів для реалізації раціональної

сукупності методів паралельної обробки інформації, параметр часу в таких програмах відсутній).

Рішення задачі п’ятого етапу розглянуті в [58].

Шостий етап (символ 13 Рис. 3.2). На цьому етапі здійснюється компіляційна верифікація тексту статичної мультипаралельної програми, яка описана в розділі 2.

На *сьомому етапі* (символи 14 та 15 Рис. 3.2) з тексту статичної мультипаралельної програми синтезуються структури семантико-числової специфікації програми *BF_SP* і *CF_ SP* згідно методики, яка представлена в [58]. Після цього проводиться компіляційна верифікація структур СЧС *BF_SP*, *CF_ SP* статичної мультипаралельної програми.

Восьмий етап (символ 16 Рис. 3.2). Змістом етапу є видалення зі структур СЧС *BF_SP* і *CF_ SP*, що є формальною специфікацією заданої мультипаралельної програми, операторів та зв'язків, введених у вхідну послідовну програму задачі для забезпечення можливості застосування раціональної сукупності методів паралельної обробки даних відповідно до вимог і обмежень замовника. Етап виконується відповідно до кроків декомпіляційної верифікації часових мультипаралельних моделей задач, викладених в етапах з 1 по 4 даного методу. Результатом виконання восьмого етапу є отримання декомпільованих структур *DBF_SP*, *DCF_SP* мультипаралельної програми.

Дев’ятий этап (символ 17 Рис. 3.2). Забезпечує перевірку логічної еквівалентності структур СЧС *BF*, *CF* вхідної послідовної програми та декомпільованих структур *DBF_SP*, *DCF_SP* мультипаралельної програми і, при виконанні умови (*DBF_SP == BF*) & (*DCF_SP == CF*), видає інформацію про коректність мультипаралельної програми та її логічній еквівалентності вхідній послідовній програмі задачі.

3.3. Приклади застосування методу декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС.

Семантику основних етапів декомпіляційної верифікації паралельних програмних засобів ІУС пояснімо на прикладі паралельно-конвеєрної моделі задачі, яка синтезована з використанням методу суміщення незалежних операцій і методу конвеєрної обробки інформації.

Вхідними даними для прикладу декомпіляційної верифікації мультипаралельних програм є:

– вхідний текст послідовної програми (послідовної програми) (Рис. 3.3), структури СЧС (табл. 3.1, табл. 3.2) та граф задачі (Рис. 3.4);

– структури семантико-числової специфікації *BF_CNW, CF_CNW* паралельно-конвеєрної моделі задачі, що представлені в табл. 3.3 і табл. 3.4;

– паралельно-конвеєрна модель задачі в графовому вигляді, представлена на Рис. 3.5;

– тривалості виконання операторів паралельно-конвеєрної моделі (табл.3.5);

– необхідна величина такту паралельно-конвеєрної моделі задачі $DTD_{зад}=44$ *нс.*

```
#include <stdio.h>
void main(void)
{
int a,b, k,z,p,s;
        scanf(" %d ", &a);
        scanf (" %d ", &b);
        if (a == b)
        { k = a % 2; printf(" %3d ", k);
         z = a * b; printf(" %3d ", z);
        }
        else
        { p = a | b; printf(" %3d ", p);
         s = b / a; printf(" %3d ", s);
        }
}
```

Рис. 3.3. Вхідна послідовна програма (Послідовна програма).

Таблиця 3.1.

Базова структура *BF* операторів вхідної послідовної програми

N	*MET*	*TYP*	*NSJ*	*SJD*	*BJ*	*NWJ*	*WJD*	*MP1*	*MP2*	*VH*	*VIH*	*RES*	*N*	*MET*	*TYP*	*NSJ*	*SJD*	*BJ*	*NWJ*	*WJD*	*MP1*	*MP2*	*VH*	*VIH*	*RES*
0	0	58	–1	0	0	0	1	0	0	0	1	*a_in*	15	0	3	12	3	1	26	1	0	0	3	1	*
1	0	58	–1	0	0	1	1	0	0	0	1	*b in*	16	0	12	15	2	1	27	2	0	0	2	2	=
2	0	47	–1	0	0	2	1	0	0	0	2	*a*	17	0	50	17	2	1	29	1	3	0	2	1	*bp*
3	0	47	–1	0	0	3	1	0	0	0	2	*b*	18	2	34	19	3	2	30	1	0	0	3	1	*|*
4	0	47	–1	0	0	4	1	0	0	0	2	*k*	19	0	12	22	2	2	31	2	0	0	2	2	=
5	0	47	–1	0	0	5	1	0	0	0	2	*z*	20	0	4	24	3	2	33	1	0	0	3	1	*/*
6	0	47	–1	0	0	6	1	0	0	0	2	*P*	21	0	12	27	2	2	34	2	0	0	2	2	=
7	0	47	–1	0	0	7	1	0	0	0	2	*s*	22	0	50	29	2	2	36	1	3	0	2	1	*bp*
8	0	12	0	2	0	8	5	0	0	2	1	=	23	3	54	31	2	3	37	1	0	0	2	1	*l.o*
9	0	12	2	2	0	13	4	0	0	2	1	=	24	0	49	33	1	3	–1	0	0	0	1	0	*stop*
18	0	23	4	2	0	17	1	0	0	2	1	==	25	0	48	34	1	3	–1	0	0	0	1	0	*k_out*
11	0	51	6	1	0	18	4	1	2	1	2	*upl*	26	0	48	35	1	3	–1	0	0	0	1	0	*z_out*
12	0	57	–1	0	1	22	1	0	0	0	1	*C2_*	27	0	48	36	1	3	–1	0	0	0	1	0	*p_out*
13	1	5	7	3	1	23	1	0	0	3	1	*%*	28	0	48	37	1	3	–1	0	0	0	1	0	*s_out*
14	0	12	10	2	1	24	2	0	0	2	2	=													

На Рис. 3.4 показано граф вхідної програми задачі, текст якої представлений на Рис. 3.2.

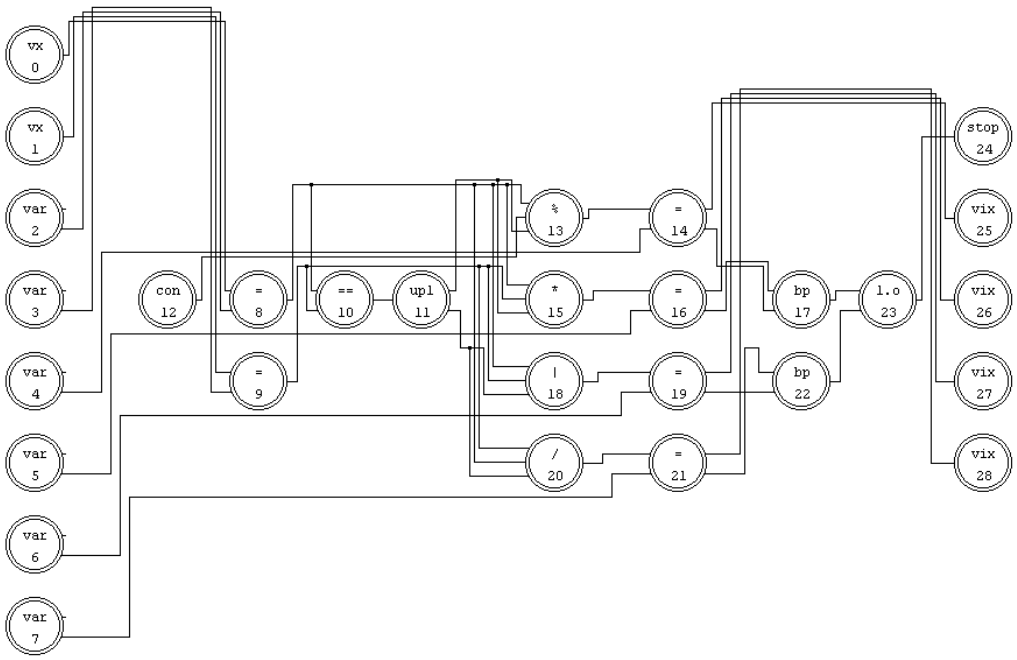


Рис. 3.4. граф вхідної послідовної програми.

Таблиця 3. 2.

Структура *CF* зв’язків операторів вхідної послідовної програми

N	***SJD***	***SPJD***	***SNWIH***	***SNWHO***	***TSS***	***JWD***	***WPJD***	***WNWHO***	***WNWIH***	***TVS***	***N***	***SJD***	***SPJD***	***SNWIH***	***SNWHO***	***TSS***	***JWD***	***WPJD***	***WNWHO***	***WNWIH***	***TVS***
0	1	0	0	0	0	–1	8	0	0	0	19	20	8	0	0	0	20	15	2	0	1
1	–1	2	1	1	2	–1	9	0	0	0	20	21	9	0	1	0	21	18	2	1	1
2	3	1	0	0	0	–1	8	1	1	2	21	–1	11	1	2	1	–1	20	2	1	1
3	–1	3	1	1	2	–1	9	1	1	2	22	23	6	1	1	2	–1	13	1	0	0
4	5	8	0	0	0	–1	14	1	1	2	23	–1	18	0	0	0	–1	14	0	0	0
5	–1	9	0	1	0	–1	16	1	1	2	24	25	9	0	0	0	25	17	1	1	1
6	–1	10	0	0	0	–1	19	1	1	2	25	26	8	0	1	0	–1	25	0	0	0
7	8	8	0	0	0	–1	21	1	1	2	26	–1	11	1	2	1	–1	16	0	0	0
8	9	12	0	1	0	9	10	0	0	0	27	28	7	1	1	2	28	17	0	1	1
9	–1	11	0	2	1	10	13	0	0	0	28	–1	20	0	0	0	–1	26	0	0	0
10	11	4	1	1	2	11	15	0	0	0	29	30	21	1	0	1	–1	23	0	0	1
11	–1	13	0	0	0	12	18	0	0	0	30	–1	19	1	1	1	–1	19	0	0	0
12	13	8	0	0	0	–1	20	1	0	0	31	32	22	0	1	1	32	22	1	1	1
13	14	9	0	1	0	14	10	1	0	0	32	–1	17	0	0	1	–1	27	0	0	0
14	–1	11	0	2	1	15	15	1	0	0	33	–1	23	0	0	1	–1	21	0	0	0
15	16	5	1	1	2	16	18	1	0	0	34	–1	14	0	0	0	35	22	0	1	1
16	–1	15	0	0	0	–1	20	0	0	0	35	–1	16	0	0	0	–1	28	0	0	0
17	18	16	1	0	1	–1	11	0	0	0	36	–1	19	0	0	0	–1	23	1	0	1
18	–1	14	1	1	1	19	13	2	0	1	37	–1	21	0	0	0	–1	24	0	0	1

На Рис. 3.5 представлена часова паралельно-конвеєрна модель задачі з величиною такту конвеєрної ступені *DTD* = 42.34 нс.

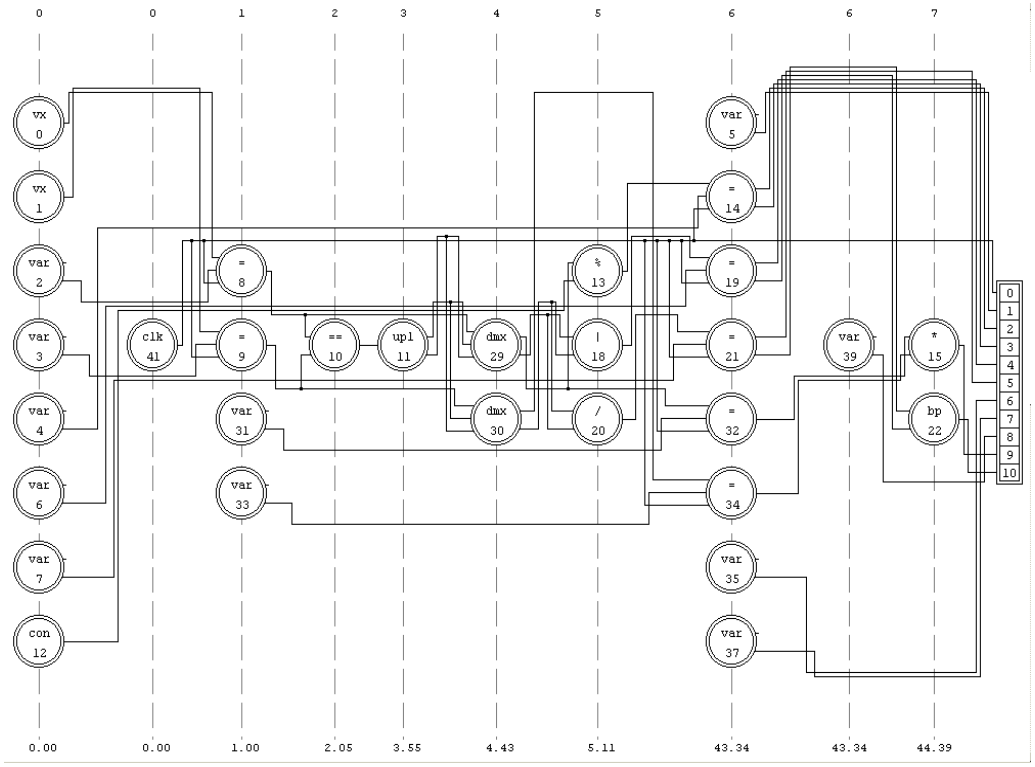


Рис. 3.5. Часова паралельно-конвеєрна модель (*DTD = 42.34 нс, початок*)

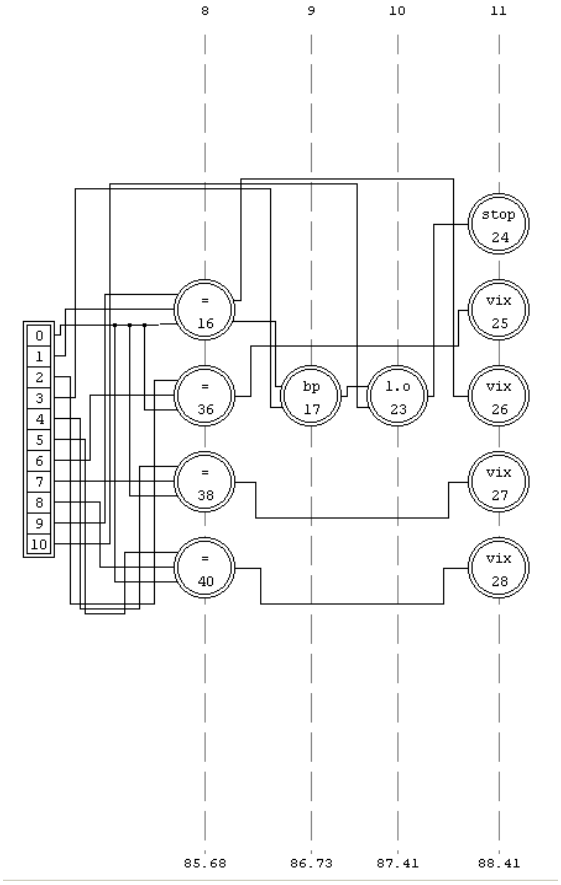


Рис. 3.5. Часова паралельно-конвеєрна модель (*DTD = 42.34 нс. закінчення*).

Тривалість виконання різних типів операцій являє собою таблиця 3.3.

Таблиця 3.3

Тривалість t_o (typ) виконання операторів P_j конвеєрної моделі (нс)

typ	vx	var	/	bp	clk	vix	*	dmx	l.o	ctr	==	*,=	upl	stop
t_j^0	1.0	1.0	11.0	0.68	1.0	1.0	41.29	0.68	1.0	1.7	1.5	1.05	1.5	1.0

Масиви структури *BF_CNW* мають наступну семантику:

N – масив номерів операторів часової паралельно-конвеєрної моделі (Рис. 3.5),

TYP – масив номерів «типів» операторів,

NSJ – маса покажчиків на початок ланцюжка номерів пов'язаних операторів P_i (в структурі зв'язків *CF_CNW*) для поточного оператора P_j,

SJD – масив кількості пов'язаних операторів P_i (в структурі зв'язків *CF_CNW*) для поточного оператора P_j,

BJ – масив, що задає номер b_j природної частини, якій належить оператор P_j,

NWJ – масив покажчиків на початок ланцюжка номерів зовнішніх операторів P_i (в структурі зв'язків *CF_CNW*) для поточного оператора P_j,

WJD – масив кількості зовнішніх операторів P_i (в структурі зв'язків *CF_CNW*) для поточного оператора P_j,

BJF – масив, що задає для кожного оператора P_j номер ступені (фрагмента) конвеєра, якій він належить,

NT – масив, що задає для кожного оператора P_j момент t_j^H початку виконання оператора,

NY – масив номерів часових ярусів паралельно-конвеєрної моделі, яким належать оператори з однаковим часом початку виконання,

RES – масив «ідентифікаторів» операторів паралельної моделі,

RES_BEG – масив, що задає інформацію про склад додаткових операторів P_j, введених при формальному синтезі паралельно-конвеєрної моделі (значення *RES_BEG (j) = – 1*), про склад операторів структури *BF*

вихідної послідовної програми, які не змінювалися при синтезі паралельно – конвеєрної моделі (значення *RES_BEG (j) = 0*) і про операторів – іменах даних структури *BF*, які були змінені і підлягають відновленню при декомпіляціній верифікації (значення *RES_BEG (j)* задає початкове або базове ім'я оператора – цієї структури *BF* послідовної програми.

На першому етапі (символи 2...5 Рис. 3.2) в структурі *BF_CNW* (табл.3.4) виділяються оператори, що задовольняють умові RES_*BEG (j) = – 1*. Такими операторами є оператори P_J введені в структуру *BF* вхідної програмив інтересах забезпечення можливості застосування паралельно-конвеєрного методу обробки даних. У структурі *BF_CNW* (табл. 3.4) в їх склад входять оператори P_{29}, P_{30}, ... P_{41}.

Для кожного оператора P_J (P_{29}, P_{30}, ... P_{41}), в зв'язку з їх подальшим видаленням зі структури *BF_CNW* виконується попереднє видалення їх зв'язків з операторами – джерелами даних і з операторами, що використовують результати виконання операторів P_J (P_{29}, P_{30}, ... P_{41}), задаються в структурі *CF_CNW* сполученими S_j і зовнішніми W_j множинами.

Після видалення для P_J всіх його сполучених і зовнішніх зв'язків оператор P_J і все його атрибути, тобто рядок з номером *j* структури *BF_CNW* виключається з цієї структури. Ці дії становлять зміст другого, третього та четвертого етапів (символи 6, 7, 8, 9 Рис. 3.2).

Для першого з видалених операторів $P_J = P_{41}$ потужність $sjd_{41} = 0$ (поєднана множина є порожньою $S_j = 0$), зовнішня множина (покажчик $nwj_{41} = 50$) має потужність $wjd_{41} = 11$ і $W_j = (8, 9, 14, 16, 19, 21, 32, 34, 36, 38, 40)$. Відповідно оператору P_{41} множина елементів видаляється з рядків з номерами $N = 50 ... 60$ масивів *JWD, WPJD, WNWHO, WNWIH, TVS* і з рядка з номером $N = 60$ масивів *JSD, SPJD, SNWIH, SNWHO, TSS*, відповідних порожній поєднаній множині $S_{41} = 0$.

На завершення зі структури *BF_CNW* (табл.3.4) видаляється рядок з номером $N = 41$.

Таблиця 3.4.

Базова структура *BF_CNW* операторів
часової паралельно-конвеєрної моделі послідовної програми

N	*TYP*	*NSJ*	*SJD*	*BJ*	*NWJ*	*WJD*	*BJF*	*NT*	*NY*	*RES*	*RES_BEG*	*N*	*TYP*	*NSJ*	*SJD*	*BJ*	*NWJ*	*WJD*	*BJF*	*NT*	*NY*	*RES*	*RES_BEG*
0	58	–1	0	0	0	1	0	0.00	0	*a_in*	0	21	12	23	3	2	29	2	1	43.34	6	=	0
1	58	–1	0	0	1	1	0	0.00	0	*b_in*	0	22	50	25	2	2	31	1	1	44.39	7	*bp*	0
2	47	–1	0	0	2	1	0	0.00	0	*a*	0	23	54	27	2	3	32	1	2	87.41	10	*l.o*	0
3	47	–1	0	0	3	1	0	0.00	0	*b*	0	24	49	29	1	3	–1	0	2	88.41	11	*stop*	0
4	47	–1	0	0	4	1	0	0.00	0	*k_l*	k	25	48	30	1	3	–1	0	2	88.41	11	*k_out*	0
5	47	–1	0	0	5	1	1	43.34	6	*z 2*	z	26	48	31	1	3	–1	0	2	88.41	11	*z_out*	0
6	47	–1	0	0	6	1	0	0.00	0	*p_l*	p	27	48	32	1	3	–1	0	2	88.41	11	*p_out*	0
7	47	–1	0	0	7	1	0	0.00	0	*s_1*	s	28	48	33	1	3	–1	0	2	88.41	11	*s_out*	0
8	12	0	3	0	8	2	0	1.00	1	=	0	29	56	34	3	0	33	4	0	4.43	4	*dmx*	–1
9	12	2	3	0	10	2	0	1.00	1	=	0	30	56	37	3	0	37	3	0	4.43	4	*dmx*	–1
10	23	4	2	0	12	1	0	2.05	2	==	0	31	47	–1	0	1	40	1	0	1.00	1	*a_l*	–1
11	51	6	1	0	13	4	0	3.55	3	*upl*	0	32	12	40	3	1	41	1	1	43.34	6	=	–1
12	57	–1	0	0	17	1	0	0.00	0	*C2_*	0	33	47	–1	0	1	42	1	0	1.00	1	*b_l*	–1
13	5	7	2	1	18	1	0	5.11	5	*%*	0	34	12	42	3	1	43	1	1	43.34	6	=	–1
14	12	9	3	1	19	2	1	43.34	6	=	0	35	47	–1	0	1	44	1	1	43.34	6	*k_2*	–1
15	3	11	2	1	21	1	1	44.39	7	*	0	36	12	44	3	1	45	1	2	85.68	8	=	–1
16	12	13	3	1	22	2	2	85.68	8	=	0	37	47	–1	0	2	46	1	1	43.34	6	*p_2*	–1
17	50	15	2	1	24	1	2	86.73	9	*bp*	0	38	12	46	3	2	47	1	2	85.68	8	=	–1
18	34	17	2	2	25	1	0	5.11	5	/	0	39	47	–1	0	2	48	1	1	43.34	6	*s_2*	–1
19	12	19	3	2	26	2	1	43.34	6	=	0	40	12	48	3	2	49	1	2	85.68	8	=	–1
20	4	21	2	2	28	1	0	5.11	5	/	0	41	65	–1	0	0	50	11	0	0.00	0	*clk*	–1

Стосовно до видалення оператора P_{32} типу «=» записи в пам’ять значення змінної *a_1* відповідне поєднане S_{32} (покажчик на початок ланцюжка операторів поєднаної множини ($nsj_{32} = 40$, $sjd_{32} = 3$) і зовнішнє W_{32} (покажчик на початок ланцюжка операторів зовнішнього множини $nw_j = 41$, $wjd_{32} = 1$) множини мають такий вигляд: S_{32} = = (*29, 31, 41*), W_{32} = *(15).* Відповідні оператору P_{32} множини видалення елементів формується рядками *40, 41, 56* з структур СЧС *BF_CNW* і *CF_CNW*, які задають безліч *S32*, рядком *41*, яка задає зовнішнє безліч *W32*, і рядком 32 у структурі *BF_CNW*, що містить базові атрибути оператора *P32*.

Завершення виключення операторів P_J (P_{29}, P_{30}, ... P_{41}) забезпечує перехід до етапу відновлення «модифікованих» (в інтересах підтримки програмою методу конвеєрної обробки) ідентифікаторів даних масиву *RES* шляхом їх заміни на вхідні імена, які зберігаються в масиві *RES_BEG* з подальшою перевіркою ідентичності декомпільованої структури СЧС паралельно-конвеєрної програми і відповідних структур СЧС *BF, CF* вихідної послідовної програми.

Таблиця 3.5

Структура *CF_CNW* зв'язків операторів
часової паралельно-конвеєрної моделі послідовної програми

N	JSD	SPJD	SNVIH	SNVHO	JVD	VPJD	VNWHO	WNWIH	N	JSD	SPJD	SNVIH	SNVHO	JVD	VPJD	VNWHO	WNWIH
0	1	0	0	0	–1	8	0	0	31	–1	16	0	0	–1	23	1	0
1	50	2	1	1	–1	9	0	0	32	–1	38	0	0	–1	24	0	0
2	3	1	0	0	–1	8	1	1	33	–1	40	0	0	34	13	0	0
3	51	3	1	1	–1	9	1	1	34	35	8	0	0	35	18	0	1
4	5	8	0	0	–1	14	1	1	35	36	11	0	1	36	20	1	1
5	–1	9	0	1	–1	16	1	1	36	–1	11	1	2	–1	32	0	0
6	–1	10	0	0	–1	19	1	1	37	38	9	0	0	38	18	1	1
7	8	29	0	0	–1	21	1	1	38	39	11	0	1	39	20	0	1
8	–1	12	0	1	9	10	0	0	39	–1	11	1	2	–1	34	0	0
9	10	13	0	0	–1	29	0	0	40	41	29	0	0	–1	32	1	1
10	52	4	1	1	11	10	1	0	41	56	31	1	1	–1	15	0	0
11	12	32	0	0	–1	30	0	0	42	43	30	0	0	–1	34	1	1
12	–1	34	0	1	–1	11	0	0	43	57	33	1	1	–1	15	1	0
13	14	15	0	0	14	29	1	0	44	45	14	0	0	–1	36	1	1
14	53	5	1	1	15	29	2	1	45	58	35	1	1	–1	25	0	0
15	16	16	1	0	16	30	1	0	46	47	19	0	0	–1	38	1	1
16	–1	14	1	1	–1	30	2	1	47	59	37	1	1	–1	27	0	0
17	18	29	1	0	–1	13	1	0	48	49	21	0	0	–1	40	1	1
18	–1	30	1	1	–1	14	0	0	49	60	39	1	1	–1	28	0	0
19	20	18	0	0	20	17	1	1	50	–1	41	0	2	51	8	2	0
20	54	6	1	1	–1	36	0	0	51	–1	41	0	2	52	9	2	0
21	22	30	1	0	–1	16	0	0	52	–1	41	0	2	53	14	2	0
22	–1	29	1	1	23	17	0	1	53	–1	41	0	2	54	16	2	0
23	24	20	0	0	–1	26	0	0	54	–1	41	0	2	55	19	2	0
24	55	7	1	1	–1	23	0	0	55	–1	41	0	2	56	21	2	0
25	26	21	1	0	–1	19	0	0	56	–1	41	0	2	57	32	2	0
26	–1	19	1	1	27	22	1	1	57	–1	41	0	2	58	34	2	0
27	28	17	0	0	–1	38	0	0	58	–1	41	0	2	59	36	2	0
28	–1	22	0	1	–1	21	0	0	59	–1	41	0	2	60	38	2	0
29	–1	23	0	0	30	22	0	1	60	–1	41	0	2	–1	40	2	0
30	–1	36	0	0	–1	40	0	0									

```
#include <stdio.h>
void main(void)
{
int a,b,k_1,z_2,p_1,s_1,a_1;
int b_1,k_2,p_2,s_2;
scanf("%d",&a);
scanf("%d",&b);
if (a == b)
{
k_1 = a % 2;
a_1 = a;
b_1 = b;
k_2 = k_1;
z_2 = a_1 * b_1;
printf(" %3d ",k_2);
printf(" %3d ",z_2);
}
else
{
p_1 = a | b;
s_1 = b / a;
p_2 = p_1;
s_2 = s_1;
printf(" %3d ",p_2);
printf(" %3d ",s_2);
}


}
```

Рис. 3.6 Текстова специфікація паралельно-конвеєрної моделі вхідної послідовної програми (послідовної програми).

Використовуючи розглянутий вище (для табл. 3.4 та 3.5) алгоритм «видалення», можна перевірити: якщо в розглянутому прикладі декомпіляційна верифікація завершується позитивним результатом – має місце збіг структур *DBF_CNW i BF*, а також структур *DCF_CNW i CF* (табл. 3.1 та 3.2) вхідної послідовної послідовної програми.

Як зазначалося, змістом першого етапу декомпіляційної верифікації є автоматичний синтез текстового подання паралельно-конвеєрної послідовної програми, виходячи з структур семантико-числової специфікації *BF_CNW, CF_CNW* паралельно-конвеєрної моделі задачі (табл.3.4 і табл.3.5). Результат виконання першого етапу декомпіляційної верифікації для розглянутого прикладу представляє Рис. 3.6. Порівняння вхідної програми(Рис. 3.3) і

паралельно-конвеєрної моделі (Рис. 3.6) показує, що використання методу поєднання незалежних операцій конвеєрної обробки досягається автоматичним «паралельно-конвеєрним розширенням» складу імен даних і відповідною зміною складу операторів вхідної програми(для першого ступеня конвеєра це k_1, a_1, b_1, p_1, s_1; для другого ступеня конвеєра це k_2, z_2, p_2, s_2). Задля доказу ідентичності вхідної програмиі тексту паралельно-конвеєрної моделі задачі необхідно провести редукцію. Для розглянутого прикладу редукція полягає в автоматичному виконанні наступних операцій:

а) виключення з імен даних в текстовій специфікації добавок до імен, що мають вигляд «ім’я даної вихідної послідовної програми_ номер ступені конвеєра» (Результат цього Кроку представляє текст Рис. 3.7);

```
#include <stdio.h>
 void main(void)
{
 int a,b;
        scanf("%d",&a);
        scanf("%d",&b);
        if (a == b)
        {
         k= a % 2;
         z = a * b;
         printf(" %3d ",k);
         printf(" %3d ",z);
        }
        else
        {
         p = a | b;
         s = b / a;
         printf(" %3d ",p);
 printf(" %3d ",s);
        }
```

Рис. 3.7. Вхідна Послідовна програма.

b) видалення з текстової специфікації одержаних операторів присвоювання виду «Ім’я даного вихідної програми= ім’я даного вхідної послідовної програми»:,

a=a; b=b, k=k; h=h, s=s (результат цього кроку представляє текст Рис. 3.8);

```
#include <stdio.h>
 void main(void)
{
 int a,b,k_1,z_2,p_1,s_1,a_1;
 int b_1,k_2,p_2,s_2;
        scanf("%d",&a);
        scanf("%d",&b);
        if (a == b)
        {
         k= a % 2;
         z = a * b;
         printf(" %3d ",k_2);
         printf(" %3d ",z_2);
        }
        else
        {
         p = a | b;
         s = b / a;
         printf(" %3d ",p_2);
 printf(" %3d ",s_2);
        }
```

Рис. 3.8. Результат видалення з текстової специфікації одержаних операторів присвоювання.

с) корекція операторов printf( ) та видалення з декларативної частини дублюючих імен даних (результат цього кроку представляє текст Рис. 3.9);

```
#include <stdio.h>
 void main(void)
{
 int a,b;
        scanf("%d",&a);
        scanf("%d",&b);
        if (a == b)
        {
         k= a % 2;
         z = a * b;
         printf(" %3d ",k);
         printf(" %3d ",z);
        }
        else
        {
         p = a | b;
         s = b / a;
         printf(" %3d ",p);
 printf(" %3d ",s);
        }
```

Рис. 3.9. Результат декомпіляції Сі-тексту часової паралельно-конвеєрної моделі задачі.

Наступним етапом декомпіляційної верифікації є автоматичний синтез структур *DBF_CNW, DCF_CNW* семантико-числової специфікації результату декомпіляції Сі-тексту часової паралельно-конвеєрної моделі задачі (Рис. 3.9). Результати синтезу представлені в табл. 3.6 та 3.7).

Таблиця 3.6

Базова структура *DBF_CNW* операторів декомпільованої моделі задачі

N	*MET*	*TYP*	*NSJ*	*SJD*	*BJ*	*NWJ*	*WJD*	*MP1*	*MP2*	*VH*	*VIH*	*RES*	*N*	*MET*	*TYP*	*NSJ*	*SJD*	*BJ*	*NWJ*	*WJD*	*MP1*	*MP2*	*VH*	*VIH*	*RES*
0	0	58	–1	0	0	0	1	0	0	0	1	*a i*	15	0	3	12	3	1	26	1	0	0	3	1	*
1	0	58	–1	0	0	1	1	0	0	0	1	*b in*	16	0	12	15	2	1	27	2	0	0	2	2	=
2	0	47	–1	0	0	2	1	0	0	0	2	*a*	17	0	50	17	2	1	29	1	3	0	2	1	*bp*
3	0	47	–1	0	0	3	1	0	0	0	2	*b*	18	2	34	19	3	2	30	1	0	0	3	1	/
4	0	47	–1	0	0	4	1	0	0	0	2	*k*	19	0	12	22	2	2	31	2	0	0	2	2	=
5	0	47	–1	0	0	5	1	0	0	0	2	*z*	20	0	4	24	3	2	33	1	0	0	3	1	/
6	0	47	–1	0	0	6	1	0	0	0	2	*P*	21	0	12	27	2	2	34	2	0	0	2	2	=
7	0	47	–1	0	0	7	1	0	0	0	2	*s*	22	0	50	29	2	2	36	1	3	0	2	1	*bp*
8	0	12	0	2	0	8	5	0	0	2	1	=	23	3	54	31	2	3	37	1	0	0	2	1	*l.o*
9	0	12	2	2	0	13	4	0	0	2	1	=	24	0	49	33	1	3	–1	0	0	0	1	0	*stop*
18	0	23	4	2	0	17	1	0	0	2	1	==	25	0	48	34	1	3	–1	0	0	0	1	0	*k ou*
11	0	51	6	1	0	18	4	1	2	1	2	*upl*	26	0	48	35	1	3	–1	0	0	0	1	0	*z ou*
12	0	57	–1	0	1	22	1	0	0	0	1	*C2*	27	0	48	36	1	3	–1	0	0	0	1	0	*p o*
13	1	5	7	3	1	23	1	0	0	3	1	*%*	28	0	48	37	1	3	–1	0	0	0	1	0	*s ou*
14	0	12	10	2	1	24	2	0	0	2	2	=													

Таблиця 3.7

Структура *DCF_CNW* зв’язків операторів декомпільованої моделі задачі

N	*SJD*	*SPJD*	*SNWIH*	*SNWHO*	*TSS*	*JWD*	*WPJD*	*WNWHO*	*WNWIH*	*TVS*	*N*	*SJD*	*SPJD*	*SNWIH*	*SNWHO*	*TSS*	*JWD*	*WPJD*	*WNWHO*	*WNWIH*	*TVS*
0	1	0	0	0	0	–1	8	0	0	0	19	20	8	0	0	0	20	15	2	0	1
1	–1	2	1	1	2	–1	9	0	0	0	20	21	9	0	1	0	21	18	2	1	1
2	3	1	0	0	0	–1	8	1	1	2	21	–1	11	1	2	1	–1	20	2	1	1
3	–1	3	1	1	2	–1	9	1	1	2	22	23	6	1	1	2	–1	13	1	0	0
4	5	8	0	0	0	–1	14	1	1	2	23	–1	18	0	0	0	–1	14	0	0	0
5	–1	9	0	1	0	–1	16	1	1	2	24	25	9	0	0	0	25	17	1	1	1
6	–1	10	0	0	0	–1	19	1	1	2	25	26	8	0	1	0	–1	25	0	0	0
7	8	8	0	0	0	–1	21	1	1	2	26	–1	11	1	2	1	–1	16	0	0	0
8	9	12	0	1	0	9	10	0	0	0	27	28	7	1	1	2	28	17	0	1	1
9	–1	11	0	2	1	10	13	0	0	0	28	–1	20	0	0	0	–1	26	0	0	0
10	11	4	1	1	2	11	15	0	0	0	29	30	21	1	0	1	–1	23	0	0	1
11	–1	13	0	0	0	12	18	0	0	0	30	–1	19	1	1	1	–1	19	0	0	0
12	13	8	0	0	0	–1	20	1	0	0	31	32	22	0	1	1	32	22	1	1	1
13	14	9	0	1	0	14	10	1	0	0	32	–1	17	0	0	1	–1	27	0	0	0
14	–1	11	0	2	1	15	15	1	0	0	33	–1	23	0	0	1	–1	21	0	0	0
15	16	5	1	1	2	16	18	1	0	0	34	–1	14	0	0	0	35	22	0	1	1
16	–1	15	0	0	0	–1	20	0	0	0	35	–1	16	0	0	0	–1	28	0	0	0
17	18	16	1	0	1	–1	11	0	0	0	36	–1	19	0	0	0	–1	23	1	0	1
18	–1	14	1	1	1	19	13	2	0	1	37	–1	21	0	0	0	–1	24	0	0	1

На Рис. 3.10 показаний граф вихідної програми задачі, текст якої представлено на Рис. 3.7.

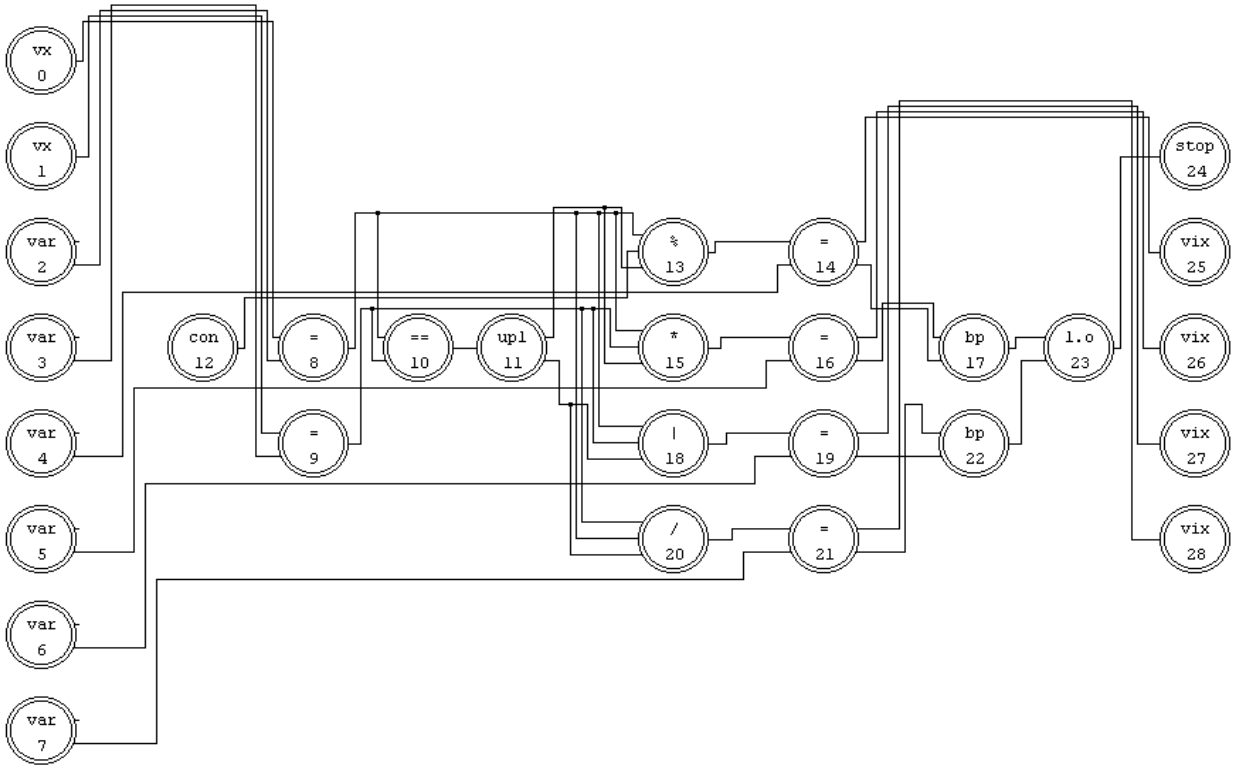


Рис. 3.10. Граф вихідної розгалуженої послідовної програми.

Неважко перевірити, що структури *DBF_CNW, DCF_CNW* (табл. 3.5 та 3.6) семантико-числової специфікації результат декомпіляції Сі-тексту часової паралельно-конвеєрної моделі завдання мають ідентичний Граф. Фактична автоматична перевірка коректності здійснюється шляхом перевірки збігу структур СЧС *BF, CF* вихідної програми і структур СЧС *DBF_CNW, DCF_CNW,* що представляють результат декомпіляції паралельно-конвеєрної моделі.

Для зручності перевірки коректності людиною можуть виводитися текстові специфікації декомпільованої моделі та вихідної програми (Рис. 3.11).

```
#include <stdio.h>
void main(void)
{
 int a,b, k,z,p,s;
  scanf("%d %d %d %d",&a,&b);
  if(a == b)
  {
   k = a % 2;
   z = a * b;
   printf("%4d\n",k);
   printf("%4d\n",z);
  }
  else
  {
   p = a | b;
   s = b / a ;
  }
   printf("%4d\n",p);
  printf("%4d\n",s);
}
```

а) Текст редукції специфікації паралельно-конвеєрної моделі.

```
#include <stdio.h>
void main(void)
{
int a,b,k,z,p,s;
   scanf("%d %d %d %d",&a,&b);
   if(a == b)
   {
    k = a % 2;
    z = a * b;
    printf("%4d\n",k);
  printf("%4d\n",z);
 }
   else
   {
    p = a | b;
    s = b / a ;
   }
    printf("%4d\n",p);
 printf("%4d\n",s);
}
```

б) Оригінальний текст Сі-програми

Рис. 3.11. Текст редукції специфікації паралельно-конвеєрної моделі та оригінальний текст послідовної програми.

Відзначено, що з метою досягнення наочності викладався зміст етапів декомпіляційної верифікації та пояснювалося за допомогою текстових і графічних специфікацій об’єктів і дій над ними. Фактично формальна декомпіляціонная верифікація оперує зі структурами семантико-числової специфікації синтезованих програмних об’єктів і зводиться до виконання множини операцій над просторово-часовими структурами СЧС [58].

Висновок до розділу 3.

В розділі описано сутність декомпіляційної верифікації часопараметризованих програмних засобів інформаційних управляючих систем. Починаючи з аналізу бінарного коду послідовної програми, проводиться його декомпіляція з метою відновлення вхідного коду.

Отриманий вхідний код піддається формальній верифікації за допомогою математичних методів. Відбувається доведення коректності програми, перевірка властивостей безпеки, аналіз відповідності вимогам тощо. Аналізуючи вхідний код програми, можна виявити можливі вразливі місця, дефекти або помилки, що можуть впливати на її безпеку та надійність.

Описано етапи постановки задачі декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС. Зокрема метод декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС. Представлено структурну схему модифікованого методу декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС та детальний опис процедур, які реалізують етапи методу.

Наведено приклади застосування методу декомпіляційної верифікації часопараметризованих паралельних програмних засобів ІУС. Задля цього семантику основних етапів декомпіляційної верифікації паралельних програмних засобів ІУС прояснено на прикладі паралельно-конвеєрної моделі задачі, яка синтезована з використанням методу суміщення незалежних операцій і методу конвеєрної обробки інформації. З метою досягнення наочності зміст етапів декомпіляційної верифікації викладався та пояснювалося за допомогою текстових і графічних специфікацій об’єктів і дій над ними. Зазначено, що додатковою перевагою методу є здатність перевіряти відповідність отриманого декомпільованого вихідного коду формальній специфікації та вимогам. Це дозволяє переконатися, що програма відповідає поставленим завданням та функціональним вимогам. Отже, метод декомпіляційної верифікації є потужним інструментом для забезпечення коректності, безпеки та надійності програмного забезпечення. Його застосування може значно підвищити рівень довіри до програм та систем, особливо в умовах зростаючої складності та критичності інформаційних технологій.

Основні положення цього розділу викладені у публікаціях автора [1, 2, 9, 23, 24, 28].

РОЗДІЛ 4.
РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СЕМАНТИКО-ЧИСЛОВОЇ ВЕРИФІКАЦІЇ ЧАСОПАРАМЕТРИЗОВАНИХ МУЛЬТИПАРАЛЕЛЬНИХ ПРОГРАМ ІУС.

Технологію верифікації визначимо як сукупність формальних методів верифікації та програмних засобів реалізації [13, 60].

Технології верифікації є важливим компонентом в області формальної верифікації програмного та апаратного забезпечення. Ці технології використовуються для перевірки правильності функціонування систем та компонентів, особливо в контексті високоризикових індустрій, де навіть найменша помилка може призвести до серйозних наслідків. Основні аспекти технологій верифікації включають:

Математичні моделі: верифікація базується на математичних моделях, які визначають поведінку системи або компонента. Ці моделі включають в себе інформацію про структуру та функціональність системи.

Формальні методи: технології верифікації використовують формальні методи, такі як математична логіка та теорія автоматів, для аналізу системи. Це дозволяє здійснювати точну перевірку системи на виконання певних властивостей.

Автоматизація: сучасні інструменти та платформи для верифікації надають автоматизовані засоби для аналізу системи. Це дозволяє розробникам швидше та ефективніше виявляти та виправляти помилки.

Підтримка високого рівня абстракції: технології верифікації можуть працювати на різних рівнях абстракції, від високорівневих моделей до деталей реалізації. Це дозволяє аналізувати систему на різних стадіях розробки.

Перевірка властивостей: технології верифікації дозволяють перевіряти різні властивості системи, такі як коректність, відсутність гонок даних, безпеку та інші. Це допомагає забезпечити правильність та надійність системи.

Покращення якості програмного та апаратного забезпечення: верифікація допомагає зменшити кількість програмних та апаратних помилок, що може призвести до покращення якості продукту та зниження витрат на підтримку.

Загалом, технології верифікації є надзвичайно важливим інструментом для забезпечення правильності та надійності інформаційних управляючих систем у різних галузях, включаючи авіацію, медицину, автомобільну промисловість, фінанси та інші. Вони дозволяють виявляти та виправляти проблеми на ранніх стадіях розробки, що зменшує ризики та витрати на виправлення помилок у майбутньому.

Все сказане в повній мірі стосується технології семантико-числової верифікації часопараметризованих мультипаралельних програм ІУС.

Ще однією складовою верифікації часопараметризованих мультипаралельних програм ІУС, окрім компіляційної та декомпіляційної, є семантична верифікація [13, 60].

4.1 Метод семантичної верифікації мультипаралельних часопараметризованих програм ІУС

Основою семантичної верифікації є використання одиниць вимірювання вхідних даних і вихідних результатів задачі, що задаються користувачем, автоматичне визначення в процесі вирішення задач одиниць вимірювання проміжних і вихідних результатів та їх порівняння з одиницями вимірювання вихідних результатів, заданими користувачем.

В якості вхідних даних для семантичної верифікації є:

- вхідна послідовна програма задачі;
- текст часової мультипаралельної програми, що відповідає вхідній послідовній програмі та відповідає вимогам або обмеженням користувача;
- семантична база даних БД «*SEM*» одиниць виміру фізичних величин;
- семантика (одиниці виміру) вхідних даних та вихідних послідовної

програми, що задаються користувачем – структура *US_SEM.*

В ході виконання семантичної верифікації необхідно:

- перевірити семантичну коректність вхідної послідовної програми задачі;
- перевірити семантичну коректність статичної мультипаралельної програми.
- перевірити логічну еквівалентність часової мультипаралельної програми та вхідної послідовної програми задачі.

Вихідними даними семантичної верифікації є:

1. Семантико-числова специфікація вхідного коду програми (список одиниць виміру вхідних даних задачі, значень розмірності результатів проміжних обчислень та значень розмірності результатів виконання послідовної програми).

2. Результати перевірки ідентичності значень розмірності результатів виконання послідовної програми та користувальницької семантичної специфікації задачі.

3. Результати перевірки семантичної коректності часової моделі та тексту часпараметризованої мультипаралельної програми.

Основні етапи методу семантичної верифікації мультипаралельних програм показані на Рис. 4.1. Розглянемо зміст основних етапів методу семантичної верифікації [13, 60, 61, 68–69].

Етап 1 (*символ 2 Рис. 4.1*) забезпечує синтез для послідовної програми структур *BF і CF* семантико-числової специфікації.

На етапі 2 (*символ 3 Рис. 4. 1*) забезпечується графічна візуалізація початкової послідовної програми у вигляді графа, виходячи із структур СЧС *BF,* сформованих при виконанні першого етапу. Побудова графа здійснюється за допомогою засобів візуалізації паралельних апаратно-програмних об'єктів, описаних в [58].

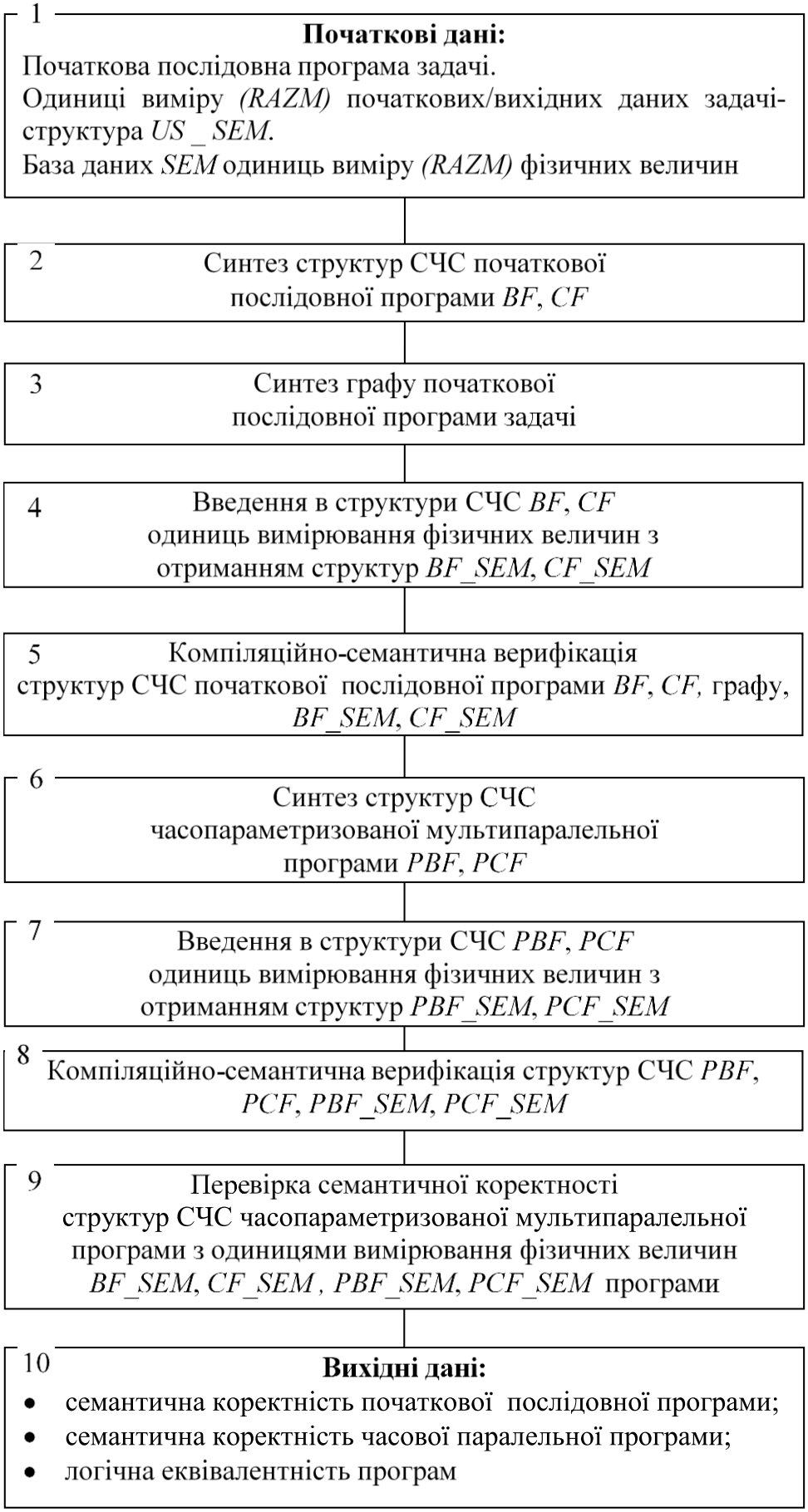


Рис. 4.1. Основні етапи методу семантичної верифікації мультипаралельних програм.

Етап 3 (*символ 4 Рис. 4.1*) забезпечує синтез одиниць виміру даних, що формуються "внутрішніми" і "вихідними" операторами P_j структур СЧС і графа, виходячи із заданих користувачем одиниць виміру початкових і вихідних даних послідовної програми задачі, типів операцій, що виконуються операторами P_j, послідовної програми, і загальноприйнятої бази даних одиниць виміру фізичних величин.

Етап 4 (*символ 5 Рис. 4. 1*) виконує шляхом компіляційно-семантичної верифікації перевірку синтезу структур СЧС послідовної програми і відповідного графа. Методика компіляційної верифікації розглянута в розділі 2. А також забезпечує перевірку семантичної коректності структур СЧС *BF_SEM, CF_SEM* початкової послідовної програми шляхом порівняння розрахованих одиниць вимірювання даних, що відповідають вихідним операторам синтезованої структури BF_SEM початкової послідовної програми, з одиницями вимірювання вихідних даних, що задаються користувачем.

Семантична верифікація часопараметризованих паралельних програм виконується символами 6–9 (Рис. 4.1).

Синтез структур *PBF, PCF* СЧС часопараметризованої мультипаралельної програми виконується на етапі 5 (*символ 6 Рис. 4.1*) аналогічно етапу 1 (символ 2 Рис. 4.1).

На етапі 6 (*символ 7 Рис. 4.1*) здійснюється введення в структури СЧС *PBF, PCF* одиниць виміру фізичних величин з отриманням структур *PBF _ SEM, PCF _ SEM.*

Етап 8 (*символ 8 Рис. 4. 1*). Верифікація компіляції структур СЧС *PBF, PCF* з одночасною перевіркою семантичної коректності розширених структур СЧС *PBF _ SEM, PCF _ SEM* часопараметризованої мультипаралельної програми шляхом порівняння розрахованих одиниць виміру даних, що відповідають вихідним операторам синтезованої структури *PBF _ SEM,* з одиницями виміру вихідних даних, заданими користувачем.

Етап 9 (*символ 9 Рис. 4.1*) здійснює перевірку семантичної коректності структур СЧС *BF_SEM, CF_SEM, PBF_SEM, PCF_SEM* та логічної еквівалентності результатів їх декомпіляції.

4.2. Технологія верифікації паралельних часопараметризованих програм інформаційних і управляючих систем

Новизна класу часопараметризованих мультипаралельних програм робить необхідним рішення науково-прикладної задачі створення інформаційної технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем, яка забезпечує перевірку синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації часопараметризованих мультипаралельних програм в динаміці їх проєктування з одночасною перевіркою збігу одиниць вимірювання фізичних величин, отриманих та одиниць вимірювання вхідних та вихідних даних задачі, що задаються користувачами [4–6,13, 58, 60–62, 70-71, 78–60]

На Рис. 4.2 представлено концептуальну модель технології верифікації часопараметризованих паралельних програм ІУС

Вхідними даними, які використовуються технологією для верифікації паралельних програмних засобів є:

– тексти послідовних програм завдань ІУС, для яких синтезуються часові паралельні програми;

– склад типів даних, операторів або функцій мови програмування високого рівня (наприклад, Сі) та процедур обміну повідомленнями;

– бібліотека тривалості виконання (у процесорних тактах) операторів або функцій процесорами різних типів (наприклад, суперскалярного типу).

До основних компонентів архітектури технології верифікації часопараметризованих паралельних програм відносяться наступні.

Зберігання одиниць вимірювання фізичних величин, які використовуються для перевірки семантичної коректності формального синтезу програмних засобів паралельних обчислювальних систем зберігається в базі даних «*Sem*» (*символ 0, Рис. 4.*2).

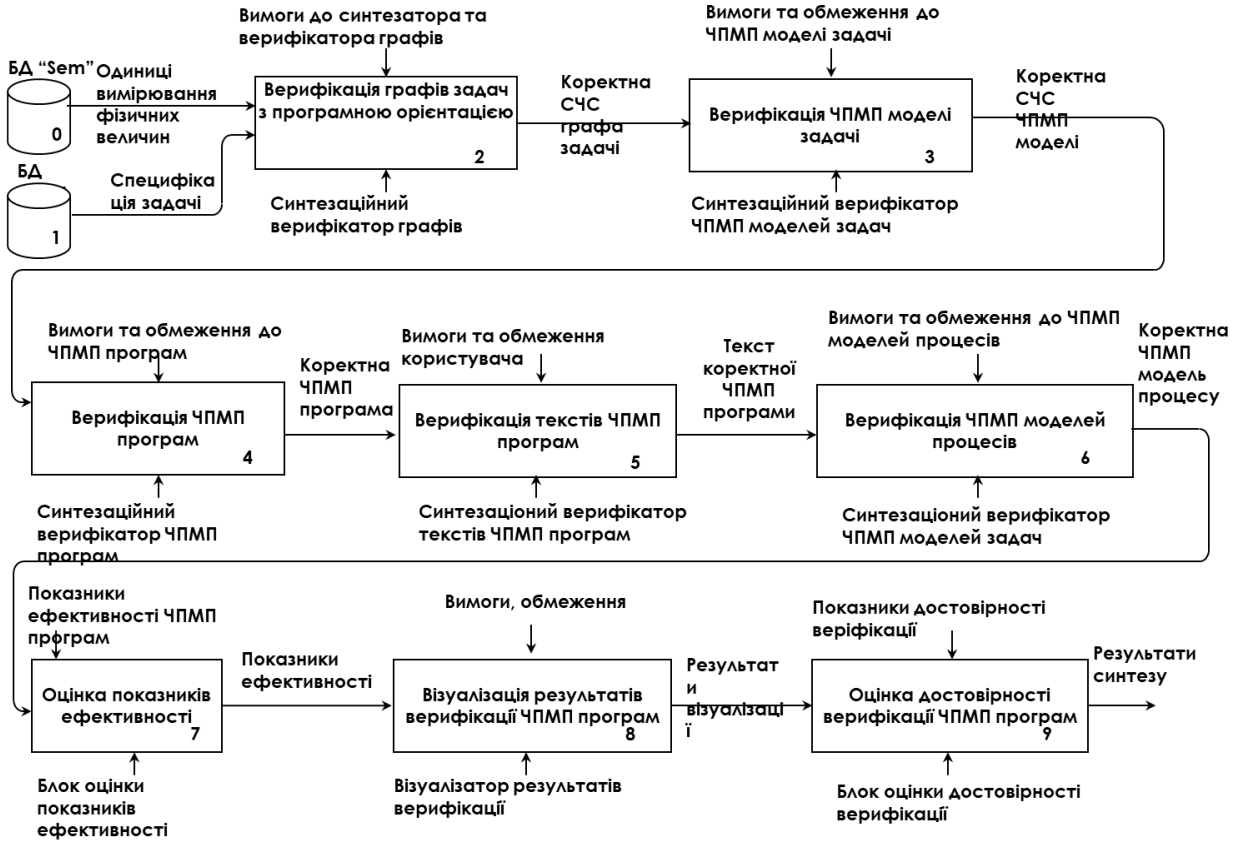


Рис. 4.2 Концептуальна модель технології верифікації мультипаралельних часопараметризованих програм ІУС.

Синтезаційний верифікатор графів, представлений символом 2 (Рис. 4.2), здійснює синтез семантико-числової специфікації графів задач та їх верифікацію. Забезпечується проведення компіляційно-семантичної (*CSV*) та декомпіляційно-семантичної (*DSV*) верифікації графів вихідних програм задач та структур їх семантико-числової специфікації.

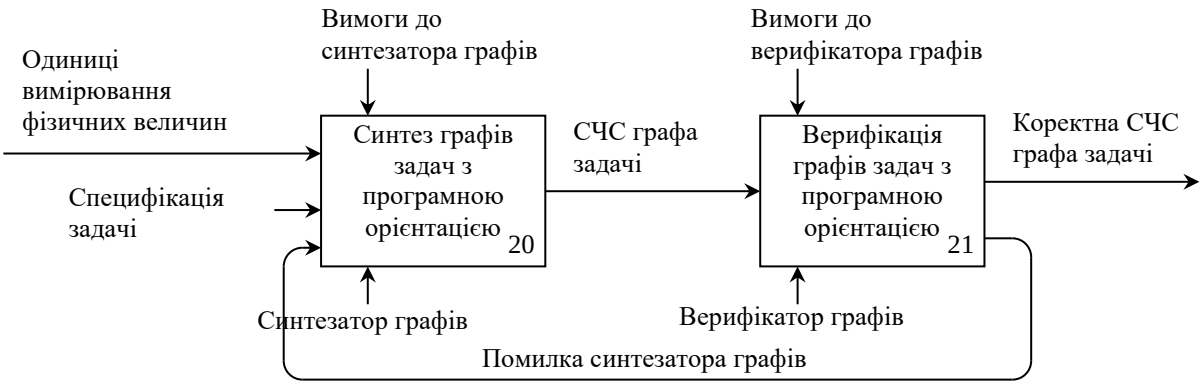


Рис. 4.3. Синтезаційний верифікатор графів.

В більш деталізованому вигляді структура синтезаційного верифікатору графів представлена на Рис. 4.3.

Результатами роботи синтезаційного верифікатору графів є:

- коректна, с точки зору верифікації, семантико-числова специфікація (СЧС) вихідного тексту послідовної програми;

- графічна специфікація вихідної послідовної програми.

Синтезаційний верифікатор ЧПМП моделей задач (символ 3, Рис. 4.2) здійснює формальний синтез часових мультипаралельних моделей вихідних послідовних програм, синтезованих з урахуванням характеристик архітектури паралельних обчислювальних систем (ОС) і заданих вимог, та їх верифікацію. Структура синтезаційного верифікатора ЧПМП моделей задач представлена на Рис. 4.4. Верифікатор забезпечує компіляційно-семантичну (CSV) та декомпіляційно-семантичну (DSV) верифікацію часових мультипаралельних моделей вихідних послідовних програм, орієнтованих на конкретні класи ОС та задані вимоги.

Результатом роботи синтезаційного верифікатора ЧПМП моделей задач є коректна семантико-числова специфікація часопараметризованої архітектурно-орієнтованої моделі вихідної послідовної програми

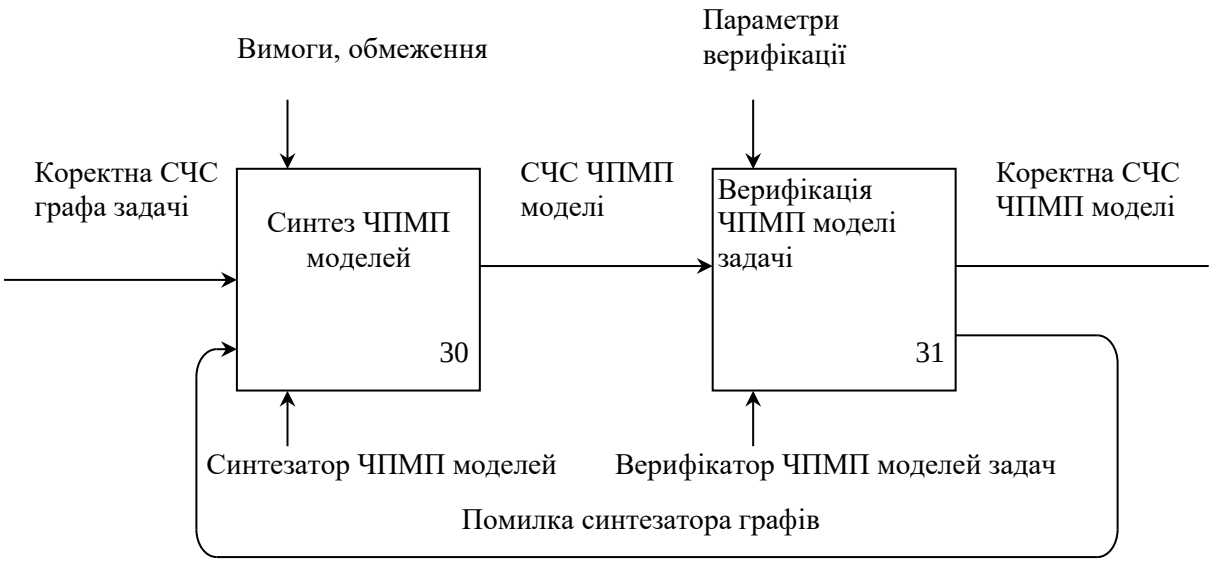


Рис. 4.4. Синтезаційний верифікатор ЧПМП моделей задач.

Верифіковані тексти часопараметризованих мультипаралельних програм та часові моделі їх виконання з урахуванням вимог та обмежень, які задаються користувачами, автоматично синтезуються верифікатором текстів ЧПМП програм (символ 5, Рис. 4.2). Даний етап також забезпечує декомпіляційно-семантичну верифікацію текстів синтезованих часових мультипаралельних програм (з урахуванням типів та розмірностей даних, складу обчислювальних та керуючих операторів/функцій, засобів обміну даними та операторів часової синхронізації процесів).

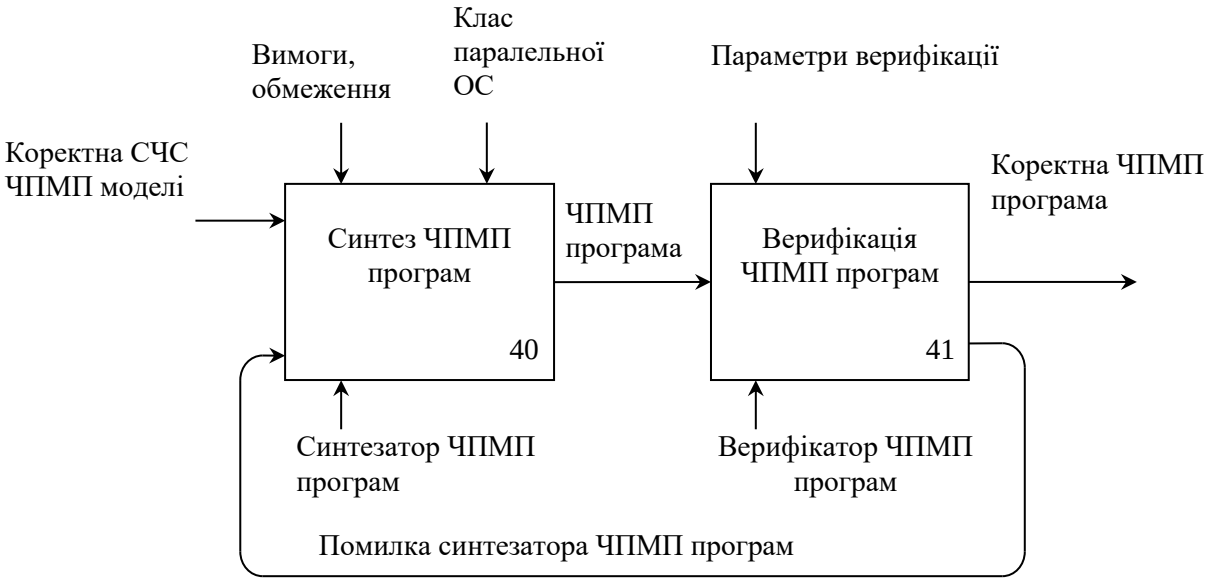


Рис. 4.5. Синтезаційний верифікатор ЧПМП програм.

Структуру синтезаційного верифікатора текстів ЧПМП програм представлено на Рис. 4.6.

Синтезатор моделей процесів виконання ЧПМП програм (символ 6, Рис. 4.2) забезпечує синтез множини моделей процесів, що задовольняють вимогам та обмеженням замовника, а також компіляційно-семантичну (CSV) декомпіляційно-семантичну (DSV) верифікацію часових моделей. На Рис. 4.7 представлено структуру синтезаційного верифікатора ЧПМП моделей процесів.

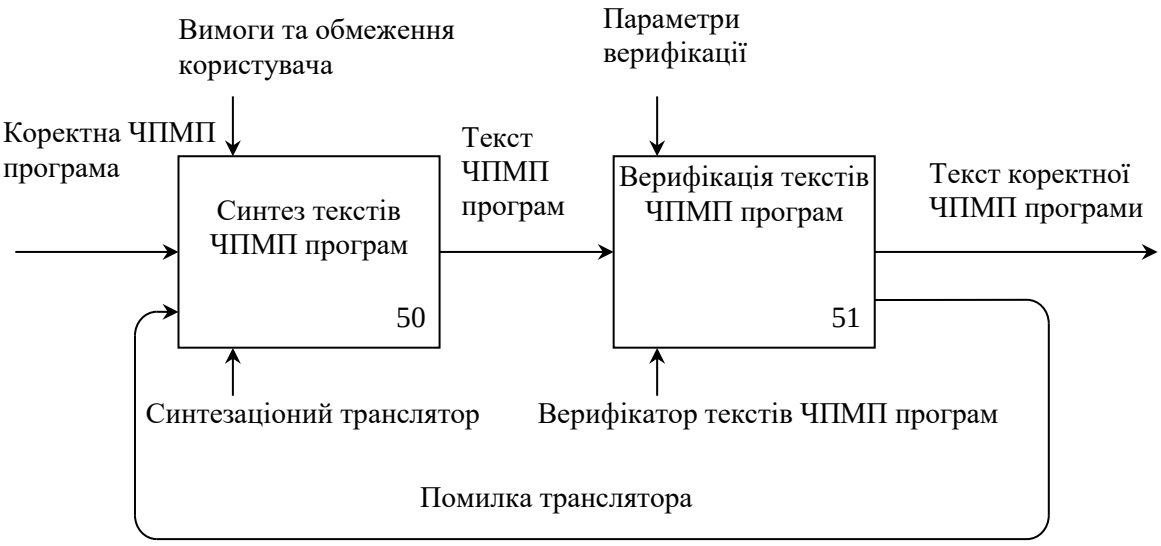


Рис. 4.6. Синтезаційний верифікатор текстів ЧПМП програм.

Одним з важливіших етапів розробки паралельних програм є оцінка показників їх ефективності. Блок оцінки показників ефективності ЧПМП програм здійснює розрахунок показників ефективності паралельних алгоритмів та програм: часу паралельної реалізації програми (Tpar), прискорення (Sp), ефективності (Ef) залежно від кількості паралельних процесорних елементів (символ 7, Рис. 4.2).

Візуалізатор результатів верифікації (символ 8, Рис. 4.2) забезпечує зручну для користувачів наочну форму представлення вихідних результатів технології верифікації ЧПМП програм.

Оцінка достовірності верифікації ЧПМП програм є останнім етапом розроблюваної технології. Він забезпечує генерацію операторів – «дефектів» – зовнішніх впливів різних типів, що спотворюють відповідні статичні і часові елементи семантико-числових і графічних специфікацій об’єктів, що верифікуються, і виконує оцінку достовірності, як відношення кількості виявлених дефектів до кількості дефектів, фактично введених у паралельну програму (символ 9, Рис. 4.2).

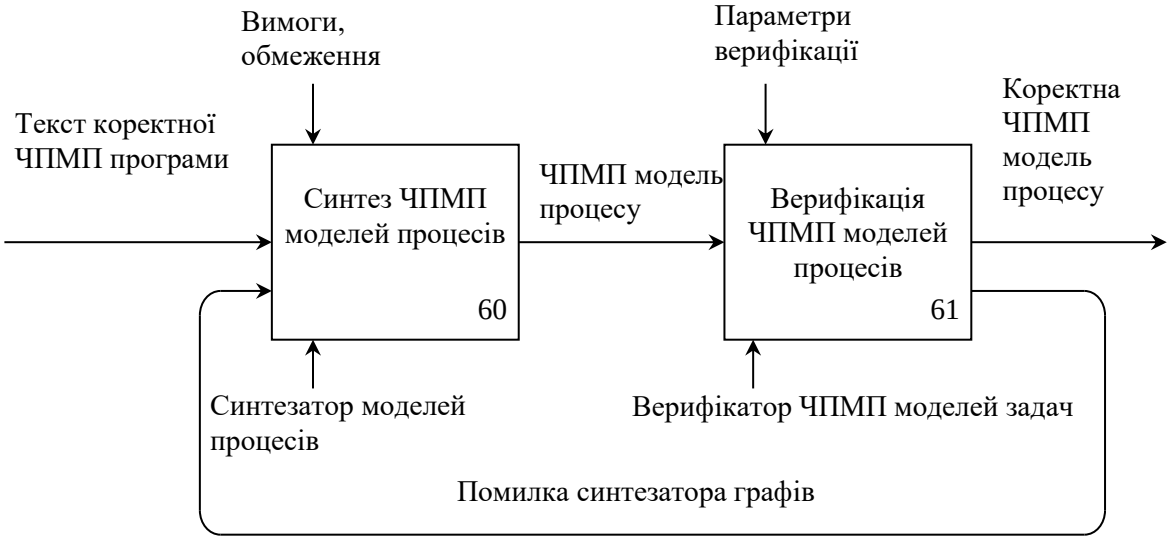


Рис. 4.7. Синтезаційний верифікатор ЧПМП моделей процесів.

4.3 Показники технологій верифікації ІУС

Показники технологій верифікації ІУС визначають ефективність і надійність процесу верифікації [58–59, 63–67]. Приведемо кілька ключових показників, які можна використовувати для оцінки технологій верифікації ІУС:

– Покриття коду (*Code Coverage*): Цей показник вказує на те, який відсоток програмного коду був охоплений верифікацією. Важливо мати високе покриття коду, оскільки це допомагає виявити більше можливих дефектів. Наприклад, якщо покриття коду становить 95%, це означає, що 95% коду було перевірено. Це може бути особливо важливим для вимогливих до

якості проєктів. Приклад: Верифікація покриття всіх рядків коду інтерфейсу користувача в програмі для банківського додатку.

– Складність програми (*Cyclomatic Complexity*): Ця метрика визначає складність програми на основі кількості шляхів керування в коді. Низька складність може спрощувати верифікацію.

– Час виявлення дефектів (*Time to Defect Detection*): Показує, скільки часу треба, щоб виявити дефекти в процесі верифікації. Чим швидше дефекти виявляються, тим менше вони коштують в плані виправлення. Приклад: Виявлення помилок в коді під час ранніх етапів розробки проти виявлення їх під час тестування перед випуском продукту.

– Кількість виявлених дефектів (*Number of Defects Found*): Показує загальну кількість виявлених дефектів під час верифікації. Це може слугувати метрикою для порівняння ефективності різних технологій верифікації. Приклад: Виявлення 50 дефектів під час верифікації програмного забезпечення.

– Показник ложних позитивів (*False Positives*): Показує, скільки з виявлених дефектів виявилися помилковими і не потребують корекції. Велика кількість ложних позитивів може затяжити процес верифікації. Приклад: Виявлення 20 ложних позитивів під час аналізу коду.

– Метрика виявлення критичних дефектів (*Critical Defect Detection Rate*): Показує, як ефективно технологія верифікації виявляє критичні дефекти, які можуть призвести до серйозних проблем у роботі системи. Приклад: Виявлення 90% критичних дефектів під час верифікації.

– Покращення продуктивності розробника (*Developer Productivity Improvement*): Показує, наскільки швидко розробники можуть виявляти і виправляти дефекти завдяки технології верифікації. Приклад: Зменшення часу, витраченого розробником на виправлення дефектів, з 4 годин до 1 години завдяки використанню автоматизованого інструменту верифікації.

– Зменшення витрат на підтримку (*Support Cost Reduction*): Показує, як використання технології верифікації може вплинути на витрати на підтримку

інформаційної управляючої системи після випуску в експлуатацію. Приклад: Зменшення кількості скарг і виправлень програми після випуску завдяки ефективній верифікації.

– Масштабованість (*Scalability*): Масштабованість технології верифікації оцінює, наскільки добре вона працює на великих та складних проєктах. Технологія повинна бути здатною адаптуватися до зростаючої складності програми.

– Точність (*Accuracy):* Цей показник вказує на точність результатів верифікації. Технологія повинна бути надійною і не призводити до надмірних виявлених помилок (*false positives*) або пропущених дефектів (*false negatives*).

Ці показники допомагають оцінити, наскільки ефективно та надійно технологія верифікації працює на конкретному проєкті і чи відповідає вона потребам розробки інформаційної управляючої системи.

Загальна ефективність верифікації ІУС може бути визначена враховуючи всі ці показники. Успішна верифікація веде до зменшення ризику помилок, а також зменшення витрат на подальшу розробку та обслуговування.

Показники ефективності технології семантико-числової верифікації інформаційних управляючих систем (ІУС) спрямовані на оцінку точності та надійності результатів верифікації [72–76, 78–88]. В якості *показників ефективності технології семантико-числової верифікації ІУС*, як найбільше впливаючи на її ефективність, обрані наступні:

1. Час виконання верифікації (T_{ver}): час, який потрібно для проведення семантико-числової верифікації. Швидкість верифікації може бути критичною для великих і складних систем.

2. Використання ресурсів (V): кількість обчислювальних та ресурсів пам’яті, які потрібні для проведення верифікації. Це може включати обсяг пам'яті, обчислювальну потужність та інші ресурси.

3. Кількість виявлених дефектів (N_{df}): загальна кількість виявлених дефектів за допомогою технології семантико-числової верифікації.

Ці показники використані для оцінки ефективності технології семантико-числової верифікації ІУС та допомагають приймати рішення щодо її використання в конкретному проєкті [77–101].

4.4 Приклад роботи інформаційної технології верифікації мультипаралельних часопараметризованих програм ІУС.

Проілюструємо результати роботи інформаційної технології верифікації ЧПМП програм ІУС на прикладі. На Рис. 4.8 представлена вихідна послідовна програма написана мовою високого рівня (послідовна програма).

До складу початкових даних входять також фрагмент бази даних одиниць виміру фізичних величин (табл. 4. 1) та перелік даних, що задаються користувачем (табл. 4.2).

Таблиця 4.1.

Фрагмент семантичної бази даних «*SEM*».

KOD_RAZM	*RAZM*	*KOD_RAZM*	*SEM*
2	м	14	рад/с
3	кг	15	м/(с*с)
4	с	16	рад/(с*с)
5	А	17	1/м
6	К	18	кг/(м*м*м)
7	моль	19	м*м*м/кг
8	кд	20	А/(м*м)
9	рад	21	А/м
10	ср	22	моль/(м*м*м)
11	м*м	23	1/с
12	м*м*м	24	м*м/с
13	м/с	25	кд/(м*м)

Таблиця 4.2.

Структура *US_SEM* одиниць вимірювання, що задаються користувачем.

N_OP	*REZ*	*VHOD_VIH*	*KOD_SEM*
3	a	0	2
4	b	0	2
5	c	0	2
11	s	1	2
12	t	1	11

```
#include <stdio.h>
void main(void)
{
int a,b,c,r, k,l,m,p, s,t;
        scanf("%d %d
%d\n",&a,&b,&c);
        k = a * b;
        l = b % a;
        if(k < a-c)
        { m = (k % 2) * 2;
         r = l * 2;
        p = k + l;
        }
        else
        { p = 2 * l;
         r = l - k;
         m = p + l;
        }
        s = p - r;
        t = (m * 2) / a;
        printf ("%4d %4d\n",s,t);
}
```

Рис. 4.8. Вихідна послідовна програма (мовою Сі).

Базова структура *BF* (табл. 4.3) описує номери та склад операторів P_j задачі (масив *N*), їх типи (масив *TYP*), число вхідних (масив *SJD*) та вихідних (масив *WJD*) зв'язків кожного оператора P_j, ідентифікатори операторів (масив *RES*), покажчики початку ланцюжків сполучених і зовнішніх операторів (масиви *NSJ, NWJ*) для кожного оператора послідовної програми. Синтез структур *BF* і *CF* СЧС здійснюється відповідно до методики, викладеної в [58]. Виходячи зі структур СЧС *BF* і *CF*, здійснюється графічна візуалізація вихідної послідовної програми у вигляді графа (Рис. 4.9). Побудова графа здійснюється за допомогою засобів автоматичної візуалізації паралельних апаратно-програмних об'єктів, описаних у [59].

Таблиця 4.3.

Базова структура *BF_* операторів початкової послідовної програми.

N	***MET***	***TYP***	***NSJ***	***SJD***	***BJ***	***NWJ***	***WJD***	***MP1***	***MP2***	***VH***	***VIH***	***RES***	***N***	***MET***	***TYP***	***NSJ***	***SJD***	***BJ***	***NWJ***	***WJD***	***MP1***	***MP2***	***VH***	***VIH***	***RES***
0	0	58	–1	0	0	0	1	0	0	0	1	*a_in*	26	0	12	24	2	1	48	2	0	0	2	2	=
1	0	58	–1	0	0	1	1	0	0	0	1	*b_in*	27	0	3	26	3	1	50	1	0	0	3	1	*
2	0	58	–1	0	0	2	1	0	0	0	1	*c_in*	28	0	12	29	2	1	51	2	0	0	2	2	=
3	0	47	–1	0	0	3	1	0	0	0	2	*a*	29	0	1	31	3	1	53	1	0	0	3	1	+
4	0	47	–1	0	0	4	1	0	0	0	2	*b*	30	0	12	34	2	1	54	2	0	0	2	2	=
5	0	47	–1	0	0	5	1	0	0	0	2	*c*	31	0	50	36	3	1	56	1	3	0	3	1	*bp*
6	0	47	–1	0	0	6	2	0	0	0	2	*r*	32	2	3	39	3	2	57	1	0	0	3	1	*
7	0	47	–1	0	0	8	1	0	0	0	2	*k*	33	0	12	42	2	2	58	2	0	0	2	1	=
8	0	47	–1	0	0	9	1	0	0	0	2	*l*	34	0	2	44	3	2	60	1	0	0	3	1	–
9	0	47	–1	0	0	10	2	0	0	0	2	*m*	35	0	12	47	2	2	61	2	0	0	2	2	=
10	0	47	–1	0	0	12	2	0	0	0	2	*P*	36	0	1	49	2	2	63	1	0	0	2	1	+
11	0	47	–1	0	0	14	1	0	0	0	2	*s*	37	0	12	51	2	2	64	2	0	0	2	2	=
12	0	47	–1	0	0	15	1	0	0	0	2	*t*	38	0	50	53	2	2	66	1	3	0	2	1	*bp*
13	0	12	0	2	0	16	4	0	0	2	1	=	39	3	54	55	2	3	67	1	0	0	2	1	*l.o*
14	0	12	2	2	0	20	2	0	0	2	1	=	40	0	53	57	2	3	68	1	0	0	2	1	*a.o*
15	0	12	4	2	0	22	1	0	0	2	1	=	41	0	53	59	2	3	69	1	0	0	2	1	*a.o*
16	0	3	6	2	0	23	1	0	0	2	1	*	42	e	53	61	2	3	70	1	0	0	2	1	*a.o*
17	0	12	8	2	0	24	4	0	0	2	1	=	43	e	2	63	3	3	71	1	0	0	3	1	–
18	0	5	10	2	0	28	1	0	0	2	1	%	44	e	12	66	2	3	72	2	0	0	2	2	=
19	0	12	12	2	0	29	5	0	0	2	1	=	45	e	3	68	2	3	74	1	0	0	2	1	*
20	0	2	14	2	0	34	1	0	0	2	1	–	46	e	4	70	2	3	75	1	0	0	2	1	/
21	0	25	16	2	0	35	1	0	0	2	1	<	47	e	12	72	2	3	76	2	0	0	2	2	=
22	0	51	18	1	0	36	5	1	2	1	2	*upl*	48	e	50	74	2	3	78	1	980	0	2	1	*bp*
23	0	57	–1	0	1	41	5	0	0	0	1	*C2_*	49	980	49	76	1	4	–1	0	0	0	1	0	*stop*
24	1	5	19	3	1	46	1	0	0	3	1	%	50	0	48	77	1	4	–1	0	0	0	1	0	*s out*
25	0	3	22	2	1	47	1	0	0	2	1	*	51	0	48	78	1	4	–1	0	0	0	1	0	*t_out*

На першому етапі (символ 2 Рис. 4.2) здійснюється автоматичний синтез структур *BF* і *CF* (табл. 3 представляє *BF*) семантико-числової специфікації для вихідної послідовної програми (Рис. 4.3), а також візуалізація отриманого графа (Рис. 4.9).

Синтез семантико-числової специфікації *BF_SEM* послідовної програми (символ 4, Рис. 4.2) показано у табл. 4.4. У таблиці прийнято такі позначення: масив *RAZM* – одиниці виміру вихідних даних та даних – результатів виконання операцій: «м» – метр, «ні» – відсутність обчисленого значення вихідної змінної, «*безрозмір.*» – безрозмірна величина, «м* м», «м*м*м» – синтезовані одиниці виміру похідних величин.

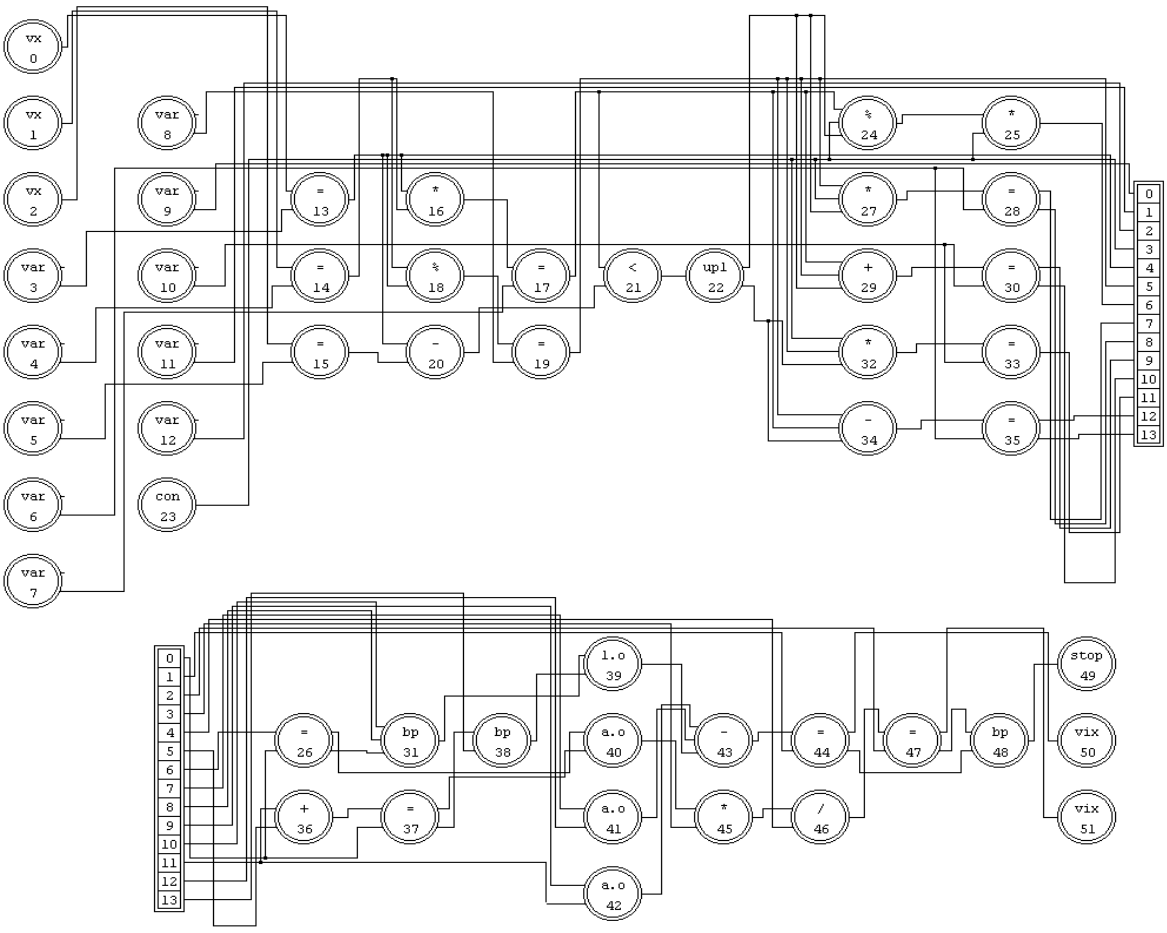


Рис. 4.9. Граф вихідної послідовної програми.

На Рис. 4.10 представлені результати другого етапу технології (символ 3 на Рис. 4.2).

```
Файл елементів:                    C:\My_prog\C\Result\CNSVWVIXVIX1.TXT
Файл зв'язків елементів:           C:\My_prog\C\Result\CNSVWVIXVIX2.TXT

 ТЕСТ КОРЕКТНОСТІ ФАЙЛІВ:
 максимальна кількість елементів:  0-35
максимальна кількість зв'язків     0-38

 ТЕСТ ВІДПОВІДНОСТІ ЧИСЛА, СПОЛУЧЕНИХ І ЗОВНІШНІХ ЗВ'ЯЗКІВ: ОК

 ТЕСТ ЧИСЛА ЗВ'ЯЗКІВ ПО СПОРУЖЕНИХ ЕЛЕМЕНТАХ: ОК

 ТЕСТ ЧИСЛА ЗВ'ЯЗКІВ ПО ЗОВНІШНІМ ЕЛЕМЕНТАМ: ОК

 ТЕСТ ВІДПОВІДНОСТІ ВИСНОВКІВ ПО СПОЛУЧЕНИХ ЕЛЕМЕНТАХ: ОК

 ТЕСТ ВІДПОВІДНОСТІ ВИСНОВКІВ ПО ЗОВНІШНІМ ЕЛЕМЕНТАМ: ОК

 ТЕСТ ВІДПОВІДНОСТІ ЧИСЛА ВХОДІВ ЕЛЕМЕНТУ І КІЛЬКОСТІ ЙОГО СПОЛУЧЕНИХ: ОК
```

Рис. 4.10. Результати компіляційної верифікації структур семантикочислової специфікації вхідної послідовної програми та її графа (*View of test results).*

Перевірка семантичної коректності структур СЧС *BF_SEM, CF_SEM* вихідної послідовної програми здійснюється шляхом порівняння розрахованих одиниць вимірювання даних, що відповідають вихідним операторам синтезованої структури *BF_SEM* вихідної програми з одиницями вимірювання вихідних даних, заданими користувачем.

Таблиця 4.4.

Структура *BF_SEM* – результат синтезу одиниць виміру операторів початкової послідовної програми.

N	***TYP***	***NSJ***	***SJD***	***BJ***	***NWJ***	***WJD***	***RES***	***SEM***	***N***	***TYP***	***NSJ***	***SJD***	***BJ***	***NWJ***	***WJD***	***RES***	***SEM***
0	58	–1	0	0	0	1	a_in	*м*	26	12	24	2	1	48	2	=	*м*м*
1	58	–1	0	0	1	1	b_in	*м*	27	3	26	3	1	50	1	*	*безрозмір.*
2	58	–1	0	0	2	1	c_in	*м*	28	12	29	2	1	51	2	=	*безрозмір.*
3	47	–1	0	0	3	1	a	*м*	29	1	31	3	1	53	1	+	*м*м*
4	47	–1	0	0	4	1	b	*м*	30	12	34	2	1	54	2	=	*м*м*
5	47	–1	0	0	5	1	c	*м*	31	50	36	3	1	56	1	bp	*безрозмір.*
6	47	–1	0	0	6	2	r	*безрозмір.*	32	3	39	3	2	57	1	*	*безрозмір.*
7	47	–1	0	0	8	1	k	*м*м*	33	12	42	2	2	58	2	=	*безрозмір.*
8	47	–1	0	0	9	1	1	*безрозмір.*	34	2	44	3	2	60	1	–	*м*м*
9	47	–1	0	0	10	2	m	*м*м*	35	12	47	2	2	61	2	=	*м*м*
10	47	–1	0	0	12	2	P	*м*м*	36	1	49	2	2	63	1	+	*безрозмір.*
11	47	–1	0	0	14	1	s	*м*	37	12	51	2	2	64	2	=	*безрозмір.*
12	47	–1	0	0	15	1	t	*м*м*	38	50	53	2	2	66	1	bp	*безрозмір.*
13	12	0	2	0	16	4	=	*м*	39	54	55	2	3	67	1	l.o	*безрозмір.*
14	12	2	2	0	20	2	=	*м*	40	53	57	2	3	68	1	a.o	*м*м*
15	12	4	2	0	22	1	=	*м*	41	53	59	2	3	69	1	a.o	*безрозмір.*
16	3	6	2	0	23	1	*	*м*м*	42	53	61	2	3	70	1	a.o	*м*м*
17	12	8	2	0	24	4	=	*м*м*	43	2	63	3	3	71	1	–	*м*м*
18	5	10	2	0	28	1	%	*безрозмір.*	44	12	66	2	3	72	2	=	*м*м*
19	12	12	2	0	29	5	=	*безрозмір.*	45	3	68	2	3	74	1	*	*м*м*
20	2	14	2	0	34	1	–	*м*	46	4	70	2	3	75	1	/	*м*
21	25	16	2	0	35	1	<	*безрозмір.*	47	12	72	2	3	76	2	=	*м*
22	51	18	1	0	36	5	upl	*безрозмір.*	48	50	74	2	3	78	1	bp	*безрозмір.*
23	57	–1	0	1	41	5	C2_	*безрозмір.*	49	49	76	1	4	–1	0	stop	*безрозмір.*
24	5	19	3	1	46	1	%	*м*м*	50	48	77	1	4	–1	0	s_out	*м*м*
25	3	22	2	1	47	1	*	*м*м*	51	48	78	1	4	–1	0	t_out	*м*

Перевірка семантичної коректності структур СЧС *BF_SEM, CF_SEM* вихідної послідовної програми здійснюється шляхом порівняння розрахованих одиниць вимірювання даних, що відповідають вихідним

операторам синтезованої структури *BF_SEM* вихідної програми з одиницями вимірювання вихідних даних, заданими користувачем [13].

Оцінка показників ефективності розробленої технології проводилась по цілому ряду практичних прикладних задач. До них можна віднести: метод Гауса для рішення систем лінійних рівнянь великої розмірності, алгоритм адаптивного згладжування і екстраполяції траєкторії, алгоритм цілерозподілу, то що.

В додатках Б та В представлено розроблений код програми синтезу паралельних часопараметризованих моделей та графічну візуалізацію паралельної часпараметризованої моделі алгоритму Гаусса.

Результати роботи технології семантико-числової верифікації ІУС при позитивному проходженні верифікації представлено в вигляді гістограми на Рис. 4.11.

Дослідження показників ефективності верифікації паралельної реалізації алгоритму Гауса було проведено на обчислювальній робочій станції R-LINE з настільним процесором Intel Core i-5 10400F S1200 BOX 2.9G BX807 (Intel Core i5-10400F, Сокет s-1200, DDR4). Використання 4-х модулей пам'яті DDR4 16GB Kingston KVR32N22D8/16 з частотою 3200 МГц дозволило провести дослідження показників ефективності верифікації паралельної реалізації алгоритму Гауса рішення системи лінійних рівнянь з 4 000 000 елементів. За рахунок використання 4-х ядер та технології семантико-числової верифікації часопараметризованих мультипаралельних програм ІУС було досягнуто мінімальний час верифікації рішення СЛАР методом Гауса для 4 000 000 елементів, який складає T_{ver}=37 хв.

Кількість виявлених дефектів N_{df} співпадає с кількістю засіяних дефектів (N_{df}=*100*%).

Обсяг пам’яті *V*, яка потрібна для проведення верифікації дорівнює 417 Кбайт, обчислювальна потужність – 4 ядра. Для порівняння, час, потрібний експерту на проведення верифікації рішення СЛАР методом Гауса склав порядку 5 годин. Кількість виявлених дефектів при цьому складала N_{df}=*78*%.

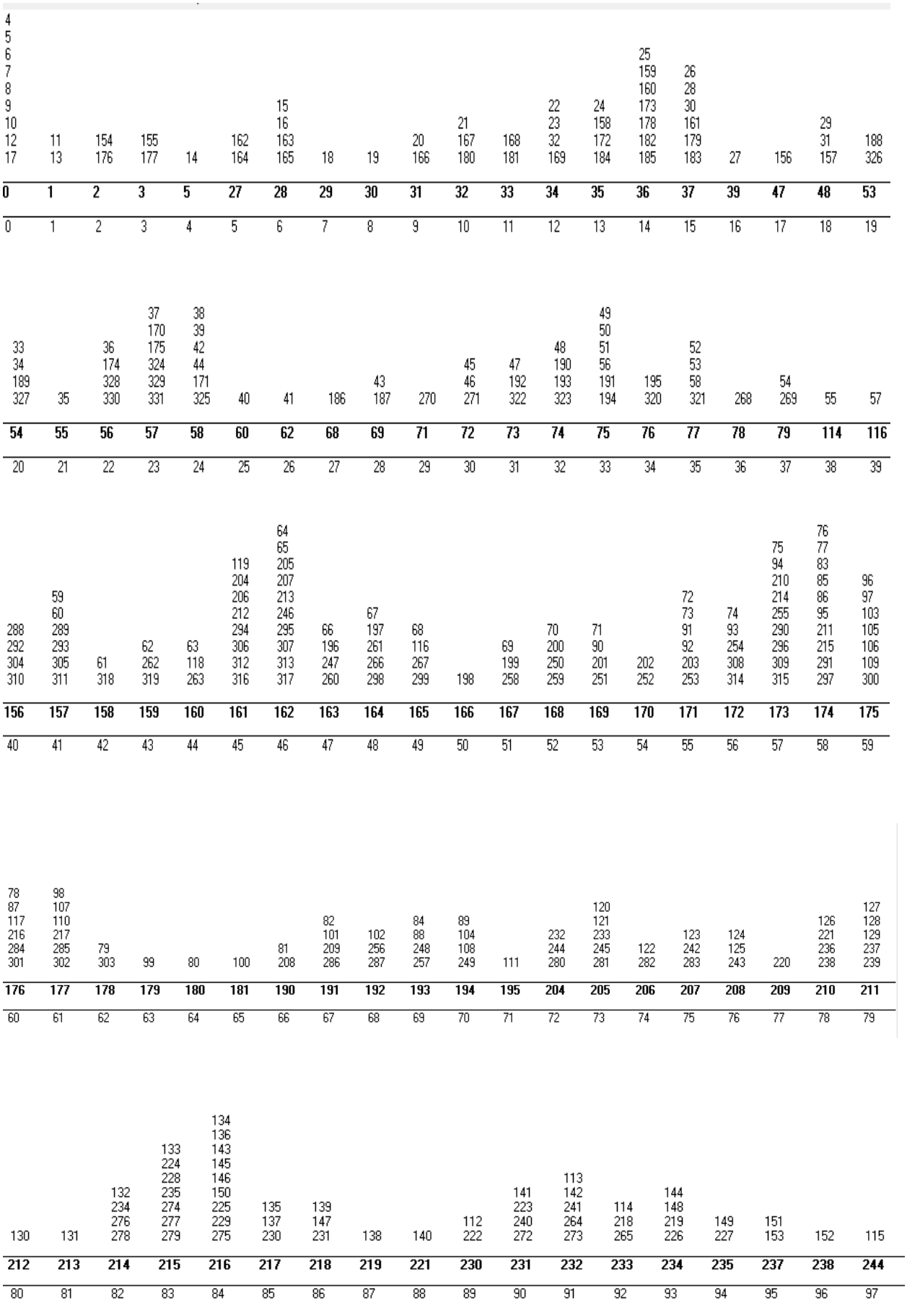


Рис. 4.11. Гістограмна візуалізація кластерної часової паралельної моделі алгоритму Гауса.

Дослідження показників ефективності верифікації паралельної реалізації алгоритму адаптивного згладжування і екстраполяції траєкторії показало, що він досить ефективно розпаралелюється при використанні методу поєднання незалежних операцій. Про це свідчить той факт, що при багатоядерній реалізації (обчислювальна потужність –5 ядер) час виконання алгоритму зменшується приблизно в 4.2 разу в порівнянні з послідовним виконанням, т. ч. практично пропорційно збільшенню числа ядер.

Подальше збільшення їх числа ($NM > 5$) виявляється практично недоцільним зважаючи на незначний приріст виграшу в часі. Кількість виявлених дефектів N_{df} співпадає с кількістю засіяних дефектів ($N_{df}=100\%$). Час необхідний на синтез часопараметризованої мультипаралельної програми алгоритму адаптивного згладжування і екстраполяції траєкторії по одній цілі та її верифікацію T_{ver}=12 хв. Час потрібний експерту на розробку семантико-числових специфікацій часопараметризованої мультипаралельної програми складає порядку однієї години та час на проведення верифікації склав порядку 1.5 год. Кількість виявлених дефектів $N_{df}=86\%$.

Висновки до розділу 4

У розділі описано метод семантичної верифікації часопараметризованих мультипаралельних програм, змістовно розглянуті основні етапи методу семантичної верифікації мультипаралельних програм та описано побудову графу, що здійснюється за допомогою засобів візуалізації паралельних апаратно-програмних об’єктів. Представлена інформаційна технологія верифікації часопараметризованих мультипаралельних програм ІУС та змістовно описано основні компоненти архітектури інформаційної технології верифікації. В більш деталізованому вигляді описана структура синтезаційного верифікатору графів та представлено структуру синтезаційного верифікатора текстів ЧПМП програм, синтезаційного верифікатора ЧПМП моделей процесів та саму структуру синтезаційного верифікатора текстів ЧПМП програм, що забезпечує декомпіляційно-

семантичну верифікацію текстів синтезованих часових мультипаралельних програм з урахуванням типів та розмірностей даних, складу обчислювальних та керуючих операторів, засобів обміну даними та операторів часової синхронізації процесів.

Обрано показники ефективності технології семантико-числової верифікації ІУС спрямовані на оцінку точності та надійності результатів верифікації. Оцінка показників ефективності розробленої технології проводилась по цілому ряду практичних прикладних задач: метод Гауса для рішення систем лінійних рівнянь великої розмірності, алгоритм адаптивного згладжування і екстраполяції траєкторії, алгоритм цілерозподілу, тощо.

Основні положення цього розділу викладені у публікаціях автора [7–20, 28].

ВИСНОВКИ

1. Останнім часом до інформаційних управляючих систем висуваються високі та суперечливі вимоги: з одного боку, це необхідність обробки великої кількості інформації у реальному часі, а з іншого – підвищення точнісних характеристик результатів обробки. Це робить необхідним дослідження можливостей і шляхів підвищення ефективності ІУС.

Показано, що до числа основних практично доцільних шляхів підвищення ефективності ІУС відносяться розпаралелювання алгоритмів опрацювання інформації та застосування для їхньої реалізації паралельного програмного забезпечення. Аналіз наявних технологій проєктування паралельних програм показав, що для розробки паралельного програмного забезпечення ІУС доцільно використовувати технологію розробки мультипаралельного програмного забезпечення на основі семантико-числових специфікацій. Використання часопараметризованих мультипаралельних програм вимагає розробки нових методів і технологій верифікації.

2. Найбільш важливими науковими і прикладними результатами, отриманими в роботі, є такі:

– метод компіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, що містить етапи верифікації структур семантико-числової специфікації послідовної програми, часової паралельної моделі програми з урахуванням особливостей обраної раціональної сукупності методів паралельної обробки інформації, часопараметризованої паралельної програми та перевірку відповідності показників ефективності заданим вимогам і обмеженням;

– метод декомпіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, головними етапами якого є декомпіляція структур часової моделі та перевірка еквівалентності декомпіляційної структури та СЧС вхідної послідовної програми, синтез текстової і СЧС специфікації паралельної програми та перевірка

еквівалентності текстової/графічної специфікації програми і тексту послідовної програми, для зменшення часу на верифікацію ураховуються особливості обраної сукупності методів паралельної обробки інформації;

– метод семантичної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, який містить етапи перевірки семантичної коректності вхідної послідовної програми задачі, статичної мультипаралельної програми, логічної еквівалентності часової мультипаралельної програми та вхідної послідовної програми задачі;

– інформаційна технологія верифікації паралельних часопараметризованих програм інформаційних управляючих систем, яка забезпечує перевірку синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації часопараметризованих мультипаралельних програм в динаміці їх проєктування з одночасною перевіркою збігу одиниць вимірювання фізичних величин, отриманих та одиниць вимірювання вхідних та вихідних даних задачі, що задаються користувачами.

3. Значення отриманих результатів для науки полягає в узагальненні та формалізації розв'язування задачі компіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, на основі верифікації процесу синтезу семантико-числових специфікацій паралельних часових моделей, часопараметризованих мультипаралельних програм і засобів оцінки їхніх показників ефективності; в узагальненні та формалізації розв'язування задачі декомпіляційної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, на основі декомпіляції структур часової моделі та перевірки еквівалентності декомпіляційної структури та СЧС вхідної послідовної програми з урахованням особливості обраної раціональної сукупності методів паралельної обробки інформації; в узагальненні та формалізації розв'язування задачі семантичної верифікації паралельних часопараметризованих програм для інформаційних управляючих систем, на основі використання семантико-

числових специфікацій задля перевірки семантичної коректності вхідної послідовної програми задачі, статичної мультипаралельної програми, логічної еквівалентності часової мультипаралельної програми та вхідної послідовної програми задачі; в новому формальному розв'язанні задачі верифікації паралельних часопараметризованих програм інформаційних управляючих систем, на основі застосування семантико-числових специфікацій для компіляційної, декомпіляційної та семантичної верифікації.

4. Значення отриманих результатів для практики полягає в такому. За рахунок використання методів та моделей паралельної часопараметризованої обробки даних ІУС здійснюється підвищення швидкості обробки інформації обчислювальними системами в процесі керування аеродінамічними об’єктами різних класів та призначень, що засвідчують патенти на корисну модель. При розробці та модернізації існуючих ІУС, в верифікації програмного забезпечення, яке реалізує вдосконалені комплекси алгоритмів (окремі алгоритми) управління та обробки інформації, в оцінці можливості їх реалізації в умовах заданого циклу (обмежень на час виконання), у виробленні рекомендацій щодо оптимізації конфігурації апаратних засобів їх паралельного виконання.

5. Під час розв'язання зазначеної наукової задачі використовували такі методи досліджень: теорія складних систем, теорія множин, теорія графів та апарат часових паралельних граф-схем (під час розроблення методів верифікації та інформаційної технології); методи імітаційного моделювання (під час оцінювання коректності та достовірності часових паралельних моделей).

6. Обґрунтованість і достовірність наукових положень, висновків і рекомендацій забезпечуються коректним застосуванням загальновизнаного математичного апарату, несуперечливістю з відомими положеннями теорії множин, теорії графів, методів імітаційного моделювання, обґрунтованим вибором, основних припущень та обмежень, прийнятих під час моделювання,

а також збіжністю результатів, отриманих під час практичного застосування розробленої системи.

Основні результати роботи реалізовано в Харківському національному університеті імені В.Н. Каразіна у рамках НДР «Моделі інформаційних процесів та методи їх обробки» за 2016–2020 рр.(ДР № 0116U003141); НДР «Виконання завдань Перспективного плану розвитку наукового напряму «Технічні науки» Харківського національного університету імені В. Н. Каразіна» (ДР № 0121U11306883) за 2021–2022 рр.; НДР «Моделювання інформаційних процесів у складних і розподілених системах» з 1.03.2021 р. до 31.12.2023 р. (ДР № 0121U109183).

7. Показано, що для оцінки показників ефективності розробленої технології проводились дослідження по цілому ряду практичних прикладних задач. До них можна віднести: метод Гауса для рішення систем лінійних рівнянь великої розмірності, алгоритм адаптивного згладжування і екстраполяції траєкторії, алгоритм цілерозподілу, то що. Дослідження показників ефективності верифікації паралельної реалізації алгоритму Гауса було проведено на обчислювальній робочій станції R-LINE з настільним процесором Intel Core i-5 10400F S1200 BOX 2.9G BX807 (Intel Core i5-10400F, Сокет s-1200, DDR4). Використання 4-х модулей пам'яті DDR4 16GB Kingston KVR32N22D8/16 з частотою 3200 МГц дозволило провести дослідження показників ефективності верифікації паралельної реалізації алгоритму Гауса рішення системи лінійних рівнянь з 4000000 елементів. За рахунок використання 4-х ядер та технології семантико-числової верифікації часопараметризованих мультипаралельних програм ІУС було досягнуто мінімальний час верифікації рішення СЛАР методом Гауса для 4000000 елементів, який складає T_{ver}=37 хв. Кількість виявлених дефектів N_{df} співпадає с кількістю засіяних дефектів (N_{df}=*100*%). Обсяг пам'яті *V*, яка потрібна для проведення верифікації дорівнює 417 Кбайт, обчислювальна потужність – 4 ядра. Для порівняння, час, потрібний експерту на проведення верифікації

рішення СЛАР методом Гауса склав порядку 5 год. Кількість виявлених дефектів $N_{df}=78\%$.

8. Отримані в роботі результати можуть бути рекомендовані до використання в науково-дослідних, проєктних організаціях під час розроблення паралельних часових моделей виконання алгоритмів і вибору якісних алгоритмів опрацювання інформації та керування, які задовольняють задані часові вимоги.

9 Сукупність отриманих у дисертації нових наукових результатів, позитивна оцінка їхньої достовірності, наукової та практичної значущості дають змогу вважати сформульовану наукову задачу розробки технології верифікації паралельних часопараметризованих програм інформаційних управляючих систем з метою підвищення ефективності верифікації за рахунок застосування компіляційної, декомпіляційної та семантичної верифікації на основі семантико-числових специфікацій, – розв'язаною, а поставлену мету – досягнутою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tanenbaum A. Computer Networks. Englewood Cliffs / A. Tanenbaum. – NJ: Prentice Hall, 3rded., 1996.

2. Дорошенко А.Ю. Паралельні обчислювальні системи. Методичний посібник і конспект лекцій. Київ: Видавничий дім «КМ Академія», 2013. 46 с.

3. Кузьменко Б.В., Чайковська О.А. Технологія розподілених систем та паралельних обчислень. (конспект лекцій, частина 1. Розподілені об’єктні системи, паралельні обчислювальні системи та паралельні обчислення, паралельне програмування на основі MPI) Навчальний посібник. К.: Видавничий центр КНУКІМ, 2011. 126. с

4. Коцовський В. М. Теорія паралельних обчислень : навчальний посібник. Ужгород: ПП «АУТДОР-Шарк», 2021. 188 с.

5. Харченко В. С., Замирець М. В., Засуха С. О., Поночовний Ю. Л. Елементи методології оперативної коригувальної верифікації програмних засобів інформаційно-управляючих систем космічних апаратів. Авиационно-космическая техника и технология. 2011р. № 6. С. 81–95.

6. Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Parallel programming: techniques and applications using networked workstations and parallel computers. 2nd Edition. University of North Carolina at CharlotteWestern Carolina University. 2005. 467p.

7. Семеренко В. П. Технології паралельних обчислень: навчальний посібник. Вінниця: ВНТУ, 2018. 104 с.

8. Рольщиков В. Б. Технології розподілених систем та паралельних обчислень. Змістовний модуль №1: конспект лекцій. Одеса, Одеський державний екологічний університет, 2018. 181 с.

9. Butenko V., Kharchenko V., Odarushchenko O., Popov P., Sklyar V., Odarushchenko E. Markov’s Model and Tool-Based Assessment of Safety-Critical I&C Systems: Gaps of the IEC 61508. Probabilistic Safety Assessment and Management PSAM 12, June 2014. - Honolulu, Hawaii, 2014.

10. Лучкова А. В. Аналіз методів верифікації програмного забезпечення, Матер. XLV наук.-техн. конф. ф-ту інформаційних технологій та комп'ютерної інженерії, 2–11 березня 2016 р., Вінниця: ВНТУ, с. 803–805.

11. Мірошник М. А., Клименко Л. А., Корольова Я. Ю. Технології та автоматизація проєктування цифрових пристроїв складних комп'ютерних систем на ПЛІС: Навч. посіб. Харків: УкрДУЗТ, 2021. 220 с.

12. Michael J. Quinn. Parallel Programming in C with MPI and OpenMP McGraw-Hill Education. Science/Engineering/ Math. 2003. 554p.

13. Поляков Г. А., Шматков С. И., Толстолужская Е. Г. Толстолужский Д. А. Синтез и анализ параллельных процессов в адаптивных времяпараметризованных вычислительных системах. Х.: ХНУ имени В.Н. Каразина, 2012. 672с.

14. Barry Wilkinson, Michael Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2/e Paperback – January 1, 2006. 496p.

15. Henri Casanova, Arnaud Legrand, Yves Robert. Parallel Algorithms. 1st Edition. New York. 2008. 360p. Режим доступу: https://doi.org/10.1201/9781584889465

16. George Em Karniadakis and Robert M. Kirby II. Parallel Scientific Computing in C++ and MPI. Cambridge University Press. Режим доступу: https://yangpl.files.wordpress.com/2017/12/parallel-scientific-computing-in-c-and-mpi-2003.pdf

17. Мороз О.Ю., Толстолузька О.Г., Савченко Р.В. Аналіз існуючих технологій верифікації паралельних програм. Вісник Харківського національного університету імені В. Н. Каразіна, Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», 46, 2020. 76–81.

18. Панченко Т. В. Композиційні методи специфікації та верифікації програмних систем. Автореф. дис. канд. фіз.-мат. наук. К., 2006.

19. Мороз О.Ю. Толстолузька О.Г. Анализ средств верификации параллельных программ. Проблеми інформатизації. Тези доповідей четвертої міжнародної науково-технічної конференції. – Черкаси – Баку – Бельсько-Бяла – Полтава, 3–4 листопада 2016р. С. 39–40

20. Інформаційно-керуючі системи. Локальні інформаційно-керуючі системи. Лабораторний практикум [Електронний ресурс] : навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Інтегровані інформаційні системи» спеціальності 126 «Інформаційні системи та технології» / П. І. Кравець, В. М. Шимкович, Ю. М. Бердник ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,9 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022. – 142 с. – Назва з екрана.

21. Xu Q., de Roever W.-P., He J. The Rely-Guarantee Method for Verifying Shared Variable Concurrent Programs. Formal Aspects of Computing. 1997. Vol. 9, N 2. P. 149–174.

22. Одарущенко О.Н. Оцінювання та забезпечення функційної безпеки при розробленні та ліцензуванні модулів і платформ для програмно-технічних комплексів інформаційно-керуючих систем. Системи управління, навігації та зв’язку. 2020. Вип.3(61). С.90–93

23. Winskel G. The Formal Semantics of Programming Languages: An Introduction. – London: MIT Press Foundations of Computing Series, 1993. – 361p

24. Trümper J., Bohnet J., Döllner J. Understanding complex multithreaded software systems by using trace visualization. SOFTVIS '10 Proceedings of the 5th international symposium on Software visualization. – ACM, Salt Lake City, Utah, USA, 2010. P.133–142

25. Moroz O., Sinyuk T. The computer decompilation verification model of parallel programs for distributed systems. Проблеми інформатизації. Тези доповідей п’ятої міжнародної науково-технічної конференції 13–15 листопада 2017 р., Черкаси – Баку – Бельсько-Бяла – Полтава, 2017р. С. 29

26. Biorner D., Jones C.B. The Vienna Development Methods (VDM): The Meta – Language. Vol. 61 of Lecture Notes in Computer Science. Springer Verlag, Heiderberg, Germany, 1978. 215p.

27. Dolores R. Wallase M. Ippolito, Cuthill B. Reference Information for the Software Verification and Validation Process. NIST Special Publication. 1996. 80p.

28. Лавріщева К. М. Програмна інженерія. Підручник. К. 2008. 319 с

29. Praun C. Detecting Synchronization Defects in Multi-Threaded Object-Oriented Programs, PhD thesis / Swiss Federal Institute of Technology. – Zurich, 2004.

30. Grama A., Gupta A., Karypis G., Kumar V. Introduction to parallel computing: design and analysis of algorithms (2nd Edition). Pearson, 2003. 656 p.

31. Бабенко Л.П., Лавріщева К.М. Основи програмноі інженерії. Київ: Знання, 2001. 269 с.

32. The RAISE Language Group. The RAISE Spesification Language. BCS Practitioner Series. Prentice Hall, 1982. 397 p.

33. Омельчук Л. Л. Формальні методи специфікації програм. К.: УкрІНТЕІ, 2010. 78 с.

34. Jones C. B. Systematic Software Development using VDM. London: PrenticeHall International. 1986.

35. RAISE Language Group. The RAISE Specification Language. BCS Practitioner Series. Prentice Hall. 1992. 397 p.

36. RAISE Method Group. The RAISE Development Method. BCS Practitioner Series. Prentice Hall. 1995. 493 p.

37. Brock S., George C.W. The RAISE Method Manual. Technical Report LACOS/CRI/DOC/3. CRI: Computer Resources International. 1990. 475 p.

38. Аналіз існуючих підходів до задання семантики мов специфікацій предметних областей та програмних систем: Звіт про науково-дослідну роботу (проміжний). Київський національний університет імені Тараса Шевченка. № ДР 0108Г002463. Київ: 2008. 57с.

39. M. Barnett, K. Rustan, M. Leino, W. Schulte. The Spec# programming system: An Overview. CASSIS 2004 Proceedings. 2004. P. 29–69.

40. Shengchao Qin; Guanhua He, Linking Object-Z with Spec# //Proc. of 12th IEEE International Conference on Engineering Complex Computer Systems, 11–14 July 2007. P. 185 – 196.

41. Толстолузька О. Г., Мороз О. Ю., Толстолузький Д. О. Метод компеляційно-семантичної верифікації часопараметризованих мультипаралельних програм. Computer Science and Cybersecurity, ISSNE 4(4), 2016. С. 26–34.

42. Дифучина О. Ю. Тестування паралельних програм на моделях. Математичне та імітаційне моделювання систем. МОДС 2018: тези доповідей Тринадцятої міжнародної науково-практичної конференції (м. Київ – с. Жукін, 25 червня – 29 80 червня 2018 р.) М-во осв. і наук. України, Нац. Акад. наук України, Академія технологічних наук України, Інженерна академія України та ін. Чернігів: ЧНТУ, 2018 С.231–234.

43. Youngjoo Woo, Seon Yeong Park, and Euiseong Seo. Virtual battery: A testing tool for power-aware software. Journal of Systems Architecture, 2013. URL: https://doi.org/10.1016/j.sysarc.2013.06.006.

44. Шликов В. В., Данілова В. А. Високопродуктивні розподілені обчислювальні системи: Практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки», спеціалізації «Інформаційні технології в біології та медицині»; КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2018. 108 с.

45. Ясько М. М. Навчальний посібник до вивчення курсів «Паралельна обробка даних» та «Мови обчислень та кластерні системи» [Текст] / М. М. Ясько. Д.: РВВ ДНУ, 2010. 76с.

46. Навчальний посібник з дисципліни Системи візуалізації та розпізнавання образів [навчальний посібник] / Смолій В. В., Савицька Я. А., Місюра М.Д., Шкарупило В. В. // К.: ФОП Ямчинський О.В., 2020. 200 с.

47. Інформаційні технології: Системи комп’ютерної математики [Електронний ресурс] : навч. посіб. для студ. спеціальності «Автоматизація та комп’ютерно-інтегровані технології» / І. В. Кравченко, В. І. Микитенко; КПІ ім. Ігоря Сікорського . Електронні текстові дані (1 файл: 5,57 Мбайт). Київ : КПІ ім. Ігоря Сікорського, 2018. 243с.

48. Юнчик В., Федонюк А. Порівняльна характеристика функціональних можливостей систем комп’ютерної математики в процесі розв’язування задач. 2019. С. 90 – 102/ https://doi.org/10.23939/sisn2019.02.090

49. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley, Reading, MA. 1999.

50. Filho, J., Braga, J. (2008). UML: Unified Modeling Language. In: Shekhar, S., Xiong, H. (eds) Encyclopedia of GIS. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35973-1_1419

51. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник. Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. 434 с.

52. Foster, Ian. Designing and Building Parallel Programs [Text] / Ian Foster - AddisonWesley, Inc. Boston, MA, 1995. 381pp.

53. Толстолужская В. В., Толстолужская Е. Г., Мороз О. Ю. Программная модель исследования эффективности трудозатрат на разработку параллельных программ. Труды международной НТК «Компьютерное моделирование в наукоемких технологиях», Харьков, 28 – 31 мая 2014 С.378–381.

54. Benjamin Kormann, Dmitry Tikhonov, and Birgit Vogel-Heuser. Automated plc software testing using adapted UML sequence diagrams. IFAC Proceedings Volume. URL: https://doi.org/10.3182/20120523-3-RO-2023.00148.

55. Мороз О.Ю., Синюк Б.В., Синюк Т.В., Уразов С.А. Методи верифікації паралельних програм. Сучасні напрями розвитку інформаційно-комунікацій-них технологій та засобів управління. Матеріали сьомої міжнародної науково-технічної конференції, Полтава – Баку – Кіровоград – Харків, 2017р. С. 7.

56. Толстолузький Є. Д., Бердніков А. Г., Будько В. В., Толстолузька О. Г., Мороз О. Ю. Розробка та верифікація СЧС моделі мережевого планування. Вісник Харківського національного університету імені В. Н. Каразіна серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», випуск 51, 2021, С. 81–86 DOI: https://doi.org/10.26565/2304-6201-2021-51

57. Moroz O., Sinyuk T. The computer decompilation verification model of parallel programs for distributed systems. Проблеми інформатизації. Тези доповідей п’ятої міжнародної науково-технічної конференції 13–15 листопада 2017 р., Черкаси – Баку – Бельсько-Бяла – Полтава, 2017р. С. 29.

58. Мороз О. Ю., Толстолузька О. Г. Толстолузький Є.Д. Концептуальна модель технології семантико-числової верифікації часопараметризованих мультипаралельних програм. Moderní aspekty vědy: XXXI. Díl mezinárodní kolektivní monografie / Mezinárodní Ekonomický Institut s.r.o.. Česká republika: Mezinárodní Ekonomický Institut s.r.o., 2023. Oddíl 12. Počítačová vědy §12.1 str. 433–452.

59. Мороз О. Ю., Толстолузька О. Г. Аналіз засобів технологій верифікації паралельних програм. Комп’ютерне моделювання в наукоємних технологіях: Збірник наукових праць міжнародної науково-технічної конференції (м. Харків, 21–23 квітня 2021 року) Харків: ХНУ ім. В.Н. Каразіна, 2021. С. 224–227.

60. Dolores R. Wallase M. Ippolito, Cuthill B. Reference Information for the Software Verification and Validation Process. NIST Special Publication. 1996. 80p.

61. Мороз О. Ю. Технологія семантико-числової верифікації часопараметризованих паралельних програм для інформаційних і управляючих систем. *Вісник Харківського національного університету імені В. Н.Каразіна, сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2022. випуск. 55. С.43–48. DOI: https://doi.org/10.26565/2304-6201-2022-55-04

62. Мороз О. Ю. Компіляційно-семантична верифікація часопараметризованих мультипаралельних програм для інформаційних управляючих систем. *Вісник Харківського національного університету імені В.Н.Каразіна, серія. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2022. випуск 56. С.43–50. DOI: https://doi.org/10.26565/2304-6201-2022-56-04

63. Галузева система управління якістю. Гарантоздатність програмно-технічних комплексів критичного призначення. Настанова Національного космічного агентства України СОУ-Н НКАУ 0060:2010 / Харченко В.С. (наук. керівник розробки). 2011. 60 с.

64. Авраменко А. С., Авраменко В. С., Косенюк Г. В. Тестування програмного забезпечення: навч. посіб. Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284с.

65. Галузева система управління якістю. Методи оцінки показників якості програмного забезпечення програмно-технічних комплексів критичного призначення. Настанова Національного космічного агентства України СОУ- Н НКАУ 0031:2007 / Конорев Б.М. (наук. керівник розробки). 2007. 127с.

66. Галузева система управління якістю. Процеси життєвого циклу програмного забезпечення програмно-технічних комплексів критичного призначення. Настанова Національного космічного агентства України СОУ-Н НКАУ 0061:2011. Харченко В.С. (наук. керівник розробки). 2011. 123с.

67. Галузева система управління якістю. Верифікація програмного забезпечення програмно-технічних комплексів критичного призначення. Настанова Державного космічного агентства України СОУ-Н ДКАУ (проект). Харченко В.С. (наук. керівник розробки). 2011. 80 с.

68. Бурячок Л. В. Стратегія обґрунтування раціонального набору параметрів і функцій програмних засобів спеціального призначення, спрямованих на розв’язання завдань інформаційної діяльності. Сучасний захист інформації. К.: ДУТ, 2016. № 3. С. 3–9.

69. Козачок В.А., Рой Я.В., Бурячок Л.В. Технології протидії шкідливим програмам та завідомо фальшивому програмному забезпеченню. Сучасний захист інформації. К.: ДУТ, 2017. № 2(30). С. 30–34.

70. Singh R. An Optimized Task Duplication Based Scheduling in Parallel System. International Journal of Intelligent Systems and Applications. 2016. №8 (8). P.26–37

71. Сайт Української команди розподілених обчислень. – Режим доступу: http://distributed.org.ua/

72. Мороз О. Ю., Синюк Б. В., Синюк Т. В., Уразов С. А. Методи верифікації паралельних програм. Сучасні напрями розвитку інформаційно-комунікацій-них технологій та засобів управління. Матеріали сьомої міжнародної науково-технічної конференції, Полтава – Баку – Кіровоград – Харків, 2017р. С. 7.

73. Толстолузька О. Г., Мороз О. Ю., Паршенцев Б. В., Дослідження розпаралелювання градієнтного бустингу. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали восьмої міжнародної науково-технічної конференції. – Полтава –Баку– Харків–Жиліна, 26 – 27 квітня 2018р. С. 84.

74. Будько М. М., Василенко В. С., Короленко М. П. Варіант формалізації процесу захисту інформації в комп'ютерних системах та оптимізації його цільової функції // Реєстрація, зберігання і оброб. даних. 2000. Т 2, № 2. С. 73–84.

75. ISO/IEC/IEEE 12207:2017 Systems and software engineering. Software life cycle processes.

76. Харченко В. С. Гарантоздатність комп'ютерних систем: межа універсальності в контексті інформаційно-технічного стану // Радіоелектронні і комп'ютерні системи. 2007. № 8. С. 8–16

77. Dmytro Chumachenko, Ievgen Meniailov, Andrii Hrimov, Vladislav Lopatka, Olha Moroz, Olena Tolstoluzka. Simulation and forecasting of the influenza epidemic process using seasonal autoregressive integrated moving average

model. Radioelectronic and Computer Systems, 2021, no. 4(100), Pp.22–35 DOI: https://doi.org/10.32620/reks.2021.4.02 (*Scopus*).

78. BS ISO/IEC/IEEE 16085:2021 Systems and software engineering. Life cycle processes Risk management. [Publication date 2021-01-18]. 2021. p.62.

79. IEC 62138:2018 Nuclear power plants. Instrumentation and control systems important for safety-Software aspects for computer-based systems perfoming category A or C functions. [Publication date 2018-07-31]. 2018. p.106.

80. IEC 60987:2021 Ed. 2.1. Nuclear power plants. Instrumentation and control important to safety. Hardware design requirements for computer-based systems. [Publication date 2021-02-03]. 2021. p.101.

81. BS ISO/IEC/IEEE 12207:2017 Systems and software engineering Software life cycle processes. [Publication date 2018-03-05]. 2018. p.158.

82. ДСТУ ISO/IEC 25010:2016 Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів (ISO/IEC 25010:2011, IDT). [Чинний від 27-12-2016]. Київ, 2016. (Інформація та документація). 2016.

83. ISO/IEC 25023:2016 Systems and software engineering Systems and software Quality Requirements and Evaluation (SQuaRE) Measurement of system and software product quality. Part 3: Internal metrics». [Publication date 2016-06-05]. 2016. p.45.

84. IEEE 1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation. [Publication date 2016-09-29]. 2017.

85. IEEE 7-4.3.2-2016. IEEE Standard Criteria for Programmable Digital Devices in Safety Systems of Nuclear Power Generating Stations. [Publication date 25-08-2016]. 2016. p.86

86. IEEE 1008:1987 (R2002) Standard for Software Unit Testing. [Publication date 1986-12-11]. 1986.

87. IEEE 1016:2009 Standard for Information technology. Systems Design. Software Design Descriptions. [Publication date 2009-07-20]. 2009.

88. ДСТУ ISO 9001:2015. Системи управління якістю. Вимоги. [Чинний від 31-12-2015]. Київ, 2015. (Інформація та документація). 2015.

89. ДСТУ ISO/IEC TR 9126-3:2012 Програмна інженерія. Якість продукту. [Чинний від 28-11-2012]. (Інформація та документація). 2012.

90. ДСТУ ISO/IEC 25023:2019 Інженерія систем і програмних засобів. Вимоги до якості систем програмних засобів та їхнього оцінювання (SQuaRE). Вимірювання якості систем та програмних продуктів. [Чинний від 16-10-2019]. Київ, 2019. (Інформація та документація). 2019.

91. Корисна модель. Канал вимірювання кутових швидкостей літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Толстолузька О. Г., Звєрєв О.О., Садовий К. В. та ін., усього 10 осіб// Патент України на корисну модель №120560 від 10.11.2017 G01S 17/42, G01S 17/66.

92. Корисна модель. Канал вимірювання радіальної швидкості літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Толстолузька О. Г., Садовий К. В., Звєрєв О. О., та ін., усього 10 осіб// Патент України на корисну модель №120559 від 10.11.2017 G01S 17/42, G01S 17/66

93. Корисна модель. Канал автоматичного супроводження літальних апаратів за напрямком оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Кузнєцов О. Л., Сачук І. І., Довбня О. В. та ін., усього 10 осіб // Патент України на корисну модель №120557 від 10.11.2017 G01S 17/66, G01S 17/42.

94. Корисна модель. Канал вимірювання похилої дальності до літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Садовий К. В., Толстолузька О. Г., Довбня О. В. та ін., усього 10 осіб //Патент України на корисну модель №121427 від 11.12.2017 G01S 17/66, G01S 17/42.

95. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання радіальної швидкості літальних апаратів з можливістю

формування і обробки їх зображення та кібернетичним захистом інформації, 05.01.2022, № 150146, Бюлетень № 1, 2022

96. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал автоматичного супроводження літальних апаратів за напрямком з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150147, 05.01.2022, бюл. № 1/2022

97. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання кутових швидкостей літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150148, 05.01.2022, бюл. № 1/2022

98. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання похилої дальності до літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150194, 12.01.2022, бюл. № 2/2022

99. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання радіальної швидкості літальних апаратів з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150891, 04.05.2022, бюл. № 18/2022

100. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал автоматичного супроводження літальних апаратів за напрямком з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150892, 04.05.2022, бюл. № 18/2022

101. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання кутових швидкостей літальних апаратів з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150893, 04.05.2022, бюл. № 18/2022.

Додаток А.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях, що входять до міжнародних наукометричних баз

29. Dmytro Chumachenko, Ievgen Meniailov, Andrii Hrimov, Vladislav Lopatka, Olha Moroz, Olena Tolstoluzka. Simulation and forecasting of the influenza epidemic process using seasonal autoregressive integrated moving average model. Radioelectronic and Computer Systems, 2021, no. 4(100), Pp.22–35

DOI: https://doi.org/10.32620/reks.2021.4.02 **(*Scopus*).**

(Особистий внесок: розробка методу формальної верифікації та засобів його програмної реалізації в ході експериментальних результатів моделювання епідемічного процесу грипу, а також написання частини тексту та його переклад).

Статті у наукових фахових виданнях України

30. Olena Tolstoluzka, Dmitriy Tolstoluzkiy, Olga Moroz. Compilations method and semantic verification time parameterized of multiparallel programs. Computer Scienceand Cybersecurity, ISSNE 4(4), 2016 С. 26–34. DOI: https://periodicals.karazin.ua/cscs/article/view/8264

(Особистий внесок: участь у розробці методу семантичної верифікації часопараметризованих мультипаралельних програм, а також написання тексту та його переклад).

31. Parshentsev B., Tolstoluzka O., Moroz O. Parallel implementation of the method of gradient boosting. Advanced Information Systems. 2018. Vol. 2, No. 3 P. 19–23. DOI: https://doi.org/10.20998/2522-9052.2018.3.03

Особистий внесок: участь у процесі перевірки синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації, обробка результатів, а також написання частини тексту та його пеерклад.

32. Moroz O.Yu., Tolstoluzka O. G., Savchenko R.V. Analysis of existing technologies for verification of parallel programs. Bulletin of V. Karazin Kharkiv National University series «Mathematical Modelling. Information Technology. Automated Control Systems». 2020. Issue 46. P. 76–81 DOI: 10.26565/2304-6201-2020-46-07 DOI: https://doi.org/10.26565/2304-6201-2020-46

(Особистий внесок: опис методів верифікації паралельних програм, експертизи програмного забезпечення, статичного аналізу, динамічних та формальних методів верифікації, а також написання частини тексту та його переклад).

33. Толстолузький Є. Д., Бердніков А. Г., Будько В. В., Толстолузька О. Г., Мороз О. Ю. Розробка та верифікація СЧС моделі мережевого планування. Вісник Харківського національного університету імені В. Н. Каразіна серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», випуск 51, 2021, С. 81–86. DOI: https://doi.org/10.26565/2304-6201-2021-51

(Особистий внесок: участь у розробленні, підготовці та верифікації семантико-числової специфікації моделі мережевого планування, а також написанні частини тексту).

34. Мороз О. Ю. , Толстолузька О. Г. Використання методів формального синтезу та веріфікації паралельних часопараметризованих моделей для рішення системи лінійних рівнянь методом Гауса. Вісник Харківського національного університету імені В. Н. Каразіна серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», випуск 52, 2021 С. 55–71. DOI: https://doi.org/10.26565/2304-6201-2021-52-07

(Особистий внесок: обробка та перевірка синтаксичної та часової коректності формального синтезу структур семантико-числової специфікації паралельних часопараметризованих моделей для рішення системи лінійних рівнянь методом Гауса, участь в обговоренні отриманих результатів та написанні тексту).

35. Мороз О. Ю. Технологія семантико-числової верифікації часопараметризованих паралельних програм для інформаційних і управляючих систем. *Вісник Харківського національного університету імені В. Н. Каразіна, сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2022. випуск. 55. С.43–48. DOI: https://doi.org/10.26565/2304-6201-2022-55-04

36. Мороз О. Ю. Компіляційно-семантична верифікація часопараметризованих мультипаралельних програм для інформаційних управляючих систем. *Вісник Харківського національного університету імені В.Н.Каразіна, серія. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2022. випуск 56. С.43–50. DOI: https://doi.org/10.26565/2304-6201-2022-56-04

Монографії

37. Мороз О. Ю., Толстолузька О. Г. Толстолузький Є.Д. Концептуальна модель технології семантико-числової верифікації часопараметризованих мультипаралельних програм. Moderní aspekty vědy: XXXI. Díl mezinárodní kolektivní monografie / Mezinárodní Ekonomický Institut s.r.o.. Česká republika: Mezinárodní Ekonomický Institut s.r.o., 2023. Oddíl 12. Počítačová vědy §12.1 str. 433-452. URL: http://perspectives.pp.ua/public/site/mono/mono-31.pdf

Особистий внесок: участь у побудові моделі технології семантико-числової верифікації часопараметризованих мультипаралельних програм, обговоренні результатів застосування методів формальної верифікації паралельних часопараметризованих моделей, участь в обговоренні та інтерпретації отриманих результатів, написання тексту.

Патенти

38. Корисна модель. Канал вимірювання кутових швидкостей літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Толстолузька О. Г., Звєрєв О.О., Садовий К. В. та ін., усього 10 осіб// Патент України на корисну модель №120560 від 10.11.2017 G01S 17/42, G01S 17/66.

39. Корисна модель. Канал вимірювання радіальної швидкості літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Толстолузька О. Г., Садовий К. В., Звєрєв О. О., та ін., усього 10 осіб// Патент України на корисну модель №120559 від 10.11.2017 G01S 17/42, G01S 17/66

40. Корисна модель. Канал автоматичного супроводження літальних апаратів за напрямком оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Кузнєцов О. Л., Сачук І. І., Довбня О. В. та ін., усього 10 осіб // Патент України на корисну модель №120557 від 10.11.2017 G01S 17/66, G01S 17/42.

41. Корисна модель. Канал вимірювання похилої дальності до літальних апаратів з оптико-електронним модулем для мобільної суміщеної лазерної вимірювальної системи. /Садовий К. В., Толстолузька О. Г., Довбня О. В. та ін., усього 10 осіб //Патент України на корисну модель №121427 від 11.12.2017 G01S 17/66, G01S 17/42.

42. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання радіальної швидкості літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації, 05.01.2022, № 150146, Бюлетень № 1, 2022

43. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал автоматичного супроводження літальних апаратів за напрямком з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150147, 05.01.2022, бюл. № 1/2022

44. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання кутових швидкостей літальних апаратів з можливістю формування і обробки їх зображення та кібернетичним захистом інформації. № 150148, 05.01.2022, бюл. № 1/2022

45. Артюх О. А., Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання похилої дальності до літальних апаратів з можливістю

формування і обробки їх зображення та кібернетичним захистом інформації. № 150194, 12.01.2022, бюл. № 2/2022

46. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання радіальної швидкості літальних апаратів з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150891, 04.05.2022, бюл. № 18/2022

47. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал автоматичного супроводження літальних апаратів за напрямком з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150892, 04.05.2022, бюл. № 18/2022

48. Коломійцев О. В., Мороз О. Ю., Толстолузька О. Г. Канал вимірювання кутових швидкостей літальних апаратів з використанням частот міжмодових биттів та можливістю формування і обробки їх зображення з кібернетичним захистом отриманої інформації. № 150893, 04.05.2022, бюл. № 18/2022

Наукові праці, які засвідчують апробацію матеріалів дисертації:

49. Толстолужская В. В., Толстолужская Е. Г., Мороз О. Ю. Программная модель исследования эффективности трудозатрат на разработку параллельных программ. Труды международной НТК «Компьютерное моделирование в наукоемких технологиях», Харьков, 28 – 31 мая 2014р. С.378–381

50. Поляков Г. О., Мороз О.Ю., Толстолузький Д. О. Метод компиляционно-семантической верификации статических и времяпараметризованных мультипараллельных программ. Проблеми автоматизації, Черкаси, 12–13 листопада 2015 р. С. 61

51. Мороз О.Ю. Толстолузька О.Г. Анализ средств верификации параллельных программ. Проблеми інформатизації. Тези доповідей четвертої міжнародної науково-технічної конференції. – Черкаси – Баку – Бельсько-Бяла – Полтава, 3–4 листопада 2016р. С. 39–40

52. Мороз О.Ю., Синюк Б.В., Синюк Т.В., Уразов С.А. Методи верифікації паралельних програм. Сучасні напрями розвитку інформаційно-комунікацій-них технологій та засобів управління. Матеріали сьомої міжнародної науково-технічної конференції, Полтава – Баку – Кіровоград – Харків, 2017р. С. 7.

53. Moroz O., Sinyuk T. The computer decompilation verification model of parallel programs for distributed systems. Проблеми інформатизації. Тези доповідей п’ятої міжнародної науково-технічної конференції 13–15 листопада 2017 р., Черкаси – Баку – Бельсько-Бяла – Полтава, 2017р. С. 29

54. Толстолузька О. Г., Мороз О. Ю., Паршенцев Б. В., Дослідження розпаралелювання градієнтного бустингу. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали восьмої міжнародної науково-технічної конференції. – Полтава –Баку– Харків–Жиліна, 26 – 27 квітня 2018р. С. 84.

55. Мороз О. Ю., Толстолузька О. Г. Аналіз існуючих технологій верифікації паралельних програм. Комп’ютерне моделювання в наукоємних технологіях: Збірник наукових праць міжнародної науково-технічної конференції (м. Харків, 22–24 квітня 2020 року) – Харків: ХНУ ім. В.Н. Каразіна, 2020. №6 С.215–218

56. Мороз О. Ю., Толстолузька О. Г. Аналіз засобів технологій верифікації паралельних програм. Комп’ютерне моделювання в наукоємних технологіях: Збірник наукових праць міжнародної науково-технічної конференції (м. Харків, 21–23 квітня 2021 року) – Харків: ХНУ ім. В.Н. Каразіна, 2021. С. 224–227.

Додаток Б.

Код програми синтезу паралельних чаоспараметризованих моделей алгоритму Гауса

```
include<iostream>
#include<string>
#include <fstream>
#include<windows.h>
using namespace std;

void Read1(int **&mas, int &n, int &m, string *&str, string *&st)
{
	for(int i = 0; i < n; i++)
		delete [] mas[i];
	delete [] mas;

	delete [] str;

	ifstream f;
	f.open("D:\\vixvix1.txt");
	int count = 0;

	string s;

	while(!f.eof())
	{
		for(int i = 0; i < m + 1; i++)
			f >> s;
		count++;
	}

	count -= 2;

	n = count;

	mas = new int* [n];
	for(int i = 0; i < n; i++)
		mas[i] = new int [m];

	str = new string [n];

	f.close();

	ifstream f1;
	f1.open("D:\\vixvix1.txt");

	for(int j = 0; j < m + 1; j++)
		f1 >> st[j];


	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < m; j++)
		{
			f1 >> mas[i][j];
		}

		f1 >> str[i];
	}
	f1.close();
}

//////////////////////////////////////////////////////////////////////
void Read2(int **&mas, int &n, int &m, string *&st)
{
	for(int i = 0; i < n; i++)
		delete [] mas[i];
	delete [] mas;

	ifstream f;
	f.open("D:\\vixvix2.txt");
	int count = 0;

	string s;

	while(!f.eof())
```

```
	{
		for(int i = 0; i < m; i++)
			f >> s;
		count++;
	}

	count -= 2;

	n = count;

	mas = new int* [n];
	for(int i = 0; i < n; i++)
		mas[i] = new int [m];

	f.close();

	ifstream f1;
	f1.open("D:\\vixvix2.txt");

	for(int j = 0; j < m; j++)
		f1 >> st[j];


	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < m; j++)
		{
			f1 >> mas[i][j];
		}
	}
	f1.close();
}
////////////////////////////////////////////////////////
void Read3(double **&mas, int &n, int &m, string *&st)
{
	for(int i = 0; i < n; i++)
		delete [] mas[i];
	delete [] mas;

	ifstream f;
	f.open("D:\\NATA.txt");
	int count = 0;

	string s;

	while(!f.eof())
	{
		for(int i = 0; i < m; i++)
			f >> s;
		count++;
	}

	count -= 2;

	n = count;

	mas = new double* [n];
	for(int i = 0; i < n; i++)
		mas[i] = new double [m];

	f.close();

	ifstream f1;
	f1.open("D:\\NATA.txt");

	for(int j = 0; j < m; j++)
		f1 >> st[j];


	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < m; j++)
		{
			f1 >> mas[i][j];
		}
	}
	f1.close();
}

////////////////////////////////////////////////////////////
```

```
void Write1(int **&mas, int &n, int &m, string *&str, string *&st)
{
	ofstream f;
	f.open("D:\\vixvix1.txt");
	int p = 3;

	for(int j = 0; j < m + 1; j++)
	{
		p -= st[j].length() - 1;
		for(int i = 0; i < p; i++)
			f << " ";
		p = 5;
		f << st[j];
	}

	f << endl;

	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < m; j++)
		{
			if(j == 0)
				p = 4;
			else
				p = 5;

			if(mas[i][j] > -1 && mas[i][j] < 10)
				p -= 1;
			if((mas[i][j] > 9 && mas[i][j] < 100) || mas[i][j] < 0)
				p -= 2;
			if(mas[i][j] > 99 && mas[i][j] < 1000)
				p -= 3;
			if(mas[i][j] > 999)
				p -= 4;

			for(int z = 0; z < p; z++)
				f << " ";
			f << mas[i][j];
			f << " ";
		}
		p = 2;
		for(int z = 0; z < p; z++)
				f << " ";
		f << str[i];
		f << endl;
	}
	f.close();
}
////////////////////////////////////////////////////////////
void Write2(int **&mas, int &n, int &m, string *&st)
{
	ofstream f;
	f.open("D:\\vixvix2.txt");
	int p = 3;

	for(int j = 0; j < m; j++)
	{
		p -= st[j].length() - 1;
		for(int i = 0; i < p; i++)
			f << " ";
		p = 5;
		f << st[j];
	}

	f << endl;

	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < m; j++)
		{
			if(j == 0)
				p = 4;
			else
				p = 5;

			if(mas[i][j] > -1 && mas[i][j] < 10)
				p -= 1;
			if((mas[i][j] > 9 && mas[i][j] < 100) || mas[i][j] < 0)
				p -= 2;
			if(mas[i][j] > 99 && mas[i][j] < 1000)
```

```
                        p -= 3;
                if(mas[i][j] > 999)
                        p -= 4;

                for(int z = 0; z < p; z++)
                        f << " ";
                f << mas[i][j];
                f << " ";
        }
        f << endl;
    }
    f.close();
}
////////////////////////////////////////////////////////////////
void Write3(double **&mas, int &n, int &m, string *&st)
{
    ofstream f;
    f.open("D:\\NATA.txt");
    int p = 3;

    for(int j = 0; j < m; j++)
    {
        p -= st[j].length() - 1;
        for(int i = 0; i < p; i++)
                f << " ";
        p = 5;
        f << st[j];
    }

    f << endl;

    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < m; j++)
        {
                if(j == 0)
                        p = 4;
                else
                        p = 5;

                if(mas[i][j] > -1 && mas[i][j] < 10)
                        p -= 1;
                if((mas[i][j] > 9 && mas[i][j] < 100) || mas[i][j] < 0)
                        p -= 2;
                if(mas[i][j] > 99 && mas[i][j] < 1000)
                        p -= 3;
                if(mas[i][j] > 999)
                        p -= 4;

                for(int z = 0; z < p; z++)
                        f << " ";
                f << mas[i][j];
                f << " ";
        }
        f << endl;
    }
    f.close();
}


////////////////////////////////////////////////////////////////

void Fun(int op2, double *&mas0, int **mas1, int **mas2, double **mas3)
{
    mas3[op2][1] += 2;

    if(mas1[op2][9] != -1)
    {
        int adr = mas1[op2][9];

        if(mas3[mas2[adr][7]][1] == mas0[mas2[adr][7]])
                Fun(mas2[adr][7], mas0, mas1, mas2, mas3);

        while(mas2[adr][6] != -1)
        {
                adr++;
                if(mas3[mas2[adr][7]][1] == mas0[mas2[adr][7]])
                        Fun(mas2[adr][7], mas0, mas1, mas2, mas3);
        }
    }
```

```
}

//*****************************************************************************
void main()
{
	int **mas1;
	int **mas2;
	double **mas3;
	double *mas0;
	int n1 = 10, n2 = 10, n3 = 10;
	int m1 = 18, m2 = 11, m3 = 4;
	string *str1;
	string *st1;
	string *st2;
	string *st3;



//=============================================================================

	bool flag = true;

	while(flag)
	{
		mas1 = new int* [n1];
		for(int i = 0; i < n1; i++)
			mas1[i] = new int [m1];


		mas2 = new int* [n2];
		for(int i = 0; i < n2; i++)
			mas2[i] = new int [m2];

		mas3 = new double* [n3];
		for(int i = 0; i < n3; i++)
			mas3[i] = new double [m3];

		str1 = new string [n1];
		st1 = new string [m1 + 1];
		st2 = new string [m2];
		st3 = new string [m3];


		Read1(mas1, n1, m1, str1, st1);
		Read2(mas2, n2, m2, st2);
		Read3(mas3, n3, m3, st3);

		mas0 = new double[n3];
		for(int i = 0; i < n3; i++)
			mas0[i] = mas3[i][1];


		int op1;
		int op2;
		system("cls");
		cout << "Vvedite operator 1 -> \n";
		cin >> op1;

		cout << "Vvedite operator 2 -> \n";
		cin >> op2;

		int adr = mas1[op1][9];
		int c = 0;

		system("cls");

		while(mas2[adr][7] != op2)
		{
			if(mas2[adr][6] != -1)
			{
				adr++;
			}
			else
			{
				c = 1;
				break;
			}
		}

		if(c == 1)
```

```
{
	cout << "Svazi net!\n";
}
else
{

	int **mas_temp;
	double **mas_temp1;
	string *str_temp;

	mas_temp = new int* [n1];
	for(int i = 0; i < n1; i++)
		mas_temp[i] = new int [m1];

	str_temp = new string [n1];

	for(int i = 0; i < n1; i++)
		str_temp[i] = str1[i];

	for(int i = 0; i < n1; i++)
		for(int j = 0; j < m1; j++)
			mas_temp[i][j] = mas1[i][j];

	for(int i = 0; i < n1; i++)
		delete [] mas1[i];
	delete [] mas1;

	delete [] str1;

	n1 += 2;

	mas1 = new int* [n1];
	for(int i = 0; i < n1; i++)
		mas1[i] = new int [m1];

	str1 = new string [n1];

	for(int i = 0; i < n1 - 2; i++)
		for(int j = 0; j < m1; j++)
			mas1[i][j] = mas_temp[i][j];

	for(int i = 0; i < n1 - 2; i++)
		str1[i] = str_temp[i];

	delete [] str_temp;

	for(int i = 0; i < n1 - 2; i++)
		delete [] mas_temp[i];
	delete [] mas_temp;

	////////////////////////////////////
	mas_temp = new int* [n2];
	for(int i = 0; i < n2; i++)
		mas_temp[i] = new int [m2];

	for(int i = 0; i < n2; i++)
		for(int j = 0; j < m2; j++)
			mas_temp[i][j] = mas2[i][j];

	for(int i = 0; i < n2; i++)
		delete [] mas2[i];
	delete [] mas2;

	n2 += 2;

	mas2 = new int* [n2];
	for(int i = 0; i < n2; i++)
		mas2[i] = new int [m2];

	for(int i = 0; i < n2 - 2; i++)
		for(int j = 0; j < m2; j++)
			mas2[i][j] = mas_temp[i][j];

	for(int i = 0; i < n2 - 2; i++)
		delete [] mas_temp[i];
	delete [] mas_temp;
	////////////////////////////////

	mas_temp1 = new double* [n3];
	for(int i = 0; i < n3; i++)
```

```
        mas_temp1[i] = new double [m3];

for(int i = 0; i < n3; i++)
        for(int j = 0; j < m3; j++)
                mas_temp1[i][j] = mas3[i][j];

for(int i = 0; i < n3; i++)
        delete [] mas3[i];
delete [] mas3;

n3 += 2;

mas3 = new double* [n3];
for(int i = 0; i < n3; i++)
        mas3[i] = new double [m3];

for(int i = 0; i < n3 - 2; i++)
        for(int j = 0; j < m3; j++)
                mas3[i][j] = mas_temp1[i][j];

for(int i = 0; i < n3 - 2; i++)
        delete [] mas_temp1[i];
delete [] mas_temp1;



str1[n1 - 2] = "send";
str1[n1 - 1] = "recive";
////////////////////////////////////mas1
mas1[n1 - 2][0] = n1 - 2;
mas1[n1 - 1][0] = n1 - 1;

for(int i = 1; i < 5; i++)
{
        mas1[n1 - 2][i] = 0;
        mas1[n1 - 1][i] = 0;
}

mas1[n1 - 2][5] = 301;
mas1[n1 - 1][5] = 302;

mas1[n1 - 2][6] = n2 - 2;
mas1[n1 - 1][6] = n2 - 1;

mas1[n1 - 2][7] = 1;
mas1[n1 - 1][7] = 1;

mas1[n1 - 2][8] = mas1[op1][8];
mas1[n1 - 1][8] = mas1[op2][8];

mas1[n1 - 2][9] = n2 - 2;
mas1[n1 - 1][9] = n2 - 1;

for(int i = 10; i < m1; i++)
{
        if(i == 10 || i == 13)
        {
                mas1[n1 - 2][i] = 1;
                mas1[n1 - 1][i] = 1;
        }
        else
        {
                mas1[n1 - 2][i] = 0;
                mas1[n1 - 1][i] = 0;
        }
}
////////////////////////////////////mas2
mas2[n2 - 2][0] = n2 - 2;
mas2[n2 - 1][0] = n2 - 1;

mas2[n2 - 2][1] = -1;
mas2[n2 - 1][1] = -1;

mas2[n2 - 2][2] = op1;
mas2[n2 - 1][2] = n1 - 2;


adr = mas1[op1][9];

while(mas2[adr][7] != op2)
```

```
                {
                        adr++;
                }

                mas2[adr][7] = n1 - 2;

                mas2[n2 - 2][3] = mas2[adr][9];
                mas2[n2 - 2][4] = mas2[adr][8];

                mas2[n2 - 2][5] = 0;

                for(int i = 3; i < 6; i++)
                        mas2[n2-1][i] = 0;

                mas2[n2 - 2][6] = -1;
                mas2[n2 - 1][6] = -1;

                mas2[n2 - 2][7] = n1 - 1;
                mas2[n2 - 1][7] = op2;

                for(int i = 8; i < 11; i++)
                        mas2[n2 - 2][i] = 0;



                adr = mas1[op2][6];

                while(mas2[adr][2] != op1)
                {
                        adr++;
                }

                mas2[adr][2] = n1 - 1;

                mas2[n2 - 1][8] = mas2[adr][4];
                mas2[n2 - 1][9] = mas2[adr][3];

                mas2[n2 - 1][10] = 0;

                mas3[n3 - 2][0] = n3 - 2;
                mas3[n3 - 1][0] = n3 - 1;

                mas3[n3 - 2][1] = mas3[op1][1] + 1;
                mas3[n3 - 1][1] = mas3[n3 - 2][1] + 1;

                mas3[n3 - 2][2] = 1;
                mas3[n3 - 1][2] = 1;

                mas3[n3 - 2][3] = 0;
                mas3[n3 - 1][3] = 0;
                Fun(op2, mas0, mas1, mas2, mas3);

                Write1(mas1, n1, m1, str1, st1);
                Write2(mas2, n2, m2, st2);
                Write3(mas3, n3, m3, st3);
        }
        for(int i = 0; i < n1; i++)
                        delete [] mas1[i];
        delete [] mas1;

        for(int i = 0; i < n2; i++)
                delete [] mas2[i];
        delete [] mas2;

        for(int i = 0; i < n3; i++)
                delete [] mas3[i];
        delete [] mas3;
        delete [] str1;
        delete [] st1;
        delete [] st2;
        delete [] st3;
        delete [] mas0;

        cout << "Prodolzutj?\n";
        char ch;
        cin >> ch;
        if(ch == 'n')
                flag = false;
}
```

Додаток В.

Графічна візуалізація паралельної часпараметризованої моделі алгоритму Гаусса

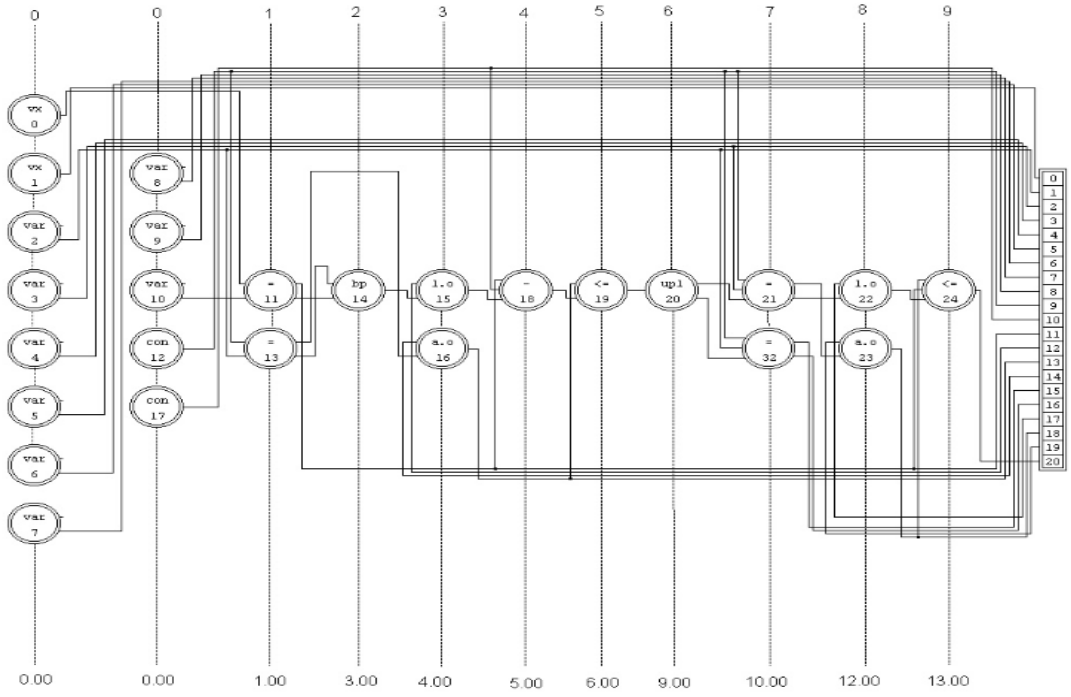


Рис. В.1. Паралельна тимчасова модель розв'язання систем лінійних рівнянь методом Гауса (початок).

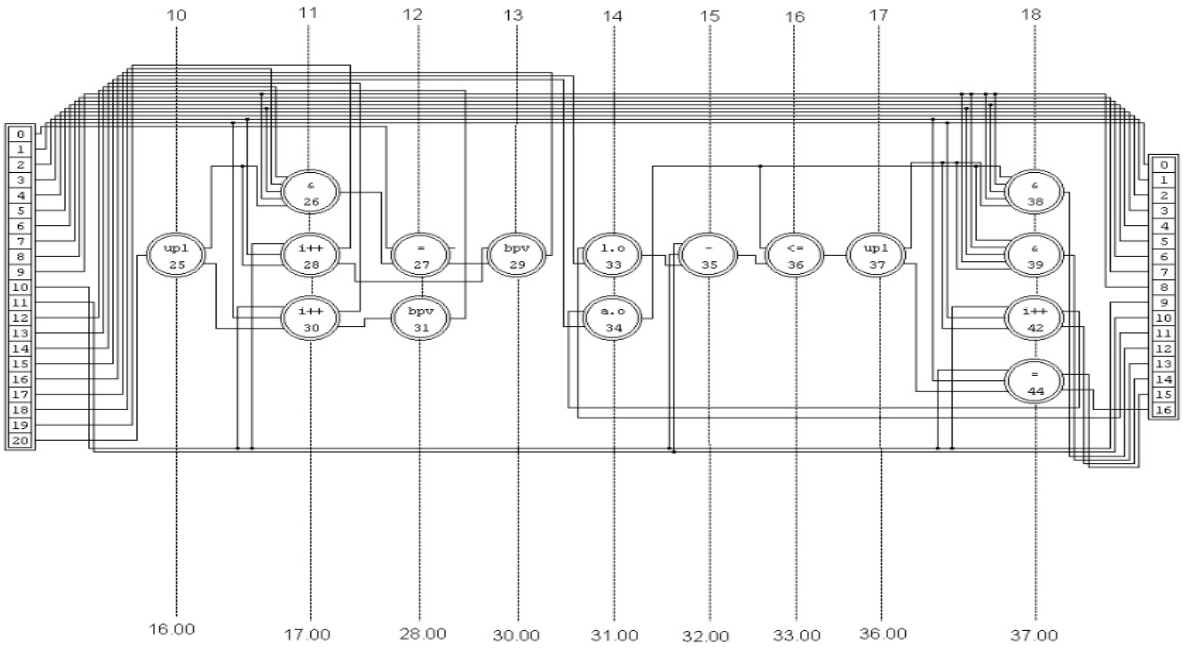


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гауса (продовження 1).

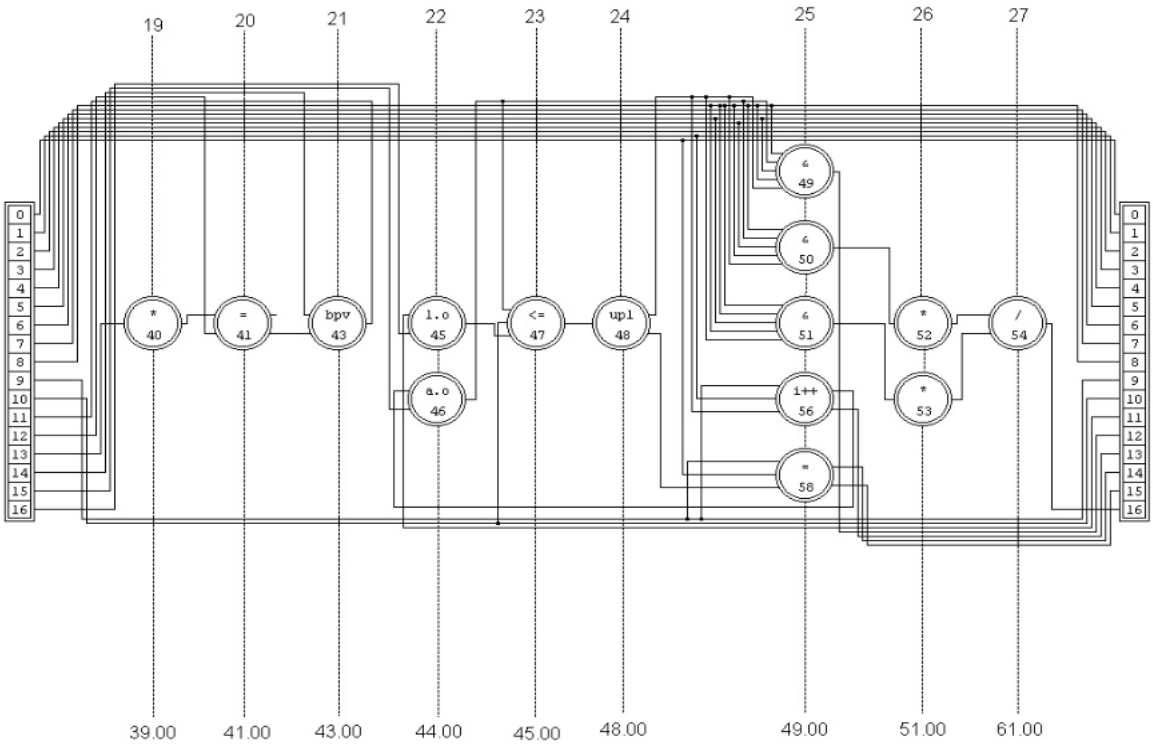


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гауса (продовження 2).

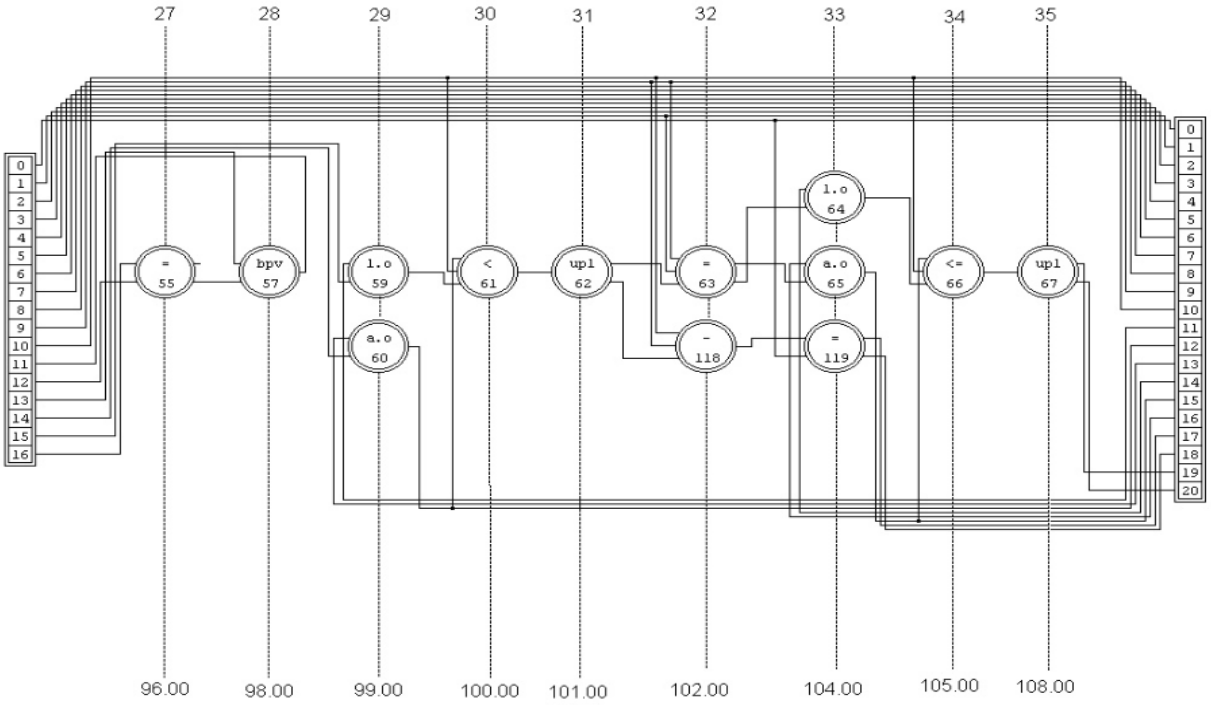


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 3).

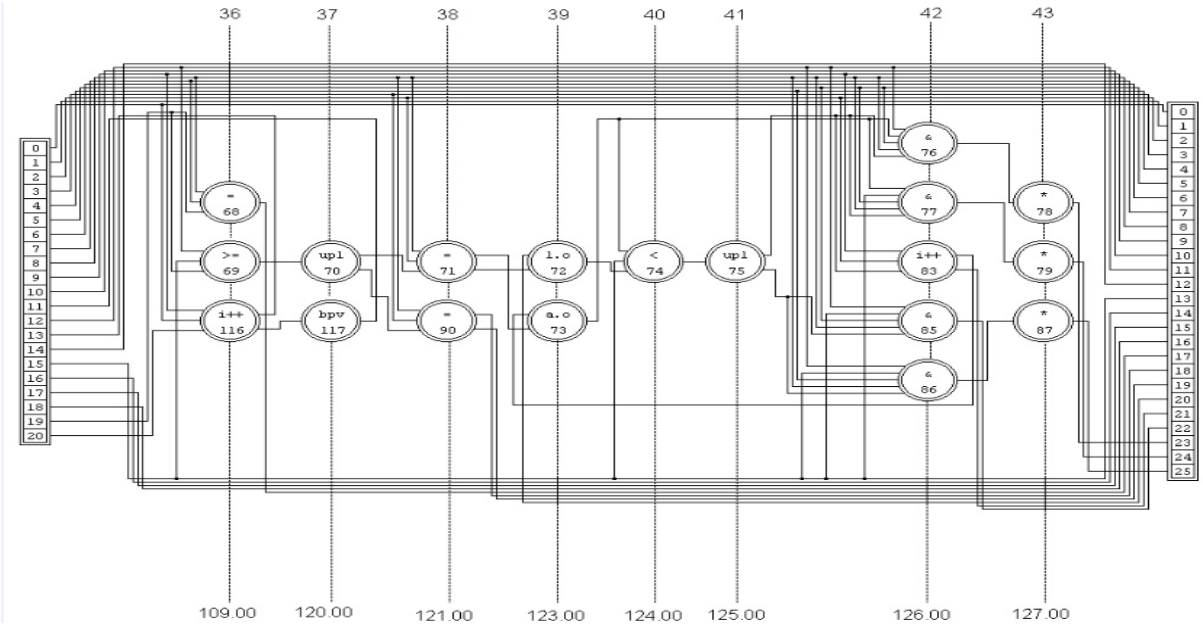


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 4).

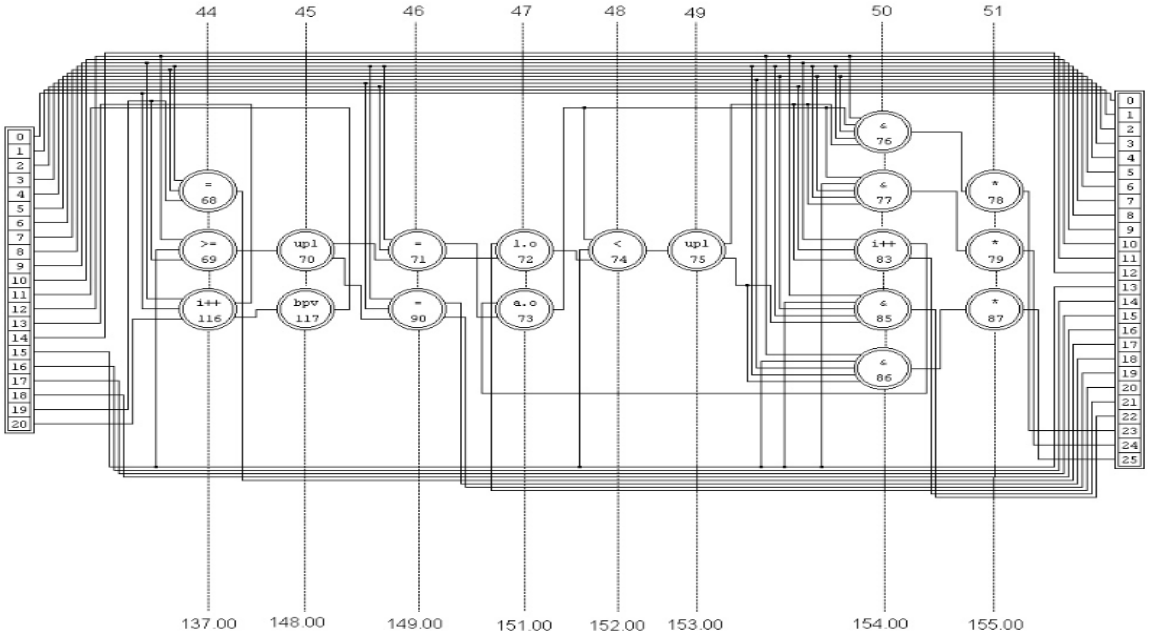


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 5).

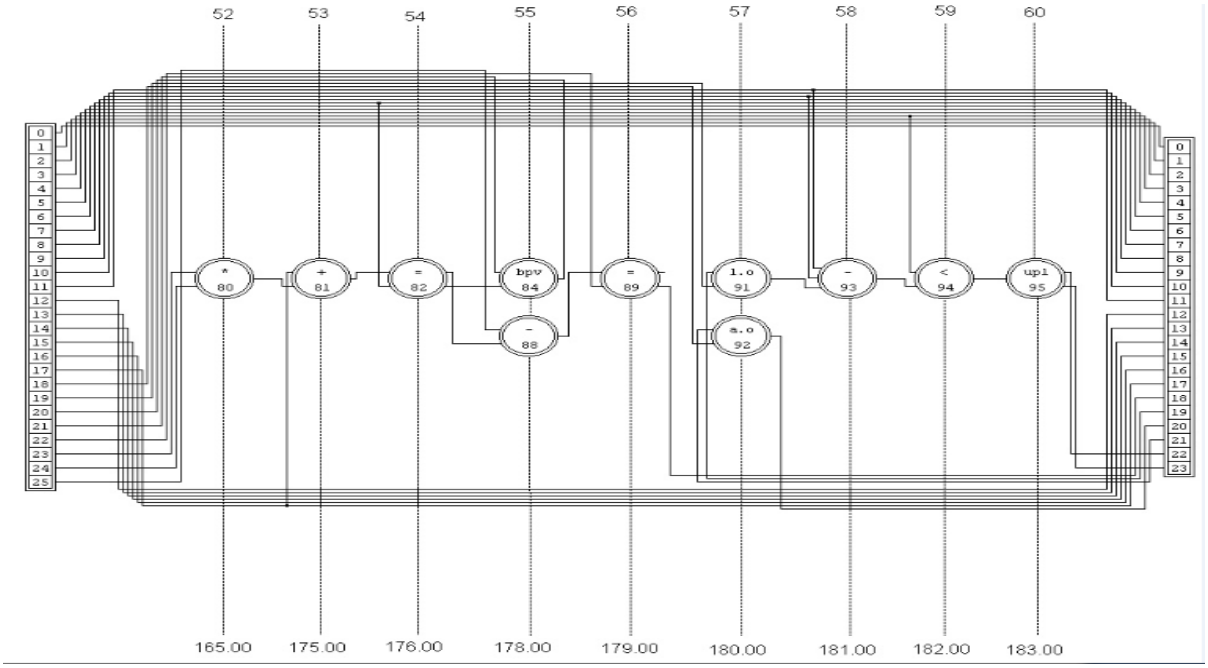


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 6).

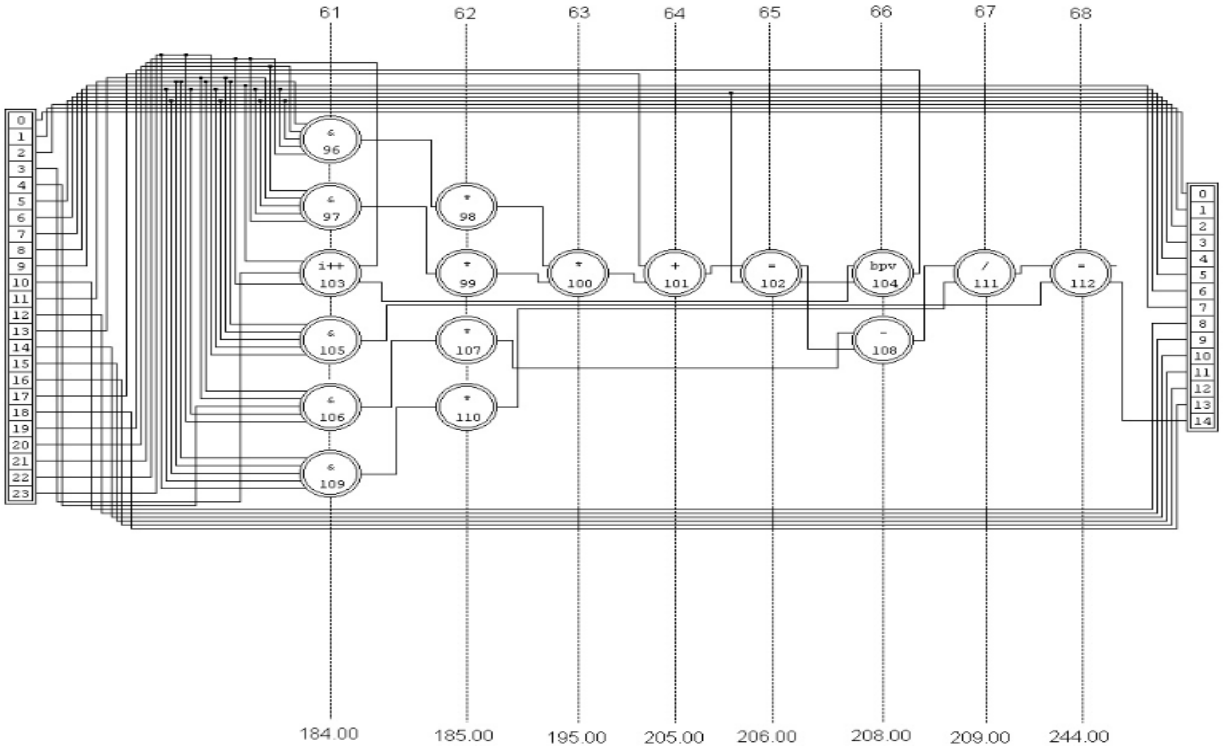


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 7).

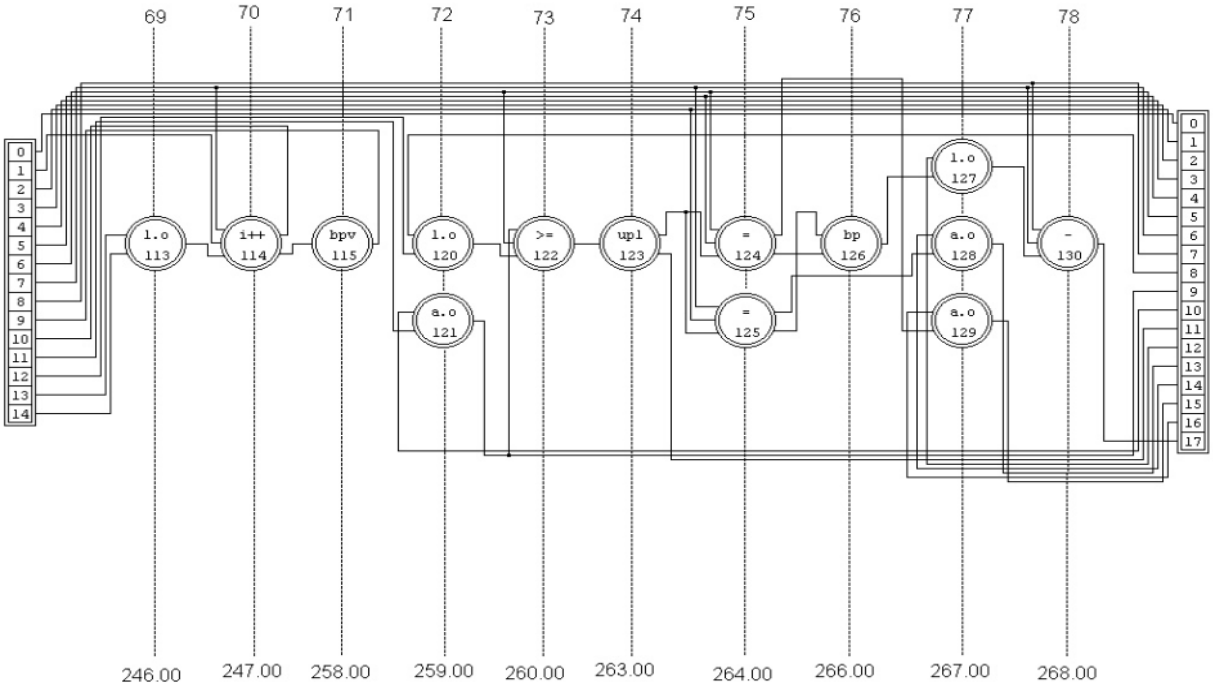


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 8).

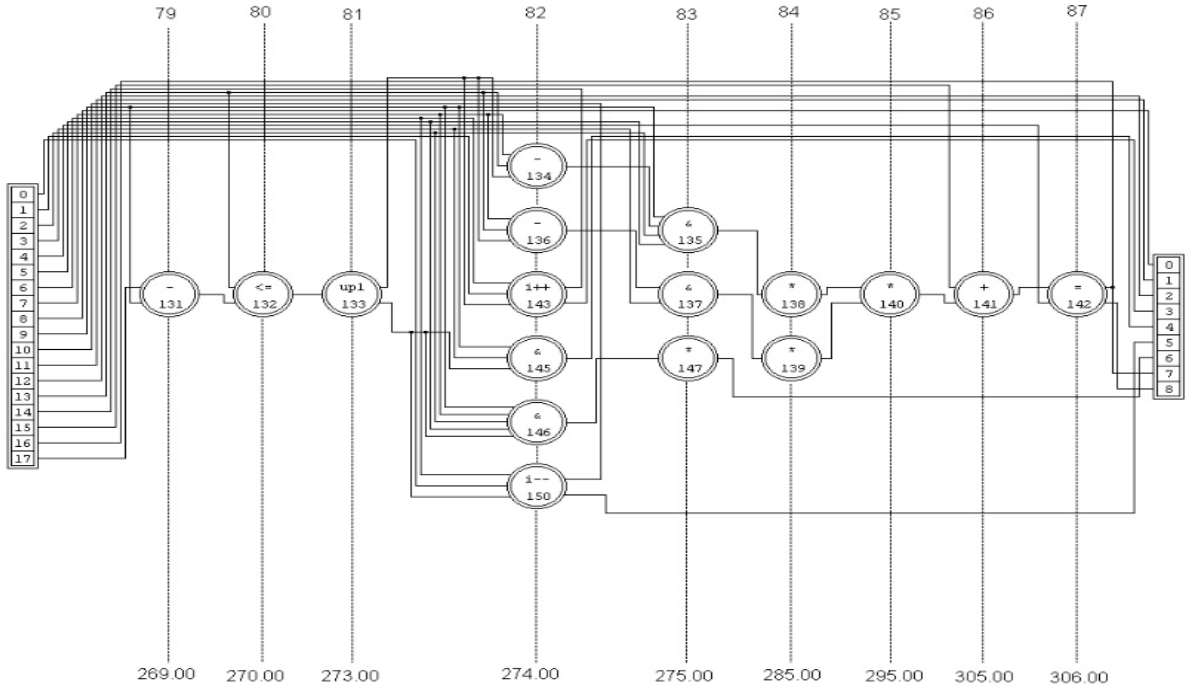


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (продовження 9).

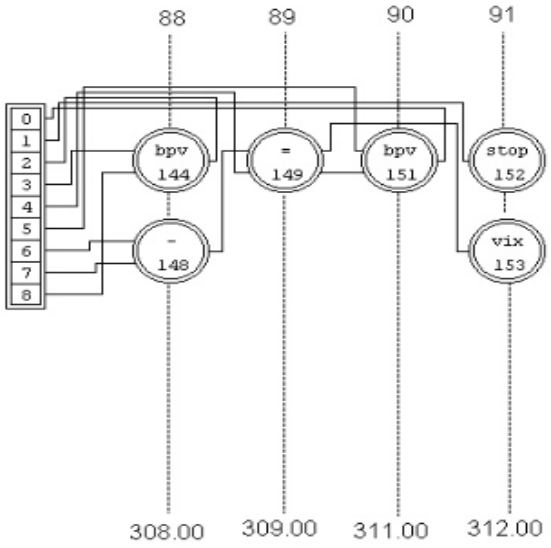


Рис. В.1. Паралельна часпараметризована модель розв'язання систем лінійних рівнянь методом Гаусса (закінчення).

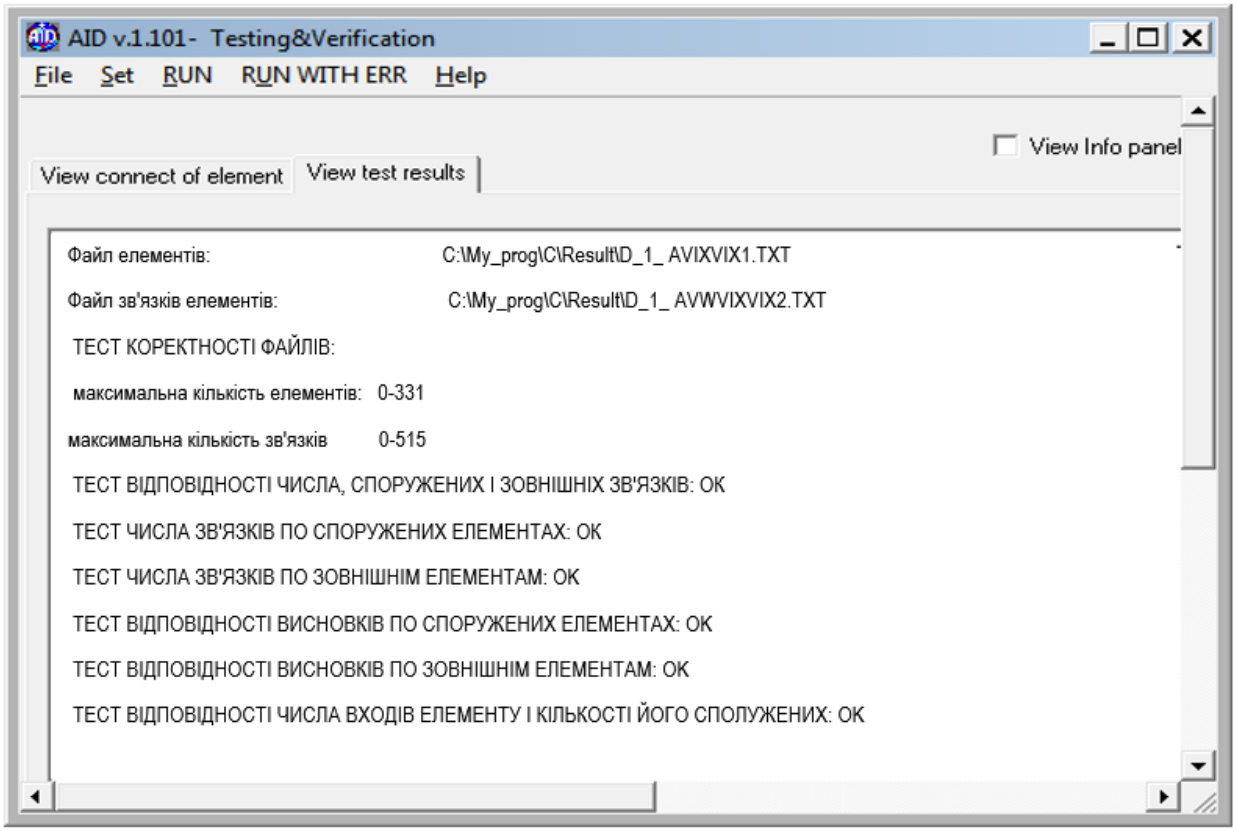


Рис. В.2. Результати верифікації семантико-числових специфікацій вхідної послідовної програми алгоритму Гаусса.

Додаток Г.

Акти про практичне застосування отриманих результатів.

ООО фирма "Л Э Г"
код ЕГРПОУ 14111810
61068, Украина, г. Харьков,
пр. Московский, 179-Б
т./ф.: +38 (057) 703-30-35
тел.: +38 (057) 717-74-33
E-mail: info@leg.ua
http://www.leg.ua

ТОВ фірма "Л Е Г"
код ЄДРПОУ 14111810
61068, Україна, м. Харків,
пр. Московський,179-Б
т./ф.: +38 (057) 703-30-35
тел.: +38 (057) 717-74-33
E-mail: info@leg.ua
http://www.leg.ua

АКТ

про практичне застосування отриманих результатів дисертаційної роботи на здобуття ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки (Галузь знань 12 – Інформаційні технології) Мороз Ольги Юріївни до верифікації програмного забезпечення обладнання виробництва графітових матеріалів

Результати наукових досліджень Мороз Ольги Юріївни, старшого викладача ЗВО кафедри теоретичної та прикладної системотехніки факультету комп'ютерних наук Харківського національного університету імені В. Н. Каразіна впроваджено при верифікації програмного забезпечення обладнання, що використовується для виробництва графітових матеріалів фірмою «ЛЕГ». В результаті збільшився ККД використання пресового обладнання, верстатів, що виготовляють вироби з вуглецевих матеріалів, нагрівачів в високотемпературних печах, які працюють у вакуумі або захисному середовищі, для спікання твердих сплавів, плавки кварцу.

Підрахунок економічного ефекту від результатів впровадження не передбачався.

Головний інженер
ТОВ фірма «ЛЕГ» Руденко О.В.
06.12.2022р.

Код ЄДРПОУ 25468796
61068, м. Харків, пр. Московський, 179-Б
тел./факс: +38(057) 703-30-35, тел. +38(057) 717-74-33
E-mail: gr@leg.ua, br@leg.ua
http://www.leg.ua
Товариство з обмеженою відповідальністю «Науково-технологічний центр «ВУГЛЕЦЬ»

АКТ

щодо впровадження результатів дисертаційної роботи на здобуття ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки (Галузь знань 12 – Інформаційні технології) Мороз Ольги Юріївни

В дисертаційній роботі наведені методи компіляційної, декомпіляційної та семантичної верифікації структур семантико-числової специфікації паралельних часопараметризованих програм, реалізація яких гарантує підвищення ефективності верифікації паралельних програм для інформаційних управляючих систем. Основною відмінністю їх від існуючих методів верифікації паралельних програм є: урахування раціональної сукупності використанихрізних методів паралельної обробки даних та урахування вимог й обмежень різних прикладних областей, що дозволяє підвищити ефективність верифікації паралельного програмного забезпечення інформаційних управляючих систем.

ТОВ «НТЦ «Вуглець» було впроваджено технологію семантико-числової верифікації часопараметризованих мультипаралельних програм для верифікації програмного забезпечення управління роботою теплових вузлів для вакуумних печей (типу ВВНК).

Результати даної роботи представляють інтерес для нашого підприємства. Зазначено, що при використанні верифікації програмного забезпечення вдалося досягти вдосконалення технології виробництва виробів з вуглецевих матеріалів, зменшення часу на розробку окремого виробу та економії електроенергії.

Акт виданий для пред'явлення за місцем захисту дисертації та не є підставою для фінансових розрахунків.

Начальник лабораторії
ТОВ «НТЦ «Вуглець» Костенко О.В.
15.08.2023р.

Онлайн сервіс створення та перевірки кваліфікованого та удосконаленого електронного підпису

ПРОТОКОЛ
створення та перевірки кваліфікованого та удосконаленого електронного підпису

Дата та час: 16:30:04 24.11.2023

Назва файлу з підписом: ДИСЕРТАЦІЯ_PhD_МОРОЗ_О_Ю.pdf.p7s
Розмір файлу з підписом: 5.9 МБ

Перевірені файли:
Назва файлу без підпису: ДИСЕРТАЦІЯ_PhD_МОРОЗ_О_Ю.pdf
Розмір файлу без підпису: 5.9 МБ

Результат перевірки підпису: Підпис створено та перевірено успішно. Цілісність даних підтверджено

Підписувач: МОРОЗ ОЛЬГА ЮРІЇВНА
П.І.Б.: МОРОЗ ОЛЬГА ЮРІЇВНА
Країна: Україна
РНОКПП: 2689900940
Організація (установа): ФІЗИЧНА ОСОБА
Час підпису (підтверджено кваліфікованою позначкою часу для підпису від Надавача): 16:30:03 24.11.2023
Сертифікат виданий: КНЕДП АЦСК АТ КБ "ПРИВАТБАНК"
Серійний номер: 5E984D526F82F38F04000000E39C320148E09804
Алгоритм підпису: ДСТУ-4145
Тип підпису: Удосконалений
Тип контейнера: Підпис та дані в CMS-файлі (CAdES)
Формат підпису: З повними даними ЦСК для перевірки (CAdES-X Long)
Сертифікат: Кваліфікований