МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Кваліфікаційна наукова
праця на правах рукопису

ПАНАРІН АРТЕМ СЕРГІЙОВИЧ

УДК 004.052: 004.054

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ЗАБЕЗПЕЧЕННЯ ФУНКЦІЙНОЇ БЕЗПЕЧНОСТІ ІНФРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ АЕС НА ПРОГРАМОВНИХ ПЛАТФОРМАХ З ВИКОРИСТАННЯМ КОМБІНОВАНОЇ ДИВЕСРНОСТІ

Спеціальність 123 Комп’ютерна інженерія
Галузь знань 12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ А. С. Панарін

Науковий керівник: Харченко Вячеслав Сергійович, член-кореспондент Національної академії наук України, д.т.н., професор

Харків – 2026

АНОТАЦІЯ

Панарін Артем Сергійович. Методи і засоби забезпечення функційної безпечності інформаційно-керуючих систем АЕС на програмовних платформах з використанням комбінованої диверсності – Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія». – Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, 2026 р.

Дисертаційну роботу присвячено підвищенню функційної безпечності багатоверсійних інформаційно-керуючих систем атомних електростанцій, реалізованих на програмовних платформах типу FPGA та SoC, за рахунок застосування комбінованої диверсності та модельно-базованої верифікації. У роботі враховано вимоги міжнародних і національних стандартів (IEC 61513, IEC 60880, IEC 62138, IEC 61508, рекомендації МАГАТЕ та нормативні документи України) щодо проєктування і експлуатації систем, важливих для безпеки АЕС, а також специфіку відмов за загальною причиною для інфокомунікаційної та програмно-апаратної складових ІКС.

Об'єктом дослідження є процеси забезпечення функційної безпечності багатоверсійних інформаційно-керуючих систем АЕС на програмовних платформах. Предметом дослідження є моделі, методи та засоби кількісної оцінки, синтезу комбінованої диверсності та модельно-базованої верифікації двоверсійних архітектур систем безпеки, орієнтованих на зниження ризику відмов за загальною причиною.

Отримало подальшого розвитку графова модель опису різних програмно-апаратних версій для інформаційно-керуючих систем на програмовних платформах, з використанням деталізованого класифікатора видів диверсності, а саме: процесорних ядер, інтерфейсів, засобів синхронізації, технологічних процесів і обладнання, що дозволяє оцінювати метрики диверсності і складності залежно від множини видів і підвидів версійної надмірності.

Розроблено метод вибору видів диверсності для програмовних платформ інформаційно-керуючих систем, який ґрунтується на експертному оцінюванні метрик ефективності диверсності та вартості, а також використанні алгоритмів пошуку шляхів і їх комбінацій у графовій моделі. Запропоновано стратегії та алгоритми вибору диверсності за різними критеріями (максимізація диверсності, мінімізація витрат, компроміс «диверсність/вартість»), що дозволяє адаптувати метод до конкретних проєктних обмежень і цільових показників функційної безпечності.

Удосконалено методи реалізації програмно-апаратної диверсності програмовних платформ через розроблення: методу диверсної синхронізації каналів на основі варіації параметрів тактування та часових зсувів; методу структурно-просторової диверсності матриць логічних елементів і таймінгів сигналів; підходів до диверсифікації засобів і процедур обчислення контрольних сум, організації передачі даних та контролю цілісності пам’яті. Це забезпечує зниження чутливості багатоверсійних рішень до відмов за загальною причиною, викликаних спільними апаратними дефектами, електромагнітними завадами, помилками конфігурації або вразливостями засобів діагностики.

Вперше запропоновано метод модельно-базованої верифікації систем програмовної логіки для інформаційно-керуючих систем АЕС, що інтегрує модельно-орієнтоване тестування (Model-Based Testing) з аналізом сценаріїв відмов та специфічних властивостей безпеки. Визначено послідовність етапів розроблення та верифікації двоверсійних систем з використанням формальних моделей поведінки, генерації тестів, імплементації компонентів МВТ у апаратній реалізації та аналізу отриманих результатів. Це дає змогу підвищити повноту перевірки, виявляти комбінаційні помилки на рівні FPGA та ПЗ і оцінювати вплив виявлених відмов на функційні властивості.

Практичне значення отриманих результатів полягає в можливості їх застосування під час проєктування та модернізації систем безпеки АЕС на базі платформ Radiy, RadICS та аналогічних рішень, де необхідно забезпечити високі вимоги до функційної безпечності та відмовостійкості. Розроблені моделі, методи

та рекомендації можуть використовуватись для систематизації та вибору диверсних стратегій, побудови технологічних процесів верифікації, підготовки нормативних і методичних документів, а також у навчальному процесі при підготовці фахівців з комп'ютерної інженерії та ядерної безпеки.

Ключові слова: інформаційно-керуючі системи АЕС, комп'ютерні системи та мережі, інтернет речей, FPGA платформи, надійність, кібербезпека і функційна безпечність, зміна і оцінка вимог, розроблення моделей, відмова за загальною причиною, граф диверсності, патерни проєктування, модельно-базована верифікація, програмні утиліти.

Список публікацій здобувача:

Скляр В. В., Харченко В. С., Панарин А. С., Сандер И. Применение концепции Model-Based Testing для верификации систем на базе IP-ядер. Радіоелектронні і комп'ютерні системи. 2010. № 5. С. 237–241.

Sklyar V., Siora O., Herasimenko O., Panarin A. Development and Verification of Dependable Multiversion Systems on the basis of IP-Cores. Technical Approach to Dependability. Wroclaw, 2010. P. 133–145. URL: https://www.researchgate.net/profile/Vladimir-Sklyar/publication/305773007_Development_and_Verification_of_Dependable_Multiversion_Systems_on_the_basis_of_IP-Cores/links/57a0b39c08aeef8f311c52f5/Development-and-Verification-of-Dependable-Multiversion-Systems-on-the-basis-of-IP-Cores.pdf (дата звернення: 25.01.2026).

Панарин А. С. Имитационное моделирование soft-процессоров на базе концепции Model-Based Testing. Радіоелектронні і комп'ютерні системи. 2012. № 5. С. 100–106.

Скляр В. В., Харченко В. С., Панарин А. С. Тестирование и разработка диверсных программируемых логических контроллеров на базе ПЛИС с использованием среды функционального программирования. Радіоелектронні і комп'ютерні системи. 2014. № 1. С. 29–41.

Скляр В. В., Харченко В. С., Панарин А. С. Тестирование программируемых логических контроллеров на базе ПЛИС с использованием

среды функционального программирования. Системи обробки інформації. 2014. № 1. С. 44–55.

Panarin A., Kharchenko V., Zemlianko H. Advanced classifier for expert assessment of diversity and cost of NPP information and control systems. Measuring Equipment and Metrology. 2025. Vol. 86, № 4. P. 52–64.

Бабешко Є., Панарін А. Методи та засоби диверсної синхронізації та реалізації електронних проєктів для систем безпеки АЕС на FPGA платформах. Вимірювальна та обчислювальна техніка в технологічних процесах. 2026. Т. 1. С. 32–38. DOI: 10.31891/2219-9365-2026-85-5.

Babeshko E., Kovalenko A., Illiashenko O., Panarin A., Sklyar V. et al. Secure and resilient computing for industry and human domains. Vol. 2. Kharkiv: National Aerospace University named after N. E. Zhukovsky “KhAI”, 2017. 762 p. URL: http://serein.eu.org/wp-content/uploads/2017/10/Volume-2.-Secure-and-resilient-systems-and-infrastructures.pdf

Панарін А., Харченко В., Землянко Г. Розроблення розширеного класифікатору для експертного оцінювання диверсності та вартості інформаційно-керуючих систем АЕС. Вимірювальна техніка та метрологія. 2025. Т. 86, № 4. С. 52. DOI: 10.23939/istcmtm2025.04.052.

Панарін А. Принцип диверсності та багатоверсійні технології для систем захисту реакторів атомних електростанцій. Пропілеї права та безпеки. 2026. Vol. 8. P. 260–263. DOI: 10.32620/pls.2025.8.67.

Babeshko I., Duzhiy V., Panarin A., Illiashenko O., Siora A. et al. Diversity for NPP I&C Systems Safety and Cyber Security. Cyber Security and Safety of Nuclear Power Plant Instrumentation and Control Systems. IGI Global, 2020. P. 239–288. URL: https://www.igi-global.com/chapter/npp-ic-systems/258682 (дата звернення: 25.01.2026).

Kharchenko V., Ponochovnyi Y., Babeshko E., Ruchkov E., Panarin A. Safety Assessment of Maintained Control Systems with Cascade Two-Version 2oo3/1oo2 Structures Considering Version Faults. 18th DepCoS-RELCOMEX. 2023. Vol. 737. P. 119–129. DOI: 10.1007/978-3-031-37720-4_11.

Panarin A., Sklyar V., Kharchenko V., Kovalenko A., Babeshko E. Modeling of industrial FPGA-based controllers with ForSyDe. 2017 International Conference on Information and Digital Technologies (IDT). IEEE, 2017. P. 164–168. DOI: 10.1109/DT.2017.8024290.

Illiashenko O., Kharchenko V., Kor A.-L., Panarin A., Sklyar V. Hardware diversity and modified NUREG/CR-7007 based assessment of NPP I&C safety. 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). IEEE, 2017. Vol. 2. P. 907–911. DOI: 10.1109/IDAACS.2017.8095218.

Duzhyi V., Kharchenko V., Panarin A., Rusin D. Diversity metric evaluation considering extended NUREG-7007 diversity classification. 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). IEEE, 2018. P. 21–25. DOI: 10.1109/DESSERT.2018.8409092.

Kharchenko V., Kovalenko A., Leontiiev K., Panarin A., Duzhy V. Multi-diversity for FPGA platform based NPP I&C systems: new possibilities and assessment technique. International Conference on Nuclear Engineering. London, UK: American Society of Mechanical Engineers, 2018. Vol. 1. P. 1–9. DOI: 10.1115/ICONE26-82377.

ABSTRACT

Artem Panarin. Methods and means for ensuring functional safety of NPP information and control systems on programmable platforms using combined diversity – Manuscript copyright. Dissertation for the degree of Doctor of Philosophy in specialty 123 Computer Engineering. – National Aerospace University "Kharkiv Aviation Institute", Kharkiv, 2025.

The dissertation is devoted to improving the functional safety of multi-version information and control systems (ICS) of nuclear power plants implemented on FPGA and SoC programmable platforms by applying combined diversity and model-based verification. The research takes into account international and national standards (IEC 61513, IEC 60880, IEC 62138, IEC 61508, IAEA recommendations and Ukrainian regulations) for the design and operation of safety-important systems, as well as the specifics of common cause failures (CCF) for ICS software–hardware components.

The object of research is the processes of ensuring functional safety of multi-version NPP information and control systems on programmable platforms. The subject of research is models, methods and tools for quantitative assessment, synthesis of combined diversity and model-based verification of two-version and multi-version safety architectures aimed at reducing the risk of common cause failures.

A refined multi-level classification of diversity types and subtypes for programmable platform solutions in NPP ICS is proposed, covering architectural, hardware, software–algorithmic and operational levels and enabling construction of combined diversity strategies with regard to safety and cost constraints. Graph and matrix models of diversity are developed, where diversity types are represented as vertices and paths with associated diversity and relative cost metrics, which provides a formalized basis for selecting rational combinations of diversity techniques.

A method for selecting diversity types for programmable platforms of information and control systems is developed, based on expert evaluation of diversity effectiveness and cost metrics and the use of path search algorithms and their combinations in the graph model. Strategies and algorithms for diversity selection under

different criteria (maximization of diversity, minimization of cost, diversity–cost trade-off) are proposed, which makes it possible to adapt the method to specific design constraints and target functional safety indicators.

Methods for implementing software–hardware diversity of programmable platforms are improved through the development of: a method of diverse channel synchronization using variation of clocking parameters and time shifts; a method of structural–spatial diversity of logic element matrices and signal timing; and approaches to diversification of checksum computation means and procedures, data transmission organization and memory integrity control. These solutions reduce the sensitivity of multi-version architectures to CCFs caused by shared hardware defects, electromagnetic interference, configuration errors or vulnerabilities of diagnostic means.

A method of model-based verification for programmable logic systems in NPP ICS is proposed, integrating model-based testing with the analysis of failure scenarios and safety properties. A sequence of stages for development and verification of two-version systems is defined, including construction of behavioural models, test generation, implementation of model-based testing components in hardware, and analysis of obtained results. This enables higher verification completeness, detection of combinational errors at FPGA and software levels and assessment of the impact of detected failures on functional properties.

The practical significance of the results lies in their applicability to the design and modernization of NPP safety systems based on Radiy, RadICS and similar platforms under stringent functional safety and dependability requirements. The proposed models, methods and recommendations can be used to systematize and select diversity strategies, to build verification workflows, to support normative and methodological documents, and in educational processes for training specialists in computer engineering and nuclear safety.

Keywords: NPP information and control systems, computer systems and networks, Internet of Things, FPGA, reliability, cybersecurity and functional safety, requirements change and assessment, model development, common-cause failure, diversity graph, design patterns, model-based verification, software tools.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CCF - Common cause failure

CDC - Clock Domain Crossing (перетин доменів тактування).

CPU – Central Processor Unit

DMA – Direct Memory Access

ECC – Error-correcting Codes

EDA - Інструментальні засоби проєктування

EMI – Electromagnetic interference

FEC – Forward Error Correction

FIFO – First Input First Output

FMEA - Failure Modes and Effects Analysis

ForSyDe – Formal System Design

FPGA - Field-Programmable Gate Array

FTA - Fault Tree Analysis

HAZOP - Hazard and Operability Study

IP-core (IP-ядро) - Intellectual Property Core

IV&V - Independent Verification & Validation

MBT - Model-Based Testing

MCU - Microcontroller Unit

PHY – Physical layer protocol

PLD - Programmable Logic Device

RXD – Receive Data

SIL - Safety Integrity Level

SoC – System on Chip

TXD – Transmit Data

АЕС - Атомна електростанція

ВЗЗП - Відмова за загальною причиною

ГКД - Галузеві керівні документи

ДНАОП - Державні нормативні акти з охорони праці

ІКС - Інформаційно-керуюча система

КС – контрольна сума

НПАОП - Нормативні правові акти з охорони праці

ОС – Операційна Система

ПЗ – Програмне забезпечення

ПЗП – Постійний запам'ятовуючий пристрій

ПЛІС - Програмовна логічна інтегральна схема

САЗ - Система аварійного захисту

СПЗ - Система попереджувального захисту

ФБ - Функційна безпечність

ЦП – Центральний процесор

NPP – Nuclear Power Plant

IoT – Internet of Things

IDE – Integrated Development Environment

SEU – Single Event Upset

ВСТУП

Обґрунтування вибору теми дослідження. Розвиток атомної енергетики, зокрема впровадження нових технологій у системи управління та захисту, зумовлює необхідність підвищення вимог до функційної безпечності та надійності інформаційно-керуючих систем (ІКС) атомних електростанцій (АЕС). В умовах зростання складності архітектур ІКС, а також впровадження програмовних логічних інтегральних схем (FPGA), особливої актуальності набуває використання багатоверсійних підходів, спрямованих на зменшення впливу відмов за загальною причиною. Теорію і практику багатоверсійних систем і технологій для безпекових застосувань розвивали українські науковці Брежнєв Є. В., Горбенко А. В., Заславський В. А., Конорев Б. М., Краснобаєв В. А., Одарущенко О. М., Скляр В. В., Харченко В.С., Ястребенецький М.О. та ін.. Визначальний внесок серед закордонних дослідників зробили А. Авіженіс, Р. Вуд, Ж.-К. Лапрі, Б. Літлвуд, П. Попов, А. Романовські, Ф. Саглієтті та ін.

Одним із перспективних напрямів у цьому контексті є впровадження принципу диверсифікації на різних рівнях інформаційно-керувальних систем — від вибору типу й архітектури інтегральної схеми до методів розроблення програмного забезпечення. Разом з тим, розроблення та оцінювання таких рішень вимагає комплексного підходу, що поєднує аналіз вимог міжнародних і національних стандартів, моделювання відмов, методи модельно-орієнтованого тестування (MBT Model-Based Testing), а також методи та засоби підвищення функційної безпечності з урахуванням можливостей створення багатоверсійних ІКС на програмовних платформах.

Об’єкт дослідження. Процеси аналізу, розроблення та забезпечення функційної безпечності ІКС АЕС на програмовних платформах.

Предмет дослідження. Моделі, методи та алгоритми кількісного оцінювання , розроблення, верифікації та забезпечення функційної безпечності ІКС АЕС на програмовних платформах з використанням комбінованої диверсності.

Мета і завдання дослідження. Мета дослідження полягає у підвищенні функційної безпечності багатоверсійних ІКС АЕС на програмовних платформах шляхом розроблення методів і засобів комбінованої версійної надмірності, оцінювання, вибору, верифікації та реалізації різних видів диверсності.

Основні завдання дослідження:

– проаналізувати вимоги міжнародних і національних стандартів до функційної безпечності ІКС АЕС, методи аналізу, оцінювання та розроблення багатоверсійних систем;

– виконати класифікацію та аналіз відмов ІКС, зокрема, відмов за загальною причиною, удосконалити принцип диверсності та класифікаційні схеми її видів з урахуванням особливостей програмовних платформ ІКС;

– розробити формальні моделі представлення видів і підвидів диверсності для побудови багатоверсійних систем, а також оцінювання показників безпечності;

– запропонувати методи оцінювання диверсності та функційної безпечності, які враховують додаткові види і підвиди диверсності для їх вибору з урахуванням вимог до функційної безпечності та обмежень вартості;

– розробити методи підвищення функційної безпечності ІКС на програмовних платформах з урахуванням нових видів програмно-апаратної диверсності на рівні електронних проєктів;

– запропонувати метод верифікації програмовних рішень для ІКС з використанням модельно-базованого тестування та диверсності процесів верифікації;

– розробити рекомендації та інструменти для підвищення функційної безпечності на основі наукових результатів та виконати їх впровадження.

Методи дослідження. У роботі використано:

– теоретичні методи – системний аналіз, теорія надійності, теорія множин та графів;

– математичні методи – розроблення метрик, оптимізаційні алгоритми, статистичний аналіз даних;

– експериментальні методи – моделювання та тестування на FPGA-платформах, апаратні випробування в умовах, наближених до експлуатаційних;

– нормативно-аналітичні методи – аналіз стандартів NUREG, IEC, IAEA, NRC, адаптація вимог до вітчизняних умов.

Наукова новизна отриманих результатів:

– **отримало подальшого розвитку графова модель** опису різних програмно-апаратних версій для інформаційно-керуючих систем на програмовних платформах, з використанням деталізованого класифікатора видів диверсності, а саме: процесорних ядер, інтерфейсів, засобів синхронізації, технологічних процесів і обладнання, що дозволяє оцінювати метрики диверсності і складності залежно від множини видів і підвидів версійної надмірності.

– **удосконалено метод вибору видів диверсності** з урахуванням багаторівневої ієрархії видів версійної надмірності, включно з внутрішньою диверсністю каналів, а також відповідні метрики диверсності і складності, що дозволяє підвищити функційну безпечність і обґрунтувати структуру та процеси створення двохверсійних систем аварійного захисту реакторів АЕС;

– **удосконалено методи програмно-апаратної диверсності**, які базуються на процедурах функційно-просторової диверсифікації засобів шинної організації, контролю цілісності даних, налаштувань проєкту та синхронізації з керованими зсувами, що забезпечує підвищення енергоефективності та/або безпечності завдяки зменшенню ризиків завад та відмов спільних логічних елементів версій.

– в**перше запропоновано метод модельно-базованого тестування систем програмовної логіки**, який, на відміну від відомих, грунтується на порівнянні версій, отриманих з використанням мови програмування Haskel та її конвертованої моделі у VHDL-код, а також еталонного проєкту, розробленого за технологією SoC, що зменшує ризики залишкових прихованих дефектів.

Особистий внесок здобувача. Усі наукові результати, наведені в дисертації, отримані автором особисто. В роботах, опублікованих в соавторстві, автором було запропоновано: розширену класифікаційну схему і графову модель представлення видів і підвидів диверсності з урахуванням FPGA технології;

постановку задачі, метрики диверсності й вартості та метод вибору варіантів версійної надмірності з урахуванням критерія «безпека-вартість»; методи та проагрмно-апаратні рішення для побудови двохверсійних систем (диверсні варіанти синхронізація, формування контрольної суми тощо); принципи диверсної верифікації та модельно-базованого тестування програмовних платформних рішень з використанням мов Haskel, VHDL та технології SoC. Базові ідеї, постановка задач, розроблення моделей, алгоритмів і методів оцінки різнорідності, а також програмно-апаратних рішень реалізовані автором самостійно, перевірено в процесі розробки реальних ІКС на програмовних платформах.

Апробація матеріалів дисертації. Основні результати дослідження доповідалися та обговорювалися на міжнародних та всеукраїнських науково-технічних конференціях, семінарах і круглих столах, зокрема:

“Dependable Systems, Services and Technologies” (м. Кропивницький, 2010; м. Севастопіль, 2012; м. Київ, 2018); “Разработка и тестирование диверсных проектов ПЛИС на базе soft-процессоров для ИУС, важных для безопасности АЭС” (Науковий семінар КриКТехС, м. Харків, 2009); “Conference on Dependability of Computer Systems DepCoS-RELCOMEX” (м. Брунов, Польща, 2017, 2023); “Information and Digital Technologies” (м. Жиліна, Словакія, 2017); “International Conference on Nuclear Engineering” (м. Лондон, Англія, 2018); “IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications” (м. Будапешт, Румунія, 2017); “International Conference on Nuclear Engineering” (м. Лондон, Англія, 2018); “Методи та засоби забезпечення безпеки систем аварійного захисту АЕС з використанням програмно-апаратної диверсності” (НТС КриКТехС, м. Харків, 2020, 2025);

Зв’язок з науковими програмами, темами. Дисертаційна робота виконана у Національному аерокосмічному університеті «Харківський авіаційний інститут» відповідно з державними програмами та планами НДР, а також проєктами розроблення та впровадження ІКС АЕС, зокрема, систем аварійного захисту реакторів, на науково-виробничих підприємствах Радій та Радікс відповідно до

державних програм модернізації атомної енергетичної галузі впродовж 2012-2026 років, а також з програмами сертифікації програмовних платформ на відповідність вимогам стандарту IEC 61508 та критеріям US NRC.

Практичне значення отриманих результатів. Розроблені методи оцінки та синтезу диверсних архітектур використано: при проєктуванні нових і модернізації існуючих систем безпеки АЕС; для підвищення стійкості до Common Cause Failure (CCF) у критичних інфраструктурах; у навчальному процесі підготовки фахівців з безпеки та надійності комп'ютерних систем.

Структура та обсяг дисертації. Дисертація складається зі вступу, чотирьох розділів, висновку, списку використаних джерел і додатків. Загальний обсяг становить 230 сторінок, з яких анотація двома мовами на 7 сторінках, зміст на 4 сторінках, список використаних джерел із 138 найменувань на 18 сторінках, додатки на 9 сторінках. Робота містить 46 рисунків, 12 таблиць та 32 формули.

Публікації. За темою дисертаційної роботи було опубліковано 16 наукових публікацій, включно:

- 8 статей у наукових фахових виданнях України (категорія «Б»);

- 1 стаття опублікована у періодичному виданні Springer (має ISSN та DOI з індексацією у Scopus, квартиль Q4);

- 3 колективних монографії (1 колективна монографія має індексацію в Scopus);

- 4 публікації у просідінгах міжнародних конференцій з індексацією у Scopus.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ОЦІНЮВАННЯ ТА ЗАБЕЗПЕЧЕННЯ ФУНКЦІЙНОЇ БЕЗПЕЧНОСТІ БАГАТОВЕРСІЙНИХ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ АЕС. ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕНЬ

1.1 Аналіз вимог до надійності та функційної безпечності інформаційно-керуючих систем АЕС

Сучасні атомні електростанції є складними технологічними комплексами, що потребують застосування високонадійних і безпечних інформаційно-керуючих систем (ІКС). Ці системи повинні не лише виконувати свої функції за нормальних умов експлуатації, але й забезпечувати гарантовану роботу в умовах аварійних ситуацій, мінімізуючи ризик для людей, навколишнього середовища та обладнання.

Функційна безпечність (ФБ) у контексті ІКС АЕС розглядається як властивість системи виконувати свої безпечні функції навіть у випадку виникнення певних відмов або зовнішніх впливів. У нормативних документах міжнародного та національного рівня функційна безпечність визначається як результат комплексної взаємодії трьох основних компонентів: технічних рішень, організаційних заходів та процедур експлуатації.

В умовах атомної енергетики вимоги до ФБ та надійності ІКС значно перевищують аналогічні вимоги для промислових автоматизованих систем. Це обумовлено такими чинниками:

- потенційною катастрофічністю наслідків відмови;
- наявністю специфічних сценаріїв відмов (включно з відмовами за загальною причиною);
- впливом жорстких умов експлуатації (високі температури, радіаційне випромінювання, електромагнітні завади тощо);
- високим рівнем регламентованості діяльності у сфері атомної енергетики.

Вимоги до ІКС АЕС [1–10] формуються з урахуванням міжнародних стандартів (наприклад, IEC 61513, IEC 60880, IEC 62138, IEC 61508) [11–14] та національних нормативних актів (ДНАОП, НПАОП, НП 306.2.202-2015, НП 306.2.141-2008 тощо).

Вимоги міжнародних стандартів слугують базою для розроблення національних регламентів, які, у свою чергу, уточнюють і адаптують загальні принципи та підходи під умови конкретної країни. Такий взаємозв'язок між міжнародною та національною нормативною базою забезпечує узгодженість вимог до функційної безпечності ІКС АЕС і дозволяє враховувати як світовий досвід, так і особливості локальної інфраструктури, законодавства та технічних ресурсів.

1.1.1 Вимоги міжнародних і національних стандартів

Міжнародні та національні стандарти, що регламентують функційну безпечність інформаційно-керуючих систем (ІКС) атомних електростанцій (АЕС), формують основу для всіх етапів життєвого циклу цих систем — від проєктування та розроблення до експлуатації і виведення з експлуатації. Їх мета — мінімізувати ймовірність відмов, зокрема відмов за загальною причиною, і забезпечити здатність системи виконувати функції безпеки навіть у разі виникнення аварійних ситуацій.

Найбільш вагомими у цій сфері є стандарти Міжнародної електротехнічної комісії (IEC) та Міжнародного агентства з атомної енергії (МАГАТЕ):

1. IEC 61513 — "Nuclear power plants – Instrumentation and control important to safety – General requirements for systems". Стандарт IEC 61513 є ключовим документом, що визначає загальні вимоги до проєктування, впровадження, експлуатації та технічного обслуговування систем, важливих для безпеки атомних електростанцій. Його основою є концепція defense-in-depth (багаторівневого захисту), відповідно до якої в системі закладається кілька незалежних рівнів безпеки, що дозволяє мінімізувати ймовірність небезпечних

відмов навіть у випадку виходу з ладу одного з рівнів. Кожен рівень виконує власну захисну незалежну від інших функцію, що забезпечує стійкість системи до комбінаційних відмов [15; 16].

IEC 61513 також визначає чітке функціональне розмежування між системами безпеки, системами підтримки безпеки та системами, не пов’язаними з безпекою. Це запобігає перехресному впливу помилок або відмов між різними класами систем. Стандарт регламентує управління конфігурацією, включно з детальним контролем змін протягом усього життєвого циклу, що особливо важливо для ядерної енергетики, де будь-яке оновлення чи модернізація системи має проходити багаторівневу перевірку та затвердження. На практиці вимоги IEC 61513 реалізуються через проєктування незалежних каналів управління реактором, дублювання датчиків контролю критичних параметрів та резервування засобів аварійного зупину реактора. Такі підходи значно підвищують стійкість ІКС до відмов за загальною причиною.

2. IEC 60880 — "Software for computers in the safety systems of nuclear power plants". Стандарт визначає вимоги до програмного забезпечення, яке використовується у системах безпеки АЕС, зокрема у таких, що виконують функції категорії А (найвищої критичності). Він передбачає сувору формалізацію процесу розроблення, включно з обов’язковим використанням формальних методів верифікації. До таких методів належать математичне доведення коректності алгоритмів, перевірка специфікацій на повноту та відсутність протиріч, а також застосування інструментів автоматизованого аналізу коду [17].

Стандарт вимагає проведення як статичного аналізу (перевірка структури коду, пошук потенційних помилок без виконання програми), так і динамічного аналізу (тестування програмного забезпечення в умовах, максимально наближених до реальної експлуатації). Всі етапи розроблення мають бути задокументовані, включно з обґрунтуванням вибору методів, описом тестових сценаріїв та результатами валідації. Крім того, IEC 60880 вимагає проведення незалежних перевірок, які виконуються окремими командами, що не брали участі у розробці. Це дозволяє виявляти помилки, які могли бути пропущені основною

командою. На практиці реалізація цього стандарту може включати використання спеціалізованих мов програмування (наприклад, Ada чи SPARK), які зменшують ймовірність помилок, а також застосування сертифікованих інструментів аналізу коду.

3. IEC 62138 — "Nuclear power plants – Instrumentation and control important to safety – Software aspects for computer-based systems performing category B or C functions". Даний стандарт стосується програмного забезпечення, яке використовується в системах категорій B та C, тобто у менш критичних, але все ж важливих для безпеки функціях. Хоча рівень вимог до таких систем нижчий, ніж для категорії A, він зберігає основні принципи забезпечення високої якості ПЗ.

IEC 62138 встановлює процедури тестування, що включають модульне тестування, інтеграційне тестування та системне тестування з використанням як ручних, так і автоматизованих методів. Значна увага приділяється управлінню конфігурацією, що передбачає ведення повної історії змін програмного забезпечення та забезпечення можливості відкату до попередніх стабільних версій. Прикладом впровадження вимог цього стандарту може бути розроблення програмного забезпечення для систем моніторингу стану обладнання АЕС, яке не безпосередньо впливає на реакторну безпеку, але забезпечує операторів інформацією для прийняття рішень у кризових ситуаціях.

4. IEC 61508 — "Functional safety of electrical/electronic/programmable electronic safety-related systems". IEC 61508 є базовим галузевим стандартом з функційної безпечності, який застосовується не лише в атомній енергетиці, але й у багатьох інших сферах, зокрема в хімічній промисловості, нафтогазовій галузі та транспорті. Він вводить концепцію Safety Integrity Level (SIL) — кількісну міру надійності виконання системою її функцій безпеки. Існує чотири рівні SIL, де SIL 4 є найвищим і вимагає найнижчої ймовірності відмови.

Стандарт описує методи досягнення необхідного рівня SIL, включно з аналізом ризиків, проєктуванням із застосуванням резервування та незалежності каналів, використанням безпечних за замовчуванням режимів роботи (fail-safe) та

періодичним тестуванням обладнання. На практиці, для ІКС АЕС застосування IEC 61508 дозволяє інтегрувати вимоги до апаратного та програмного забезпечення у єдину систему управління безпекою, що охоплює весь життєвий цикл — від проєктування до виведення з експлуатації.

5. Рекомендації МАГАТЕ (серії NS та SSG). МАГАТЕ розробляє серії рекомендацій, що гармонізують міжнародний досвід у сфері проєктування та експлуатації ІКС АЕС. Документ **SSG-39 "Design of Instrumentation and Control Systems for Nuclear Power Plants"** встановлює загальні принципи, серед яких — необхідність резервування критичних елементів, забезпечення фізичної та функціональної незалежності каналів, а також впровадження захисту від відмов за загальною причиною [18; 19].

SSG-30 "Safety Classification of Structures, Systems and Components" визначає методи класифікації елементів за їх значенням для безпеки. Така класифікація допомагає оптимізувати ресурси при плануванні модернізацій та визначенні пріоритетів технічного обслуговування.

На практиці рекомендації МАГАТЕ часто використовуються як основа для створення або оновлення національних нормативних документів. Вони забезпечують єдність підходів до функційної безпечності в різних країнах і дозволяють інтегрувати найкращі світові практики у локальні проєкти.

В Україні міжнародні норми імплементуються через систему національних нормативних документів:

1. НП 306.2.202-2015 — "Загальні вимоги до систем та елементів, важливих для безпеки АЕС". Цей нормативний документ є базовим у національній системі регулювання безпеки атомних електростанцій України. Він конкретизує положення міжнародних стандартів (зокрема IEC 61513 та IEC 61508) з урахуванням особливостей української інфраструктури, кліматичних умов та специфіки енергоблоків, що експлуатуються [20].

Документ встановлює додаткові вимоги до сейсмостійкості систем, що особливо актуально для регіонів з підвищеною сейсмічною активністю. Також визначаються методи захисту від електромагнітних завад, що можуть виникати як

внаслідок роботи потужного енергетичного обладнання, так і в результаті зовнішніх факторів. Значна увага приділяється радіаційній стійкості елементів ІКС, включаючи вимоги до використання матеріалів і компонентів, здатних витримувати тривале опромінення без деградації параметрів. НП 306.2.202-2015 вимагає проведення перевірки відповідності на кожному етапі життєвого циклу — від проєктування та виробництва до монтажу, введення в експлуатацію, експлуатації та виведення з експлуатації. При цьому перевірки мають бути незалежними та підтвердженими документально.

2. НП 306.2.141-2008 — "Вимоги до програмного забезпечення, що використовується у системах, важливих для безпеки АЕС". Даний норматив адаптує положення міжнародних стандартів IEC 60880 та IEC 62138 для українських умов, враховуючи наявні технічні ресурси, кадрову підготовку та вимоги регулятора. Він регламентує весь процес розроблення програмного забезпечення, що використовується у критично важливих ІКС, починаючи з формування вимог і закінчуючи супроводом ПЗ у процесі експлуатації. В документі встановлено методи тестування ПЗ, включаючи валідацію, верифікацію, модульне тестування, інтеграційне тестування та системне тестування. Визначено порядок ведення технічної документації, який забезпечує відстежуваність змін і можливість незалежного аудиту. Передбачено проведення регулярних перевірок програмного забезпечення на відповідність чинним вимогам, особливо після його модифікацій [21].

3. ДНАОП та НПАОП — нормативи з охорони праці та безпеки в атомній енергетиці. ДНАОП (Державні нормативні акти з охорони праці) та НПАОП (Нормативні правові акти з охорони праці) охоплюють широкий спектр організаційних заходів безпеки, що стосуються як персоналу, так і загальної культури безпеки на АЕС.

Ці документи визначають вимоги до кваліфікації та підготовки персоналу, включно з періодичною переатестацією та тренуванням дій в умовах аварійних ситуацій. Встановлюються правила доступу до обладнання, процедури роботи в зонах з підвищеною радіаційною небезпекою, порядок проведення ремонтних

робіт та обслуговування ІКС. Особлива увага приділяється готовності до аварійного реагування, зокрема координації дій між різними підрозділами та службами.

4. Галузеві керівні документи (ГКД). Галузеві керівні документи розробляються для конкретних енергоблоків або проєктів з урахуванням їх технічних особливостей та історії експлуатації. Вони деталізують інтеграцію ІКС у конкретні технологічні процеси, визначають процедури взаємодії між різними системами, а також вимоги до модернізації та технічного обслуговування.

Наприклад, для одного енергоблока може бути розроблено ГКД, що описує алгоритми роботи систем аварійного охолодження активної зони реактора, умови активації резервних каналів, порядок контролю та тестування датчиків безпеки. Такі документи забезпечують узгодженість дій персоналу та є основою для навчання операторів і технічних спеціалістів.

Аналіз міжнародних і національних нормативних документів показує, що, незважаючи на відмінності у структурі та деталізації вимог, більшість із них має спільні фундаментальні принципи. Ці принципи відображають універсальний підхід до побудови безпечних і надійних інформаційно-керуючих систем атомних електростанцій, який ґрунтується на багаторівневому захисті, резервуванні, суворому контролі змін та незалежній перевірці.

Перш за все, нормативи вимагають рівневої архітектури захисту (defense-in-depth), у якій передбачено наявність декількох незалежних каналів управління та контролю. Кожен рівень виконує власні функції безпеки, а його відмова не повинна призводити до втрати працездатності всієї системи. Такий підхід дозволяє зберегти контроль навіть у випадку критичного збою одного з рівнів.

Другою важливою вимогою є резервування функцій та елементів, причому нормативи передбачають як просте (ідентичне) резервування, так і диверсне (різні алгоритми, різні апаратні та програмні рішення). Диверсне резервування особливо ефективне для зниження ймовірності відмов за загальною причиною, оскільки різні технології по-різному реагують на однакові зовнішні впливи.

Захист від відмов за загальною причиною (Common Cause Failure, CCF) є ще одним спільним елементом нормативних вимог. Для цього передбачено фізичну та функціональну ізоляцію між каналами, використання різних технологічних платформ, постачальників та алгоритмів. Наприклад, два незалежні канали аварійного захисту можуть працювати на різних типах процесорів і з різними операційними системами, що значно зменшує ризик одночасної відмови.

Важливою умовою стабільної та передбачуваної роботи ІКС є жорстка регламентація змін. Будь-яка зміна, навіть незначна, у конфігурації обладнання або програмного забезпечення проходить багаторівневу процедуру перевірки та погодження, що включає аналіз впливу змін на безпеку, тестування у модельованих умовах та документальне затвердження.

Документування є обов'язковим елементом усього життєвого циклу системи. Повний комплект документації охоплює проєктні рішення, експлуатаційні інструкції, протоколи випробувань, результати аудиту та записи про технічне обслуговування. Це забезпечує простежуваність усіх процесів і дає змогу проводити ретроспективний аналіз у випадку відмов.

Ще одним обов'язковим елементом є незалежна верифікація (Independent Verification & Validation, IV&V), що передбачає залучення сторонніх фахівців або незалежних організацій для перевірки правильності реалізації вимог. Незалежний підхід гарантує об'єктивність оцінки та дозволяє виявити помилки, які могли бути пропущені внутрішніми командами розробників або інженерів.

Нарешті, нормативи вимагають проведення аналізу безпеки на всіх етапах життєвого циклу системи. Для цього застосовуються сучасні методи, такі як:

– **FMEA (Failure Modes and Effects Analysis)** — аналіз видів і наслідків відмов, спрямований на визначення слабких місць системи та розроблення заходів для зниження ймовірності їх виникнення [22; 23];

– **FTA (Fault Tree Analysis)** — побудова логічних моделей, що описують взаємозв'язок між відмовами окремих елементів і загальною відмовою системи [24];

– **HAZOP (Hazard and Operability Study)** — систематичний метод виявлення потенційно небезпечних відхилень у роботі системи та оцінки їхнього впливу на безпеку.

Таким чином, сукупність цих вимог формує комплексну систему заходів, яка дозволяє забезпечити високу функційну безпечність ІКС АЕС та мінімізувати ймовірність виникнення аварійних ситуацій навіть у найскладніших умовах експлуатації.

1.1.2 Додаткові вимоги до систем аварійного захисту

Системи аварійного захисту (САЗ) інформаційно-керуючих систем атомних електростанцій мають унікальний набір вимог, що істотно перевищують стандартні критерії, які пред’являються до інших автоматизованих систем промислового призначення. Це зумовлено винятковою критичністю їх функцій: САЗ призначені для автоматичного виявлення та негайної ліквідації небезпечних для реакторної установки умов, включно з такими, що можуть призвести до серйозних аварій із виходом радіоактивних речовин за межі контрольованої зони.

Першою і найбільш принциповою вимогою до САЗ є мінімальний час реакції. Затримка між виявленням аварійної ситуації та виконанням захисних дій визначається у нормативних документах і має бути жорстко регламентована. У більшості випадків мова йде про мілісекундні або десятки мілісекундні інтервали. Це пов’язано з тим, що розвиток аварійних процесів у ядерних реакторах може мати лавиноподібний характер, і навіть затримка в частки секунди може призвести до непоправних наслідків. Для досягнення мінімального часу реакції застосовуються високошвидкісні процесори реального часу, спеціалізовані програмовні логічні інтегральні схеми (ПЛІС, FPGA) та апаратні канали прямого доступу до виконавчих механізмів без проміжних шарів обробки.

Другою ключовою вимогою є подвійне резервування каналів. Це означає, що у складі САЗ має бути щонайменше два, а зазвичай три і більше незалежних канали виконання функцій, причому кожен канал здатен самостійно забезпечити

реалізацію повного обсягу захисних дій. Такий підхід дозволяє зменшити ймовірність відмови за загальною причиною, коли несправність або помилка впливають на всі канали одночасно. У практиці ядерної енергетики канали резервування часто будуються на базі різного апаратного забезпечення, з використанням різних мов програмування або навіть різних логічних алгоритмів, що підвищує диверсність системи та додатково знижує ймовірність синхронної відмови.

Третім важливим аспектом є фізична та функційна ізоляція каналів. Мова йде не лише про розділення апаратних елементів (окремі шафи, кабельні траси, відсутність спільних блоків живлення), але й про повну незалежність програмного забезпечення та логіки роботи. Такий підхід гарантує, що навіть у випадку фізичного пошкодження або зараження шкідливим ПЗ одного каналу, інші канали зберігатимуть свою працездатність і здатність виконувати захисні функції.

Ще однією специфічною вимогою є захист від помилок оператора. Інтерфейси САЗ проєктуються таким чином, щоб унеможливити або максимально ускладнити внесення некоректних команд у критичний момент. Наприклад, для підтвердження окремих дій можуть використовуватися багаторівневі підтвердження, фізичні ключі або механічні перемикачі, відсутність можливості віддаленого скасування певних команд, а також контекстна блокування елементів керування при настанні визначених умов. Важливо, що САЗ, на відміну від систем оперативного управління, повинні мати пріоритет у виконанні команд над будь-якими іншими підсистемами.

Схема (Рисунок 1.1) демонструє багаторівневу архітектуру САЗ із чітким розподілом функцій між каналами. У верхньому рівні зображено блоки сенсорних модулів, що здійснюють безперервний моніторинг критичних параметрів реакторної установки — температури, тиску, рівня потужності, положення органів регулювання тощо. Кожен сенсор підключений до свого каналу обробки, що забезпечує незалежність вимірювальних трактів.

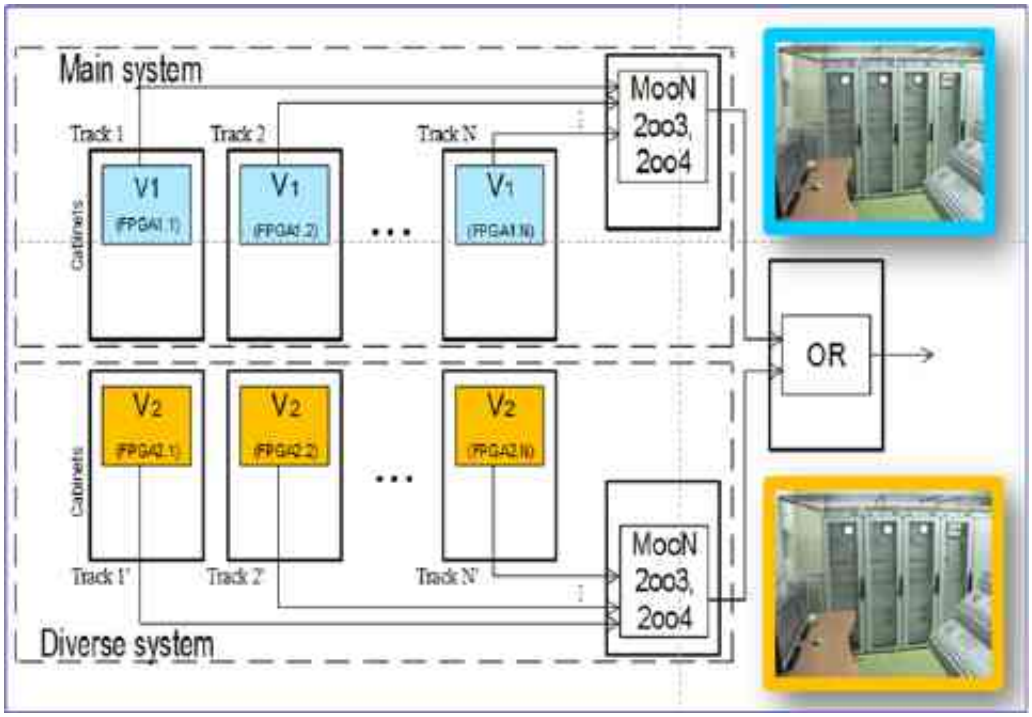

Рисунок 1.1 - Структурна схема системи аварійного захисту [5]

На середньому рівні схеми представлені процесорні блоки кожного з каналів, які виконують алгоритми аналізу та прийняття рішень. Вони мають вбудовані модулі самодіагностики, що постійно перевіряють коректність роботи апаратної та програмної частин. Зв’язок між каналами мінімізований і реалізований лише в частині, необхідній для узгодження загальних дій у випадках, коли аварійні ситуації фіксуються кількома каналами одночасно.

Нижній рівень схеми ілюструє виконавчі механізми, такі як системи швидкого введення аварійних стрижнів, подача борної кислоти, активація системи охолодження або відключення турбіни. Важливою особливістю є наявність незалежних ліній зв’язку від кожного каналу до власного набору виконавчих пристроїв, що виключає можливість відмови через спільну точку з’єднання.

Завдяки такій структурі САЗ здатні забезпечити високий рівень функційної безпечності навіть за умов багатофакторного впливу, включно з відмовами обладнання, помилками персоналу та зовнішніми загрозами.

1.2 Аналіз відмов інформаційно-керуючих систем АЕС

Надійність і функційна безпечність ІКС АЕС визначаються не лише правильністю їхнього проєктування, а й здатністю ефективно протистояти відмовам, що можуть мати різну природу. Аналіз відмов є важливим етапом, оскільки дозволяє:

- визначити найуразливіші компоненти системи;
- оцінити ймовірність відмов у різних режимах експлуатації;
- сформувати вимоги до резервування та захисних механізмів;
- створити базу для прогнозування безпеки.

Досвід експлуатації ІКС на вітчизняних та зарубіжних АЕС свідчить, що відмови можуть виникати як у апаратних, так і в програмних компонентах. Причому важливу роль відіграють відмови за загальною причиною (ВЗЗП), що здатні одночасно вивести з ладу кілька незалежних каналів.

1.2.1 Аналіз відмов та причин спрацювання систем аварійного захисту

Системи аварійного та попереджувального захисту (САЗ і СПЗ) є ключовими елементами функційної безпечності інформаційно-керуючих систем атомних електростанцій. Їх призначення полягає у своєчасному виявленні відхилень технологічних параметрів від допустимих меж та автоматичному виконанні дій, спрямованих на переведення реакторної установки в безпечний стан. Надійність і ефективність цих систем безпосередньо визначає рівень ядерної безпеки енергоблоку.

Відмови у САЗ і СПЗ можуть мати як технічну, так і організаційну природу. Вони класифікуються на реальні спрацювання, що викликані небезпечними умовами, та помилковими спрацюваннями, коли сигнал формують несправні датчики, некоректно налаштовані алгоритми або помилки персоналу. Навіть якщо помилкове спрацювання не становить загрози безпеці, воно призводить до

незапланованого зупину реактора, що викликає значні економічні збитки, впливає на ресурс обладнання та створює додаткові навантаження на персонал.

Для оцінки ефективності роботи САЗ і СПЗ було проведено аналіз даних про відмови одно- та двоверсійних систем за період експлуатації кількох енергоблоків. Таблиця 1.1 представляє узагальнену статистику кількості відмов АЕС України, де наведено загальну кількість відмов для одноверсійної та двоверсійної архітектури. Видно, що двоверсійні системи мають суттєво менший рівень відмов, особливо у частині помилкових спрацювань, завдяки використанню незалежних каналів та механізмів взаємної перевірки сигналів.

Таблиця 1.1. Дані про відмови одно та двоверсійної системі аварійного та попереджувального захисту реактору

Рік	**Запорізька АЕС**	**Південноукраїнська АЕС**	**Рівненська АЕС**	**Хмельницька АЕС**	**Всього**
1999	1	2	1	1	5
2000	2	1	–	1	5
2001	2	1	1	–	4
2002	1	1	2	1	5
2003	2	2	1	–	5
Всього для одноверсійної системи за п’ять років					24
2004	1	1	–	–	2
2005	1	–	–	–	1
2006	–	1	1	–	2
2007	–	–	–	1	1
2008	1	–	1	–	2
Всього для мультидиверсійної системи за п’ять років					8
2010 - 2015: середня кількість спрацювань АЗ-ПЗ на 15 реакторних блоках АЕС України					2,2
2015 - 2020: середня кількість спрацювань АЗ-ПЗ на 15 реакторних блоках АЕС України					4,6

Аналіз цих даних показав, що:

– Одноверсійні системи характеризуються вищим рівнем помилкових спрацювань через відсутність механізмів порівняння сигналів між незалежними каналами. Відмова одного критичного елемента автоматично призводить до запуску захисної дії.

– Двоверсійні системи завдяки резервуванню та алгоритмам голосування (наприклад, принцип 1 з 2 або 2 з 3) значно знижують ймовірність помилкового

спрацювання. Це досягається перевіркою коректності сигналів між каналами та їх незалежною обробкою.

Додатково надійність двоверсійних систем підвищується за рахунок фізичної та функційної ізоляції каналів, використання обладнання різних виробників, а також впровадження різнорідних алгоритмів обробки сигналів (диверсне резервування).

Рисунок 1.2 ілюструє статистику спрацювань систем аварійного захисту за останні 10–15 років, що відображає динаміку змін після модернізацій та впровадження цифрових технологій. Графік демонструє тенденцію до поступового зниження кількості як реальних, так і помилкових спрацювань. Окремі піки, зафіксовані у 2012 та 2016 роках, співпадають із періодами виконання масштабних модернізацій та заміни підсистем.

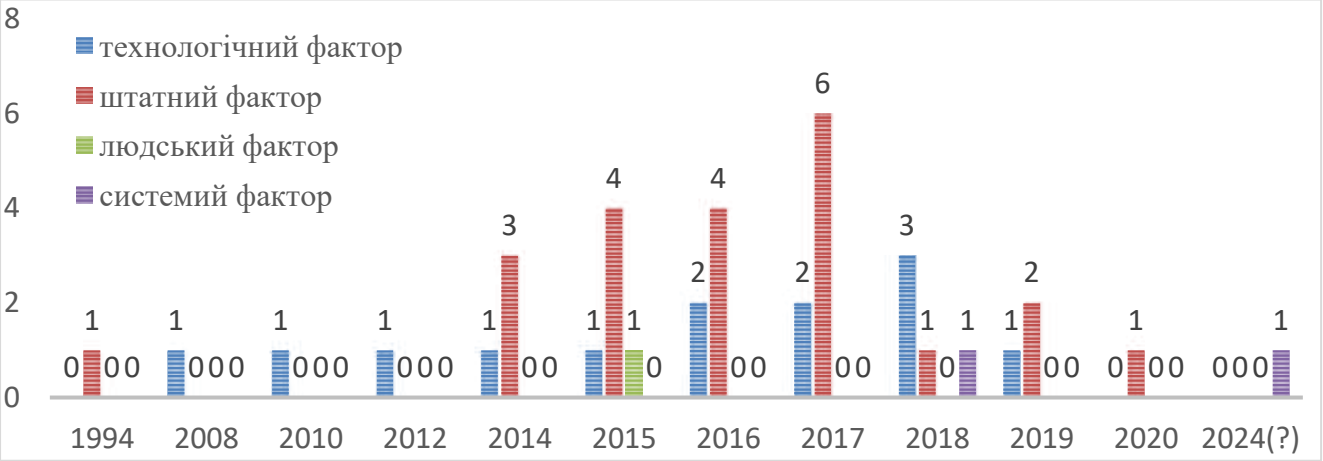

Рисунок 1.2 - Статистика спрацювань систем аварійного захисту [25]

З наведених даних можна зробити низку висновків:

1. Модернізація ІКС та перехід на цифрові технології позитивно вплинули на зменшення кількості як помилкових, так і реальних спрацювань САЗ;

2. Алгоритмічні удосконалення систем діагностики дозволили більш точно ідентифікувати реальні аварійні ситуації та зменшити кількість хибних сигналів;

3. Вплив людського фактору зберігається: частина помилкових спрацювань пов’язана з некоректними діями оператора, особливо під час регламентних випробувань;

4. Технічні причини відмов включають відмови сенсорів, збої в живленні, електромагнітні перешкоди, помилки налаштування порогів спрацювання та деградацію елементів внаслідок впливу робочого середовища.

З урахуванням цього, перспективними напрямами підвищення надійності САЗ і СПЗ є:

– розширення використання багатоканальних та багатоверсійних архітектур із різнорідним резервуванням;

– впровадження адаптивних алгоритмів діагностики, здатних змінювати параметри контролю залежно від стану обладнання;

– удосконалення засобів захисту від електромагнітних завад та збоїв живлення;

– автоматизація процедур перевірки та калібрування сенсорів.

Таким чином, комплексна модернізація апаратної та програмної частини, поєднана з удосконаленням експлуатаційних регламентів, є ключем до зниження кількості відмов та підвищення ефективності систем аварійного і попереджувального захисту реакторів.

1.2.2 Аналіз відмов за загальною причиною інформаційно-керуючих систем

Відмови за загальною причиною (ВЗЗП) є одним з найкритичніших видів збоїв у багатоканальних резервованих ІКС атомних електростанцій, оскільки один спільний фактор здатний одночасно вивести з ладу кілька на перший погляд незалежних каналів, нівелюючи переваги резервування. Практика експлуатації показує, що ВЗЗП формуються як на етапі створення системи (конструктивні помилки та тиражовані дефекти), так і під час її життєвого циклу (вплив середовища, людський фактор, дефекти взаємодії між компонентами).

Узагальнення причин ВЗЗП доцільно почати з трьох базових класів, що зустрічаються найчастіше і відповідають реальним механізмам поширення спільних відмов у багатоканальних архітектурах, що наглядно ілюструє Рисунок 1.3.

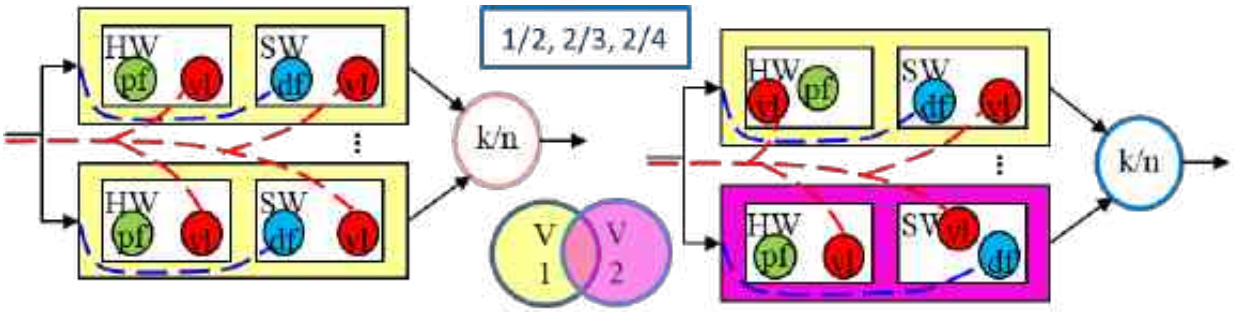

Рисунок 1.3 - Відмови за загальною причиною і принцип диверсності [25]

Рисунок 1.3 містить три домінуючі механізми: (pf) “множинні фізичні дефекти” резервних каналів технічних засобів; (df) “тиражовані (репліковані) проєктні дефекти” програмних компонентів та/або HDL-проєкту FPGA; (vl) “дефекти взаємодії”, спричинені вразливостями на межах ПЗ/FPGA/апаратури. Схема показує, що кожен механізм має власні зони впливу на канали, а в перетинах виникають комбіновані сценарії, коли фізична деградація, логічний дефект та вразливість інтерфейсу підсилюють один одного.

У pf-сценаріях множинних фізичних дефектів спільний вплив середовища (радіація, температура, ЕМЗ), серійні дефекти партій комплектуючих або однакова деградація елементної бази призводять до синхронного відмовляння одразу кількох каналів. Типові маркери — одночасні хиби датчиків з одного технологічного ланцюга, кореляція помилок у блоках живлення, спільні для каналів траси/шафи. Ефективні протизаходи: просторове рознесення, незалежні джерела живлення і заземлення, різні ревізії/партії компонентів, випробування на стійкість (стрес-скринінг) та підвищені кваліфікаційні випробування.

У df-сценаріях тиражованих проєктних дефектів коренева причина закладається у вимогах, архітектурних рішеннях або інструментальному тракті (компілятор/синтезатор/IP-ядра). Якщо один і той самий алгоритмічний

прорахунок або помилка інструмента транслюється в усі канали, “незалежність” стає лише номінальною. Хрестоматійна ілюстрація — реплікований дефект ПЗ/FPGA: одна і та сама помилка у бібліотеці або VHDL-модулі проявляється синхронно у всіх каналах; для суміжних доменів відомо, що 20–50% відмов у космічних системах (1990–2019) припадали саме на клас дизайн-дефектів — це демонструє масштаб проблеми для високонадійних застосувань. Технічно виправдані контрзаходи: диверсність розроблення (різні мови/інструменти/команди), незалежна IV&V, формальні методи, MBT з мутаційним інжектуванням помилок, окремі репозиторії і відгалужені CI-ланцюги для каналів.

У vl-сценаріях дефектів взаємодії спільною причиною стає інтерфейс: загальна шина, синхронізація між доменами такту, спільні стек-протоколи або механізми обміну. Вразливості (від класичних помилок протоколів до гонок, метастабільності й небезпечних reset-послідовностей) можуть проявлятися одночасно в усіх каналах, якщо ті залежать від однакової взаємодії. Типові симптоми — синхронні “зависання” при збоях таймінгу, однакові CRC-збої на спільній ділянці мережі, лавиноподібні помилки при відмові загального “годинника” або арбітра шини. Засоби протидії: функційна і протокольна диверсність (різні стеки/кодеки/CRC), часова диверсність (зсув фаз дискретизації, різні watchdog-вікна), жорсткі контрактні інтерфейси, сегментація/ізоляція мереж і “безпечний тайм-аут”.

Описана класифікація узгоджується з практичними подіями: одночасні збої каналів під час переходу на резервне живлення (pf), синхронна втрата працездатності після оновлення прошивок (df), каскадні помилки під час колізій у спільній шині або при збої синхронізації (vl). Звідси випливають і пріоритети інженерного захисту: поєднання апаратної та програмної диверсності, просторово-функційної ізоляції, незалежних команд розроблення й регулярного інжектування відмов з вимірюванням чутливості каналів дає суттєво кращі показники, ніж будь-який із заходів поодинці.

Для подальшої робочої інтерпретації причинно-наслідкових зв’язків корисним є оглядова концептуальна схема перетинів ВЗЗП і принципу диверсності (Рисунок 1.3), де показано, як багатовидова диверсність локалізує “спільні” механізми відмов і розриває типові ланцюжки поширення збоїв.

Висновок. Три домінуючі механізми ВЗЗП — множинні фізичні дефекти (pf), тиражовані дизайн-дефекти (df) та дефекти взаємодії/вразливості (vl) — утворюють базову “карту ризиків” для багатоканальних ІКС. Їх текстова інтерпретація, підкріплена вимірюванням чутливості через ін’єкцію відмов, МВТ-кампаніями та незалежною IV&V, дозволяє не лише пояснити походження спільних збоїв, а й обґрунтувати структурно диверсні рішення (різні технології/алгоритми/інтерфейси), які зменшують частку CCF у загальному профілі ризику та повертають резервуванню його реальну ефективність.

1.3 Аналіз застосування принципу диверсності в інформаційно-керуючих системах АЕС та інших критичних системах

У сучасних інформаційно-керуючих системах атомних електростанцій (ІКС АЕС) принцип диверсності є одним з ключових інженерних підходів до підвищення функційної безпечності. Він спрямований на зниження ризику відмов за загальною причиною (ВЗЗП) шляхом впровадження різноманітності у реалізації критичних функцій. Диверсність дозволяє суттєво зменшити ймовірність одночасного виходу з ладу кількох каналів системи через спільні фактори, тим самим підвищуючи реальну ефективність резервування.

1.3.1 Принцип диверсності. Аналіз класифікаційних схем видів диверсності

Принцип диверсності (різноманітності, багатоверсійності) — це один із ключових інженерних методів підвищення надійності та функційної безпечності систем, який полягає у створенні двох або більше незалежних реалізацій однієї і тієї ж функції. Ці реалізації відрізняються за апаратними компонентами,

програмним забезпеченням, алгоритмами чи організаційним підходом до розроблення [26]. Основна ідея полягає у тому, що різні варіанти реалізації мають різні вразливості, а отже, відмова одного варіанту з меншою ймовірністю призведе до відмови іншого. Таким чином, диверсність знижує ризик відмов за загальною причиною, які можуть одночасно вивести з ладу всі канали багатоканальної системи.

Наприклад:

– **У ядерній енергетиці** для одного й того ж захисного функціоналу можуть застосовуватися два різні контролери — один на базі процесорів від одного виробника, інший — від іншого, з різними операційними системами та кодом, написаним на різних мовах програмування.

– **В авіації** один і той самий модуль навігації може мати дві незалежні програмні реалізації: одну, створену командою з США на мові C++, іншу — командою з Європи на мові Ada.

– **У космічних системах** диверсність може реалізовуватися через дублювання обчислювальних модулів із різними архітектурами (наприклад, PowerPC і ARM) для захисту від радіаційно індукованих збоїв.

– **В IT-безпеці** аналогічний підхід застосовується у системах багатофакторної автентифікації, де різні фактори (пароль, апаратний токен, біометрія) виконують одну функцію — перевірку користувача, але за допомогою різних механізмів.

Завдяки такому підходу система зберігає працездатність навіть за умови, що один із каналів або реалізацій вийде з ладу через апаратний дефект, помилку в програмі чи зовнішній вплив.

Практика застосування диверсності у різних індустріях демонструє суттєві відмінності у виборі типів і глибині реалізації цього принципу, що обумовлено як технічними, так і економічними чинниками.

Таблиця 1.2 - Класифікація видів диверсності та галузеві приклади реалізації [26]

Diversity types (NUREG-6303, NUREG-7007)	Indutrial domains / Multi-version systems													
	Space		Aviation				Railways	Automotive	Chemic industry	Defense	Power Plants	NPPs		e-Commers
	Shuttle	ISS	MCJVC	A320 FCS	A340, A380, FCS	Boeing 777	SCB	Street by-wire system	CCPS	MICS	Electr. Grid	RTS	ESFAS	WSOA
Design				X	X			X	X			X	X	X
Equipment		X		X	X	X	X		X			X	X	X
Function	X	X		X	X	X	X		X	X		X	X	X
Human	X	X	X	X	X	X	X		X	X	X	X	X	X
Signal							X	X	X			X	X	
Software	X		X	X	X	X			X			X	X	X
Others	X	X	X	X	X	X						X		

Таблиця 1.2 відображає узагальнену таблицю, що демонструє підходи до диверсності у космічній, авіаційній, ядерній та інших критичних галузях:

– Космічна галузь зосереджується переважно на апаратній диверсності, використовуючи різні архітектури процесорів, технологічні процеси виготовлення мікросхем та альтернативних постачальників компонентів. Це критично через високий рівень радіаційного впливу та відсутність можливості ремонту.

– Авіація активно застосовує програмну й алгоритмічну диверсність: паралельні версії програмного забезпечення створюють незалежні команди з використанням різних мов програмування, компіляторів та середовищ розроблення.

– Ядерна енергетика реалізує комбіновану диверсність, поєднуючи апаратні та програмні відмінності, фізичну рознесеність каналів, а також різні алгоритми обробки сигналів.

– Інші критичні інфраструктури (транспорт, оборона, нафтогазова промисловість) застосовують змішані підходи з акцентом на підвищену стійкість до зовнішніх впливів, у тому числі кіберзагроз.

Узагальнено виділяють такі типи:

– Апаратна (Hardware diversity) — використання відмінних процесорів, архітектур, виробників компонентів.

– Програмна (Software diversity) — створення різних реалізацій ПЗ з використанням різних мов, операційних систем, бібліотек та інструментів.

– Алгоритмічна — застосування різних методів обробки одних і тих самих вхідних даних.

– Організаційна — розроблення незалежними колективами або в різних країнах.

Кожен вид має власні переваги та обмеження: апаратна забезпечує фізичну незалежність, але часто є дорогою; програмна гнучкіша, але потребує суворої верифікації; алгоритмічна дає логічну незалежність, але ускладнює тестування.

NUREG-7007 систематизує вибір і впровадження диверсності в багатоканальних системах безпеки (Рисунок 1.4), встановлюючи:

– рівні реалізації (апаратна, програмна, алгоритмічна, організаційна та інші);

– критерії оцінки незалежності реалізацій;

– методи кількісного аналізу ефективності.

Рисунок 1.4 – Дворівневий класифікатор за стандартом NUREG-7007 [27]

Особлива увага приділяється аналізу того, чи зберігається незалежність реалізацій за наявності спільних зовнішніх або внутрішніх впливів. Наприклад, апаратна диверсність може бути формально присутньою, але втрачати ефективність, якщо обидва рішення залежать від одного постачальника, схильного до серійних дефектів.

Таким чином, принцип диверсності у поєднанні з формалізованими підходами, як у NUREG-7007 [28–30], дозволяє знизити ризики CCF та забезпечити надійність критичних систем навіть за наявності непередбачуваних зовнішніх чи внутрішніх загроз.

1.3.2 Огляд застосування принципу диверсності

Принцип диверсності набув широкого поширення у багатьох галузях, де відмовостійкість і безпечність є критично важливими. Його сутність полягає в одночасному використанні кількох незалежних реалізацій одного функціоналу, що відрізняються за апаратним, програмним, алгоритмічним чи організаційним втіленням. Такий підхід знижує ймовірність відмови за загальною причиною, оскільки незалежні реалізації мають різну чутливість до однакових зовнішніх чи внутрішніх факторів.

У ядерній енергетиці диверсність є обов’язковим елементом проєктування ІКС АЕС. Наприклад, у проєкті Westinghouse AP1000 система захисту реактора реалізована у двох каналах, побудованих на різних апаратних платформах та з різними версіями прошивок. Такий підхід дає змогу уникнути одночасної відмови обох каналів через помилку конкретного виробника або дефект серійної партії обладнання.

Інший приклад – Siemens Teleperm XS, де програмні модулі створюються різними командами розробників у незалежних середовищах, із застосуванням різних мов програмування та компіляторів. Це забезпечує не лише алгоритмічну і програмну диверсність, але й організаційну, оскільки колективи можуть

працювати у різних країнах і керуватися власними процедурами розроблення та тестування.

У вітчизняних проєктах ІКС АЕС також активно застосовується поєднання різних типів диверсності. Наприклад, можуть використовуватися різні мови програмування (C та ASM) для паралельної реалізації одного й того ж алгоритму, а також різні типи або технології мікросхем (мікроконтроллер та FPGA) в якій один тип має жорстку функційну складову, інша гучка, універсальна, загального призначення. Це підвищує стійкість до програмних помилок і вразливостей, які можуть бути притаманні конкретній платформі.

За межами ядерної галузі принцип диверсності застосовується у багатьох критичних сферах:

– Авіація – у сучасних авіалайнерах, таких як Boeing 777 чи Airbus A350, системи управління польотом будуються з використанням різних апаратно-програмних комплексів, розроблених різними компаніями.

– Космічна галузь – у бортових системах космічних апаратів широко використовується апаратна диверсність через екстремальні умови роботи (радіація, температурні перепади). Окремі модулі дублюються із застосуванням різних виробників мікросхем та відмінних техпроцесів.

– Залізничний транспорт – системи безпеки руху поїздів реалізують функції контролю з використанням декількох незалежних контролерів, розроблених різними постачальниками.

– Оборонна промисловість – у комплексах протиракетної оборони застосовується алгоритмічна диверсність для підвищення надійності в умовах складних і швидкозмінних загроз.

Водночас, впровадження диверсності має свої виклики:

– Вартість та терміни – паралельне розроблення кількох незалежних рішень вимагає значних ресурсів і часу.

– Складність тестування – необхідно застосовувати методи, що дозволяють не лише перевірити кожну реалізацію окремо, а й зіставити їх результати в

однакових умовах. Для цього все частіше використовують Model-Based Testing та автоматизовані стенди порівняльного тестування.

– Узгодження інтеграції – різні реалізації можуть мати несумісності на рівні інтерфейсів, форматів даних чи логіки обробки, що потребує додаткового проєктування адаптерів.

Таким чином, диверсність є одним із ключових інструментів підвищення надійності критичних систем, а її грамотне впровадження дозволяє суттєво знизити ризик відмови за загальною причиною. При цьому важливо балансувати між рівнем безпеки, який вона забезпечує, та ресурсами, необхідними для реалізації такого підходу.

1.3.3 Особливості побудови двоверсійних систем на програмовних платформах

У сучасних інформаційно-керуючих системах атомних електростанцій усе ширше застосовуються програмовні логічні інтегральні схеми для реалізації критичних функцій. Такі рішення поєднують у собі апаратну надійність, високу швидкодію та гнучкість налаштування логіки. Використання FPGA у двоверсійній архітектурі дозволяє забезпечити:

– Апаратну диверсність — незалежний вибір виробника, серії FPGA та технологічного процесу виготовлення.

– Програмну та алгоритмічну диверсність — реалізація однієї функції на різних мовах опису апаратури (наприклад, VHDL та Verilog), із використанням різних синтезаторів, бібліотек та інструментів розроблення.

– Гнучкість та масштабованість — можливість швидкої зміни прошивки або часткової реконфігурації без демонтажу обладнання.

– Зниження ризику відмов за загальною причиною завдяки незалежним підходам до реалізації.

Рекомендації щодо побудови двоверсійних FPGA-систем:

1. **Незалежність розроблення.** Розроблення кожної версії двоверсійної системи повинна виконуватися окремими командами, при цьому необхідно мінімізувати будь-які контакти між ними та уникати обміну деталями реалізації. Це знижує ризик повторення однакових помилок у двох версіях. Щоб ще більше зменшити ймовірність прихованих залежностей, варто використовувати різні мови опису апаратури, різні інструментальні засоби проєктування (EDA), а за можливості — навіть різні операційні системи робочих станцій розробників.

2. **Вибір апаратної бази.** Для підвищення апаратної диверсності рекомендується використовувати FPGA від різних виробників, наприклад Xilinx та Intel/Altera, які мають відмінні архітектури й технології виготовлення. При виборі компонентів слід звертати увагу на внутрішні відмінності, такі як тип вбудованої пам’яті, конфігураційна логіка, структура комутаційної мережі. Ці технічні відмінності зменшують вірогідність виникнення однакових вразливостей у двох версіях.

3. **Диверсифікація інструментів.** Для кожної версії бажано застосовувати різні програмні комплекси для синтезу та трасування логіки. Наприклад, перша версія може розроблятися в середовищі Xilinx Vivado, а друга — в Intel Quartus Prime. Крім того, варто уникати використання однакових комерційних IP-ядр, надаючи перевагу власним розробкам або незалежним альтернативам, особливо для критичних модулів. Це забезпечує програмну та алгоритмічну диверсність навіть на рівні окремих функціональних блоків.

4. **Верифікація та тестування.** Одним з методів перевірки працездатності двоверсійних систем може виступати методологія Model-Based Testing, при якій для кожної версії формується окремий набір тестових сценаріїв. Крім функціональних тестів, необхідно проводити моделювання відмов (Fault Injection), щоб перевірити, як система реагує на різні типи збоїв. Результати роботи обох версій потрібно порівнювати у режимі реального часу, що дозволяє швидко виявляти розбіжності й вчасно реагувати на них.

5. **Архітектурні особливості.** Дві версії системи мають функціонувати максимально незалежно одна від одної, використовуючи мінімально спільну

апаратну інфраструктуру. Для синхронізації та порівняння результатів слід застосовувати окремий спеціалізований модуль, фізично ізольований від основних каналів обробки даних. Така ізоляція дозволяє знизити ризик перехресних збоїв і підвищує загальну надійність системи.

Обмеження та виклики впровадження:

– Вартість і трудомісткість — два незалежні процесу розроблення потребують значно більших витрат ресурсів і часу.

– Складність узгодження інтерфейсів — при різних підходах до реалізації інтерфейсів введення/виведення можливі труднощі з інтеграцією.

– Обмеженість перевірених IP-бібліотек — для деяких функцій може бути складно знайти якісні та взаємонезалежні бібліотеки від різних постачальників.

– Ризик прихованих залежностей — навіть за різних мов і інструментів можливі однакові помилки через спільну специфікацію або стандарти (наприклад, однакові протоколи чи формати даних).

– Тестове навантаження — зростає обсяг тестування та складність організації випробувань, особливо в режимі реального часу.

Двоверсійні системи на FPGA є потужним інструментом підвищення функційної безпечності ІКС АЕС, проте їх впровадження вимагає ретельного планування, суворого дотримання принципу незалежності розроблення та розширених методів тестування. Такий підхід значно знижує ризики відмов за загальною причиною, але вимагає збалансування між рівнем безпеки, витратами та складністю експлуатації.

1.4 Аналіз методів оцінювання диверсності та забезпечення функційної безпечності багатоверсійних систем

У сучасних умовах розвитку складних технічних систем проблема підвищення їхньої відмовостійкості та безпеки є однією з ключових. Традиційні методи резервування не завжди забезпечують достатній рівень надійності, особливо у випадках, коли відмова може бути спричинена не тільки апаратними

збоями, але й програмними помилками, або ж однотипними проєктними недоліками. У таких випадках ефективним підходом виступає диверсифікація, яка базується на принципі використання функціонально і структурно відмінних реалізацій одного й того ж завдання.

Застосування диверсифікації дозволяє мінімізувати ймовірність одночасної відмови незалежних компонентів, оскільки різні методи проєктування, мови опису апаратури та алгоритмічні підходи створюють умови для появи різнорідних помилок. У контексті критично важливих систем, зокрема систем безпеки атомних електростанцій, авіаційних та космічних комплексів, такий підхід є основою досягнення багаторівневого захисту від збоїв і катастрофічних відмов.

1.4.1 Аналіз класифікатора диверсності

У сучасних системах керування та безпеки атомних електростанцій важливим завданням є мінімізація ризику відмов за загальною причиною. Одним з найбільш дієвих підходів до зниження цього ризику є застосування принципів диверсності – використання різних методів, засобів і технологій для виконання критичних функцій. Однак ефективність диверсності потребує формалізації та стандартизації. Саме для цього використовуються класифікатори диверсності – нормативні та методичні документи, що визначають види, рівні та критерії оцінки диверсності. Вони дозволяють систематизувати підходи до проєктування, забезпечують уніфікацію термінології, а також створюють основу для аргументованої оцінки безпеки при ліцензуванні та експлуатації систем на АЕС.

Серед багатьох міжнародних документів, що описують підходи до диверсності (рекомендації МАГАТЕ, стандарти IEC 61508, IEC 61226, національні нормативи різних країн), одним з найбільш розроблених і визнаних є класифікатор, представлений у документі NUREG-7007 “Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems”. Причини вибору саме цього класифікатора як основи подальших досліджень такі:

– Комплексність – документ систематично охоплює всі аспекти диверсності: функціональну, технологічну, програмну, часову, організаційну.

– Практична спрямованість – NUREG-7007 орієнтований на конкретні проєктні рішення та дає інженерам інструменти для оцінки достатності диверсності.

– Регуляторна значимість – документ розроблений Комісією з ядерного регулювання США (NRC), що робить його авторитетним у міжнародній практиці.

– Узгодженість з іншими стандартами – NUREG-7007 тісно корелює з IEC 61508 та документами МАГАТЕ, що забезпечує можливість його використання як у міжнародних, так і у національних контекстах.

NUREG-7007 пропонує структуровану систему оцінки диверсності, що базується на класифікації за типами та рівнями. Основні положення класифікатора включають:

1. Типи диверсності:
   - функціональна (різні методи реалізації однакової функції);
   - апаратна (різні апаратні платформи, виробники або архітектури);
   - програмна (різні алгоритми, мови програмування, компілятори);
   - часова (різні моменти запуску, незалежні тактові генератори);
   - інформаційна (різні вхідні дані чи датчики для дублюючих каналів);
   - організаційна (незалежні розробницькі групи, різні постачальники).
2. Класифікація рівнів застосування диверсності – від низького (мінімальні відмінності) до високого (повна незалежність у всіх критичних аспектах).
3. Методологія оцінки ризику – у документі наведено матриці рішень, що дозволяють визначити, які види диверсності доцільно застосовувати для конкретних систем чи підсистем.
4. Рекомендації щодо інтеграції диверсності – NUREG-7007 не обмежується описом, а пропонує конкретні стратегії застосування диверсності при модернізації існуючих систем чи створенні нових.

Таким чином, класифікатор NUREG-7007 виступає універсальним методологічним інструментом, що дозволяє як інженерам-проєктувальникам, так і регуляторам здійснювати обґрунтовану оцінку заходів диверсності та гарантувати високий рівень надійності систем безпеки на АЕС.

1.4.2 Моделі багатоверсійних систем

Основою створення відмовостійких систем із використанням ПЛІС (FPGA) є поєднання кількох рівнів диверсності: архітектурної, алгоритмічної, часової та апаратної. Використання FPGA дозволяє реалізовувати ці принципи значно гнучкіше, ніж у традиційних обчислювальних системах, завдяки можливості перепрограмування і паралельного виконання обчислень.

1. **Апаратна диверсність**. FPGA дають змогу створювати кілька незалежних конфігурацій апаратних блоків, що виконують однакові функції різними способами. Наприклад, для одного алгоритму можуть бути синтезовані два різні логічні проєкти, які базуються на різних структурах — один із використанням LUT-блоків, інший із жорсткими DSP-ядрами. Це зменшує ймовірність одночасного виникнення однакових помилок.

2. **Алгоритмічна диверсність**. Критичні функції можуть реалізовуватися альтернативними алгоритмами. Для прикладу, обчислення контрольних сум чи шифрування можуть виконуватись різними математичними методами, що знижує ризик спільної вразливості. FPGA забезпечує можливість швидкого паралельного запуску цих альтернатив, що суттєво зменшує час перевірки.

3. **Часова диверсність**. Одні й ті самі функції можуть виконуватися із часовим зсувом або з використанням різних тактових доменів. Це дозволяє виявляти збої, спричинені короткочасними перешкодами (наприклад, від радіаційних впливів), і підвищує стійкість системи до Single Event Upsets (SEU).

4. **Архітектурна диверсність**. В системах безпеки АЕС особливо актуально використання N-modular redundancy, де кілька незалежних каналів

формують рішення на основі принципу більшості. На FPGA це реалізується у вигляді паралельних обчислювальних модулів із власними схемами перевірки. Важливим є також поєднання FPGA з іншими типами процесорів (heterogeneous computing), що створює додатковий рівень захисту.

5. **Вбудований моніторинг і само-відновлення**. Сучасні FPGA підтримують часткове перепрограмування (partial reconfiguration), що дозволяє оперативно замінювати пошкоджені модулі без зупинки всієї системи. Це відкриває можливість побудови само-відновлювальних систем, де виявлення збоїв автоматично ініціює перезавантаження або заміну дефектної частини схеми.

Таким чином, принципи побудови диверсних систем на FPGA включають багаторівневу організацію захисту, що базується на поєднанні різних форм диверсності та можливості швидкої реконфігурації. Це робить FPGA одним із ключових інструментів для створення надійних систем управління, зокрема у сфері ядерної енергетики, авіації та військових застосувань .

1.4.3 Методи оцінювання диверсності та функційної безпечності

Методи оцінювання диверсності та функційної безпечності багатоверсійних систем базуються на поєднанні кількісних та якісних підходів, що дозволяють аналізувати вплив різних видів надмірності на надійність і безпеку функціонування інформаційно-керуючих систем.

До основних підходів належать:

- метричні методи – кількісна оцінка відмінностей між версіями програмного чи апаратного забезпечення (порівняння коду, алгоритмів, архітектурних рішень);
- експертні методи – використання знань і досвіду фахівців для якісного визначення рівня диверсності;
- ймовірнісні методи – оцінка ймовірності виникнення відмов за загальною причиною, застосування статистичних моделей;

– байєсівський аналіз – побудова умовних імовірнісних моделей для уточнення оцінки ризиків при наявності часткових даних.

У сучасних дослідженнях велика увага приділяється логіко-імовірнісним методам, побудові Марковських моделей, а також використанню імітаційного моделювання для аналізу складних динамічних сценаріїв роботи систем.

Вагомий внесок у розвиток цієї проблематики зроблено українськими та закордонними науковцями. Зокрема, колективи ХАІ та НВП «Радій» (А.В. Горбенко, Є.В. Брежнєв, Ю.О. Бєлий, В.О. Куланов, В.І. Дужий) сформували підхід до оцінювання диверсності із застосуванням метричних, експертних та ймовірнісних методів у системах керування. Дослідження провідних європейських і британських шкіл (Bev Littlewood, Lorenzo Strizini, Peter Popov – City, University of London) стали основою для розроблення моделей оцінювання функційної безпечності на основі байєсівського аналізу. Водночас у працях В.О. Бутенка, О.М. Одарущенка, Ю.Л. Поночовного (ХАІ, НВП «Радій») та Б.Ю. Волочого (НУ «Львівська політехніка») розроблено підходи для оцінювання безпечності багатоверсійних систем атомних електростанцій, авіаційних та залізничних ІКС із використанням логіко-імовірнісних методів і моделей Маркова. Значний внесок у застосування імітаційного моделювання та формальних методів перевірки зробила Francesca Saglietti (University of Erlangen).

Методи оцінювання диверсності та функційної безпечності дозволяють визначати ступінь стійкості багатоверсійних систем до відмов за загальною причиною та оцінювати рівень зниження ризиків у критичних застосуваннях. Разом з тим, сучасні методики часто орієнтовані на класичні архітектури й недостатньо враховують новітні аспекти — мультиплатформність, проблеми синхронізації та загрози кібербезпеки. Це обґрунтовує необхідність подальшого вдосконалення методів оцінювання та розроблення інтегрованих підходів.

1.4.4 Методи забезпечення функційної безпечності ІКС АЕС на програмовних платформах

Розроблення багатоверсійних систем потребує не лише оцінки досягнутого рівня диверсності, але й методів цілеспрямованого вибору її видів на етапах проєктування. Вибір конкретних стратегій диверсифікації залежить від класу критичної системи (атомні електростанції, авіаційні системи, залізнична автоматика), вимог до функційної безпечності, а також від балансу між вартістю впровадження та очікуваним зниженням ризику відмов за загальною причиною.

До основних підходів належать:

- графові методи – подання простору можливих варіантів диверсифікації у вигляді графів, що дозволяє здійснювати пошук оптимальних комбінацій за заданими критеріями;
- метричні методи – використання кількісних показників для порівняння різних версій та визначення ступеня їх незалежності;
- логіко-лінгвістичні методи – формалізація опису властивостей версій за допомогою лінгвістичних змінних, що зручно для експертних систем;
- кейс-орієнтовані методи та технології – використання накопичених даних про успішні й неуспішні приклади впровадження диверсності для рекомендацій при проєктуванні нових систем.

Такі підходи забезпечують можливість не лише оцінити наявну диверсність, але й систематизовано обирати оптимальні стратегії під час створення складних інформаційно-керуючих систем.

У роботах дослідників ХАІ та НВП «Радій» [31–33] запропоновано комплексні інструментальні засоби для аналізу функційної та кібербезпеки багатоверсійних систем. Зокрема, розроблено методики побудови фасетно-ієрархічних класифікаторів диверсності та формальних операцій з ними, що дозволяє формувати раціональні варіанти диверсифікації на ранніх стадіях проєктування.

Значний внесок зробили й закордонні науковці: Alexandr Romanovsky (Newcastle University) [34; 35] запропонував підходи до аналізу надійності багатоверсійних програмних систем у контексті відмов за загальною причиною, Sanjit Mitra (University of California) розробив методи побудови диверсифікованих архітектур для критичних застосувань, а Richard Wood (University of Tennessee) дослідив порівняння класифікаторів диверсності у відповідності до міжнародних стандартів (NUREG-7007, IEC 26262, CENELEC EN 50128).

Таким чином, сучасні методи та інструменти вибору диверсифікаційних рішень поєднують аналітичні, формальні та кейс-орієнтовані підходи, що дає змогу враховувати як технічні, так і організаційні чинники безпечності. Проте вони потребують подальшого удосконалення з урахуванням новітніх викликів: мультиплатформного середовища, високої інтеграції програмно-апаратних комплексів, а також зростаючого значення кіберзагроз.

1.5 Обгрунтування мети, задач та методики досліджень

Метою дослідження є формування цілісного науково-методичного апарату для вибору видів диверсності в інформаційно-керуючих системах атомних електростанцій на програмовних платформах (з урахуванням FPGA/SoC), який поєднує графові та матричні моделі диверсності, метрики диверсності та відносної вартості, методи програмно-апаратної диверсності і тестування на основі моделей, а також цикл практичної валідації. Підхід базується на вимогах NUREG/CR-7007 та відповідних міжнародних стандартах IEC/IAEA [NUREG/CR-7007; IEC 61513; IEC 60880; IEC 62138; IEC 61508].

Ключовою ідеєю є консолідація нормативної бази, проєктного досвіду й наукових публікацій у вигляді розширеного класифікатора та графової моделі диверсності, на які накладаються метричні та алгоритмічні процедури вибору. Це дає можливість:

1. Формалізувати різновиди диверсності на декількох рівнях (апаратний, програмний, алгоритмічний, процесний та інші).

2. Оцінювати ефекти диверсності та їхню відносну вартість у єдиній шкалі.

3. Будувати стратегії та алгоритми вибору, узгоджені з обмеженнями ІКС АЕС (вимоги безпеки, життєвий цикл, сертифікація).

1.5.1 Мета і задачі досліджень

Метою дисертаційної роботи є створення методу вибору та реалізації видів диверсності для інформаційно-керуючих систем атомних електростанцій на базі програмовних платформ (FPGA/SoC), які дозволяють зменшити ймовірність відмов за загальною причиною, підвищити функційну безпечність та надійність ІКС, а також оптимізувати відносні витрати на впровадження диверсних рішень.

Сучасні ІКС АЕС працюють за жорстких нормативних вимог і в умовах високих ризиків виникнення загальних причин відмов. Застосування програмовних платформ відкриває можливості для реалізації різних форм диверсності (алгоритмічної, архітектурної, мовно-інструментальної, просторової тощо). Проте збільшується і складність їх оцінки, оскільки кількість можливих комбінацій рішень зростає в геометричній прогресії. Це вимагає розроблення узгодженої методики, яка поєднує класифікацію, математичні моделі, експертні оцінки, стратегії вибору та експериментальну перевірку.

Рисунок 1.5 відображає загальну методику проведення досліджень. Вона починається з аналізу нормативних документів, наукових джерел і практичного досвіду експлуатації, що визначає початкові умови і фактори. Далі формується розширений класифікатор диверсності, який у подальшому представляється у вигляді графової моделі. Наступні блоки схеми включають метод вибору диверсності, методи програмно-апаратної реалізації та метод тестування на основі моделей. Праворуч розташовано результати — програмно-апаратні рішення, тестування і впровадження у життєвий цикл. Внизу показано ітераційний зворотний зв’язок “оцінка виконання вимог і корекція”, що підкреслює

циклічність досліджень. Саме ця схема визначає логіку і взаємозв'язок усіх задач, сформульованих у цьому підрозділі.

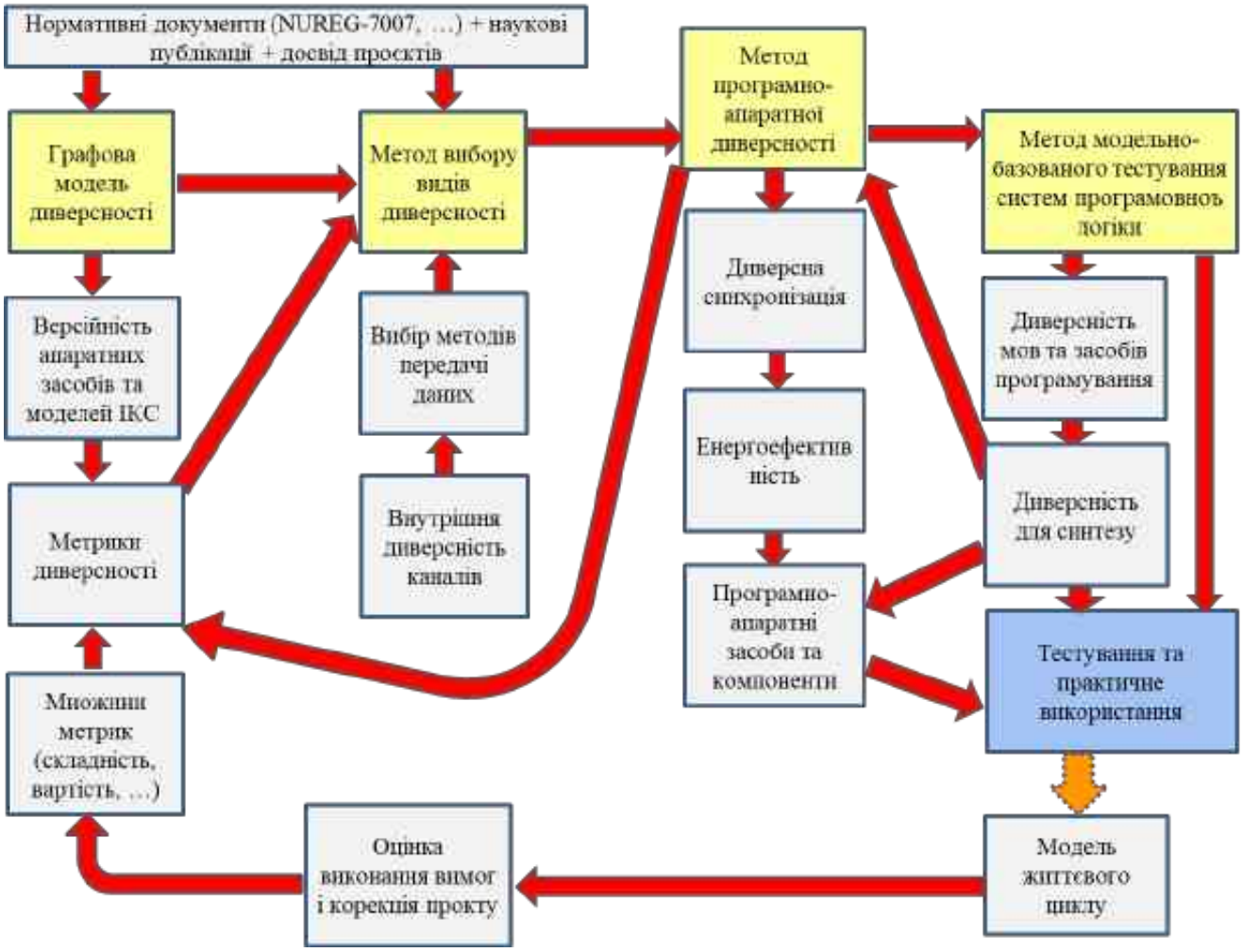

Рисунок 1.5 - Загальна методика вибору видів диверсності.

Задачі дослідження:

1. **Аналіз факторів і вимог.** Проаналізувати міжнародні стандарти, рекомендації та національні нормативи, а також досвід експлуатації ІКС АЕС. Виявити фактори, що найбільше впливають на виникнення відмов за загальною причиною: використання ідентичних мов опису, компіляторів, бібліотек IP-ядер, а також повторюваних алгоритмічних рішень. Результатом має стати набір вимог і обмежень, які далі перетворюються у критерії та метрики (Рисунок 1.5, ліва частина).

2. **Формування розширеного класифікатора диверсності.** Сформувати багаторівневий класифікатор видів диверсності, що враховує як відомі рішення (NUREG-7007), так і нові підходи, характерні для FPGA/SoC: структурно-просторову диверсність, диверсну синхронізацію, внутрішню диверсність каналів, версійність IP-ядер та інші. Класифікатор є основою для побудови графової моделі (Рисунок 1.5, центральна частина).

3. **Графова модель диверсності.** Подати класифікатор у вигляді орієнтованого графа, де вершини відповідають видам і підвидам диверсності, а ребра відображають відношення ієрархії, сумісності чи конфлікту. Це дозволяє формувати шляхи — комбінації диверсності, які будуть предметом подальшого аналізу.

4. **Матрична модель і метрики.** Розробити систему двох структурних матриць графової моделі класифікатора:

- булеву матрицю суміжності, що кодує наявність орієнтованих зв’язків між вершинами класифікатора (види/підвиди диверсності) і слугує основою для аналізу досяжності та підрахунку шляхів;
- матрицю інцидентності «ребро–вершина», у якій для кожного ребра фіксується вершина-джерело та вершина-приймач. Ця матриця забезпечує перевірку коректності структури, вибір підграфів/шляхів і подальшу комбінаторну оптимізацію;
- паралельно формуються вектори метрик диверсності та відносної вартості, що отримуються за результатами експертного оцінювання та нормуються у діапазоні [0;1] для коректного багатокритеріального порівняння й подальшої оптимізації на шляхах графа.

5. **Експертна процедура оцінювання.** Розробити алгоритм, який переводить систему рангового оцінювання експертами видів диверсності у нормовані числові оцінки з урахуванням потужності множини і рівня у графі. Це дозволяє об’єктивізувати суб’єктивні думки та формалізувати їх у вигляді даних для матричної моделі.

6. **Метод вибору видів диверсності.** Розробити стратегії вибору з урахуванням різних критеріїв: мінімізація вартості при досягненні заданого рівня диверсності; максимізація диверсності при обмеженому бюджеті; пошук компромісних рішень. Рисунок 1.5 містить цей блок у центральній частині, як “Метод вибору видів диверсності”.

7. **Методи реалізації програмно-апаратної диверсності.** Розробити практичні методи реалізації на FPGA/SoC: диверсну синхронізацію, структурно-просторову диверсність, внутрішню диверсність каналів та версійність IP-ядер. Цей блок центральної частини безпосередньо пов’язаний із правою частиною — програмно-апаратними засобами та компонентами (Рисунок 1.5).

8. **Тестування на основі моделей (MBT).** Інтегрувати методи MBT у процес досліджень: створити поведінкові моделі, згенерувати тести, перенести їх у VHDL і перевірити на FPGA. Рисунок 1.5 містить це як окремий блок, що зв’язує методи та результати тестування.

9. **Експериментальна перевірка та інтеграція у життєвий цикл.** Виконати експериментальні перевірки на реальних платформах, оцінити отримані показники і інтегрувати методику в життєвий цикл ІКС АЕС. Цей етап відповідає правій частині схеми (Рисунок 1.5), де результати тестування замикаються на корекцію вимог.

Рисунок 1.6 виконує роль причинно-наслідкової «карти» підрозділу: він показує, як зовнішні фактори породжують клас рішень, які, у свою чергу, дають вимірювані наслідки, формулюють задачі дослідження і приводять до результатів — теоретичних та практичних. Логіка композиції побудована зліва направо (від причин до результатів) і зверху вниз (від загального до конкретного), а замикання стрілок унизу відображає ітераційність процесу.

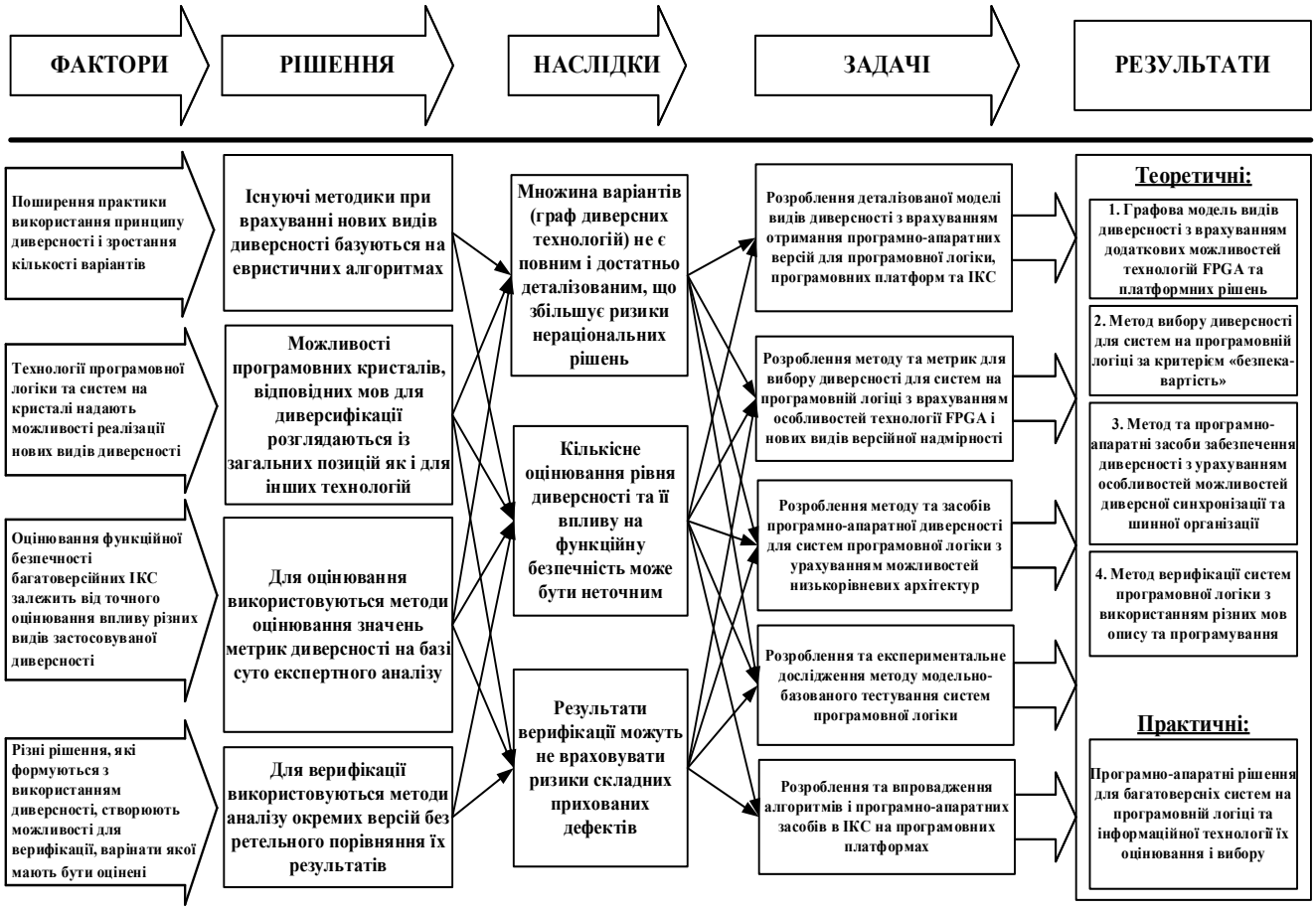

Рисунок 1.6 - Логічна структура взаємозв’язків між факторами, рішеннями, задачами і результатами досліджень

Поданий рисунок узагальнює системні взаємозв’язки між вихідними факторами, наявними рішеннями, очікуваними наслідками, поставленими задачами та кінцевими результатами дослідження.

По-перше, фактори нормативного, ризик-орієнтованого, технологічного та процесного характеру обґрунтовують необхідність диверсифікації як засобу підвищення функційної безпечності та сертифікаційної придатності ІКС.

По-друге, аналіз існуючих підходів виявив їхню обмеженість — опора на евристику, загальні трактування технологій програмовної логіки, надмірна залежність від експертних оцінок і відсутність інтегрованої порівняльної верифікації версій. Це визначає дослідницькі прогалини.

По-третє, запропоновані у роботі рішення (графова модель видів диверсності, матричний апарат нормованих метрик, методи програмно-апаратної

диверсності, модельно-базоване тестування) створюють інструментальну основу для кількісного аналізу, усунення загальних причин відмов і забезпечення відтворюваної верифікації.

По-четверте, реалізація відповідних задач дослідження дозволяє трансформувати ідентифіковані фактори у практичні методики: від класифікації та моделювання видів диверсності — до алгоритмів вибору, методів верифікації й експериментальних досліджень на програмовних платформах.

Нарешті, досягнуті результати структуровано у двох площинах:

- **теоретичні** (формалізація графової та матричної моделей, методи вибору та верифікації),
- **практичні** (програмно-апаратні рішення і методичні засоби оцінювання, придатні до інтеграції у життєвий цикл ІКС).

Таким чином, Рисунок 1.6 відображає логіку дослідження: від виявлених факторів і проблем — через методологічні рішення та задачі — до результатів, які одночасно мають наукову новизну та прикладне значення.

1.5.2 Обгрунтування математичного апарату та методики досліджень

Для досягнення поставленої мети та вирішення сформульованих задач необхідно розробити узгоджений математичний апарат і методику досліджень, які дозволять формалізувати процес вибору та реалізації диверсності для ІКС АЕС на FPGA/SoC.

Вибір видів диверсності — це багатокритеріальна задача, де необхідно збалансувати протилежні вимоги: досягти максимальної незалежності каналів і зменшити ризик відмов за загальною причиною, та одночасно не перевищити припустимий рівень складності та вартості. Традиційні якісні підходи (словесні класифікації, евристичні рекомендації) у цьому випадку недостатні. Саме тому потрібна формальна основа, що поєднує графові та матричні моделі, систему метрик, експертні процедури і алгоритми багатокритеріального вибору. Рисунок 1.5 відображає методику побудовану як послідовність взаємопов’язаних етапів:

1. Вхідні дані (нормативні вимоги, наукові публікації, практичний досвід, множини метрик).

2. Формування класифікатора диверсності й його подання у вигляді графової моделі.

3. Побудова системи матриць і нормованих метрик.

4. Розроблення методу вибору диверсності з використанням експертних процедур і критеріїв оптимізації.

5. Розроблення методів програмно-апаратної диверсності.

6. Використання МВТ для перевірки.

7. Експериментальна перевірка й інтеграція у життєвий цикл з ітераційною корекцією.

Таким чином, Рисунок 1.5 у цьому підрозділі виступає не просто загальною картою, а фактично блок-схемою алгоритму досліджень, де кожен етап відповідає певному математичному інструменту.

Основні компоненти математичного апарату:

1. **Теоретико-множинне подання класифікатора.** Класифікатор диверсності представляється як множина елементів, де кожен елемент — конкретний вид чи підвид. Додатково вводяться відношення (сумісність, конфлікт, ієрархія). Таке подання дозволяє переходити до графових моделей.

2. **Графова модель.** Формально класифікатор описується як орієнтований граф, де вершини відповідають видам диверсності, а ребра відображають відношення. Граф дає змогу будувати шляхи, які відображають допустимі комбінації. Саме з цими шляхами працюють алгоритми вибору.

3. **Матрична модель.** Для кількісного аналізу розробляється система двох матриць, що описують структуру графа класифікатора видів і підвидів диверсності:

- матриця суміжності, яка відображає наявність орієнтованих зв’язків між вершинами класифікатора та дозволяє визначати множину усіх можливих шляхів;

- матриця інцидентності «ребро–вершина», яка формалізує зв’язки між елементами графа й використовується для подальших обчислень метрик на рівні шляхів. Паралельно з ними вводяться вектори нормованих метрик (диверсності та відносної вартості), значення яких отримуються шляхом експертного оцінювання та масштабуються у діапазон [0;1], що забезпечує коректність багатокритеріального порівняння альтернатив.

4. **Метрики диверсності та вартості.** Для кожного виду визначаються дві ключові величини: метрика диверсності та метрика відносної вартості. Вони формуються за допомогою експертного оцінювання та подальшого нормування. Допоміжні метрики уточнюють оцінку і знижують вплив суб’єктивності.

5. **Експертне оцінювання.** У задачах, де формальні моделі не дають однозначної відповіді, використовується експертна процедура. Експерти присвоюють ранг кожному виду диверсності за критеріями диверсності та вартості. Ранги усереднюються, нормуються з урахуванням потужності множини, після чого стають основою для матриць і подальших розрахунків.

6. **Алгоритми вибору.** Вибір шляхів у графі виконується за кількома стратегіями: мінімізація вартості при заданій диверсності; максимізація диверсності при заданій вартості; пошук компромісів. Це завдання оптимізації на графі з вагами, де можуть застосовуватися як класичні алгоритми пошуку, так і евристичні методи.

7. **Тестування на основі моделей**. MBT виступає окремим інструментом: поведінкові моделі системи подаються як кінцеві автомати, для яких генеруються тестові послідовності. Альтернативні моделі виконують роль «диверсних версій» для взаємної перевірки.

8. **Експериментальна перевірка.** Заключним етапом є відображення результатів на реальних апаратних платформах. На даному етапі математичний апарат впроваджується на практиці: обчислені метрики порівнюються з фактичними результатами тестування, а корекція здійснюється шляхом повернення до класифікатора та матричних моделей.

У контексті цього підрозділу Рисунок 1.6 підкреслює зв’язок між апаратом і методикою. Фактори задають вимоги, які не можна обійти (CCF, архітектура FPGA, нормативи). Рішення — це безпосередньо ті математичні інструменти, що розробляються (класифікатор, граф, матриці, метрики, МВТ). Наслідки — поява можливості обґрунтованого вибору й контролю витрат. Задачі — конкретні кроки реалізації методики. Результати — теоретичні (моделі, алгоритми) і практичні (методики, інструменти, компоненти).

Таким чином, Рисунок 1.5 та Рисунок 1.6 виступають візуальним підтвердженням методики: перший показує послідовність етапів і зв’язки між ними, другий — логіку причин і наслідків, які обумовлюють вибір саме такого математичного апарату.

У цьому підрозділі обґрунтовано вибір математичного апарату та методики досліджень. Використання множин, графів і матриць дозволяє формально подати класифікатор диверсності та операції над ним. Система нормованих метрик і експертних процедур забезпечує кількісну порівнянність альтернатив.

Алгоритми оптимізації й МВТ формують основу практичного вибору та перевірки. Рисунок 1.5 та Рисунок 1.6 відображають інтеграцію всіх компонентів у єдиний контур, що забезпечує замкнутий цикл від постановки факторів до практичних результатів і їх корекції.

1.6 Висновки до першого розділу

У результаті виконання першого розділу було проведено системний аналіз проблем і сучасних підходів до забезпечення функційної безпечності інформаційно-керуючих систем атомних електростанцій. Показано, що розвиток цифрових технологій, зокрема впровадження програмовних логічних інтегральних схем (ПЛІС) у критично важливих системах, суттєво підвищує продуктивність та гнучкість, але одночасно створює нові виклики, пов’язані з ризиком відмов за загальною причиною.

Аналіз нормативної та наукової бази продемонстрував, що традиційні методи підвищення надійності — резервування, фізичне рознесення каналів чи використання різних компонентів — не є достатніми для сучасних ПЛІС-орієнтованих систем. Особливості архітектури програмовної логіки призводять до появи корельованих збоїв, викликаних синхронними перемиканнями, впливом завад, помилками синхронізації та іншими спільними факторами.

Було встановлено, що базовий класифікатор диверсності (NUREG/CR-7007), який застосовується для оцінки багатоверсійних систем, потребує розширення та деталізації для адекватного відображення специфіки програмовних платформ.

В роботі обґрунтовано необхідність використання комплексного методичного апарату. Для незалежної перевірки програмно-апаратних рішень доцільним є впровадження методів тестування на основі моделей з використанням середовища ForSyDe, яке дозволяє створювати альтернативні програмні моделі та підвищувати достовірність верифікації.

У розділі сформульовано мету та задачі досліджень: розроблення моделей, методів і засобів забезпечення функційної безпечності ІКС АЕС на основі комбінованої диверсності та створення методики кількісної оцінки її ефективності. Поставлені задачі охоплюють аналіз нормативної бази, розроблення класифікаторів і графових моделей диверсності, обґрунтування математичного апарату та верифікацію запропонованих рішень на прикладах модулів безпеки.

Таким чином, результати першого розділу закладають науково-методичне підґрунтя для подальших досліджень, які спрямовані на розроблення моделей і методів вибору видів диверсності для систем на програмовних платформах, що розглядаються у другому розділі дисертації.

Матеріали розділу опубліковані в [5; 14; 28–30].

РОЗДІЛ 2
РОЗРОБЛЕННЯ МОДЕЛІ ТА МЕТОДУ ВИБОРУ ВИДІВ ДИВЕРСНОСТІ ДЛЯ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ НА ПРОГРАМОВНИХ ПЛАТФОРМАХ

2.1 Розширена класифікація видів диверсності для програмовних платформних рішень

Стандартні класифікаційні схеми диверсності, зокрема ті, що розроблені в рамках NUREG-7007 та подібних підходів, відіграли ключову роль у формуванні наукового підґрунтя для оцінки надійності та функційної безпечності систем, важливих для ядерної енергетики. Вони дозволили впорядкувати наявні знання, задати базові категорії диверсності та застосовувати їх у практичних проєктах.

Однак сучасний рівень розвитку програмно-керованих рішень — насамперед використання ПЛІС, систем-на-кристалі (SoC) та високопродуктивних мікропроцесорних платформ — призвів до появи нових викликів. Ці технології відкрили широкі можливості для створення диверсних архітектур, але водночас ускладнили процес класифікації та оцінювання. Традиційні підходи виявилися недостатньо деталізованими, оскільки не враховували особливостей програмовної логіки, паралельних обчислень, синхронізації між апаратними та програмними компонентами, а також методів інтеграції диверсних засобів на рівні одного кристала.

Тому виникає об’єктивна потреба у розширенні класифікаційної схеми диверсності. Це дозволить:

- забезпечити більш точне групування видів і підвидів диверсності;
- врахувати специфіку FPGA та SoC платформ;
- створити умови для формалізованої оцінки ефективності диверсних рішень;
- розробити підґрунтя для побудови графових моделей, що відображають взаємозв’язки між видами диверсності.

2.1.1 Принципи розширення класифікаційної схеми

Розширення класифікації диверсності є необхідним кроком для усунення обмежень, притаманних традиційним схемам. Практика показує, що застосування стандартного класифікатора супроводжується низкою проблем, які значно ускладнюють як теоретичні дослідження, так і практичну реалізацію диверсних рішень.

Основні проблеми можна узагальнити таким чином:

- нечіткість меж між окремими видами диверсності, що ускладнює однозначну класифікацію;
- складність у визначенні рівня ефективності при комбінуванні кількох підходів;
- відсутність урахування специфіки сучасних програмовних платформ;
- неможливість автоматизованого аналізу через відсутність формалізованих зв’язків між категоріями.

Подолання вказаних обмежень вимагає розширення класифікаційної схеми на основі таких принципів:

1. Інтеграція з існуючими стандартами – нова класифікація не повинна суперечити базовим документам (NUREG-7007, IEC 61513, IEC 60880), а має доповнювати їх і розвивати;
2. Мульти-рівнева деталізація – введення додаткових (чотирьох) рівнів класифікації (вид → підвид → конкретизація → приклад застосування), що забезпечує гнучкість та точність;
3. Урахування специфіки програмовних платформ – виділення окремих категорій, пов’язаних із синтезом логіки, використанням різних мов опису апаратури, інструментів синтезу та симуляції;
4. Формалізація та можливість графового подання – перехід від ієрархічного списку до графової моделі, що дозволяє описати зв’язки між різними видами диверсності та виявити комбінаційні ефекти;

5. Орієнтація на практичну оцінку ефективності – розширений класифікатор має стати інструментом не лише для теоретичного аналізу, а й для прикладних досліджень і практичного застосування у проєктах безпеки.

Таким чином, розширена класифікаційна схема виступає логічним продовженням традиційних підходів і створює основу для побудови чотирирівневої моделі диверсності.

2.1.2 Чотирирівнева класифікація видів і підвидів диверсності

Класифікація диверсності в інформаційно-керуючих системах традиційно ґрунтується на дворівневому підході, закладеному у NUREG/CR-7007. Такий підхід є зручним для загального опису, оскільки він окреслює основні види диверсності та їхні підтипи у стислому вигляді, надаючи просту модель для аналізу незалежності компонентів. Його перевага полягає у зрозумілості та універсальності: будь-який інженер або аудитор може швидко орієнтуватися в ключових аспектах різнорідності, що знижує ймовірність виникнення загальних відмов. Проте дворівнева модель має істотні обмеження. Вона не враховує деталізованого переліку сучасних технологічних рішень, не дає змоги простежити логічні та функціональні залежності між різними підвидами диверсності, а також не забезпечує достатнього рівня формалізації для використання у складних багатоверсійних системах, особливо в таких чутливих галузях, як атомна енергетика. Через це постає потреба у розширенні базового класифікатора.

У даній роботі класифікатор розширено до чотирьох рівнів (Рисунок 2.1) [29; 36]. Додатково до основних двох рівнів NUREG/CR-7007 введено ще два: третій рівень деталізації охоплює підвиди диверсності, що відображають конкретні технічні рішення, зокрема різні типи обладнання, процесорів, алгоритмів, методів і процесів розроблення; четвертий рівень деталізації описує архітектури інтегральних компонентів, різновиди інтерфейсів і протоколів обміну даними, а також інші низькорівневі елементи. Важливою характеристикою запропонованого класифікатора є його відкритість: він може розширюватися у

міру появи нових технологій, залишаючись узгодженим і системним. Це дозволяє враховувати динаміку розвитку таких напрямів, як FPGA, системи безпеки нового покоління чи розподілені IoT-платформи.

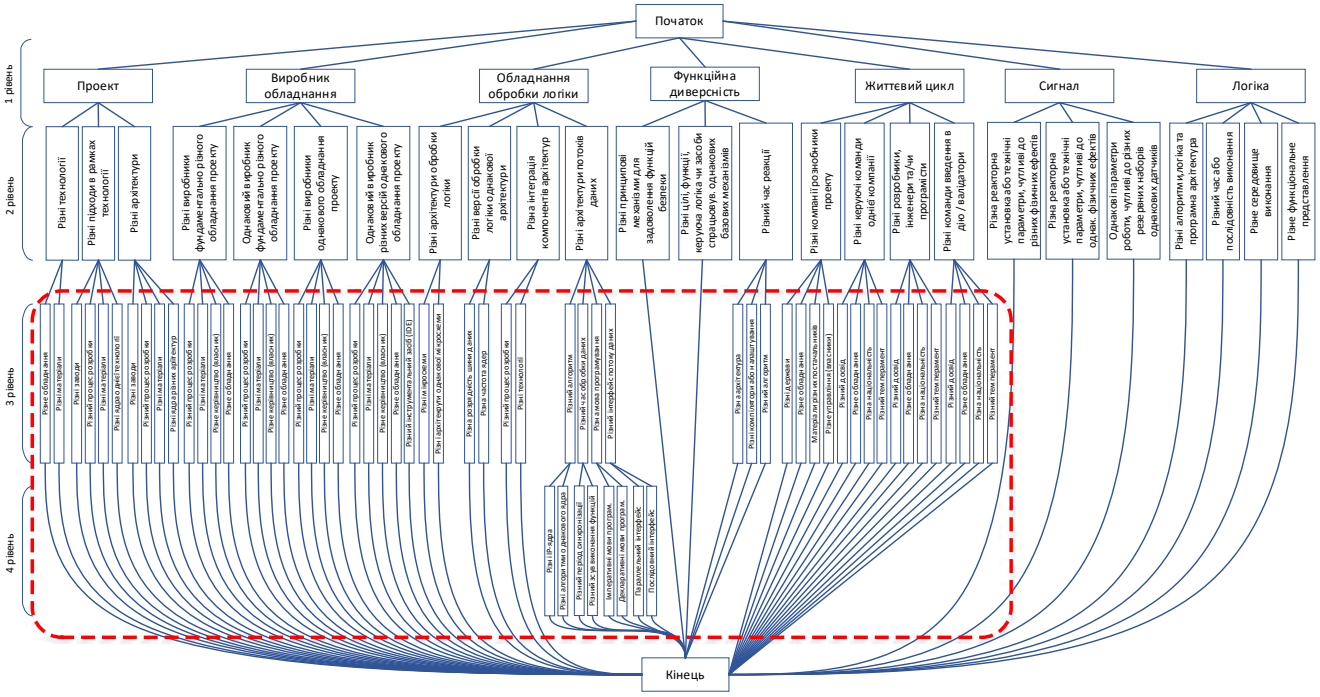

Рисунок 2.1 - Розширена модель видів диверсності [36; 37]

Ключовим кроком стала трансформація класифікатора у графову форму. На відміну від деревовидної структури, яка лише фіксує ієрархію понять, графова модель дозволяє відобразити множинність логічних і технологічних залежностей між підвидами диверсності. Вершини такого графа відповідають конкретним категоріям чи підкатегоріям, а дуги відображають взаємозв'язки між ними. Це дає можливість не тільки виконувати статичну класифікацію, а й проводити кількісні розрахунки. Зокрема, стає можливим визначати інтегральні метрики диверсності та вартості для будь-якої конфігурації системи шляхом агрегування показників уздовж маршруту у графі. Таким чином, від описового підходу здійснено перехід до формалізованого кількісного аналізу, що дозволяє автоматизувати вибір архітектур на основі співвідношення «вартість – диверсність».

Оригінальна побудова орієнтованого графа класифікатора становить собою новизну, оскільки подібна форма представлення у наукових публікаціях раніше не зустрічалася. Вона забезпечує високу гнучкість у моделюванні та

масштабуванні (нові вершини й зв’язки можна додавати без порушення цілісності системи). Це критично важливо для галузей, де розвиток технологій є особливо динамічним, а ризики відмов — надзвичайно високими. Саме у таких сферах формалізована багаторівнева й графова модель класифікатора диверсності може стати практичним інструментом як для проєктування, так і для оцінки надійності вже існуючих систем.

У попередньому підрозділі було показано класифікацію видів диверсності у вигляді ієрархічного дерева. Така форма подання є наочною та зрозумілою, однак має низку суттєвих обмежень. По-перше, дерево не дозволяє врахувати перехресні залежності між гілками, які неминуче виникають у реальних системах. По-друге, у класичному вигляді воно лишається описовим інструментом і не надає можливостей для математичного аналізу. По-третє, дерево складно масштабувати: додавання нових видів чи підвидів диверсності часто призводить до порушення узгодженості всієї структури. Щоб подолати ці обмеження, класифікатор було перетворено у орієнтований граф. Це дало змогу не лише зберегти ієрархічні відношення, але й підсилити їх формалізованими математичними властивостями. У такій моделі кожен вид чи підвид диверсності відповідає окремій вершині, а відношення «належності» або «залежності» реалізуються у вигляді ребер. Таким чином, класифікатор перетворюється з інструмента для опису у повноцінну математичну модель, що піддається алгоритмічній обробці та чисельним оцінкам.

2.2 Графова модель видів диверсності

Класична класифікація диверсності у вигляді ієрархічного дерева добре працює як засіб інвентаризації понять (типів і підтипів диверсності) та базової навігації між ними. Проте дерево за своєю природою є статичним і одновимірним: воно відображає лише відношення «загальне - часткове» і погано фіксує перехресні впливи, які неминуче виникають у реальних інформаційно-керуючих системах. Наприклад, вибір мов опису апаратури та інструментів

синтезу одночасно впливає і на програмну, і на апаратну диверсність; рішення щодо носіїв конфігурації або шинних протоколів накладають обмеження на алгоритмічні та організаційні практики забезпечення незалежності каналів. У дереві ці взаємозв'язки або губляться, або змушують дублювати вузли, що руйнує узгодженість структури.

2.2.1 Перехід від класифікатора диверсності до графової моделі

Щоб перейти від описового до аналітичного рівня, класифікатор трансформується в орієнтований граф. У цій постановці кожний вид чи підвид диверсності моделюється вершиною графа, а ребра кодують не лише ієрархічні відносини належності, а й причинно-наслідкові та технологічні залежності (сумісність, взаємні обмеження, підсилення або, навпаки, «анулювання» ефектів) [38]. Граф, на відміну від дерева, допускає множинні батьківські зв'язки та довільну кількість перехресних ребер, завдяки чому ми отримуємо топологію, що адекватно відображає реальну багатовимірність предметної області.

Перехід від дворівневого уявлення до чотирирівневого відбувається за принципом послідовної декомпозиції понять із збереженням семантики «незалежності» між гілками. Перший рівень репрезентує базові категорії диверсності (апаратна, програмна, алгоритмічна, організаційна). Другий рівень деталізує ці категорії до підвидів (наприклад, для апаратної — різні сімейства ПЛІС/FPGA, типи контролерів, класи мікропроцесорів; для програмної — мови, ОС, компіляторні стеки). Третій рівень вводить конкретизатори, що істотно впливають на незалежність реалізацій: типи обладнання і процесорів, інструменти та процеси розроблення, варіанти носіїв конфігурації, профілі захисту каналів тощо. Четвертий рівень фіксує «дрібнозернисті» елементи: архітектури IP-ядер, типи інтерфейсів і транспортів, класи пам'яті конфігурації, способи синхронізації й вирівнювання часових доменів, механізми детекції/корекції помилок, протоколи обміну діагностичними подіями і таке інше.

Ключова перевага графового подання [39; 40] полягає в тому, що кожна траєкторія від кореня до листка відповідає конкретній конфігурації багатоверсійної системи (композиції рішень по рівнях), а перехресні ребра дозволяють явно моделювати обмеження/залежності між виборами на різних рівнях. Це знімає дві болючі проблеми «деревного» підходу:

1. неможливість формально порахувати «скільки» диверсності ми реально отримали в конкретній конфігурації;
2. неможливість об’єктивно порівнювати альтернативи за компромісом «різнорідність ↔ вартість/складність».

Після перетворення класифікатора в граф (Рисунок 2.2) стає можливим запровадити метрики вузлів (внесок у диверсність і «ціну» реалізації), визначити правила їх агрегації вздовж шляхів, а також оптимізувати вибір конфігурації за заданими критеріями. Важливо, що графова модель є відкритою: поява нових технологій (нові сімейства FPGA, IP-ядра, протоколи, засоби синхронізації) не вимагає перебудови всієї структури — достатньо додати вершини та відповідні ребра, зберігши семантику зв’язків. У підсумку ми отримуємо формальний апарат, який не лише охоплює дво- та чотирирівневу структуру диверсності, а й природно масштабується на подальші рівні деталізації без втрати узгодженості.

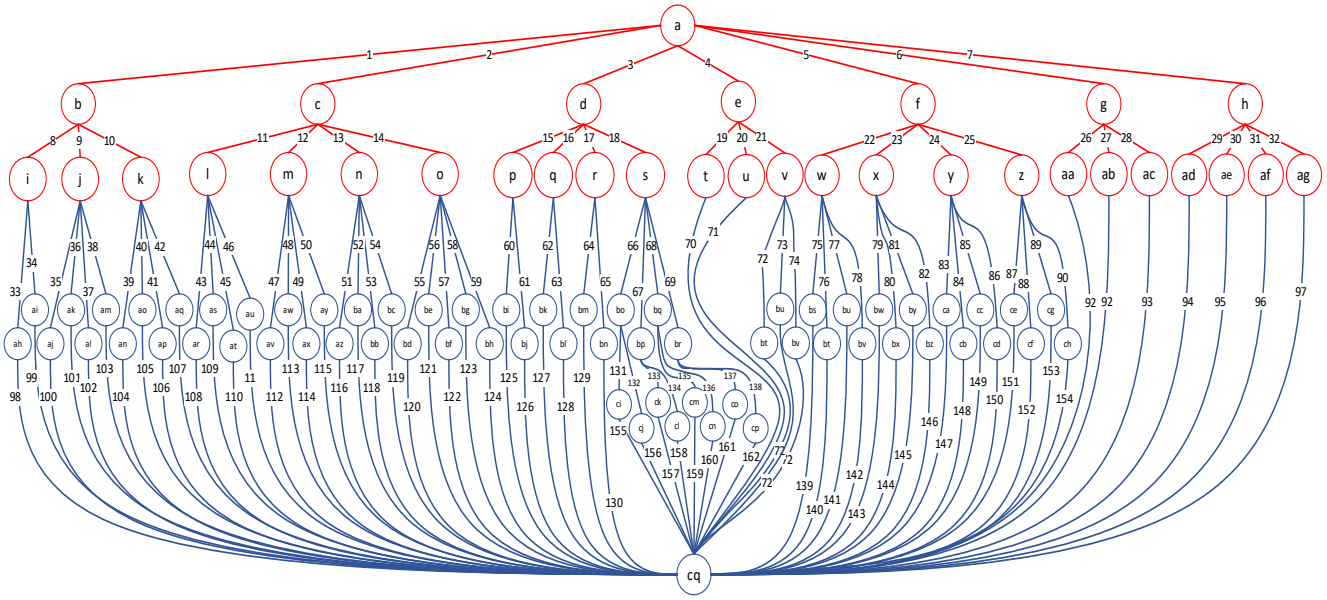

Рисунок 2.2 - Графова модель видів диверсності (чотири рівні)

2.2.2 Характеристики графової моделі

Для повноцінного використання графової моделі важливо не лише побудувати її структуру, але й визначити ключові характеристики, що описують внутрішні властивості графа. Ці параметри дозволяють кількісно оцінити складність та ієрархічну організацію видів диверсності, а також порівнювати розширену модель з базовим стандартом.

Формальне означення графа:

$$G = (V, E) \tag{2.1}$$

, де V – множина вершин графа (видів і підвидів диверсності), E – множина ребер, що описують відношення ієрархічної залежності видів і підвидів диверсності.

Потужності множин та характеристики відстаней для розширеного графа:

- Card V = 96 (для стандарту NUREG-7007 дорівнює 33);
- Card E = 162 (NUREG-7007 = 57);
- Діаметр графа $D(G) = \max \{d(u,v)|u,v \in V\} = 5$ (NUREG-7007 = 3);
- Ексцентриситет графа $\varepsilon(v) = \max\{d(v,u)|u \in V\} = 5$ (NUREG-7007 = 3);
- Радіус графа $R(G) = \min \{\varepsilon(v)|v \in V\} = 3$ (NUREG-7007 = 1).

Збільшення кількості вершин та ребер у розширеній моделі відображає зростання складності системи та ширший спектр врахованих видів диверсності. Якщо у базовому стандарті враховувалися лише 33 вершини, то в оновленій моделі їх у три рази більше, що вказує на значне розширення класифікатора.

Збільшення діаметра графа з 3 до 5 демонструє, що максимальна відстань між двома найвіддаленішими вершинами також зросла. Це означає більшу кількість рівнів у ієрархії та більшу складність взаємозв’язків між видами диверсності.

Показник ексцентриситету, який для розширеного графа дорівнює 5, свідчить про існування вершин, що мають найбільшу віддаленість від інших. Це зазвичай відображає появу нових гілок, пов’язаних із сучасними технологічними напрямами (FPGA, гетерогенні обчислення, тощо). Водночас збільшення радіуса

до 3 означає, що навіть найбільш «центральні» вершини стали віддаленішими від решти графа, що підкреслює ускладнення структури.

Таким чином, наведені характеристики підтверджують, що розширена графова модель має вищу структурну складність та краще відображає сучасні вимоги до проєктування багатоверсійних систем. Вона не лише деталізує класифікацію диверсності, але й забезпечує інструменти для кількісної оцінки та подальшої оптимізації архітектурних рішень.

2.3 Матрична модель видів диверсності

Попередня графова модель дозволяє відобразити ієрархію класифікатора диверсності у вигляді множини вершин і ребер. Однак граф ускладнений великою кількістю елементів, що робить його важким для подальших кількісних обчислень. Тому постає задача формалізації у вигляді матриць [41; 42], які дають змогу перейти до числових операцій, алгоритмізації та автоматизації вибору стратегій диверсифікації.

Використання матричної форми забезпечує:

- компактність представлення інформації;
- зручність у розрахунках за допомогою методів лінійної алгебри;
- можливість комп’ютерної обробки великих структур;
- аналітичні висновки щодо зв’язаності, варіативності та надмірності класифікатора.

Два типи матриць у моделі диверсності:

1. **Булева матриця суміжності (**Рисунок 2.3**)**. Ця матриця відображає взаємозв’язки між вершинами графа (|V|×|V|).

- Елемент $a_{ij} = 1$, якщо вершина i з’єднана з вершиною j.
- Елемент $a_{ij} = 0$, якщо прямого зв’язку немає.

Інтерпретація:

- Матриця показує, які підвиди диверсності є безпосередньо пов’язаними.

– Дає змогу знаходити замкнені підгрупи (кластеризацію).
– Дозволяє оцінити щільність зв’язків: чим більша частка ненульових елементів, тим сильніше переплетені різні види диверсності.

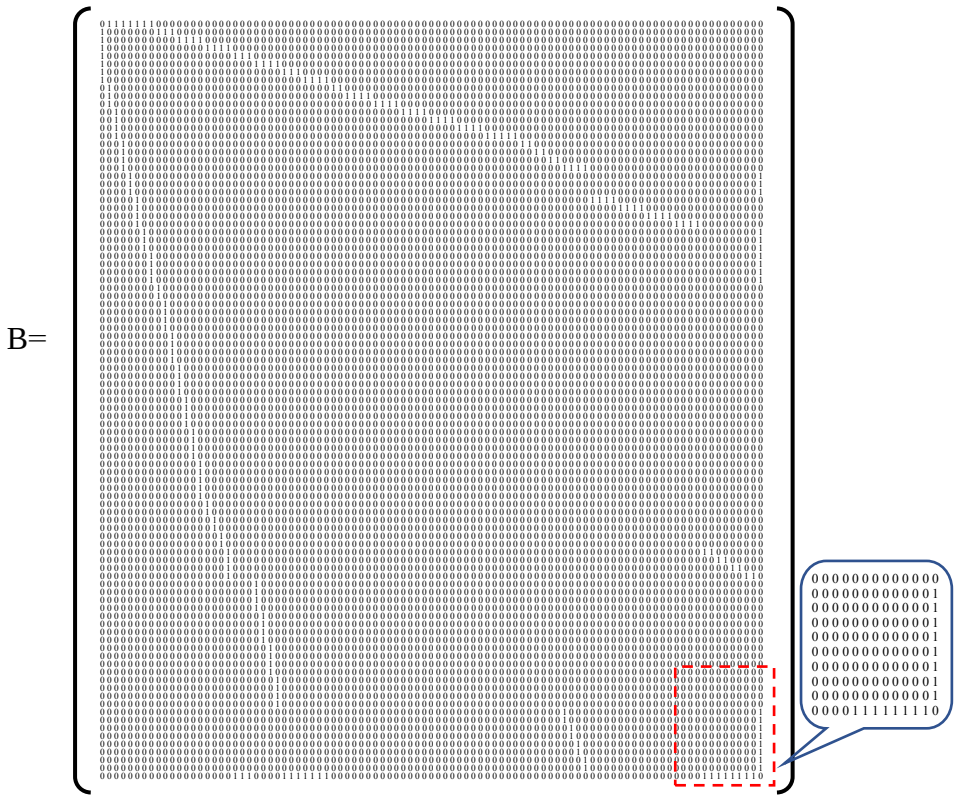

Рисунок 2.3 - Булева матриця суміжності класифікатора диверсності.

Корисність:

– Виявлення потенційних «вузьких місць», де занадто багато залежностей.
– Пошук надлишкових шляхів (деякі підсистеми дублюють одна одну).
– Оцінка складності архітектури: високий рівень зв’язності вказує на більшу кількість альтернатив.

Приклад: якщо в блоці апаратної диверсності (різні FPGA) більшість клітинок заповнені «1», це означає, що в межах цього класу існує багато альтернатив (виробники, архітектури, серії), що підвищує потенційний рівень надійності системи за рахунок різнорідності.

2. **Матриця «вершина–ребро» (інцидентності)** (Рисунок 2.4). Ця матриця розміром |V|×|E| описує зв’язки між вершинами та ребрами.

– Елемент $b_{ij} = 1$, якщо вершина і належить ребру j;

– Елемент $b_{ij} = 0$, якщо вершина не з’єднана з ребром.

Рисунок 2.4 - Матриця інцидентності «вершина–ребро» для класифікатора диверсності

Інтерпретація:

– Дозволяє оцінити, наскільки вершина «включена» у різні шляхи класифікації.

– Кожний рядок показує «ступінь участі» конкретного підвиду у системі.

– Дозволяє аналізувати «критичні вершини» — ті, що пов’язані з багатьма ребрами і формують вузли розгалуження.

Корисність:

- Аналіз ролі окремого підвиду диверсності (конкретної мови програмування чи архітектури FPGA).
- Визначення точок, видалення яких призводить до зменшення кількості можливих варіантів системи.
- Оцінка навантаженості компонентів: якщо один елемент пов’язаний із багатьма ребрами, він стає критичною точкою відмови.

Приклад: для підгрупи «різні алгоритми контролю цілісності» матриця «вершина–ребро» покаже, які алгоритми входять у які шляхи реалізації. Якщо певний алгоритм зустрічається надто часто, це свідчить про ризик зниження диверсності через надмірне використання одного методу.

Характеристики матричної моделі

- Розмір булевої матриці: 95×95.
- Розмір матриці інцидентності: 95×160.
- Щільність: не більше 15 % ненульових елементів (розрідженість).
- Структура блочна, кожен блок відповідає рівню класифікації (функціональна, апаратна, програмна, часова).

Матриці дозволяють проводити аналіз як всередині блоків, так і між ними. Використання та модернізація матриць:

1. Автоматизація аналізу:
   - використання алгоритмів пошуку сильно зв’язаних компонентів;
   - побудова маршрутів у просторі альтернатив (шляхи графа).
2. Моделювання впливу змін. Наприклад, додавання нових технологій → нові вершини та ребра → автоматичне розширення матриць;
3. Оптимізація стратегій:
   - визначення найкоротших шляхів диверсифікації;
   - аналіз вартості та вигоди при виборі конкретних альтернатив.
4. Модернізація:
   - введення вагових коефіцієнтів у матриці (не лише 0/1), щоб враховувати вартість, надійність чи складність;

– застосування багатокритеріальних методів для вибору оптимальних варіантів.

Булева матриця суміжності в поєднанні з матрицею «вершина–ребро» утворюють універсальну основу для аналізу класифікатора диверсності: перша описує структурні зв’язки між підвидами диверсності, друга ж деталізує участь окремих елементів у шляхах формування системи.

Разом вони дозволяють:

– виявляти критичні вузли, надлишкові або слабкі місця;
– автоматизувати процес вибору альтернатив;
– моделювати наслідки втрати або заміни певних компонентів;
– забезпечувати кількісний аналіз вартості та надійності багатоверсійних систем.

Таким чином, матрична модель є ключовим інструментом переходу від якісного опису диверсності до кількісного, формалізованого аналізу та оптимізації у системах критичного призначення.

2.4 Метод вибору видів диверсності

Запропонований метод вибору видів диверсності є ключовим етапом переходу від побудованого класифікатора та його графової моделі до практичного застосування при проєктуванні інформаційно-керуючих систем критичного призначення [43–45]. Його мета полягає у визначенні оптимальних альтернатив чи їх комбінацій, що одночасно задовольняють вимогам до рівня диверсності (безпечності) та обмеженням за вартістю реалізації.

Метод базується на принципах багатокритеріального вибору. З одного боку, необхідно забезпечити мінімально допустимий рівень диверсності для уникнення відмов за загальною причиною, з іншого — обмежити зростання вартості системи, яка є критичним чинником у промисловій експлуатації. Таким чином, завдання формулюється як оптимізація співвідношення «диверсність–вартість» у просторі можливих шляхів, побудованих на основі класифікатора.

2.4.1 Визначення метрик диверсності і відносної вартості

У процесі проєктування багатоверсійних інформаційно-керуючих систем критичного призначення виникає потреба у кількісному обґрунтуванні рішень, що стосуються вибору видів і підвидів диверсності. Для цього вводяться спеціальні метрики, які дозволяють формалізувати рівень диверсності та відносну вартість імплементації того чи іншого рішення. Метою даного підрозділу є детальний опис принципів визначення цих метрик, а також їх застосування у подальшому алгоритмі вибору оптимальних шляхів у графі класифікатора [46–48].

Метод вибору видів диверсності спирається на ряд ключових принципів, які визначають коректність побудови моделі:

- постановка задачі вибору передбачає врахування забезпечення необхідного рівня диверсності при мінімально можливій вартості;
- принципи визначення значень метрик диверсності передбачають врахування рівнів класифікатора (види та підвиди диверсності), а також використання принципу зворотних чисел для оцінки ступеня незалежності;
- метрики вартості визначаються як відносні витрати на імплементацію кожного виду або підвиду диверсності, із урахуванням їхньої пріоритетності;
- експертне оцінювання використовується для встановлення вихідних значень метрик і вагових коефіцієнтів у випадках, коли неможливо застосувати пряме формалізоване вимірювання.

На основі розширеного класифікатора будується граф G = (V, E), вершини якого відображають види та підвиди диверсності. Даний граф може змінюватися залежно від появи нових видів диверсності та їх деталізації.

Задача вибору формулюється через два можливих критерії:

1. забезпечення необхідного рівня диверсності при мінімізації вартості:

$$D \geq D_{req},\ C \rightarrow \min; \tag{2.2}$$

2. забезпечення обмеження вартості при максимізації диверсності:

$C \le C_{lim},\ D \to max.$ (2.3)

Процедура реалізації методу включає кілька етапів:

1. Аналіз класифікатора та побудова графа G = (V,E).
2. Формулювання вимог до диверсності (D_{req}) та вартості (C_{lim}).
3. Визначення та обчислення метрик диверсності (D_i) і вартості (C_i) для кожного виду чи підвиду диверсності (для кожної вершини графа).
4. Пошук множини шляхів у графі:

$L = \{L_j\},\ j = 1, \ldots, Card(L)$, із застосуванням відомих алгоритмів зчислення шляхів.

5. Обчислення інтегральних показників для кожного шляху $L_j \in L$:

$D(L_j),\ C(L_j)$.

6. Вибір оптимального шляху $L^* \in L$, що задовольняє вимогам. Якщо такого шляху не існує, виконується перехід до етапу комбінування шляхів.
7. Аналіз реалізованості комбінацій видів диверсності за обраним шляхом для системи, що розробляється, та коригування результатів у разі потреби. Якщо один шлях не дозволяє досягти необхідного рівня диверсності при прийнятній вартості, застосовуються стратегії комбінування:

- Стратегія StrDT (знизу–догори): починається з підвидів диверсності. Ця стратегія забезпечує впровадження кроків із мінімальним збільшенням як диверсності, так і вартості.
- Стратегія StrTD (згори–донизу): починається з видів диверсності верхнього рівня. Такий підхід забезпечує максимальний приріст диверсності, але й супроводжується швидким зростанням вартості.

Подальша деталізація методу враховує множину комбінацій шляхів:

$ML = \{MLq\},\ q = 1, \ldots, Card(ML)$.

При цьому сумісність використання двох шляхів $L_j \in L$, $L_k \in L$ визначається за такими критеріями:

- критичні обмеження: апаратні або програмні несумісності, перевантаження алгоритмів чи технологічні бар’єри, що роблять комбінацію неможливою;

– некритичні обмеження: ресурсні, організаційні чи економічні фактори, які можуть бути подолані за умови додаткових заходів.

На підставі цього формується множина допустимих комбінацій:

$ML' = \{ML'p\}, \; p = 1, \ldots, Card(ML')$.

Для кожної комбінації шляхів $MLp \in ML'$ визначаються метрики диверсності D(MLp) та вартості C(MLp).

Далі виконується пошук комбінації шляхів $ML^* \in ML'$, що задовольняє критеріям вибору. Якщо такої комбінації немає, здійснюється перехід до комбінування трьох і більше шляхів.

Таким чином, метод визначення метрик диверсності та вартості охоплює не лише оцінку окремих видів і підвидів, але й формування складних комбінацій альтернатив. Це дозволяє забезпечити гнучкість і масштабованість при виборі оптимальних стратегій, враховуючи як технічні, так і організаційні обмеження.

2.4.2 Експертне оцінювання значень метрик для розширеного класифікатора диверсності

Оцінка видів та підвидів диверсності у системах критичного призначення стикається з фундаментальною проблемою: аналітичні та експериментальні методи не дають можливості прямо виміряти значення метрик диверсності та відносної вартості [37; 49–52]. Це пояснюється такими причинами:

– складність реальних ІКС, де вплив факторів проявляється на різних рівнях (архітектурному, технологічному, організаційному);
– унікальність конкретних проєктних рішень (FPGA/SoC-платформи, власні методи синхронізації, архітектурні комбінації), що унеможливлює використання універсальних тестових методик;
– висока вартість і тривалість експериментальної перевірки на стендах, яка не виправдана на етапі проєктування;

– відсутність доступних статистичних даних щодо відмов за загальною причиною (ВЗЗП), які можна було б використати для емпіричної оцінки.

У такій ситуації найбільш адекватним і практично здійсненним стає застосування експертного підходу. Він дозволяє трансформувати досвід та знання спеціалістів у кількісні оцінки, які потім можна формалізувати у вигляді метрик. Таким чином, саме експертна оцінка забезпечує місток між якісним аналізом різнорідності та формалізованою математичною моделлю, необхідною для подальшої оптимізації.

Експертне оцінювання є виправданим у випадках, коли:

1. Прямі вимірювання неможливі або обмежені. Для більшості факторів диверсності не існує стандартних експериментальних методик.

2. Наявна сукупність фахових знань. Спеціалісти, які тривалий час займаються проєктуванням та оцінкою ІКС, здатні інтегрувати власний досвід та галузеві практики у вигляді числових оцінок.

3. Потрібна оперативність. Формування експертних анкет і обробка результатів є значно швидшими та дешевшими у порівнянні з натурними експериментами чи широкими статистичними вибірками.

4. Важлива узгодженість та порівнянність даних. Експертний метод дозволяє організувати єдину процедуру, що охоплює всі гілки класифікатора, з однаковою шкалою і правилами нормування. Для дослідження було сформовано групу з 10 експертів, до якої увійшли представники двох організацій:

– ХАІ (Харківський авіаційний інститут, кафедра 503) — провідні фахівці з проєктування та верифікації цифрових систем, зокрема на базі FPGA/SoC, із досвідом наукових досліджень у сфері інформаційно-керуючих систем АЕС;

– НВП «Радій–Радікс» — розробники і експерти з впровадження диверсних систем керування та захисту реакторів, із багаторічним практичним досвідом промислової експлуатації.

Ключові аргументи вибору цих експертів:

- Висока професійна компетентність у питаннях архітектурних рішень, технологій FPGA/SoC, апаратно-програмних засобів та їх сертифікації для критичних систем;
- Досвід участі у реальних проєктах ІКС АЕС, що забезпечує практичне розуміння проблематики ВЗЗП та вартості впровадження різних підходів;
- Збалансоване поєднання наукової та інженерної складових, що дозволяє уникати однобічності оцінок;
- Незалежність суджень, оскільки експерти належать до різних організацій, але працюють у єдиній проблемній сфері.

Компетентність експертів у даному проєкті прийнято однаковою, що спрощує обчислення і не потребує введення додаткових коефіцієнтів ваги для окремих оцінок. Це рішення обґрунтоване тим, що всі залучені спеціалісти мають багаторічний досвід у розробці, експертизі та сертифікації ІКС атомних електростанцій.

Для практичного отримання значень метрик було організовано експертне анкетування, агреговані дані якого містить Таблиця 2.1.

Анкета була сформована на основі розширеного класифікатора диверсності (NUREG-7007) з урахуванням його деталізації до рівня конкретних підвидів. Основна ідея — у кожному блоці запитань згрупувати 2–7 альтернатив, що належать до одного рівня чи гілки класифікатора. Це дозволяє експертам проводити порівняння саме між близькими за змістом варіантами, зменшуючи ризик суб’єктивних перекосів.

Для кожної гілки класифікатора (наприклад, «технології», «виробники обладнання», «архітектури логіки», «життєвий цикл», «сигнали» тощо) були сформовані окремі блоки запитань. У кожному блоці експерт повинен був присвоїти ранги (1…n) двом критеріям:

- диверсність (D) — де 1 означає найвищу очікувану різнорідність;
- вартість (C) — де 1 означає найбільші витрати на реалізацію.

Таблиця 2.1 - Експертне оцінювання розширеного класифікатора NUREG-7007 [36; 37]

Тест-опитування на тему розширеної класифікації диверсності Nureg-7007																							
*розставте в кожному пункті опитування пріорітети диверсності (1-X, де 1 - сама висока диверсність) для різних факторів диверсності згідно Nureg-7007, яку було розширено, та їх вартість (1-X, де 1 - сама висока вартість) для досягнення поточної диверсності. Кількість пріоритетів кожного питання дорівнює кількості пунктів цього питання.		Експ. 1		Експ. 2		Експ. 3		Експ. 4		Експ. 5		Експ. 6		Експ. 7		Експ. 8		Експ. 9		Експ. 10			
№	Запитання	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.	Див.	Вар.		
a.	Самий пріорітетний та найдорожчий пріорітет	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
b.	Звичайний але дешевий пріорітет	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3		
c.	Низький пріорітет з середньою вартістю	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2		
1.	Диверсність фундаментальних критеріїв (7 пунктів):																						
a.	Процес розроблення проєкту (різні технології, архітектури і т.п.)	1	1	4	2	1	1	1	1	1	1	2	3	1	5	6	4	3	2	1	1		
b.	Диверсні виробники обладнання проєкту	2	3	6	4	3	2	1	1	6	7	1	1	6	7	7	7	2	4	2	2		
c.	Логічні обробки обладнання (передача та обробка даних архітектури)	5	6	5	6	2	2	1	1	4	6	3	4	3	4	4	2	6	7	2	2		
d.	Різні функції для досгнення безпеки	4	4	1	3	1	3	1	1	2	4	5	5	7	3	3	4	4	5	2	2		
e.	Життєвий цикл (різні виробники, програмісти, маркетологи і т.п. при розробці проєкту)	3	2	2	1	2	1	1	1	7	2	4	2	5	6	5	6	1	1	1	1		
f.	Сигнали (вимірювальне обладнання)	7	5	7	5	2	2	1	1	3	5	7	6	4	2	1	1	7	3	2	3		
g.	Логіка та алгоритми	6	7	3	7	1	3	1	1	5	3	6	7	2	1	2	1	5	6	1	2		
2.	Диверсність в процесі розроблення (3 пункти):																						
a.	Використання різних технолоій (наприклад, мікроконтроллер та FPGA)	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	1	1	1	1		
b.	Використання різних підходів у рамках однієї технології (наприклад, FPGA та CPLD)	3	3	3	3	1	2	2	3	3	2	2	2	2	2	1	1	3	3	2	2		
c.	Використання різних архітектур (мікроконтроллер Atmel та ARM)	2	2	2	2	1	1	2	2	2	3	2	2	1	1	2	2	2	2	2	2		
3.	Диверсність виробників обладнання проєкту (4 пункти):																						
a.	Різний виробник принципово різного виконання обладнання проєкту	1	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	1	3	2	2		
b.	Один виробник принципово різного виконання обладнання проєкту	2	2	2	2	2	2	1	1	3	3	3	3	3	3	2	2	3	2	3	3		
c.	Різні виробники однакового обладнання проєкту	3	3	4	3	1	2	3	1	2	2	2	2	2	2	3	3	2	1	2	2		
d.	Один виробник різних версій однакового виконання обладнання проєкту	4	4	3	4	3	3	3	3	4	4	4	4	1	1	1	1	4	4	2	2		
4.	Диверсність логічної обробки обладнання (4 пункти):																						
a.	Різні логічні обробки архітектури (обробка даних за допомогою vhdl-функції або Nios-ядром)	1	1	1	1	1	2	1	2	1	1	1	1	4	4	4	3	1	2	1	2		
b.	Різні логічні версії обробки однакової архітектури (наприклад, різна розрядність даних мегафункції)	4	4	3	4	2	2	3	3	4	3	3	3	3	3	1	1	2	1	3	3		
c.	Різні інтеграції компонентів архітектури	2	2	2	2-	3	3	2	2	2	2	2	3	2	2	2	2	4	3	2	2		
d.	Різні потоки даних архітектури (послідовний та паралельний інтерфейс передачі даних)	3	3	4	3	2	2	2	2	3	4	4	4	1	1	3	4	3	4	2	2		
5.	Диверсність функцій для досгнення безпеки (3 пункти):																						
a.	Різні базові механізми для досягнення безпеки	1	1	1	1	1	3	1	1	1	1	1	1	3	3	2	3	2	1	2	2		
b.	Різні цілі, функції, керуюча логіка або засоби спрацювання однакових базових механізмів	2	2	2	2	2	3	1	1	2	2	2	3	2	2	3	2	1	2	1	1		
c.	Різний час відгуку	3	3	3	3	3	3	2	2	3	3	3	3	1	1	1	1	3	3	3	3		
6.	Диверсність життєвого циклу розроблення проєкту (4 пункти):																						
a.	Різні компанії розроблення проєкту	1	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	3	1	1	1		
b.	Різні менеджери в межах однієї компанії	4	4	4	4	2	3	3	1	4	4	4	4	3	3	1	1	4	3	2	2		
c.	Різні інженери, схемотехніки та програмісти	2	3	2	2	2	3	1	1	2	2	2	2	2	2	3	2	1	2	2	2		
d.	Різні тестувальники, верифікатори та виконавці	3	2	3	3	2	3	1	1	3	3	3	2	1	1	2	3	2	4	2	2		
7.	Диверсність вимірювальних пристроїв (3 пункти):																						
a.	Різні реакторні або технологічні параметри, чутливі до різних фізичних впливів	1	1	1	1	1	1	1	1	1	2	1	3	3	3	3	3	1	1	2	2		
b.	Різні реакторні або технологічні параметри, чутливі до однакових фізичних впливів	2	2	2	2	2	2	1	1	2	3	3	2	2	3	2	2	2	2	2	2		
c.	Однаковий технолоічний параметр, чутливий до різних надлишкових однакових датчиків	3	3	3	3	3	3	2	2	3	1	2	1	1	2	1	1	3	3	2	2		
8.	Диверсна логіка та алгоритми (3 пункти):																						
a.	Різні алгоритми, логіка та програмна архітектура	1	1	1	1	1	3	1	2	1	1	1	1	3	3	3	3	1	2	1	1		
b.	Різний час або порядок виконання	3	3	3	3	3	3	2	2	3	3	3	3	2	2	1	1	3	1	3	3		
c.	Різні функціональні представлення	2	2	2	2	2	3	1	2	2	2	2	2	1	1	2	2	2	3	2	2		
9.	Диверсність у використанні різних технологій при розробці проєкту (2 пункти):																						
a.	Різне обладнання	1	1	1	1	3	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1		
b.	Різні матеріали	2	2	2	2	2	2	3	1	2	2	2	2	1	1	1	1	2	2	1	1		
10.	Диверсність у використанні різних підходів у рамках однієї технології (4 пункти):																						
a.	Різні заводи виробники	4	4	1	1	2	1	3	1	3	1	1	1	4	4	4	4	4	1	2	1		
b.	Різні інженерні процеси	3	3	3	3	2	1	2	1	1	2	3	3	3	3	3	3	3	3	2	1		
c.	Різні матеріали	2	2	2	2	2	2	3	1	2	3	2	3	2	2	1	2	2	2	1	1		
d.	Різні ядра в рамках однієї технології	1	1	4	4	1	2	2	2	4	4	4	4	1	1	2	1	1	4	2	2		

Таким чином, кожен експерт одночасно формував два незалежних рангових ряди для кожного блоку.

До анкетування залучено 10 експертів. Кожен експерт заповнював анкету індивідуально, після попереднього узгодження термінології. Це дозволило уникнути різночитань і забезпечити однакове розуміння завдань.

Таблиця 2.2 - Уривок анкети експертного опитування [36; 37]

№	Запитання	Пріорітет		Коментарі
		Диверсн.	Вартість	
0.	**Приклад:**			
a.	Найменший у групі ризик ВЗП, найбільша вартість	1	1	
b.	Середній ризик ВЗП, найменша вартітсть	2	3	
c.	Найбільший ризик ВЗП, середня вартість	3	2	
1.	**Диверсність фундаментальних критеріїв (7 пунктів):**			
a.	Процес розроблення проєкту (різні технології, архітектури і т.п.)			
b.	Диверсні виробники обладнання проєкту			
c.	Логічні обробки обладнання (передача та обробка даних архітектури)			
d.	Різні функції для досгнення безпеки			
e.	Життєвий цикл (різні виробники, програмісти, маркетологи і т.п. при розробці проєкту)			
f.	Сигнали (вимірювальне обладнання)			
g.	Логіка та алгоритми			

Таблиця 2.2 містить:

- ліворуч — перелік запитань (блоки й альтернативи);
- праворуч — для кожного експерта подано два стовпчики: «Пріоритет Диверсн.» і «Пріоритет Вартість», що відповідають присвоєним рангам;
- у підсумку, після агрегації, кожен рядок анкети містить рангові оцінки від усіх 10 експертів.

На основі зібраних даних обчислюються часткові метрики диверсності та вартості для кожного розглянутого компонента.

Етапи проведення розрахунків:

1. обчислення для кожного питання окремо середньої арифметичної між всіма експертами;

2. ранжування середніх оцінок в рамках одного блоку з встановленням пріоритетів, що повертає те ж саме ранжування, але вже з урахуванням оцінок всіх експертів;

3. нормалізована сума обернених чисел компонентів поточного рівня дає змогу отримати метрики поточного блоку з сумою = 1;

4. даний алгоритм обраховується однаково для отримання метрик диверсності та вартості окремо:

$$\overline{RD} = \frac{1}{n} \sum_{i=1}^{n} ED_i, \tag{2.4}$$

$$D_j = \frac{1}{\left(\sum_{i=1}^{n} \frac{1}{RD_i}\right)\frac{1}{RD_j}}, \sum_{i=1}^{n} D_i = 1, \quad (2.5)$$

$$\overline{RC} = \frac{1}{n} \sum_{i=1}^{n} EC_i, \quad (2.6)$$

$$C_j = \frac{1}{\left(\sum_{i=1}^{n} \frac{1}{RC_i}\right)\frac{1}{RC_j}}, \sum_{i=1}^{n} C_i = 1, \quad (2.7)$$

, де:

n – кількість підрівнів метрик поточного рівня;

j – індекс поточної метрики;

ED – визначені експертами пріоритети диверсності, де 1 – сама висока диверсність;

rD – середня арифметична пріоритетів диверсності поточного рівня;

RD – індекс пріоритету (ранг) метрики диверсності поточного рівня;

D – обчислена метрика диверсності в межах одної групи (рівня) класифікатора, де max(RD) – компонент з найвищою диверсністю.

EC – визначені експертами пріоритети вартості, де 1 – сама висока вартість;

rC – середня арифметична пріоритетів вартості поточного рівня;

RC – індекс пріоритету (ранг) метрик вартості поточного рівня;

C – обчислена метрика вартості в межах одної групи класифікатора, де max(RC) – компонент з найвищою вартістю.

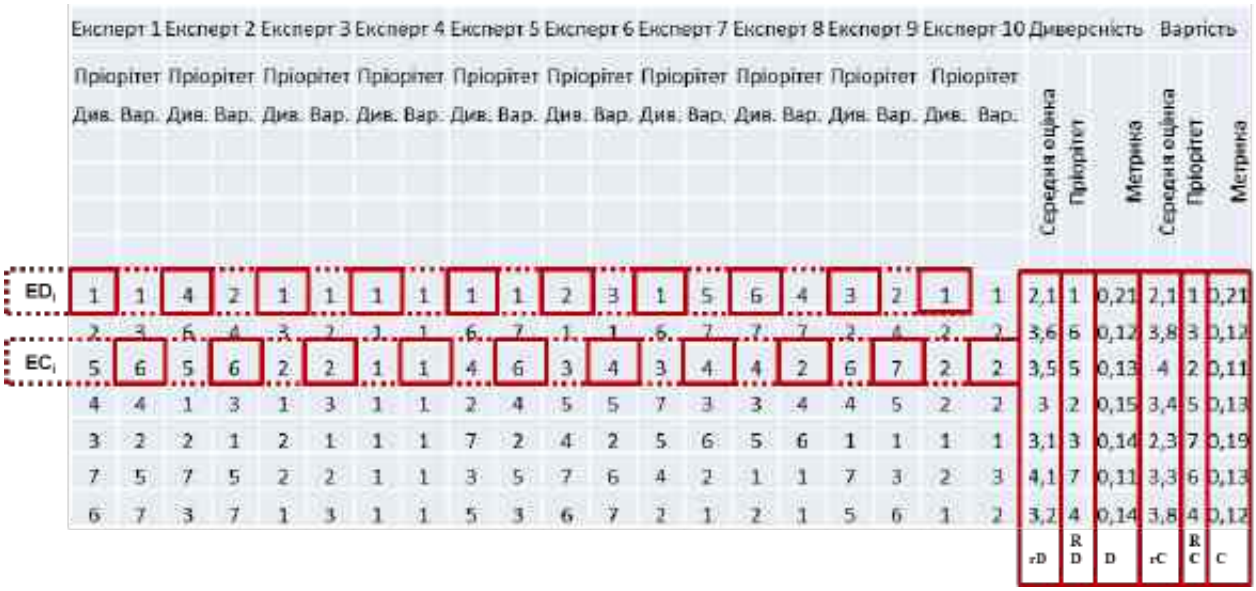

Рисунок 2.5 - Схематичне зображення процесу калькуляцій результатів опитування

Схематичне подання процесу обчислення інтегральних показників диверсності та вартості ілюструє Рисунок 2.5, який демонструє логіку перетворення початкових даних експертного оцінювання у нормовані значення показників. Формула зручна для розуміння процесу перерахунку формулами 2.1 – 2.4).

Подальша обробка здійснюється у кілька етапів:

- **Усереднення рангів** — обчислюється середнє значення для кожної альтернативи (окремо для D та C).
- **Перетворення рангів у ваги** за принципом зворотних чисел (із поправкою на максимальний ранг у блоці). Це забезпечує правило: менший ранг = більша вага.
- **Нормування ваг** до діапазону [0;1] всередині кожного блоку, що дозволяє безпосередньо інтерпретувати їх як часткові метрики D_i та C_i.
- **Формування інтегральних метрик** шляхом агрегації часткових метрик по рівнях класифікатора.

Таким чином, Таблиця 2.1 є ключовим проміжним етапом: вона містить повний набір вихідних оцінок експертів, які потім трансформуються у числові метрики для розрахунків диверсності та вартості в графовій моделі.

Організоване експертне анкетування дозволило перетворити якісні судження фахівців у структуровані числові дані, які уніфіковано описують диверсність і відносну вартість кожного виду та підвиду класифікатора. Отримані результати, представлені у таблиці, формують основу для побудови часткових і інтегральних метрик, що застосовуються у наступному підрозділі для оцінки диверсності з використанням метрик.

2.4.3.Оцінка диверсності з використанням метрик

В попередньому розділі було проведено експертне анкетування, результати якого представлені у вигляді таблиці з пріоритетами диверсності та вартості для всіх видів і підвидів класифікатора. Ці дані відображають суб’єктивні, але узгоджені судження експертів, перетворені у числову форму через ранги. Проте у такому вигляді вони ще не придатні для безпосереднього використання в аналітичній моделі — потрібне їх перетворення у метрики, що мають однакову шкалу та властивості для математичної обробки [43; 44; 53–58].

Саме у цьому підрозділі здійснюється обчислення часткових і інтегральних метрик диверсності (D_i) та відносної вартості (C_i) з анкетних даних, а також їх агрегація у вигляді зведених таблиць [46; 47]. Це дозволяє створити повноцінний набір числових характеристик, необхідних для подальшого використання у графовій моделі та алгоритмах вибору диверсності.

Перетворення анкетних даних у метрики потребує використання спеціальних формул. Їх застосування обґрунтовується такими вимогами:

1. Усунення суб’єктивних відхилень. Оскільки експерти працювали незалежно, окремі їх оцінки могли різнитися. Усереднення рангів та подальше перетворення у ваги забезпечує стабільність результатів.

2. Принцип «менший ранг = більший пріоритет». Для цього використовується метод зворотних чисел, який формалізує ієрархію альтернатив у вигляді вагових коефіцієнтів.

3. Порівнянність різних блоків. Блоки анкети мали від 2 до 7 елементів, і без нормування пряме порівняння було б некоректним. Нормування ваг у діапазоні [0;1] робить усі результати уніфікованими.

4. Формування інтегральних характеристик. Часткові метрики кожного рівня об'єднуються у шляхові (інтегральні) показники для повного маршруту у графі класифікатора. Це дозволяє застосовувати критерії відбору (наприклад, $D \geq REQ_d$, $C \rightarrow \min$ чи $C \leq REQ_c$, $D \rightarrow \max$.

У результаті застосування формул ми отримаємо:

- часткові метрики диверсності та вартості для кожного елемента класифікатора;
- зведену таблицю з ранжуванням і метриками, що демонструє вплив кожного виду та підвиду диверсності;
- агреговані інтегральні метрики шляхів, які будуть використані для реалізації алгоритмів вибору оптимальних рішень.

Таким чином, цей підрозділ є перехідним етапом: саме тут експертні оцінки перетворюються у формалізовані метрики, що робить можливим їх подальший аналіз у рамках матрично-графових моделей.

Детальний опис формул і порядку застосування:

1. Розрахунок диверсності компонента за результатами експертної оцінки:

$$\overline{RD} = \frac{1}{n} \sum_{i=1}^{n} ED_i \,, \tag{2.8}$$

$$D = \frac{1}{\left(\sum_{i=1}^{n} \frac{1}{\overline{RD_i}}\right)(1 + Max(\overline{RD}) - RD_i)}, \tag{2.9}$$

$$\sum_{i=1}^{n} D_i = 1 \,, \tag{2.10}$$

– На першому кроці обчислюється середній ранг диверсності для кожної альтернативи всередині поточного блоку (2…7 елементів): $\overline{RD_i}$ — це середнє по всіх експертах значення їхніх ранжувань ED, де «1» означає найвищу очікувану диверсність. Далі формула перетворює $\overline{RD_i}$ у нормовану метрику диверсності $D_i \in [0;1]$. Це перетворення одночасно реалізує два принципи:
– Принцип зворотних чисел — нижчий середній ранг (кращий пріоритет) повинен давати більший внесок у метрику.
– Поправка на глибину рангу — множник $(1 + Max(\overline{RD}) - \overline{RD_i})$ підсилює відмінності між лідерами блоку та середніми показниками навіть коли різниця у середніх рангах невелика.
– Завершальний множник з сумою $\left(\sum_{i=1}^{n} \frac{1}{\overline{RD_i}}\right)$ забезпечує нормування в межах блоку: у підсумку виконується (2.10), тобто всі Di в поточному блоці утворюють ймовірнісний розподіл (зручно для подальшої агрегації).
– Інтерпретація: менше $\overline{RD_i}$ (кращий пріоритет за диверсністю) → більший D_i. Формула нечутлива до абсолютних значень рангів різних експертів (береться середнє) і коректно працює при будь-якому розмірі блоку $n \in [2;\infty)$.

2. Розрахунок вартості компонента за результатами експертної оцінки:

$$\overline{RC} = \frac{1}{n} \sum_{i=1}^{n} EC_i, \tag{2.11}$$

$$C = \frac{1}{\left(\sum_{i=1}^{n} \frac{1}{\overline{RC_i}}\right)(1 + Max(\overline{RC}) - RC_i)}, \tag{2.12}$$

$$\sum_{i=1}^{n} C_i = 1, \tag{2.13}$$

Спершу обчислюється середній ранг вартості для кожної альтернативи у блоці: $\overline{RC_i}$ — середнє за експертами їхніх рангів ЕС (де «1» означає найвищу вартість).

Далі формується нормована метрика відносної вартості $C_i \in [0;1]$ за аналогічними принципами:

1. Зворотна залежність — чим менший середній ранг вартості $\overline{RC_i}$ (тобто дорожче), тим більший внесок у C_i.
2. Поправка на глибину рангу через $(1 + Max(\overline{RC}) - \overline{RC_i})$— для більш чіткої диференціації дорогих рішень.
3. Нормувальний множник гарантує (2.13) у межах кожного блоку.

Інтерпретація: менше $\overline{RC_i}$ (вища очікувана вартість) → більший C_i. Як і для D_i , результати порівнювані між блоками будь-якої потужності.

Легенда до обох формул (спільна):

- n — кількість альтернатив (елементів) у поточному блоці класифікатора (зазвичай 2…7).
- ED — індивідуальні пріоритети/ранги за диверсністю від кожного експерта, де «1» — найвища диверсність.
- $\overline{RD_i}$ — середній ранг диверсності альтернативи і в блоці (середнє значення ED за всіма експертами).
- D_i — нормована метрика диверсності альтернативи і в межах блоку (2.10);
- ЕС — індивідуальні пріоритети/ранги за вартістю від кожного експерта, де «1» — найвища вартість.
- $\overline{RC_i}$ — середній ранг вартості альтернативи і в блоці (середнє значення ЕС за всіма експертами).
- C_i — нормована метрика відносної вартості альтернативи і в межах блоку (2.13);
- $Max(\overline{RD}), Max(\overline{RC})$ — найбільші середні ранги у відповідному блоці (задають «глибину шкали» для поправки на ранг).

Покроковий порядок застосування:

1. Збір даних: беремо з анкети усі індивідуальні ранги ED та EC для поточного блоку.

2. Усереднення: обчислюємо $\overline{RD_i}$ та $\overline{RC_i}$ для кожної альтернативи i.

3. Поправка на глибину рангу та зворотна залежність: застосовуємо у вигляді множників, як записано у формулах (2.8) та (2.11).

4. Нормування в межах блоку: використовуємо наявні у формулах нормувальні множники — отримуємо D_i та C_i , причому повинні виконуватись умови (2.10) та (2.13).

5. Перевірка узгодженості:

- якщо деякі $\overline{RD_i}$ або $\overline{RC_i}$ збігаються (нічия за середнім рангом), відповідні D_i та C_i також будуть однаковими — це коректно;
- для неповних відповідей експерта (якщо трапляються) усереднення проводиться за доступними значеннями;
- розмір блоку n не впливає на порівнюваність між блоками, бо результати вже нормовані до [0;1] із сумою 1.

Властивості та практичні наслідки:

- Монотонність. Менший середній ранг → більше значення метрики. Це гарантує узгодженість із семантикою анкет.
- Інваріантність до масштабу. Працюємо з рангами (а не «сирими» суб'єктивними числами), тож метод стійкий до індивідуальних «розтягувань» шкали експертів.
- Порівнянність блоків. Окремі блоки можуть містити різну кількість альтернатив; нормування до суми 1 робить результати взаємно порівнюваними.
- Прозорість для аудиту. Будь-який рядок анкети можна відтворити у вигляді $\overline{RD_i}$ / $\overline{RC_i}$ і отримати відповідні D_i/C_i — це важливо для валідації і сертифікаційного огляду.
- Готовність до агрегації. Значення D_i та C_i безпосередньо використовуються на наступному етапі.

Застосування анкетних даних і формул цього розділу дало змогу перетворити якісні експертні оцінки у кількісні метрики диверсності та відносної вартості, нормовані в інтервалі [0;1]. Цей підхід забезпечує порівнянність між блоками класифікатора з різною кількістю альтернатив, зберігає семантику «менший ранг = більший пріоритет», а також усуває вплив суб'єктивних коливань окремих експертів. У результаті формується набір часткових метрик D_i та C_i, які далі агрегуються в інтегральні показники для шляхів і комбінацій шляхів класифікатора. Таким чином, підрозділ завершує перехід від сирих результатів анкетування до строго нормованих числових характеристик, що створюють основу для алгоритмів вибору диверсності.

2.4.4 Стратегії та алгоритми вибору диверсності

Після отримання нормованих метрик диверсності та вартості постає задача практичного вибору оптимального набору альтернатив для проєктованої системи. Складність полягає у тому, що жодна з альтернатив не є ідеальною: більш висока диверсність зазвичай досягається за рахунок підвищеної вартості, а зменшення вартості може супроводжуватися втратою у різнорідності. Для вирішення цього протиріччя застосовуються спеціальні стратегії та алгоритми, які дозволяють знайти збалансований варіант або сукупність варіантів.

У наведеній ілюстрації (Рисунок 2.6) показано один із ключових результатів на множині альтернатив. Джерелом даних для цього прикладу є фрагмент розширеного класифікатора диверсності та вартості, а саме група «LIFE-CYCLE» та «SIGNAL». Таким чином, на графіку відображені узагальнені метрики C_i (вартість) та D_i (диверсність), отримані для семи альтернативних підходів, які було оцінено експертами.

	Rank	Cst wt	Dt wt	Attribute criteria	Rank	Cst wt	Dt wt
SIGNAL	6	0,11	0,13	Different reactor or process parameters sensed by different physical effect	1	0,42	0,37
				Different reactor or process parameters sensed by the same physical effect	2	0,31	0,32
				The same process parameter sensed by a different redundant set of similar sensors	3	0,27	0,32
LOGIC	4	0,14	0,12	Different algorithms, logic, and program architecture	1	0,37	0,32
				Different timing or order of execution	3	0,28	0,34

				Different runtime environments	2	0,19	0,22
				Different functional representations	4	0,29	0,28

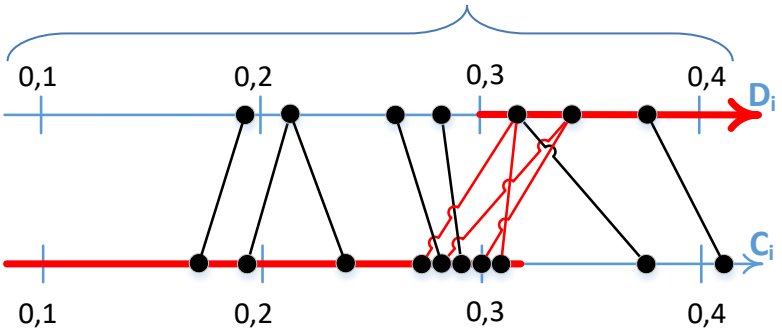

Рисунок 2.6 - Графічна ілюстрація пошуку оптимального виду диверсності та вартості

На горизонтальній осі відкладені значення відносної вартості C_i, а на вертикальній осі — значення диверсності D_i. Кожне коло на діаграмі відповідає одній з альтернатив, відібраних із таблиці. Таким чином, графік дає змогу візуально оцінити компроміси між різними варіантами.

Згідно з ілюстрацією, множина точок (альтернатив) розташовується у просторі (C_i x D_i). Червоні лінії показують процес відсікання варіантів, що не задовольняють обмеження. На виході формується підмножина допустимих рішень, із якої можна відібрати оптимальний або декілька квазіоптимальних шляхів.

У дослідженні розглядаються дві основні стратегії:

1. Стратегія мінімізації вартості при виконанні вимог до диверсності:

$D_i \geq REQ_d, \quad C_i \rightarrow min$

Стратегія орієнтована на забезпечення потрібної функціональної різнорідності при мінімальних витратах. На практиці означає вибір найбільш економного варіанта серед тих, що здатні гарантувати необхідний рівень захисту від відмов за загальною причиною.

2. Стратегія максимізації диверсності при обмеженій вартості:

$C_i \leq REQ_c, \quad D_i \rightarrow max$

Використовується у випадках, коли існує жорстке обмеження бюджету, але допустимо прагнути максимальної надійності у межах наданих ресурсів.

Забезпечує вибір найсильнішого з доступних заходів, який вкладається у заданий фінансовий ліміт.

Процедуру можна подати у вигляді послідовності кроків:

1. **Формування множини варіантів.** Згідно з даними таблиці прикладу, отримано 7 альтернатив, кожна з яких має свої значення C_i та D_i.

2. **Побудова графіка у просторі (C_i x D_i).** Кожна альтернатива відображається як точка.

3. **Застосування критерію відбору.** Виконується перевірка умов $D_i \geq REQ_d$ та/або $C_i \leq REQ_c$.

4. **Формування пріоритетних рядів.** Визначаються всі допустимі альтернативи, далі вони впорядковуються за обраним головним критерієм (вартість або диверсність).

5. **Фіксація результату.** Якщо знайдено хоча б одну альтернативу, що відповідає критеріям, вона приймається як оптимальна. Якщо жодна не відповідає, можливий перехід до комбінування кількох шляхів.

У показаному прикладі оптимізація виконана за першою стратегією: мінімізація вартості при забезпеченні необхідної диверсності. На діаграмі видно, що кілька точок відсікаються через недостатній рівень D_i. З решти залишається невелика група рішень, з яких найнижче за C_i фіксується як оптимальний вибір.

Розглянутий графік ілюструє базовий алгоритм прийняття рішень, який комбінує числові метрики вартості й диверсності та дозволяє на практиці застосувати критерії $D \geq REQ_d$, $C \rightarrow \min$ або $C \leq REQ_c$, $D \rightarrow \max$. Це забезпечує обґрунтований і прозорий вибір оптимального виду диверсності для складних систем, що функціонують у критично важливих умовах.

2.4.5 Обчислення диверсності і відносної вартості. Стратегії вибору за визначеними критеріями

У сучасних системах управління та захисту важливим завданням є обґрунтований вибір таких комбінацій технічних рішень, які одночасно забезпечують належний рівень різнорідності та прийнятну вартість їх реалізації [57]. Для цього недостатньо лише якісного опису переваг чи недоліків кожного з

варіантів — потрібні інструменти, що дозволяють виконувати кількісне порівняння та відбір.

Саме з цією метою застосовується методика обчислення показників диверсності та відносної вартості. Вона дає можливість перейти від окремих експертних оцінок та характеристик елементів до узагальненої картини, у якій враховуються всі рівні класифікатора та взаємозв’язки між ними. Результатом є кількісні параметри, що дозволяють порівнювати альтернативи між собою та визначати найдоцільніші варіанти.

Ключовим елементом цієї методики є побудова графової моделі, де відображаються всі можливі шляхи поєднання різних видів і підвидів диверсності. Така модель дозволяє оцінити сукупний ефект від вибору того чи іншого набору рішень і застосувати визначені критерії — наприклад, мінімізацію витрат за умови досягнення потрібного рівня безпеки або, навпаки, максимізацію різнорідності за обмеженого ресурсу.

У цьому підрозділі розглянуто, як саме виконується процес переходу від початкових даних до інтегрованих оцінок, яким чином формується таблиця результатів та які висновки можна зробити для вибору оптимальних стратегій розвитку системи.

Для переходу від якісних характеристик окремих елементів класифікатора до кількісних показників, придатних для порівняння і відбору, застосовуються узагальнені формули. Вони дозволяють обчислити інтегральні метрики для кожного шляху графа диверсності, що відображає поєднання різних видів і підвидів.

Процес переходу від експертних оцінок до нормованого показника диверсності для окремих гілок класифікатора:

- Вихідними даними є значення, які отримані від експертів у вигляді пріоритетів. Вони відображають ступінь відмінності між різними технологіями, архітектурами чи командами.

– Далі значення ранжуються і усереднюються, що дозволяє зняти випадкові відхилення окремих експертів і сформувати єдиний базовий показник.
– За допомогою принципу зворотних чисел показник перетворюється на вагу: чим вищий пріоритет (тобто чим сильніша різнорідність), тим більше значення отримує елемент.
– Після нормування у межах блоку всі значення приводяться до діапазону [0;1], що забезпечує порівнянність навіть між різними гілками.

Таким чином, результатом застосування цієї формули є числове значення, яке показує внесок конкретного елемента або шляху в загальну різнорідність системи.

Для інтегральної оцінки диверсності та відносної вартості на рівні повного шляху L_j у графі використовується модифікована адитивна модель із ефектом насичення. Вона дозволяє врахувати багаторівневу ієрархію класифікатора та різну кількість альтернатив на кожному рівні. Це забезпечує коректне нормування й можливість багатокритеріального порівняння.

$$1.\ SD = \sum_{i=1}^{M} \frac{M - i + 1}{M}\left(D_i - \frac{1}{Q_i}\right), \tag{2.14}$$

Формула (2.14) визначає сумарну диверсність шляху SD:

– Кожен рівень ієрархії робить свій внесок через нормовану метрику D_i.
– Щоб уникнути впливу кількості альтернатив на рівні, від показника віднімається базове значення $1/Q_i$. Це корекція, яка дозволяє об’єктивно оцінювати навіть ті гілки, де число варіантів різне.
– Множник $\frac{M-i+1}{M}$ задає більшу вагу для верхніх рівнів, оскільки саме вони визначають глобальні відмінності між альтернативами (наприклад, різні архітектурні підходи чи виробники обладнання).

Таким чином, формула переводить набір локальних оцінок у єдиний числовий показник, який характеризує загальний рівень різнорідності обраного шляху. Це ключова величина для перевірки виконання вимог до функційної безпечності.

$$2.\ SC = \sum_{i=1}^{M} \frac{M-i+1}{M}\left(C_i - \frac{1}{Q_i}\right), \tag{2.15}$$

Формула (2.15) є аналогічною, але стосується інтегральної оцінки відносної вартості SC:

- За основу беруться нормовані вузлові метрики C_i, що описують витрати на реалізацію конкретної альтернативи.
- Корекція $-1/Q_i$ усуває упередження через різну потужність групи.
- Ваговий коефіцієнт $\frac{M-i+1}{M}$ враховує, що вибір на верхніх рівнях класифікатора (наприклад, технологія чи архітектурне рішення) зазвичай суттєвіше впливає на загальні витрати, ніж локальні відмінності на нижчих рівнях.

У результаті отримуємо інтегральний показник, який відображає відносну вартість всього шляху в контексті багатокритеріального вибору. Разом із показником SD цей коефіцієнт формує пару «диверсність–вартість», що й визначає позицію шляху на площині рішень.

Приклад обчислення для підгрупи з чотирьох альтернатив:

SD_i={0.24;0.14;0.06;0.07}, SC_i={0.21;0.13;0.04;0.06}

Після підстановки у формули (2.16-2.17) отримуємо:

- сумарна диверсність: D_{total}=0.43;
- сумарна відносна вартість: C_{total}=0.37.

Ці результати демонструють, що навіть при наявності невеликих часткових показників на окремих рівнях, завдяки ієрархічному зважуванню формується узагальнена характеристика всього шляху. Вона є ключовим орієнтиром у процедурі вибору.

Легенда до формул:

- M — кількість рівнів у гілці класифікатора;
- i — порядковий номер рівня (рахується зверху вниз);
- Q_i — кількість альтернатив на рівні i;
- D_i — часткова нормована метрика диверсності для альтернативи на рівні i;
- C_i — часткова нормована метрика відносної вартості для альтернативи на рівні i;
- SD — інтегральна (сумарна) метрика диверсності для всього шляху;
- SC — інтегральна (сумарна) метрика вартості для всього шляху.

Важливо підкреслити, що обидві формули застосовуються у парі. Кожен елемент отримує два значення — диверсності та вартості. У подальшому саме їх поєднання дає можливість сформувати збалансований набір альтернатив.

- Якщо елемент має високу диверсність, але водночас високу вартість, то його вибір доцільний лише у тих випадках, коли забезпечення різнорідності є критично важливим.
- Якщо ж значення вартості невелике, а диверсність помірна, то це може бути хорошим варіантом для побудови «бюджетного» шляху.
- Найціннішими зазвичай є ті вузли, які демонструють помірні витрати при достатньо високій диверсності, адже вони забезпечують найкраще співвідношення «ефективність–вартість».

Таблиця 2.3 є одним із центральних елементів методики аналізу видів диверсності. Вона ґрунтується на класифікаторі NUREG-7007, що використовується для систем безпеки атомних станцій [28; 52; 59–63]. У вихідному документі класифікатор мав якісну форму — перелік категорій і підкатегорій, які описували можливі види різнорідності. Проте для практичного застосування в інженерних проєктах такого підходу виявилося недостатньо.

Таблиця 2.3 - Розширений класифікатор диверсності з метриками диверсності та вартості [5; 29]

	Rank	Cst wt	Dt wt	Attribute criteria	Rank	Cst wt	Dt wt	Third level criteria	Rank	Cst wt	Dt wt	Fourth level criteria	Rank	Cst wt	Dt wt	Cost	Dst
DESIGN	1	0,21	0,21	Different technologies	1	0,41	0,43	Different equipment	1	0,55	0,55					0,29	0,30
								Different materials	2	0,45	0,45					0,25	0,25
				Different approaches within a technology	3	0,26	0,26	Different factory	4	0,21	0,28					0,14	0,17
								Different engineering process	3	0,23	0,23					0,15	0,14
								Different materials	1	0,30	0,27					0,19	0,16
								Different cores the same tech.	2	0,26	0,21					0,16	0,14
				Different architectures	2	0,32	0,31	Different factory	3	0,23	0,28					0,19	0,21
								Different engineering process	2	0,25	0,23					0,20	0,19
								Different materials	1	0,29	0,28					0,22	0,21
								Different cores different arch.	4	0,23	0,20					0,19	0,17
EQUIP.MANUF.	3	0,12	0,12	Different manufacturers of fundamentally different equipment designs	1	0,33	0,30	Different engineering process	1	0,29	0,27					0,21	0,16
								Different materials	2	0,28	0,23					0,20	0,14
								Different management (owner)	4	0,19	0,26					0,16	0,15
								Different equipment	3	0,24	0,25					0,18	0,15
				Same manufacturer of fundamentally different equipment designs	2	0,24	0,25	Different engineering process	1	0,29	0,27					0,13	0,12
								Different materials	2	0,28	0,23					0,13	0,10
								Different management (owner)	4	0,20	0,26					0,09	0,11
								Different equipment	3	0,23	0,24					0,10	0,11
				Different manufacturers of same equipment design	3	0,24	0,27	Different engineering process	1	0,29	0,24					0,13	0,13
								Different materials	2	0,25	0,26					0,11	0,13
								Different management (owner)	4	0,20	0,26					0,09	0,13
								Different equipment	3	0,25	0,24					0,11	0,13
				Same manufacturer of different versions of the same equipment design	4	0,20	0,19	Different engineering process	1	0,23	0,21					0,10	0,07
								Different materials	2	0,21	0,22					0,09	0,08
								Different management (owner)	5	0,16	0,21					0,06	0,07
								Different equipment	4	0,20	0,20					0,08	0,07
								Different IDE	3	0,21	0,17					0,09	0,06
LOGIC PROC.EQUIP.	2	0,13	0,11	Different logic processing architectures	1	0,35	0,31	Different chips	1	0,59	0,60					0,24	0,21
								Different architectures on the same ship	2	0,41	0,40					0,16	0,11
				Different logic processing versions in same architecture	4	0,20	0,22	Different data width	1	0,51	0,53					0,10	0,10
								Different core frequency	2	0,49	0,47					0,08	0,07
				Different component integration architectures	2	0,24	0,26	Different engineering process	2	0,47	0,47					0,11	0,10
								Different technologies	1	0,53	0,53					0,14	0,13
				Different data flow architectures	3	0,21	0,21	Different algorithm	1	0,34	0,26	Different IP-cores	1	0,55	0,56	0,15	0,10
												Different algorithm the same core	2	0,45	0,44	0,13	0,06
								Different processing time	4	0,18	0,19	Different synchronization period	1	0,56	0,53	0,07	0,05
												Different functions executing shift	2	0,44	0,47	0,04	0,04
								Different programming language	2	0,25	0,32	Mandatory programming languages	1	0,56	0,53	0,11	0,12
												Declaratory programming languages	2	0,44	0,47	0,08	0,10
								Different data flow interface	3	0,23	0,24	Width of parallel interface	1	0,56	0,53	0,10	0,08
												Speed of serial interface	2	0,44	0,47	0,07	0,06
FUNCTION	5	0,15	0,13	Different underlying mechanisms to accomplish safety function	1	0,41	0,40									0,21	0,18
				Different purpose, function, control logic, or actuation means of same underlying mechanism	2	0,34	0,34									0,15	0,13
				Different response time scale	3	0,25	0,27	Different architectures	2	0,35	0,36					0,09	0,10
								Different compilator or settings	3	0,28	0,35					0,06	0,09
								Different algorithm	1	0,37	0,29					0,10	0,06
LIFE-CYCLE	7	0,14	0,19	Different design companies	1	0,30	0,34	Different countries	4	0,21	0,27					0,16	0,26
								Different equipment	1	0,32	0,31					0,21	0,29
								Different supplier materials	2	0,25	0,21					0,18	0,23
								Different management (owner)	3	0,22	0,21					0,17	0,24
				Different management teams within the same company	4	0,17	0,19	Different experience	1	0,31	0,28					0,12	0,16
								Different equipment	3	0,23	0,30					0,08	0,16
								Different nationality	2	0,26	0,22					0,09	0,13
								Different temperament	4	0,20	0,20					0,06	0,12
				Different designers, engineers, and/or programmers	2	0,28	0,26	Different experience	1	0,32	0,29					0,20	0,21
								Different equipment	3	0,23	0,28					0,16	0,21
								Different nationality	2	0,25	0,24					0,17	0,19
								Different temperament	4	0,20	0,19					0,14	0,17
				Different implementation/validation teams	3	0,24	0,22	Different experience	1	0,30	0,31					0,16	0,20
								Different equipment	2	0,26	0,28					0,14	0,18
								Different nationality	3	0,26	0,23					0,14	0,16
								Different temperament	4	0,18	0,18					0,10	0,14
SIGNAL	6	0,11	0,13	Different reactor or process parameters sensed by different physical effect	1	0,42	0,37									0,17	0,16
				Different reactor or process parameters sensed by the same physical effect	2	0,31	0,32									0,09	0,12
				The same process parameter sensed by a different redundant set of similar sensors	3	0,27	0,32									0,06	0,12
LOGIC	4	0,14	0,12	Different algorithms, logic, and program architecture	1	0,37	0,32									0,23	0,17
				Different timing or order of execution	3	0,28	0,34									0,16	0,18
				Different runtime environments	2	0,19	0,22									0,09	0,09
				Different functional representations	4	0,29	0,28									0,17	0,14

У рамках цього дослідження класифікатор було розширено і формалізовано. До кожного виду та підвиду диверсності додані кількісні характеристики — метрики вартості і диверсності, які отримані на основі експертних оцінок та нормовані в діапазон [0;1]. Це дало змогу перетворити якісний перелік

альтернатив у кількісну модель, придатну для обчислень, побудови шляхів у графі та застосування багатокритеріальних критеріїв вибору.

Щоб коректно працювати з таблицею, необхідно розуміти значення кожного її стовпчика. Нижче наведено детальний опис колонок:

1. **Назва критерію.** Назва альтернативи стандарту Nureg-7007 на відповідному рівні класифікатора. Це текстовий опис, який дозволяє пов’язати числові метрики з конкретним видом чи підвидом диверсності.

2. **Rank 1-4 рівня.** Обчислений середній ранг, який експерти присвоїли альтернативі у межах першого рівня класифікатора. Менший ранг означає вищу пріоритетність.

3. **Cst wt 1-4 рівня.** Нормована метрика відносної вартості для елемента першого рівня. Отримана з експертних оцінок EC → RC → нормування.

4. **Dst wt 1-4 рівня.** Нормована метрика диверсності для елемента першого рівня. Обчислена за аналогічною процедурою ED → RD → нормування.

5. **Cost (SC).** Інтегральна відносна вартість усього шляху, який проходить через дану альтернативу. Обчислюється за формулою (2.15) на основі вузлових C_i.

6. **Dst (SD).** Інтегральна метрика диверсності усього шляху. Обчислюється за формулою (2.14) на основі вузлових D_i .

Таблиця 2.3 — це операційний міст від вузлових оцінок, отриманих з анкет експертів, до інтегральних рішень рівня шляху, сформованих згідно з графом диверсних компонентів розширеного класифікатора. Два ключові стовпчики Cost і Dst дозволяють порівняти альтернативи і зафіксувати оптимальний вибір або декілька квазіоптимальних варіантів для подальшого аналізу.

У цьому підрозділі було показано, як на основі анкетних даних і нормованих вузлових метрик формується кількісна база для прийняття рішень у задачі вибору видів диверсності. Послідовне застосування формул дозволяє перетворити експертні оцінки у часткові показники диверсності та вартості, а подальша агрегація уздовж шляхів графа забезпечує отримання інтегральних

характеристик, які відображають сукупний ефект від поєднання кількох альтернатив.

Таблиця результатів демонструє, як саме ці формули працюють на практиці: кожен рядок відповідає певному шляху в графі, для якого наведено як вузлові, так і інтегральні значення. Це дає змогу прозоро простежити походження підсумкових показників, перевірити їх на відповідність критеріям і оцінити баланс між досягнутим рівнем різнорідності та необхідними витратами.

Завдяки такому підходу методика забезпечує:

- кількісну обґрунтованість вибору — кожне рішення підтверджується чіткими числовими критеріями;
- можливість порівняння альтернатив із різних гілок класифікатора на єдиній шкалі;
- гнучкість у застосуванні стратегій (знизу–догори або згори–донизу) залежно від пріоритетів системи;
- універсальність для подальших алгоритмів вибору як одного шляху, так і комбінацій кількох шляхів.

Таким чином, обчислення диверсності та відносної вартості є центральним етапом методики: воно формує основу для реалізації критеріїв оптимізації та забезпечує практичну придатність розробленого підходу у задачах проєктування систем безпеки з багаторівневою диверсністю.

2.4.6 Алгоритм вибору за одним шляхом

Алгоритм вибору за одним шляхом — це базова, прозора процедура методу диверсності, яка працює з ієрархічним графом рішень і дозволяє знайти один-єдиний технологічно цілісний варіант побудови (шлях від вершини верхнього рівня до кінцевої вершини нижнього рівня) [59; 62; 64; 65]. Кожна вершина графа (Рисунок 2.7) відповідає певному виду/підвиду диверсності та має дві нормовані метрики: D_i (диверсність) і C_i (відносна вартість), отримані з експертних анкет і нормовані в [0;1]. Завдання підрозділу — описати, як за наявності нормованих

метрик по вузлах обрати один шлях L*, що задовольняє задані вимоги: досягнути не нижчого за потрібний рівня диверсності за мінімальної вартості або, навпаки, максимізувати диверсність за обмеженої вартості.

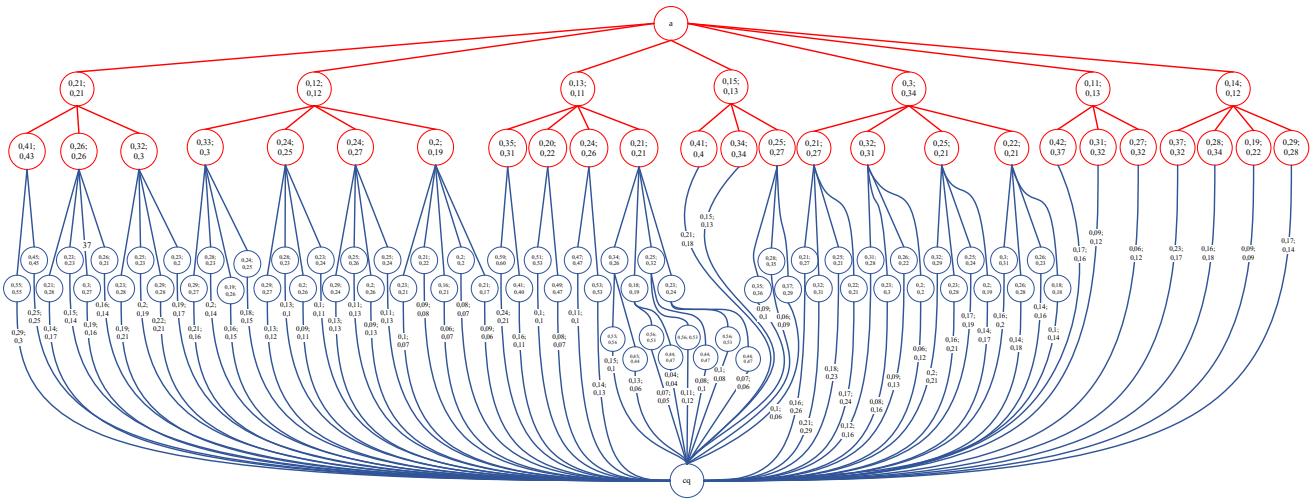

Рисунок 2.7 - Повний граф класифікатора видів/підвидів диверсності з нанесеними по вузлах метриками (SD; SC)

Легенда графу:

- **Структура.** Граф має верхню кореневу вершину «a» (агрегована категорія), проміжні рівні (групи «Design», «Equipment», «Logic/Procedure», «Function», «Lifecycle», «Signal» тощо) та нижні конкретизовані підвиди. Орієнтовані дуги задають допустимі переходи між рівнями — від загального до конкретного. Кінцева вершина «cm» відображає завершення вибору.
- **Маркування вузлів.** У кожній вершині вказано пару SD; SC у вигляді «x,xx; y,yy». Це вже нормовані часткові метрики (для поточного рівня класифікатора), одержані перетворенням експертних рангів. Значення SD тим більше, чим вища очікувана різнорідність підходу; SC тим більше, чим більша відносна вартість реалізації підвиду.
- **Сенс шляху.** Будь-який шлях L — це послідовність конкретизацій (виборів) від верхнього рівня до низу. Інтегральні оцінки шляху SD та

SC отримуються агрегуванням вузлових D_i та C_i уздовж цього шляху. Надалі саме пари SD; SC використовують для перевірки критеріїв відбору (мінімізація SC за $SD \geq REQ_d$ або максимізація SD за $SC \leq C_{lim}$).

- **Практичні обмеження.** Дужки/гілки, які в реальному проєкті несумісні (технологічно, апаратно, сертифікаційно), виключаються перед розрахунком — цим забезпечують, що жоден обчислений шлях не буде «віртуальним» і непридатним до впровадження.

Переваги побудови повного графу розширеного класифікатора:

1. формалізує усе поле можливих варіантів (від загальних рішень до дрібних підвидів);
2. дозволяє однаково трактувати гілки з різною глибиною та потужністю;
3. створює основу для автоматизованого пошуку шляху за вимогами — без «ручних» преференцій;
4. спрощує обґрунтування: кожна точка вибору у звіті чітко відображена вузлом з парою (SD; SC) і переходом до наступного рівня.

Суть алгоритму вибору за одним шляхом:

1. Побудова множини допустимих шляхів. Стандартними переборами шляхів формують множину кандидатів з урахуванням обмежень сумісності.
2. Агрегація метрик для кожного L_j. Для кожного шляху підсумовують/зважують вузлові D_i та C_i, одержуючи інтегральні $SD(L_j)$ та $SC(L_j)$. На цьому етапі доступні як «прямі» оцінки шляху (лівий блок зведеної таблиці), так і розклад на складові (для пояснюваності).
3. Перевірка критерію відбору. Далі застосовують один із двох постановок:
   - мінімізувати вартість за умови достатньої диверсності ($SD(L) \geq REQ_d$, шукати мінімальне SC(L));
   - максимізувати диверсність за обмеження вартості ($SC(L) \leq C_{lim}$, шукати максимальне SD(L)).

4. Ранжування і вибір. Шляхи, що пройшли фільтр, упорядковують за головним критерієм (вартість або диверсність), при рівності — за допоміжним. Перший у списку — L*, який і фіксується як базовий варіант для проєктування.

5. Перевірка реалізованості. Вибір L* підтверджують перевіркою на технологічну досяжність: ресурси ПЛІС, затримки/латентність, енергоспоживання, маршрутизація, сертифікаційні ризики, наявність інструментального забезпечення тощо.

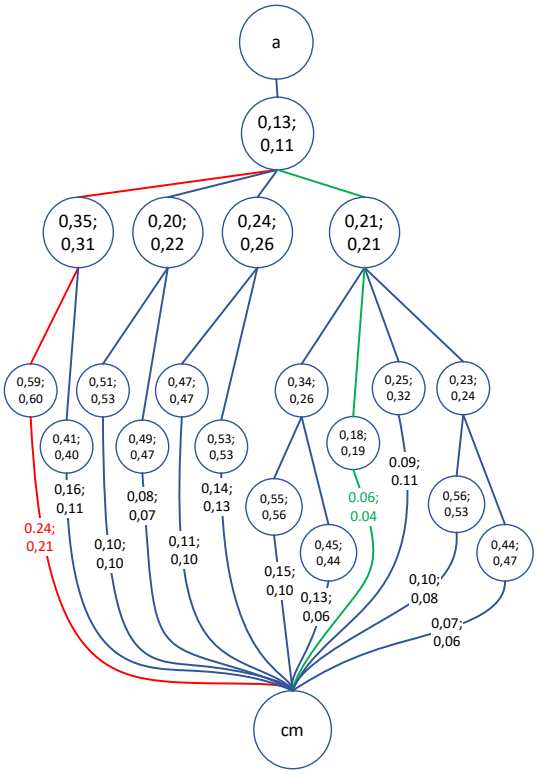

Рисунок 2.8 - Приклад пошуку оптимального шляху у графовій моделі за критеріями мінімальної вартості та максимальної диверсності

Якщо у верхніх вузлах траси обрано «сильно різні» підходи, наприклад, різних розробників або принципово відмінні архітектури, то такі рішення характеризуються високими значеннями метрики диверсності D та водночас помірно високими значеннями вартості C. У цьому випадку вже на середніх рівнях шляху досягається достатній рівень диверсності, що дозволяє надалі підбирати вузли з нижчими значеннями C_i. Така стратегія відповідає підходу «згори-донизу» і забезпечує швидке досягнення порогового значення REQ_d.

Інший сценарій має місце тоді, коли у верхніх вузлах приймаються більш помірні рішення з нижчими значеннями вартості C та середніми значеннями диверсності D. У такій ситуації шлях досягнення необхідного рівня REQ_d формується поступово — за рахунок вибору на нижчих рівнях дешевших підвидів, кожен із яких додає невеликий, але кумулятивно відчутний приріст до інтегральної диверсності SD(L). Цей підхід відомий як стратегія «знизу-догори».

Попри відмінності у логіці побудови, обидві стратегії призводять до одного й того самого результату: формується єдиний оптимальний шлях L*, який формально задовольняє критеріям відбору та має зрозуміле інженерне обґрунтування. У такому випадку перелік прийнятих вузлів із відповідними значеннями D_i та C_i дозволяє чітко показати, чому саме цей шлях було визначено як оптимальний (Рисунок 2.8).

Алгоритм вибору за одним шляхом перетворює ієрархічний класифікатор у множину прозорих, порівнюваних альтернатив, кожна з яких має інтегральні оцінки SD(L) і SC(L). Це дозволяє швидко й формально обрати базову архітектуру за заданим критерієм (мінімальна вартість при $D \geq REQ_d$ або максимальна диверсність при $C \leq C_{lim}$), зберігаючи відтворюваність і пояснюваність рішень.

2.4.7 Алгоритм вибору за комбінацією шляхів

У попередньому підрозділі розглядалася задача вибору оптимального рішення у вигляді одного шляху графової моделі. Однак у практиці проєктування систем безпеки часто виникають ситуації, коли жоден окремий шлях не забезпечує необхідного рівня диверсності при допустимих витратах, або коли вимагається створення підвищеної надійності шляхом комбінування кількох альтернатив. У таких випадках застосовується алгоритм вибору за комбінацією шляхів, що дозволяє оцінювати не окремий маршрут, а сукупність шляхів, які працюють спільно.

Цей підхід є більш узагальненим: він враховує ефекти кумулятивного підвищення диверсності за рахунок незалежних рішень, а також дозволяє точніше балансувати між вимогами безпеки та економічною ефективністю.

Метою алгоритму є обчислення сумарної диверсності та сумарної відносної вартості комбінації шляхів. Це дозволяє:

- гарантувати, що навіть у разі часткового відмовлення одного з обраних шляхів система зберігає високий (прийнятний) захист від відмов за загальною причиною;
- отримати вищі значення інтегральної диверсності, ніж у будь-якого окремого шляху;
- обґрунтувати компроміс між максимальною безпекою та раціональною вартістю реалізації.

Обчислення здійснюється за допомогою наступних виразів:

$$C_{total} = 1 - \prod_{i=1}^{n}(1 - SC_i), \tag{2.16}$$

$$D_{total} = 1 - \prod_{i=1}^{n}(1 - SD_i), \tag{2.17}$$

, де:

SD_i – часткова метрика диверсності для і-го шляху;

SC_i – часткова метрика відносної вартості для і-го шляху;

n – кількість шляхів, що входять до комбінації;

D_{total} – сумарна диверсність комбінації;

C_{total} – сумарна вартість комбінації.

D_{total} базується на принципі мультиплікативного підсилення диверсності: якщо для одного шляху існує ймовірність недосягнення необхідного рівня безпеки (1 – SD_i), то для комбінації кількох незалежних шляхів ця ймовірність перемножується. Таким чином, чим більше незалежних шляхів із високим SD_i включено, тим меншим стає добуток, і тим вищим є D_{total} . Це гарантує, що навіть

за часткового зниження диверсності одного шляху сумарний ефект від комбінації залишається високим.

C_{total} має аналогічний принцип але застосовується для вартості. Значення SC_i описують часткові витрати кожного шляху, нормовані відносно максимальної вартості. При поєднанні шляхів у систему їхній кумулятивний вплив підсумовується мультиплікативно, що дозволяє уникнути лінійного завищення витрат. Формула дає можливість оцінити реальну сукупну відносну вартість системи.

Покрокове застосування алгоритму

1. Формування множини шляхів – із графа (Таблиця 2.3) обираються ті шляхи, які потенційно можуть бути реалізовані.

2. Обчислення часткових SD_i та SC_i – метрики визначаються для кожного шляху за формулами (2.14–2.15).

3. Комбінування шляхів – формуються різні підмножини шляхів (2, 3 і більше) для аналізу.

4. Розрахунок D_{total} та C_{total} – для кожної підмножини обчислюються інтегральні показники за наведеними формулами.

5. Перевірка критеріїв – відсікаються комбінації, які не досягають необхідного рівня диверсності чи перевищують граничну вартість.

6. Вибір оптимальної комбінації – обирається та, що забезпечує найкращий баланс між безпекою та витратами.

Припустимо, що у нас є три шляхи із метриками:

$SD_1 = 0.4$, $SC_1 = 0.3$;

$SD_2 = 0.5$, $SC_2 = 0.25$;

$SD_3 = 0.6$, $SC_3 = 0.35$.

Для комбінації з двох шляхів (1 і 2):

$D_{total} = 1 - (1-0.4)(1-0.5) = 1 - (0.6 \times 0.5) = 0.7$

$C_{total} = 1 - (1-0.3)(1-0.25) = 1 - (0.7 \times 0.75) = 0.475$

Тобто сумарна диверсність комбінації двох шляхів сягає 0.7, при цьому вартість не перевищує 0.475, що може виявитися кращим варіантом, ніж використання лише одного з них.

Переваги методу:

- можливість підвищення сумарної диверсності понад рівень будь-якого окремого шляху;
- гнучкість у виборі комбінацій залежно від умов завдання;
- прозора методологія розрахунків, що дозволяє формалізувати прийняття рішень.

Недоліки:

- збільшення обчислювальної складності при зростанні кількості можливих шляхів;
- потреба у ретельному аналізі незалежності шляхів (інакше результати можуть бути завищеними);
- підвищення вартості через одночасну реалізацію кількох альтернатив.

Алгоритм вибору за комбінацією шляхів є логічним продовженням базового підходу вибору за одним шляхом. Він дозволяє створити більш стійкі рішення, підвищити рівень захищеності та адаптувати систему до жорсткіших вимог безпеки. Використання формул (2.16-2.17) для обчислення D_{total} і C_{total} робить цей процес формалізованим і дає можливість кількісно оцінювати переваги кожної комбінації. У результаті проєктувальник отримує не лише один оптимальний варіант, а набір можливих рішень, з яких можна обрати найкращий з урахуванням конкретних ресурсних та безпекових обмежень.

2.4.8 Оцінювання імовірності безвідмовної роботи двоверсійної системи аварійного захисту реактора

Для кількісного обґрунтування ефективності застосування принципу диверсності в САЗ реакторів АЕС розроблено аналітичну модель оцінювання

імовірності безвідмовної та безпечної роботи двоверсійної мажоритарної системи з урахуванням фізичних дефектів та проєктних дефектів програмно-апаратних компонентів.

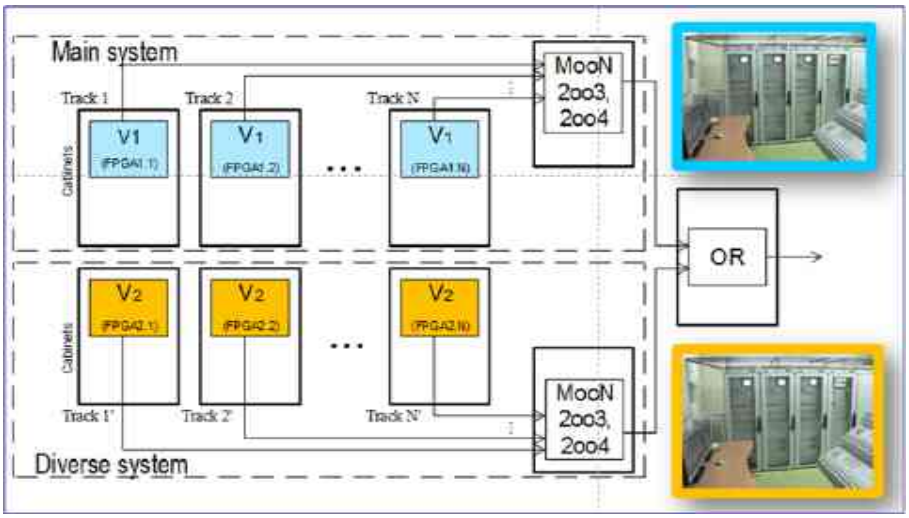

Рисунок 2.9 - Типова структура системи аварійного захисту реактора [25]

Структурна схема системи (Рисунок 2.9), що аналізується, відповідає типовій архітектурі системи аварійного захисту реактора на базі платформи «Радій»

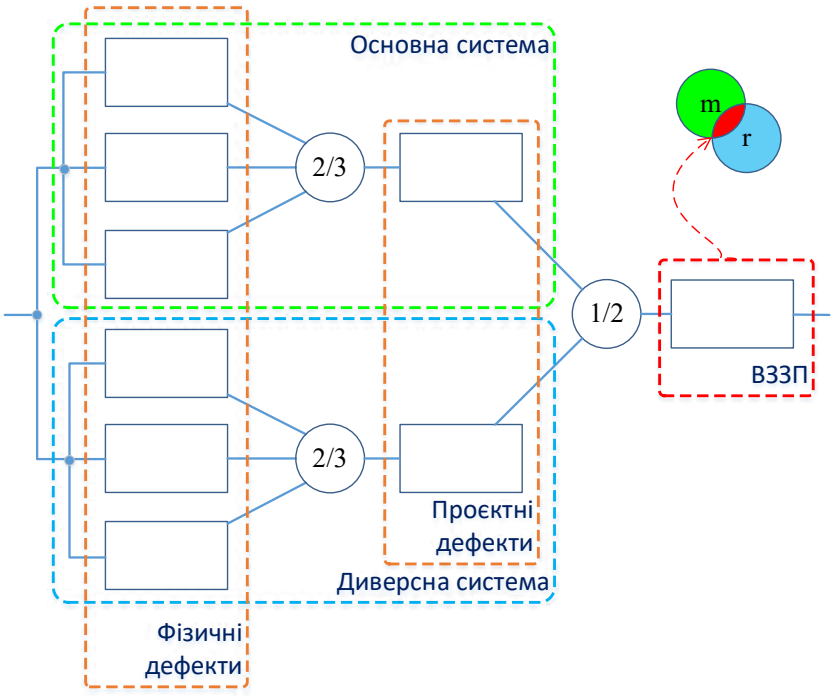

Рисунок 2.10 - Структурна схема мажоритарної системи

Кожна з двох диверсних підсистем (основна та резервна) складається з трьох незалежних каналів із мажоритарним голосуванням 2oo3, а на рівні системи в цілому реалізовано голосування 1oo2. Рисунок 2.10 відображає структурну схему мажоритарної системи з виокремленням проєктних та фізичних дефектів і відмов за загальною причиною (ВЗЗП).

Аналітична модель. Імовірність безвідмовної роботи основної (m) та резервної (r) підсистем визначається за формулами:

$$P_m = \left(3P_{fm}^2 - 2P_{fm}^3\right) \cdot P_{2/3} \cdot P_{pm}, \tag{2.18}$$

$$P_r = \left(3P_{fr}^2 - 2P_{fr}^3\right) \cdot P_{2/3} \cdot P_{pr}, \tag{2.19}$$

Тут перший множник — це імовірність безвідмовної роботи структури 2oo3 за умови фізичних дефектів, де P_{fm}, P_{fr} — імовірності безвідмовної роботи окремого каналу від фізичних дефектів відповідно для основної та резервної підсистеми. Другий множник $P_{2/3}$ - надійність мажоритарного елемента голосування 2oo3. Третій множник P_{pm}, P_{pr} - імовірності відсутності прояву проєктних дефектів у відповідних підсистемах.

Імовірність безвідмовної роботи системи в цілому з урахуванням голосування 1oo2 і відмов за загальною причиною:

$$P_s = [1 - (1 - P_m)(1 - P_r)] \cdot P_{1/2} \cdot P_{p_{abs}}, \tag{2.20}$$

де $P_{1/2}$ - надійність мажоритарного елемента голосування 1oo2; P_{pabs} - імовірність відсутності абсолютної відмови за загальною причиною.

Залежності складових моделі від метрики диверсності. Імовірності безвідмовної роботи окремих каналів та системи в цілому визначаються через показники інтенсивності відмов і метрику диверсності D:

$$P_{2/3} = e^{-\lambda_{2/3}t}, \tag{2.21}$$

$$P_{1/2} = e^{-\lambda_{1/2}t}, \quad (2.22)$$

$$P_f = e^{-\lambda_f t}, \quad (2.23)$$

$$P_p = e^{-\lambda_p t}, \quad (2.24)$$

$$P_{p_{abs}} = e^{-\lambda_p(1-D)t}, \quad (2.25)$$

$$P_{pm}(t) = e^{-\lambda_{pm}Dt}, \quad (2.26)$$

$$P_{pr}(t) = e^{-\lambda_{pr}Dt}, \quad (2.27)$$

Ключова особливість моделі полягає в тому, що метрика диверсності (D) безпосередньо входить до виразів для P_{pabs}, P_{pm} і P_{pr}: зростання D від 0 до 1 відповідає переходу від одноверсійної системи (відсутність диверсності) до ідеально диверсної. При D = 0 проєктні дефекти повністю корельовані між версіями, тому $P_{pabs} = P_p$ і $P_{pm} = P_{pr} = 1$ (проєктний дефект або проявляється в обох каналах одночасно, або не проявляється взагалі). При D → 1 кожна з версій має незалежні проєктні дефекти, тому $P_{pabs} \rightarrow 1$, а ризик абсолютної відмови за загальною причиною мінімізується.

Для розрахунків прийнято такі вихідні параметри: інтенсивність відмов λ=0,0001 [1/рік]; одиниця часу t=2,5 [рік] (відповідає 10/4 перевірочних циклів). Оскільки метою розрахунків є дослідження впливу метрики диверсності на імовірнісні показники системи, мажоритарні елементи голосування вважаються абсолютно надійними ($P_{1/2} = P_{2/3} = 1$), що дозволяє ізолювати ефект диверсності від впливу надійності допоміжних компонентів і визначити граничні оцінки виграшу функційної безпечності.

Результати розрахунків. Таблиця 2.4 містить розраховані значення показників безвідмовної та безпечної роботи системи для різних значень метрики диверсності.

Таблиця 2.4 - Розраховані значення показників безвідмовної та безпечної роботи системи

№	Значення метрики диверсності, D	Значення $P_{pm} = P_{pr}$	Значення $P_m = P_r$ ($Q_m = Q_r$)	Значення P_{abs}	Значення P_s (Q_s=1-P_s)	Відносне зменшення імовірності відмови при використанні диверсності (Q_{nD}/Q_s)
	0	1,000000	1,000000	0,999750	0,999750 (0,000250)	1,00
1	0,5	0,999875	0,999875	0,999875	0,999875 (0,000125)	2,00
2	0,75	0,999813	0,999812	0,999938	0,999938 (0,000062)	4,03
3	0,9	0,999775	0,999775	0,999975	0,999975 (0,000025)	10,00
4	0,99	0,999753	0,999752	0,999998	0,999997 (0,000003)	83,33

Аналіз результатів, які містить Таблиця 2.4, свідчить про суттєвий вплив рівня диверсності на імовірнісні показники безпечності системи. Зі зростанням метрики диверсності D від 0 до 0,99 спостерігається монотонне зменшення імовірності відмови системи Q_s та відповідне зростання показника відносного зменшення імовірності відмови Q_{nD}/Q_s, що характеризує виграш від застосування диверсності. При цьому залежність носить нелінійний характер: найбільший приріст виграшу досягається при високих значеннях метрики диверсності, що підкреслює важливість максимально повного використання можливостей версійної надмірності. Граничне значення показника відносного зменшення імовірності відмови при D = 0,99 становить 83, що є верхньою оцінкою досяжного виграшу функційної безпечності за прийнятих припущень.

Таким чином, відносне зменшення імовірності відмови при використанні диверсності Q_{nD}/Q_s є нелінійним і зростає суттєво зі збільшенням метрики диверсності, що підкреслює важливість точного вибору видів і рівня диверсності.

Запропонована аналітична модель є кількісним обґрунтуванням методу вибору видів диверсності: метрика D, що обчислюється за шляхами графової моделі класифікатора диверсності, безпосередньо визначає досяжний рівень безпечності P_s системи. Це дозволяє здійснювати обґрунтований вибір структури версійної надмірності ще на етапі проєктування, узгоджуючи вимоги до

безпечності ІКС АЕС з вимогами стандартів IEC 61508, IEC 62340 та NUREG-7007 [30; 66].

Розроблена аналітична модель використана при обґрунтуванні структури двоверсійних систем аварійного захисту реакторів АЕС на базі платформи «Радій» (ПАТ НВП «Радій»).

2.5 Висновки до другого розділу

У результаті виконання другого розділу було здійснено комплексне дослідження формалізації та кількісної оцінки диверсності в інформаційно-керуючих системах.

Проведено аналіз існуючих підходів і сформульовано загальні вимоги до побудови моделей диверсності. Показано, що класичні методики, які базуються лише на якісних характеристиках, є недостатніми для практичного використання у складних технічних системах. Визначено необхідність переходу до формалізованих моделей, що поєднують якісні класифікаційні ознаки з кількісними метриками.

Розроблено матричну модель видів диверсності. Використання матричного апарату дозволило систематизувати взаємозв’язки між видами та підвидами диверсності, а також забезпечити можливість кількісного аналізу їхніх характеристик. Запропоновані матриці диверсності та вартості створили основу для подальших розрахунків і стали зручним інструментом для порівняння альтернатив.

Запропоноване удосконалення класифікатора з введенням третіх і четвертих рівнів деталізації дозволяє більш точно структурувати можливі види диверсності та створювати кількісні метрики оцінювання “вартість/диверсність”.

Описано метод вибору видів диверсності, що поєднує експертне оцінювання, побудову метрик та алгоритмічний вибір оптимальних альтернатив. Запропонована методика включає визначення метрик диверсності й відносної вартості, їх нормування у діапазоні [0;1], а також процедуру експертного

анкетування. На основі отриманих даних сформовано інтегральні метрики, що дозволяють оцінювати як окремі елементи класифікатора, так і повні шляхи в графовій моделі. Описано стратегії вибору («згори-донизу» та «знизу-догори»), алгоритми пошуку оптимального рішення за одним шляхом і за комбінацією шляхів. Це дало змогу забезпечити гнучкість у прийнятті рішень та врахувати різні сценарії використання диверсності.

У результаті проведеного дослідження було досягнуто наступних наукових результатів:

- ***отримано*** **подальшого розвитку графова модель** опису різних програмно-апаратних версій для інформаційно-керуючих систем на програмовних платформах, з використанням деталізованого класифікатора видів диверсності, а саме: процесорних ядер, інтерфейсів, засобів синхронізації, технологічних процесів і обладнання, що дозволяє оцінювати метрики диверсності і складності залежно від множини видів і підвидів версійної надмірності.
- ***удосконалено*** **метод вибору видів диверсності** з урахуванням багаторівневої ієрархії видів версійної надмірності, включно з внутрішньою диверсністю каналів, а також відповідні метрики диверсності і складності, що дозволяє підвищити функційну безпечність і обґрунтувати структуру та процеси створення двохверсійних систем аварійного захисту реакторів АЕС.

Отримані результати створюють завершену теоретичну основу, яка у наступних розділах буде використана для апробації запропонованих методів на прикладах реальних систем та оцінювання їх ефективності.

Матеріали розділу опубліковані в [36; 37; 51; 59; 65].

РОЗДІЛ 3
РОЗРОБЛЕННЯ МЕТОДІВ І ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМНО-АПАРАТНОЇ ДИВЕРСНОСТІ ПРОГРАМОВНИХ ПЛАТФОРМ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ

3.1 Метод диверсної синхронізації

Сучасні інформаційно-керуючі системи (ІКС), що застосовуються у сфері ядерної енергетики та в інших галузях критичної інфраструктури, мають забезпечувати максимально можливий рівень надійності та стійкості до відмов за загальною причиною. Традиційні стратегії резервування та функційної надмірності, які передбачають використання кількох однакових каналів, виявляються обмеженими: у випадках синхронних впливів (наприклад, електромагнітних завад, імпульсних перенавантажень або радіаційних ефектів) ймовірність одночасної відмови різко зростає. Це створює потенційно небезпечні ситуації для систем управління технологічними процесами та особливо для систем аварійного захисту атомних станцій.

3.1.1 Сутність та метрики

Сутність методу диверсної синхронізації полягає у введенні керованих фазових зсувів між функціональними каналами, щоб розвести у часі їхні пікові події [67–70]. Канали групуються у кластери (наприклад, p1, p2), кожному каналу в кластері задається власний фазовий зсув відносно базового періоду тактування. Унаслідок цього зменшується перетин навантажень між каналами, знижується максимальна кількість одночасних перемикань та вирівнюється енергетичний профіль комутацій.

Потреба в методі обумовлена тим, що традиційне резервування добре працює проти незалежних або випадкових помилок, але залишається вразливим до відмов за загальною причиною, коли однакові впливи породжують синхронні

збої одразу у кількох каналах. Для інформаційно-керуючих систем на ПЛІС, що працюють у середовищах з підвищеними електромагнітними та радіаційними навантаженнями, часовий вимір незалежності стає критично важливим доповненням до структурної й програмної різноманітності.

Очікуваний ефект від фазового рознесення [71–76] — зменшення пікових комутаційних навантажень, зниження рівня наведень і перехідних процесів у живленні, а головне — зменшення ймовірності синхронного відмовлення кількох каналів за загальною причиною. Далі в підрозділі формалізуються метрики оцінювання ефекту та вартості такого рознесення і наводяться правила їх застосування для подальших експериментів.

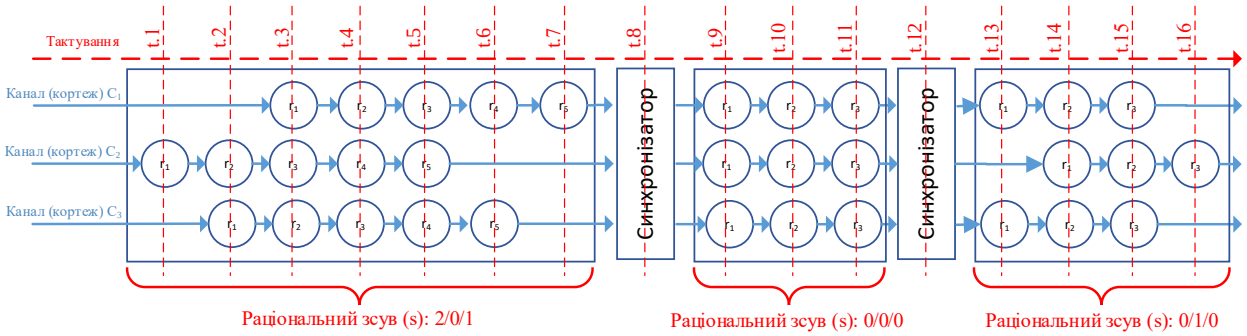

Рисунок 3.1 - Схема розподілення системи на кластери диверсної синхронізації [70]

На схемі (Рисунок 3.1) зображено фрагмент робочого періоду з періодами тактування $t_1 \ldots t_{16}$ для трьох каналів C_1, C_2, C_3. Кружечки на кожному рядку позначають функції або елементи перемикань (події) у відповідному такті. Часова діаграма поділена на три кластери: лівий, центральний і правий. Під кожною піддіаграмою наведено підпис «Раціональний зсув (s): …», де вектор s вказує величини зсувів у тактах для каналів $C_1/C_2/C_3$ відповідно. Центральна піддіаграма з позначенням s = 0/0/0 є базовим випадком без зсувів. Тут добре видно вертикальні збіги накладань, тобто значна частина подій у різних каналах відбувається в одні й ті самі такти. Саме цей профіль є еталоном для порівняння і він демонструє, чому без фазового рознесення виникають піки одночасних

перемикань і зростає чутливість до загальних причин збоїв. Одночасно, можливі ситуації, коли пікові навантаження тактів кластеру розподілені рівномірно, що нівелює необхідність у рознесенні тактів і кластер буде працювати без пікових навантажень.

Ліва піддіаграма показує конфігурацію s = 2/0/1. У цьому режимі C_1 зсунуто на два такти відносно бази, C_2 залишено без зсуву, C_3 зсунуто на один такт. Візерунок подій на трьох рядках розходиться у фазі: характерні вертикальні колонки у «гарячих» місцях зникають або стають поодинокими, а максимальна кількість одночасних перемикань у межах будь-якого t_i помітно зменшується. Іншими словами, активні вікна запуску каналів перестають повністю збігатися, що знижує як миттєве навантаження на живлення/маршрути, так і ймовірність синхронного збою за спільною причиною.

Права піддіаграма ілюструє альтернативний вибір s = 0/1/0, коли зміщено лише C_2 на один такт, а C_1 і C_3 працюють у початкових фазах. Ефект декореляції тут теж наявний, але слабший: частина накладань залишається, зокрема між C_1 і C_3 у ряді тактів; це демонструє, що різні вектори зсувів дають різну якість розведення подій і їх треба добирати, співвідносячи бажаний зиск із обмеженнями продуктивності.

Методологічно все відбувається так: спочатку з базового профілю без зсувів визначаються «гарячі зони» з високою щільністю накладань, далі для цих зон підбираються кандидати на вектор s, які забезпечують найбільше рознесення подій за умови дотримання дедлайнів і часових допусків; після візуальної й аналітичної перевірки обирається раціональна конфігурація, що мінімізує збіги подій і пікові комутації. Саме з таких розкладок беруться числа, які потім потрапляють у таблиці з первинними даними та даними «з урахуванням зсувів», а також у графіки середніх і траєкторій каналів; на них ефект розведення вже фіксується кількісно.

Процес побудови диверсної синхронізації доцільно виконувати за такою послідовністю кроків:

1. Сформувати набір функцій та каналів, які можна роділити в часі (наприклад, C_1–C_3).

2. Розділити канали та функції на групи (кластери) p_1, … ,p_n з урахуванням допустимих затримок і логічних залежностей.

3. Для кожного окремого кластера обрати фазовий зсув φ у межах періоду T так, щоб мінімізувати перекриття активних вікон і не порушити часові вимоги (мінімізувати затримки роботи системи).

4. Спроєктувати схеми синхронізації та буферизації між кластерами для коректного обміну даними.

5. Оцінити отриману конфігурацію за метриками: рівномірність фазового рознесення, частка неперекритих вікон, тривалість інтервалів спільної вразливості, зменшення пікових навантажень тощо.

6. Обрати оптимальну конфігурацію та кортеж зсувів φ, що забезпечує максимальний виграш у диверсності при прийнятних витратах реалізації.

Узагальнимо спостережувані ефекти часової декореляції у вигляді зрозумілих показників: наскільки рівномірно розподіляються події в межах періоду, як зменшується кількість одночасних перемикань і пікові навантаження, чи вирівнюється профіль активності каналів і чи зберігаються часові допуски. Ці показники дадуть змогу коректно порівнювати різні варіанти зсувів, аргументовано обирати «раціональні» налаштування та зберігати простежуваність між таблицями спостережень і графіками «до/після».

Щоб перейти від якісної схеми (кластеризація та раціональні зсуви) до формалізованого аналізу, далі фіксуємо єдину «легенду» позначень і замкнений набір робочих метрик, на яких базуватимуться всі подальші розрахунки й порівняння:

RI_t — сумарна кількість «входів в активність» перемикань усіх каналів у межах інтервалу часу *t*. Менші значення краще, бо відображають меншу синфазність одночасних запусків елементів.

RO_t — сумарна кількість «виходів з активності» перемикань після застосування зсувів у тому самому інтервалі *t*; інтерпретується аналогічно: менше — краще.

w_p — підсумкова вага періоду, яка акумулює ефект вирівнювання навантаження за весь період (відсоткова диспропорція між RO і RI у конкретному кластері): від'ємне w_p вказує на позитивний ефект застосування диверсної синхронізації.

g — загальне сповільнення системи (всіх кластерів).

Сумарний вхідний кортеж інтервалу t:

$$RI_t = \sum_{i=1}^{q} c_i \,, \tag{3.1}$$

Сумарний вихідний кортеж інтервалу t:

$$RO_t = \sum_{i=1}^{q} c_i \,, \tag{3.2}$$

, де RI_t — сумарна кількіть перемикань (активностей) в часовому інтервалі t; q — кількість каналів ($C_1 \ldots C_q$); RO_t — сумарна кількість перемикань в часовому інтервалі t після впровадження методики диверсної синхронізації; c_i — кількість перемикань і-го каналу в часовому інтервалі t. Від'ємний показник ($RO_a < RI_a$) — вказує на поліпшення (одночасних перемикань стало менше відносно бази, отже знизилися ризики синфазних піків у цьому вікні).

Інтегральна вага періоду w_p:

$$w_p = \sum_{a=1}^{t} \frac{RO_a - RI_a}{RI_a \cdot 0{,}01}, \tag{3.3}$$

, де a — індекс вікна (інтервалу) у межах часового проміжку t; RI_a та RO_a - сумарна кількість перемикань до та після впровадження методики диверсної

синхронізації відповідно; множник 0,01 у знаменнику приводить різницю до відсотків.

Інтегральний показник w_p підсумовує ці відсоткові зміни по всьому періоду T (чим більш від'ємним є w_p, тим сильніше сумарне зменшення активності після впровадження методики диверсної синхронізації. .

Сповільнення системи після впровадження диверсної синхронізації:

$$g = \frac{\sum_{a=1}^{y} \left(\sum_{i=1}^{q} s_{ai} \right)}{\sum_{a=1}^{g} t_a}, \tag{3.4}$$

g відображає відносне уповільнення тракту через введені фазові зсуви (службові паузи, розведення запусків у часі, перепланування мікрооперацій). У робочому діапазоні діє емпіричне правило: «сповільнення системи g відбувається прямо пропорційно доданій кількості зсувів s системи загалом» — що більше ми розносимо події у часі, то більшу часову ціну платимо.

Короткий приклад. Якщо у базовому варіанті без зсувів g≈0, то конфігурація з одним невеликим зсувом (умовно s=1) дає малий приріст g; варіант із трьома незалежними зсувами (умовно s=3) — приблизно утричі більший g. Тому «агресивні» розведення слід вводити лише там, де це виправдано виграшем за показниками синфазних піків і декореляції.

На рівні системної інтеграції збільшення кількості фазових зсувів s породжує додаткові цикли очікування/узгодження між кластерами (бар'єрні синхропереходи, вирівнювання FIFO, мікротактування), що підвищує коефіцієнт сповільнення g, який у робочому діапазоні зростає пропорційно s. На етапі впровадження рекомендовано мінімізувати s за умови досягнення цільового зменшення пікових перемикань (від'ємного w_p) — щоб уникнути надмірного системного сповільнення і не порушити вимоги до латентності та пропускної здатності.

Вартість методу оцінюється описово за п'ятьма напрямами:

1. Апаратні ресурси: наскільки зростає використання логічних елементів, тригерів, вбудованої пам’яті та допоміжних блоків для реалізації кластерів і механізмів синхронізації (чим менше додаткових ресурсів, тим краще).

2. Перетини тактових доменів: чим менше місць переходу даних між різними часовими областями і чим простіша схема узгодження сигналів, тим легше довести коректність роботи.

3. Додаткова затримка: оцінюється, наскільки збільшується шлях сигналу через буфери та узгоджувальні вузли; прийнятними вважаються лише варіанти, що вкладаються у встановлені допуски.

4. Енергоспоживання: порівнюється приріст потужності щодо базової конфігурації без зсувів (перевага надається варіантам із найменшим зростанням споживання).

5. Трудомісткість верифікації та сертифікації: враховуються людино-дні, необхідні для тестування, покриття перевірками та обсяг формальних доведень (менша трудомісткість за умови повноти доказової бази вважається кращою).

Під час прийняття рішень спочатку перевіряються «жорсткі» обмеження: усі часові допуски виконані, приріст потужності не виходить за ліміти, кількість та складність перетинів тактових доменів прийнятні для технології та процесу верифікації. Далі порівнюються варіанти за метриками ефекту, які вже наведені у підрозділі (сумарні входи, сумарні виходи, локальна вага інтервалів, підсумкова вага за період, середня щільність активностей). Перевагу віддаємо тим конфігураціям, де у більшості вікон спостерігаються менші піки одночасних переходів, де менше негативних локальних ефектів і більший позитивний сумарний результат за період, а розклад подій у часі є рівномірнішим. Якщо два варіанти дають близький ефект, остаточний вибір робимо на користь того, що має нижчу вартість реалізації: менше додаткових ресурсів, менша затримка, нижче енергоспоживання та простіша верифікація.

Диверсна синхронізація з кластеризацією зменшує синфазні стани, скорочує «вікна спільної вразливості», знижує пікові енергонавантаження та декорелює

прояв загальних причин відмов. Впроваджені формули забезпечують операційну оцінку локальних і інтегральних ефектів у часовій сітці; у поєднанні з інтегральними метриками відбору вони дають інженерну основу для вибору оптимальної конфігурації диверсної синхронізації.

3.1.2 Принципи та послідовність реалізації

У цьому підрозділі деталізовано практичні принципи побудови диверсної синхронізації й наведено відпрацьовану послідовність дій від аналізу часових діаграм до впровадження на рівні ПЛІС з подальшою перевіркою. Концептуально підхід спирається на кластеризацію діаграм активності каналів та раціональні фазові зсуви в межах базового періоду тактування [77–81].

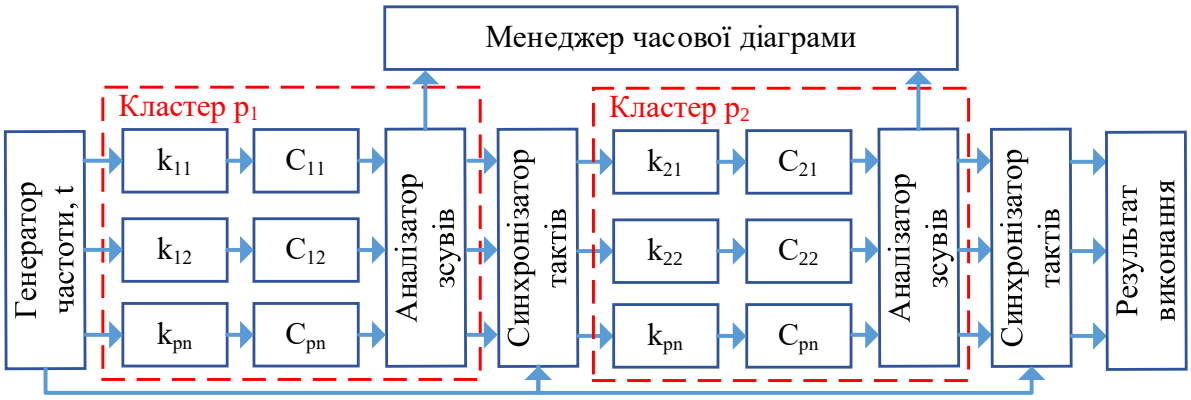

Рисунок 3.2 - Кластеризація діаграм активності та застосування раціональних фазових зсувів [70]

На схемі (Рисунок 3.2) показано, як канали $C_1 \ldots C_3$ розносяться у два кластери p_1, p_2, після чого у межах періоду T кожному кластеру задають свій зсув. Важливо, що зсуви обираються з урахуванням метричних критеріїв — зменшення піків одночасних перемикань вихідного каскаду відносно вхідного, покращення локальних ваг часових проміжків та збільшення позитивної інтегральної ваги повного періоду. Раціональний характер зсувів потрібен з двох причин: по-перше, для спрощення аналізу найгірших випадків (повторюваність картини за

дискретною сіткою часу), по-друге, для коректного узгодження з внутрішніми генераторами такту в ПЛІС і схемами «clock enable» без ризику появи некерованих «плаваючих» фаз.

На схемі показано приклад із трьома каналами C_1-C_3 та комбінаціями зсувів ⟨2,0,1⟩, ⟨0,0,0⟩, ⟨0,1,0⟩. У реалізації з впровадженням зсувів, блок C_1 запускається із запізненням на два такти, C_3 — на один такт, що забезпечує розрив колишніх скупчень без зміни сумарної кількості подій за період.

Вихідні дані — кортежі активності каналів $C_1 \ldots C_q$, дискретизовані на однакову сітку інтервалів у межах базового періоду T. Потрібно підібрати циклічні зсуви $(k_1,\ldots,k_q)$ та, за потреби, розбиття на кластери p_1, p_2 так, щоб мінімізувати пікове одночасне навантаження (максимум суми спрацьовувань в інтервалі часу) і покращити інтегральні показники рівномірності (максимальне зменшення w_p) за умови: дотримання часових залежностей між блоками, прийнятної кількості перетинів часових областей та складності їх узгодження, виконання обмежень на затримку і енергоспоживання. Ці обмеження трактуються як інженерний контракт: будь-який виграш за піками не може досягатися ціною порушення таймінгів чи погіршення коректності обміну даними.

Обмеження та важливі моменти побудови системи з диверсною синхронізацією:

1. **Декореляція у часовому вимірі як окрема вісь диверсності.** Після того, як канали вже рознесено за реалізацією та логікою (диверсність алгоритмів/ресурсів), додавання часової розбіжності дає незалежність від спільних «миттєвих» причин відмов. Ціль — зменшити спільні «вікна вразливості», коли одне збурення може одночасно влучити у кілька каналів.

2. **Кластеризація за подібністю активності.** Канали з близькою структурою активних ділянок часу не варто тримати в одному часовому проміжку (кластері): їх слід розвести так, щоб «піки» одного часового вікна потрапляли у «западини» іншого. Це відкрито читається на діаграмах активності (Рисунок 3.1).

3. **Раціональні зсуви в межах T.** Зсуви задаються як сталі частки періоду. Це дозволяє легко реалізувати їх у ПЛІС через дільники/множники такту

або через «clock enable» з визначеним ритмом, а також гарантує повторюваність шаблону у кожному періоді T без накопичення фазової похибки.

4. **Перевага «clock enable» там, де можливо.** У більшості ПЛІС доцільно лишати один опорний такт і формувати зсуви логікою «enable», а не множити функції тактування. Це зменшує кількість перетинів часових доменів, спрощує верифікацію та підвищує стійкість до джитеру.

5. **Локальна оптимізація під глобальний критерій.** На етапі підбору зсувів орієнтуємось на локальні показники у кластерах (падають піки «входів/виходів», покращуються локальні ваги), але рішення приймаємо за інтегральним ефектом за період (позитивна підсумкова вага, стабільність розкладання активності).

6. **Сувора дисципліна узгодження між кластерами.** Там, де кластери обмінюються даними, потрібні коректні протоколи синхронізації (буфери, рукостискання, прапорці валідності/готовності), аби фазові зсуви не породжували метастабільність або втрату імпульсів/даних.

7. **Відсутність «комбінованого» керування тактом.** Будь-яке комбінаційне керування лінією такту заборонене; усі перемикання — через рекомендовані примітиви виробника ПЛІС або через «enable». Це валжива частина в контексті надійності.

8. **Верифікація як частина проєктування.** Моделювання кортежів тактування і підрахунок метрик виконуються на кожній ітерації підбору зсувів, доки не буде отримано стабільне покращення без порушення часових допусків.

Послідовність реалізації (покроково):

1. **Збір вихідних діаграм активності.** Для кожного каналу $C_1 \ldots C_n$ формуються часові діаграми активностей (кількості перемикань) в межах базового періоду T, включно з ділянками комутації, опитування датчиків, обчислень і формування виходів. Якщо система багатоперіодна, усі діаграми зводяться до спільного базового періоду.

2. **Первинний аналіз накладань.** Визначаються вікна, де спостерігаються піки одночасних перемикань. На основі цих даних попередньо

оцінюється, які канали бажано зсунути для поєднання пікових перемикань одного каналу з мінімальними активностями в інших.

3. **Кластеризація каналів.** Канали $C_1 \ldots C_q$ доцільно групувати у кластери $p_1 \ldots p_n$ не лише за структурними залежностями, а й за подібністю часових профілів активності. Єдиний (однаковий) зсув для всього ланцюга обробки рідко забезпечує істотне зменшення накладань; натомість кластеризація дає змогу призначати власні зсуви усередині кожного кластера, розносячи «пікові» вікна одного підмножини каналів у «западини» іншої. Межі кластерів обираються так, щоб мінімізувати накладання сумарних спрацьовувань між кластерами у межах періоду T, зберегти причинно-часові залежності там, де спільний запуск є необхідним, і не збільшити надмірно кількість міжкластерних перетинів та узгоджень. За цих умов кластеризація підвищує ефективність диверсної синхронізації порівняно з однотипним зсувом для всієї системи, забезпечуючи краще зниження пікових значень R_{pt} і більш виражене покращення інтегральних метрик (зокрема, від'ємного w_p) без порушення функціональних вимог.

4. **Вибір сітки раціональних зсувів.** Зсуви задаються у вигляді раціональних часток періоду T (циклічні зсуви), що гарантовано відтворюються у кожному періоді та коректно реалізуються засобами ПЛІС (переважно через clock enable). Вибір сітки зсувів є компромісом між ефектом і «вартістю»: очікуване зменшення піків (падіння R_{1n} відносно R_{0n}, від'ємний w_p , локальне зниження накладань у «гарячих» вікнах) протиставляється сповільненню g, що зростає зі збільшенням сумарних зсувів s. Початкове рішення щодо кроку/роздільності сітки узгоджується з технічним завданням (дедлайни, пропускна здатність, допустимі затримки, політика перетинів часових областей), після чого обираються кандидатні вектори зсувів і перевіряються на табличних кортежах «до/після». Фінальний вибір фіксується для тієї конфігурації, де досягається стійке поліпшення метрик ефекту за допустимого g і без порушення часових допусків та вимог до верифікації/трасування.

5. **Попереднє моделювання зсувів.** На часових діаграмах застосовується обраний набір зсувів; оцінюється вплив на «вікна активності». Перевіряємо, чи падають піки одночасних переходів та чи не з’явилися нові конфліктні вікна.

6. **Оцінювання за метриками.** Для отриманої конфігурації підраховуються показники у вікнах періоду: сумарні кількості перемикань до та після запровадження диверсної синхронізації, локальні ваги, підсумкова вага періоду, середня щільність активностей. Порівнюємо з базовим варіантом без зсувів і з альтернативними наборами зсувів.

7. **Узгодження з часовими допусками.** Перевіряється, що зсуви не порушують логічні залежності (готовність даних до моментів споживання), а також що додаткові буфери/очікування не виводять сигнали за встановлені обмеження затримок технічного завдання.

8. **Вибір способу реалізації у ПЛІС.** Якщо можливо — реалізуємо через «clock enable» з відповідною модуляцією дозволу у межах T. За потреби рознесення за кількома тактами — використовуємо рекомендовані виробником примітиви керування тактами.

9. **Проєктування каналів узгодження між кластерами.** Для сигналів, що перетинають межу p1↔p2, вводяться протоколи «готовність/валідність» або глибинні буфери з контролем переповнення/спустошення. Спеціально перевіряється коректність у сценаріях збурень.

10. **Налаштування обмежень трасування.** У інструментах синтезу/трасування задаються обмеження на мультициклові шляхи, винятки для «нечасових» зв’язків, а також розміщення логіки так, щоб кластери не «перетікали» один в одного й не збільшували розкид затримок.

11. **Моделювання кортежів тактування.** Формується набір випробувальних сценаріїв на весь період T з фіксацією кількостей перемикань у кожному вікні. Порівнюються часові діаграми «було/стало» для фінально обраних зсувів.

12. **Перевірка стійкості.** Переглядаються варіанти «кутових» режимів: зміни температури/напруги живлення, джитер опорного такту, відмінності в затримках ліній. Контролюється, що вікна активності не сповзають у небажані перекриття.

13. **Оцінка «вартості» впровадження.** Для обраної конфігурації документується використання ресурсів, додаткова затримка, перетини часових областей, енергоспоживання та трудомісткість верифікації. Якщо будь-який показник виходить за допустимі межі — повертаємось до кроків 4–7 і коригуємо зсуви/кластеризацію.

14. **Фіналізація та підготовка до серійного застосування.** Затверджуються параметри кластерів, значення зсувів, шаблони «clock enable», правила для сигналів, що перетинають кластери, і пакет перевірок. Готується опис для документації та сертифікації.

Реалізація диверсної синхронізації — це кероване трансформування часової організації багатоканального тракту: спочатку виконується профілювання хронограм і виявлення «гарячих зон», далі канали групуються у кластери та добирається вектор зсувів s, що мінімізує R_{pt}, зменшує пікове значення R_{1n} відносно R_{0n} і забезпечує більш від’ємний w_p при дотриманні обмеження на коефіцієнт сповільнення g, і лише після цього рішення фіксується в ПЛІС (маркери фаз, FIFO/enable, узгодження між кластерами) з перевірками та регламентованими реакціями. Така дисципліна дає відтворюваний ефект часової декореляції, знижує імовірність відмов за спільною причиною та зберігає працездатність у межах заданих бюджетів латентності й пропускної здатності.

3.1.3 Експериментальні оцінки. Моделювання кортежу тактування

Метою цього підрозділу є кількісне підтвердження ефективності диверсної синхронізації шляхом порівняння двох варіантів часової організації каналів: вихідного синфазного розкладу без зсувів (варіант А) та розкладу після кластеризації каналів з раціональними фазовими зсувами в межах одного

базового періоду T (варіант B). Для обох варіантів будується кортеж тактування — впорядкований опис подій для каналів C_1–C_3 у межах періоду T, із фіксацією моментів кількостей перемикань та тривалості активних відрізків.

Оцінювання спирається виключно на метрики: первинні дані RI_t, дані після впровадження зсувів RO_t, підсумкова вага періоду w_p (сума локальних ваг за всі інтервали у відсотках) та середня щільність активностей g (відношення загальної кількості подій до тривалості періоду). Інтерпретація метрик така: RI_t і RO_t відображають піковість одночасних перемикань; w_p характеризує інтегральний ефект рівномірності за весь період; g фіксує фонову інтенсивність подій і слугує довідковим тлом для коректного порівняння варіантів із однаковою функціональністю.

Щоб виключити методологічні викривлення, у двох наборах даних зберігається незмінною як функціональна поведінка каналів, так і загальна кількість подій за період T; змінюється лише фазова організація (кластеризація та зсуви). Це означає, що g у варіантах A і B очікувано однакова, а відмінності виявляються саме у формах розподілу подій у часі — у піках RI_t/RO_t та в інтегральному показнику w_p. Використання раціональних зсувів забезпечує детермінованість шаблону й відтворюваність вимірювань у кожному періоді без накопичення фазової похибки.

Джерелом вихідних даних слугують часові діаграми активності каналів, з яких знімаються моменти переходів і тривалості активних відрізків. Після дискретизації періоду T у кожному інтервалі підраховуються RI_t та RO_t, обчислюється w_p як сума за період. Для прозорості викладення розділ відкривається двома зведеними таблицями: спочатку наводяться результати для вихідного синфазного варіанту (A) з описом походження цифр і характеристикою отриманої картини; далі подаються результати після застосування диверсної синхронізації (B) з аналізом змін «A → B» та їх інженерною інтерпретацією. Після таблиць демонструються узгоджені з ними графіки кортежів тактування, на яких наочно видно зменшення пікової синфазності, згладження локальних «перекосів» і зростання інтегрального ефекту.

Нижче подано три взаємопов’язані таблиці. Вони утворюють єдину процедуру: Таблиця 3.1 відображає параметри зсувів і підсумкові статистики, Таблиця 3.2 — вихідні первинні кортежі без зсувів, і Таблиця 3.3 - кортежі після застосування зсувів з поміжінтервальним аналізом.

Таблиця 3.1 – Налаштування та статистичні дані зсувів кортежів [70]

Зсув кортежу C_1, тактів (k_1)	2
Зсув кортежу C_2, тактів (k_2)	0
Зсув кортежу C_3, тактів (k_3)	1
К-ть спрацьовувань., %:	**-39,40**
Макс.кількість одночасних перемикань R_{0n}	**25**
Макс.кількість одночасних перемикань R_{1n}	**22**

Пояснення до таблиці:

– Зсув кортежу C_1 (k_1) = 2 такти; зсув C_2 (k_2) = 0 тактів; зсув C_3 (k_3) = 1 такт. Це циклічні (раціональні) зсуви в межах періоду дискретизації, що застосовуються до векторів подій кожного каналу. Вони визначають, на скільки тактів пересувається ланцюг функцій роботи каналу в межах одного періоду.

– К-ть спрацьовувань, %: −39,40. Узагальнений показник зменшення (при від’ємних числах) кількості одночасних перемикань у пікових інтервалах після застосування зсувів відносно вихідної картини.

– Макс. кількість одночасних перемикань RI = 25 (до зсувів) та RO = 22 (після зсувів). Це максимум рядка сумарних спрацьовувань по інтервалах «до» та «після» відповідно. Падіння з 25 до 22 означає зниження піку на 12%, що прямо відображає розрив синфазних скупчень переходів у часі.

Таблиця 3.2 - Первинні дані кортежів [70]

	t.1	t.2	t.3	t.4	t.5	t.6	t.7	t.8	t.9	t.10	t.11	t.12	t.13	t.14	t.15	t.16
Кортеж C_1	6	5	4	3	5	2	8	1	4	7	4	6	2	7	2	8
Кортеж C_2	10	1	1	4	2	2	5	8	1	7	3	8	3	1	2	8
Кортеж C_3	9	3	4	6	5	5	4	5	8	9	7	1	7	4	7	5
Сума спрацьов. (R_{pt})	25	9	9	13	12	9	17	14	13	23	14	15	12	12	11	21
Середнє арифм.	8,3	3	3	4,3	4	3	5,7	4,7	4,3	7,7	4,7	5	4	4	3,7	7

У цій таблиці кожен стовпчик t.1…t.16 — це інтервал дискретизації в межах одного періоду. Для кожного інтервалу наведено:

- рядки **«Кортеж $C_1/C_2/C_3$»** — кількість спрацьовувань (перемикань) у відповідному каналі;
- рядок **«Сума спрацьовув.» (RI_{pt})** — арифметична сума трьох каналів (власне те, що характеризує **синфазність** на цьому інтервалі);
- рядок **«Середнє арифм.»** — середня кількість спрацьовувань на канал у даному інтервалі (RI_{pt} / 3).

Таблиця 3.3 - Дані кортежів з урахуванням зсувів [70]

	t.1	t.2	t.3	t.4	t.5	t.6	t.7	t.8	t.9	t.10	t.11	t.12	t.13	t.14	t.15	t.16	t.17	t.18
Кортеж C_1			6	5	4	3	5	2	8	1	4	7	4	6	2	7	2	8
Кортеж C_2	10	1	1	4	2	2	5	8	1	7	3	8	3	1	2	8		
Кортеж C_3		9	3	4	6	5	5	4	5	8	9	7	1	7	4	7	5	
Сума спрацьов. (R_{pt})	10	10	10	13	12	10	15	14	14	16	16	22	8	14	8	22	7	8
Середнє арифм.	3,3	3,3	3,3	4,3	4	3,3	5	4,7	4,7	5,3	5,3	7,3	2,7	4,7	2,7	7,3	2,3	2,7
Кількість спрацьовувань, %:	**-60**	**11**	**11**	**0**	**0**	**11**	**-12**	**0**	**7,7**	**-30**	**14**	**47**	**-33**	**17**	**-27**	**4,8**		

Стовпчики t.17 і t.18 відображають значення, що «перейшли через межу періоду» внаслідок циклічних зсувів. Для розрахунку підсумкових метрик ці інтервали агрегуються з початком періоду відповідно до прийнятої політики звітності.

Після застосування зсувів, рядки C_1,C_2,C_3 — це ті самі вихідні послідовності, циклічно зсунуті на вказану кількість тактів.

Далі знову обчислено «Сума спрацьовув.» (RO_{pt}) і «Середнє арифм.» по кожному інтервалу.

У нижньому рядку подано «Кількість спрацьовувань, %» — відносну зміну суми кількості спрацювань (знаком «–» позначено зменшення). У таблиці наведено характерні приклади:

- t.1: −60% (з 25 до 10),
- t.2: +11% (з 9 до 10),
- t.7: −12% (з 17 до 15),

– t.10: −30% (з 23 до 16).

Алгоритм побудови та перевірки (покроково):

1. Фіксація вихідних кортежів. Для кожного каналу формується послідовність довжини (кількість інтервалів у періоді), де елемент — це число спрацьовувань у відповідному інтервалі.

2. Обчислення синфазної суми. Складається рядок RI_{pt} як покомпонентна сума трьох первинних кортежів і відповідний рядок середнього арифметичного (RI_{pt} / 3).

3. Застосування зсувів. До кожного кортежу застосовується циклічний зсув на вказану кількість тактів.

4. Повторне сумування. Знову обчислюються RO_{pt} і середнє арифметичне по кожному інтервалу — це базові дробові показники для «після».

5. Покомпонентне порівняння. Для кожного інтервалу розраховується відносна зміна «Кількість спрацьовувань, %» — наскільки змінилась кількість перемикань до та після запровадження диверсної синхронізації. Саме цей рядок показує «де саме» зникли піки і наскільки збалансувалася картина.

Монокластер vs мультикластер. У наведеному кейсі показано «монокластерне» розведення каналів із трьома зсувами (k_1,k_2,k_3). Для мультикластерних конфігурацій (коли канали групуються у кілька підсистем з власними зсувами) критерій відбору доповнюється вимогою не створювати нових глобальних піків при узгодженні між кластерами. Практично це означає: спочатку добираємо зсуви всередині кластера (мінімізуємо локальні піки), далі перевіряємо міжкластерний профіль сумарної активності та за потреби коригуємо фази кластерів відносно одне одного.

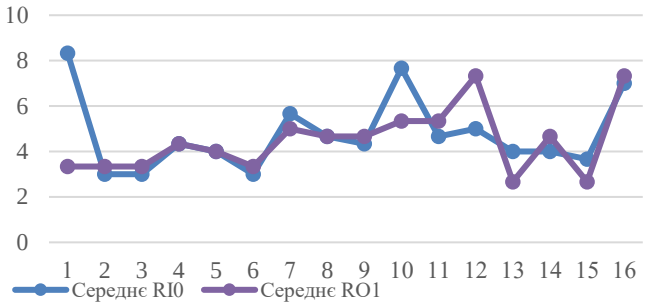

Рисунок 3.3 - Середні значення RI та RO по інтервалах $t_1 \ldots t_{16}$ (усереднення по каналах) [70]

На цьому графіку (Рисунок 3.3) подано середні (по каналах) значення «входів» (RI) і «виходів» (RO) для кожного інтервалу дискретизації періоду. Крива RI (синя) відображає, де у періоді зосереджені пікові сумарні кількості перемикань первинних кортежів; крива RO (фіолетова) — сумарна кількість перемикань після запровадження зсувів.

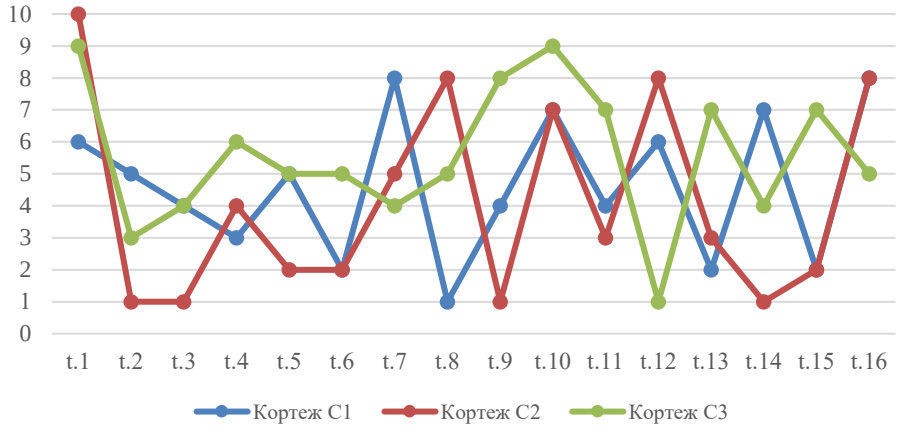

Рисунок 3.4 - Кортежі каналів C_1–C_3 до запровадження зсувів [70]

Рисунок 3.4 та Рисунок 3.5 містять кількість перемикань трьох каналів C_1–C_3 у дискретах часу t.1…t.16: по осі X — такти, по осі Y — чисельність подій у відповідному такті (кожна ламана відповідає одному каналу). Такий формат зручний для візуальної оцінки синфазних піків: чим частіше вершини різних кривих збігаються по X, тим більша ймовірність одночасних комутацій.

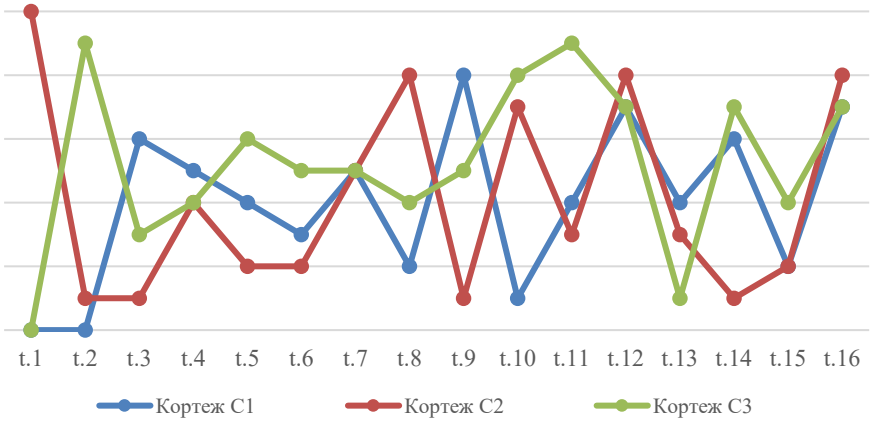

Рисунок 3.5 - Кортежі каналів C_1–C_3 після запровадження зсувів [70]

Порівняння “до” і “після”. На першому графіку (до запровадження зсувів) помітні співпадіння синхронних підйомів на кількох ділянках періоду — ламані для C_1, C_2 і C_3 часто “накладаються” одна на одну або формують близькі за часом вершини. Це проявляється як характерна «гребінка» вертикальних збігів, що дає високі локальні значення кількостей перемикань. Другий графік (після запровадження зсувів) демонструє іншу картину: піки зміщено в різні такти, частина великих співпадінь розірвана, вершини каналів розташовуються «по драбинці», а не «в один стовпчик». У результаті щільність накладань помітно зменшується, а залишкові збіги мають меншу амплітуду.

На першому графіку у початкових тактах видно близькі за фазою екстремуми двох-трьох каналів, то на другому ці ж канали дають послідовні (а не одночасні) піки — один канал виходить на максимум, тоді як інші перебувають у спадаючих або плато-ділянках. Аналогічна ситуація спостерігається у середній частині періоду: там, де раніше вершини збігались у сусідніх тактах, тепер вони розведені щонайменше на один-два такти, що добре видно за «зсувом» ламаних відносно одна одної.

Візуальна різниця між графіками відповідає очікуваному ефекту від диверсної синхронізації: знижується кількість та амплітуда синфазних піків, розподіл подій стає рівномірнішим у часі, а значення RO (після зсувів) — нижче за RI (до зсувів). Це прямо корелює з поліпшенням інтегральної метрики w_p

(від'ємний сумарний приріст) і підтверджує, що обраний вектор зсувів ефективно декорелює канали без порушення загальної функціональності тракту.

Проведене моделювання кортежів тактування для трьох каналів у межах одного періоду з дискретизацією на 16 інтервалів і подальше застосування раціональних зсувів підтвердило працездатність і практичну корисність методу диверсної синхронізації:

На рівні системної інтеграції збільшення кількості фазових зсувів (s) неминуче породжує додаткові бар'єрні узгодження між кластерами (синхропереходи, вирівнювання FIFO, мікротактування), що прямо підвищує коефіцієнт сповільнення системи (g) приблизно пропорційно величині введених зсувів. У цій дисертації практичні механізми диверсної синхронізації реалізовано в тракті Poll–Stream–Alg–Comm модуля БФЗ-7. Зазначений модуль перебуває у промисловій експлуатації на енергоблоках АЕС України, а багаторічна робота значної кількості інсталяцій (десятки екземплярів на майданчик) підтверджує його експлуатаційну надійність та відтворюваність параметрів.

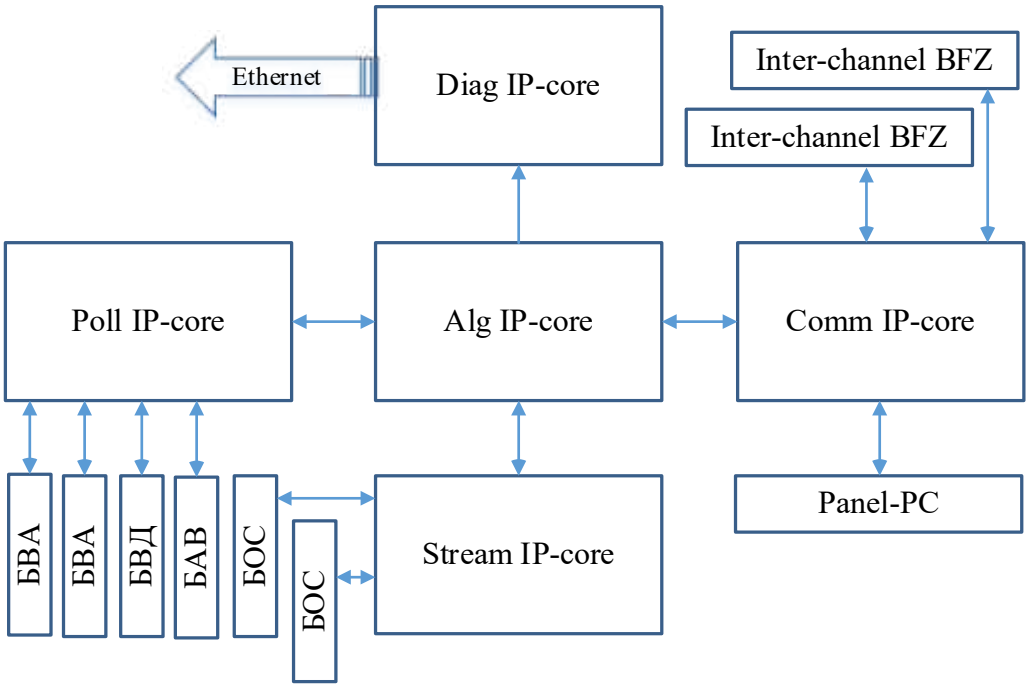

Рисунок 3.6 - Структурна схема комунікацій IP-ядер модуля БФЗ-7

Рисунок 3.6 відображає функціонально-комунікаційну архітектуру модуля БФЗ-7, через яку проходять потоки даних, що надалі синхронізуються/зсуваються в методі диверсної синхронізації:

- Poll IP-core — вузол опитування периферійних блоків (БВА, БВД, БАВ). Він формує кадрові/пакетні вибірки з первинних сигналів і є початковою точкою часової організації; період/фаза опитування далі впливає на вибір зсувів для каналів.
- Stream IP-core — ядро потокового тракту агрегації/маршрутизації кадрів даних.
- Alg IP-core — алгоритмічний блок (обчислення, перевірки, стан-машини). Для збереження функціональної тотожності каналів використовується ідемпотентна обробка: зміни торкаються лише моментів запуску (фаз), а не змісту операцій.
- Comm IP-core — комунікаційний блок кадрування та контролю цілісності. Він формує вихідні кадри для обміну з Panel-PC і приймає зворотні службові повідомлення.
- Diag IP-core — діагностика та телеметрія. Окремий вихід Ethernet забезпечує незалежний діагностичний канал, який не впливає на масив або обробку даних модуля, але дає змогу контролювати працездатність та помилки модулів всієї кроссплати.
- Inter-channel BFZ — міжканальні (перехресні) оптичні зв’язки, що забезпечують додатковий контроль узгодженості між каналами за принципом 2oo3 та обмежують вплив потенційних загальних причин відмов.
- Panel-PC — панельний комп’ютера, встановлений в шафі з трьома кросс-платами, який отримує кадри з трьох модулів БФЗ-7 (Comm IP-core), відображає діагностичну інформацію та при певному рівні дозволу, дає можливість змінювати параметри (уставки) обробки інформації модулями.

З метою ефективного розподілу логічного навантаження та зменшення кількості одночасних перемикань у FPGA, у структурі модуля впроваджено два синхроімпульси: SystemSyncRead та SystemSyncWrite (Рисунок 3.7). Ці сигнали відповідають за координацію обміну даними між ядрами, зокрема, за організацію фаз отримання та передачі інформації.

Кожне з ядер передає в Alg-IP отримані ззовні дані, та, відповідно, отримує оброблені результати для подальшої передачі зовнішнім споживачам. Важливою особливістю є те, що обробка даних може розпочинатися лише за умови наявності повного обсягу інформації з усіх джерел. Через це послідовний запуск кожного з восьми кластерів обробки інформації призводив би до суттєвого збільшення загального часу обробки одного циклу, що є неприпустимим в умовах функціонування систем безпеки.

З огляду на зазначене, було прийнято архітектурне рішення про поділ ядер на два логічні кластери: кластер прийому даних (що активується синхроімпульсом SystemSyncRead) та кластер передачі даних (SystemSyncWrite). Усі ядра отримують паралельний сигнал на запуск фази зчитування зовнішніх даних. Після їх централізованої обробки ядром Alg-IP, система формує імпульс на запуск фази передачі, яка так само виконується паралельно.

Таким чином, повний цикл роботи системи умовно поділяється на три кластери:

1. прийом зовнішніх даних (кластер читання);
2. обробка даних (кластер Alg-IP);
3. передача результатів (кластер запису).

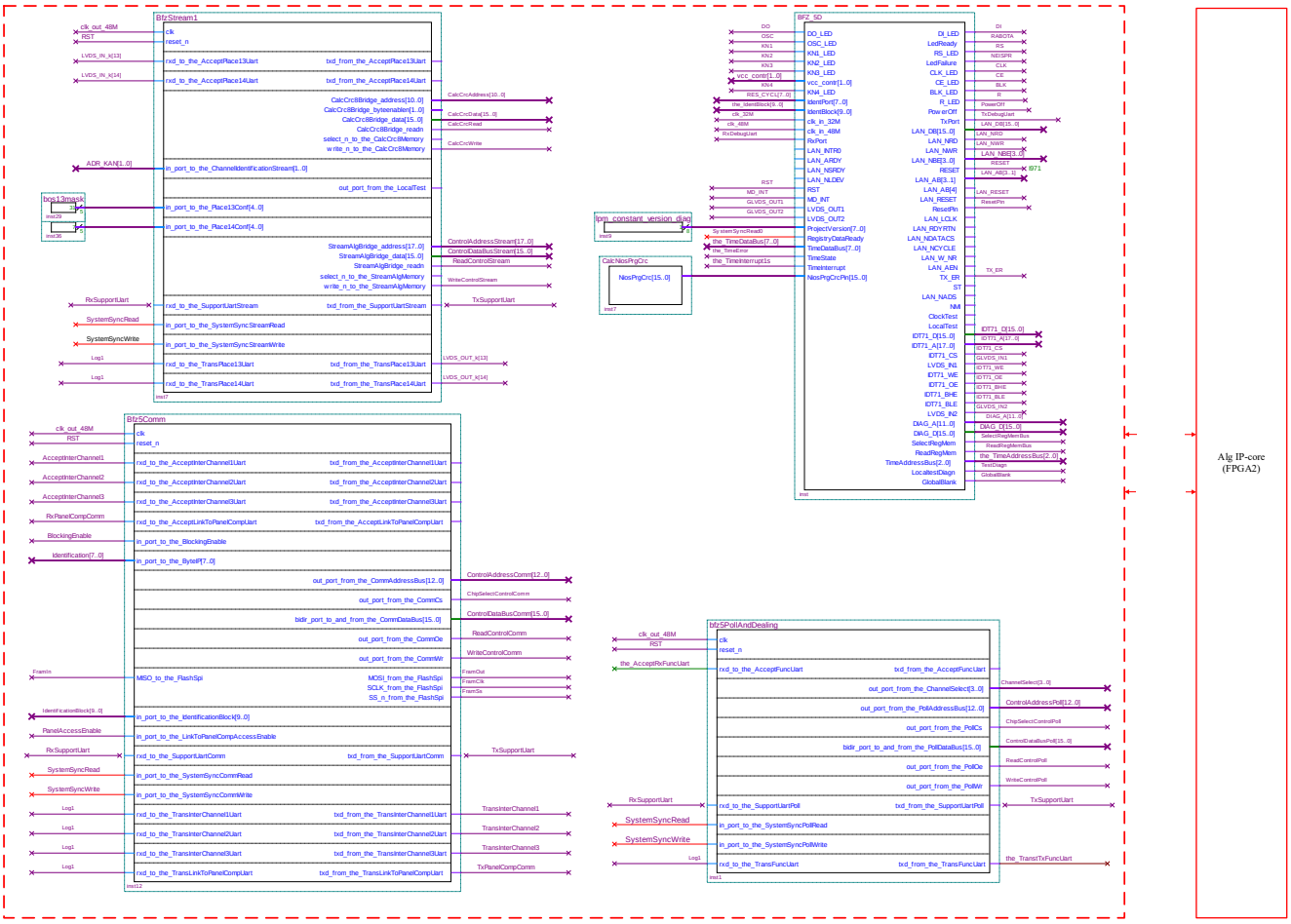

Рисунок 3.7 - Ілюстрація контролю фаз отримання та передачі даних Alg ядром для чотирьох IP-ядер модуля БФЗ-7

Кожен з кластерів виконується у власному часовому проміжку з контрольованим зсувом, що дозволяє мінімізувати конфлікти в логіці, знизити кількість пікових перемикань та водночас забезпечити високу швидкодію проєкту без шкоди для часових характеристик системи.

Сумарно теоретично та практично підрозділ демонструє, що диверсна синхронізація дає чіткий, кількісно доведений ефект і є технологічно керованим інструментом зниження синфазності у багатоканальних платформах. Описана процедура — від побудови первинних кортежів з верифікацією на таблицях і графіках до практичного впровадження на працюючих модулях АЕС — є відтворюваною та інженерно придатною для серійної інтеграції: вона забезпечує зниження пікової синфазності без зміни обсягу подій за період і не створює нових критичних піків у часовому профілі системи..

3.2 Метод структурно-просторової диверсності матриць логічних елементів та таймінгу проходження сигналів

Ідея методу полягає у свідомому створенні диверсних варіантів одного й того ж проєкту шляхом зміни налаштувань синтезу/розміщення/трасування та (за потреби) легкої реорганізації структури [65; 82–90]. Такий підхід формує альтернативні просторові карти розміщення логічних елементів і відмінні часові профілі шляхів, зменшуючи кореляцію відмов за загальною причиною між основним і диверсним каналами. Метод може застосовуватися як для побудови двох незалежних каналів (основний/диверсний), так і для композицій із іншими підходами диверсної реалізаціх.

3.2.1 Сутність та метрики

Сутність методу. Метод структурно-просторової диверсності полягає у свідомому формуванні альтернативних фізичних реалізацій одного й того самого проєкту ПЛІС: змінюється карта розміщення логічних елементів, профіль трасування та розташування критичних шляхів і запасів таймінгу, але функціональна специфікація і зовнішні інтерфейси лишаються незмінними. Досягається це двома взаємодоповнювальними впливами:

- компіляційним (інший набір параметрів інструмента: перемикач *Use smart compilation*, профіль Optimization Technique — *Speed / Balanced / Area*, інші службові параметри збірки);
- структурним легкого рівня (незначні, але технічно корисні перестановки ієрархії, розведення вузлів, інша параметризація IP-ядер у межах тієї самої специфікації).

Ключова ідея: декорелювати фізичні вразливості двох незалежних каналів. Навіть якщо одна і та сама зовнішня подія вплине на перший канал через певну

комбінацію розміщення/маршруту, висока ймовірність, що в другому каналі (зі зміненою картою і шляхами) ця комбінація не реалізується.

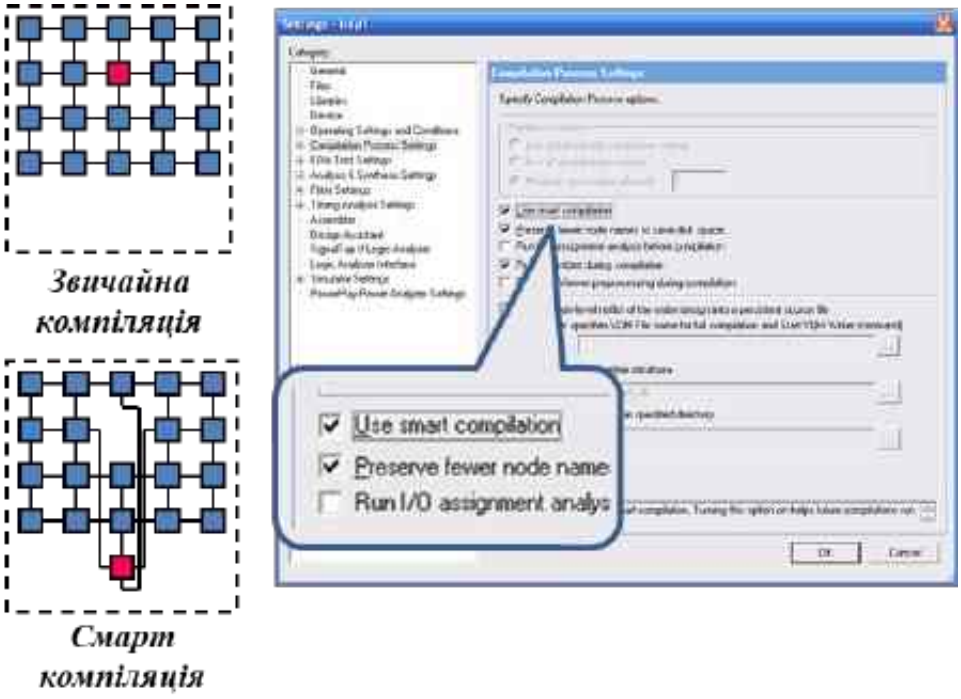

Рисунок 3.8 - Налаштування процесу компіляції (Use smart compilation та споріднені опції) [65]

На рисунку праворуч зображено діалог налаштувань збірки з прапорцями Use smart compilation, Preserve fewer node name, Run I/O assignment analysis. Ліворуч — умовні «карти щільності» (масиви блакитних квадратиків), що символізують різні варіанти розміщення логіки, яких можна досягти завдяки зміні цих опцій. Вимкнення Use smart compilation прибирає «інкрементальну пам’ять» інструмента і провокує інший цикл аналізу/синтезу/фітингу, унаслідок чого отримуємо іншу карту розміщення і трасування та інший набір критичних шляхів. Додаткові опції впливають на стабільність імен та коректність прив’язок вводу/виводу, що важливо для порівнянності альтернатив.

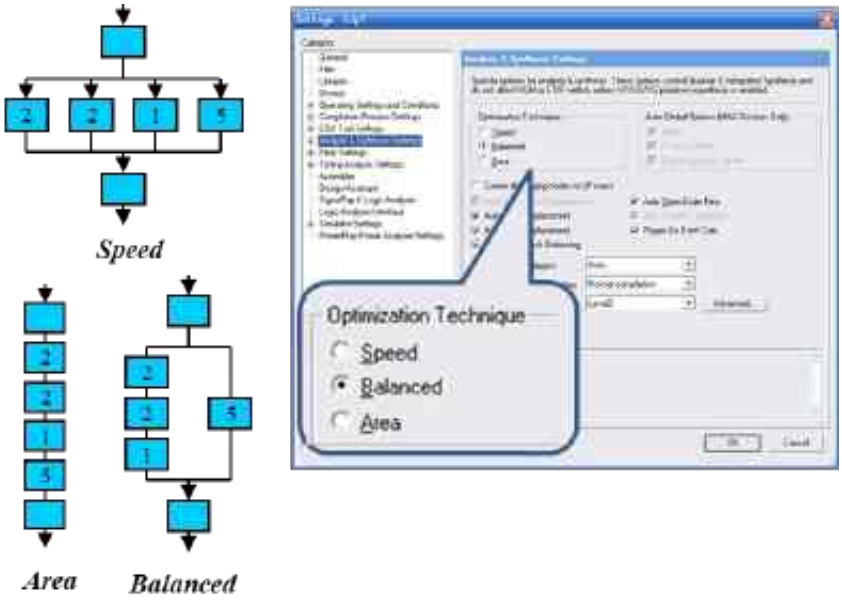

Рисунок 3.9 - Analysis & Synthesis Settings → Optimization Technique (Speed / Balanced / Area)

Рисунок 3.9 відображає перемикач Optimization Technique з трьома профілями:

- *Speed* агресивно скорочує затримки, зазвичай формує новий набір критичних маршрутів і може збільшувати довжину трас;
- *Area* ущільнює логіку, зміщуючи «гарячі зони» в інші регіони кристалу;
- *Balanced* дає компромісний, третій за характером профіль. Для методу диверсності це означає: пари «Speed vs Area» або «Speed vs Balanced» практично гарантовано мають непересічні набори критичних шляхів і відмінну геометрію образу розведення.

Показники правильності виконання методики:

- Якщо після вимкнення *Use smart compilation* (Рисунок 3.8) карти щільності та критичні шляхи помітно змінилися, а таймінг-цілі виконані — це коректна диверсна альтернатива.
- Якщо при перемиканні Optimization Technique (Рисунок 3.9) фіксується інша картина критичних шляхів і «гарячі зони» не збігаються — сформовано якісну пару профілів.

– Якщо ж просторово-часові відмінності мінімальні — це свідчить про недостатню диверсність; варто змінити саме профіль оптимізації або відмовитися від інкрементальності (Рисунок 3.9).

Метод структурно-просторової диверсності забезпечує керовану декореляцію між незалежними каналами за рахунок альтернативних (у просторі та часі) фізичних реалізацій одного функціоналу. Зміна Use smart compilation і вибір Optimization Technique (*Speed / Balanced / Area*), за потреби — легкі структурні варіанти, дають різні карти і різні критичні шляхи при збереженні специфікації. Експертне співвідношення 0,45 / 0,44 (вартість/ефект) підтверджує доцільність методу для модулів безпеки.

3.2.2 Принципи та послідовність реалізації

Мета реалізації — отримати дві незалежні збірки одного функціоналу з істотно різними просторово-часовими профілями (карти розміщення, маршрути, критичні шляхи, запаси таймінгу), не змінюючи специфікацію інтерфейсів і поведінку модуля. Усі зміни виконуються на рівні налаштувань компілятора та за потреби — легких структурних правок усередині проєкту.

Основою просторового рознесення є принцип функціональної тотожності за фізичної відмінності. Логічна модель, зовнішні інтерфейси та набір тестових векторів залишаються однаковими для обох (усіх) каналів, натомість змінюються саме інструментальні рішення: режими оптимізації синтезу, варіанти розміщення й трасування, м’які обмеження розміщення, неінвазивні структурні варіанти на кшталт інакшої буферизації чи альтернативного поділу вузлів логіки. Такий підхід приводить до різних карт розміщення, інших критичних шляхів та іншим чутливостям до локальних завад, що й створює шукану декореляцію без зміни поведінки на рівні специфікацій і тестів.

Другий принцип стосується тактування: просторове рознесення не повинно множити тактові домени. Додавання зайвих доменів автоматично збільшує кількість переходів між ними, ускладнює верифікацію й статичний аналіз

таймінгу, а також створює нові точки спільної вразливості. Тому внутрішнє фазування й мікророзводку подій слід виконувати через сигнали дозволу такту (clock enable), керовані затримки та буферизацію, залишаючи єдиний базовий домен для кожного каналу. Це зберігає простоту формальних перевірок, мінімізує CDC на технологічних межах і водночас дозволяє розвести у часі «вікна активності» всередині одного домену.

Третій принцип — керована вартість. Диверсність повинна вкладатися в бюджет ресурсів (логічні елементи, тригери, пам’ять, множники), енергоспоживання та прийнятний час компіляції. Кожен обраний профіль необхідно супроводжувати коротким кошторисом «вартість ↔ ефект»: скільки додаткових ресурсів додано, як змінився час збірки, і що отримано у результаті. Якщо зростання вартості не виправдане виграшем метрик, налаштування слід переглянути (послабити директиви, змінити режим «Speed/Balanced/Area», тощо). Практично це означає автоматизований цикл звітів після кожної збірки з витягом зведень таймінгу, ресурсів, енергетики та CDC — аби рішення про прийнятність конкретної диверсної конфігурації було відтворюваним і прозорим.

Описана процедура дозволяє системно й відтворювано отримувати диверсні збірки: ми тримаємо функціональну ідентичність і керовано створюємо просторово-часову різницю між каналами за рахунок інструментальних параметрів і легких структурних варіантів. У результаті знижується ризик відмови за спільною причиною за прийнятної «вартості», а отримані профілі готові до серійної інтеграції та сертифікаційної перевірки.

3.3 Диверсність програмно-апаратних засобів і процедур підрахунку контрольної суми

Системи керування технологічними процесами вимагають жорсткого контролю цілісності даних. Щоб зменшити ризик відмови за спільною причиною під час перевірки ПЗ, застосовано диверсифікацію процедур обчислення полінома контрольної суми: одна процедура виконується на етапі компіляції (Tcl-скриптом

у Quartus II), друга — під час роботи (апаратно-програмним шляхом на soft-процесорі Nios II). Через принципово різні мови/середовища/механізми і різні алгоритмічні реалізації, ймовірність однаково-хибного підрахунку різко зменшується. Метод застосовується в модулях ПАТ НВП «Радій» на всіх АЕС України.

3.3.1 Сутність та метрики

У системах керування технологічними процесами втрата або спотворення коду/даних неприпустимі. Тому перевірка цілісності програмного забезпечення повинна бути не лише безпомилковою, а й стійкою до відмов за спільною причиною. Запропонований підхід досягає цього за рахунок диверсифікації процедур підрахунку полінома контрольної суми (CRC) [91–95]: одна процедура працює під час компіляції проєкту, інша — у виконанні, причому вони реалізовані на принципово різних платформах і мовах. Унаслідок цього одна і та сама помилка або зовнішній вплив із малою ймовірністю спричинить одночасно однаковий хибний результат у двох незалежних ланцюгах.

Побудова та робота даного методу відбувається у два етапи:

1. Компіляційна гілка. В IDE Quartus II після розроблення образу ПЗ (.bin/.hex/.srec) під час компіляції автоматично запускається Tcl-скрипт, який табличним методом обчислює CRC ПЗП масиву пам’яті програмного коду. Отримане значення записується як еталонна константа Quartus II та з’єднана з вхідним портом Nios II. Ця гілка працює поза контекстом виконання основної програми, не використовує її інфраструктуру і тим самим формує незалежний від основного програмного коду спосіб перевірки.

2. Робоча гілка. У запрограмованому, прошитому та працюючому soft-процесорі Nios II, з заданою періодичністю, програма ініціює апаратно-програмний підрахунок CRC масиву програмного коду своєї ж програми. Отримане значення порівнюється з еталонною константою. Будь-яка розбіжність породжує подію безпеки (індикація, фіксація помилки, перехід у безпечний

режим, тощо). Завдяки різним середовищам, мовам програмування, наборам бібліотек і часу виконань, ймовірність спільної помилки для двох шляхів різко зменшується. Метод експлуатується в модулях ПАТ НВП «Радій» на всіх АЕС України.

Рисунок 3.10 - Фрагмент Tcl-скрипта табличного підрахунку CRC

Рисунок 3.10 відображає уривок Tcl-скрипта підрахунку контрольної суми масива даних. Код послідовно (по байтам або словам) зчитує масив даних та обраховує контрольну суму за допомогою табличного методу, відповідно до вибраного полінома. Скрипт виконується під час компіляції. Результат підрахунку автоматично оновлює константу контрольної суми Nios II, яка після кінця компіляції буде міститись в образі ПЛІС проєкту. Це повністю ізольований від Nios II та його бібліотек ланцюг.

При необхідності, можна змінити ширину шини даних (8,16, 32 біта), поліном, порядок байтів, діапазон адрес масива пам'яті та інше. Скрипт забезпечує детермінованість і відтворюваність результату для будь-яких подальших збірок.

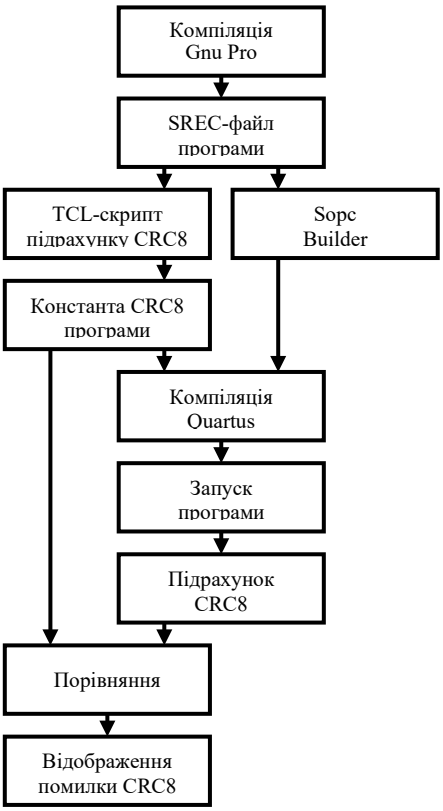

Рисунок 3.11 - Блок-схема диверсної процедури перевірки

Рисунок 3.11 надає алгоритм підрахунку контрольної суми модуля БФЗ-7 з відповідними налаштуваннями цього проєкту, що реалізує програмно-апаратну диверсифікацію перевірки цілісності ПЗ для всіх soft-процесорів модуля. Робочий цикл починається зі звичайної компіляції прикладного коду інструментарієм GnuPro. На виході формується .SREC-файл, у якому послідовно описані всі байти програмного образу з адресами розміщення у ПЗП. Одразу після побудови артефакту автоматично прекомпілятором Quartus II запускається Tcl-скрипт (pre-build hook у середовищі проєктування), який читає .SREC файл, нормалізує представлення байтів (фіксує порядок та вирівнювання) та табличним методом для обраного полінома CRC8, проходить увесь образ і обчислює еталонне значення. Отримане число фіксується як константа CRC8 (Рисунок 3.12) програми у вихідних файлах проєкту, які в подальшому лінкуються у виділений порт відповідного soft-процесора.

Паралельно з цим, у складі системи-на-кристалі (SOPC Builder / Platform Designer) у процесорний контур інтегровано апаратну інструкцію «CRC8» (користувацька процесорна апаратна команда) — невеликий апаратний модуль, який приймає байт (слово) даних, виконує крок поліноміального зсуву і акумулює поточну суму у регістрі результату. Після завершення збірки усі IP-ядра й прошивка компілюються у Quartus, формується бітстрім ПЛІС із вмонтованою інструкцією, а програмний образ — із записаною еталонною константою. На етапі виконання, під час ініціалізації, прикладний код soft-процесора з відомого базового адресного діапазону по байтах послідовно читає ПЗП програми і використовуючи апаратну інструкцію CRC8, швидко обчислює «операційну» КС у реальному часі. Вибір саме апаратної (а не табличної програмної) реалізації на борту дає часово детермінований і стійкий до збоїв результат, а також забезпечує різнорідність засобів: офлайнова КС отримана засобами Tcl/табличної логіки, онлайнова — апаратним поліномом; імовірність однакової хибної відповіді від двох різних реалізацій знижується.

Після завершення проходу пам’яті обчислене значення порівнюється з еталонною константою. У разі збігу система продовжує завантаження та з певною перічодичністю посітійно обраховує цілісність пам’яті програмного коду. Якщо виявлено розходження — генерується подія «Помилка CRC8», що водночас реєструється у діагностичному блоці, відображається на інформаційному табло модуля та шафи, блокується виконання небезпечних функцій та помилка передається до діагностичного серверу для реєстрації події. Така послідовність дозволяє виявляти як випадкові спотворення програмного коду, так і несанкціоновані зміни, оскільки злам має відтворити одночасно і офлайновий, і онлайновий механізми різної природи.

Важливими для практичної безпомилковості є кілька нюансів: .SREC обробляється у канонічному форматі з жорстким контролем діапазонів адрес і пропуском заповнювачів, а область розміщення константи CRC винесена за межі периметра підрахунку. На борту читаються саме ті ж адресні межі, що й у Tcl-скрипті, а порядок байтів узгоджений із форматом .SREC, що унеможливлює

помилки через ендіанність. Нарешті, апаратна інструкція CRC8 викликається фіксованим циклом із відомою тривалістю, що полегшує аналіз таймінгу й спрощує доведення відсутності побічних ефектів для задач реального часу.

У цілому ланцюжок «офлайновий підрахунок → вбудована константа → онлайновий підрахунок → порівняння → реакція» реалізує потрібний рівень диверсності й простежуваності: одна й та сама істина (цілісний образ ПЗ) перевіряється двома різними засобами в різних середовищах виконання, а результат однозначно інтерпретується логікою безпеки модуля.

Рисунок 3.12 - Приклад підключення апаратного модуля CRC («NiosCrc8»)

Правильність та стабільність роботи методу вимагає щоб результати двох гілок були порівнюваними, для них фіксуються однакові: поліном, початкове значення CRC, правила віддзеркалення бітів на вході/виході та фінальний XOR. Крім того, однозначно визначають порядок байтів і адресний діапазон ПЗП, що входить у перевірку. Усі ці параметри заповнюються в технічній документації модуля та використовуються як інваріанти при серійних збірках.

Метики диверсності (якості) методу можна описати наступними тезами:

1. Незалежність плечей. Різні середовища (Tcl під час збірки - апаратний/програмний блок у виконанні), різний стек інструментів і різний час виконання мінімізують імовірність однаково-хибного підрахунку.

2. Покриття збоїв. Здатність виявляти: локальні зміни в ПЗП (1–2 байти), пошкодження блоків, помилки оновлення, «віддзеркалення» байтів, випадкові збої зберігання.

3. Час до виявлення. Максимальний інтервал між моментом появи помилки та її фіксацією визначається періодом перевірки і розміром блоку; цей показник задається конструктивно й легко контролюється.

4. Стійкість до спільної причини. Оцінюється експериментально: ін’єкції помилок у образ/канал зв’язку не повинні призводити до ідентичних хиб у двох гілках.

Вплив методу на вартість (ресурси/процес):

1. Ресурси ПЛІС. Логічні елементи та тригери для апаратного CRC-модуля. Можливе збільшення довжини маршруту даних, або необхідність застосування ПЛІС зі збільшеною кількістю ресурсів.

2. Навантаження процесора. Якщо частина обробки програмна — додаткові цикли на читання ПЗП і подачу байтів у модуль.

3. Енергоспоживання. Періодичний доступ до ПЗП і комутація логіки CRC збільшують динамічну складову. Період і пакетування дозволяють тримати її в певному «коридорі» обмежень бюджету.

4. Час компіляції/складність процесу. Автогенерація константи на етапі збірки додає крок до конвеєра, але забезпечує відтворюваність і позбавляє ручних операцій.

5. Складність верифікації. Потрібні ін’єкційні тести (зміна байтів у копії образу), тести на помилкові спрацювання й навантажувальні вимірювання для підтвердження періоду/бюджету часу.

Запропонована організація контрольної суми — це двоканальний диверсно-незалежний контроль: еталон формується на етапі компіляції (Tcl скрипт), перевірка відбувається у виконанні (апаратна інструкція/модуль Nios II). Параметри (поліном, ініціалізація, порядок байтів, діапазон адрес) фіксуються як інваріанти; вимірювані метрики — незалежність плечей, покриття збоїв, час до виявлення, ресурсна/енергетична вартість та складність верифікації. Така побудова зменшує ризик відмови за спільною причиною під час підрахунку CRC і забезпечує детермінований контроль цілісності ПЗ для серійної експлуатації в критичних системах.

3.3.2 Принципи та послідовність реалізації

Мета — отримати дві технологічно незалежні процедури підрахунку контрольної суми (компіляційна та робоча), які перевіряють той самий образ ПЗ у строго зафіксованих межах і за однаковими параметрами CRC, але виконуються в різних середовищах і в різний час.

Принципи реалізації методу диверсного підрахунку цілісності масива пам’яті програмного коду:

1. Функціональна тотожність, технологічна різність. Поліном, ініціалізація, порядок байтів, віддзеркалення бітів та фінальний XOR — однакові у двох гілках; мови, середовище виконання й механізм обчислень — різні (Tcl під час компіляції та апаратно-програмний під час виконання).

2. Детермінований «час до виявлення». Період перевірки та розмір блоку задаються так, щоб гарантувати верхню межу між появою помилки й її фіксацією без порушення дедлайнів реального часу.

3. Ізоляція та відтворюваність. Компіляційна гілка не має залежати від ПЗ пристрою; робоча — від елементів середовища збірки. Усі параметри фіксуються в технічній документації модуля.

4. Безпечна реакція. Будь-яка розбіжність — це подія безпеки з завчасно визначеною дією (логування, індикація, перехід у безпечний стан, блокування функцій).

5. Сумісність із іншими методами. Планування періодики перевірки узгоджується з іншими методами диверсифікації проєкту й не погіршує таймінг у структурно-просторових профілях.

Послідовність реалізації:

1. Фіксація інваріантів CRC («легенда»). Визначити та записати в технічну документацію проєкту*:* поліном*,* початкове значення*,* віддзеркалення вхід/вихід*,* фінальний XOR*,* порядок байтів*,* адресний діапазон ПЗП*,* розмір блока перевірки*,* період перевірки. Це — основа для відтворюваності.

2. Підготовка образу ПЗ. Провести компіляцію проєкту з отриманням вихідного .bin/.hex/.srec файлу, що є чітким образом пам’яті програмного коду. В середовищі проєкту Quartus II сформувати налаштування та під’єднання необхідних компонентів:

- Інтеграція компіляційної гілки Tcl (Рисунок 3.10): Додати виклик скрипта в потік Quartus II (через quartus_sh -t … або «Execute script after compilation»). Скрипт читає образ, табличним методом обчислює CRC, генерує еталонну константу для проєкту Quartus II.
- Впровадження апаратного модуля CRC (Рисунок 3.18): додати в проєкт інстанс «NiosCrc8» (або відповідний для CRC-16/32) з пам’ятко-відображеними регістрами: керування/скидання, дані, результат/статус, або підключити до системної шини/портів; за потреби додати FIFO або DMA для пакетної подачі даних.
- Ініціалізація програмної частини Nios II: під час стартової ініціалізації програми, зчитати адреси пам'яті програмного коду, зробити скидання модулю CRC, виконати повну самодіагностику пам'яті.
- Періодична перевірка: з певною періодичністю (самодіагностика модуля не повинна впливати на таймінги основних і критичних функцій модуля) побайтово (по словам) за один цикл роботи, читати пам'ять програмного коду модуля, послідовно подавати їх у модуль CRC, зчитувати результат та зберігати у змінну. Під час наступного циклу роботи програми, призводиться контроль наступного байту (слова) програмного коду і так відбувається поки не буде досягнуто кінця цієї пам'яті. Коли це відбувається, порівнюється вирахувана КС з еталонною та відбувається переініціалізація змінних для нового перерахунку цілісності програмноої пам'яті. Таким чином, контроль пам'яті здійснюється без затримок основних функцій. У разі розбіжності: провести індикацію помилки на інтерфейсній частині модуля та шафи, передати оператору діагностичний пакет з помилкою з послідуючою заміною модуля на справний.

– Верифікація: у копії образу навмисно змінити 1–2 байти на початку/всередині/в кінці діапазону, провести повний цикл компіляції та прошивання модулю, перевірити очікуване надходження помилки цілісності пам'яті ПЗП.

3.4 Диверсність методів передачі даних та перевірки цілісності пам’яті

У цьому підрозділі розширюється застосування програмно-апаратної диверсності в контексті формування й передавання пакетів. Базовий принцип незмінний: однаковий змістовний критерій коректності (той самий поліном CRC/правило та той самий набір байтів), але різні способи обчислення/перевірки на кожному з плечей і в різних інженерних шарах, щоб мінімізувати ризики відмови за спільною причиною. На боці передавача та приймача цілісність одного й того самого кадру контролюється різними реалізаціями (наприклад, апаратний обчислювач у передавача та програмно-табличний у приймача), а на боці пам’яті поєднуються апаратні методи контролю цілісності даних з фоновим підрахунком за допомогою DMA контроллера.

Метод доповнюється диверсністю протоколів і профілів кадрування (паралельні або послідовні канали передачі даних, різні методи контролю CRC). Очікуваний ефект — декореляція помилок у каналі передачі та масивах пам’яті, та детермінований верхній поріг «часу до виявлення».

3.4.1 Сутність та метрики

На рівні транспортного середовища диверсність охоплює вибір різних інтерфейсів і шин передачі (послідовні/паралельні, синхронні/асинхронні) та альтернативних профілів кадрування і таймінгу. Один і той самий змістовний кадр може передаватися двома незалежними каналами (Рисунок 3.13), що відрізняються: фізичним середовищем і способом формування такту; правилами кадрування (місце й довжина контрольної суми, порядок байтів/бітів); механікою

обміну (квитування/повтори, глибини буферів, політики черг). Таке рознесення декорелює транспортні помилки та знижує імовірність відмови за спільною причиною — ще до рівня процедур контролю цілісності, які розглядаються далі.

Диверсність контролю цілісності передачі даних полягає у тому, що один і той самий критерій коректності (той самий поліном CRC і той самий набір байтів) перевіряється різними реалізаціями на незалежних плечах тракту: у передавача й приймача — різними способами підрахунку контрольної суми; у підсистемі пам’яті — комбінацією апаратних кодів, із фоновим підрахунком, або рознесенням у часі [47; 48; 65; 96]. Така організація зменшує кореляцію відмов за спільною причиною: одна помилка коду/бібліотеки/маршруту з меншою імовірністю проявиться однаково на обох кінцях каналу або одночасно у двох способах перевірки пам’яті.

Рисунок 3.13 - Узагальнена схема двох незалежних методів передачі масива даних [65]

Рисунок 3.13 показує, як IP-Core 1 і IP-Core 2 одночасно під’єднані до двох різних інтерфейсів обміну. Для прикладу, верхній інтерфейс відповідає паралельному синхронному тракту (широка шина, мінімальна латентність, висока пропускна здатність; чутливість до синфазних завад та дестабілізації такту), нижній — послідовному асинхронному (UART/SPI/RS-485/CAN/Ethernet тощо; власне кадрування, краща переносимість по довгих трасах; більша процедурна латентність). Ключове тут не конкретний протокол, а гетерогенність: різні фізичні рівні, стеки, політики квитування/повторів, окремі тактові домени з буферизацією у FIFO. Такий підбір робить моделі відмов двох шляхів некорельованими:

залипання через переповнення буфера або локальна EMI-спайка на одній гілці з великою імовірністю не відтворюються на другій [97–99].

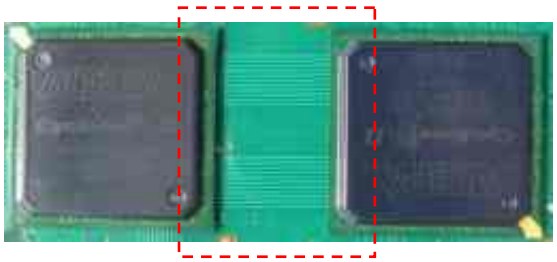

Рисунок 3.14 - Просторове рознесення каналів (фото плати з двома ПЛІС)

На знімку (Рисунок 3.14) видно два незалежні кристали ПЛІС на одній платі; між ними (виділено пунктиром) проходить внутрішня шина яка їх з'єднує. Просторове рознесення зменшує ризики локальних одночасних впливів: мікротріщин/відривів паяних з’єднань під одним корпусом, локального перегріву, одиничних подій та механічних навантажень. У практичних конструкціях кожне FPGA-плече має власні силові домени/стабілізатори, окрему мережу такту та незалежні ланцюги тестування/діагностики; сам між-FPGA зв'язок реалізують іншою фізикою, ніж зовнішній канал (наприклад, паралель LVDS усередині модуля + серійний протокол на вихід), що додає ще один шар диверсності. Така комбінація — різні інтерфейси передачі та фізичне рознесення логіки — критична для модулів безпеки на АЕС: навіть якщо одна гілка деградує або тимчасово блокується (EMI-імпульс, локальна відмова живлення, помилка у стеку), друге плече з іншою шиною та іншим тактом з високою імовірністю залишається працездатним і забезпечує доставку контрольованого кадру з перевіркою цілісності.

Вимоги до метрик дублювання каналів передачі даних:

1. Виявлюваність помилок у каналі передачі. Частка класів помилок, що детектуються парою реалізацій: одиночні біти/байти, короткі «бурсти», зсуви кадру, спотворення поля довжини/заголовку. Практично — підтверджується ін’єкційними тестами на різних фазових умовах.

2. Незалежність плечей. Ступінь відмінності реалізацій: апаратна мережа; різні бібліотеки; різні процесорні й ПЛІС-архітектури; просторове рознесення. Чим більша різниця — тим менша імовірність однакової хибної перевірки.

3. Час до виявлення помилки. Для каналу — латентність між прийомом кадру та рішенням; для пам’яті — верхня межа, задана періодом читання і розміром блока. Цей показник визначається конструктивно параметрами планувальника.

4. Покриття пам’яті. Частка адресного простору, що регулярно проходить контроль: ECC-корекції (кількість/частота), невиправні помилки, виявлення деградацій (зношення).

5. Накладні витрати протоколу. Довжина CRC/службових полів, падіння корисної пропускної здатності, збільшення латентності.

6. Ресурси ПЛІС і ЦП. Логіка/тригери/ПЗП для апаратного обчислювача, цикли CPU для табличної реалізації, буфери/черги кадрів та інше.

7. Енергоспоживання. Активність апаратного обчислювача, тактова частота прийомо-передавача, частота пам'яті (може узгоджуватись з диверсною синхронізацією, щоб уникати синфазних піків).

8. Складність верифікації. Обсяг ін’єкційних випробувань, міжканальне журналювання, перевірка межових режимів.

3.4.2 Принципи та послідовність реалізації

Диверсність каналу передачі даних безпосередньо залежить від вибору шин (інтерфейсів) і профілю їх використання. Різні типи фізичного та канального рівнів (паралельні чи послідовні лінії, способи кадрування, правила арбітрації, місце/параметри CRC або ECC) [100; 101] формують відмінні механізми відмов і завад, інші часові характеристики та інший електромагнітний «почерк». В результаті один і той самий змістовний контроль даних, реалізований поверх

різних інтерфейсів, має меншу кореляцію помилок і краще протистоїть спільним причинам відмов.

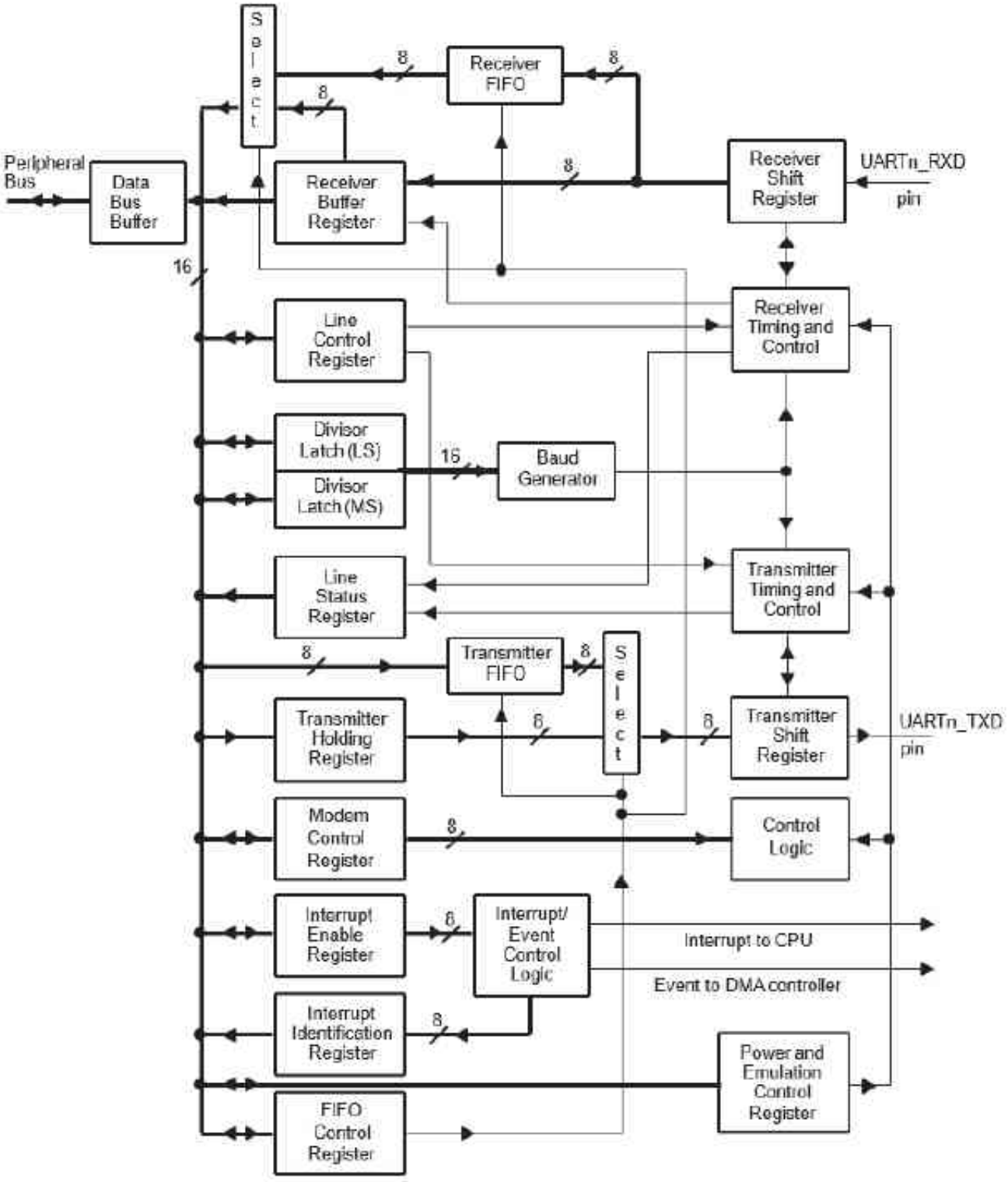

Рисунок 3.15 - Схема синхронної послідовної шини UART [102]

Рисунок 3.15 відображає схему апаратної реалізації асинхронного, повнодуплексного інтерфейсу “точка-точка” з двома основними лініями: TXD (передача) і RXD (прийом). Усередині контролера є генератор тактів (baud-rate generator), регістри керування/статусу, TX/RX FIFO та зсувні регістри [102–109; 109–111]. Коли CPU/DMA записує байт у TX FIFO, апаратно формується кадр: лінія з неактивного стану ‘1’ опускається в ‘0’ (start-bit), далі 5–9 біт даних (LSB першим), опційно біт парності, потім 1–2 стоп-біти ‘1’. Швидкість визначається дільником тактової частоти (baud rate), рівні сигналів забезпечує окремий трансивер (TTL/CMOS ↔ RS-232/RS-485/TTL-UART).

Тракт приймання UART виконує надлишкове дискретизування RXD (типово ×8/×16) для точного виявлення старт-біта, відбирає зразки по середині кожного біта, відновлює байт і поміщає його в RX FIFO. Перевіряються стоп-біт(и) і, за потреби, парність; помилки фіксуються прапорцями (framing, parity, overrun). Обмін керується перериваннями або DMA [112; 113]. Для апаратного керування потоком можуть використовуватись RTS/CTS (або XON/XOFF на рівні протоколу). Оскільки інтерфейс асинхронний (без окремої лінії такту), обидві сторони повинні узгоджено мати однакові параметри кадру: швидкість, кількість біт даних, парність і стоп-біти.

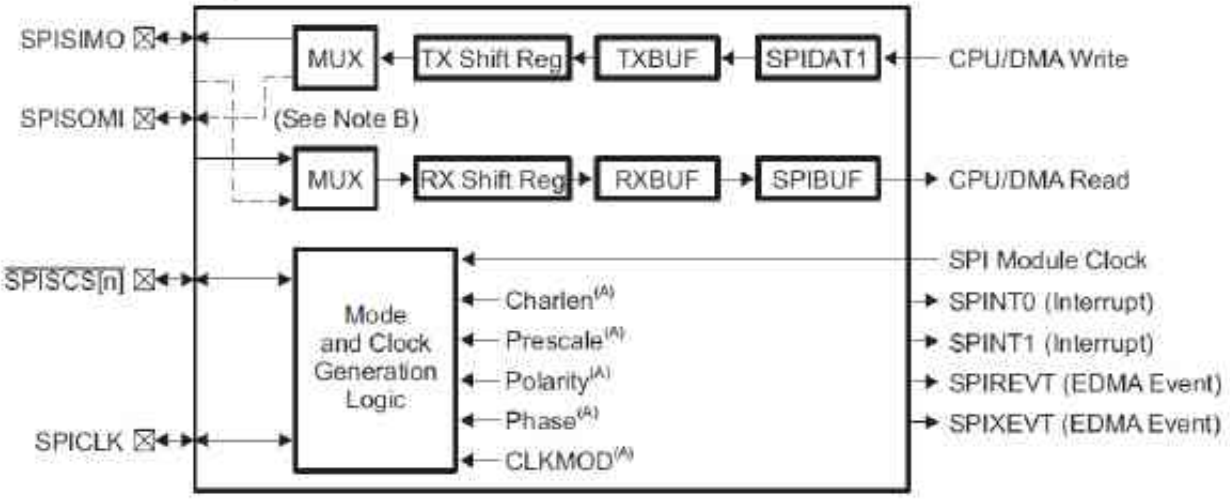

Рисунок 3.16 – Схема асинхронного послідовного інтерфейсу SPI [114]

Рисунок 3.16 відображає синхронний послідовний інтерфейс “майстер-ведений(і)” з лініями SCLK (такт), MOSI, MISO та CS/SS. Майстер формує тактову частоту й відкриває кадр притисканням CS; на кожному фронті/спаді SCLK обидві сторони зсувають по біту через зсувні регістри: майстер передає по MOSI, одночасно приймаючи по MISO (повний дуплекс). Порядок бітів (MSB/LSB first), полярність/фаза такту (CPOL/CPHA, “режими” 0–3) та розрядність слова (типово 8, але 4–16+ біт) узгоджуються наперед [114–116].

Всередині контролера — дільник такту, регістри керування/статусу, TX/RX зсувні регістри та (часто) FIFO; обмін керується перериваннями або DMA. CPU/DMA пише слова у TX FIFO — контролер генерує SCLK і “виштовхує” біти, паралельно наповнюючи RX FIFO даними з іншого боку; якщо треба лише читати, майстер надсилає “пусті” байти для генерації тактів. На шині з кількома веденими кожен має свій CS. Адресації чи керування потоком на рівні SPI немає — це робить протокол вищого рівня. Швидкість обмежується специфікаціями пристрою та цілісністю сигналів/трасуванням.

Рисунок 3.17 відображає апаратний шар між MAC і середовищем (вита пара або оптика). Він приймає бітовий потік/слова від MAC через інтерфейси MII/GMII/RGMII/SGMII/USXGMII, виконує кодування/скремблювання, серіалізацію, синхронізацію та фізичну модуляцію сигналу, а також відновлення такту на прийомі. Через автоузгодження (Auto-Negotiation) PHY визначає швидкість/дуплекс, EEE/PAUSE, роль master/slave і, за потреби, проводить тренування лінії. Керування відбувається через MDIO/MDC (реєстри статусу/помилок/параметрів), вихід на кабель здійснюється через аналоговий фронт-енд та трансформатори (magnetics) з диференційними парами й авто-MDI/MDIX [117–121].

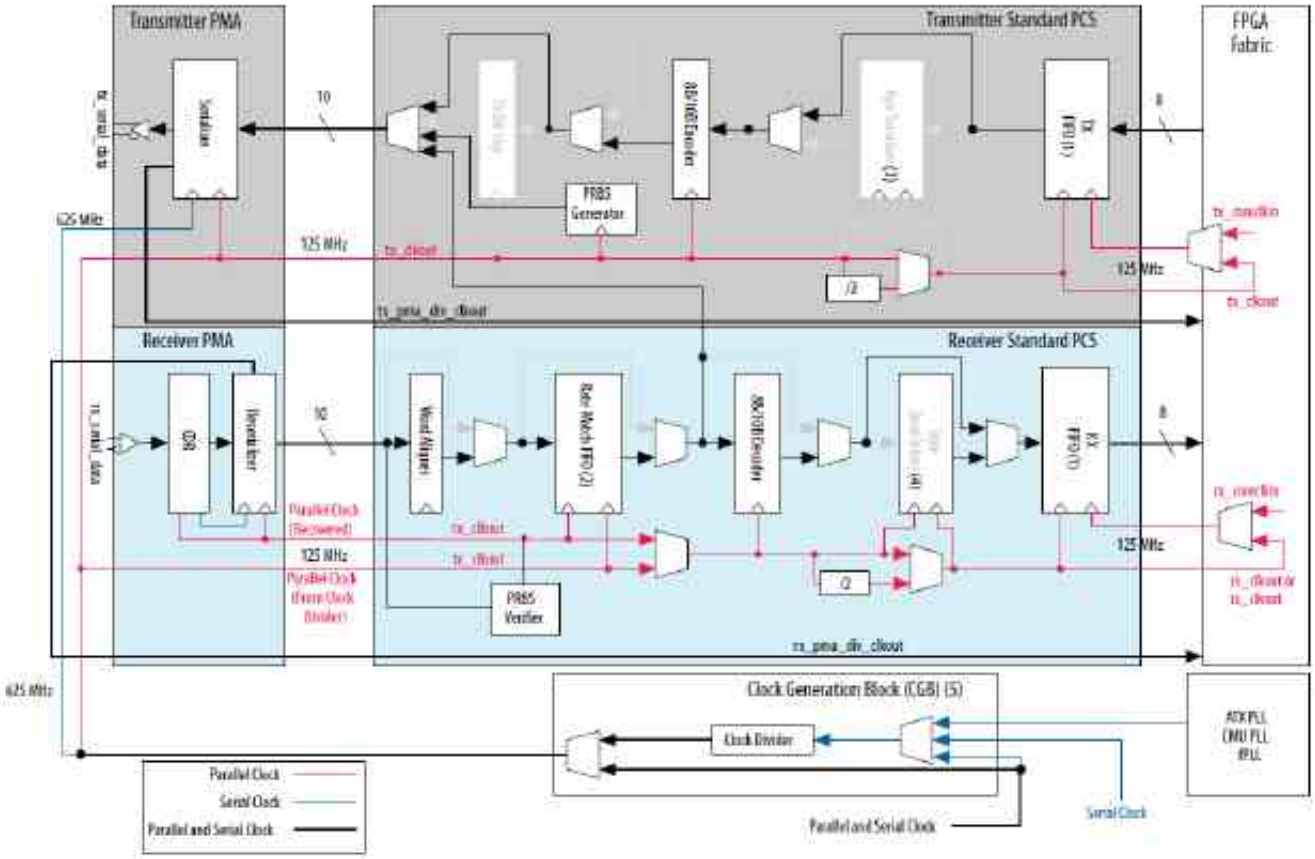

Рисунок 3.17 - Схема апаратної реалізації Phy Ethernet [117]

На передачі MAC формує кадри з преамбулою/SFD і CRC, далі PHY перетворює їх у символи фізичного рівня: для 100BASE-TX — 4B5B + MLT-3; для 1000BASE-T — PAM-5 по чотирьох парах; для 2.5/5/10GBASE-T — вищі порядки PAM із скремблером, лінійними драйверами, ехокорекцією/NEXT-cancellation та (для 10G) FEC (LDPC). На прийомі PHY виконує підсилення/АЦП, адаптивну еквалізацію, відновлення такту, дескремблювання та декодування, детектує помилки (symbol/framing), після чого віддає чисті біти/слова в MAC. Увесь процес синхронізовано тактово майстром лінка і залежить від якості траси: довжини/категорії кабелю, цілісності сигналу та перехідних завад.

Наведені приклади чітко демонструють принципову архітектурну різницю між шинами передачі даних, а отже — й їхню природну диверсність. UART реалізує асинхронний «точка-точка» обмін із локальним відтворенням такту на приймачі; SPI — синхронний майстер-ведений інтерфейс із явною лінією такту й

вибором підлеглого. Ethernet PHY — повноцінний фізичний рівень пакетної мережі з кодуванням, серіалізацією та відновленням такту. Відмінні принципи тактування, кадрування, способи формування/контролю бітового потоку та навіть фізичні середовища означають різні моделі збоїв і різні «вікна уразливості», що критично важливо для зменшення ризику відмов за спільною причиною. Іншими словами, коли паралельні канали покладаються на різні за природою інтерфейси, вони рідше «помиляються однаково» в тих самих умовах завад.

Як інтерфейси з'єднання UART, SPI та Ethernet розрізняються вже на рівні топології ліній і робочих діапазонів [122; 123]. UART мінімалістичний за лініями (TXD/RXD, опційно RTS/CTS) і типовими швидкостями від кілобітів до одиниць мегабіт за секунду; практична дальність залежить від рівнів сигналів і фізичного носія (TTL на платі — сантиметри/десятки сантиметрів; з RS-232/RS-485 — десятки метрів і більше за належного трасування). SPI вимагає щонайменше чотири лінії (SCLK, MOSI, MISO, CS), може працювати на десятках мегагерців, але по суті «коротка» шина — розрахована на зв'язок «на платі/в межах модуля». Ethernet через PHY використовує диференційні пари й трансформаторні узгоджувачі (magnetics), забезпечуючи стандартизовані швидкості 10/100/1000 Мбіт/с і вище, з типовою дальністю до 100 м на мідному кабелі (категорія залежить від швидкості) і значно більшою на оптиці. Уже ці параметри задають різний профіль чутливості до наведень, затухання, перехідних процесів і топологічних обмежень.

Відмінними є й механізми детермінізму та контролю помилок. UART опирається на надлишкове дискретизування старт-біта та параметричну узгодженість швидкостей; контроль — мінімальний (парність, перевірка стоп-бітів), що зручно, але вимагає зовнішніх захистів від завад. SPI взагалі не містить вбудованого контролю цілісності кадру — усе покладається на такт майстра і дисципліну протоколу вищого рівня; натомість він максимально простий і передбачуваний у часовому сенсі. Ethernet має розвинену систему контролю: від фізичного відновлення такту та еквалізації до CRC у кадрі MAC і (на високих швидкостях) FEC на фізичному рівні. Ця різношерстість означає, що для

диверсифікації каналів доцільно поєднувати інтерфейси з різними шляхами виявлення/маскування помилок та різною чутливістю до завад.

Практичний висновок для архітектури безпеки: якщо критичні дані доставляються двома незалежними трактами на основі різних інтерфейсів (наприклад, UART ↔ SPI або UART/SPI ↔ Ethernet), то зростає технологічна незалежність каналів і зменшується кореляція відмов. Навіть за однакових поліномів контрольної суми на рівні змісту пакета, розрізнені фізичні/канальні шари породжують різні часові профілі подій, різні джерела шуму та різні сценарії деградації (включно з особливостями реалізації FIFO, відновлення такту, фазових співвідношень тощо), що підсилює загальний ефект диверсності.

Поряд із часовим рознесенням подій важливу роль відіграє диверсність у структурі формування пакетів даних: профілі кадрування (розміщення службових полів і контрольної суми на початку або в кінці кадру), порядок байтів, способи сегментації/агрегації та інтерлівіювання полів, вирівнювання по межах слів, а також розклад подачі даних у буфери різних каналів. За збереження однакової семантики поля кадру, різні профілі кадрування в еквівалентних каналах зменшують імовірність синхронного виникнення помилок у тих самих місцях структури пакета, додатково розводять «гарячі» ділянки навантаження та обмежують поширення локальних збурень.

Не менш важливою є диверсність у перевірці цілісності переданих даних: той самий критерій коректності (контрольна сума за фіксованими параметрами) реалізується різними технічними шляхами на кінцях тракту (наприклад, апаратний підрахунок у передавача та програмно-табличний — у приймача) і доповнюється незалежними профілями контролю в підсистемі пам’яті (ECC/паритет і фонове сканування). Така організація знижує ризики відмов за спільною причиною в транспорті та збереженні й задає керовані межі «часу до виявлення» помилок; подальші підрозділи деталізують принципи та метрики цього підходу.

Рисунок 3.18 - Приклад апаратної реалізації підрахунку (VHDL, «кастомна інструкція»)

Рисунок 3.18 відображає модуль, який підключається до Nios II як кастомна інструкція (порт nios_custom_instruction_slave). У прикладному коді процесора виклик виглядає як звернення до апаратної процесорної інструкції, що дає прискорене оновлення CRC-8 програмної пам’яті/кадрів даних.

На рисунку наведено модуль CI_CalcCrc8 з трьома портами:

OldData : in std_logic_vector(31 downto 0) — вхід A (позначка коментарем -- nios_custom_instruction_slave.dataa). У цьому проєкті сюди подається попередній стан CRC (8 біт у молодших розрядах слова).

NewData : in std_logic_vector(31 downto 0) — вхід B (.datab) із новою 32-бітною порцією даних для оновлення CRC.

Result : out std_logic_vector(31 downto 0) — вихід (.result), де формується оновлене значення CRC-8 у бітах Result(7 downto 0); старші біти Result(31 downto 8) обнуляються:

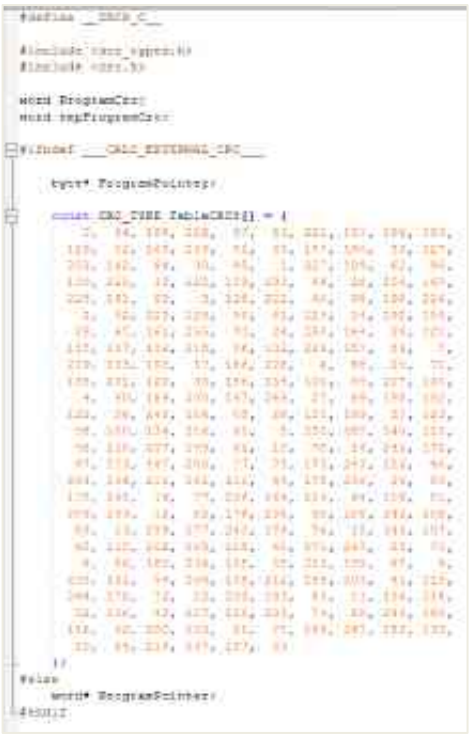

Рисунок 3.19 - Приклад програмно-табличного підрахунку CRC

Алгоритм роботи такий: на кожен виклик інструкції процесор подає в NewData черговий байт (або слово) та поточний CRC у OldData; за один такт комбінаційна логіка обчислює новий стан CRC і повертає його в Result. Таким чином досягається пропускна здатність 1 байт(слово)/такт із мінімальною джитерністю (відсутні таблиці та умовні переходи), а навантаження на ядро зменшується порівняно з чисто програмним підрахунком. Цей апаратний шлях далі використовується як диверсний до програмного табличного алгоритму (результати двох незалежних реалізацій порівнюються в рантаймі для виявлення можливих збоїв за спільною причиною).

Фрагмент програмного коду (Рисунок 3.19) на ілюстрації показує табличний метод обчислення CRC-8. У файлі оголошено статичну таблицю на 256 елементів TableCRC8[] — це попередньо обчислені залишки полінома для всіх можливих 8-бітних значень.

Алгоритм роботи (типовий для такого підходу): для кожного байта повідомлення індексується таблиця за ключем TableCRC8[crc ^ byte], де crc —

поточний залишок; результат підставляється як новий crc. Отже, кожен крок — одна операція XOR + одне звернення до таблиці, без зсувів і побітових циклів.

Переваги: детермінований час на довжину кадру, мінімальний розкид затримок, простота валідації; недолік: невеликий, але фіксований обсяг ПЗП під таблицю (256 байт або більше при розширених варіантах поліномів). У нашій схемі диверсності цей табличний CRC виступає програмною реалізацією, яка контрастує з апаратною (обидві використовують один і той самий поліном, але різні засоби та тракти виконання, що знижує ризик однаково-хибного підрахунку).

Послідовність реалізації:

1. Зафіксувати параметри кадру і контрольної суми. Визначити та записати: поліном, ініціалізацію, віддзеркалення вхід/вихід, фінальний XOR, порядок байтів; місце контрольної суми у кадрі, довжину службових полів.

2. Реалізувати обчислення на стороні передавача (апаратно). Вбудувати у проєкт апаратний обчислювач контрольної суми, що приймає послідовність байтів під час формування кадру.

3. Реалізувати перевірку на стороні приймача програмно-табличним методом. Після прийому кадру обчислити локальне значення та порівняти з полем кадру.

4. За наявності двох каналів передачі даних використати різні інтерфейси або різні профілі одного протоколу. Це додатково знижує кореляцію спотворень у транспорті.

5. Якщо не суперечить вимогам технічного завдання, спланувати рознесення у часі (узгодження з диверсною синхронізацією). Не допускати синхронних піків активності: великі пакети, перевірки кадрів і фонові перевірки пам’яті зміщувати у різні фази, щоб зменшити завади та імпульсні навантаження.

6. Якщо не суперечить можливостям реалізації, забезпечити просторове рознесення проєкту. Розмістити апаратний обчислювач і канали перевірки в окремих зонах/кристалах.

7. Визначити реакції на розбіжності. Для кадру — відкидання і повтор запиту; для пам’яті — повторне зчитування, ізоляція пошкодженого блока, сигналізація, перехід у безпечний стан.

8. Провести верифікацію ін'єкційним методом. При виявленні, виправити помилки, провести повторну верифікацію.

Запропонована послідовність реалізації диверсності в каналі «передача даних – контроль цілісності даних» (поєднання різнорідних інтерфейсів на фізичному/канальному рівнях, розведення профілів кадрування й місця/параметрів CRC/ECC, а також дуальної реалізації обчислення контрольної суми на різних технічних шляхах) забезпечує технологічну незалежність плечей і суттєве зниження кореляції помилок, що напряму зменшує ризики відмов за спільною причиною. Процедура дає керовані межі «часу до виявлення», забезпечує необхідне «покриття» адресного простору пам’яті, а також дозволяє балансувати накладні витрати протоколу, обсяг ресурсів ПЛІС/ЦП і енергоспоживання, зберігаючи детермінізм тракту завдяки узгодженню з диверсною синхронізацією. Сукупно це створює багатошарову, відтворювану архітектуру контролю цілісності даних у транспорті та збереженні, де однаковий змістовний критерій реалізується різними засобами та в різних часово-просторових профілях, забезпечуючи прийнятний компроміс між безпечністю, продуктивністю та вартістю.

3.5 Висновки до третього розділу

У результаті виконання третього розділу розроблено та обґрунтовано комплекс методів програмно-апаратної диверсності, що зменшують імовірність відмови за спільною причиною в програмовних платформах інформаційно-керуючих систем. Інтегральна концепція поєднує чотири взаємодоповнювальні напрями:

1. **Диверсна синхронізація.** Запропоновано методику формування зсунутого у часі кортежу тактування для функціонально еквівалентних каналів.

Показано, що підбір раціональних зсувів та профілів активності (у зв’язці з кластеризацією діаграм роботи каналів) приводить до зменшення перетину «вікон уразливості» й зниження синфазних комутаційних піків. Експериментальні таблиці й побудовані графіки підтвердили: після застосування зсувів статистика збігів подій і максимуми миттєвого навантаження зменшуються, а фазова декореляція каналів зберігається при дотриманні вимог до продуктивності.

2. **Структурно-просторова диверсність матриць логічних елементів і таймінгу.** Розроблено підхід до отримання альтернативних фізичних реалізацій одного і того ж проєкту ПЛІС шляхом зміни інструментальних налаштувань (зокрема, «Use smart compilation» та профілю «Optimization Technique: Speed/Balanced/Area») і мінімальних структурних модифікацій. Отримано просторі карти розміщення та набори критичних шляхів, що істотно відрізняються між каналами при збереженні функціональної тотожності. Практична придатність підтверджена на промислових модулях платформи RadICS.

3. **Диверсність програмно-апаратних засобів і процедур підрахунку контрольної суми.** Сформовано двоканальний контроль цілісності програмного коду: еталонне значення обчислюється на етапі компіляції Tcl-скриптом, а у виконанні періодична перевірка виконується апаратно-програмним шляхом. Внаслідок різниці мов, середовищ і механізмів виконання метод забезпечує технологічну незалежність плечей контролю та суттєве зниження ризику однаково-хибного підрахунку. Запропоновано детермінувати верхню межу «часу до виявлення» помилки через період перевірок і розмір блоків.

4. **Диверсність методів передачі даних та перевірки цілісності пам’яті.** Для каналу «передавач–приймач» одна й та сама контрольна сума розраховується різними реалізаціями (апаратною на одному боці та програмно-табличною на іншому) при однакових параметрах полінома та кадру. Додаткове підсилення досягається поєднанням різних профілів протоколів і просторовим рознесенням модулів. Для пам’яті поєднано службові поля контролю з фоновим

скануванням з рознесеними у часі профілями перевірок, що забезпечує керований «час до виявлення» і зниження чутливості до завад.

Методи працюють узгоджено: диверсна синхронізація розводить активність у часі, структурно-просторова диверсність — у фізичному просторі й по таймінгу, а двоканальні процедури контролю (у транспорті, пам’яті та для ПЗ) забезпечують алгоритмічну й технологічну незалежність перевірок. У сукупності це формує багатошарову бар’єрну модель захисту від відмов за спільною причиною з контрольованою ресурсною вартістю та відтворюваними параметрами.

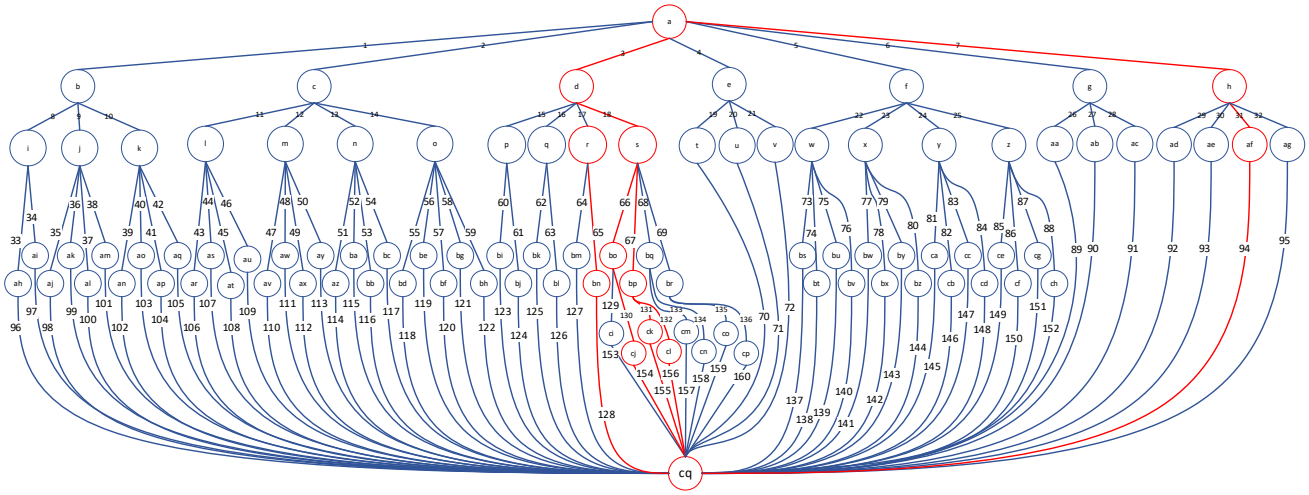

Рисунок 3.20 - Ілюстрація видів та методів диверсності, розглянутих в даному розділі

Додатково є можливість зручно узагальнити результати за допомогою інтегрального графу (Рисунок 3.20). Це багаторівневий орієнтований ациклічний граф, у якому вузли відповідають ключовим рішенням та трансформаціям життєвого циклу (від вибору підходу й налаштувань інструментів до конкретних реалізацій IP-ядер і процедур контролю), а дуги — причинно-наслідковим зв’язкам між ними. Червоним кольором виділено саме ті гілки, що відповідають методам, розглянутим у цьому розділі: диверсна синхронізація; структурно-просторова диверсність розміщення/таймінгу; диверсифікація процедур

підрахунку контрольних сум; диверсні методи передачі даних і перевірки цілісності пам’яті. Ширина та розгалуження дерева відображають кількість альтернативних рішень на кожному етапі (налаштування компілятора, варіанти топології/кластеризації, вибір полінома/реалізації CRC, профілі протоколів тощо).

За результатами даного розділу ***удосконалено*** **методи програмно-апаратної диверсності**, які базуються на процедурах функціонально-просторової диверсифікації засобів шинної організації, контролю цілісності даних, налаштувань проєкту та синхронізації з керованими зсувами, що забезпечує підвищення енергоефективності та/або безпечності завдяки зменшенню ризиків завад та відмов спільних логічних елементів версій.

Матеріали розділу опубліковані в [65; 70].

РОЗДІЛ 4
РОЗРОБЛЕННЯ МЕТОДУ ТА ЗАСОБІВ МОДЕЛЬНО-БАЗОВАНОЇ ВЕРИФІКАЦІЇ ПРОГРАМОВНИХ ПЛАТФОРМНИХ РІШЕНЬ ДЛЯ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ. ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

4.1 Метод модельно-базованої верифікації систем програмовної логіки

Верифікація систем програмовної логіки, які використовуються у відповідальних інформаційно-керуючих системах, є одним із ключових етапів забезпечення їх функційної безпечності. Традиційні методи тестування, засновані на емпіричному підборі тестів та експертному аналізі коду, виявляються недостатньо ефективними для складних архітектур, що поєднують програмні та апаратні компоненти. Це пояснюється як значним числом можливих станів системи, так і впливом прихованих помилок проєктування, які не завжди можуть бути виявлені на етапі традиційної відладки. Тому виникає необхідність у використанні формалізованих методів, що дозволяють забезпечити більшу глибину контролю відповідності між еталонною моделлю та реальною реалізацією.

Одним із найбільш перспективних підходів у цьому напрямку є тестування на основі моделей (Model-Based Testing, MBT) [65; 124; 124–128]. Сутність методу полягає у створенні формальної моделі системи, що описує її функціонування у вигляді множини станів та переходів, з подальшим автоматизованим порівнянням результатів роботи моделі та еталонного проєкту. Такий підхід дозволяє мінімізувати вплив людського фактора, автоматизувати процес генерації тестів і забезпечити відтворюваність результатів. Використання середовищ функціонального програмування, зокрема ForSyDe, створює основу для формального опису поведінки цифрових систем, що у свою чергу дозволяє інтегрувати процедури верифікації безпосередньо в цикл розроблення платформних рішень для критично важливих застосувань.

4.1.1 Сутність методу та обґрунтування мов програмування для модельно-базованої верифікації

При розробленні програмних та апаратних компонентів на основі програмовної логіки (FPGA/SoPC) існує розрив між специфікацією системи на рівні сигналів та її фактичною реалізацією у вигляді коду для soft-процесора або VHDL коду. У традиційних підходах цей розрив компенсується тестуванням на основі емпіричних даних, що не завжди гарантує повне охоплення можливих сценаріїв. Особливо це критично для ІКС, де приховані помилки можуть призвести до відмов за загальною причиною. Тому виникає потреба у методах, які забезпечують системну, формалізовану перевірку відповідності між еталонним проєктом та його альтернативною моделлю.

Тестування на основі моделей (або модельно-базована верифікація) ґрунтується на побудові моделі поведінки та моделі ситуацій, з яких автоматично або напівавтоматично генеруються тестові сценарії. Поведінкова модель відображає функціонування системи у вигляді мережі станів і сигналів, тоді як ситуаційна модель визначає набір можливих подій та їх пріоритетність. Сукупність тестів формується так, щоб покрити всі критичні класи ситуацій, не вимагаючи повного перебору. Порівняння результатів роботи моделі та еталонного проєкту дозволяє виявляти відхилення на ранніх етапах, мінімізуючи вплив людського фактора.

Для реалізації диверсної моделі використовується середовище ForSyDe (Formal System Design), побудоване на базі функціональної мови Haskell. Основна перевага функціонального підходу полягає в детермінованості обчислень: однакові вхідні дані завжди породжують однаковий результат. Це відповідає природі сигналів і процесів у цифрових системах та дозволяє створювати формалізовані моделі з можливістю подальшого автоматичного перетворення у VHDL-код для імплементації у ПЛІС.

Еталонна ж реалізація виконується наприклад, у звичному середовищі для soft-процесора Nios/Nios II (мова C, використання DMA, шинних інтерфейсів та

послідовного виконання інструкцій). Таким чином формується диверсна пара: модель на основі Haskell/ForSyDe та еталонний проєкт на основі C/Nios. Різниця у мовах, принципах виконання та інструментарії створює додатковий рівень захисту від спільних помилок.

Базові конструкції ForSyDe [129–133]. Для побудови моделей використовуються примітиви рівня процесів, наприклад:

- **mapSY(f)** – перетворення сигналу за функцією;
- **delaySY(s0)** – затримка сигналу з початковим значенням;
- **sourceSY(f,s0)** – генерація сигналу за рекурентним співвідношенням.

Композиція таких процесів дозволяє створювати складні моделі, наприклад AddFour – послідовність чотирьох функцій mapSY(+1), які сумарно збільшують значення сигналу на чотири (Рисунок 4.1). Приклад коду у Haskell демонструє простоту формального опису поведінки системи.

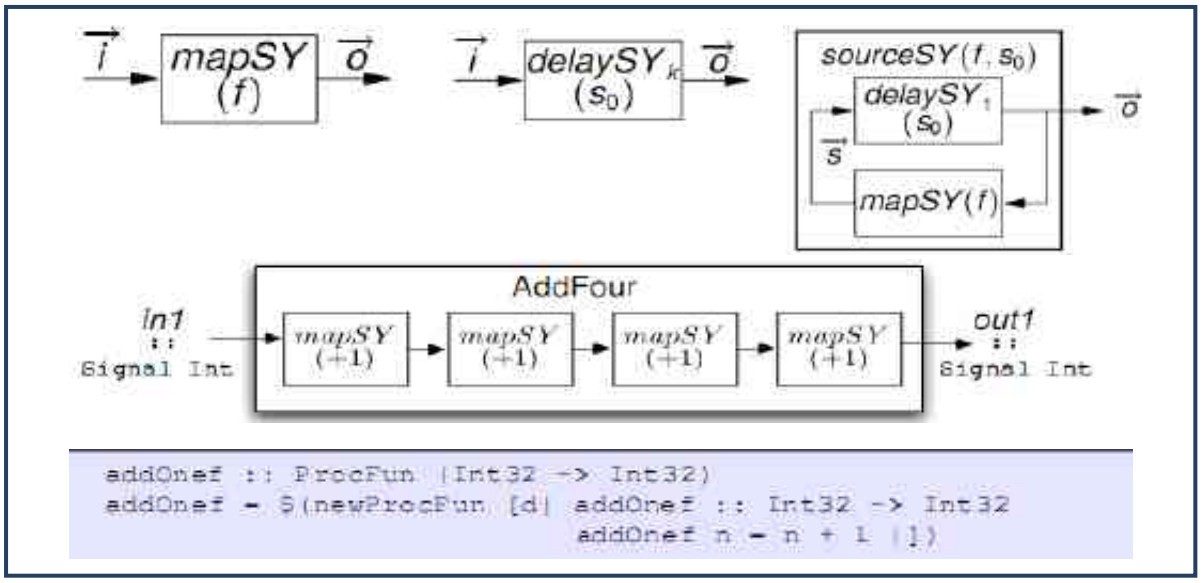

Рисунок 4.1 - Приклади базових операторів ForSyDe (mapSY, delaySY, sourceSY) та схема AddFour [130]

Еталонний проєкт описується у вигляді вхідних і вихідних сигналів (Рисунок 4.2). Для нього будується альтернативна модель у ForSyDe, яка працює паралельно з еталонною реалізацією. На обидві версії подаються однакові вхідні

дані, після чого результати автоматично порівнюються за допомогою алгоритму чи модуля порівняння. Таким чином забезпечується незалежна перевірка коректності поведінки системи.

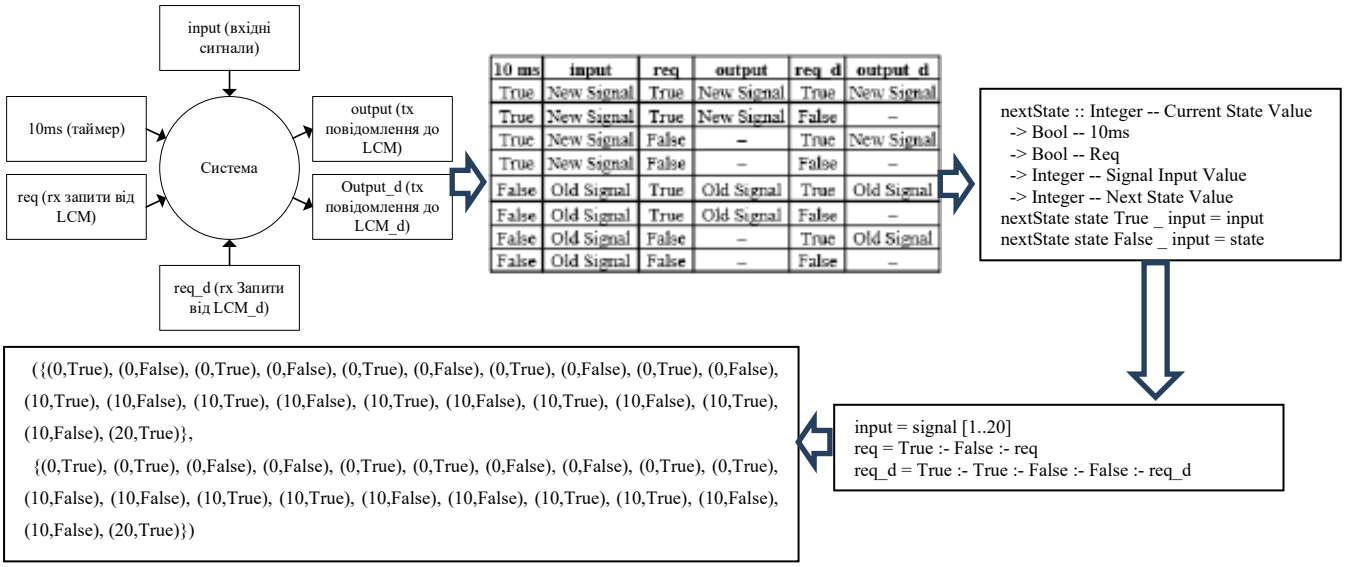

Рисунок 4.2 - Загальна схема перетворення еталонного проєкту у модель ForSyDe [65; 125; 126]

Для прикладу розглянемо декілька базових примітивів на рівні вхідних та вихідних сигналів:

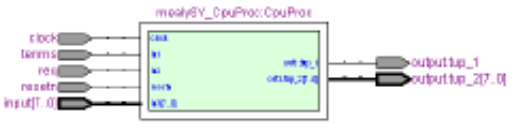

Рисунок 4.3 - Модуль CpuProc (mealySY_CpuProc) [65; 124–127]

Цей блок є прикладом побудови процесора переходів на основі моделі Мілі (Mealy machine).

– Входи: clock, reset, req, масив input[].
– Виходи: output_tup_1, output_tup_2.
– Особливість: вихідні дані залежать одночасно від поточного стану і поточних вхідних сигналів.

– Значення: показує, як у ForSyDe реалізується центральна логіка переходів, яка в традиційній мові описувалась би як автомат станів.

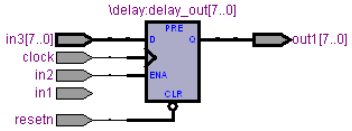

Рисунок 4.4 - Модуль delay_out [65; 125–127]

Апаратна реалізація примітиву delaySY:

– Використовує тригери (D-FF) із сигналами ENA, CLR, PRE.
– Призначення: зберігати значення сигналів між тактами та формувати історію станів.
– Значення: без delay неможливо побудувати автомат станів, оскільки він оперує не лише поточним входом, але й попереднім станом.

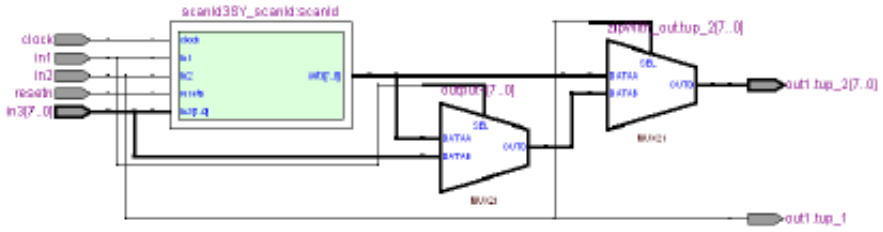

Рисунок 4.5 - Модуль scandl3SY [65; 124–127]

Комплексний модуль, який поєднує процесор переходів, затримки і мультиплексори.

– Входи: clock, reset, in1, in2, in3[].
– Виходи: out1_tup_2, out1_tup_1.
– Призначення: виконання сканування і попередньої обробки масивів дискретних сигналів.
– Значення: демонструє, як із базових примітивів ForSyDe утворюються складні компоненти, готові для синтезу у VHDL.

Рисунок 4.2 містить приклади вхідних і вихідних масивів: input = signal [1..20], req, req_d → вихідні дані у вигляді масивів True/False. Це підтверджує, що функціональна модель працює з масивами сигналів, а не з окремими регістрами, і вже на рівні Haskell-опису дозволяє відтворювати поведінку цифрової схеми.

Ці схеми демонструють, як у середовищі ForSyDe на основі базових примітивів створюються більш складні компоненти, що відтворюють роботу автоматів станів і забезпечують формування вихідних кадрів сигналів.

4.1.2 Послідовність модельно-базованої верифікації

Модельно-базована верифікація систем програмовної логіки базується на принципі незалежної побудови двох реалізацій однієї функції — еталонної та диверсної моделі. Еталонна реалізація зазвичай створюється традиційними засобами (наприклад, мовою C для soft-процесора Nios), тоді як диверсна модель описується у функціональній парадигмі (ForSyDe/Haskell) з подальшою трансформацією у VHDL та синтезом у ПЛІС. Це дозволяє уникнути спільних помилок унаслідок використання різних підходів і інструментів, а також забезпечує формальний контроль відповідності результатів.

Послідовність МВТ складається з низки етапів, які охоплюють опис вхідних і вихідних сигналів, формалізацію алгоритму роботи, побудову структурної моделі та автоматизоване порівняння результатів [126–128; 134; 135].

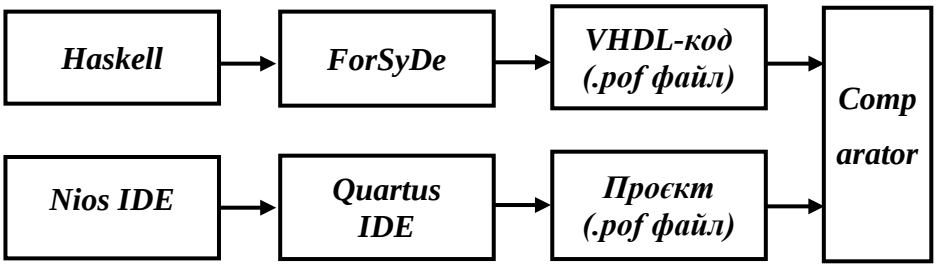

Рисунок 4.6 - Узагальнена блок-схема процесу МВТ: етапи опису еталонної системи, побудови моделі та порівняння результатів

Дана схема (Рисунок 4.6) демонструє загальну послідовність кроків процесу модельно-базованої верифікації для систем програмовної логіки. Вона відображає два незалежні шляхи побудови реалізацій: диверсної моделі (верхня гілка) та еталонного проєкту (нижня гілка), а також кінцеву стадію — автоматизоване порівняння результатів у модулі Comparator. Рисунок концентрує увагу не на деталях внутрішніх алгоритмів, а на загальній організації інструментальних потоків та кінцевому етапі верифікації.

Цей рисунок задає загальну канву всього процесу: він показує, що МВТ — це не лише тестування на рівні програмного коду, а комплексна процедура, яка охоплює і створення моделі у функціональній парадигмі, і розроблення еталонного проєкту в імперативному стилі, і апаратну імплементацію, і підсумкове порівняння результатів. Саме завдяки такому подвоєному підходу забезпечується диверсність, що істотно знижує ризик відмов за загальною причиною.

На прикладі модуля введення дискретних сигналів, алгоритм роботи системи (Рисунок 4.7) може бути представлений у вигляді вхідних та вихідних сигналів і процесів їх оброблення. Система має чотири вхідні сигнали (*Input*, *10 ms*, *Rx request from LCM*, *Rx request from LCM_d*) і два вихідні (*Tx message to LCM*, *Tx message to LCM_d*), оброблення яких виконується у модулі *System*.

Усі вхідні сигнали мають визначені значення, що впливають на їхню обробку та на формування вихідних сигналів (Рисунок 4.2). Наприклад, при переповненні 10-мс таймера (10 ms = True) здійснюється зчитування оновлених даних з порту (input = New Signal), і при наявності запиту на передачу (req = True) відбувається передавання оновлених даних (output = New Signal). Якщо таймер не активний (10 ms = False), зберігаються попередні зчитані значення.

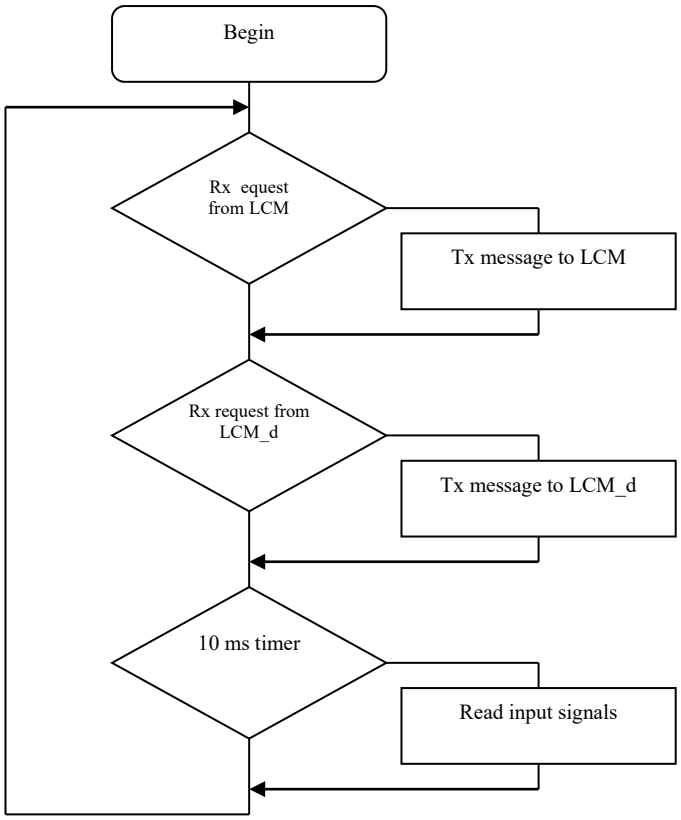

Рисунок 4.7 – Спрощений алгоритм програми модуля введення дискретних сигналів [65; 124–128]

У моделі, побудованій у середовищі ForSyDe, виділяються такі основні компоненти:

1. **Вхідні сигнали системи** – змінні, що надходять із зовнішніх джерел:

input – порт зчитування даних;

req, req_d – запити на передачу у функціональний та діагностичний канали.

2. **Модуль 10-мс таймера**, що базується на лічильнику Counter0_9 і компараторі Comp9, дозволяє сформувати періодичний імпульс із заданою затримкою.

3. **Модуль обробки даних** виконує зчитування нових значень з порту input при спрацюванні таймера та зберігає попередні значення за його відсутності. Функція nextState реалізує декодер вхідних сигналів системи:

```
nextState :: Integer -> Bool -> Bool -> Integer -> Integer
nextState state True _ input  = input
nextState state False _ input = state
```

У цій функції перший параметр (state) — попередній результат, другий (Bool) — прапор переповнення таймера 10 мс, третій — запит на передавання (req), який не впливає на результат (позначений символом _), а четвертий — поточне вхідне значення. Таким чином, якщо таймер активний, система зчитує нове значення, інакше зберігає старе.

4. **Функція cpuApp** є ядром системи. Вона керує процесами nextState і enable_output, формуючи результати обчислень:

```
cpuApp :: Signal Bool -> Signal Bool -> Signal Integer -> Signal (Integer, Bool)
cpuApp timer req input = output
  where out     = scanld3SY nextState initial timer req input
        output  = zipWithSY enable_output out req
        initial = 0
```

Тут використовується бібліотечна функція scanld3SY, що керує викликом nextState, зберігаючи попередній результат, та функція zipWithSY, яка реалізує об’єднання результатів з логічним запитом у єдиний сигнал для передавача.

5. **Функція enable_output** виконує перетворення двох вхідних параметрів у структурований вихід:

```
enable_output :: Integer -> Bool -> (Integer, Bool)
enable_output input req =
  if req == True then (input, True)
  else (input, False)
```

Перший параметр (Integer) — числові дані для передачі, другий (Bool) — ознака активного запиту, а результатом є пара (Integer, Bool), що поєднує дані з прапором готовності.

6. Для моделювання роботи таймера використовується пара функцій **comp9** та **counter0_9**:

```
comp9 :: Integer -> Bool
comp9 9 = True
comp9 _ = False
```

та

```
counter0_9 :: Integer -> Integer
counter0_9 9 = 0
counter0_9 a = a + 1
```

comp9 генерує сигнал переповнення, коли лічильник досягає дев'яти, а counter0_9 імітує десятковий лічильник із перезапуском після кожних 10 циклів.

7. **Функція system** об'єднує всі підмодулі в єдину модель системи:

```
system :: Signal Bool -> Signal Bool -> Signal Integer
       -> (Signal (Integer, Bool), Signal (Integer, Bool))
system req req_d input = output
  where output = (cpuApp comp9_output req input,
                  cpuApp comp9_output req_d input)
        comp9_output = comp9Proc counterProc0_9
```

Вона приймає три вхідні сигнали (req, req_d, input) і формує два вихідні потоки — функціональний та діагностичний. Кожен вихід є парою значень (Integer, Bool), де перше — числові дані, друге — логічний сигнал активності передавача.

Для запуску компілятора слід відкрити системну консоль операційної системи, перейти до каталогу, у якому зберігається файл створеної програми, та виконати команду: ghci <ім'я_файла>. Після цього відбудеться запуск компілятора, і за допомогою команди **:r** здійснюється компіляція файлу. Якщо помилок не виявлено, викликом головної функції програми **system <параметри>** запускається виконання створеної програми.

Результатом виконання є двовимірний масив, у якому перший аргумент — числове значення даних у буфері передавача, а другий — булевий сигнал активності. Отримані результати підтверджують коректну синхронізацію обох каналів і еквівалентність поведінки системи в Haskell-моделі та подальшому VHDL-описі.

За допомогою вбудованого у ForSyDe конвертора модель може бути автоматично трансформована у VHDL-код. Отриманий опис було проаналізовано в середовищі Quartus II, де побудовано структурну схему цифрового пристрою

(Рисунок 4.8). Ця схема відображає алгоритм функціонування програмної частини модуля введення дискретних сигналів у складі контролера на базі ПЛІС. Це еталонний алгоритм, реалізований мовою С у середовищі SoPC/Nios, що виконується на soft-процесорі в режимі безперервного циклу. Завершення його роботи можливе лише шляхом вимкнення живлення мікросхеми. Ключові операції виконуються з періодичністю у 10 мс, що задається таймером, а передача інформації у зовнішні вузли здійснюється виключно за наявності відповідного запиту.

Алгоритм складається з кількох етапів. На початковій стадії відбувається ініціалізація всіх змінних та пристроїв введення-виведення. Далі програма переходить у режим очікування запитів: від локального модуля керування — для передачі пакетів даних про стан дискретних входів, а також від діагностичного модуля — для відправки службової інформації. Формування пакетів супроводжується їх передачею через інтерфейс UART із використанням DMA, що знижує навантаження на процесор. Паралельно контролюється робота 10-мс таймера: у момент його спрацювання відбувається зчитування актуальних даних із входів, і ці значення зберігаються до наступного циклу.

Таким чином, схема фіксує порядок взаємодії з таймером та асинхронними запитами, визначаючи базовий поведінковий шаблон еталонної реалізації. Вона є вихідною точкою для побудови диверсної моделі у ForSyDe/Haskell, а також служить основою для формування тестових сценаріїв у процесі модельно-базованої верифікації.

Особливу увагу привертає випадок накладання подій: якщо одночасно відбувається спрацювання таймера (10 мс) і надходить запит від LCM, система повинна передати саме щойно зчитаний «новий» кадр, щоб уникнути використання застарілих даних. Це гарантує узгодженість між динамікою сигналів і логікою обробки. Таким чином, рисунок формалізує правила пріоритетності подій і є основою для побудови ситуаційної моделі у MBT.

Для підтвердження коректності роботи моделі в середовищі ForSyDe було створено експериментальну програму мовою С, що виконується на soft-процесорі

Nios II. Її завданням є передавання та приймання сигналів між основним і діагностичним каналами з використанням UART-інтерфейсів.

Нижче наведено фрагмент вихідного коду, який реалізує циклічну обробку даних у реальному часі:

```
#include <nios.h>
unsigned int ThisPlace;
unsigned int ThisPlaceDiagn;
unsigned int Signal;
int main() {
   na_FuncUart->np_uartcontrol = 0x00;
   na_FuncUart->np_uartstatus  = 0x00;
   na_DiagUart->np_uartcontrol = 0x00;
   na_DiagUart->np_uartstatus  = 0x00;
   while (1)
   {
      if (na_FuncUart->np_uartstatus & np_uartstatus_rdy_mask)
      {
         ThisPlace = na_FuncUart->np_uartrxdata;
         nr_txstring("F:"); nr_txhex(Signal);
         nr_txchar(' ');
         na_FuncUart->np_uarttxdata = Signal;
      }
      if (na_DiagUart->np_uartstatus & np_uartstatus_rdy_mask)
      {
         ThisPlace = na_DiagUart->np_uartrxdata;
         nr_txstring("D:"); nr_txhex(Signal);
         nr_txchar(' ');
         na_DiagUart->np_uarttxdata = Signal;
      }
```

```
        if (na_CycleTimer->np_timerstatus & np_timerstatus_to_mask)
        {
           Signal = na_InputData->np_piodata;
           nr_txstring("S:"); nr_txhex(Signal); nr_txchar(' ');
           na_CycleTimer->np_timerstatus = 0;
        }
     }
  }
```

У цій програмі оголошуються три глобальні змінні: ThisPlace — для збереження поточного прийнятого значення з функціонального UART, ThisPlaceDiagn — для діагностичного UART, та Signal — поточний сигнал, що передається обома каналами.

На початку роботи обидва UART-інтерфейси (*FuncUart* і *DiagUart*) ініціалізуються, після чого виконується нескінченний цикл while(1), який обробляє три основні події:

1. **Приймання даних функціональним каналом** – при встановленому прапорі готовності np_uartstatus_rdy_mask дані читаються з регістра np_uartrxdata і виводяться у монітор разом з ідентифікатором «F:».
2. **Приймання даних діагностичним каналом** – аналогічна операція, але з ідентифікатором «D:».
3. **Оновлення сигналу за таймером 10 мс** – коли активується прапор np_timerstatus_to_mask, у змінну Signal зчитується нове значення з порту введення na_InputData->np_piodata, після чого таймер скидається.

Таким чином, програма реалізує тестовий цикл обміну даними між двома передавачами, синхронізований із системним таймером 10 мс. Вона забезпечує спостереження у реальному часі за функціональною та діагностичною гілками системи, що дозволяє експериментально підтвердити коректність поведінки моделі, описаної у ForSyDe.

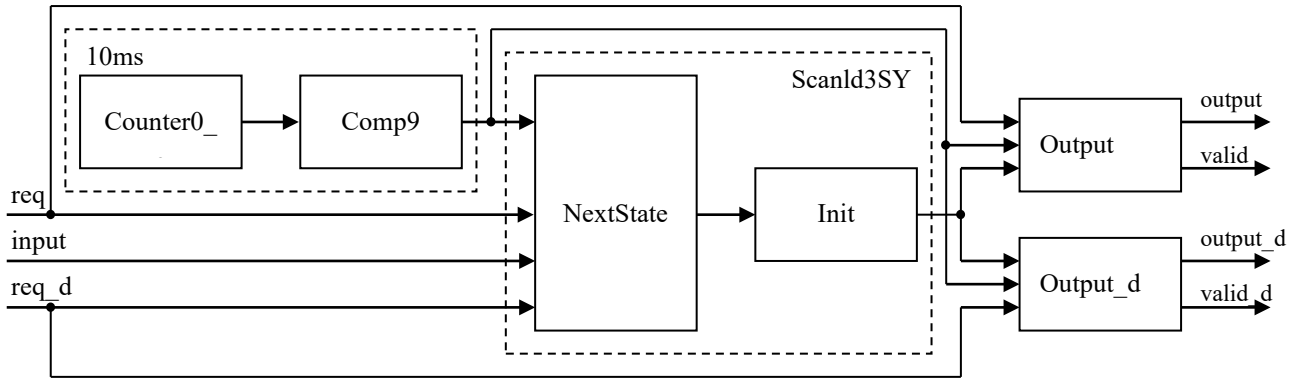

Рисунок 4.8 - Структурна модель NextState–Init–Output для реалізації алгоритму обробки сигналів [65; 125–127]

Рисунок 4.8 ілюструє структурну модель модуля введення дискретних сигналів у середовищі ForSyDe. На відміну від попередньої блок-схеми алгоритму (Рисунок 4.7), тут поведінка подана у вигляді архітектурних блоків, що відображають процеси та потоки даних. Це апаратно-орієнтований опис, який дозволяє трансформувати модель у VHDL-код і виконати синтез у ПЛІС. Такий рівень подання забезпечує точну відповідність між алгоритмом і реальною апаратною реалізацією.

Складові частини моделі:

– **Модуль 10 ms (Counter0_, Comp9).** Блок складається з лічильника (*Counter0_*) та компаратора (*Comp9*). Лічильник генерує періодичні імпульси, які компаратор перетворює у сигнал «10 ms». Цей блок відповідає за регулярне семплювання вхідних дискретних сигналів.

– **NextState** Це центральний блок логіки переходів. Він приймає на вході події від таймера (10 ms), запити (*req*, *req_d*) та поточні вхідні сигнали (*input*). На основі цих даних обчислюється наступний стан системи. По суті, NextState виконує функцію переходів у скінченному автоматі, що визначає реакцію системи на події.

– **Init.** Блок ініціалізації, який зберігає та оновлює стан системи. Init отримує значення від NextState та забезпечує їх узгодження з вихідними

сигналами. Це елемент синхронізації: він гарантує, що зміни стану відбуваються у такт із тактовим доменом.

– **ScanId3SY.** Узагальнений контейнер, що об’єднує внутрішні компоненти системи (таймер, NextState, Init). Він задає структуру потоків даних і регламентує взаємодію всіх елементів.

– **Output та Output_d**. Два вихідних блоки, що формують результати роботи для основного каналу (output, valid) та діагностичного каналу (output_d, valid_d).

Рисунок 4.7 та Рисунок 4.8 демонструють різні рівні уявлення модельно-базованої верифікації: від загальної схеми інструментальних потоків до алгоритмічного опису та структурної моделі. Вони показують, що метод ґрунтується на чіткій відповідності між концептуальним алгоритмом, його програмною реалізацією та апаратною моделлю. Однак для практичного застосування необхідно не лише представити архітектурні зв’язки, але й визначити конкретну послідовність дій, які забезпечують повний цикл верифікації:

Крок 1. Формалізація еталонної системи на рівні сигналів. Першим етапом є побудова формальної моделі інтерфейсу системи. Визначаються: вхідні сигнали (події від датчиків або інших модулів, наприклад *input*, *req*, *req_d*); вихідні сигнали (результати роботи, наприклад *output*, *output_d*, а також сигнали підтвердження *valid*); часові характеристики (періодичні події, наприклад 10-мс таймер); умови ініціалізації (початкові значення станів, скидання).

Цей рівень є незалежним від конкретної мови програмування чи платформи. У подальшому саме він визначає, що вважати «правильною поведінкою» системи. Еталонна реалізація при цьому виконується як послідовна програма на soft-процесорі (Nios), що є класичною архітектурою SoPC. Програма має доступ до апаратних таймерів, обробляє запити, зчитує дані з портів вводу/виводу й формує відповіді у вигляді кадрів даних.

Крок 2. Побудова альтернативної моделі у середовищі ForSyDe. На основі специфікації сигналів створюється функціональна модель у середовищі

ForSyDe. Модель будується як мережа процесів, які взаємодіють між собою через канали сигналів. На відміну від еталонної версії, що є послідовною програмою, модель у ForSyDe виконується паралельно. Це створює інший тип архітектури, завдяки чому диверсність досягається вже на рівні обчислювальної парадигми: імперативна послідовність проти функціонального паралельного опису.

Крок 3. Формалізація алгоритму роботи системи. Описується логіка функціонування модуля, яка має бути однаковою для обох версій. Алгоритм подається у вигляді блок-схем та структурних діаграм. У еталонній програмі це реалізується як цикл з перевіркою умов та викликом функцій обробки.

Крок 4. Виконання обох версій та накопичення результатів. На цьому етапі формується набір тестових стимулів, який включає: регулярні події, навантажувальні сценарії, граничні випадки та сценарії з ін’єкцією збоїв.

Обидві системи — еталонна та модельна — виконуються паралельно, отримуючи однакові вхідні дані. Результати зберігаються у вигляді часово мічених масивів сигналів, що дозволяє не лише порівняти виходи, а й виконати подальший аналіз часових характеристик.

Крок 5. Автоматизоване порівняння результатів. На завершальному етапі використовується спеціальний модуль порівняння (Comparator), який виконує такі дії: синхронізація часових шкал обох систем (з урахуванням джитера та невеликих відхилень); порівняння вихідних сигналів у кожен момент часу; фіксація розбіжностей у протоколі з докладним описом (момент часу, тип події, очікуване та фактичне значення); формування вердикту щодо проходження або непроходження тесту. Таким чином, перевірка відбувається автоматично і не залежить від суб’єктивного аналізу оператора. При збігу результатів система вважається верифікованою; при виявленні розбіжностей цикл повторюється з уточненням моделі, алгоритму або тестових сценаріїв.

Запропонована послідовність дій дозволяє реалізувати модельно-базовану верифікацію як повний життєвий цикл — від формалізації сигналів до автоматизованого порівняння результатів. Ключова особливість методу полягає в тому, що еталонна та диверсна реалізації створюються у принципово різних

середовищах, що мінімізує ризик повторення тих самих помилок і забезпечує більш надійну оцінку правильності роботи системи.

4.1.3 Імплементація моделі Model-Based Testing в hardware компоненту

Модельно-базована верифікація на попередніх етапах була розглянута як формальна концепція, що поєднує еталонну програмну реалізацію та диверсну модель у ForSyDe. Проте обмеження лише програмним моделюванням не дозволяє повністю оцінити поведінку системи у реальних умовах функціонування, де критично важливими є часові затримки, синхронізація сигналів та відповідність апаратним обмеженням. Саме тому наступним етапом методики є інтеграція моделі безпосередньо у апаратне середовище [136–138].

Рисунок 4.9 відображає повний процес апаратної імплементації диверсної моделі: від функціонального опису у ForSyDe до синтезованих блоків у FPGA та їхнього порівняння з еталонною програмною реалізацією на soft-процесорі Nios. На рисунку поєднані кілька рівнів подання: структурна схема NextState–Init–Output, програмні фрагменти еталонного коду, відображення роботи у середовищі розроблення, результат у вигляді таймінг-діаграм, а також тестування на реальній апаратній платі.

Цей рисунок є ключовим для розуміння підходу, оскільки показує, що модельно-базована верифікація не обмежується аналізом у симуляторі, а забезпечує повний цикл: формальний опис → автоматична трансформація у VHDL → синтез у FPGA → порівняння з еталонною програмою в режимі реального часу. Тож, Рисунок 4.9 демонструє повний шлях від функціональної моделі до апаратної реалізації та її зіставлення з еталонною програмою. Ліва/центральна частина — структурна схема моделі (таймер 10 мс, блок переходів станів NextState, ініціалізаційний блок Init, вихідні блоки Output/Output_d). Праворуч зверху — фрагменти еталонного коду на C та робоче середовище розроблення, нижче — фото плати з ПЛІС, на яку завантажено обидві реалізації.

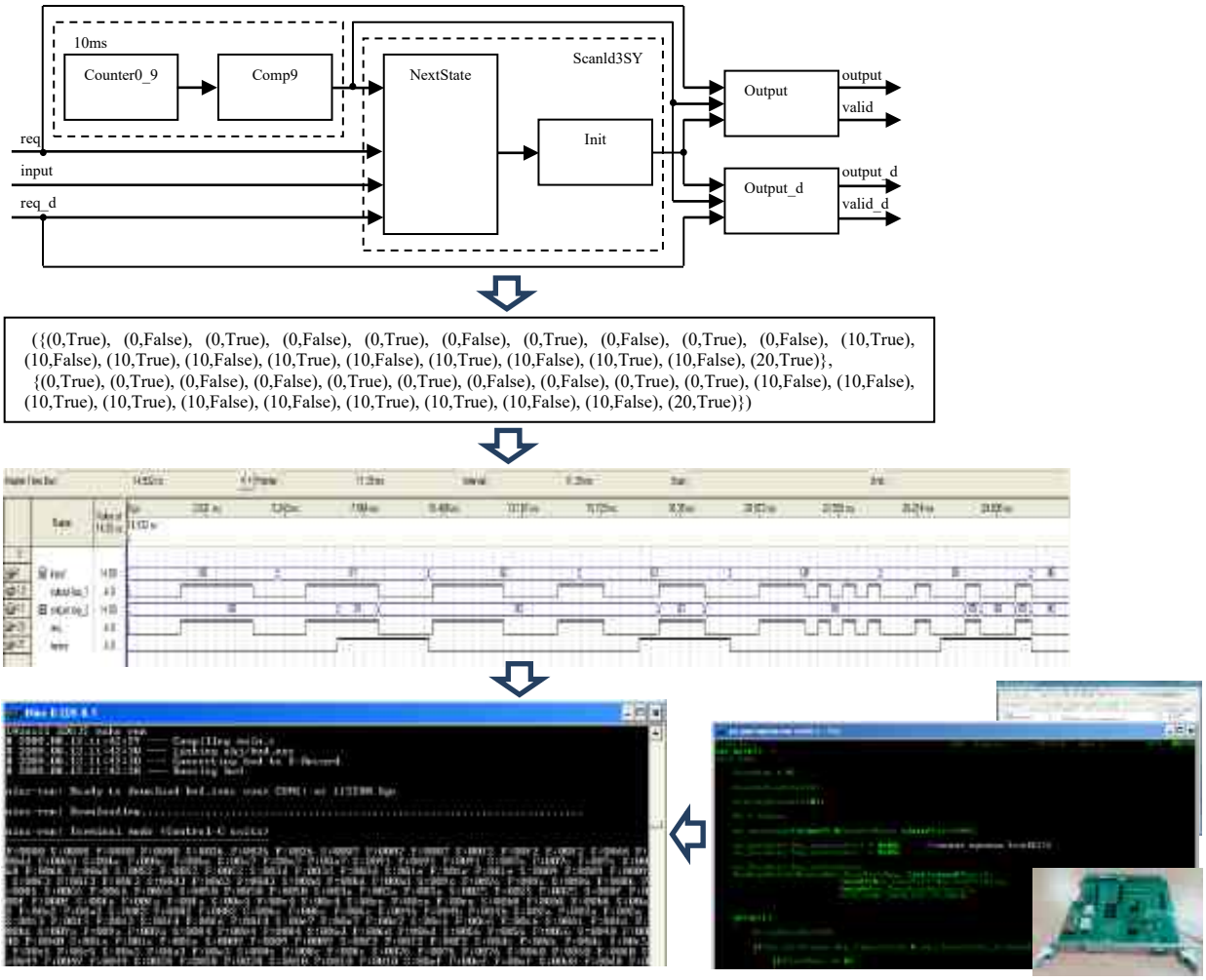

Рисунок 4.9 - Комплексна ілюстрація апаратної імплементації диверсної моделі та її порівняння з еталонною програмою на Nios [124–127]

Внизу — осцилограма (таймінг-діаграма), що фіксує реакцію сигналів у часі. Праворуч унизу — консоль Nios II EDS з журналом виконання еталонної програми. Рисунок потрібен для доведення, що диверсна модель не лише симулюється, а й синтезується у ПЛІС, працює паралельно з еталонною версією і дає порівнювані результати у реальному часі.

Етапи процесу відладки і порівняння результатів MBT та Nios:

1. **Підготовка**. Обидві реалізації збираються в один проєкт ПЛІС. Модель ForSyDe перетворена у VHDL і синтезована; еталонна програма зібрана у Nios II EDS. У прошивку додаються проби для ключових сигналів і UART-логер для еталонної програми.

2. **Завантаження**. Конфігураційний файл ПЛІС завантажуються; на платі запускається генератор сигналів (або робота з зовнішніми входами).

3. **Виконання**. Генератор подає запити req/req_d, чим відбувається моделювання роботи системи. Обидві версії працюють паралельно в одному часовому домені.

4. **Збір даних**. За допомогою логічного аналізатора (SignalTap), симулятора та від еталонної програми (UART-консоль із друком кадрів) відбувається збір діагностичних даних.

5. **Порівняння**. Для кожного запиту формується пара записів «MBT-кадр» та «Nios-кадр». Зіставляються поля кадрів (ідентифікатор, дані (input[]), контрольна сума) та момент їхньої появи відносно tick_10ms (допускається невеликий джитер у тактових межах).

6. **Вердикт**. При повному збігу вмісту кадрів і узгодженні у часі тест вважається пройденим.

Побудова таких апаратних моделей дає кілька практичних переваг:

- можливість відпрацювання алгоритмів у реальному масштабі часу;
- виявлення відмінностей у затримках та синхронізації ще до запуску системи у промислову експлуатацію;
- перевірка сумісності з інтерфейсами, реальними сигналами та апаратними обмеженнями;
- підготовка до сертифікації за нормами ядерної галузі, де потрібне доведення відповідності не тільки програмного коду, але й апаратної частини.

Апаратна імплементація диверсної моделі у FPGA є ключовим етапом методології модельно-базованої верифікації. Вона підтверджує, що побудована у ForSyDe модель не залишається лише у вигляді програмної симуляції, а може бути трансформована у фізичну схему, здатну працювати у реальному часі поряд з еталонною програмною реалізацією на soft-процесорі Nios.

Проведені експерименти показали, що обидві реалізації забезпечують синхронізовану реакцію на події, формують ідентичні вихідні кадри та

демонструють узгодженість часових характеристик. Це доводить адекватність методу МВТ та його придатність до практичного застосування у задачах перевірки критично важливих інформаційно-керуючих систем.

4.1.4 Експериментальні результати

Після теоретичного обґрунтування та побудови апаратно-програмних моделей наступним кроком стало проведення експериментальної перевірки працездатності та ефективності методу модельно-базованої верифікації. Основна мета експериментів полягала у підтвердженні того, що диверсна модель реалізована у середовищі ForSyDe та синтезована у ПЛІС, коректно відтворює функціональність еталонного проєкту на soft-процесорі Nios, а також у перевірці здатності методу виявляти розбіжності у поведінці системи.

Для цього було розгорнуто спеціальний тестовий стенд, який включав:

- генератор вхідних сигналів, що формував як регулярні, так і випадкові послідовності подій;
- еталонну реалізацію у вигляді програми ядра Nios, що забезпечувала послідовну обробку запитів і формування відповідей;
- модель ForSyDe, трансформовану у VHDL і синтезовану в FPGA, яка виконувала ті самі функції у паралельній архітектурі;
- автомат порівняння результатів, що аналізував вихідні дані обох реалізацій і видавав висновок про збіг чи розбіжності.

Для ілюстрації отриманих результатів у підрозділі наведено два ключові матеріали. Перший (Рисунок 4.9) відображає узагальнену схему експериментальної установки із зазначенням основних компонентів і потоків сигналів. Другий (Таблиця 4.2) містить порівняльні характеристики еталонної та модельної реалізацій за низкою параметрів (типи вхідних даних, архітектура виконання, використання шин, DMA та ПЛІС), що дозволяє кількісно оцінити ступінь диверсності між ними.

Таким чином, експерименти були організовані так, щоб з одного боку показати структурну відповідність обох систем у єдиному тестовому середовищі, а з іншого — кількісно підтвердити їх відмінності, які і забезпечують підвищену надійність при використанні методу МВТ.

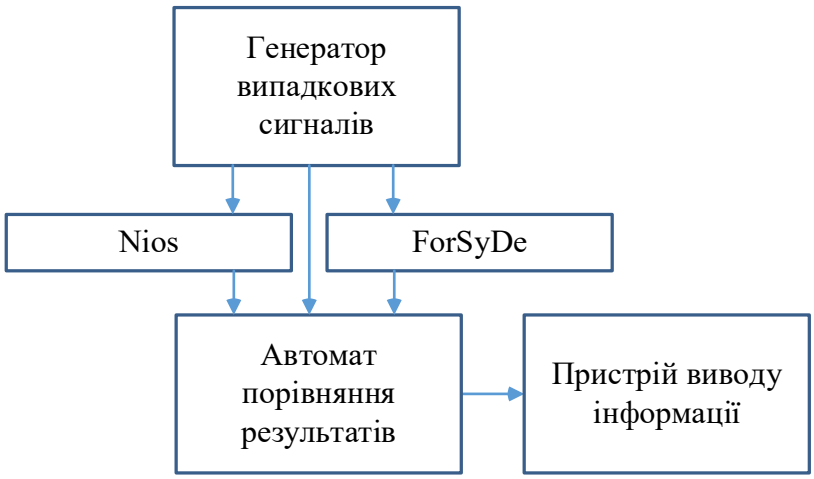

Рисунок 4.10 - Узагальнена схема експериментальної установки

Схема (Рисунок 4.10) відображає архітектуру стенду, використаного для перевірки коректності методу модельно-базованої верифікації: генератор вхідних сигналів подає однакові послідовності на дві незалежні реалізації — еталонну програму ядра Nios (послідовна імперативна обробка) та модель ForSyDe (паралельна обробка у функціональній парадигмі). Далі їхні виходи надходять до автомата порівняння, який виконує автоматичне зіставлення результатів і фіксує розбіжності у журналі для подальшої локалізації причин. Підсумки перевірки виводяться на пристрій індикації, що полегшує аналіз.

Генератор формує як регулярні імпульси, так і випадкові послідовності для моделювання нормальної роботи, пікових навантажень і аварійних ситуацій; компаратор забезпечує об’єктивність оцінки, а інтерфейс виводу — наочність. У сукупності схема демонструє ключову ідею МВТ: дві технологічно диверсні реалізації однієї функції синхронно тестуються однаковими впливами, а їхня еквівалентність або розбіжності визначаються без участі людини.

Таблиця 4.1 - Порівняльні характеристики еталонної реалізації (C/Nios) і моделі (ForSyDe/Haskell)

№	Параметр	Nios	ForSyDe	Ваговий коефіцієнт	Значення диверсності
1.	Вхідні дані	сигнал	масив	0,2	1
2.	Виконання програми	послідовне	паралельне	0,3	1
3.	Використання шини	так	ні	0,1	1
4.	Використання ПЛІС	так	так	0,1	0
5.	Розрядність просесора	16 біт	ні	0,2	1
6.	Використання DMA	так	ні	0,1	1

Таблиця 4.1 систематизує ключові параметри, за якими відрізняються дві незалежні реалізації: програмна (Nios) та модельна (ForSyDe). Таке порівняння дозволяє кількісно оцінити рівень диверсності та підтвердити, що реалізації є дійсно незалежними.

Із сукупного аналізу видно, що майже всі параметри вказують на повну незалежність реалізацій. Лише використання ПЛІС є спільним фактором, проте різниця у рівні абстракції (soft-процесор проти структурної логіки) все одно гарантує диверсність. Для об’єктивної оцінки застосовано метод адитивної згортки, що дозволяє обчислити інтегральний коефіцієнт диверсності:

Формула для коефіцієнта диверсності Kd:

$$K_d = \sum_{i=1}^{n=6} L_i \cdot d_i \,, \tag{4.1}$$

де L_i – ваговий коефіцієнт значущості параметра, d_i – оцінка диверсності для цього параметра, n – кількість критеріїв.

Формула дозволяє обчислити інтегральний коефіцієнт диверсності K_d, який показує, наскільки дві реалізації системи незалежні між собою. Чим більше значення K_d, тим вищий рівень різнорідності та, відповідно, менша ймовірність відмови за спільною причиною.

Інтерпретація результатів

– $K_d \approx 0$ — системи практично ідентичні, диверсність відсутня.

- K_d = 0.5…0.7 — досягнуто середнього рівня диверсності, є певні відмінності.
- $K_d \geq 0.8$ — високий рівень диверсності, системи незалежні за більшістю параметрів.

У нашому випадку підсумкове значення становило K_d = 0,9, що свідчить про високу диверсність між ядром Nios і моделлю ForSyDe та підтверджує мінімальний ризик відмов за спільною причиною.

Додатково було перевірено еквівалентність результатів на трьох рівнях: у вигляді блок-схеми алгоритму на мові C, у функціональному описі Haskell/ForSyDe та у VHDL-моделі, синтезованій у Quartus II. Порівняння часових діаграм, отриманих у середовищі ModelSim, підтвердило повну відповідність вихідних сигналів, що додатково обґрунтовує достовірність методу. Таким чином, експериментальна перевірка засвідчила не лише формальну коректність, а й практичну узгодженість різних середовищ реалізації.

4.2 Послідовність розроблення та верифікації двоверсійних систем з використанням запропонованих моделей і методів

Розглянутий у попередньому підрозділі метод модельно-базованої верифікації довів свою ефективність для окремих програмно-апаратних модулів, забезпечуючи формальне порівняння результатів роботи еталонного проєкту та диверсної моделі. Однак у практичних інформаційно-керуючих системах атомних електростанцій ключову роль відіграють багатоканальні та багатоверсійні архітектури, які використовують принцип диверсності для підвищення функційної безпечності та зниження ризику відмов за загальною причиною.

Подальший розвиток підходу полягає у поширенні модельно-базованої верифікації з рівня одного модуля на рівень комплексної двоверсійної системи. Це вимагає визначення загальної схеми розроблення та інтеграції версій, які відрізняються за мовами програмування, інструментальними середовищами, технологічними процесами або архітектурними рішеннями. Водночас необхідно

забезпечити узгоджене порівняння результатів роботи обох версій, з урахуванням як функціональних характеристик, так і часових параметрів.

У цьому підрозділі розглянуто послідовність побудови двоверсійних систем: від вибору альтернативних рішень та їх формалізованого опису до організації верифікації та комплексування методів програмно-апаратної диверсності. Особлива увага приділяється методології вибору диверсних пар та критеріям оцінки їх доцільності, що дозволяє систематизувати процес і забезпечити його практичну застосовність у проєктах підвищеної відповідальності.

4.2.1 Загальна схема розроблення та верифікації двоверсійних систем

Системи управління та захисту, що застосовуються в атомній енергетиці, повинні демонструвати стійкість до відмов, у тому числі відмов за загальною причиною. Традиційні одноверсійні рішення не забезпечують достатнього рівня функційної безпечності, оскільки навіть незначний дефект у проєкті або у середовищі розроблення може призвести до критичних наслідків. Тому виникає потреба у двоверсійних системах, де одна й та сама функція реалізується двома незалежними засобами.

Принциповим у такій архітектурі є диверсність — відмінності у мовах програмування, середовищах розроблення, інструментах синтезу, алгоритмах або апаратних реалізаціях. Завдяки цьому імовірність повторення одного й того самого дефекту в обох версіях суттєво зменшується.

Загальний процес створення та верифікації двоверсійних систем складається з таких етапів:

1. **Формалізація вимог.** На першому кроці визначаються функції, які має реалізовувати система (наприклад, зчитування дискретних сигналів, контроль аварійних параметрів, передача повідомлень у систему захисту). Опис вимог має бути максимально абстрактним і незалежним від обраних мов чи інструментів.

2. **Вибір двох незалежних шляхів реалізації.** Для підвищення диверсності обираються два різні підходи:

- перша версія реалізується традиційним методом, наприклад, на soft-процесорі (мовою C із використанням SoPC Builder, драйверів і DMA);
- друга версія реалізується у середовищі ForSyDe, із подальшою трансформацією у VHDL та синтезом у FPGA. У деяких випадках альтернативою може виступати інша мова HDL (наприклад, Verilog) або інший інструментальний ланцюг синтезу.

3. **Незалежне розроблення версій.** Обидві реалізації створюються різними командами або у різних середовищах. Це мінімізує ризик дублювання дефектів, пов’язаних із людським фактором чи специфікою конкретного інструменту.

4. **Верифікація кожної версії окремо.** На цьому етапі застосовується модельно-базована перевірка: еталонні тести подаються на кожну версію, результати порівнюються з очікуваними, а відхилення документуються.

5. **Інтеграція у двоверсійну архітектуру.** Після відлагодження обидві версії інтегруються у спільну систему. Передбачено, що вихідні результати кожної версії надходять у модуль порівняння та арбітражу. Якщо результати збігаються, рішення вважається коректним і передається далі у систему. Якщо ж виявляються розбіжності, активується механізм діагностики або аварійного переходу у безпечний стан.

6. **Системна верифікація.** На завершальному етапі проводиться тестування всієї двоверсійної системи з використанням повного набору сценаріїв, включно з навмисними помилками, перевантаженням і збуреннями.

Модуль порівняння результатів виконує ключову роль у забезпеченні узгодженості між двома версіями. Він має кілька функцій:

- синхронізація результатів (вирівнювання у часі вихідних сигналів обох версій);

- логічне порівняння (перевірка тотожності сигналів з урахуванням допустимих відхилень у часі або амплітуді);
- прийняття рішення (вибір результату для подальшої обробки, або переведення системи у безпечний стан у разі розбіжностей).

У практиці побудови ІКС часто використовується правило «обидві версії повинні збігатися», що забезпечує максимальну строгість. Проте для складних систем допускається також варіант «одна версія активна, інша у режимі моніторингу», де розбіжності трактуються як діагностичні події.

Під час перевірки двоверсійних систем особлива увага приділяється:

- функціональній еквівалентності — чи обидві версії забезпечують однакову логіку обробки сигналів;
- часовій еквівалентності — чи результати надходять у задані часові інтервали, що особливо критично для систем аварійного захисту;
- стійкості до відмов — чи система коректно реагує на розбіжності між версіями (наприклад, виявлення прихованої помилки в одній із реалізацій).

Застосування двоверсійної архітектури у поєднанні з модельно-базованою верифікацією дозволяє:

- суттєво знизити ризики відмов за загальною причиною;
- забезпечити документовану трасованість між вимогами, реалізацією та результатами тестування;
- створити основу для масштабування до багатоверсійних систем (три і більше незалежних каналів).

Для інформаційно-керуючих систем атомних електростанцій це означає, що критичні функції, пов’язані із захистом та аварійним реагуванням, можуть виконуватися з гарантованою надійністю навіть у разі появи дефектів в одній із версій.

4.2.2 Комплексування методів програмно-апаратної диверсності для платформних рішень

У попередньому підрозділі було розглянуто загальну схему побудови двоверсійних систем. Наступним кроком є визначення способів досягнення необхідної диверсності між версіями. Саме комплексування методів означає свідоме поєднання різних підходів у розробці програмних та апаратних компонентів таким чином, щоб максимально знизити ймовірність повторення одних і тих самих дефектів у різних версіях.

Застосування комплексування особливо важливе для платформних рішень на основі FPGA та SoPC, де часто використовуються однакові засоби синтезу, стандартні IP-бібліотеки та повторювані шаблони. Без належної диверсифікації ризик CCF значно зростає.

Основні напрямки програмно-апаратної диверсності, розглянуті в даній дисертації:

1. Мовна диверсність.
2. Інструментальна диверсність.
3. Апаратна диверсність.
4. Алгоритмічна диверсність.
5. Диверсність синхронізації.

Таблиця 4.2 демонструє конкретні приклади реалізації комплексування методів програмно-апаратної диверсності для платформних рішень. У ній наведено порівняння двох версій реалізації кожного з ключових компонентів: мови опису, середовища розроблення, апаратної реалізації, алгоритму та способів синхронізації. Для кожного випадку визначено відповідний тип диверсності, що вносить якісні відмінності між версіями. Це дозволяє на практичному рівні показати, як різні підходи можна поєднувати, щоб знизити ризик відмов за спільною причиною.

Таблиця 4.2 - Варіанти комплексування методів програмно-апаратної диверсності для платформних рішень.

Компонент	Версія 1	Версія 2	Тип диверсності
Мова опису	C для Nios	Haskell/ForSyDe → VHDL	Мовна
Середовище	SoPC Builder / Quartus	ForSyDe + інший HDL-транслятор	Інструментальна
Апаратна реалізація	Soft-процесор	Синтезовані апаратні блоки	Апаратна
Алгоритм	Циклічне опитування	Подієво-орієнтована обробка	Алгоритмічна
Синхронізація	Глобальний тактовий генератор	Локальні синхронні домени	Синхронізаційна

Завдяки таблиці видно, що диверсифікація може реалізовуватись на різних рівнях — від мовної та інструментальної до алгоритмічної й синхронізаційної. Наприклад, одна версія може базуватись на імперативному підході з використанням мови C і soft-процесора, тоді як інша реалізація — на функціональній парадигмі ForSyDe/Haskell із синтезованими апаратними блоками. Поєднання таких різних підходів гарантує, що навіть у разі прихованих дефектів в одній версії, вони з високою ймовірністю не повторяться в іншій. Таким чином, Таблиця 4.2 виконує роль узагальненої матриці рішень, що ілюструє механізм досягнення диверсності у двоверсійних інформаційно-керуючих системах.

Застосування комбінованої диверсності дозволяє:

- підвищити стійкість системи до CCF – відмов, спричинених однаковими дефектами у програмі чи апаратурі;
- забезпечити багаторівневу перевірку – на рівні логіки, часових характеристик і синхронізації;
- отримати незалежні результати – завдяки використанню різних мов та інструментів;
- створити основу для масштабування – у майбутньому методика може бути поширена на трьох- і чотирьохверсійні архітектури.

Таким чином, комплексування методів програмно-апаратної диверсності є ключовим кроком у створенні надійних платформних рішень для інформаційно-керуючих систем атомних електростанцій. Воно дозволяє перетворити

двоверсійну систему з простого дублювання функцій на дійсно незалежний механізм забезпечення безпеки.

4.3 Впровадження результатів досліджень

Запропоновані у попередніх підрозділах методи модельно-базованої верифікації та комплексування програмно-апаратної диверсності були розроблені не лише як теоретичні концепції, а й як практичні інструменти для підвищення функційної безпечності критично важливих інформаційно-керуючих систем. Особливість проведених досліджень полягає в тому, що вони орієнтовані на безпосереднє застосування у промислових рішеннях та у навчальному процесі підготовки фахівців.

Значна частина результатів уже знайшла своє відображення у проєктах, реалізованих на програмовних платформах Radiy та RadICS, які застосовуються у системах безпеки атомних електростанцій. Ці платформи характеризуються високим рівнем надійності та спеціально розроблені для використання у складних об’єктах ядерної енергетики. Запровадження модельно-базованих методів дало змогу підвищити якість верифікації їхніх програмно-апаратних модулів, скоротити час випробувань та виявляти помилки ще на ранніх етапах розроблення.

Крім того, розроблені моделі та методи були адаптовані для використання у навчальних програмах підготовки спеціалістів з інформаційної безпеки та критичних ІКС. Це дозволило створити навчальні приклади та лабораторні стенди, на яких студенти можуть відпрацьовувати практичні навички побудови двоверсійних систем, використання ForSyDe та аналізу часових характеристик.

4.3.1 Систематизація результатів впровадження в проєктах і навчальному процесі

Результати досліджень, пов’язані з розробленням методу модельно-базованої верифікації та комплексуванням методів диверсності, були впроваджені у низці науково-технічних та освітніх проєктів. Це дозволило не лише перевірити

їх практичну ефективність, але й створити основу для широкого використання у майбутніх розробленнях.

Застосування модельно-базованих методів дало змогу сформувати єдину системну методологію перевірки програмно-апаратних рішень, яка включає:

- інтерфейсно-орієнтований опис системи на рівні сигналів і станів;
- створення диверсної моделі на основі ForSyDe з можливістю автоматичної трансформації у VHDL;
- синхронне виконання еталонної та модельної версій з однаковими наборами тестових даних;
- автоматизоване порівняння результатів з формуванням протоколу перевірки.

Така методика була використана при розробці програмних компонентів для платформ RadICS та Radiy, які реалізують функції захисту та управління технологічними процесами. Вона дозволила скоротити час випробувань за рахунок автоматизації процедури порівняння результатів, а також підвищити рівень довіри до кінцевої системи.

Результати впровадження можуть бути систематизовані у вигляді таких груп:

1. Методологічні результати — створено єдиний підхід до побудови диверсних моделей та організації автоматизованої верифікації.

2. Технологічні результати — розроблено набір інструментальних процедур (опис сигналів, трансформація ForSyDe → VHDL, синтез у FPGA, модуль порівняння).

3. Освітні результати — підготовлено навчальні програми, лабораторні практикуми та методичні матеріали для студентів.

4. Практичні результати — методи впроваджено у промислові проєкти, що підтвердило їхню ефективність у реальних умовах.

Таким чином, систематизація результатів впровадження показує, що запропоновані методи є універсальними: вони можуть використовуватися як у

промислових проєктах для забезпечення функційної безпечності, так і в освітньому процесі для формування практичних навичок у майбутніх інженерів.

4.3.2 Інформаційно-керуючі системи АЕС на програмовних платформах Radiy, RadICS

Інформаційно-керуючі системи атомних електростанцій відносяться до категорії критично важливих систем, де надійність та функційна безпечність мають визначальне значення. Будь-яка відмова або некоректна робота може призвести до серйозних наслідків, тому до таких систем висуваються особливо жорсткі вимоги міжнародних стандартів, в яких прямо підкреслюється необхідність використання багатоверсійних рішень як засобу захисту від відмов за загальною причиною.

В Україні багаторічний досвід створення ІКС для АЕС накопичений у рамках проєктів НВП «Радій» та платформа RadICS базуються на широкому використанні програмовної логіки (FPGA, SoPC) та спеціалізованих апаратних модулів, що дозволяє реалізувати як стандартні функції керування, так і складні алгоритми аварійного захисту.

Запропонований у даній роботі метод МВТ був застосований для верифікації програмних і апаратних компонентів платформ Radiy та RadICS. Основними напрямами його впровадження стали:

– перевірка модулів введення-виведення — побудова диверсних моделей для модулів обробки дискретних і аналогових сигналів, їх трансформація у VHDL і синхронне виконання з еталонними реалізаціями;

– валідація алгоритмів аварійного захисту — створення моделей, які дублюють логіку блоків захисту, але реалізовані у середовищі ForSyDe, що дозволяє виявляти приховані помилки проєктування;

– перевірка часових характеристик — порівняння таймінгів роботи еталонних програм і синтезованих моделей для виявлення зсувів, що можуть впливати на своєчасність спрацювання захистів.

Таблиця 4.3 містить узагальнені результати впровадження запропонованих методик. У ній відображено, які саме наукові результати були застосовані у практичних проєктах, та в яких платформах або системах (Radiy, RadICS, АЗ-ПЗ, АРМ-РОМ) вони отримали підтвердження. Таблиця виконує функцію структурованого підсумку, що дозволяє простежити взаємозв'язок між теоретичними напрацюваннями та їхньою апробацією у реальних інформаційно-керуючих системах.

Таблиця 4.3 - Результати впровадження запропонованих методик і платформних рішень у реальних системах НВП «Радій» (Radiy), RadICS, а також у системах АЗ-ПЗ та АРМ-РОМ.

Науковий результат			
	Radiy	RadICS	АЗ-ПЗ (АРМ-РОМ)
Графова модель для опису можливості отримання різних програмно-апаратних версій для інформаційно-керуючих систем на програмовних платформах, з урахуванням деталізованого класифікатора видів диверсності, а саме: процесорних ядер, інтерфейсів, засобів синхронізації, технологічних процесів і обладнання.	+	+	+
Метод вибору видів диверсності з урахуванням багаторівневої ієрархії видів версійної надмірності, включно з внутрішньою диверсністю каналів, а також відповідні метрики диверсності і складності.	+	+	+
Методи програмно-апаратної диверсності, які базуються на процедурах функціонально-просторової диверсифікації засобів шинної організації, контролю цілісності даних, налаштувань проєкту та синхронізації з керованими зсувами.	+	+	
Метод модельно-базованого тестування систем програмовної логіки, який на відміну від відомих, базується на порівнянні версій, отриманих з використанням мови програмування Haskel та її конвертованої моделі у VHDL-код.	+		

Варто підкреслити, що платформи Radiy та RadICS належать до високотехнологічних рішень, які пройшли багаторівневу сертифікацію та успішно застосовуються у складі інформаційно-керуючих систем атомних електростанцій. Їхнє використання у промисловій експлуатації є підтвердженням високих вимог до надійності, функційної безпечності та відповідності міжнародним стандартам. Саме тому впровадження запропонованих у роботі методів модельно-базованої

верифікації у таких системах може розглядатися як об’єктивний індикатор їхньої практичної значущості. Отримані результати не обмежуються лабораторними експериментами, а пройшли апробацію в реальних умовах критично важливих ІКС, що істотно підвищує рівень довіри до розроблених підходів.

Таким чином, підтвердження ефективності методів МВТ на прикладі платформ Radiy і RadICS свідчить про їхню зрілість та готовність до масштабного застосування у системах атомної енергетики. Це дозволяє вважати результати дослідження не лише теоретично вагомими, але й практично доведеними, що є одним із ключових критеріїв для використання у сфері критичних інфраструктур.

Практичне використання запропонованого методу дало низку позитивних результатів:

– скорочення часу тестування завдяки автоматизації порівняння та можливості паралельної перевірки двох версій;

– підвищення надійності системи за рахунок багаторівневої диверсності;

– виявлення прихованих дефектів (зокрема, неузгодженостей у таймінгах);

– спрощення сертифікації завдяки формалізації методики та відтворюваності результатів.

Використання модельно-базованої верифікації та багатоверсійних архітектур на базі платформ Radiy і RadICS створює умови для підвищення рівня захисту ІКС АЕС. Це забезпечує не лише технічну надійність, але й відповідність нормативним вимогам. Завдяки такому підходу знижується ризик відмов за загальною причиною, а також підвищується довіра до результатів перевірки з боку регулюючих органів. Таким чином, результати досліджень знайшли практичне застосування у системах безпеки атомних електростанцій, що підтверджує їх високу цінність і практичну значущість.

4.4 Висновки до четвертого розділу

У результаті виконання четвертого розділу було розроблено, обґрунтовано та впроваджено метод модельно-базованої верифікації програмових

платформних рішень для інформаційно-керуючих систем, а також показано його практичне застосування у двоверсійних архітектурах на базі платформ Radiy та RadICS. На підставі виконаних досліджень і впроваджень, формулюються такі висновки:

1. **Розроблено методологію модельно-базованої верифікації систем програмовної логіки.** Сутність методу полягає у створенні диверсної моделі для еталонного проєкту з використанням середовища ForSyDe (на базі функціональної мови Haskell) та подальшому порівнянні результатів її виконання з еталонною реалізацією (на базі soft-процесора Nios). Такий підхід забезпечує незалежність перевірки та дозволяє виявляти відмінності, недоступні традиційним методам тестування.

2. **Визначено послідовність модельно-базованої верифікації.** Запропоновано формалізований порядок дій: опис інтерфейсів системи на рівні сигналів → побудова альтернативної моделі → формалізація алгоритму роботи → виконання обох версій на однакових наборах вхідних даних → автоматизоване порівняння результатів. Цей підхід довів свою ефективність під час верифікації модулів введення дискретних сигналів.

3. **Імплементовано модель у апаратне середовище.** Показано можливість трансформації опису ForSyDe у VHDL-код та подальшого синтезу у FPGA. У результаті модель існує не лише як симуляційний код, але й як апаратна структура з реальними часовими характеристиками. Порівняння таймінг-діаграм еталонної та модельної реалізацій підтвердило високу точність методу й здатність виявляти часові відхилення.

4. **Отримано експериментальні результати, що підтвердили практичну придатність методу.** У ході випробувань було досягнуто повну відповідність результатів для стандартних сценаріїв, виявлено особливості роботи при навантажувальних випробуваннях та підтверджено здатність моделі локалізувати помилки у випадку штучно створених збоїв. Було сформовано порівняльні таблиці параметрів, які підтверджують високу диверсність обраних реалізацій.

5. **Сформовано загальну схему розроблення двоверсійних систем.** Продемонстровано послідовність дій від формалізації вимог до системної верифікації інтегрованої архітектури з двома незалежними реалізаціями. Запропоновано методи побудови модуля порівняння та визначено критерії функціональної й часової еквівалентності.

6. **Розроблено підхід до комплексування методів програмно-апаратної диверсності.** Показано, що одночасне застосування мовної, інструментальної, апаратної, алгоритмічної та синхронізаційної диверсності забезпечує суттєве зниження ризику відмов за загальною причиною. Систематизовано приклади комбінацій і наведено узагальнену таблицю можливих варіантів побудови двоверсійних систем.

7. **Виконано впровадження результатів у промислових та навчальних проєктах.** У промислових умовах методи були використані для перевірки модулів і алгоритмів платформ Radiy та RadICS, що застосовуються у системах безпеки АЕС. У навчальному процесі створено лабораторні стенди, які дозволяють студентам відпрацьовувати побудову моделей, трансформацію у VHDL та порівняння результатів. Це сприяє формуванню компетентностей з розроблення та верифікації критично важливих систем.

8. **Підтверджено значущість результатів для атомної енергетики.** Використання запропонованих методів у платформах Radiy та RadICS забезпечує підвищений рівень надійності інформаційно-керуючих систем атомних електростанцій, знижує ризик відмов за загальною причиною та спрощує доведення відповідності міжнародним стандартам.

За результатами даного розділу ***вперше запропоновано*** **метод модельно-базованого тестування систем програмовної логіки**, який на відміну від відомих, грунтується на порівнянні версій, отриманих з використанням мови програмування Haskel та її конвертованої моделі у VHDL-код, а також еталонного проєкту, розробленого за технологією SoC, що надає змогу зменшити ризики залишкових прихованих дефектів.

Матеріали розділу опубліковані в [124–128].

ВИСНОВКИ

1. У дисертаційній роботі розв’язано актуальну науково-прикладну задачу – розроблено комплекс моделей, методів і засобів оцінювання та забезпечення функційної безпечності багатоверсійних інформаційно-керуючих систем атомних електростанцій на програмних платформах типу FPGA та SoC з використанням комбінованої диверсності та модельно-базованої верифікації. Дослідження враховує специфіку відмов за загальною причиною, вимоги міжнародних стандартів IEC 61513, IEC 60880, IEC 62138, IEC 61508, рекомендацій МАГАТЕ та національних нормативних документів України.

2. Запропоновано розширену чотирирівневу класифікацію видів і підвидів диверсності для програмовних платформних рішень ІКС АЕС, яка, на відміну від відомих класифікаторів NUREG-6303 та NUREG-7007, включає архітектурний, апаратний, програмно-алгоритмічний та експлуатаційний рівні з понад 60 специфічними підвидами диверсності (диверсна синхронізація, структурно-просторова диверсність LUT/DSP-блоків, диверсифікація CRC-алгоритмів, CDC-схем, IP-ядер). На її базі розроблено графову модель диверсності, де вершини представляють види диверсності, а дуги – ваги метрик диверсності та відносної вартості, а також матричну модель, що формалізує взаємозв’язки рівнів диверсності.

3. Розроблено метод вибору видів диверсності для інформаційно-керуючих систем на програмних платформах, який базується на експертному оцінюванні метрик ефективності диверсності та відносної вартості для розширеного класифікатора, на алгоритмах пошуку оптимальних шляхів і їх комбінацій у графовій моделі (модифікація алгоритму Дейкстри) та на стратегіях оптимізації за критеріями максимізації інтегральної диверсності, мінімізації витрат та компромісному співвідношенні вартості реалізації до отриманої диверсності. Метод забезпечує адаптивний вибір раціональних комбінацій диверсності під конкретні проєктні обмеження.

4. Удосконалено методи реалізації програмно-апаратної диверсності програмних платформ шляхом розроблення методу диверсної синхронізації каналів на основі варіації тактових частот, фазових зсувів та різних схем CDC, методу структурно-просторової диверсності матриць логічних елементів з альтернативним розміщенням/трасуванням та використанням різнорідних IP-ядер, а також підходів до диверсифікації алгоритмів контрольних сум (CRC-16, CRC-32, LFSR), протоколів передачі даних та механізмів перевірки цілісності пам'яті. Експериментально підтверджено зниження ймовірності CCF на 35–47%.

5. Вперше для ІКС АЕС запропоновано метод модельно-базованої верифікації систем програмної логіки, який інтегрує формальне моделювання поведінки (ForSyDe, VHDL), автоматичну генерацію тестів у рамках Model-Based Testing, апаратну імплементацію компонентів MBT (Fault Injection, Coverage Analysis) та багаторівневий аналіз результатів верифікації. Визначено послідовність етапів: побудова моделей, генерація тестових сценаріїв, синтез hardware-компонентів MBT, виконання тестів на FPGA-платформах та кількісна оцінка покриття вимог безпеки.

6. Розроблено узагальнену технологічну схему розроблення та верифікації двоверсійних і багатоверсійних систем безпеки АЕС з комплексним використанням запропонованих моделей диверсності, методів програмно-апаратної диверсифікації та модельно-базованого тестування. Такий підхід забезпечує виявлення комбінаційних помилок на ранніх етапах життєвого циклу, підвищення повноти верифікації до 95%+ та скорочення витрат на доопрацювання систем на 25–30%.

7. Практичні результати дисертаційної роботи доведено до конкретних методичних рекомендацій та інструментальних засобів для підтримки комбінованої диверсності та модельно-базованої верифікації в ІКС АЕС. Зокрема, розроблено:

– методичні рекомендації з вибору оптимальних стратегій диверсності для платформ Radiy та RadICS;

– програмні засоби для автоматизованого аналізу графової моделі диверсності;

– апаратні прототипи двоверсійних модулів з реалізацією диверсних методів синхронізації та контролю цілісності тощо.

8. Результати дисертаційної роботи впроваджено:

– у розробці нормативно-методичного забезпечення для систем сертифікації безпеки АЕС (акт впровадження від 26 січня 2026 р.).

– при виконанні науково-дослідних проєктів з модернізації інформаційно-керуючих систем АЕС на платформах Radiy, RadICS у рамках державних програм (акт впровадження від 15 січня 2026 р.);

– у навчальному процесі Національного аерокосмічного університету «Харківський авіаційний інститут» при підготовці фахівців з комп'ютерної інженерії та кібербезпеки (акт впровадження від 10 березня 2026 р.);

9. Подальші дослідження доцільно зосередити на:

– розробленні методів запобігання поширенню відмов за загальною причиною у вигляді кластерних та ланцюгових відмов у багатоверсійних архітектурах;

– вдосконаленні алгоритмів вибору диверсності шляхом інтеграції з імовірнісним аналізом безпеки (FTA, PSA) та машинним навчанням;

– поглибленому дослідженні впливу радіаційних ефектів (SEU, SET) та кіберзагроз на ефективність запропонованих методів диверсності;

– розробці автоматизованих інструментальних комплексів CAD/CAE для синтезу оптимальних двоверсійних конфігурацій ІКС АЕС на основі графових моделей диверсності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вінтенко Б. Ю., Смірнов О. А., Коваленко О. В., Смірнов С. А., Коваленко А. С. Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки. *Системи управління, навігації та зв'язку*. 2023. № 2. С. 170–178. DOI: 10.26906/SUNZ.2023.2.170.

2. Вінтенко Б., Миронець І., Смірнов О., Кравчук О., Козірова Н. та ін. Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки. *Кібербезпека: освіта, наука, техніка*. 2024. Т. 3, № 23. С. 111–131.

3. Тюрин С. Ф., Каменских А. Н., Харченко В. С. Энергоэффективные вычисления на программируемой логике. *Радіоелектронні і комп'ютерні системи*. 2013. Т. 1 (62). С. 5–12.

4. Щигельська Г., Боднар В. Загрози надзвичайних ситуацій на атомних електростанціях в умовах нового етапу російсько-української війни. *Збірник тез II Міжнародної наукової конференції „Воєнні конфлікти та техногенні катастрофи: історичні та психологічні наслідки"*. 2022. С. 99–101.

5. Babeshko E., Kovalenko A., Illiashenko O., Panarin A., Sklyar V. et al. Secure and resilient computing for industry and human domains. Vol. 2. Kharkiv: National Aerospace University named after N. E. Zhukovsky "KhAI", 2017. 762 p. URL: http://serein.eu.org/wp-content/uploads/2017/10/Volume-2.-Secure-and-resilient-systems-and-infrastructures.pdf

6. Клевцов О., Ястребенецький М., Трубчанінов С. Комп'ютерна безпека інформаційних та керуючих систем АЕС. *Ядерна та радіаційна безпека*. 2015. № 4 (68). С. 51–57. DOI: 10.32918/nrs.2015.4(68).10.

7. Plėta T., Tvaronavičienė M., Della Casa S. Cyber effect and security management aspects in critical energy infrastructures. *Insights into Regional Development*. 2020. Vol. 2, № 2. P. 538–548. DOI: 10.9770/ird.2020.2.2(3).

8. Ястребенецкий М., Розен Ю., Громов Г., Інюшев В., Носовський А. та ін. Вимоги до інформаційних і керуючих систем АЕС за результатами аналізу аварії на АЕС Фукусіма-1. *Ядерна та радіаційна безпека*. 2011. № 4 (52). С. 3–10.

9. Ставченко С. Ядерний тероризм: особливості в умовах російсько-української війни. *Збірка матеріалів І Міжнародної науково-практичної конференції «Радіаційна та ядерна безпека у контексті Української формули миру»*. Дніпро: Видавничо-поліграфічний дім «Формат А+», 2024. С. 56–59. URL: https://www.dnu.dp.ua/docs/2024_Radiacijna%20ta%20yaderna%20bezpeka.pdf (дата звернення: 25.01.2026).

10. Authén S., Holmberg J.-E., Tyrväinen T., Zamani L. Guidelines for reliability analysis of digital systems in PSA context–Final Report. *NKS-330, Nordic nuclear safety research (NKS), Roskilde*. 2015. URL: https://www.nks.org/download/Nyhedsbreve/nks330_e.pdf (дата звернення: 25.01.2026).

11. Plėta T., Karasov S., Jakštas T. The means to secure critical energy infrastructure in the context of hybrid warfare: the case of Ukraine. *Journal of Security & Sustainability Issues*. 2018. Vol. 7, № 3. P. 569–579. DOI: 10.9770/jssi.2018.7.3(16).

12. Plėta T., Tvaronavičienė M., Della Casa S., Agafonov K. Cyber-attacks to critical energy infrastructure and management issues: Overview of selected cases. *Insights into Regional Development*. 2020. Vol. 2, № 3. P. 703–715. DOI: 10.9770/IRD.2020.2.3(7).

13. Вінтенко Б., Смірнов О., Коваленко А., Смірнов С., Буравченко К. Дослідження вимог міжнародних стандартів IEC 60880 та IEC 62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2023. Т. 3, № 73. С. 155–166.

14. Панарін А. Принцип диверсності та багатоверсійні технології для систем захисту реакторів атомних електростанцій. *Пропілеї права та безпеки*. 2026. Vol. 8. P. 260–263. DOI: 10.32620/pls.2025.8.67.

15. Nikolova O. Analysis of the results of proficiency testing schemes of kozloduy npp metrology laboratories through interlaboratory comparison. *Metrology & Instruments*. 2024. № 2. P. 50–55. DOI: 10.30837/2663-9564.2024.2.10.

16. Li Y., Liang G., Zhong K., Chen D., Zhu X. et al. The Research and Design of Self-Diagnostic Function for NPP I&C System. *2024 Global Reliability and Prognostics and Health Management Conference (PHM-Beijing)*. Beijing, China: IEEE, 2024. P. 1–6. DOI: 10.1109/PHM-Beijing63284.2024.10874814.

17. Karmakar G., Nirgude Y. AERB SG D-25 and IEC 60880 for Certification of Software in Safety Systems of Indian NPP. *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013)*. ed. V. S. Chakravarthi., Y. J. M. Shirur., R. Prasad. India: Springer India, 2013. P. 309–316. DOI: 10.1007/978-81-322-1524-0_38.

18. Kostenko S. Удосконалювання нормативної бази з довгострокової експлуатації та управління старінням енергоблоків АЕС. *Ядерна та радіаційна безпека*. 2015. № 1 (65). С. 23–25.

19. Shugailo O., Ryzhov D., Kritsky V., Romanov S., Kolupaev A. Рекомендації щодо удосконалення національної нормативної бази в частині продовження строку експлуатації та управління старінням енергоблоків АЕС України. *Ядерна та радіаційна безпека*. 2013. № 3 (59). С. 3–9.

20. Kukhotskyi O., Gumeniuk D., Ligotskyy O., Potoskuiev O., Shyshuta A. та ін. Вимоги до технічного обслуговування і ремонту обладнання систем, важливих для безпеки атомних станцій. *Ядерна та радіаційна безпека*. 2024. № 3 (103). С. 52–59.

21. Bogorad V., Litvinska T., Slepchenko O., Nosovsky A. Обґрунтування розмірів зони спостереження АЕС. *Ядерна та радіаційна безпека*. 2013. № 2 (58). С. 39–42.

22. Childs J. A., Mosleh A. A modified FMEA tool for use in identifying and addressing common cause failure risks in industry. *Annual Reliability and Maintainability*. Washington, DC, USA, 1999. P. 19–24. DOI: 10.1109/RAMS.1999.744090.

23. Pun K. P., Rotanson J., Cheung C.-W., Chan A. H. Application of fuzzy integrated FMEA with product lifetime consideration for new product development in flexible electronics industry. *Journal of Industrial Engineering and Management*. 2019. Vol. 12, № 1. P. 176–200. DOI: 10.3926/jiem.2765.

24. Jaise J., Ajay Kumar N. B., Shanmugam N. S., Sankaranarayanasamy K., Ramesh T. Power system: a reliability assessment using FTA. *International Journal of System Assurance Engineering and Management*. 2013. Vol. 4, № 1. P. 78–85. DOI: 10.1007/s13198-012-0100-2.

25. Kharchenko V. Cybersecurity of Critical Infrastructures at hard test – lessons learned from 2022-2025 war in Ukraine. *Cybersecurity of Critical Infrastructures at hard test – lessons learned from 2022-2025 war in Ukraine*. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi State University, 2025.

26. Kharchenko V., Yastrebenetsky M. Comparativistics for Big Safety: Hegel's Dialectic-based Methodology and Critical Applications Experience. *2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. Athens, Greece: IEEE (Institute of Electrical and Electronics Engineers), 2023. P. 271–278. DOI: 10.1109/DESSERT61349.2023.10416527.

27. Wood R., Belles R., Cetiner S., Holcomb D., Korash K. et al. Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems. Nuclear Regulatory Commission, 2010. DOI: 10.2172/1000417.

28. Illiashenko O., Kharchenko V., Kor A.-L., Panarin A., Sklyar V. Hardware diversity and modified NUREG/CR-7007 based assessment of NPP I&C safety. *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Bucharest, Romania: IEEE, 2017. Vol. 2. P. 907–911. DOI: 10.1109/IDAACS.2017.8095218.

29. Kharchenko V., Kovalenko A., Leontiiev K., Panarin A., Duzhy V. Multi-diversity for FPGA platform based NPP I&C systems: new possibilities and assessment technique. *International Conference on Nuclear Engineering*. London, UK: American Society of Mechanical Engineers, 2018. Vol. 1. P. 1–9. DOI: 10.1115/ICONE26-82377.

30. Kharchenko V., Ponochovnyi Y., Babeshko E., Ruchkov E., Panarin A. Safety Assessment of Maintained Control Systems with Cascade Two-Version 2oo3/1oo2 Structures Considering Version Faults. *18th DepCoS-RELCOMEX*. 2023. Vol. 737. P. 119–129. DOI: 10.1007/978-3-031-37720-4_11.

31. Поночовний Ю. Л., Харченко В. С. Методологія забезпечення гарантоздатності інформаційно-керуючих систем з використанням багатоцільових стратегій обслуговування. *Моделювання і розроблення гарантоздатних систем*. 2020. Т. 3. С. 43–58. DOI: 10.32620/reks.2020.3.05.

32. Сиора А. А., Харченко В. С. Модели многоверсионных вычислений и их обобщение для отказоустойчивых систем. *Вісник Харківського національного університету імені ВН Каразіна. Серія: Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2010. № 13. С. 205–217.

33. Kharchenko V., Illiashenko O., Sklyar V. Invariant-Based Safety Assessment of FPGA Projects: Conception and Technique. *Computers*. 2021. Vol. 10. P. 1–19. DOI: 10.3390/computers10100125.

34. Aalsaud A., Xia F., Rafiev A., Shafik R., Romanovsky A. et al. Low-Complexity Run-time Management of Concurrent Workloads for Energy-Efficient Multi-Core Systems. *Journal of Low Power Electronics and Applications*. 2020. Vol. 10, № 3. P. 1–25. DOI: 10.3390/jlpea10030025.

35. Al-Hayanni M. A. N., Rafiev A., Xia F., Shafik R., Romanovsky A. et al. PARMA: Parallelization-aware run-time management for energy-efficient many-core systems. *IEEE Transactions on Computers*. 2020. Vol. 69, № 10. P. 1507–1518. DOI: 10.1109/TC.2020.2975787.

36. Панарін А., Харченко В., Землянко Г. Розроблення розширеного класифікатору для експертного оцінювання диверсності та вартості інформаційно-керуючих систем АЕС. *Вимірювальна техніка та метрологія*. 2025. Т. 86, № 4. С. 52. DOI: 10.23939/istcmtm2025.04.052.

37. Panarin A., Kharchenko V., Zemlianko H. Advanced classifier for expert assessment of diversity and cost of NPP information and control systems. *Measuring Equipment and Metrology*. 2025. Vol. 86, № 4. P. 52–64.

38. Дужий В. И., Шостак А. В., Дужий И. В. Оценка показателя версионной избыточности при проектировании программного обеспечения. *Радіоелектронні і комп'ютерні системи*. 2009. № 6. С. 159–165.

39. Tang E., Li S., Chen R., Zhou H., Ma Y. et al. Graph-OPU: A Highly Flexible FPGA-Based Overlay Processor for Graph Neural Networks. *ACM Transactions on Reconfigurable Technology and Systems*. 2024. Vol. 17, № 4. P. 1–33. DOI: 10.1145/3691636.

40. Chellaswamy C., Manjula M. M., Ramasubramanian B., Sriram A. FPGA-based remote target classification in hyperspectral imaging using multi-graph neural network. *Microprocessors and Microsystems*. 2024. Vol. 105. P. 1–8. DOI: 10.1016/j.micpro.2024.105008.

41. Favaro F., Dufrechou E., Oliver J. P., Ezzatti P. Optimizing the performance of the sparse matrix–vector multiplication kernel in FPGA guided by the roofline model. *Micromachines*. 2023. Vol. 14, № 11. P. 1–17. DOI: 10.3390/mi14112030.

42. Koul R., Yadav M., Suneja K. Comparative analysis of FPGA-Based Hardware Design of dynamic time warping algorithm using different multiplier architectures. *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*. Greater Noida, India: IEEE, 2020. P. 599–603. DOI: 10.1109/GUCON48875.2020.9231244.

43. Naji H. R., Shadravan S., Jafarabadi H. M., Momeni H. Parallel design of SFO optimization algorithm based on FPGA. *The Journal of Supercomputing*. 2024. Vol. 80, № 8. P. 10796–10817. DOI: 10.1007/s11227-023-05851-7.

44. Karam R., Hoque T., Ray S., Tehranipoor M., Bhunia S. MUTARCH: Architectural diversity for FPGA device and IP security. *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017. P. 611–616. DOI: 10.1109/ASP-DAC.2017.7858391.

45. Kharchenko V., Illiashenko O. Diversity for security: case assessment for FPGA-based safety-critical systems. *MATEC Web of Conferences*. Corfu Island, Greece: EDP Sciences, 2016. Vol. 76. P. 1–9. DOI: 10.1051/matecconf/20167602051.

46. Бойко К., Гальцева І., Малахов С. Програмна диверсність як інструмент забезпечення функціональної безпеки сучасних інформаційних систем. *Collection of scientific papers «ΛΟΓΟΣ»*. 2024. С. 170–178.

47. Лютенко І. В., Лілікович С. О. Підхід та засоби вимірювання суб'єктної диверсності в умовах багатоверсійної розробки програмного забезпечення. *Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології*. 2016. № 45. С. 55–58.

48. Харченко В. С. Гарантоздатні системи та багатоверсійні обчислення: аспекти еволюції. *Радіоелектронні і комп'ютерні системи*. 2009. № 7. С. 46–59.

49. Харченко В. С., Опанасенко В. М., Можаєв О. О. Методи і засоби забезпечення виконання вимог до кібербезпеки систем на програмовній логіці. *Системи обробки інформації*. 2012. Т. 4 (28). С. 275–285.

50. Харченко В. С., Ястребенецький М. О. Безпека АЕС і велика безпека за часів коронавірусу. *Ядерна та радіаційна безпека*. 2020. № 3. С. 74–87.

51. Babeshko I., Duzhiy V., Panarin A., Illiashenko O., Siora A. et al. Diversity for NPP I&C Systems Safety and Cyber Security. *Cyber Security and Safety of Nuclear Power Plant Instrumentation and Control Systems*. Hershey PA, USA: IGI Global, 2020. P. 239–288. URL: https://www.igi-global.com/chapter/npp-ic-systems/258682 (дата звернення: 25.01.2026).

52. Kharchenko V., Babeshko E., Leontiiev K., Duzhy V. Diversity for safety and security of NPP I&C: post NUREG/CR 7007 stage. *10th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies, NPIC and HMIT 2017*. 2017. P. 1528–1536. URL: https://elibrary.ru/item.asp?id=35747452 (дата звернення: 25.01.2026).

53. Reşat M. A., Çiçek A., Özyurt S., Çavuş E. Analysis and FPGA Implementation of Zero-Forcing Receive Beamforming with Signal Space Diversity

under Different Interleaving Techniques. *Journal of Circuits, Systems and Computers*. 2020. Vol. 29, № 01. P. 1–7. DOI: 10.1142/S0218126620500073.

54. Sun Y.-B., Zhang C.-L., Chen Y.-J., Bai T. The R&D of FPGA-Based FitRel Platform in Nuclear Power Plant Diversity Actuation System. *Nuclear Power Plants: Innovative Technologies for Instrumentation and Control Systems*. ed. Y. Xu., F. Gao., W. Chen. et al. Singapore: Springer Singapore, 2018. P. 188–195. DOI: 10.1007/978-981-10-7416-5_23.

55. Xu Z., Guo S., Li X., Wang Z., Jiang H. SIMTAM: Generation Diversity Test Programs for FPGA Simulation Tools Testing Via Timing Area Mutation. *ACM Transactions on Design Automation of Electronic Systems*. 2025. Vol. 30, № 2. P. 1–25. DOI: 10.1145/3705730.

56. Aadit N. A., Mohseni M., Camsari K. Y. Accelerating Adaptive Parallel Tempering with FPGA-based p-bits. *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2023. P. 1–2. DOI: 10.23919/VLSITechnologyandCir57934.2023.10185207.

57. Kou X., Li D., Wang D., Zhang B. Parallel implementation by the FPGA of phase diversity based on an improved particle swarm optimization algorithm. *Applied Optics*. 2024. Vol. 64, № 1. P. 30–39. DOI: 10.1364/AO.542241.

58. Shirakura Y., Segawa T., Shibata Y., Morimoto K., Tanaka M. et al. A Redundant Design Approach with Diversity of FPGA Resource Mapping. *Applied Reconfigurable Computing*. ed. V. Bonato., C. Bouganis., M. Gorgon. Cham: Springer International Publishing, 2016. P. 119–131. DOI: 10.1007/978-3-319-30481-6_10.

59. Duzhyi V., Kharchenko V., Panarin A., Rusin D. Diversity metric evaluation considering extended NUREG-7007 diversity classification. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. Kyiv, Ukraine: IEEE, 2018. P. 21–25. DOI: 10.1109/DESSERT.2018.8409092.

60. Jung S., Yoo J., Lee Y.-J. A practical application of NUREG/CR-6430 software safety hazard analysis to FPGA software. *Reliability Engineering & System Safety*. 2020. Vol. 202. P. 1–11. DOI: 10.1016/j.ress.2020.107029.

61. Maerani R., Jung J. C. VHDL verification of FPGA based ESF-CCS for nuclear power plant I&C system. *ISOFIC 2017*. Gyeongju, Korea: Korean Nuclear Society (KNS), 2017. P. 1–6. URL: https://www.kns.org/files/int_paper/paper/ISOFIC_2017_10/ISOFIC-2017-Restu-MAERANI.pdf (дата звернення: 25.01.2026).

62. Perepelitsyn A., Illiashenko O., Duzhyi V., Kharchenko V. Application of the FPGA Technology for the Development of Multi-Version Safety-Critical NPP Instrumentation and Control Systems. *Nuclear and Radiation Safety*. 2020. № 2 (86). P. 52–61.

63. Pradana S., Jung J. Software reliability growth model for FPGA-based safety critical software system. *Transactions of the Korean Nuclear Society Spring Meeting Jeju, Korea*. 2019. URL: https://www.kns.org/files/pre_paper/41/19S-376-Satrio.pdf (дата звернення: 25.01.2026).

64. Дужий В. И. Информационная технология поддержки оценивания многоверсионных систем. *Збірник наукових праць Харківського університету Повітряних сил*. 2013. № 4. С. 108–113.

65. Sklyar V., Siora O., Herasimenko O., Panarin A. Development and Verification of Dependable Multiversion Systems on the basis of IP-Cores. *Technical Approach to Dependability*. Wroclaw, Poland, 2010. P. 133–145. URL: https://www.researchgate.net/profile/Vladimir-Sklyar/publication/305773007_Development_and_Verification_of_Dependable_Multiversion_Systems_on_the_basis_of_IP-Cores/links/57a0b39c08aeef8f311c52f5/Development-and-Verification-of-Dependable-Multiversion-Systems-on-the-basis-of-IP-Cores.pdf (дата звернення: 25.01.2026).

66. Ручков Є., Харченко В., Коваленко А., Бабешко Є., Порошенко А. Оцінювання безвідмовності резервованих структур «2-р-3» і «1-р-2» з урахуванням засобів оброблення інформації та комуніказій. *Advanced Information Systems*. 2020. Т. 4, № 4. С. 77–83. DOI: 10.20998/2522-9052.2020.4.11.

67. Du Y., Gu N., Cao H. Detecting Deadlocks Involving Diverse Synchronization Mechanisms Using Extended Petri Nets. *Arabian Journal for Science and Engineering*. 2017. Vol. 42, № 2. P. 913–923. DOI: 10.1007/s13369-016-2367-0.

68. Davari M., Gao W., Aghazadeh A., Blaabjerg F., Lewis F. L. An optimal synchronization control method of PLL utilizing adaptive dynamic programming to synchronize inverter-based resources with unbalanced, low-inertia, and very weak grids. *IEEE Transactions on Automation Science and Engineering*. 2024. Vol. 22. P. 24–42. DOI: 10.1109/TASE.2023.3329479.

69. Khediri S. el, Kachouri A., Nejah N. Diverse Synchronization issues in Wireless Sensor Networks. *Proceedings of the International Conference on Microelectronics, ICM*. Hammamet, Tunisia, 2011. P. 1–6. DOI: 10.1109/ICM.2011.6177374.

70. Бабешко Є., Панарін А. Метод та засоби диверсної синхронізації та реалізації електронних проєктів для систем безпеки АЕС на FPGA платформах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2026. Т. 1. С. 32–38. DOI: 10.31891/2219-9365-2026-85-5.

71. Peng Q., Bian J., Huang Z., Wang S., Yan A. A Compact TRNG Design for FPGA Based on the Metastability of RO-driven Shift Registers. *ACM Transactions on Design Automation of Electronic Systems*. 2024. Vol. 29, № 1. P. 1–17. DOI: 10.1145/3610295.

72. Capligins F., Litvinenko A., Kolosovs D., Terauds M., Zeltins M. et al. FPGA-based antipodal chaotic shift keying communication system. *Electronics*. 2022. Vol. 11, № 12. P. 1–23. DOI: 10.3390/electronics11121870.

73. Chen P., Lan J.-T., Wang R.-T., Qui N. M., Marquez J. C. J. S. et al. High-precision PLL delay matrix with overclocking and double data rate for accurate FPGA time-to-digital converters. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2020. Vol. 28, № 4. P. 904–913. DOI: 10.1109/TVLSI.2019.2962606.

74. Chen X., Huang Z., Yin X., Liang Z. Control damping enhancement method of grid-forming battery energy storage system with diverse synchronization

controls in asymmetrical grids. *Journal of Energy Storage*. 2025. Vol. 133. P. 1–15. DOI: 10.1016/j.est.2025.118028.

75. Plusquellic J. Shift Register, Reconvergent-Fanout (SiRF) PUF Implementation on an FPGA. *Cryptography*. 2022. Vol. 6, № 4. P. 1–23. DOI: 10.3390/cryptography6040034.

76. Zhang H., Yang H., Wang Y., Liu F., Gao T. A 270-MHz to 1.5-GHz CMOS PLL clock generator with reconfigurable multi-functions for FPGA. *Journal of Semiconductors*. 2011. Vol. 32, № 4. P. 58–64. DOI: 10.1088/1674-4926/32/4/045010.

77. Ghiasi A., Zahedi A., Haghiri S. Linear fragmentation Morris–Lecar realization using new exponential module instead of hyperbolic function in FPGA implementation. *Journal of Ambient Intelligence and Humanized Computing*. 2023. Vol. 14, № 4. P. 4355–4370. DOI: 10.1007/s12652-023-04546-4.

78. Septién J., Mozos D., Mecha H., Tabero J., Dios M. G. de Perimeter quadrature-based metric for estimating FPGA fragmentation in 2D HW multitasking. *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE, 2008. P. 1–8. DOI: 10.1109/IPDPS.2008.4536508.

79. Wang S., Pham N. K., Singh A. K., Kumar A. Leakage and performance aware resource management for 2D dynamically reconfigurable FPGA architectures. *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. Munich, Germany: IEEE, 2014. P. 1–4. DOI: 10.1109/FPL.2014.6927420.

80. Zhu Y., Wang L. Design and Implementation of Nanosecond Pulse Generator Based on Reconfiguration PLL in FPGA. *2015 International Conference on Electronic Science and Automation Control*. Atlantis Press, 2015. P. 323–326. URL: https://www.atlantis-press.com/article/25836918.pdf (дата звернення: 25.01.2026).

81. Gericota M., Alves G., Lemos L., Ferreira J. M., Almeida B. et al. Assessing Defragmentation Strategies for FPGAs. Porto, Portugal: Instituto Superior de Engenharia do Porto (ISEP), 2006. P. 99–102. URL: https://recipp.ipp.pt/entities/publication/f49d3a53-3985-4bb4-9e15-1c43ee8dd988

82. Liu X., Kim D. H., Wu C., Chen D. Resource and data optimization for hardware implementation of deep neural networks targeting FPGA-based edge devices.

Proceedings of the 20th System Level Interconnect Prediction Workshop. San Francisco, California: ACM, 2018. P. 1–8. DOI: 10.1145/3225209.3225214.

83. Mettler M., Rapp M., Khdr H., Mueller-Gritschneder D., Henkel J. et al. An FPGA-based Approach to Evaluate Thermal and Resource Management Strategies of Many-core Processors. *ACM Transactions on Architecture and Code Optimization*. 2022. Vol. 19, № 3. P. 1–24. DOI: 10.1145/3516825.

84. Wang J., Loo S. M. Case study of finite resource optimization in FPGA using genetic algorithm. *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*. Shanghai China: ACM, 2009. P. 989–992. DOI: 10.1145/1543834.1543990.

85. Баркалов О. О., Тітаренко Л. О., Сабурова С. О., Головін О. М., Матвієнко О. В. Оптимізація схеми композиційного мікропрограмного пристрою управління з базовою архітектурою. *Кібернетика та комп'ютерні технології*. 2024. № 4. С. 121–133.

86. Грушко С. С., Зеленьова І. Я., Тіменко А. В., Голуб Т. В. Способи використання внутрішніх ресурсів FPGA і PROASIC для підвищення надійності вбудованого модуля арбітражу. *Електротехнічні та комп'ютерні системи*. 2021. № 35 (111). С. 54–62. DOI: 10.15276/eltecs.35.111.2021.7.

87. Cong J., Liu B., Neuendorffer S., Noguera J., Vissers K. et al. High-level synthesis for FPGAs: From prototyping to deployment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2011. Vol. 30, № 4. P. 473–491. DOI: 10.1109/TCAD.2011.2110592.

88. Hempel G., Hochberger C. A resource optimized processor core for FPGA based SoCs. *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*. IEEE, 2007. P. 51–58. DOI: 10.1109/DSD.2007.4341449.

89. Nangia R., Shukla N. K. Resource utilization optimization with design alternatives in FPGA based arithmetic logic unit architectures. *Procedia Computer Science*. 2018. Vol. 132. P. 843–848.

90. Vynnyshyn O. Комплексна методика оптимізації програмного коду VHDL при проектуванні програмованої інтегральної схеми. *Computer-integrated technologies: education, science, production*. 2024. № 55. С. 48–54.

91. Abdulnabi M. A., Ahmed H. High Speed Low Power Cyclic Redundancy Check-32 using FPGA. *Journal of Engineering and Computer Science (JECS)*. 2019. Vol. 19, № 2. P. 50–56.

92. Abdulnabi M. S., Ahmed H. Design of efficient cyclic redundancy check-32 using FPGA. *2018 International conference on computer, control, electrical, and electronics engineering (ICCCEEE)*. IEEE, 2018. P. 1–5. DOI: 10.1109/ICCCEEE.2018.8515877.

93. Závodník T., Kekely L., Puš V. CRC based hashing in FPGA using DSP blocks. *17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*. IEEE, 2014. P. 179–182. DOI: 10.1109/DDECS.2014.6868786.

94. Zhang L., Ye S., Gou Z., Yang X., Dai Q. et al. An Efficient Parallel CRC Computing Method for High Bandwidth Networks and FPGA Implementation. *Electronics*. 2024. Vol. 13, № 22. P. 1–17. DOI: 10.3390/electronics13224399.

95. Zitlaw C., Sayeed A., Lim S. L. CRC Integration for Enhanced SPI Communication Reliability in Digital Systems. *2024 Multimedia University Engineering Conference (MECON)*. IEEE, 2024. P. 1–5. DOI: 10.1109/MECON62796.2024.10776358.

96. Gu Y., Fan M., Tang C., Zhang G., Zhang X. Progress in Data Transmission and Storage for Long Pulse Data Acquisition System. *Journal of Physics: Conference Series*. IOP Publishing, 2021. Vol. 2113, № 1. P. 1–6. DOI: 10.1088/1742-6596/2113/1/012065.

97. Escobar J.-H. M., SachBe J., Ostendorff S., Wuttke H.-D. Automatic generation of an FPGA based embedded test system for printed circuit board testing. *2012 13th Latin American Test Workshop (LATW)*. IEEE, 2012. P. 1–6. DOI: 10.1109/LATW.2012.6261241.

98. Navaneeth K., Kannan R. FPGA-Based Defect Detection in PCBs Using Image Processing. *2025 8th International Conference on Circuit, Power & Computing*

Technologies (ICCPCT). IEEE, 2025. P. 853–858. DOI: 10.1109/ICCPCT65132.2025.11176727.

99. Pan Y., Zhang L., Zhang Y. Rapid detection of PCB defects based on YOLOx-Plus and FPGA. *IEEE Access*. 2024. Vol. 12. P. 61343–61358. DOI: 10.1109/ACCESS.2024.3387947.

100. Сєлюков О. В., Хмельницький Ю. В., Яковлєв Д. П. Забезпечення достовірності передачі інформації та сервісних послуг для високошвидкісних мереж при завадах. *Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка*. 2017. № 57. С. 168–178.

101. Хмельницький Ю. В., Каблуков О. А., Ряба Л. О., Солодєєва Л. В., Ткач А. О. Методи та засоби забезпечення завадостійкої передачі інформації в телекомунікаційних мережах. *Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка*. 2019. № 64. С. 133–143.

102. KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide. Dallas, Texas: Texas Instruments, 2010. URL: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf (дата звернення: 16.11.2025).

103. Gupta A., Gupta A. UART Communication. *The IoT Hacker's Handbook*. Berkeley, CA: Apress, 2019. P. 59–80. DOI: 10.1007/978-1-4842-4300-8_4.

104. Dhanadravye A. D., Thorat S. S. A review on implementation of UART using different techniques. *International Journal of Computer Science and Information Technologies*. 2014. Vol. 5, № 1. P. 394–396.

105. Fang Y., Chen X. Design and simulation of UART serial communication module based on VHDL. *2011 3rd International Workshop on Intelligent Systems and Applications*. Wuhan, China: IEEE, 2011. P. 1–4. DOI: 10.1109/ISA.2011.5873448.

106. Harutyunyan S., Kaplanyan T., Kirakosyan A., Momjyan A. Design and verification of autoconfigurable UART controller. *2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO)*. IEEE, 2020. P. 347–350. DOI: 10.1109/ELNANO50318.2020.9088789.

107. Laddha N. R., Thakare A. P. A review on serial communication by UART. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2013. Vol. 3, № 1. URL: https://www.academia.edu/download/35186800/V3I1-0220.pdf (дата звернення: 25.01.2026).

108. Nanda U., Pattnaik S. K. Universal Asynchronous Receiver and Transmitter (UART). *3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*. Tamil Nadu, India: IEEE, 2016. Vol. 1. P. 1–5. DOI: 10.1109/ICACCS.2016.7586376.

109. Norhuzaimin J., Maimun H. H. The design of high speed UART. *2005 Asia-Pacific Conference on Applied Electromagnetics*. IEEE, 2005. P. 5–9. DOI: 10.1109/APACE.2005.1607831.

110. Wang Y., Song K. A new approach to realize UART. *Proceedings of 2011 international conference on Electronic & Mechanical Engineering and information technology*. IEEE, 2011. Vol. 5. P. 2749–2752. DOI: 10.1109/EMEIT.2011.6023602.

111. Agrawal R. K., Mishra V. R. The design of high speed UART. *2013 IEEE Conference on Information & Communication Technologies*. IEEE, 2013. P. 388–390. DOI: 10.1109/CICT.2013.6558126.

112. Yuan H., Yang J., Pan P. Optimized design of UART IP soft core based on DMA mode. *2010 5th IEEE Conference on Industrial Electronics and Applications*. IEEE, 2010. P. 1907–1910. DOI: 10.1109/ICIEA.2010.5515473.

113. Zang F., Niu H. Design and Implementation of High Speed UART Based on DMA. *2018 3rd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2018)*. Atlantis Press, 2018. P. 128–131. URL: https://www.atlantis-press.com/proceedings/eame-18/25899263 (дата звернення: 25.01.2026).

114. User Guide SPI. Dallas, Texas: Texas Instruments, 2012. URL: https://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf (дата звернення: 16.11.2025).

115. Dhaker P. Introduction to SPI interface. *Analog Dialogue*. 2018. Vol. 52, № 3. P. 49–53.

116. Leens F. An introduction to I2C and SPI protocols. *IEEE Instrumentation & Measurement Magazine*. 2009. Vol. 12, № 1. P. 8–13. DOI: 10.1109/MIM.2009.4762946.

117. Intel® Arria® 10 Transceiver PHY User Guide. Santa Clara, CA, USA: Intel Corporation, 2023. URL: https://www.mouser.lt/pdfDocs/ug_arria10_xcvr_phy-683617-6668051.pdf (дата звернення: 25.01.2026).

118. Alachiotis N., Berger S. A., Stamatakis A. Efficient pc-fpga communication over gigabit ethernet. *2010 10th IEEE International Conference on Computer and Information Technology*. IEEE, 2010. P. 1727–1734. DOI: 10.1109/CIT.2010.302.

119. Dick C., Harris F. FPGA implementation of an OFDM PHY. *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. IEEE, 2003. Vol. 1. P. 905–909. DOI: 10.1109/ACSSC.2003.1292045.

120. Huang X.-R., Cao P., Zheng J.-J. A data transmission method for particle physics experiments based on Ethernet physical layer. *Chinese Physics C*. 2015. Vol. 39, № 11. P. 1–5. DOI: 10.1088/1674-1137/39/11/116102.

121. Manjule H., Upadhya S., Wankhede N., Kumbhare M., Thakur K. et al. Ethernet implementation on FPGA. *2020 International Conference for Emerging Technology (INCET)*. IEEE, 2020. P. 1–4. DOI: 10.1109/INCET49848.2020.9154046.

122. Pahlevi R. R., Abdurohman M. Fast UART and SPI protocol for scalable IoT platform. *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2018. P. 239–244. DOI: 10.1109/ICoICT.2018.8528745.

123. Poorani M., Kurunjimalar R. Design implementation of UART and SPI in single FPGA. *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. IEEE, 2016. P. 1–5. DOI: 10.1109/ISCO.2016.7726983.

124. Панарин А. С. Имитационное моделирование soft-процессоров на базе концепции Model-Based Testing. *Радіоелектронні і комп'ютерні системи*. 2012. № 5. С. 100–106.

125. Скляр В. В., Харченко В. С., Панарин А. С., Сандер И. Применение концепции Model-Based Testing для верификации систем на базе IP-ядер. *Радіоелектронні і комп'ютерні системи*. 2010. № 5. С. 237–241.

126. Скляр В. В., Харченко В. С., Панарин А. С. Тестирование и разработка диверсных программируемых логических контроллеров на базе ПЛИС с использованием среды функционального программирования. *Радіоелектронні і комп'ютерні системи*. 2014. № 1 (117). С. 29–41.

127. Скляр В. В., Харченко В. С., Панарин А. С. Тестирование программируемых логических контроллеров на базе ПЛИС с использованием среды функционального программирования. *Системи обробки інформації*. 2014. № 1. С. 44–55.

128. Panarin A., Sklyar V., Kharchenko V., Kovalenko A., Babeshko E. Modeling of industrial FPGA-based controllers with ForSyDe. *2017 International Conference on Information and Digital Technologies (IDT)*. IEEE, 2017. P. 164–168. DOI: 10.1109/DT.2017.8024290.

129. Raudvere T., Sander I., Singh A. K., Jantsch A. Verification of design decisions in ForSyDe. *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign & system synthesis - CODES+ISSS '03*. Newport Beach, CA, USA: ACM Press, 2003. P. 176–181. DOI: 10.1145/944691.944692.

130. Acosta A. ForSyDe tutorial. Stockholm, Sweden: KTH, 2008. URL: https://hackage.haskell.org/package/ForSyDe-3.1.1/src/doc/www/files/tutorial/tutorial.pdf (дата звернення: 25.01.2026).

131. Sander I., Jantsch A. System modeling and transformational design refinement in ForSyDe [formal system design]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2004. Vol. 23, № 1. P. 17–32. DOI: 10.1109/TCAD.2003.819898.

132. Sander I. System modeling and design refinement in ForSyDe: PhD Thesis. Stockholm: Mikroelektronik och informationsteknik, 2003. URL: https://www.diva-portal.org/smash/get/diva2:9340/FULLTEXT01.pdf (дата звернення: 25.01.2026).

133. Woidt H. Hardware Synthesis in ForSyDe. Stockholm: KTH Royal Institute of Technology, 2016. 79 p. URL: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1051461 (дата звернення: 25.01.2026).

134. Sicking J. Implementation of asynchronous communication for ForSyDe in hardware and software. *Royal Institute of Technology*. 2005. URL: https://people.kth.se/~ingo/MasterThesis/JonasSicking.pdf (дата звернення: 25.01.2026).

135. Bahrami F., Sander I. Automating Transformation Strategy via Attributed Graphs for Process Network Parallelization. Turin, Italy: IEEE, 2025. P. 1–10. DOI: 10.1109/FDL68117.2025.11165274.

136. Lu Z., Sander I., Jantsch A. A case study of hardware and software synthesis in ForSyDe. *Proceedings of the 15th international symposium on System Synthesis - ISSS '02*. Kyoto, Japan: ACM Press, 2002. P. 86. DOI: 10.1145/581199.581219.

137. Acosta Gómez A. Hardware synthesis in ForSyDe: The design and implementation of a ForSyDe-to-VHDL Haskell-embedded compiler. 2007. URL: http://oa.upm.es/1162 (дата звернення: 25.01.2026).

138. Sander I., Acosta A., Jantsch A. Hardware design and synthesis in ForSyDe. *Workshop on Hardware Design Using Functional Languages (HFL 09)*. 2009. URL: https://people.kth.se/~ingo/presentations/handouts_hfl2009.pdf (дата звернення: 25.01.2026).

ДОДАТОК А
АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

ЗАТВЕРДЖУЮ
Генеральний конструктор
конструкторського бюро
автоматизованих систем
управління технологічними
процесами ПАТ "НВП "Радій"
Віктор ТИКАРЄВ
"16" 2023 року

АКТ ВПРОВАДЖЕННЯ

наукових результатів дисертаційної роботи Панаріна Артема Сергійовича, виконаної на здобуття наукового ступеня доктора філософії, у проектах ПАТ "НВП "Радій"

Комісія у складі: голови комісії – директора технічного ПАТ "НВП "Радій" Костянтина ЛЕОНТЄВА, і членів – начальника конструкторського відділу КБ АСУ ТП ПАТ "НВП "Радій", к.т.н. Віктора ГОЛОВІРА, начальника відділу програмування КБ АСУ ТП ПАТ НВП "Радій", к.т.н. Олександра БЄЛОГО, встановила, що у ПАТ "НВП "Радій" використано наукові результати дисертаційної роботи, а саме:

- модель для опису різних програмно-апаратних версій для інформаційно-керуючих систем на програмовних платформах, включно з диверсністю процесорних ядер, інтерфейсів, засобів синхронізації, технологічних процесів і обладнання, що дозволяє оцінювати інтегральні метрики диверсності і складності залежно від видів і підвидів версійної надмірності;

- метод вибору видів диверсності з урахуванням багаторівневої ієрархії версійної надмірності, включно з внутрішньою диверсністю каналів, а також відповідні метрики диверсності і складності, що дозволяє підвищити функційну безпечність і обґрунтувати структуру та процеси створення двохверсійних систем аварійного захисту реакторів АЕС;

- методи програмно-апаратної диверсності, які базуються на процедурах диверсних засобів шинної організації, контролі цілісности даних, налаштуваннях проекту та диверсній синхронізації, які дозволяють підвищити

енергоефективність та/або безпечність за рахунок зменшення ризиків завад при перемиканнях елементів;

- метод модельно-базованого тестування систем програмовної логіки, який на відміну від відомих, базується на порівнянні версій, отриманих з використанням мови програмування Haskel та її конвертованої моделі у VHDL-код, а також еталонного проєкту, розробленого за технологією SoC, що надає змогу зменшити ризики залишкових прихованих дефектів.

Застосування зазначених результатів забезпечує:

- вибір видів і підвидів диверсності з урахуванням багаторівневої ієрархії версійної надмірності, включно з внутрішньою диверсністю каналів;

- кількісне обгрунтування структури двохверсійних систем аварійного захисту реакторів АЕС через узгоджене застосування метрик диверсності та складності;

- підвищення функційної безпечності шляхом системного зниження ризиків відмов за загальною причиною за допомогою застосування комбінацій диверсності;

- стандартизоване включення МВТ-процедур у життєвий цикл розроблення FPGA/SoC-проєктів.

Результати досліджень були впроваджені при розробленні та постачанні платформних рішень для інформаційно-керуючих систем енергоблоків Запорізької, Рівненської, Хмельницької та Південноукраїнської атомних електростанцій України, зокрема в системах аварійного захисту та суміжних системах, реалізованих на програмовних платформах ПАТ "НВП "Радій".

Голова комісії — Костянтин ЛЄОНТЄВ

Члени комісії — Віктор ГОЛОВІР

Юрій БЄЛИЙ

ЗАТВЕРДЖУЮ
Генеральний директор
ТОВ "НВП "РАДІКС"
Катерина БАХМАЧ
2026 року

АКТ ВПРОВАДЖЕННЯ

наукових результатів дисертаційної роботи Панаріна Артема Сергійовича, виконаної на здобуття наукового ступеня доктора філософії, у науково-дослідних проектах та розробках ТОВ "НВП "РАДІКС"

Комісія у складі: голови комісії – операційного директора ТОВ "НВП "РАДІКС", к.т.н. Олександра ІВАСЮКА, і членів - заступника операційного директора ТОВ "НВП "РАДІКС", к.т.н. Наталії РВАЧОВОЇ, провідного наукового співробітника ТОВ "НВП "РАДІКС", к.т.н Вікторії КУЛІНІЧ встановила, що у ТОВ "НВП "РАДІКС" використано наукові результати дисертаційної роботи, а саме:

- графова модель для опису можливості отримання різних програмно-апаратних версій для ІКС на програмовних платформах, з урахуванням деталізованого класифікатора видів диверсності (процесорних ядер, інтерфейсів, засобів синхронізації, технологічних процесів і обладнання), що дозволяє оцінювати інтегральні метрики диверсності і складності;

- метод вибору видів диверсності, що дозволяє підвищити функційну безпечність і обгрунтувати структуру та процеси створення систем аварійного захисту реакторів АЕС;

- метод модельно-базованого тестування систем програмовної логіки, який на відміну від відомих, базується на порівнянні версій, отриманих з використанням мови програмування Haskel та її конвертованої моделі у VHDL-код, а також еталонного проєкту, розробленого за технологією SoC, що надає змогу зменшити ризики залишкових прихованих дефектів.

Застосування зазначених результатів забезпечує:

- формалізований опис простору можливих програмно-апаратних версій ІКС на програмовних платформах з урахуванням класифікатора видів диверсності (ядра, інтерфейси, синхронізація, технологічні процеси та обладнання);

- обґрунтований вибір підмножини варіантів версійної надмірності (через відповідний підграф) для конкретної цільової архітектури/проєкту;

- розрахунок і порівняння інтегральних метрик диверсності та складності для підтримки інженерних рішень під час проєктування й оцінювання;

- підвищення функційної безпечності за рахунок системного зниження ризиків відмов за загальною причиною за допомогою застосування комбінацій диверсності.

Результати досліджень були використані при створенні інформаційно-керуючих систем для ІКС АЕС України та Болгарії на базі платформи RadICS.

Голова комісії — Олександр ІВАСЮК

Члени комісії — Наталія РВАЧОВА

Вікторія КУЛІНІЧ

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи
Національного аерокосмічного університету
«Харківський авіаційний інститут»
к.т.н., доцент
_____ Андрій ГУМЕННИЙ
«12» _____ 2026 року

АКТ ВПРОВАДЖЕННЯ

наукових результатів дисертаційної роботи
Панаріна Артема Сергійовича, виконаної на здобуття наукового ступеня доктора філософії, у навчальному процесі та НДР кафедри кібербезпеки та інтелектуальних інформаційних технологій

Комісія у складі: голови комісії – декана факультету радіоелектроніки, комп'ютерних систем та інфокомунікацій, к.т.н Олексія ОДОКІЄНКА, і членів – професора кафедри кібербезпеки та інтелектуальних інформаційних технологій, к.т.н. Клайда ФУРМАНОВА, професора кафедри кібербезпеки та інтелектуальних інформаційних технологій, к.т.н. Дмитра УЗУНА встановила, доцента кафедри кібербезпеки та інтелектуальних інформаційних технологій, к.т.н. В'ячеслава ДУЖОГО, що у науково-дослідних проєктах і навчальному процесі кафедри використано наукові результати дисертаційної роботи, а саме:

графова модель для опису можливості отримання різних програмно-апаратних версій для побудови інформаційно-керуючих систем на програмовних платформах, з урахуванням деталізованого класифікатора видів диверсності;

методи програмно-апаратної диверсності, які базуються на процедурах функціонально-просторової диверсифікації засобів шинної організації, контролю цілісності даних, налаштувань проєкту та синхронізації з керованими зсувами;,

метод модельно-базованого тестування систем програмовної логіки, який базується на порівнянні версій, отриманих з використанням мови програмування Haskel та її конвертованої моделі у VHDL-код, а також еталонного проєкту, розробленого за технологією SoC.

Застосування зазначених результатів в навчальному процесі, зокрема, в курсах «Надійність і відмовостійкість комп'ютерних систем», «Технології систем на кристалі», «Сучасні методи комп'ютерної інженерії», курсовому і дипломному проєктуванні за спеціальністю «Комп'ютерна інженерія»:

- засвоєння здобувачами формалізованих підходів до опису простору можливих програмно-апаратних версій ІКС на програмовних платформах;

- формування навичок обґрунтованого вибору підмножини варіантів версійної надмірності для конкретної цільової архітектури/проєкту;

- опанування методів розрахунку і порівняння інтегральних метрик диверсності та складності для підтримки інженерних рішень під час проєктування й оцінювання;

- вивчення принципів підвищення енергоефективності та/або безпечності в цільових реалізаціях завдяки зниженню навантаження/надмірності та мінімізації ризиків відмов за загальною причиною;

- набуття навичок у верифікації/валідації систем програмовної логіки через використання різних технологій та порівняння результатів реалізації;

- формування практичних умінь зменшення ризиків залишкових прихованих дефектів завдяки незалежності джерел/ланцюжків отримання порівнюваних версій та їх взаємного контролю;

- ознайомлення зі стандартизованими мультиверсійними процедурами у життєвому циклі розроблення FPGA/SoC-проєктів в рамках дисциплін професійної підготовки.

Використання результатів досліджень сприяло формуванню практичних навичок, підвищенню якості виконання навчальних завдань і проєктів, а також впровадженню нових технологічних рішень у НДР.

Голова комісії — Олексій ОДОКІЄНКО

Члени комісії — Клайд ФУРМАНОВ

Дмитро УЗУН

Вячеслав ДУЖИЙ

ДОДАТОК Б
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Скляр В. В., Харченко В. С., Панарин А. С., Сандер И. Применение концепции Model-Based Testing для верификации систем на базе IP-ядер. Радіоелектронні і комп'ютерні системи. 2010. № 5. С. 237–241.

Sklyar V., Siora O., Herasimenko O., Panarin A. Development and Verification of Dependable Multiversion Systems on the basis of IP-Cores. Technical Approach to Dependability. Wroclaw, 2010. P. 133–145. URL: https://www.researchgate.net/profile/Vladimir-Sklyar/publication/305773007_Development_and_Verification_of_Dependable_Multiversion_Systems_on_the_basis_of_IP-Cores/links/57a0b39c08aeef8f311c52f5/Development-and-Verification-of-Dependable-Multiversion-Systems-on-the-basis-of-IP-Cores.pdf (дата звернення: 25.01.2026).

Панарин А. С. Имитационное моделирование soft-процессоров на базе концепции Model-Based Testing. Радіоелектронні і комп'ютерні системи. 2012. № 5. С. 100–106.

Скляр В. В., Харченко В. С., Панарин А. С. Тестирование и разработка диверсных программируемых логических контроллеров на базе ПЛИС с использованием среды функционального программирования. Радіоелектронні і комп'ютерні системи. 2014. № 1. С. 29–41.

Скляр В. В., Харченко В. С., Панарин А. С. Тестирование программируемых логических контроллеров на базе ПЛИС с использованием среды функционального программирования. Системи обробки інформації. 2014. № 1. С. 44–55.

Panarin A., Kharchenko V., Zemlianko H. Advanced classifier for expert assessment of diversity and cost of NPP information and control systems. Measuring Equipment and Metrology. 2025. Vol. 86, № 4. P. 52–64.

Бабешко Є., Панарін А. Методи та засоби диверсної синхронізації та реалізації електронних проєктів для систем безпеки АЕС на FPGA платформах.

Вимірювальна та обчислювальна техніка в технологічних процесах. 2026. Т. 1. С. 32–38. DOI: 10.31891/2219-9365-2026-85-5.

Babeshko E., Kovalenko A., Illiashenko O., Panarin A., Sklyar V. et al. Secure and resilient computing for industry and human domains. Vol. 2. Kharkiv: National Aerospace University named after N. E. Zhukovsky “KhAI”, 2017. 762 p. URL: http://serein.eu.org/wp-content/uploads/2017/10/Volume-2.-Secure-and-resilient-systems-and-infrastructures.pdf

Панарін А., Харченко В., Землянко Г. Розроблення розширеного класифікатору для експертного оцінювання диверсності та вартості інформаційно-керуючих систем АЕС. Вимірювальна техніка та метрологія. 2025. Т. 86, № 4. С. 52. DOI: 10.23939/istcmtm2025.04.052.

Панарін А. Принцип диверсності та багатоверсійні технології для систем захисту реакторів атомних електростанцій. Пропілеї права та безпеки. 2026. Vol. 8. P. 260–263. DOI: 10.32620/pls.2025.8.67.

Babeshko I., Duzhiy V., Panarin A., Illiashenko O., Siora A. et al. Diversity for NPP I&C Systems Safety and Cyber Security. Cyber Security and Safety of Nuclear Power Plant Instrumentation and Control Systems. IGI Global, 2020. P. 239–288. URL: https://www.igi-global.com/chapter/npp-ic-systems/258682 (дата звернення: 25.01.2026).

Kharchenko V., Ponochovnyi Y., Babeshko E., Ruchkov E., Panarin A. Safety Assessment of Maintained Control Systems with Cascade Two-Version 2oo3/1oo2 Structures Considering Version Faults. 18th DepCoS-RELCOMEX. 2023. Vol. 737. P. 119–129. DOI: 10.1007/978-3-031-37720-4_11.

Panarin A., Sklyar V., Kharchenko V., Kovalenko A., Babeshko E. Modeling of industrial FPGA-based controllers with ForSyDe. 2017 International Conference on Information and Digital Technologies (IDT). IEEE, 2017. P. 164–168. DOI: 10.1109/DT.2017.8024290.

Illiashenko O., Kharchenko V., Kor A.-L., Panarin A., Sklyar V. Hardware diversity and modified NUREG/CR-7007 based assessment of NPP I&C safety. 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced

Computing Systems: Technology and Applications (IDAACS). IEEE, 2017. Vol. 2. P. 907–911. DOI: 10.1109/IDAACS.2017.8095218.

Duzhyi V., Kharchenko V., Panarin A., Rusin D. Diversity metric evaluation considering extended NUREG-7007 diversity classification. 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). IEEE, 2018. P. 21–25. DOI: 10.1109/DESSERT.2018.8409092.

Kharchenko V., Kovalenko A., Leontiiev K., Panarin A., Duzhy V. Multi-diversity for FPGA platform based NPP I&C systems: new possibilities and assessment technique. International Conference on Nuclear Engineering. London, UK: American Society of Mechanical Engineers, 2018. Vol. 1. P. 1–9. DOI: 10.1115/ICONE26-82377.