НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОБЛЕМ МОДЕЛЮВАННЯ В ЕНЕРГЕТИЦІ ІМ. Г.Є. ПУХОВА

Кваліфікаційна наукова праця
на правах рукопису

ГІЛЬГУРТ СЕРГІЙ ЯКОВИЧ

УДК 004.274:004.056

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ СТВОРЕННЯ РЕКОНФІГУРОВНИХ СИГНАТУРНИХ ЗАСОБІВ ЗАХИСТУ ІНФОРМАЦІЇ КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

Спеціальність 05.13.05 - комп'ютерні системи та компоненти

Галузь знань - інформаційні технології

Подається на здобуття наукового ступеня доктора технічних наук

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

С.Я. Гільгурт

Науковий консультант - Чемерис Олександр Анатолійович
доктор технічних наук, старший науковий співробітник

Київ – 2020

АНОТАЦІЯ

Гільгурт С.Я. Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти. Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, Київ, 2019.

Дисертаційна робота присвячена розробці та розвитку методів побудови ефективних реконфігуровних апаратних сигнатурних засобів технічного захисту інформації шляхом створення комбінованої обчислювальної структури, оптимізованої за заданими показниками в заданих умовах.

Для досягнення мети були проаналізовані як проблеми технічного захисту інформації, що вирішуються сигнатурними засобами, так і можливості сучасних ПЛІС та пристроїв на їх основі в якості апаратної бази для вирішення цих проблем.

Проведено порівняльний аналіз сигнатурних систем захисту із засобами, що не використовують сигнатури, а також – реконфігуровної елементної бази з іншими апаратними платформами. В якості предмету дослідження виокремлено реконфігуровні апаратних сигнатурні засоби технічного захисту інформації. Проведено аналіз принципів функціонування та побудови таких засобів як технічних систем.

В результаті вивчення численних розробок в галузі побудови реконфігуровних апаратних сигнатурних засобів технічного захисту інформації отримана узагальнена структура пристрою такого типу. З'ясовано, що його ключовим компонентом є модуль розпізнавання, який виконує найбільш складну в обчислювальному сенсі функцію – вирішує завдання множинного розпізнавання рядків. Як наслідок, від властивостей даного модуля в значній мірі залежать результуючі характеристики всієї системи захисту в цілому. Специфічною особливістю застосування реконфігуровних

засобів для апаратного прискорення роботи сигнатурних систем захисту є необхідність повторного синтезу деяких компонентів, що регулярно виникає. Така операція потрібна, по-перше, при появі описів нових атак, по-друге – при зміні умов роботи інформаційної системи, що захищається. Тому актуальними є методи швидкого створення ефективних реконфігуровних обчислювальних структур, що мають бути синтезовані в ПЛІС.

Щоб мати можливість оцінювати і порівнювати властивості розробок сигнатурних засобів захисту та їх компонентів, сформульовані і класифіковані показники ефективності для таких систем.

В результаті вивчення наявного світового досвіду було виокремлено три найбільш ефективних підходи до побудови апаратних засобів інформаційного захисту: асоціативна пам'ять на базі цифрових компараторів, фільтр Блума на базі геш-функцій та алгоритм Ахо-Корасік, апаратно реалізований на базі скінченних автоматів. Вивчені та формалізовані властивості кожного напряму з урахуванням особливостей реалізації на реконфігуровних засобах. З'ясувалося, що жоден з досліджених підходів не забезпечує явної переваги в порівнянні з іншими. Кожному притаманні власні переваги та недоліки.

У зв'язку з цим природним чином виникає ідея об'єднати в одному модулі декілька різнорідних підходів таким чином, щоб максимізувати переваги кожного з них. Для її реалізації були розроблені і вдосконалені методи комбінування, що дозволяють домогтися підвищення ефективності результуючого рішення за рахунок застосування оптимізаційних процедур.

Розроблено методи прискореного обчислення кількісних значень технічних характеристик компонентів реконфігуровних засобів захисту, що дозволяють позбутися від часоємних та ресурсномістких процедур повного синтезу обчислювальної структури для ПЛІС. Суть методів полягає у формуванні так званих функцій оцінки, які є функціональними залежностями технічних характеристик синтезованих схем розпізнавання від властивостей набору сигнатур, що підлягають розпізнаванню з одного боку, та параметрів

реконфігуровного прискорювача – з іншого. У дисертації розроблені алгоритми реалізації запропонованих методів і програмні засоби для їх перевірки та аналізу.

Наукова новизна дослідження полягає в наступному. В роботі вперше розроблено метод прискореного обчислення технічних характеристик компонентів реконфігуровних засобів захисту інформації, що використовуються в якості цільових функцій процедур оптимізації, який на відміну від відомих методів проектування реконфігуровних пристроїв не потребує виконання витратних за часом процедур синтезу цифрових схем, дозволяючи за рахунок формування математичного опису здійснювати швидку оцінку та порівняння обчислювальних структур за визначеними показниками ефективності. Вперше розроблено метод прискорення процедури оптимізації паралельного комбінування, який відрізняється впорядкуванням патернів в наборі за певним параметром, що дозволяє замість повного перебору комбінаторно великої кількості варіантів здійснювати поділ патернів між блоками розпізнавання, що комбінуються, за лінійним законом часу, послідовно змінюючи параметр впорядкування. Метод не гарантує знаходження глобального оптимуму, проте зменшує обчислювальну складність процесу оптимізації до прийнятних для практичного використання значень. Вперше сформульовано та вдосконалено метод паралельного комбінування, який за рахунок паралельного з'єднання різних за принципами побудови блоків розпізнавання дозволяє підвищити показники ефективності модулю розпізнавання реконфігуровних засобів захисту інформації. Запропоноване вдосконалення відрізняється застосуванням розроблених методів прискорення, що уможливлює використання процедур оптимізації за рахунок зниження часових витрат до практично прийнятних значень. Вперше сформульовано та вдосконалено метод послідовного каскадування, який за рахунок послідовного з'єднання різних за принципами побудови блоків розпізнавання та уточнення попереднього розпізнавання, використовуючи процедуру оптимізації поділу

між ними питомих патернів по довжині, дозволяє підвищити показники ефективності модулю розпізнавання реконфігуровних засобів захисту інформації. Отримав подальший розвиток метод вертикального об'єднання, який за рахунок тісного сполучання в одному блоці кількох підходів або технічних рішень дозволяє підвищити показники ефективності модулю розпізнавання реконфігуровних засобів захисту інформації. Також вперше сформульовано та вдосконалено принцип комбінування методів комбінування, який за рахунок ієрархічного використання сформульованих, вдосконалених та розвитих методів комбінування дозволяє підвищити показники ефективності модулю розпізнавання реконфігуровних засобів захисту інформації до значень, недосяжних при використанні кожного з методів комбінування окремо.

Як показали проведені експерименти, паралельне комбінування різних підходів та їх модифікацій для апаратного розпізнавання сигнатур дозволяє поліпшити показники підсумкової схеми в окремих випадках на 35-40%, послідовне каскадування – на 40-45%. Принцип комбінування методів комбінування дозволяє компонувати розглянуті методи в ієрархічну структуру довільної глибини, та забезпечує підвищення показників ефективності модулю розпізнавання реконфігуровних засобів захисту інформації в окремих випадках на 20-45% порівняно з використанням окремих методів комбінування. Порівняння з відомими розробками, в яких застосовуються методи комбінування показало, що запропоновані та вдосконалені методи за рахунок прискорення обчислень та використання процедур оптимізації дозволяють підвищити ефективність в окремих випадках на 12-19%.

Практичне значення отриманих результатів полягає в тому, що застосування запропонованих методів побудови реконфігуровних засобів захисту інформації дозволить розробникам створювати більш ефективні засоби захисту, які використовуватимуться для протидії зовнішнім та внутрішнім атакам на комп'ютерні системи та мережі. Здобуті результати

щодо підвищення ефективності вирішення задачі множинного розпізнавання рядків актуальні й для мережевих застосувань, не пов’язаних з захистом інформації: прискореної обробки XML-запитів, класифікації пакетів з відновленням з'єднань для управління технологією QoS, фільтрації IP-телефонії, вимірювання трафіку, оптимізації кешування та ін. Запропоновані методи також можуть бути застосовані для вирішення важливих проблем з інших галузей науки, зокрема, інтелектуального пошуку даних (data mining), аналізу молекул ДНК тощо.

З метою спрощення практичного застосування запропонованих методів і засобів реалізовано у вигляді макетного зразка веб-сервіс централізованого створення реконфігуровних обчислювачів з використанням розподілених і хмарних обчислювальних середовищ. Сервіс дозволяє користувачам, які не мають навичок самостійної розробки обчислювальних структур для ПЛІС і програмування реконфігуровних пристроїв, завантажувати в систему перелік сигнатур, параметри наявних прискорювачів і додаткові вимоги до створюваних засобів технічного захисту інформації, в віддалено спостерігати за ходом процесу їх синтезу та отримувати файли конфігурації для ПЛІС своїх систем інформаційного захисту.

Ключові слова: технічний захист інформації, сигнатура, задача множинного розпізнавання рядків, ПЛІС, реконфігуровний обчислювач, підвищення ефективності, комбінування.

ABSTRACT

Hilgurt S.Ya. Methods and techniques for creating reconfigurable signature-based security tools for computer systems and networks. – Qualifying scientific work as manuscript.

Thesis for the degree of Doctor of Technical Sciences in specialty 05.13.05 – Computer Systems and Components. – Pukhov Institute for Modelling in Energy Engineering, National Academy of Sciences of Ukraine, Kyiv, 2019.

The thesis is devoted to the research and development of methods for constructing effective reconfigurable hardware signature-based means of information security by combining computing structure, optimized according to specified criteria under specified conditions.

To achieve this goal, both the problems of technical protection of information solved by signature tools and the possibilities of modern FPGAs and devices based on them as a hardware base to solve these problems were analyzed.

A comparative analysis of signature security systems with tools that do not use signatures, as well as – reconfigurable element base with other hardware platforms. Reconfigurable hardware signature means of technical protection of information are singled out as a subject of research. The analysis of the principles of functioning and construction of such means as technical systems is carried out.

As a result of studying numerous developments in the field of construction of reconfigurable hardware signature means of technical protection of information the generalized structure of the device of this type is received. It was found that its key component is the recognition module, which performs the most complex in the computational sense function – solves the problem of multiple line recognition. As a result, the resulting characteristics of the entire protection system as a whole largely depend on the properties of this module. A specific feature of the use of reconfigured means for hardware acceleration of signature protection systems is the need for re-synthesis of some components, which occurs regularly. Such an operation is required, firstly, when descriptions of new attacks appear, and secondly, when the operating conditions of the protected information system change. Therefore, the methods of rapid creation of efficient reconfigurable computing structures to be synthesized in FPGA are relevant.

To be able to evaluate and compare the properties of the development of signature security tools and their components, formulated and classified performance indicators for such systems.

As a result of studying the world experience, three most effective approaches to the construction of hardware information protection were identified:

associative memory based on digital comparators, Bloom filter based on hash functions and Aho-Korasik algorithm, implemented on the basis of finite automata. The properties of each direction are studied and formalized, taking into account the peculiarities of implementation on reconfigured means. It turned out that none of the studied approaches provides a clear advantage over others. Each has its own advantages and disadvantages.

In this regard, the idea naturally arises to combine several disparate approaches in one module so as to maximize the benefits of each. For its implementation, combination methods have been developed and improved, which allow to increase the efficiency of the resulting solution through the use of optimization procedures.

Methods of accelerated calculation of quantitative values of technical characteristics of components of reconfigured means of protection, which allow to get rid of time-consuming and resource-intensive procedures of full synthesis of computer structure for FPGA, have been developed. The essence of the methods is to form the so-called evaluation functions, which are functional dependences of the technical characteristics of the synthesized recognition schemes on the properties of the set of signatures to be recognized on the one hand and the parameters of reconfigured accelerator on the other. In the dissertation algorithms of realization of the offered methods and software for their check and the analysis are developed.

The scientific novelty of the study is as follows. The paper for the first time developed a method of accelerated calculation of technical characteristics of components of reconfigurable information security tools used as target functions of optimization procedures, which, unlike known methods of designing reconfigurable devices does not require time-consuming procedures to carry out a quick assessment and comparison of computer structures according to certain efficiency indicators. For the first time a method of accelerating the procedure of optimization of parallel combination was developed, which differs in ordering patterns in a set by a certain parameter, which allows instead of a complete search of combinatorially large number of variants to divide patterns between combined

recognition units. The method does not guarantee finding a global optimum, but reduces the computational complexity of the optimization process to acceptable values for practical use. For the first time the method of parallel combination was formulated and improved, which due to the parallel connection of different recognition blocks according to the principles of construction allows to increase the efficiency of the recognition module of reconfigurable means of information protection. The proposed improvement differs in the application of the developed methods of acceleration, which allows the use of optimization procedures by reducing time costs to almost acceptable values. For the first time the method of sequential cascading was formulated and improved, which due to serial connection of different on the basis of construction of recognition blocks and refinement of preliminary recognition, using the procedure of optimization of division between specific length patterns, allows to increase efficiency of reconfigurable information protection module. The method of vertical integration has been further developed, which, due to the close combination of several approaches or technical solutions in one block, allows increasing the efficiency of the recognition module of reconfigurable means of information protection. Also, for the first time the principle of combining combination methods was formulated and improved, which due to hierarchical use of formulated, improved and developed combination methods allows to increase the efficiency of the recognition module of reconfigurable means of information protection to values unattainable when using each combination method.

As experiments have shown, the parallel combination of different approaches and their modifications for hardware signature recognition can improve the performance of the final scheme in some cases by 35-40%, sequential cascading – by 40-45%. The principle of combining methods of combining allows combining the considered methods into a hierarchical structure of arbitrary depth, and provides an increase in the efficiency of the recognition module of reconfigurable means of information protection in some cases by 20-45% compared to individual combining methods. Comparison with known

developments in which the methods of combination are used showed that the proposed and improved methods by accelerating the calculations and the use of optimization procedures can increase efficiency in some cases by 12-19%.

The practical value of the results is that the application of the proposed methods of constructing reconfigurable means of information protection will allow developers to create more effective means of protection that will be used to counter external and internal attacks on computer systems and networks. The obtained results on improving the efficiency of solving the problem of multiple line recognition are relevant for network applications not related to information security: accelerated processing of XML-requests, classification of packets with connection recovery for QoS management, IP-telephony filtering, traffic measurement, cache optimization, etc. The proposed methods can also be used to solve important problems in other fields of science, in particular, data mining, analysis of DNA molecules and more.

In order to simplify the practical application of the proposed methods and tools implemented in the form of a prototype web service of centralized creation of reconfigurable computers using distributed and cloud computing environments. The service allows users who do not have the skills to independently develop computer structures for FPGA and programming reconfigurable devices, download a list of signatures, parameters of existing accelerators and additional requirements to the created means of technical protection of information, to remotely monitor the process of their synthesis and receive configuration files for FPGA of their information security systems.

Keywords: information security, signature, DPI, multi-pattern string matching, FPGA, reconfigurable accelerator, efficiency, approach combining.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Гильгурт С.Я. Применение реконфигурируемых вычислений для решения задач распознавания / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2002. – Вип. 16. – С.41-47.

2. Гильгурт С.Я. Анализ применений программируемых ИС для решения задач вычислительной техники / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2003. – Вип. 19. – С.79-85.

3. Гильгурт С.Я. Особенности применения реконфигурируемых вычислителей для аппаратной защиты информационных систем / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2007. – Вип. 38. – С.36-41.

4. Гильгурт С.Я. О применении реконфигурируемых вычислителей для решения задач защиты информации / С.Я. Гильгурт, А.К. Гиранова // Зб. наук. праць ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2008. – Вип. 46. – С.93-99.

5. Гильгурт С.Я. Обзор современных реконфигурируемых унифицированных вычислителей / С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2008. – Вип. 49. – С.17-24.

6. Гильгурт С.Я. Анализ применения унифицированных вычислителей в интеллектуальных системах / С.Я. Гильгурт // Искусственный интеллект. – 2009. – № 1. – С.144-148.

7. Гильгурт С.Я. О применении реконфигурируемых вычислителей для решения вопросов защиты информации, пересылаемой в грид-среде / С.Я. Гильгурт, А.К. Гиранова // Зб. наук. праць ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2009. – Вип. 51. – С.65-72.

8. Коростиль Ю.М. Анализ угроз и опасностей в компьютерных системах на предмет защиты цифровыми реконфигурируемыми

устройствами / Ю.М. Коростиль, А.Н. Давиденко, С.Я. Гільгурт, М.М. Панченко // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 56. – С.10-17.

9. Гильгурт С.Я. Обзор возможностей реконфигурируемых устройств для применения в компьютерной безопасности / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 55. – С.117-124.

10. Коростиль Ю.М. Принципы построения сетевых систем обнаружения вторжений на базе ПЛИС / Ю.М. Коростиль, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 57. – С.87-94.

11. Давиденко А.Н. Алгоритмы распознавания строк в системах обнаружения вторжений на ПЛИС / А.Н. Давиденко, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 58. – С.103-109.

12. Гильгурт С.Я. Программно-аппаратная защита данных в распределенных интеллектуальных системах / С.Я. Гильгурт, А.К. Гиранова // Искусственный интеллект. – 2010. – № 3. – С.706-711.

13. Гильгурт С.Я. Анализ типовых режимов обмена данными с реконфигурируемыми вычислителями / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2011. – Вип. 59. – С.113-121.

14. Гильгурт С.Я. Аппаратное распознавание строк в интеллектуальных системах защиты информации / С.Я. Гильгурт // Искусственный интеллект. – 2012. – № 1. – С.259-266.

15. Гильгурт С.Я. Сравнительный анализ межсетевых экранов, систем обнаружения и предотвращения вторжений, контроля целостности / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2011. – Вип. 60. – С.116-120.

16. Давиденко А.Н. Аппаратное ускорение алгоритмов сигнатурного обнаружения вторжений в открытой системе информационной безопасности Snort / А.Н. Давиденко, С.Я. Гильгурт, В.И. Сабат // Моделювання та

інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2012. – Вип. 65. – С.94-103.

17. Коростиль Ю.М. Анализ базы данных системы информационной безопасности Snort и вопросы быстродействия / Ю.М. Коростиль, С.Я. Гильгурт, О.М. Назаренко // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2012. – Вип. 66. – С.77-84.

18. Коростиль Ю.М. Перспективы развития реконфигурируемых вычислителей для выполнения ресурсоемких расчетов / Ю.М. Коростиль, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2013. – Вип. 68. – С.84-91.

19. Гильгурт С.Я. Реконфигурируемые вычислители. Аналитический обзор / С.Я. Гильгурт // Электронное моделирование. – 2013. – Т. 35, № 4. – С.49-72.

20. Гильгурт С.Я. Противодействие атакам алгоритмической сложности на системы обнаружения вторжений / С.Я. Гильгурт, Б.В. Дурняк, Ю.М. Коростиль // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2014. – Вип. 71. – С.3-12.

21. Hilhurt S. Ya. Application of FPGA-based Reconfigurable Accelerators for Network Security Tasks / S. Hilhurt // Collection of scientific works. Simulation and informational technologies. – PIMEE NAS of Ukraine. – Kyiv, 2014. – Vol. 73. – P. 17-26.

22. Гильгурт С.Я. Организация вычислительного процесса синтеза файлов конфигураций для аппаратных ускорителей при решении задач информационной безопасности / С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2015. – Вип. 74. – С.29-33.

23. Євдокимов В.Ф. Створення на базі грід-сайту ІПМЕ ім. Г.Є. Пухова НАНУ системи централізованого синтезу апаратних прискорювачів для вирішення задач інформаційної безпеки в енергетичній галузі /

В.Ф. Євдокимов, А.М. Давиденко, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2017. – Вип. 79. – С.3-8.

24. Гильгурт С.Я. Анализ применения аппаратного ускорения информационной защиты в автоматизированных системах энергетической отрасли / С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2018. – Вип. 83. – С.154-164.

25. Евдокимов В.Ф. Дополнительные этапы процедуры оперативной реконфигурации аппаратных ускорителей задач информационной безопасности / В.Ф. Евдокимов, А.Н. Давиденко, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2018. – Вип. 85. – С.3-11.

26. Евдокимов В. Централизованный синтез реконфигурируемых аппаратных средств информационной безопасности на высокопроизводительных платформах / В. Евдокимов, А. Давиденко, С. Гильгурт // Захист інформації. – 2018. – Т. 20, № 4. – С.247-258.

27. Гільгурт С.Я. Побудова асоціативної пам'яті на цифрових компараторах реконфігуровними засобами для вирішення задач інформаційної безпеки / С.Я. Гільгурт // Електронне моделювання. – 2019. – Т. 41, № 3. – С.59-80.

28. Гільгурт С. Побудова фільтрів Блума реконфігуровними засобами для вирішення задач інформаційної безпеки / С. Гільгурт // Безпека інформації. – 2019. – Т. 25, № 1. – С.53-58.

29. Гільгурт С. Побудова скінченних автоматів реконфігуровними засобами для вирішення задач інформаційної безпеки / С. Гільгурт // Захист інформації. – 2019. – Т. 21, № 2. – С.111-120.

30. Гільгурт С. Методи побудови оптимальних схем розпізнавання для реконфігуровних засобів інформаційної безпеки / С. Гільгурт // Безпека інформації. – 2019. – Т. 25, № 2. – С.74-81.

Наукові праці, які засвідчують апробацію матеріалів дисертації

1. Гильгурт С.Я. Применение типовых устройств на базе программируемой логики для решения вычислительных задач / С.Я. Гильгурт // Труды II международной конф. «Параллельные вычисления и задачи управления» РАСО’2004 памяти Е.Г. Сухова. Москва, 4-6 окт. 2004 г. – М.: Институт проблем управления им. В.А. Трапезникова РАН, 2004. – С.514-530.

2. Гильгурт С.Я. К вопросу о применении реконфигурируемых вычислителей для решения ресурсоемких задач / С.Я. Гильгурт // Информационные технологии в управлении энергетическими системами (ИТУЭС-2005): тез. докл. Междунар. конф. (18-19 октября 2005 р., г. Киев), К.: Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, 2005. – С.72-74.

3. Гильгурт С.Я. Решение задач распознавания с применением реконфигурируемых вычислителей / С.Я. Гильгурт, А.Н. Давиденко // Искусственный интеллект. Интеллектуальные и многопроцессорные системы – 2006: Тез. докл. Международной научно-технической конференции. Т. 2. – Таганрог: Изд-во ТРТУ, 2006. – С.76-79.

4. Гильгурт С.Я. Применение типовых реконфигурируемых устройств на базе ПЛИС для ресурсоемких вычислений / С.Я. Гильгурт // Матеріали Міжнар. наук.-практ. конф. «Інтелектуальні системи прийняття рішень та інформаційні технології», м. Чернівці, 17-19 травня 2006 р. – Чернівці: "Рута", 2006. – С.278-279.

5. Гильгурт С.Я. О применении реконфигурируемых унифицированных вычислителей для решения научно-технических задач / С.Я. Гильгурт // Параллельные вычислительные технологии (ПаВТ'2008): Труды международной научной конференции (Санкт-Петербург, 28 января – 1 февраля 2008 г.). – Челябинск: Изд. ЮУрГУ, 2008. – С.358-363.

6. Гильгурт С.Я. Некоторые вопросы аппаратного ускорения в современных информационных технологиях / С.Я. Гильгурт // Материалы

Международной научно-технической конференции «Многопроцессорные вычислительные и управляющие системы» (МВУС-2009), Дивноморское, 28 сентября – 03 октября 2009 г. Т. 1. – Таганрог: Изд-во ТТИ ЮФУ, 2009. – С.26-29.

7. Гильгурт С.Я. Анализ применения реконфигурируемых устройств в системах обнаружения вторжений / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2010». Т. 1. – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2010. – С.260-267.

8. Гильгурт С.Я. Аппаратное распознавание строк в интеллектуальных системах защиты информации / С.Я. Гильгурт // Тез. докл. международной научно-технической конференции «Искусственный интеллект. Интеллектуальные системы ИИ-2011». – Донецк: ИПИИ «Наука і освіта». – 2011. – Т. 1. – С.271-274.

9. Гільгурт С.Я. Програмно-апаратна система захисту даних для інфраструктури грід / С.Я. Гільгурт, А.К. Гіранова // Сучасні комп'ютерні системи та мережі: розробка та використання: матеріали 5-ої Міжнар. наук.-техн. конф. ACSN-2011, 29 вересня – 1 жовтня 2011, Львів, Україна. – Л.: Вид-во Нац. ун-ту «Львів. політехніка», 2011. – С.50-53.

10. Гільгурт С.Я. Множинне розпізнавання рядків у системах виявлення вторгнення на базі реконфігурованих обчислювачів / С.Я. Гільгурт // Сучасні комп'ютерні системи та мережі: розробка та використання: матеріали 5-ої Міжнар. наук.-техн. конф. ACSN-2011, 29 вересня – 1 жовтня 2011, Львів, Україна. – Л.: Вид-во Нац. ун-ту «Львів. політехніка», 2011. – С.54-56.

11. Гильгурт С.Я. Аппаратное ускорение задач информационной безопасности на базе ПЛИС с применением грид-вычислений / С.Я. Гильгурт // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тез. доп. VII Міжнар. наук.-прак. конф. (17-19 вересня 2014 р., м. Запоріжжя), Запоріжжя: ЗНТУ, 2014. – С.329-330.

12. Гильгурт С.Я. Задача множественного распознавания строк в интенсивном потоке данных и методы ее аппаратного ускорения / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2016». – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2016. – С.166-169.

13. Гильгурт С.Я. Применение реконфигурируемых вычислителей для аппаратного ускорения сигнатурных систем защиты информации / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2018». – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2018. – С.107-110.

14. Hilgurt S.Ya. A method to construct the signature-based secure tools on reconfigurable accelerators / S.Ya. Hilgurt // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тез. доп. IX Міжнар. наук.-прак. конф. (03-05 жовтня 2018 р., м. Запоріжжя) [Електронний ресурс] / Редкол.: Д.М. Піза, С.В. Морщавка. Електрон. дані. – Запоріжжя: ЗНТУ, 2018. – 1 електрон. опт. диск (DVD-ROM); 12 см. – Назва з тит. екрана, 2018. – С.200-201.

15. Гильгурт С.Я. Централизованный синтез в грид-среде реконфигурируемых средств защиты информации / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Високопродуктивні обчислення» (HPC-UA 2018) – Київ, 2018. – С.40-47.

16. Гильгурт С.Я. Применение реконфигурируемых устройств на базе ПЛИС при построении сетевых систем обнаружения вторжений / С.Я. Гильгурт // Перспективні напрями захисту інформації: матеріали IV всеукр. наук.-пр. конф. – м. Одеса, 02-06 вересня 2018 р.: збірник тез. – Одеса: ОНАЗ ім. О.С. Попова, 2018. – С.37-42.

17. Гільгурт С.Я. Підвищення ефективності реконфігуровних систем виявлення вторгнень / С.Я. Гільгурт // Безпека інформаційних технологій: матеріали IX Міжнар. наук.-техн. конф. ITSec-2019, 22-27 березня 2019, м. Шарм-ель-Шейх, Єгипет. – К.: НАУ, 2019. – С.10-11.

18. Давыденко А.Н. Применение грид-сети для синтеза промышленных систем защиты информации на базе ПЛИС / А.Н. Давыденко, С.Я. Гільгурт // «Цифровые технологии в промышленности»: материалы республиканской научно-практической конференции, г. Актау, Казахстан, 28 марта 2019 г. – Актау, КГУТИ им. Ш. Есенова, 2019. – С.15-20.

19. Hilgurt S. Method for constructing reconfigurable multi-pattern matching modules for information security systems / S Hilgurt // Захист інформації і безпека інформаційних систем: матеріали VII Міжнар. наук.-техн. конф, м. Львів, 30-31 травня 2019. – Львів: Видавництво Львівської політехніки, 2019. – С.134-135.

20. Гільгурт С.Я. Апаратне рішення задач кібербезпеки в електроенергетичній галузі / С.Я. Гільгурт // Кібербезпека енергетики: Збірка праць конференції, м. Одеса, 28 травня – 1 червня 2019. – К.: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2019. – С.11-14.

21. Давиденко А.М., Політучий О.О. Методи підвищення ефективності мережевих систем виявлення вторгнень на базі ПЛІС / А.М. Давиденко, С.Я. Гільгурт, О.О. Політучий // Матеріали V Всеукр. наук.-практич. конф. «Перспективні напрями захисту інформації 2019», смт Затока Одеської обл., 31 серпня – 5 вересня 2019. – Одеса: 2019. – С.52-54.

Наукові праці, які додатково відображають наукові результати дисертації

1. Заявка u 2019 09353; G06F17/27; Апаратно-програмний комплекс підтримки прийняття рішень при проведенні державних експертиз комплексних систем захисту інформації / Давиденко А.М., Гільгурт С.Я., Шабан М.Р.; Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова Національної академії наук України. – заявл. 16.08.2019 р. (Рішення про видачу деклараційного патенту на корисну модель № 25510/ЗУ/19 від 28.10.2019 р.)

ЗМІСТ

Зміст..... 19
Список термінів та умовних скорочень 28
Вступ.... 34
Розділ 1. Дослідження задач захисту інформації та можливостей їх вирішення реконфігуровними засобами..... 48
1.1. Аналіз засобів захисту інформації 48
1.1.1. Основні поняття та задачі інформаційної безпеки..... 48
1.1.2. Системи виявлення вторгнень..... 50
1.1.2.1. Мережеві системи виявлення вторгнень..... 50
1.1.2.2. Склад МСВВ. 50
1.1.2.3. Системи запобігання вторгнень..... 52
1.1.2.4. Різновиди МСВВ 53
1.1.2.5. Сигнатурні засоби технічного захисту інформації..... 54
1.1.2.6. Функціонування сигнатурної МСВВ..... 54
1.1.2.7. Співвідношення між МСВВ та міжмережевими екранами. Поняття глибокого аналізу пакетів 55
1.1.2.8. Некомерційні МСВВ та їх сигнатурні бази даних..... 56
1.1.2.9. Поняття сигнатури та патерну. Склад сигнатури МСВВ..... 57
1.1.3. Інші сигнатурні засоби технічного захисту інформації..... 61
1.1.3.1. Антивірусні сигнатурні системи..... 61
1.1.3.2. Засоби протидії мережевим хробакам..... 62
1.1.3.3. Сигнатурні спам-фільтри..... 62
1.2. Задача множинного розпізнавання патернів..... 63
1.2.1. Формулювання задачі множинного розпізнавання патернів63
1.2.2. Алгоритми одиночного розпізнавання..... 66
1.2.3. Ефект самоподоби. Префікси, інфікси, суфікси..... 67
1.2.4. Алгоритми множинного розпізнавання..... 67

1.2.5. Специфіка реалізації множинного розпізнавання 68
1.3. Апаратне прискорення 68
1.3.1. Складності вирішення ЗМРП традиційними засобами 68
1.3.2. Суперкомп'ютери, кластери, грід та хмарні обчислення 69
1.3.3. Апаратне прискорення сигнатурних засобів ТЗІ 70
1.3.4. Уніфіковані обчислювачі 70
1.3.5. Існуючи апаратні платформи 71
1.3.5.1. Спецпроцесори 71
1.3.5.2. Мережеві процесори 72
1.3.5.3. TCAM 73
1.3.5.4. Багатоядерні процесори 73
1.3.5.5. Графічні прискорювачі 74
1.3.5.6. Реконфігуровні уніфіковані обчислювачі 75
1.3.6. Порівняння апаратних платформ для вирішення ЗМРП 76
1.4. Аналіз можливостей програмованої логіки 79
1.4.1. Загальна структура ПЛІС 81
1.4.2. Конфігуровні логічні блоки 82
1.4.3. Секції 82
1.4.4. Логічні комірки 82
1.4.5. Логічні таблиці 83
1.4.6. Тригери логічних комірок 84
1.4.7. Блокова пам'ять 84
1.4.8. Інші компоненти та технічні особливості ПЛІС 85
1.5. Реконфігуровні уніфіковані обчислювачі 85
1.5.1. Структура та склад РУО 85
1.5.2. Існуючі реалізації 87
1.5.3. Складності при використанні РУО та шляхи їх усунення 89
1.6. Висновки до розділу 1 90
Розділ 2. Вирішення задач захисту інформації реконфігуровними засобами 92

2.1. Основи побудови та функціонування реконфігуровних РАСЗТЗІ 92
2.1.1. Узагальнена структура МСВВ на ПЛІС 92
2.1.2. Критичні компоненти РАСЗТЗІ 94
2.1.3. Поняття процедури оперативного оновлення 95
2.1.4. Змінні та постійні складові РАСЗТЗІ 96
2.2. Показники ефективності РАСЗТЗІ 96
2.2.1. Вартісні показники 98
2.2.2. Показники продуктивності (швидкісні) 98
2.2.3. Функціональні показники 99
2.2.3.1. Спроможність протидіяти атакам на РАСЗТЗІ 100
2.2.3.2. Здатність до динамічної реконфігурації 100
2.2.3.3. Селективне розпізнавання 101
2.2.3.4. Режим запобігання вторгнень 101
2.2.3.5. Інші функціональні показники 102
2.2.4. Використання надлишковості 102
2.2.5. Показники масштабованості 102
2.2.6. Похідні показники 102
2.2.7. Питання енергоспоживання 103
2.2.8. Взаємозв’язок показників. Поняття трейдофу 103
2.3. Попередній аналіз відомих підходів до побудови МР РАСЗТЗІ 104
2.3.1. Гнучке розпізнавання 106
2.3.2. Найбільш ефективні підходи до побудови МР РАСЗТЗІ ... 107
2.4. Висновки до розділу 2 108
Розділ 3. Формалізація властивостей найбільш ефективних підходів до побудови схем розпізнавання 109
3.1. Використання асоціативної пам'яті на базі цифрових компараторів 109
3.1.1. Асоціативна та трійкова пам'ять з фіксованою логікою..... 110

3.1.2. Асоціативна та трійкова пам'ять на базі ПЛІС 111
3.1.3. Базова схема на цифрових компараторах..... 112
3.1.4. Особливості реалізації схеми з цифрових компараторів на ПЛІС 113
3.1.5. Зменшення ресурсних витрат 115
3.1.6. Подальше зменшення ресурсних витрат 118
3.1.7. Розпаралелювання..... 119
3.1.8. Небайтова розрядність обробки даних 122
3.1.9. Кластеризація словника патернів 128
3.1.10. Здатність до динамічної реконфігурації..... 130
3.1.11. Висновки щодо підходу на основі асоціативної пам'яті... 130
3.2. Використання фільтра Блума на базі геш-функцій 132
3.2.1. Основні принципи побудови та використання фільтра Блума 133
3.2.2. Кластерізація 136
3.2.3. Особливості реалізації геш-функцій на ПЛІС 138
3.2.4. Організація роботи з регістром бітів фільтра Блума на ПЛІС 140
3.2.5. Розпаралелювання..... 141
3.2.6. Уточнення результатів..... 143
3.2.7. Подальше підвищення швидкодії 144
3.2.8. Здатність до динамічної реконфігурації..... 145
3.2.9. Висновки щодо підходу на основі фільтра Блума..... 146
3.3. Використання алгоритму Ахо–Корасік на базі скінченних автоматів..... 148
3.3.1. Алгоритм Ахо–Корасік і його реалізація у вигляді кінцевого автомату..... 151
3.3.2. Класифікація типів переходів автомата СА-АК 155
3.3.3. Базова схема скінченного автомата Ахо–Корасік 155
3.3.4. Основні властивості базових схем СА-АК..... 157

3.3.5. Особливості реалізації схеми СА-АК на ПЛІС 158
3.3.6. Модифікації базової схеми СА-АК..... 159
3.3.7. Зменшення ресурсних витрат 159
3.3.8. Конвеєризація..... 160
3.3.9. Розпаралелювання 161
3.3.10. Небайтова розрядність обробки даних..... 163
3.3.11. Здатність до динамічної реконфігурації 163
3.3.12. Висновки щодо підходу на основі алгоритму Ахо–Корасік 164
3.4. Порівняльний аналіз підходів до побудови МР РАСЗТЗІ..... 165
3.5. Висновки до розділу 3 168
Розділ 4. Кількісна оцінка підходів до побудови МР РАСЗТЗІ..... 169
4.1. Набір патернів 170
4.2. Техніка впорядкування патернів 170
4.3. Функції самоподоби набору патернів..... 172
4.4. Основні кількісні технічні характеристики МР..... 173
4.4.1. Ресурсні параметри..... 174
4.4.2. Часові параметри 175
4.5. Метод прискореного обчислення технічних параметрів РАСЗТЗІ 175
4.6. Реалізація функцій оцінки для деяких схем розпізнавання..... 177
4.6.1. Техніка створення функцій оцінки для асоціативної пам'яті..... 178
4.6.1.1. Функція оцінки базової схеми BsCAM. Ресурсна складова 179
4.6.1.2. Функція оцінки базової схеми BsCAM. Часова складова 185
4.6.1.3. Функція оцінки схеми DCAM. Ресурсна складова 187
4.6.1.4. Функція оцінки схеми DCAM. Часова складова..... 197
4.6.1.5. Функція оцінки схеми DpCAM. Ресурсна складова .. 198

4.6.1.6. Функція оцінки схеми DpCAM. Часова складова 205
4.6.2. Техніка створення функцій оцінки для фільтрів Блума 206
4.6.2.1. Витрати ресурсів на генератор геш-функцій для ФБ 207
4.6.2.2. Витрати ресурсів на регістр бітів ФБ 211
4.6.2.3. Витрати ресурсів на допоміжні схеми для ФБ 216
4.6.2.4. Функція оцінки схеми повнорозмірного фільтра Блума LBF. Ресурсна складова..... 220
4.6.2.5. Функція оцінки схеми спрощеного фільтра Блума SBF. Ресурсна складова 222
4.6.2.6. Функції оцінки схем повнорозмірного фільтра Блума LBF та спрощеного фільтра Блума SBF. Часова складова 225
4.6.3. Техніка створення функцій оцінки для скінчених автоматів Ахо–Корасік 226
4.6.3.1. Техніка підрахунку ресурсів пам'яті для функцій оцінки СА-АК 228
4.6.3.2. Техніка підрахунку ресурсів керуючих пристроїв для функцій оцінки СА-АК 229
4.6.3.3. Функція оцінки СА-АК із зовнішньою пам'яттю ACRAM. Ресурсна складова 231
4.6.3.4. Функція оцінки СА-АК із зовнішньою пам'яттю ACRAM. Часова складова..... 233
4.6.3.5. Функція оцінки СА-АК із блоковою пам'яттю ACBRAM. Ресурсна складова 235
4.6.3.6. Функція оцінки СА-АК із блоковою пам'яттю ACBRAM. Часова складова..... 242
4.6.3.7. Функція оцінки СА-АК за конвеєрною схемою PipACBRAM. Ресурсна складова..... 244
4.6.3.8. Функція оцінки СА-АК за конвеєрною схемою PipACBRAM. Часова складова 246

4.6.4. Техніка створення функцій оцінки для двокаскадної схеми на базі геш-функції, ПЗП та цифрових компараторів..... 246
4.6.4.1. Функція оцінки схеми HRCmp. Ресурсна складова... 248
4.6.4.2. Функція оцінки схеми схема HRCmp. Часова складова 259
4.6.5. Способи знаходження коефіцієнтів нормалізації ресурсів РУО різних типів 259
4.6.6. Технічні характеристики РУО, потрібні для використання методів прискорення 260
4.7. Висновки до розділу 4 261
Розділ 5. Методи побудови оптимізованої структури модуля розпізнавання РАСЗТЗІ 264
5.1. Технічні параметри модулю розпізнавання та його компонентів 265
5.1.1. Ресурсні параметри 265
5.1.2. Часові параметри 266
5.1.3. Пропускна здатність 266
5.1.4. Похідні параметри 267
5.2. Критерії оптимізації 268
5.3. Метод паралельного комбінування 271
5.3.1. Структура методу МПрКмб 272
5.3.2. Функціонування методу МПрКмб 272
5.3.3. Реалізація методу МПрКмб 273
5.3.4. Узгодження швидкісних характеристик блоків розпізнавання 275
5.4. Метод послідовного каскадування 275
5.4.1. Структура методу МПсКс 276
5.4.2. Функціонування методу МПсКс 278
5.4.3. Реалізація методу МПсКс 279
5.5. Метод вертикального об’єднання 281

5.5.1. Структура методу МВрОб 281
5.5.2. Таблиця сумісності для МВрОб 282
5.5.3. Функціонування методу МВрОб 283
5.5.4. Реалізація методу МВрОб 284
5.5.4.1. Об'єднання на базі підходу СА-АК 284
5.5.4.2. Технічні рішення – мультиплікатори 285
5.6. Комбінування методів комбінування 286
5.7. Висновки до розділу 5 288
Розділ 6. Алгоритми побудови оптимізованої структури модуля розпізнавання РАСЗТЗІ 289
6.1. Алгоритм реалізації методу МПрКм 289
6.1.1. Проблема розбиття патернів у методі МПрКм 289
6.1.2. Метод прискорення процедури оптимізації для МПрКм ... 290
6.2. Алгоритм реалізації методу МПсКс 292
6.3. Алгоритм реалізації методу МВрОб 292
6.4. Алгоритм реалізації принципу КМК 293
6.5. Загальна структура використання методів комбінування 293
6.6. Експериментальне програмне забезпечення 294
6.7. Проведення експериментів 297
6.7.1. Низка експериментів № 1. Комбінування методом МПрКм297
6.7.2. Експеримент № 2. Комбінування методом МПсКс 309
6.7.3. Низка експериментів № 3. Принцип КМК 312
6.7.4. Експеримент № 4. Порівняння з опублікованими даними . 316
6.8. Висновки до розділу 6 318
Розділ 7. система централізованого синтезу компонентів РАСЗТЗІ 320
7.1. Принцип централізованого створення обчислювальних структур реконфігуровних пристроїв 320
7.2. Функціонування централізованого веб-сервісу 321
7.2.1. Створення машинних описів компонентів РАСЗТЗІ 322
7.2.2. Генерація конфігурацій, що завантажуються в ПЛІС 323

7.3. Особливості реалізації веб-сервісу 325
7.4. Висновки до розділу 7 327
Висновки 328
Список використаних джерел 332
Додатки 357
Додаток А. Список публікацій здобувача за темою дисертації 357
Додаток Б. Документи, що підтверджують впровадження результатів дисертаційної роботи, та лист підтримки 366
Додаток В. Експериментальне програмне забезпечення 372

СПИСОК ТЕРМІНІВ ТА УМОВНИХ СКОРОЧЕНЬ

ПЕ	показник ефективності
ЕПК FPM	Експериментальний програмний комплекс FPGA_Pattern_Matching
РС	ресурсна складова
ЧС	часова складова
AC	Aho–Corasick algorithm – алгоритм Ахо–Корасік
ACBRAM	схема розпізнавання на базі алгоритму Ахо–Корасік із блоковою пам'яттю (Aho–Corasick + Block Random Access Memory)
AC-DFA	Aho–Corasick deterministic finite automaton – детермінований скінченний автомат Ахо–Корасік
PipACBRAM	конвеєрна схема розпізнавання на базі алгоритму Ахо–Корасік із блоковою пам'яттю (Pipelined Aho–Corasick + Block Random Access Memory)
ACRAM	схема розпізнавання на базі алгоритму Ахо–Корасік із зовнішньою пам'яттю (Aho–Corasick + Random Access Memory)
ASIC	Application-Specific Integrated Circuit – напівзамовна надвелика інтегральна схема
BCAM	схема розпізнавання з використанням асоціативної пам'яті на базі БДР (BDD-based CAM)
BDD	Binary Decision Diagram – бінарна діаграма рішень
BF	Bloom Filter – фільтр Блума
BRAM	блокова пам'ять ПЛІС
BsCAM	базова схема розпізнавання на базі асоціативної пам'яті (Base Scheme on Content-Addressable Memory)
CAM	Content-Addressable Memory – асоціативна пам'ять

CDFA	Cached deterministic finite automaton – кешований детермінований скінченний автомат
CLB	Configurable Logic Block – конфігуровний логічний блок (КЛБ)
CPU	Central Processing Unit – центральний процесор
DCAM	схема розпізнавання на базі декодованої асоціативної пам'яті (Decoded Content-Addressable Memory)
DEL	Delay – цифрова схема затримки
DFA	deterministic finite automaton – детермінований скінченний автомат
DpCAM	схема розпізнавання на базі частково декодованої асоціативної пам'яті (Decoded partial Content-Addressable Memory)
DPI	Deep Packet Inspection – докладний аналіз пакетів
FA	finite automaton – скінченний автомат
FPGA	Field-Programmable Gate Array – різновид ПЛІС зі статичною пам'яттю, СППЛІС
FSM	finite state machine – скінченний автомат
GPGPU	General-Purpose Graphics Processing Unit – технологія обчислень загального призначення на графічних процесорах
GPU	Graphics Processing Unit – графічний процесор
HBC	схема розпізнавання на базі півбайтових компараторів (Half-Byte Comparators)
HbcDCAM	схема розпізнавання на базі півбайтових компараторів та декодованої асоціативної пам'яті (Half-Byte Comparators + Decoded Content-Addressable Memory)
HPC	High Performance Computing – високопродуктивний комп'ютінг

HRCmp	двокаскадна схема розпізнавання на базі геш-функції, ПЗП та цифрових компараторів (Hash-function + Read Only Memory + digital Comparators)
IDS	Intrusion Detection System – система виявлення вторгнень
IOB	Input/Output Block – блок введення-виведення (БВВ)
IPS	Intrusion Prevention System – система запобігання вторгнень
LBF	схема розпізнавання Повнорозмірний фільтр Блума (Large Bloom Filter)
LUT	Lookup Table – логічна таблиця
MBF	Mini Bloom Filter – міні-фільтр Блума (міні-ФБ)
NFA	nondeterministic finite automaton – недетермінований скінченний автомат
NIDS	Network Intrusion Detection System – мережева система виявлення вторгнень
NP	Network Processor – мережевий процесор
ParCAM	схема розпізнавання на базі паралельної роботи асоціативної пам'яті (Parallel Content-Addressable Memory)
ParDCAM	схема розпізнавання на базі паралельної роботи декодованої асоціативної пам'яті (Parallel Content-Addressable Memory)
ParHbcDCAM	схема розпізнавання на базі паралельної роботи півбайтових компараторів та декодованої асоціативної пам'яті (Parallel Half-Byte Comparators + Decoded Content-Addressable Memory)
PBF	Partial Bloom Filter – частковий фільтр Блума
PCI-E	системна шина PCI Express
RAM	Random Access Memory – пам'ять довільного доступу
RISC	Reduced Instruction Set Computer – різновид архітектури процесорів зі скороченим набором команд.

SATA	Serial ATA – послідовний інтерфейс обміну з накопичувачами даних в комп'ютерах
SBF	схема розпізнавання Спрощений фільтр Блума (Simplified Bloom Filter)
SCSD	Signature-based Collaborative Spam Detection – сигнатурний спам-фільтр
SM	state machine – абстрактний автомат або цифровий автомат
SoC	System on Chip – система на кристалі
TCAM	Ternary Content Addressable Memory – трійкова асоціативна пам'ять
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language – мова опису апаратури
VLIW	Very Long Instruction World – різновид архітектури процесорів "Дуже довга машинна команда"
АК	алгоритм Ахо–Корасік
АП	асоціативна пам'ять
АСЗТЗІ	апаратні сигнатурні засоби технічного захисту інформації
АСС	антивірусна сигнатурна система
БВВ	блок введення-виведення
БДР	бінарна діаграма рішень
БР	блок розпізнавання
ВІС	велика інтегральна схема
ГГФ	генератор геш-функцій
ГГф	генератор геш-функції
ГФ	геш-функція
ДСА	детермінований скінченний автомат
ЕСС	Error Correction Checking – система виявлення та корекції помилок
ЗІ	захист інформації
ЗМРП	задача множинного розпізнавання патернів

ЗП	запам'ятовуючий пристрій
ЗПМХ	засіб протидії мережевим хробакам
ЗТЗІ	засоби технічного захисту інформації
ІБ	Інформаційна безпека
КЛБ	конфігуровний логічний блок
КОЗП	контролер зовнішнього запам’ятовуючого пристрою
КП	керуючий пристрій
ЛК	логічна комірка
ЛОМ	локальна обчислювальна мережа
ЛТ	логічна таблиця – LUT (Lookup Table)
МВрОб	метод вертикального об’єднання
ММЕ	міжмережевий екран
МПрКм	метод паралельного комбінування
МПсКс	метод послідовного каскадування
МР	модуль розпізнавання
МСВВ	мережева система виявлення вторгнень
МСЗВ	мережева система запобігання вторгнень
НВІС	надвелика інтегральна схема
НСА	недетермінований скінченний автомат
ОЗП	оперативний запам'ятовуючий пристрій
ОС	операційна система
ПЕОМ	персональна електронна обчислювальна машина
ПЗ	програмне забезпечення
ПЗП	постійний запам’ятовуючий пристрій
ПК	персональний комп’ютер
ПЛІС	програмована логічна інтегральна схема
ПЛМ	програмована логічна матриця – різновид ПЛІС
ПОО	процедура оперативного оновлення
РАСЗТЗІ	реконфігуровні апаратні сигнатурні засоби технічного захисту інформації

РБ	регістр бітів
РУО	реконфігуровний уніфікований обчислювач
СА	скінченний автомат
СА-АК	скінченний автомат Ахо–Корасік
САПР	система автоматизованого проектування
СВБДР	скорочена впорядкована бінарна діаграма рішень
СВВ	система виявлення вторгнень
СЗВ	система запобігання вторгнень
СЗТЗІ	сигнатурні засоби технічного захисту інформації
СЛАР	система лінійних алгебраїчних рівнянь
СППЛІС	ПЛІС зі статичною пам'яттю (тобто типу FPGA)
ССФ	сигнатурний спам-фільтр
ТАП	трійкова асоціативна пам’ять
ТЗІ	технічний захист інформації
УЛТ	умовна логічна таблиця (одиниця виміру витрат ресурсів)
УО	уніфікований обчислювач
ФБ	фільтр Блума
ФО	функція оцінки
ЦА	цифровий автомат
ЦК	цифровий компаратор
ЦСЗ	цифрова схема затримки

ВСТУП

Обґрунтування вибору теми дослідження. Загрози інформаційної безпеки відносяться до числа найважливіших проблем, що стоять сьогодні перед людиною. Використання досягнень інформаційних технологій надає безліч переваг в різних галузях його діяльності від приватного сегменту до банківської сфери та промислового виробництва. Але, разом з тим, застосування IT-підходів привносить і властиві їм недоліки, включаючи проблеми інформаційної безпеки. Причому, наслідки від реалізації кіберзагроз при цифровізації саме підприємств народного господарства можуть бути значно більш важкими в порівнянні з традиційними застосуваннями інформаційних технологій, особливо коли йдеться про автоматизацію критичної інфраструктури, зокрема, енергетичної галузі.

Урядові організації та галузеві фахівці починають усвідомлювати небезпеку кібератак на об'єкти критичної інфраструктури щодо нанесення катастрофічного збитку на рівні країни. Популярний раніше принцип "Безпека через неясність" ("Security through obscurity"), що полягає в досягненні інформаційної безпеки шляхом приховування внутрішнього устрою системи і особливостей її реалізації, вже не відповідає реаліям часу [1]. Це наочно продемонстрували підтверджені атаки із застосуванням хробака Stuxnet на іранські об'єкти атомної промисловості в 2010-му році та троянської програми BlackEnergy – на енергопостачальні підприємства України у 2015-му [2].

Для боротьби з інформаційною небезпекою використовуються різні заходи – юридичні, адміністративні, організаційні, технічні, криптографічні тощо. Кожному напряму захисту притаманні свої складності та виклики. Серед технічних засобів захисту інформації важливу роль відіграють системи, принцип дії яких заснований на пошуку в інформаційних потоках заздалегідь відомих ознак шкідливої активності – так званих сигнатур. На

відміну від альтернативних підходів, які також активно та успішно розвиваються останнім часом, сигнатурні засоби досі демонструють кращі результати щодо точності виявлення зловживань, генерують менше помилок розпізнавання як першого, так и другого роду.

Сучасні сигнатурні засоби технічного захисту (таки як мережеві системи виявлення вторгнень (МСВВ), противірусні сканери, фільтри спаму, засоби протидії мережевим хробакам тощо), мають вирішувати в реальному часі обчислювально складну задачу множинного розпізнавання рядків. У зв'язку з припиненням зростання частоти традиційних процесорів, а також через сталий зріст зловмисної активності програмні додатки вже не забезпечують потрібної продуктивності. Тому останнім часом розробники вимушені звертатися до апаратних рішень, найчастіше – на базі програмованих логічних інтегральних схем (ПЛІС). Висока продуктивність програмованої логіки в поєднанні з гнучкістю, близької до програмної, природним чином відповідає складній та динамічній сутності завдань інформаційного захисту. В якості апаратної платформи таких рішень найчастіше використовуються реконфігуровні обчислювачі (або прискорювачі), які зазвичай містять мікросхему ПЛІС, пристрої оперативної пам'яті, інші допоміжні компоненти, а також мають ефективні та уніфіковані засоби взаємодії з традиційною комп’ютерною технікою.

Але використання реконфігуровних засобів само по собі не вирішує проблеми браку продуктивності. Збільшення об’ємів мережевого трафіку триває. Складність та витонченість зловмисної активності зростають. Також з часом змінюється коло об'єктів, що наражаються на небезпеку: якщо сьогодні зовнішні атаки загрожують переважно локальним мережам компаній та організацій, то згодом, по мірі застосування інформаційних технологій інтернету речей та кіберфізичних систем, до зони ризику почнуть підпадати новітні виробничі підприємства.

Слід зауважити, що стан справ з проблеми інформаційного захисту сучасних промислових об'єктів на сьогоднішній день знаходиться на

початковій стадії. Кіберфізичні системи, до яких відносяться, зокрема, сучасні цифрові підстанції (які будуються за стандартом МЕК-61850), вимагають інших підходів при створенні систем кіберзахисту. Наявний в IT-галузі досвід боротьби з кібератаками, з одного боку, не можна застосувати безпосередньо до інтелектуальних промислових систем, з іншого – його доцільно використовувати, але з урахуванням специфічних особливостей галузі [1].

У науковій літературі з'являється все більше публікацій, присвячених аналізу та розробкам засобів інформаційного захисту систем автоматизації. Однак, як визнають самі дослідники, ситуація поки що залишається на початковому етапі. У зв'язку з цим питання аналізу особливостей та підвищення ефективності вирішення завдань інформаційної безпеки АСУ ТП, зокрема, в енергетиці, є актуальними і злободенними.

Розробкою та вдосконаленням реконфігуровних апаратних сигнатурних засобів технічного захисту інформації (РАСЗТЗІ) в комп’ютерних системах і мережах сьогодні займаються численні дослідники зі всього світу. Запропоновано багато технічних рішень та їх модифікацій, заснованих на різних за своєю природою принципах та підходах до побудови швидкісних цифрових схем розпізнавання патернів (фіксованих послідовностей символів) у вхідному потоці. Кожному з підходів притаманні як переваги, так і недоліки. Так, цифрові компаратори та побудовані на їх основі різновиди асоціативної пам'яті забезпечують максимальну швидкодію, але є більш витратними щодо споживання апаратних ресурсів і електроенергії, вони також програють в плані масштабування. Фільтр Блума на базі геш-функцій більш економний та краще масштабується, але накладає обмеження по довжині сигнатур; він також вимагає додаткових витрат на доуточнення результатів роботи через системну наявність помилок розпізнавання другого роду. Скінчені автомати Ахо–Корасік забезпечують стабільну, але відносно невисоку пропускну здатність, складні в побудові,

вимагають "вибухового" зростання об’єму пам'яті для великих словників сигнатур.

Отже, пануючого підходу, який за більшістю показників випереджав би інші, досі не виявлено.

Принцип комбінування тобто об’єднання в єдиному пристрої розпізнавання різних підходів таким чином, щоб їх переваги посилювалися, а недоліки нівелювалися, тою чи іншою мірою застосовується розробниками апаратних засобів захисту. Але його реалізація зазвичай базується на евристичних принципах.

Тому виникає актуальна науково-технічна проблема, що потребує вирішення, яка може бути сформульована як задача розробки та розвитку методів побудови комбінованих обчислювальних структур для підвищення ефективності реконфігуровних сигнатурних технічних засобів захисту інформації.

Дослідженнями щодо використання ПЛІС в сигнатурних системах захисту інформації займається багато вчених. Починали цю роботу науковці з США – J.W. Lockwood, S. Dharmapurikar, V.K. Prasanna Z.K. Baker, C.R. Clark, D.E. Schimmel, Y.H. Cho, W.H. Mangione-Smith, H. Chen, Y. Chen, D.H. Summerville та ін.; продовжували вчені з Греції, Нідерландів, Великої Британії, Швейцарії та інших країн Європи – S. Vassiliadis, I. Sourdis, D.N. Pnevmatikatos, K.G. Anagnostakis, E.P. Markatos, T. Koçak, J. Lunteren та ін.; згодом долучилися азійські вчені з Індії, Ірану, В'єтнаму, Кореї та Японії, а також численні дослідники з Китаю.

В СРСР та на пострадянському просторі проблемами реконфігуровних обчислень активно займалися такі вчені, як В.І. Варшавський, Е.В. Єврєінов, А.В. Каляєв, І.І. Лєвін та ін., в Україні – В.М. Глушков, О.В. Палагин, В.М. Опанасенко, А.О. Мельник, А.М. Сергієнко, В.І. Хаханов, В.Ф. Євдокимов, В.В. Мохор, О.А. Чемерис та багато інших. Але застосуванню програмованої логіки саме для вирішення ресурсоємних задач сигнатурного

аналізу в галузі захисту інформації вітчизняними спеціалістами суттєвої уваги досі не приділялося.

Вивчення та аналіз світового досвіду й численних праць попередніх дослідників дозволили автору розробити нові та вдосконалити існуючі методи щодо створення ефективних засобів захисту інформації в комп'ютерних системах і мережах, які базуються на використанні реконфігуровних обчислювачів. Це зробило можливим вирішення поставленої науково-технічної проблеми.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, що проводились при виконанні дисертаційної роботи, здійснювалися у відповідності з планом науково-дослідних робіт Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України в рамках наступних науково-дослідних робіт:

«Дослідження та розробка методів оцінювання захищеності інформації в розподілених високопродуктивних інформаційних системах при вирішенні задач енергетики (шифр МОД-Д)», що виконувалась у 2014-2018 рр. (номер державної реєстрації 0114U002361) – створено методологію побудови систем технічного захисту інформації на базі реконфігуровних обчислювачів (відповідальний виконавець);

«Розвиток теорії, розробка новітніх інформаційних технологій в задачах комплексного моделювання та управління процесами перетворення та використання енергії (шифр НОВІНТЕХ)», що виконується з 2017 року по теперішній час (номер державної реєстрації 0117U004347) – запропоновані підходи щодо використання хмарних та грід-ресурсів для синтезу конфігурацій апаратних засобів інформаційної безпеки, здійснено дослідження та вибір апаратної платформи для реалізації сигнатурних методів технічного захисту інформації на об'єктах енергетиці (виконавець);

«Підтримка та розвиток грід-сайту Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАНУ, як ресурсного центра NGI_UA, та створення грід-сервіса централізованого синтезу конфігурацій для апаратних

прискорювачів задач інформаційної безпеки в енергетичній галузі (шифр ГРІДІПМЕМОН-15)», що виконувалась у 2015 р. згідно Цільової комплексної програми наукових досліджень НАН України «Грід-інфраструктура і грід-технології для наукових і науково-прикладних застосувань» (номер державної реєстрації 0115U002876) – запропоновано принципи централізованого синтезу конфігурацій для реконфігуровних прискорювачів задач інформаційної безпеки (відповідальний виконавець);

«Підтримка та розвиток грід-сайту Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАНУ та створення системи централізованого програмування реконфігуровних прискорювачів задач інформаційної безпеки в енергетичній галузі (шифр ГРІДІПМЕМОН-16)», що виконувалась у 2016 р. згідно Цільової комплексної програми наукових досліджень НАН України «Грід-інфраструктура і грід-технології для наукових і науково-прикладних застосувань» (номер державної реєстрації 0116U006907) – створено принципи використання грід-системи для централізованого синтезу реконфігуровних систем захисту інформації (виконавець);

«Підтримка грід-сайту Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАНУ та використання хмарної інфраструктури для централізованого програмування реконфігуровних засобів інформаційної безпеки в енергетичній галузі (шифр ГРІДІПМЕМОН-18)», що виконувалась у 2018 р. згідно Цільової комплексної програми наукових досліджень НАН України «Грід-інфраструктура і грід-технології для наукових і науково-прикладних застосувань» (номер державної реєстрації 0118U001370) – створено принципи використання хмарних обчислень для синтезу реконфігуровних систем захисту інформації (відповідальний виконавець);

«Підтримка грід-сайту ІПМЕ ім. Г.Є. Пухова НАН України та модернізація веб-сервісу централізованого програмування реконфігуровних засобів інформаційної безпеки на базі гріду та хмарної інфраструктури (шифр ГРІДІПМЕМОН-19)», що виконувалась у 2019 року згідно Програми інформатизації НАН України на 2019 р. (номер державної реєстрації

0119U001812) – розроблено методи створення ефективних структур сигнатурного розпізнавання на базі реконфігуровних прискорювачів для інформаційної безпеки (відповідальний виконавець).

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення ефективності реконфігуровних сигнатурних технічних засобів захисту інформації в комп'ютерних системах і мережах шляхом постановки та вирішення задачі комбінування різних підходів до побудови схем розпізнавання таких засобів.

Для досягнення поставленої мети в дослідженні було потрібно вирішити наступні задачі:

1. Проаналізувати проблеми технічного захисту інформації, які вирішуються сигнатурними засобами.
2. Дослідити технічні можливості сучасних ПЛІС та пристроїв на їх основі як апаратної бази для вирішення задач технічного захисту інформації.
3. Проаналізувати принципи функціонування та побудови РАСЗТЗІ як технічних систем. Сформулювати та класифікувати показники їх ефективності.
4. Виокремити ефективні підходи та їх модифікації до побудови модулів розпізнавання РАСЗТЗІ, на структурному рівні виявити та формалізувати властивості кожного з них, враховуючи специфіку реалізації реконфігуровними засобами.
5. Сформулювати та вдосконалити методи та принципи комбінування в одному пристрої декількох різних підходів із застосуванням оптимізаційних методів для використання переваг кожного з них.
6. Розробити методи прискореного виконання процедур оптимізації для вдосконалених методів комбінування з метою зменшення обчислювальної складності до прийнятних значень.
7. Застосувати розроблені методи прискореного обчислення для виявлення функціональних залежностей кількісних показників

ефективності компонентів РАСЗТЗІ, побудованих за обраними підходами та їх модифікаціями, від характеристик використаного РУО та властивостей набору патернів, що мають розпізнаватися.

8. Розробити алгоритми реалізації вдосконалених методів комбінування.
9. Розробити програмні засоби для перевірки та аналізу запропонованих методів і засобів, провести обчислювальні експерименти щодо їх кількісної оцінки.
10. Дослідити та розвити принцип централізованого програмування реконфігуровних обчислювачів з використанням високопродуктивних розподілених та хмарних середовищ з метою реалізації запропонованих методів та засобів.

Об'єктом дослідження є процеси технічного захисту інформації в комп'ютерних системах і мережах.

Предметом дослідження є РАСЗТЗІ – реконфігуровні апаратні сигнатурні засоби технічного захисту інформації в комп'ютерних системах і мережах.

Методи дослідження. При виконанні поставлених у дисертаційній роботі задач були використані загальнонаукові та спеціальні методи досліджень: положення теорії обчислювальних систем, апарат булевої алгебри, теорія автоматів, теорія обчислень на рядках, елементи теорій графів, множин, комбінаторики та алгоритмів, методи комп'ютерного моделювання.

Перевірка наукових положень та методів проводилася шляхом обчислювальних розрахунків розроблених алгоритмів за допомогою експериментальних програмних засобів а також шляхом прогонів тестових завдань на макетному зразку веб-сервісу централізованого створення реконфігуровних засобів інформаційної безпеки STRAGS, побудованого на базі кластера Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України.

Наукова новизна отриманих результатів. У дисертаційній роботі вирішена науково-технічна проблема, що полягає в розробці та розвитку методів побудови комбінованих обчислювальних структур для підвищення ефективності реконфігуровних сигнатурних технічних засобів захисту інформації. Наукова новизна одержаних результатів полягає у наступному:

1. Вперше розроблено метод прискореного обчислення технічних характеристик компонентів РАСЗТЗІ, що використовуються в якості цільових функцій процедур оптимізації, який на відміну від відомих методів проектування реконфігуровних пристроїв не потребує виконання витратних за часом процедур синтезу цифрових схем, дозволяючи за рахунок формування математичного опису здійснювати швидку оцінку та порівняння обчислювальних структур за визначеними показниками ефективності.
2. Вперше розроблено метод прискорення процедури оптимізації паралельного комбінування, який відрізняється впорядкуванням патернів в наборі за певним параметром, що дозволяє замість повного перебору комбінаторно великої кількості варіантів здійснювати поділ патернів між блоками розпізнавання, що комбінуються, за лінійним законом, послідовно змінюючи параметр впорядкування. Метод не гарантує знаходження глобального оптимуму, проте зменшує обчислювальну складність процедури оптимізації до прийнятних для практичного використання значень.
3. Вперше сформульовано та вдосконалено метод паралельного комбінування, який за рахунок паралельного з'єднання різних за принципами побудови блоків розпізнавання дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ. Запропоноване вдосконалення відрізняється застосуванням розроблених методів прискорення, що уможливлює використання процедур оптимізації за рахунок зниження часових витрат до практично прийнятних значень.

4. Вперше сформульовано та вдосконалено метод послідовного каскадування, який за рахунок послідовного з'єднання різних за принципами побудови блоків розпізнавання та уточнення попереднього розпізнавання, використовуючи процедуру оптимізації поділу між ними питомих патернів по довжині, дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ. Запропоноване вдосконалення відрізняється застосуванням розробленого методу прискореного обчислення технічних характеристик, що уможливлює використання процедур оптимізації за рахунок зниження часових витрат до прийнятних для практичного використання значень.
5. Отримав подальший розвиток метод вертикального об’єднання, який за рахунок тісного сполучання в одному блоці кількох підходів або технічних рішень дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ. Запропонований розвиток відрізняється використанням багатовимірної таблиці сумісності, що дозволяє формалізувати наявний досвід численних дослідників та спростити процедуру оптимізації вибору найбільш ефективної за наданих умов комбінації задіяних даним методом підходів або технічних рішень.
6. Вперше сформульовано та вдосконалено принцип комбінування методів комбінування, який за рахунок ієрархічного використання сформульованих, вдосконалених та розвитих методів комбінування дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ до значень, недосяжних при використанні кожного з методів комбінування окремо. Запропоноване вдосконалення відрізняється застосуванням розроблених методів прискорення, що уможливлює використання загальної процедури оптимізації в зв’язку зі зниженням часу розрахунків до прийнятних для практичного використання значень.

Практичне значення отриманих результатів полягає в тому, що застосування запропонованих методів побудови РАСЗТЗІ дозволить

розробникам створювати більш ефективні засоби захисту, яки використовуватимуться для протидії зовнішнім та внутрішнім атакам на комп’ютерні системи та мережі. Вдосконалення принципу централізованого синтезу обчислювальних структур та віддаленого програмування ПЛІС реконфігуровних обчислювачів на базі розподілених та хмарних обчислень дозволять покращити показники продуктивності та вартості обчислювальних процесів експлуатації РАСЗТЗІ. Здобуті результати щодо підвищення ефективності вирішення задачі множинного розпізнавання рядків актуальні також для мережевих застосувань, не пов’язаних з захистом інформації: прискореної обробки XML-запитів, класифікації пакетів з відновленням з'єднань для управління технологією QoS, фільтрації IP-телефонії, вимірювання трафіку, оптимізації кешування та ін. Запропоновані методи також можуть бути застосовані для вирішення важливих проблем з інших галузей науки, зокрема, інтелектуального пошуку даних (data mining), аналізу молекул ДНК тощо.

Результати дисертаційної роботи впроваджено в Національному аерокосмічному університеті ім. Н.Є Жуковського "Харківський авіаційний інститут", на приватному підприємстві "Геракс", в Національному авіаційному університеті та в Київському державному університеті ім. Т.Г. Шевченка, що підтверджується відповідними документами.

Особистий внесок здобувача. Всі основні положення й результати дисертаційної роботи, що виносяться на захист, отримано автором самостійно. У роботах, які опубліковані в співавторстві, здобувачеві належать наступні результати: [4, 18] – аналіз можливостей сучасних реконфігуровних обчислювачів в якості апаратної бази для вирішення задач інформаційної безпеки; [7, 8, 12, 39] – аналіз задач технічного захисту, які вирішуються сигнатурними реконфігуровними системами; [10, 48] – узагальнена структура РАСЗТЗІ; [11] – аналіз задачі розпізнавання рядків як обчислювально складного процесу та шляхів її вирішення реконфігуровними засобами; [16, 17] – аналіз можливостей використання некомерційних МСВВ

та їх баз даних сигнатур для оцінки ефективності РАСЗТЗІ; [20] – аналіз властивостей МСВВ щодо протидії цілеспрямованим атакам на них в якості функціонального показника ефективності РАСЗТЗІ; [23, 25, 26] – концепція та методологія централізованого синтезу реконфігуровних обчислювачів для вирішення задач захисту інформації; [33] – структура та склад реконфігуровного обчислювача; [51] – методи підвищення ефективності РАСЗТЗІ шляхом комбінування; [52] – аналіз та вибір елементної бази для реалізації апаратно-програмного комплексу.

Апробація результатів дисертації. Основні теоретичні та прикладні положення і висновки дисертаційного дослідження представлялися та обговорювалися на 16 міжнародних та всеукраїнських науково-технічних та науково-практичних конференціях і семінарах:

- Міжнародна конференція «Параллельные вычисления и задачи управления», *м. Москва, Російська Федерація*, 2004, 2006 рр.;
- Міжнародна науково-технічна конференція «ІТУЕС», *м. Київ, Україна*, 2005 р.;
- Міжнародна науково-технічна конференції «Искусственный интеллект. Интеллектуальные и многопроцессорные системы», *сел. Кацівелі, Україна*, 2006, 2008, 2011 рр.;
- Міжнародна науково-технічна конференція «ІТУЕС», *м. Київ, Україна*, 2005 р.;
- Міжнародна наукова конференція «Параллельные вычислительные технологии», *м. Санкт-Петербург, Російська Федерація*, 2008 р.;
- Міжнародна науково-технічна конференція «Многопроцессорные вычислительные и управляющие системы», *с. Дивноморське, РФ,* 2009 р.;
- Міжнародна науково-технічна конференція «ACSN-2011», *м. Львів, Україна*, 2011 р.;
- Міжнародна наукова конференція «Моделювання», *м. Київ, Україна*, 2010, 2016, 2018 рр.;

- Міжнародна науково-практична конференція «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій», *м. Запоріжжя*, *Україна*, 2018 р.;
- Міжнародна науково-технічна конференція «Високопродуктивні обчислення» (HPC-UA 2018), *м. Київ, Україна*, 2018 р.
- IV всеукраїнська науково-технічна конференція «Перспективні напрями захисту інформації», *м. Одеса, Україна*, 2018 р.
- IX міжнародна наукова конференція «Безпека інформаційних технологій» (ITSec-2019), *м. Шарм-ель-Шейх, Єгипет*, 2019 р.;
- Республіканська науково-практична конференція «Цифровые технологии в промышленности», *м. Актау, Казахстан*, 2019 р.;
- VII міжнародна наукова конференція «Захист інформації і безпека інформаційних систем», *м. Львів, Україна*, 2019 р.;
- Науково-практична конференція «Кібербезпека енергетики», *м. Одеса, Україна* 2019 р.;
- V Всеукр. наук.-практич. конф. «Перспективні напрями захисту інформації 2019», *смт Затока Одеської обл., Україна*, 2019 р.;
- Науково-практична конференція «Безпека енергетики в епоху цифрової трансформації», *м. Київ, Україна*, 2019 р.

Публікації. Наукові положення, висновки і рекомендації дисертаційного дослідження опубліковані в 52 наукових роботах, які відповідають вимогам до опублікування результатів дисертацій, у тому числі: 30 – у наукових фахових журналах та збірниках наукових праць, з яких 6 – у наукових журналах, що індексуються міжнародними наукометричними базами даних, 21 публікація у працях і матеріалах наукових конференцій, один патент, 31 публікацію підготовлено одноосібно, англійською мовою – 3 публікації.

Детальний список публікацій здобувача за темою дисертації наведено у додатку В.

Структура та обсяг дисертації. Дисертаційна робота складається з анотації, вступу, семи розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 383 сторінки, із них основного тексту дисертації – 298 сторінок, 62 рисунки, 10 таблиць, список використаних джерел включає 227 найменувань та займає 25 сторінок, обсяг додатків складає 27 сторінок.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ЗАДАЧ ЗАХИСТУ ІНФОРМАЦІЇ ТА МОЖЛИВОСТЕЙ ЇХ ВИРІШЕННЯ РЕКОНФІГУРОВНИМИ ЗАСОБАМИ

В даному розділі проаналізовано, з одного боку, проблеми технічного захисту інформації, які вирішуються сигнатурними засобами, з іншого – технічні можливості сучасних ПЛІС як апаратної бази для вирішення складних обчислювальних задач інформаційного захисту

1.1. Аналіз засобів захисту інформації

1.1.1. Основні поняття та задачі інформаційної безпеки

Інформаційна безпека (ІБ) – складна галузь діяльності людини, основними задачами якої є забезпечення [3]:

- доступності інформації;
- цілісності інформації;
- конфіденційності інформації;
- вірогідності інформації;
- юридичної значимості інформації, представленої у вигляді електронного документа;
- невідстежуваності дій користувача.

Забезпечити їх вирішення уможливлюють наступні засоби забезпечення ІБ:

- законодавчі заходи;
- адміністративні заходи (накази та інші дії керівництва організацій, пов’язаних з інформаційними системами, що захищаються);
- організаційні (процедурні) заходи (заходи безпеки, орієнтовані на людей);
- інженерно-технічні заходи.

Увесь комплекс заходів, спрямованих на забезпечення ІБ, складає поняття *захисту інформації* (ЗІ).

Перелічені вище заходи забезпечення ІБ можна розглядати як рівні або шари певної ієрархії. Досягти успіху в такої складної галузі, як ІБ, можливо тільки завдяки системному, комплексному підходу [3], вирішуючи проблеми на всіх рівнях.

В даному дослідженні розглядаються питання, що виникають на останньому зі згаданих рівнів, який в подальшому будемо називати *технічним захистом інформації* (ТЗІ).

До ключових понять ІБ належать також наступні [4], [5], [6], [7], [8].

Вразливість (Vulnerability) – притаманна об’єкту інформатизації властивість, яка, ставши відомою зловмисникові, може дозволити йому порушити безпеку інформації на цьому об’єкті. Вразливість може бути обумовлена недоліками процесу функціонування об’єкту інформатизації, особливостями архітектури автоматизованої системи, протоколами обміну, інтерфейсами, застосованим програмним забезпеченням (ПЗ), апаратною платформою, умовами експлуатації.

Загроза (Threat) – можлива небезпека (потенційна або реально існуюча) вчинення якогось діяння (дії або бездіяльності), спрямованого проти об'єкта захисту (інформаційних ресурсів), що завдає шкоди.

Атака (Attack) – реалізація загрози джерелом загрози через наявні вразливості.

Приклади класифікації зовнішніх атак наведені в роботах [9], [10].

Аналіз існуючих методів і засобів забезпечення інформаційної безпеки з метою виявлення галузей, в яких доцільно застосувати реконфігуровні засоби, а також приклади подібного використання можна знайти в роботах автора [11], [12] та [13].

Перейдемо безпосередньо до розглядання *засобів технічного захисту інформації* (ЗТЗІ).

1.1.2. Системи виявлення вторгнень

На протидію атакам цілеспрямовані спеціальні засоби ТЗІ – так звані *системи виявлення атак*, які також мають іншу назву – *системи виявлення вторгнень* (СВВ), відповідний англомовний термін – *Intrusion Detection System* (IDS).

СВВ – це програмний або апаратний засіб, який виконує безперервне спостереження за діяльністю інформаційного об’єкту, що захищаються, з метою запобігання, виявлення та протоколювання атак [14].

1.1.2.1. Мережеві системи виявлення вторгнень

В залежності від об’єкту, що захищається, розрізняють СВВ, які [14]:

- контролюють окремі комп'ютери;
- аналізують пакети трафіку локальній мережі.

СВВ другого типу використовуються частіше, оскільки ЗТЗІ орієнтовані переважно на зовнішні та внутрішні загрози інформаційно-телекомунікаційних мереж. Вони отримали окрему назву *мережеві системи виявлення вторгнень* (МСВВ), відповідний англомовний термін – *Network Intrusion Detection System* (NIDS). Більш того, в переважної більшості посилань під СВВ маються на увазі саме МСВВ.

Зауважимо, що саме МСВВ виявилися історично першими ЗТЗІ, які будувалися для масового застосування з використанням реконфігуровного прискорення [15], [16]. Тому ці засоби більш досліджені та вивчені порівняно з іншими засобами технічного захисту, що потребують високопродуктивних обчислень.

1.1.2.2. Склад МСВВ.

Типова система МСВВ згідно логіці її функціонування містить наступні елементи [7]:

- джерела даних;
- давачі;
- аналізатор;

- адміністратор;
- оператор;
- менеджер.

Джерелами даних для МСВВ є маршрутизатори, комутатори і комп'ютери локальної мережі, тобто усі компоненти, що передають, генерують і приймають інформаційні дані.

Давач копіює мережеві пакети, що циркулюють в мережі, передає їх аналізатору для виявлення підозрілої активності. Давач може являти собою окремий комп’ютер, що під'єднаних до порту комутатора/маршрутизатора, або бути програмним компонентом маршрутизатора, який має доступ до пакетів, що буферизуються на його інтерфейсі. Давач може здійснювати первинну фільтрацію пакетів, відбираючи пакети за заданими критеріями.

Аналізатор є центральним елементом МСВВ, він отримує дані від давачів і перевіряє їх на наявність ознак підозрілої активності. Аналізатор працює на основі правил, складених *адміністратором* системи безпеки підприємства відповідно до політики безпеки. При виконанні умови одного з правил аналізатор формує повідомлення тривоги і передає його *менеджеру* – компоненту, який зберігає конфігурацію МСВВ і підтримує зручний інтерфейс з оператором. Менеджер МСВВ сповіщає оператора про тривогу за допомогою певного повідомлення.

Оператор МСВВ на основі даних повідомлення приймає рішення про реакцію на підозрілу активність – це може бути відключення мережевого каналу, через який надходить підозрілий трафік, зміна правил міжмережевого екрану або ігнорування повідомлення, якщо оператор приходить до висновку, що сталася помилка розпізнавання другого роду. У будь-якому випадку, дані про потенційне вторгнення протоколюються в журналі менеджера і можуть бути використані згодом для повторного аналізу ситуації.

1.1.2.3. Системи запобігання вторгнень

Функції компоненту Оператор, який був розглянутий вище, можуть виконуватися в ручному режимі людиною, або автоматично – відповідними засобами системи виявлення вторгнень. В останньому випадку говорять, що СВВ функціонує в режимі запобігання вторгнень, тобто як *система запобігання вторгнень* (СЗВ), відповідний англомовний термін – *Intrusion Prevention Systems* (IPS). Іноді під СЗВ розуміють окремий клас ЗТЗІ, що може використовуватися поруч / замість СВВ. Подібно до мережевого варіанту СВВ використовують термін *мережева система запобігання вторгнень* (МСЗВ).

Згідно іншому визначенню СЗВ відрізняються від СВВ тим, що генерують активний відгук на виявлені випадки зловмисної активності, в той час як останні реагують на такі інциденти пасивно [17], [10].

До активних дій належать:

- припинення сеансу роботи користувача;
- призупинення роботи користувача;
- блокування облікового запису користувача;
- ініціювання додаткової автентифікації;
- блокування IP-адреси;
- вимкнення атакований порт сервісу;
- попередження зловмисника;
- спостереження за з'єднанням;
- використовування тимчасових "тіньових" файлів;
- відключення від мережі;
- відключення джерела трафіку.

До пасивних:

- генерація тривоги;
- увімкнення додаткового ідентифікатору;
- створення звіту;
- ініціювання ведення додаткового журналу;

- ініціювання ведення віддаленого журналу;
- створення резервної копії даних.

Внаслідок важливості дій, що виконують СЗВ, до них висуваються більш високі вимоги щодо швидкісних характеристик.

Оскільки в даному дослідженні СЗВ розглядаються як підклас систем класу СВВ, в подальшому будемо розуміти що під поняттям СВВ / МСВВ системи як виявлення, так і запобігання вторгнень.

1.1.2.4. Різновиди МСВВ

За останні роки запропоновано велику кількість не тільки різних підходів до побудови СВВ, але й напрямів щодо їх класифікації [18], [19], [20], [21]. На жаль досі складно виокремити якусь одну класифікацію, яка булла би строгою та несуперечливою.

Тим не менш, не претендуючи на строгість цього твердження, всі розробки систем виявлення вторгнень можна умовно поділити на два підкласи, що чітко відрізняються. До першого підкласу належать системи, що використовують сигнатури (signature-based IDS), які також часто визначаються як таки що виявляють зловживання. До другого – такі, що виявляють аномальність поведінки мережевих об'єктів (anomaly-based IDS).

Головний недолік сигнатурних СВВ полягає в тому, що вони не здатні розпізнавати нові атаки, для яких ще не створені сигнатури. Тому системи виявлення аномалій активно розвиваються останнім часом. В сучасних розробках, що базуються на виявленні аномалій, використовують таки підходи та класи підходів, як: машинне навчання (нечіткі множини, метод опорних векторів, штучні нейронні мережі); добування даних – data mining (кластерний, класифікаційний); використання баз знань (експертний, на базі графів сценаріїв або контролю змін подій); статистичний; поведінковий; евристичний. На жаль, не зважаючи на значний прогрес та певну результативність, досягнуті у цьому напряму, рішенням подібного плану все ще притаманні такі недоліки, як висока інтенсивність помилок розпізнавання

(як першого, так і другого роду), складність процесу налаштування, довготривалість навчання та створення відповідного профілю нормального стану [22]. Тому в реальних розробках на сьогоднішній день все ще привалюють сигнатурні СВВ.

1.1.2.5. Сигнатурні засоби технічного захисту інформації

Оскільки перелічені вище недоліки СВВ, що використовують виявлення аномалій, стримують їх практичне застосування в більшості прикладних випадків, в даному дослідженні розглянуті саме засоби захисту на базі сигнатур – сигнатурні засоби технічного захисту інформації (СЗТЗІ)

1.1.2.6. Функціонування сигнатурної МСВВ

Механізм функціонування сигнатурної мережевої системи виявлення вторгнень в загальному випадку складається з трьох етапів [23], [16]:

- захоплення мережевих пакетів (packet capture);
- фільтрація та збирання пакетів (filtering / fragmentation reassembly);
- розпізнавання (pattern matching).

Найбільш ресурсоємним є останній етап, який потребує великого обсягу роботи з аналізу вмісту мережевих пакетів.

Аналіз мережевого трафіку може здійснюватися двома способами:

- шляхом тотального інспектування всіх пакетів мережевого трафіку без додаткової обробки (raw);
- з урахуванням мережевих протоколів – так званий stateful-підхід, заснований на аналізі заголовків мережевих пакетів з метою повного відновлення сеансів мережевих комунікацій.

Системи, засновані на першому способі, розпізнають більше атак. Для них не є проблемою нестандартні номери портів, втрачені або умисно спотворені порушником мережеві пакети. З іншого боку, такі МСВВ набагато більш ресурсоємні в своїй реалізації. Другий спосіб аналізу економнішій щодо обробки тіл пакетів, до того ж дозволяє виявляти довгі інформаційні послідовності, які перетинають межі пакетів. Але такій підхід

потребує більш складних механізмів поводження з заголовковими полями пакетів.

1.1.2.7. Співвідношення між МСВВ та міжмережевими екранами. Поняття глибокого аналізу пакетів

Іншим засобом технічного захисту інформації, спрямованим на протидію зовнішнім атакам на локальну обчислювальну мережу (ЛОМ), є міжмережевий екран (ММЕ). Інші назви – фаєрвол чи файрвол (від англ. firewall), брандмауер, мережевий екран. Такий ЗТЗІ, використовуючи адресну інформацію, що міститься в заголовках мережевих пакетів, та керуючись переліком так званих *правил* (сформованих адміністратором під час налаштування) здатен відбивати більшість зовнішніх атак на локальну мережу, що захищається. Але в деяких випадках зловмисник здатен обійти цей захист і проникнути в захищений об'єкт. Саме на такі випадки розраховане застосування мережевих систем виявлення вторгнень сигнатурного типу. Отже МСВВ доповнює функції ММЕ та підмінює його функції, коли останній не впорається із захистом [24].

Підвищена здатність МСВВ сигнатурного типу щодо захисту порівняно з ММЕ об’єктивно обумовлена тим, що цій ЗТЗІ аналізує не тільки заголовки, а й тіла мережевих пакетів, тобто, виконує так звану глибоку обробку пакетів, відповідний англомовний термін – Deep Packet Inspection (DPI).

З невідомих автору причин поняття глибокої обробки пакетів практично не використовується у вітчизняному мовному просторі, тоді як термін DPI є широко розповсюдженим в англомовній літературі щодо захисту інформації в комп'ютерних мережах.

Глибока обробка пакетів, яку здійснює МСВВ, потребує значно більше обчислювальних витрат порівняно з функціональністю ММЕ. По-перше, це пов’язано з великим розміром мережевих пакетів, по-друге, обумовлено тим

фактом, що зміст пакету не має чіткого формату, як його заголовок, та дуже відрізняється в залежності від застосування [25], [26], [27].

Слід зауважити, що коло пристроїв, здатних виконувати DPI-функції, постійно розширюється. З іншого боку, можливості засобів ТЗІ постійно збільшуються, їх функціональність дедалі більше перетинається та частково дублюється у виробах різних класів. Більш того, з'являються, реалізуються та розвиваються концепції щодо об'єднання різнопланових захисних функцій в єдиній системі чи навіть в одному пристрої. Подібні тенденції призвели, зокрема, до появи таких рішень, як UTF (Unified Threat Management) – для середнього бізнесу чи NGFW (Next Generation Firewall) – для великих підприємств. Тому в подальшому, щоб уникнути плутанини, під термінами МСВВ, ММЕ та іншими назвами окремих засобів захисту інформації будемо розуміти насамперед функціональність, яку вони реалізують, ніж конкретні вироби або класи виробів [28], [29].

1.1.2.8. Некомерційні МСВВ та їх сигнатурні бази даних

В даний час існує велика кількість комерційних реалізацій систем виявлення вторгнень. В роботі [21] проаналізовано та наведено відомості щодо наступних розробок: Shadow; Cisco IPS; Arbor Networks Spectrum; InfoWatch ASAP; Symantec DeepSight Threat Management System; IPS; Tipping Poing NGIPS; Axoft invGUARD; DefensePro; KATA Platform.

Водночас для проведення наукових досліджень більш цікавими виявляються некомерційні МСВВ, що розповсюджуються вільно. Таких розробок на сьогодні також створено чимало. Найвідомішими з них є Zeek (у минулому – Bro) [30], Hogwash [31], Snort [32] та Suricata [33].

Завдяки своєї доступності некомерційні МСВВ зазвичай використовуються в якості універсального засобу оцінки досягнень дослідників.

В якості стандарту де-факто більшість розробників досі використовує МСВВ Snort і її сигнатурну базу даних, що дозволяє з високим ступенем

об'єктивності порівнювати між собою кількісні показники ефективності розробок різних авторських колективів [34], [35], [36].

1.1.2.9. Поняття сигнатури та патерну. Склад сигнатури МСВВ

Перед визначенням важливих для даної роботи понять "сигнатури" та "патерну" розглянемо структуру записів (правил) бази даних сигнатур МСВВ Snort [36].

Синтаксично запис бази даних Snort являє собою набір з семи обов'язкових атрибутів, після яких в круглих дужках через крапку з комою прямує в довільному порядку перелік необов'язкових опцій, що поділяються на чотири категорії. У загальному випадку правило виглядає наступним чином:

<тип> <протокол> <адр1> <порт1> <напрям> <адр2> <порт2> ([заголовки] [зміст] [доп. дії] [загальні])

Обов'язковий атрибут <тип> вказує, яке основне дію виконують в разі відповідності пакета критеріям правила [34], наприклад, ігнорувати пакет, блокувати пакет, записати його в журнал реєстрації і видати попередження тощо.

Атрибут <протокол> визначає приналежність пакета до конкретного мережевого протоколу.

Атрибути <адр1> і <адр2> задають IP-адреси (або діапазони адрес) джерела та приймача пакета відповідно.

Атрибути <порт1> і <порт2> задають номери портів (або діапазони портів) джерела і приймача пакета відповідно. Зрозуміло, що ці атрибути мають сенс тільки для протоколів транспортного рівня.

Атрибут <напрямок> визначає, чи слід аналізувати тільки пакети, що передаються від вузла з адресою <адр1> до вузла з адресою <адр2>, або в обох напрямках.

В базі Snort налічується біля сотні компонент опційної частини сигнатури, які розділені між чотирма категоріями. Не вдаючись в подробиці розглянемо приклад запису, призначеного для виявлення конкретної атаки. Сигнатура, яка розпізнає атаку, що використовує вразливість типу «переповнення буфера» в програмних продуктах Microsoft SQL Server 2000 Resolution Service та Microsoft Desktop Engine 2000 [37] в базі даних Snort, має вигляд:

alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"SQL Worm propagation attempt"; flow:to_server; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|"; fast_pattern:only; content:"sock"; content:"send"; metadata:ruleset community; reference:bugtraq,5310; reference:bugtraq,5311; reference:cve,2002-0649; reference:nessus,11214; reference:url,vil.nai.com/vil/content/v_99992.htm; classtype:misc-attack; sid:2003; rev:15;)

Атрибут "udp" в даній сигнатурі вказує, що потрібно перевіряти тільки пакети, що передаються по протоколу UDP. Вирази "$EXTERNAL_NET" та "$HOME_NET" використовуються для завдання діапазонів адрес джерела та одержувача пакетів. Атрибут "any" означає, що номера портів джерела пакетів можуть бути довільними. Символ "->" в даному випадку означає, що аналізуються тільки пакети, що надсилаються з зовнішнього світу у внутрішню мережу. Значення 1434 вказує аналізувати лише пакети, що посилаються на порт номер 1434 одержувача.

Серед чисельних опційних параметрів правила звернемо увагу на наступні параметри:

- content:"|81 F1 03 01 04 9B 81 F1 01|";
- content:"sock";
- content:"send".

Перший з них наказує системі шукати в змісті пакету послідовність байтів, що мають значення "81", "F1", "03", "01", "04", "9B", "81" і "F1 за шістнадцятирічним кодуванням. Два наступних – послідовності символів "sock" і "send" за кодуванням ASCII відповідно.

Отже, під поняттям *сигнатура* будемо розуміти сукупність інформації про конкретну загрозу, що зберігається в сигнатурній базі даних відповідного СЗТЗІ.

У випадку МСВВ Snort цій термін позначає низку ознак однієї загрози, які зберігаються у відповідному запису її бази даних. Тобто, в даному випадку термін "сигнатура" є еквівалентним поняттю "запис бази даних". В подальшому ми також будемо використовувати ці два терміни як синоніми, якщо не буде явно вказано інше.

Усю сукупність сигнатур, якою володіє СЗТЗІ називатимемо *базою даних сигнатур*. Зауважимо, що МСВВ в загальному випадку використовує не всі записи наявної бази даних. В більшості застосувань в кожній конкретній ситуації в процесі розпізнавання ознак зловмисної активності задіяна лише підмножина відомих описів атак. Наприклад, якщо на жодному з комп'ютерів, що входять до складу захищеної ЛОМ, не використовуються Windows-сумісні операційні системи (ОС), не має сенсу витрачати ресурси на пошук сигнатур, пов'язаних з атаками на вразливості цих ОС. Тому системні адміністратори в процесі налаштування МСВВ відключають відповідні записи бази даних сигнатур.

Під терміном *патерн* будемо розуміти зразок текстового рядка (фіксовану послідовність символів, закодовану байтами), що входить до складу сигнатури, який відшукується в тілі мережевого пакету, що аналізується.

У загальному випадку сигнатура містить довільну кількість патернів довільної довжини кожний, допоміжні правила, що регламентують умови їх пошуку, а також правила, за якими обробляються заголовки мережевих пакетів на випадок аналізу пакетів в режимі урахування мережевих

протоколів, тобто в stateful-режимі (див. п. 1.1.2.6 "Функціонування сигнатурної МСВВ").

Під терміном *словник патернів* розумітимемо всю сукупність патернів, що місяться загалом у базі даних сигнатур.

Терміном *набір патернів* позначимо низку патернів, що місяться тільки в деяких сигнатурах бази даних МСВВ, обраних з якоюсь метою. (Наприклад, в реальних МСВВ рідко використовуються всі сигнатури бази даних відомих атак. Якщо в локальній мережі відсутні, наприклад, FTP-сервери на Linux-подібної платформі, патерни зі складу сигнатур відповідних атак немає сенсу включати до набору, що має розпізнаватися).

Терміном *підгрупа патернів* будемо називати частину набору патернів, яку має розпізнавати окремий блок розпізнавання, що поряд з іншими блоками входить до загального модулю розпізнавання згідно використаному методу комбінування).

Термін *множина патернів* в подальшому використовуватиме для позначення абстрактної низки патернів, тобто як синонім обох наведених вище визначень, але використаний в математичному сенсі, не акцентуючи увагу на практичному застосуванні.

Слід зауважити, що крім розпізнавання фіксованих послідовностей, коли СЗТЗІ виявляє строгий посимвольний збіг фрагменту вхідної послідовності (тексту) з патерном (рядком), бази даних сигнатур фактично всіх сучасних засобів захисту містять також деяку кількість багатозначних описів атак, що дозволяють варіювати зразки певним чином, зазвичай використовуючи для цього механізм регулярних виразів, відповідний англомовний термін – RegEx (Regular Expression). На відміну від процесу виявлення фіксованих патернів, який будемо називати *точне розпізнавання*, використання зразків з варіаціями будемо іменувати *гнучке розпізнавання*.

1.1.3. Інші сигнатурні засоби технічного захисту інформації

Як було зазначено у підр. 1.1.2.1 "Мережеві системи виявлення вторгнень", мережеві СВВ були історично першими реконфігуровними засобами захисту інформації, які набули масового поширення. Але також існують інші ЗТЗІ, які використовують сигнатурний підхід до аналізу вхідної інформації. Нижче розглянуті особливості та відмінності декількох класів СЗТЗІ, які не є системами виявлення вторгнень. За принципами функціонування вони суттєво не відрізняються від сигнатурних МСВВ. Тому в подальшому, не втрачаючи загальності, в якості СЗТЗІ будемо розглядати саме мережеві системи виявлення вторгнень, якщо явно не вказане інше.

1.1.3.1. Антивірусні сигнатурні системи

Комп'ютерний вірус – це вид шкідливого програмного забезпечення, здатного проникати в код інших програм, системні області комп'ютера, зберігаючи при цьому здатність до подальшого розмноження [38], [39].

Головні відмінності *антивірусних сигнатурних систем* (АСС) [40] від МСВВ обумовлені наступними чинниками. По-перше, розмір бази даних сигнатур сучасних антивірусних систем, і, як наслідок, кількість патернів, що мають розпізнаватися, зазвичай на декілька порядків перевищує словник патернів МСВВ [41]. До того ж зменшити цю кількість в конкретній ситуації за рахунок відключення неактуальних сигнатур на відміну від МСВВ не вдається. По-друге, зловмисне ПЗ, яке відноситься до класу вірусів, свою шкідливу функцію починає здійснювати безпосередньо перед початком атаки, знаходячись локально на атакованому об’єкті. До цього моменту тіло вірусу може бути зашифровано, що унеможливлює його виявлення сигнатурними засобами. Наведені чинники призводять до недоцільності виявлення вірусів в пакетах мережевого трафіку ЛОМ шляхом використання МСВВ, просто додавши антивірусні сигнатури до її бази даних. Натомість апаратно-програмні АСС, побудовані за сигнатурним принципом, успішно використовують локально на об'єкті, що захищається [42].

Зауважимо, що подібно до МСВВ, для об'єктивної оцінки та порівняння технічних рішень противірусних СЗТЗІ дослідники використовують бази даних сигнатур некомерційних антивірусних додатків, наприклад, ClamAV [43].

1.1.3.2. Засоби протидії мережевим хробакам

Мережеві хробаки на відміну від вірусів не проникають в код інших програм або системні області комп'ютерів, але є самостійними програмами, що розмножуються і поширюються в локальних і глобальних комп'ютерних мережах, знижуючи тим самим пропускну здатність каналів зв'язку аж до повної їх непрацездатності [44], [45]. При цьому процес зараження в залежності від типу хробака варіюється від декількох годин до десятків секунд [46].

З цих причин *засоби протидії мережевим хробакам* (ЗПМХ) мають діяти дуже швидко і активно, подібно до систем запобігання вторгнень. Тому в даних СЗТЗІ на відміну від МСВВ та антивірусів часто сигнатурний принцип доповнюють іншими. Наприклад, відомі підходи, при яких протихробакові засоби діють подібно системам, заснованим на аналізі аномалій, щоб виявити факт появи великої кількості однакових послідовностей байтів у мережевому трафіку [47]. Деякі з ЗПМХ здатні навіть автоматично створювати сигнатури на основі виявлення великої кількості фрагментів даних, що повторюються [48], [46].

1.1.3.3. Сигнатурні спам-фільтри

Під терміном *спам* (від англ. spam) в загальному розумінні позначають повідомлення, що розсилаються масово одержувачам, які не мали наміру їх одержувати. У першу чергу ще явище стосується електронних листів. Один з напрямів протидії спаму пов'язаний зі створенням *сигнатурних спам-фільтрів* (ССФ), найближчий англомовний термін – *Signature-based Collaborative Spam Detection* (SCSD).

В сенсі практичного застосування важливим негативним наслідком спаму є виникнення в роботі спам-фільтрів помилок розпізнавання другого роду, відповідний англомовний термін false positive, в результаті чого легальні повідомлення хибно помічаються як спам і не потрапляють до одержувача. Сигнатурні засоби позбавлені цього недоліку, хоча їх рівень виявлення поступається іншим методам протидії спаму [49], [50].

Порівняно з іншими СЗТЗІ ССФ мають бази даних сигнатур навіть більші за ті, що використовуються в антивірусних системах [51]. Тому проблема підвищення ефективності таких засобів також є актуальною.

1.2. Задача множинного розпізнавання патернів

В процесі функціонування всі сигнатурні засоби технічного захисту інформації виявляють у вхідних даних ознаки зловмисної активності, які задані у вигляді сигнатур. В загальному випадку сигнатури можуть мати довільну природу, склад та внутрішню організацію. Але в переважній більшості застосувань незалежно від принципів їх побудови, сигнатури містять патерни (див. підр. 1.1.2.9 "Поняття сигнатури та патерну. Склад сигнатури МСВВ"). Причому кожна сигнатура може включати кілька патернів різної довжини. Оскільки бази даних сигнатур сучасних СЗТЗІ налічують велику кількість записів, сигнатурні засоби ТЗІ повинні вирішувати в реальному часі складну обчислювальну задачу одночасного виявлення великої кількості фіксованих послідовностей символів. Розглянемо цю задачу докладніше.

1.2.1. Формулювання задачі множинного розпізнавання патернів

В англомовній літературі зустрічається декілька найменувань функції, які здійснюється в процесі вирішення згаданої задачі:

- "multiple pattern matching";
- "multi-pattern matching";
- "multi-pattern signature matching";

- "multi-pattern string matching";
- "multi-string matching".

В монографії [52] (яка має російськомовний переклад [53]) відповідна задача має назву *The exact set matching problem*. Але в наукових публікаціях щодо практичної реалізації сигнатурних засобів інформаційного захисту найчастіше використовується вираз *multi-pattern matching*.

Головна складність в створенні вітчизняного аналога для даного поняття полягає у відсутності однозначного перекладу англійського дієслова "to match" та його похідних. Власне слово "to match" має такі тлумачення: підбирати (під пару, під стать; за кольором, формою тощо); знаходити відповідність; узгоджувати; вирівнювати; підганяти. Іменник "matching", що позначає процес, перекладається як узгодження; вирівнювання; підгонка, підбір; зіставлення. Словосполучення "pattern matching" словники тлумачать як співставлення або порівняння з еталоном чи зразком.

Деякі перекладачі в якості аналога виразу "matching" використовують слово "пошук". Але цей термін більш доцільно застосовувати для перекладу англомовних термінів "finding", "searching", "seeking" або "looking up". До того ж операція пошуку більш притаманна використанню в галузях алгоритмічного забезпечення та інформаційних технологій.

В галузі захисту інформації в контексті виявлення зловмисної активності дія, про котру йде мова, більш схожа на таку, що виконується в галузі технічної діагностики в процесі розпізнавання той чи іншої несправності в системі. В цьому сенсі в якості перекладу поняття "pattern matching" більш придатним уявляється термін "розпізнавання" або "виявлення".

Тому в даному дослідженні в якості аналога англомовному поняттю "multi-pattern matching" використовуватиметься найближчий за смислом згідно думки автора вираз *множинне розпізнавання патернів*. Відповідно сама задача матиме назву *задача множинного розпізнавання патернів* (ЗМРП).

Сформулюємо задачу ЗМРП в математичній постановці.

У згаданій монографії ця задача формулюється як узагальнення задачі точного розпізнавання (match): "An immediate and important generalization of the exact matching problem is to find all occurrences in text T of any pattern in a set of patterns $P = \{P_1, P_2..., P_z\}$" (стор. 53 у [52]).

Отже, якщо ми маємо текст T, який складається з записаних в рядок n символів: "$C_1C_2C_3\ldots C_n$", та множину патернів $P = \{P_1, P_2..., P_z\}$, де Z – її потужність, а кожний патерн P_i (i = 1, 2, … Z) складається з записаних в рядок m символів: "$C_1C_2C_3\ldots C_m$", де m – довжина відповідного патерну (різна для кожного з них в загальному випадку), то ЗМРП полягає у виявленні всіх входжень в тексті T будь-якого з патернів P_i з множини P.

У більш детальному та чіткому формулюванні задача множинного розпізнавання патернів полягає у наданні відповідей на наступні питання:

- чи є в складі тексту T підрядки, які посимвольно збігаються з якимись з патернів P_i;
- якщо є – з якими саме;
- скільки збігів для кожного з патернів виявляється;
- в якій позиції (відносно початку тексту T) міститься кожний з підрядків, що збігаються.

Зауважимо, що послідовності символів, що відповідають патернам у вхідних даних, можуть не тільки повторюватися, але й частково взаємно перекриватися, тобто кінець підозрілого підрядка може збігатися с початком іншого. Але такі ситуації також мають виявлятися в процесі вирішення ЗМРП [54].

Перехід від математичної постановки до фізичної характеризується додаванням певних властивостей, притаманних реальному світові. У випадку ЗМРП така властивість має сенс часу життя. Однакові в математичній постановці текст T та множина патернів P у фізичному світі змінюються в суттєво різному темпі. Якщо вхідні дані, що подаються на сигнатурний засіб захисту в загальному випадку заздалегідь абсолютно невідомі, тобто

змінюються в ритмі надходження, то словник патернів є відносно постійним: його склад оновлюється з певною періодичністю, час якої варіюється від годин для антивірусних сигнатурних систем до тижнів і навіть місяців для МСВВ та засобів протидії мережевим хробакам. Цей факт дозволяє заздалегідь виконувати зі словником патернів деякі попередні маніпуляції, що сприятимуть виконанню процедури розпізнавання у подальшому. Така можливість активно використовується розробниками СЗТЗІ для підвищення їх ефективності.

При практичної реалізації задачі ЗМРП технічними засобами слід брати до увазі той факт, що бази даних сигнатур сучасних СЗТЗІ місять від десятків тисяч до сотень тисяч і навіть мільйонів патернів [36], [55], [41]. З іншого боку, задачу потрібно встигати вирішувати "на льоту" (on-the-fly) тобто у темпі надходження даних, які пересилаються в сучасних комунікаційних каналах зі швидкістю в десятки та сотні гігабайтів на секунду. Тому бажано обробляти дані за один прохід з продуктивністю, яка не залежить від змісту інформації, що аналізується [56].

1.2.2. Алгоритми одиночного розпізнавання

Задача виявлення одиночного підрядка в більш довгому рядку символів (тексті) відома в комп'ютерної техніці майже с початку її використання. За цей час розроблено та вдосконалено багато алгоритмів, що вирішують дану задачу для різних застосувань. До найбільш поширених з них можна віднести, наприклад, алгоритми Кнута–Морріса–Пратта [57], Бойера–Мура [58], Карпа–Рабіна [59] та їх численні модифікації – див., наприклад, монографію [60] або її російськомовний переклад [61]. Однак більшість з цих алгоритмів орієнтовано на програмну (послідовну) реалізацію в однопроцесорних системах. Але, що більш важливо, дані алгоритми є однопатерними за своєю природою, тобто орієнтовані на одночасне розпізнавання тільки одного підрядка. Використання таких алгоритмів для вирішення ЗМРП шляхом послідовного застосування по черзі окремо для

кожного патерну не є ефективним з деяких причин, в тому числі розглянутих в наступному підрозділі.

1.2.3. Ефект самоподоби. Префікси, інфікси, суфікси

Якщо виконувати порівняння вхідного тексту одночасно з декількома зразками з множини патернів, виявляється важлива властивість, що притаманна базам сигнатур практично всіх сучасних СЗТЗІ: патерни в еталонному наборі частково збігаються та повторюють один одного. Тобто початок (префікс), середня частина (інфікс) або кінець (суфікс) одного патерна може збігатися з деякою частиною (префіксом, інфіксом або суфіксом) іншого.

Назвемо це явище *ефектом самоподоби*.

Використання потрібним чином даного ефекту дозволяє підвищити продуктивність систем розпізнавання. Згадані вище алгоритми одиночного розпізнавання непридатні для цього. З іншого боку в теорії та практиці поводження з рядками з'явився і активно розвивається окремий клас багатопатерних алгоритмів, тобто алгоритмів множинного розпізнавання рядків [62].

1.2.4. Алгоритми множинного розпізнавання

Одним з перших алгоритмів множинного розпізнавання став алгоритм Ахо–Корасік [63]. Розробка виявилася вдалою і стала прототипом для численних модифікацій і доопрацювань, більшість з яких орієнтована на апаратну реалізацію. Суть даного алгоритму полягає в побудові за певними правилами цифрового автомата, на вхід якого в процесі розпізнавання послідовно, символ за символом, надходять дані. В залежності від їх вмісту автомат переходить з одного стану в інший, поки не опиниться в стані, що сигналізує про виявлення якогось еталонного підрядка.

У певному сенсі алгоритм Ахо–Корасік можна вважати багатопатерною модифікацією алгоритму Кнута–Морріса–Пратта. Аналогічно алгоритм множинного розпізнавання Комменц-Вальтер [64] є

варіантом алгоритму Бойєра–Мура. Незважаючи на деякі переваги алгоритму Бойєра–Мура в порівнянні зі схемою Кнута–Морріса–Пратта, алгоритм Комменц-Вальтер для апаратної реалізації використовується рідше, ніж Ахо–Корасік.

Відомі й інші алгоритми множинного розпізнавання різного призначення, такі як алгоритм Ву–Манбера, модифікований алгоритм Ву–Манбера, Домёлкі–Бейза-Ятса–Гонета, SBMH (Set-wise Boyer-Moore-Horspool), RKBT (Rabin-Karp with binary search and two-level hashing), FNP/FNP2, SOG, BG [65], [66], [67], [68], [60], [69], [70], [40].

1.2.5. Специфіка реалізації множинного розпізнавання

Жорсткі вимоги щодо швидкодії засобів, що вирішують ресурсоємну ЗМРП, призводять до необхідності застосовувати розпаралелювання обчислювального процесу. Тому крім алгоритмічних підходів до побудови СЗТЗІ, що реалізують дану задачу, дослідники широко використовують суто апаратні, схемотехнічні методи створення таких засобів.

Докладніше апаратну реалізацію СЗТЗІ розглянуто у наступних підрозділах.

1.3. Апаратне прискорення

1.3.1. Складності вирішення ЗМРП традиційними засобами

Задача множинного розпізнавання, що була розглянута у попередньому підрозділі, яку вирішують сигнатурні засоби ТЗІ є обчислювально складною. Її здійснення традиційними обчислювальними засобами стає дедалі складнішим. По-перше, бази даних сигнатур сучасних СЗТЗІ місять від десятків до сотень тисяч і навіть мільйонів патернів [36], [55], [41], і їх об’єми продовжують збільшуватися внаслідок зростання кількості та витонченості зловмисної активності. По-друге, пропускна здатність каналів передачі даних також постійно зростає [71]. Відповідно зростають обсяги

інформації, що пересилаються. Як наслідок, СЗТЗІ мають обробляти більшу кількість даних за одиницю часу.

По-третє, біля десятиріччя тому внаслідок досягнення технологічної межі зупинилося стале зростання частоти мікропроцесорних виробів, яке тривало біля 40 попередніх років.

Всі ці чинники призвели до того, що традиційні технічні рішення, які реалізовуються програмними засобами на однопроцесорних пристроях, вже не здатні впоратися з вирішенням ЗМРП внаслідок браку швидкодії та продуктивності. Тому для її виконання потрібно використовувати інші обчислювальні платформи.

В наступних підрозділах проаналізовані напрями можливого вирішення проблеми.

1.3.2. Суперкомп'ютери, кластери, грід та хмарні обчислення

Для вирішення ресурсоємних задач призначені декілька класів обчислювальної техніки, об'єднані загальним поняттям *високопродуктивний комп'ютінг*, відповідний англомовний термін – High Performance Computing (HPC). Суперкомп'ютер, з яких починалася історія створення засобів сектору HPC згодом доповнилася класом кластерних рішень, які за рахунок використання стандартизованих компонентів дозволили суттєво знизити загальну вартість володіння технічних рішень [72]. Потім з'явилися и набули широкого розповсюдження розподілені платформи, такі, як грід. Продовжуючи тенденцію на зниження вартості володіння відносно недавно почали активно розвіватися хмарні обчислення.

На жаль, головним недоліком всього сегменту HPC-обчислень є віддаленість обчислювальних потужностей від місця споживання результатів розрахунків. Великі обсяги та інтенсивність надходження інформації, яку потрібно обробляти в процесі вирішення ЗМРП не дозволяють використовувати згадані вище високопродуктивні рішення. Інакше кажучи,

локальна природа задач технічного захисту інформації перешкоджає їх вирішенню обчислювальними ресурсами сегменту НРС.

1.3.3. Апаратне прискорення сигнатурних засобів ТЗІ

Водночас виникло і активно розвивається напрям в обчислювальній галузі, позбавлений цього недоліку. Він пов'язаний з використанням приєднаних обчислювачів (прискорювачів), які фізично розташовуються безпосередньо в місці вирішення задачі, тому не потребують пересилання великих обсягів даних на значну відстань.

Принципи побудови та можливості таких пристроїв значно розрізняються. Але головна риса, яка об'єднує їх розмаїття – це використання апаратних рішень замість програмних, оскільки головне призначення подібних пристроїв полягає в подоланні обмежень, з якими стикнулися програмні рішення, як було зазначено вище.

Підсумовуючі сказане, приходимо до висновку, що природним шляхом подолання складнощів вирішення ЗМРП традиційними однопроцесорними програмними засобами, сформульованих в підр. 1.3.1 "Складності вирішення ЗМРП традиційними засобами", є використання апаратних рішень. Тому коло пристроїв, які є предметом даного дисертаційного дослідження звужується до такого, що має назву *апаратні сигнатурні засоби технічного захисту інформації* (АСЗТЗІ).

1.3.4. Уніфіковані обчислювачі

Автором був проведений ретельний аналіз напряму, заснованому на приєднаних апаратних пристроях, якому надано назву *уніфіковані обчислювачі* (УО), оскільки саме однаковість та стандартизація дозволили даному сегменту обчислювальної техніки набути розповсюдження за рахунок зниження загальної вартості володіння. Взявши за основу класифікацію УО, наведену в роботах [73] та [74], з урахуванням специфіки мережевих задач, отримуємо наступний перелік УО, потенційно здатних вирішувати ЗМРП:

- спеціалізовані сопроцесори;

- нейромережеві прискорювачі;
- мережеві процесори;
- прискорювачі на базі багатоядерних процесорів (Multi-Core);
- графічні акселератори;
- реконфігуровні обчислювачі на базі ПЛІС.

Вироби, що належать до останньої групи наведеної класифікації УО отримали окрему назву *реконфігуровні уніфіковані обчислювачі* (РУО).

Розглянемо перелічені категорії УО на предмет здатності вирішення ЗМРП з метою подальшого порівняння.

1.3.5. Існуючи апаратні платформи

Проаналізуємо апаратні платформи, які можуть бути використаними для побудови АСЗТЗІ.

1.3.5.1. Спецпроцесори

В спеціалізованих апаратних сопроцесорах за рахунок звуження кола задач, що вирішуються, досягається максимальне прискорення, яке в принципі здатна забезпечити мікроелектронна технологія на даний час. В якості елементної бази в сучасних спец процесорах використовують напівзамовні надвеликі інтегральні схеми (НВІС), найближчий англомовний термін – Application-Specific Integrated Circuit (ASIC). Технологія НВІС фактично є вершиною еволюційного розвитку напівпровідникових схем, починаючи з приладів малої та середньої інтеграції, електричні ланцюги яких розроблялися наново для кожного застосування аж до складних виробів нанометрового технологічного процесу на базових матричних кристалах, які дозволяють розробникам синтезувати надскладні засоби обчислювальної техніки, максимально використовуючи при цьому багатий досвід, накопичений в галузі створення цифрових схем.

Створення спеціальних апаратних пристроїв для вирішення задач інформаційної безпеки, зокрема ЗМРП, на базі мікросхем ASIC цілком можливо, и такі розробки існують. За рахунок спеціалізації подібні рішення

мають найкращі показники швидкодії та енергоспоживання порівняно з іншими технологіями. Але виробництво кожної НВІС є ресурсоємним процесом як с точки зору вартості, так і за часовими витратами. Фіксована внутрішня структура ASIC не дозволяє вносити зміни у вже виготовлені вироби. Строго кажучи, спецпроцесори на базі НВІС не є УО, оскільки вироби на "жорсткій" логіці за визначенням не можуть бути універсальними та уніфікованими. Тому використання спецпроцесорів на базі ASIC є доцільним за умови достатньо великих тиражів виробництва.

1.3.5.2. Мережеві процесори

Мережеві процесори, відповідний англомовний термін Network Processors (NP), були створені для прискореного виконання спеціалізованих мережевих задач, включаючи множинне розпізнавання патернів для МСВВ. Типовий NP крім компонентів звичайної мережевої карти містить центральний процесор зазвичай, VLIW- або RISC-архітектури, багатопортову пам'ять, контролер швидкого доступу до неї та додаткові логічні ресурси для виконання типових мережевих операцій. На такому пристрої зручно організувати обчислення за алгоритмом множинного розпізнавання Ахо–Корасік чи іншого, використовуючи додаткову логіку для створення керуючого пристрою цифрового автомату, пам'ять – для зберігання таблиць переходів, а центральний процесор – для створення та завантаження цих таблиць, а також – для виконання загального управління [75], [76], [77], [78], [79], [80].

Існують також NP, мікросхеми асоціативної пам'яті з трьома станами – англ. Ternary Content Addressable Memory (TCAM) [81]. Трійкову асоціативну пам'ять також можна використовувати для вирішення ЗМРП, але зовсім інакше, ніж за допомогою цифрових автоматів, досягаючи при цьому значних швидкісних результатів [82]. На відміну від звичайної бінарної трійкова асоціативна пам'ять надає обмежені можливості гнучкого розпізнавання [83] (див. підр. 2.3.1 "Гнучке розпізнавання").

Але, незважаючи на те, що NP розроблялися спеціально для вирішення мережевих задач, жорстка архітектура та обмежені обчислювальні ресурси призвели до зниження інтересу до УО цього типу, і останнім часом вони, фактично, не використовуються.

1.3.5.3. TCAM

Трійкова асоціативна пам'ять, строго кажучи, не є окремою обчислювальною платформою [81], [84], [85]. Але швидкісні здібності TCAM (затримка відповіді – одиниці наносекунд) викликають певний інтерес та виокремлюють її від інших апаратних засобів, що орієнтовані на алгоритмічне вирішення МЗРП. Розроблена низка методів використання TCAM саме для множинного розпізнавання, але без урахування її практичної імплементації в складі прискорювачів. Також відомо кілька порівняльних досліджень, в яких TCAM розглядається як обчислювальна технологія, тому цей різновид апаратури був виділений в даному дослідженні в окремий підрозділ.

До недоліків підходу на базі трійкової асоціативної пам'яті можна віднести жорстку архітектуру та обмеження на об’єм словнику патернів.

1.3.5.4. Багатоядерні процесори

Типовим представником елементної бази сучасних багатоядерних пристроїв є виріб Intel Xeon Phi [86]. Головна перевага технології Multi-Core полягає у простоті програмування. Подібність до звичайних процесорів обумовлює використання при реалізації множинного розпізнавання алгоритмічних підходів, що добре відомими та відпрацьованими. Значні обчислювальні можливості даної технології спрощують виконання складних завдань ЗМРП, таких як гнучке розпізнавання з використанням регулярних виразів [87], [88] (див. підр. 2.3.1 "Гнучке розпізнавання").

До недоліків напряму можна віднести відомі вади мікропроцесорної техніки – жорстку обчислювальну структуру та підвищене енергоспоживання, а також складнощі в організації обміну даними с

центральним (керуючим) комп’ютером і між ядрами процесору, що ускладнює розпаралелювання ЗМРП.

1.3.5.5. Графічні прискорювачі

Технологія використання графічних акселераторів не за призначенням, а для виконання ресурсоємних обчислень отримала назву GPGPU (від англ. General-Purpose Graphics Processing Units).

Задача відображення графічної інформації на екрані персонального комп'ютера (ПК) з самого початку виявилася складнішою, чим власне обробка даних у центральному процесорі – Central Processing Unit (CPU). І ця задача з розвитком ПК лише ускладнювалася. Ринок збуту графічних адаптерів дорівнював ринкові ПК, тому виробники вкладали значні ресурси в розвиток мікросхем графічних процесорів – англ. Graphics Processing Unit (GPU). Найбільш потужні за функціональністю та продуктивністю вироби містили все більш велику кількість компонентів, здатних самостійно обробляти інформацію. З метою уніфікації технічних рішень та спрощення процесів керування ними були задіяні відомі прийоми низькорівневого програмування, в наслідок чого згодом непередбачено з'явилася можливість не тільки пересилати дані у GPU для обробки, але також отримувати назад результати обчислень.

Сучасні GPU складаються з багатьох незалежних процесорних елементів, за рахунок чого здатні одночасно обробляти велику кількість потоків даних [89]. Процесорні елементи можуть бути запрограмовані незалежно або подібно до векторної архітектури. Тобто GPGPU можуть функціонувати і як архітектура SIMD (single instruction multiple data), і як структура SPMD (signal program multiple data), що є різновидом класу MIMD (multiple instruction, multiple data). Подібно до багатоядерних процесорів на GPGPU реалізують переважно алгоритмічних методи вирішення ЗМРП.

До недоліків технології GPGPU можна віднести жорстку обчислювальну структуру (оптимізовану до того ж для вирішення графічних задач) та надмірне енергоспоживання.

В якості переваг цієї платформи слід зарахувати високу загальну продуктивність та ретельну відпрацьованість взаємодії з центральним комп'ютером, а також розвиненість інструментальних засобів для розробника [90].

Отже, графічні процесори можуть бути використані для вирішення ЗМРП.

1.3.5.6. Реконфігуровні уніфіковані обчислювачі

Найбільш гнучким та універсальним інструментом створення цифрових схем, що дозволяє розробляти обчислювальні засоби довільного призначення та майже довільного рівня складності є програмована логіка [91]. Здатність сучасних ПЛІС (які вже місять мільйоні еквівалентних логічних елементів) синтезувати в собі пристрої обробки інформації за будь-якими принципами, підходами та обчислювальними архітектурами надає розробнику широкий простір для вибору та реалізації найбільш придатному напряму щодо створення засобів розпізнавання. Саме можливість втілення у апаратурі найновітніших методів та алгоритмів обробки інформації слід вважати суттєвою перевагою програмованої логіки.

Оскільки обчислювальна структура, що створена в ПЛІС, фактично є спеціалізованим пристроєм, вона не споживає зайвої енергії на підтримку універсальних компонентів, що робить прискорювачі класу РУО більш економічною опцією у порівнянні з конкуруючими напрямами.

Використання РУО в якості приєднаних прискорювачів дозволяє будувати на базі універсальних комп'ютерів високопродуктивні обчислювальні системи [92].

До переваг реконфігуровних обчислювачів слід віднести також наявність великої кількості виробів, що можуть бути використані в якості

РУО. Електронний ресурс [93], що налічує більше тисячі найменувань, дає можливість оцінити численність і різноманітність реконфігуровних обчислювачів. Серед найбільш відомих виробників РУО можна назвати такі фірми, як Nallatech, Xilinx, Alpha Data, National Instruments, DiNI Group та багато інших [94].

Але найголовнішою перевагою напряму, заснованому на використанні ПЛІС та РУО, слід вважати здатність швидко – за кілька секунд – змінювати функціональність засобу захисту.

К недолікам обчислювачів на базі програмованої логіці слід віднести відносну витратність та нетривіальність процедур розробки цифрових схем та синтезу файлів конфігурації ПЛІС.

1.3.6. Порівняння апаратних платформ для вирішення ЗМРП

Як було сказано вище, мережеві процесори NP та мікросхеми трійкової асоціативної пам'яті TCAM останнім часом втратили інтерес з боку розробників та не використовуються. Отже, для порівняти залишаються платформи ASIC, GPGPU, Multi-Core та на базі ПЛІС.

Завдання вибору найбільш придатної апаратної платформи для вирішення задач інформаційного захисту не є простим. Строге та обґрунтоване рішення було би можливим у випадку, якщо б існували практичні реалізації для кожної платформи однієї й тій самої задачі в однакової постановці, деталізованій до рівня кількісних значень. На жаль, відомостей про такі розробки в наукових публікаціях дослідників даної теми в літературі не виявилося.

Відомі деякі одиничні випадки, коли порівнюється не більше двох технологій, та, у кращому випадку, порівняння здійснюється також із традиційним варіантом на базі CPU. Існує також низка досліджень, в яких конкуруючі напрями порівнюються ретельно та всебічно, але лише на якісному рівні.

Використовуючи джерела як першого, так і другого типу, а також інші наявні відомості, проведемо з максимально можливою об’єктивністю порівняльний аналіз розглянутих вище апаратних платформ вирішення ЗМРП.

В роботі [85] (TABLE VIII) наведене порівняння технологій, заснованих на використанні ASIC, ПЛІС типу FPGA, GPU та NP, між собою та з CPU загально призначення (табл. 1.1).

Таблиця 1.1

Порівняння ASIC, ПЛІС, GPU, NP та CPU

Параметри		**ASIC**	**FPGA**	**GPU**	**NP**	**CPU**
Фізичні обмеження	Вартість	Найвищ.	Середн.	Низьк.	Менш-середн.	Низьк.
	Енергоефективн.	Найвищ.	Менш-середн.	Висок.	Середн.	Найнизш.
	Ресурсоефектив.	Найвищ.	Найгірш.	Низьк.	Середн.	Низьк.
	Масштабованість	Висок.	Низьк.	Середн.	Низьк.	Висок.
Параметри проектування	Гнучкість	Найгірш.	Середн.	Середн.	Середн.	Найкращ.
	Час проектування	Найвищ.	Середн.	Низьк.	Низьк.	Найнизш.
Продуктивність	Пікова	Найвищ.	Середн.	Середн.	Середн.	Найнизш.
	Реальна	Найвищ.	Середн.	Вищ-середн.	Середн.	Найнизш.

Порівняння дозволяє отримати деяке уявлення щодо переваг та недоліків різних напрямів. Але, на жаль, зробити за його допомогою конкретні висновки щодо найбільшої придатності якось однієї технології важко.

В дослідженні [95] наведене (TABLE V) якісне порівняння технологій Multi-Core / ASIC / ПЛІС (табл. 1.2).

У цьому ж дослідженні наведені дані (TABLE VI) щодо декількох практичних розробок з глибокого аналізу мережевих пакетів, що використовують технології GPGPU, Multi-Core та на базі ПЛІС. Кількісних значень в цьому порівнянні також нема, але якісна оцінка продуктивності свідчить, що в реальному часі обробляти інформацію здатні всі чотири

розробки з програмованою логікою, лише одна з трьох – на базі GPU, та не здатна розробка, що використовує Multi-Core процесор.

Таблиця 1.2

Порівняння Multi-Core, ASIC та ПЛІС

Платформа	**Multi-Core**	**ASIC**	**ПЛІС (типу FPGA)**
Перевага	Можна суттєво підвищити пропускну здатність, використовуючи велику кількість потоків для паралельно обробки	Забезпечує вражаюче високу пропускну здатність на потоці	Забезпечує бажано високу продуктивність, гнучкість програмного забезпечення та реконфігуровність
Недолік	Додаткову складність викликає планування, буферизація, упорядкування та балансування навантаження	Застосовування обмежене високою вартістю впровадження та низькою перепрограмованістю	Потрібно багато часу на пересинтезування та перепрограмуваня пристрою FPGA

Кількісне порівняння деяких конкретних розробок на платформах Multi-Core, GPGPU, ASIC та ПЛІС можна знайти в [55]. Окремі дані з цієї публікації (TABLE III) наведено в табл. 1.3.

Таблиця 1.3

Кількісне порівняння розробок на Multi-Core, GPGPU, ASIC та ПЛІС

№	Джерело	Платформа	Кількість патернів	Довжина патернів	Тактова частота, МГц	Пропускна здатність, Гб/сек
1.	[87]	Multi-Core (Cell / B.E.)	8400	< 10	3200	2,50
2.	[96]	GPGPU (GeForce 8600GT)	4000	< 25	1200	2,30
3.	[97]	ASIC (0,18 мкм)	1785	довільна	763	6,10
4.	[98]	ПЛІС	1316	довільна	220 (200)	1,76
5.	[56]	ПЛІС	~8000	довільна	138 (125)	2,20
6.	[99]	ПЛІС	6944	< 64	285	4,56
7.	[55]	ПЛІС	9033	довільна	178	11,40

Додаткові дані щодо застосування кожної технології в якості апаратної платформи для високопродуктивної оброки даних в задачах інформаційного

захисту – як загального плану, так і стосовно конкретних розробок – можна знайти в наступних публікаціях: [100], [71], [97], [42], [101], [102], [103], [104], [105].

В одному з недавніх досліджень повідомляється, що реалізація на ПЛІС конкретної DPI-задачі в сфері мережевої безпеки за результатами низки експериментів перевершила традиційний процесор у 24-89 разів, а 12-ядерний процесор – у 4,7-7,4 разів. Порівняння реконфігуровного рішення з реалізацією тій самій задачі на платформі GPGPU показало перевагу ПЛІС: за швидкістю – у 3,14 разів, за затримкою розповсюдження – у 1020 разів, за енергоспоживанням – у 106 разів [106].

Слід звернути увагу ще на такий факт. В табл. 1.3 з кількісними даними щодо порівняння крізних платформ тактові частоти в розробках на Multi-Core, GPGPU та ASIC значно перевищують частоти рішень на ПЛІС. Це свідчить про високий потенціал розвитку реконфігуровної платформи, для якої, на відміну від інших напрямів, технологічна межа ще не досягнута, тобто досі діє так званий закон Мура.

Отже, підбиваючи підсумок, обґрунтовано приходимо до висновку, що РУО в якості апаратної платформи якнайкраще відповідають складній та динамічній природі задач інформаційного захисту, випереджають конкуруючи технології за економічністю та гнучкістю. Тому з усіх розглянутих напрямів реконфігуровна технологія є найбільш придатною платформою для вирішення МЗРП в якості АСЗТЗІ.

Тому в якості об'єкта даного дослідження обираємо реконфігуровні апаратні сигнатурні засоби технічного захисту інформації – РАСЗТЗІ.

Більш детально РУО розглянуті в підр. 1.5 "Реконфігуровні уніфіковані обчислювачі".

1.4. Аналіз можливостей програмованої логіки

Програмовані логічні інтегральні схеми давно і успішно використовуються в якості елементної бази при створенні цифрових виробів

для вирішення традиційних завдань електроніки: автоматичне керування і контроль, вбудовані системи, цифрова обробка сигналів тощо. Але з часом їх стали застосовувати також для виконання ресурсоємних обчислень, для яких раніше використовувалися суперкомп'ютери. Передумовою такого "нетрадиційного" використання програмованої логіки з'явилися її багаті функціональні можливості та безпрецедентна гнучкість. Теоретично, в пристрій на базі ПЛІС можна в процесі використання динамічно завантажувати (тобто майже миттєво "виготовляти") довільні обчислювальні структури високої складності.

Перші ПЛІС – програмовані логічні матриці (ПЛМ) – з'явилися приблизно 40 років тому з метою знизити витрати на розробку електронних пристроїв, що виготовляються відносно невеликим тиражем [107]. Великі інтегральні мікросхеми (ВІС), які почали в той час широко використовуватися, коштували дорого і не мали достатньої гнучкості.

Програмовані ІС швидко розвивалися: з'явилися вироби з ультрафіолетовим стиранням, електрично перепрограмовані схеми. В результаті вони ставали все більш зручними в застосуванні. Росла також їх логічна ємність [108].

З'явилося багато технологічних компаній, що займалися створенням мікросхем ПЛІС: Actel, AMD, Atmel, Cypress, DynaChip, Gatefield, Lattice, Lucent, ICT, Intel, Motorola, QuickLogic, Philips, Texas Instruments, Waferscale Integration та ін. В якості визнаних лідерів в цій області утвердилися фірми Xilinx [109] і Altera, яка пізніше увійшла до складу корпорації Intel [110].

Крім власне кристалів програмованої логіки виробники ПЛІС створювали також інструментальні засоби для роботи з ними [111], [112].

В сучасних ПЛІС, що мають походження з перших логічних матриць, для зберігання конфігурації використовується незалежна електрично перепрограмована флеш-пам'ять. Такі пристрої найчастіше використовуються для створення складних комбінаційних схем. У зарубіжній літературі цей клас виробів відомий як CPLD – complex programmable logic

device. Але для синтезу складних обчислювальних схем набагато частіше використовується інший клас програмованих мікросхем – ПЛІС зі статичною пам'яттю (СППЛІС), які своїм походженням зобов'язані технології напівзамовних мікросхем на базових матричних кристалах. В англомовній літературі ці вироби отримали назву Field-Programmable Gate Array (FPGA) [113].

У структурному плані ПЛІС також інтенсивно розвивалися [114]. Сьогодні вони містять в своєму складі не тільки логічні елементи, але й тригери, мультиплексори, комутаційні схеми, тактові шини, вбудовані блоки пам'яті і буферні каскади, а також багато інших компонентів різного призначення [115]. На сучасних ПЛІС можна синтезувати цифрові пристрої дуже високої складності. Вже виробляються вироби, логічний ресурс яких має розмірність мільярдів еквівалентних логічних елементів при тактовій частоті до 850 МГц [116]. При цьому ціна програмованої логіки в перерахунку на один елемент постійно знижується. Останні на сьогоднішній день сімейства ПЛІС типу FPGA виготовляються по 20- і 14-нанометровим проектним нормам (наприклад, серія Xilinx UltraScale). [74], [110].

Але в даному дослідженні нас будуть цікавити насамперед ресурси логіки та пам'яті. Тому коротко розглянемо структуру та компоненти сучасних ПЛІС з цієї точки зору.

1.4.1. Загальна структура ПЛІС

В самому загальному уявленні структуру програмованої НВІС типу FPGA можна подати у вигляді, зображеному на рис. 1.1.

Архітектурно типова ПЛІС найбільш поширеної структури складається з масиву однакових конфігуровних логічних блоків (КЛБ), відповідний англомовний термін – Configurable Logic Blocks (CLB), рівномірно розподілених по площині кристала. По периметру кристала розміщені блоки введення-виведення (БВВ), англ. – Input/Output Blocks (IOB) [117]. Комутаційна матриця програмованих міжз'єднань забезпечує усі зв’язки

всередині кристалу, а БВВ – інтерфейс між внутрішніми компонентами НВІС і зовнішнім світом через її контакти [118].

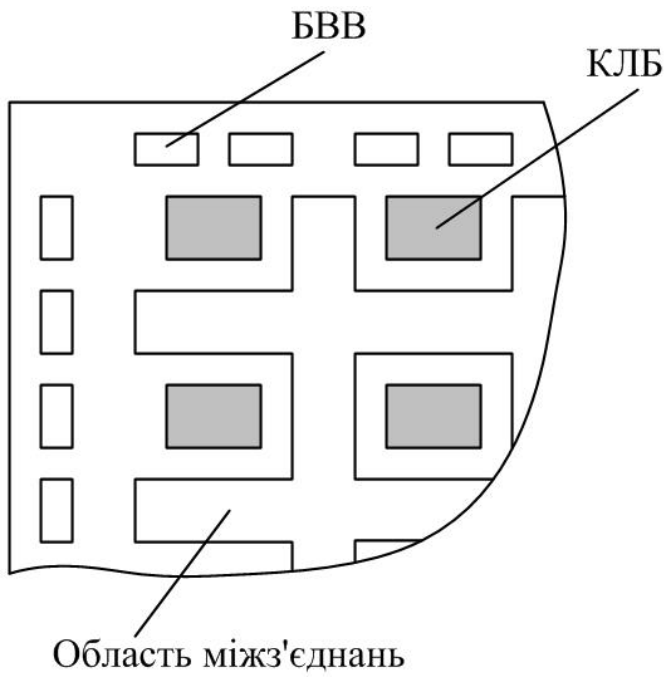


Рис. 1.1. Узагальнена структура ПЛІС типу FPGA

1.4.2. Конфігуровні логічні блоки

Конфігураційні блоки містять у собі декілька (зазвичай – дві) секції (slice).

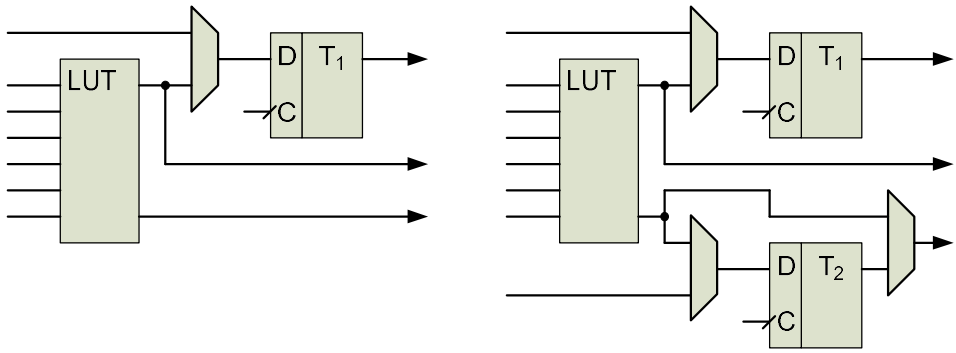
1.4.3. Секції

Секції містять у собі декілька (дві або чотири) логічних комірок (logic cell). В технічної документації розробників ПЛІС дане поняття має різне тлумачення. В даної роботі під цім терміном розуміється одна логічна таблиця, відповідний англомовний термін – Lookup Table (LUT), та низка компонентів, що безпосередньо пов'язана з нею.

1.4.4. Логічні комірки

Логічна комірка (ЛК) може мати різну структуру, але у більшості пристроїв ПЛІС крім логічної таблиці вона містить деяку кількість (один або

два) елементів пам'яті (тригерів) та допоміжні комутаційні компоненти (рис. 1.2) [116].



а б

Рис. 1.2. Різновиди логічних комірок

Стосовно даного дослідження важливо, що як LUT, так і тригері повністю розв'язані по входах та виходах, тобто можуть бути використані як окремі ресурсі. В подальшому апаратні ресурсі у вигляді LUT будемо називати *логічними ресурсами* ПЛІС, а тригери – *ресурсами розподіленої пам'яті*.

1.4.5. Логічні таблиці

Слід зауважити, що англомовний термін LUT – lookup table – в різних літературних джерелах перекладається геть по-різному. Зокрема, в перелічених нижче монографіях для позначення цього компоненту ПЛІС використовуються такі словосполучення: "функциональный блок табличного типа" [119], "таблица преобразований" [111], "логический блок табличного типа" [120], "логическая таблица" [121]та [122], "просмотровая таблица" [91], "таблица соответствия" [123], "таблица преобразования" [124], "таблица истинности (логический генератор)" [117]. Кожний з цих термінів обґрунтовано має право на застосовування, тому з міркувань простоти обираємо для подальшого використання термін *логічна таблиця* (ЛТ).

ЛТ є бататофункціональним пристроєм, який зазвичай має 4, 6 або 8 входів. В залежності від потреби розробника вона може бути сконфігурована

по-різному. Найчастіше – як комбінаційна логічна схема, що реалізує необхідну бінарну функцію від потрібної (в межах можливої) кількості входів. В деяких виробах на однієї ЛТ можна реалізувати дві бінарні функції від одного й того ж набору вхідних змінних, але кількісно – на одиницю менше. Тобто на одній такій 6-входовій ЛТ можна створити дві 5-аргументні комбінаційні схеми зі спільними входами та двома окремими виходами. Слід зауважити, що допоміжні компоненти логічних комірок, а саме мультиплексори дозволяють збільшити кількість змінних при синтезі бінарних функцій на ЛТ.

Також логічна таблиця в більшості виробів ПЛІС може бути сконфігурована як постійний запам'ятовуючий пристрій ємністю $2x$ комірок, де x – кількість входів ЛТ, або у вигляді $2x$-розрядного (чи $2x$-1-розрядного) регістра зсуву [116]. З використанням такого регістра зручно створювати цифрові лінії затримки [125].

1.4.6. Тригери логічних комірок

Елементи пам'яті, що входять до складу логічних комірок, є D-тригерами. В залежності от потреб розробника вони можуть бути сконфігуровані або як асинхронні тригери (flip-flop), або як синхронні (latch) [116].

1.4.7. Блокова пам'ять

Крім ЛТ і тригерів ПЛІС містять ще такий вид ресурсів, важливий при побудові РАСЗТЗІ, як блокова пам'ять (BRAM). Цей різновид ресурсів реалізований у вигляді блоків статичної пам'яті об'ємом по кілька десятків бітів, які розподілені по площині кристалу ПЛІС.

Блоки BRAM можуть бути сконфігуровані у вигляді багатопортового (зазвичай – двопортового) оперативного запам’ятовуючого пристрою (ОЗП) з довільним доступом, або двох таких пристроїв половинного об’єму. Надійність зберігання даних забезпечується вбудованою системою виявлення

та корекції помилок – Error Correction Checking (ECC) або зовнішнім контролем парності [126].

1.4.8. Інші компоненти та технічні особливості ПЛІС

Поданий вище огляд структури та компонентів ПЛІС є спрощеним та схематичним. В ньому не враховані особливості ПЛІС від різних виробників. Крім того, в однієї й тій самій ПЛІС склад та можливості ЛК всередині однієї секції можуть відрізнятися. Але, як буде з’ясовано в подальшому дослідженні, таке спрощення уможливлює створення методів прискореного обчислення кількісних показників ефективності компонентів РАСЗТЗІ. З іншого боку, згадані технічні подробиці не складно врахувати під час практичного застосування запропонованих в даному дослідженні методів.

1.5. Реконфігуровні уніфіковані обчислювачі

У підр. 1.3.6 "Порівняння апаратних платформ для вирішення ЗМРП" було зроблено висновок, що РУО (вперше згадані у підр. 1.3.4 "Уніфіковані обчислювачі"), в якості апаратної платформи якнайкраще відповідають складній та динамічній природні задач інформаційного захисту, випереджають конкуруючи технології за економічністю та гнучкістю.

Тому розглянемо більш детально цей клас обчислювальних пристроїв.

1.5.1. Структура та склад РУО

Структура та склад реконфігуровного уніфікованого обчислювача були розглянуті у низці публікацій автора в попередні роки; вперше – у [127], найдетальніше – у [74].

Узагальнена структура такого пристрою містить як обов'язкові, так і додаткові складові частини (рис. 1.3). Його головним компонентом є одна НВІС СППЛІС типу FPGA. Використання декількох мікросхем програмованої логіки знижує гнучкість та універсальність реконфігуровного обчислювача завдяки фіксованим ("жорстким") з'єднанням між кристалами ПЛІС.

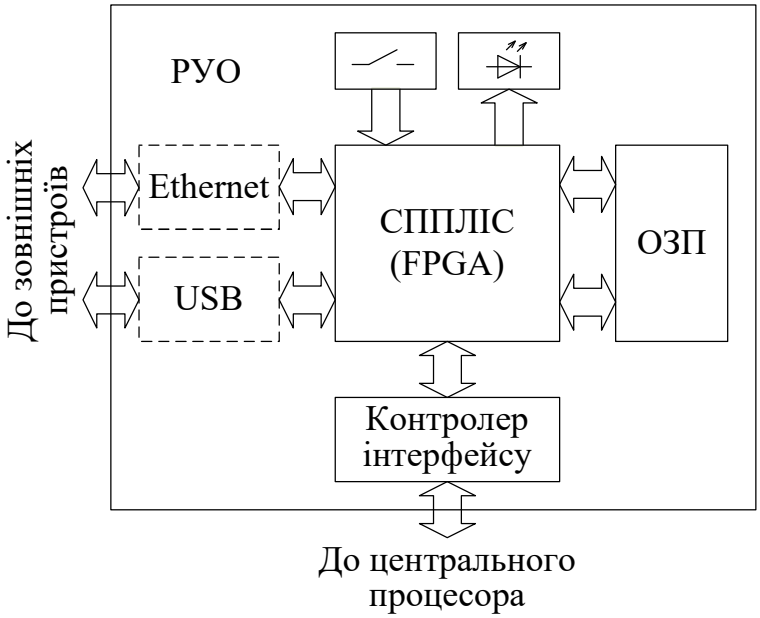


Рис. 1.3. Структура типового РУО

Іншим необхідним компонентом РУО є бортовий (on-board) ОЗП для зберігання проміжних результатів обчислень, в якості якої зазвичай використовуються стандартні вироби динамічної DDR-пам'яті великої ємності.

Обов'язковим компонентом РУО є також окремий апаратний контролер інтерфейсу. Крім обміну даними з центральним процесором (ЦП) хост-системи, на цей контролер покладається також функція програмування ПЛІС, тобто завантаження в неї файлу конфігурації, що суттєво покращує експлуатаційні властивості РУО [123], [128].

До важливих додаткових компонентів РУО слід віднести термінальне обладнання комунікаційних каналів для безпосереднього швидкісного обміну з зовнішнім середовищем, оминаючи ЦП хост-системи, які потрібні в багатьох застосуваннях РУО. В якості таких каналів використовують мережеві з'єднання, або стандартні периферійні чи комунікаційні інтерфейси.

Конструктивно РУО виготовляють або у вигляді стандартної плати розширення із системною шиною, або у формфакторі іншого типового

компонента універсального комп’ютера в залежності від використовуваного інтерфейсу обміну з ЦП. [74].

1.5.2. Існуючі реалізації

На сьогоднішній день у світі виробляється значна кількість пристроїв на базі ПЛІС, які можуть використовуватися в якості реконфігуровних прискорювачів, в тому числі – для вирішення задач інформаційної безпеки. В роботі [129] розглянуто декілька класів подібних приладів, до яких можна віднести як високопродуктивні РУО універсального призначення, так і менш функціональні, але більш доступні за ціною вироби, такі як стартові набори (Starter Kit), тренувальні плати (Trainer Board), оціночні набори (Evaluation Kit), проектні плати (Development Board) та плати-прототипи (Prototype Plate).

Електронний ресурс [93], що налічує більш ніж тисячу найменувань, дає можливість оцінити численність і різноманітність реконфігуровних обчислювачів. Серед найбільш відомих виробників РУО можна назвати такі фірми, як Nallatech [130], Xilinx [109], Alpha Data, National Instruments, DiNI Group та багато інших.

На рис. 1.4-1.6 наведено приклади зовнішнього вигляду деяких моделей РУО: "Atlys" від фірми Digilent, "NetFPGA-1G-CML" від цієї ж фірми та "XUP-VV8" від фірми Nallatech (яка була викуплена фірмою Bitware, що згодом була в свою чергу викуплена компанією Molex) відповідно.

У табл. 1.4 наведені відомості стосовно низки прикладів реконфігуровних прискорювачів, яка охоплює широкий діапазон пристроїв за технічними характеристиками та функціональними можливостями [94]. Як можна бачити, основні параметри пристроїв РУО відрізняються на декілька порядків. Це обумовлено тим фактом, що реконфігуровним обчислювачам притаманний більш тривалий, ніж у традиційної комп’ютерної техніки, термін морального старіння. Як наслідок в експлуатації одночасно

знаходяться дуже різні за можливостями прискорювачі, параметри яких відрізняються на декілька порядків.

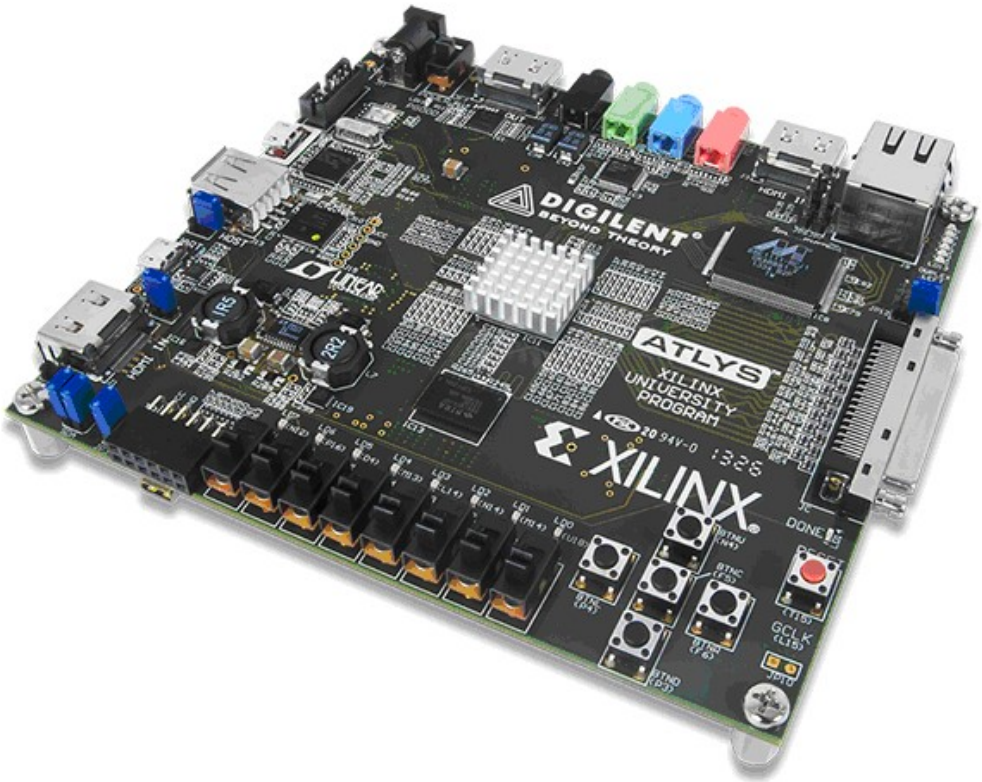


Рис. 1.4. РУО типу Atlys від фірми Digilent

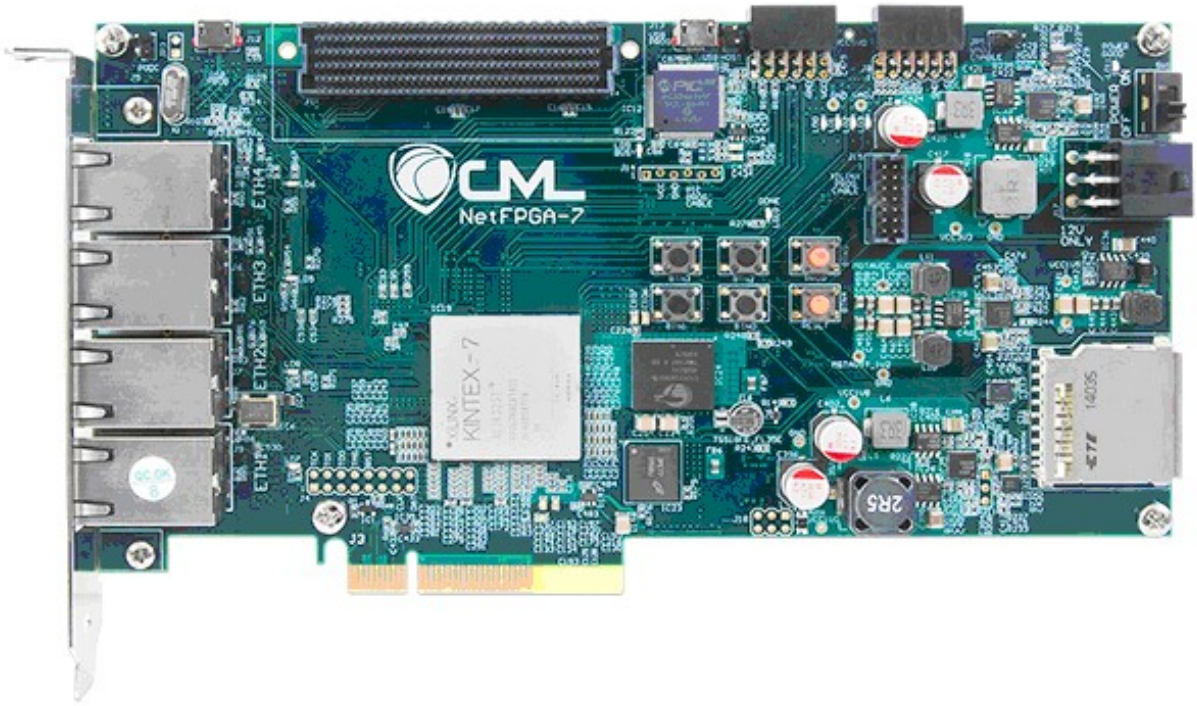


Рис. 1.5. РУО типу NetFPGA-1G-CML від фірми Digilent

Рис. 1.6. РУО типу XUP-VV8 від фірми Nallatech
(Bitware, a Molex company)

1.5.3. Складності при використанні РУО та шляхи їх усунення

Одним із важливих чинників, що перешкоджає широкому розповсюдженню РУО, є висока трудомісткість процесу розробки обчислювальних структур для ПЛІС, який вимагає від розробника високого рівня кваліфікації, володіння засобами автоматизованого проектування, знання специфіки функціонування цифрових пристроїв та особливостей застосування реконфігуровної елементної бази.

Останнім часом на дану проблему спрямовані зусилля багатьох дослідників і розробників. Крім фірмових пакетів САПР все частіше використовуються візуальні засоби розробки, а також модифіковані під проектування апаратури широко поширені високорівневі мови програмування, наприклад, Impulse C, SystemC, Handel-C.

Спростити проблему дозволяє також звуження кола задач, що вирішуються. Наприклад, при створенні цифрових схем для РАСЗТЗІ,

регулярна структура схем реалізації деяких підходів до побудови модулів розпізнавання дозволяє використовувати засоби автоматизованого синтезу.

Таблиця 1.4

Основні характеристики деяких РУО

Виробник	**Nallatech**	**Nallatech**	**Xilinx**	**Digilent**	**Digilent**
Модель	XUP-VV8	385A	VC709 Kit	NetFPGA-1G-CML	Atlys
Клас	РУО	РУО	Evaluation Kit	Development Board	Trainer Board
Тип ПЛІС	Xilinx Virtex UltraScale+ VU13P	Intel (Altera) Arria 10 GX 1150	Xilinx Virtex-7 VX690T	Xilinx Kintex-7 XC7K325T	Xilinx Spartan-6 LX45
Логічних комірок	3 780 000	1 150 000	693 120	326 080	43 661
Розмір блоку BRAM, Кбітів	36	20	36	36	18
Об’єм пам'яті BRAM, Мбітів	94,5	54,3	52,9	16,0	2,1
Тип бортового ОЗП	SDRAM DDR4	SDRAM DDR3	SDRAM DDR3	SDRAM DDR3	SDRAM DDR2
Об’єм бортового ОЗП, Гб	512	32	8	0,512	0,128
Мережеві порти, кількість	4x QSFP-DD (2x 100 Gb-Ethernet)	2x QSFP28 (100 Gb-Ethernet)	4x SFP+ (10 Gb-Ethernet)	4x RJ-45 PHY (1 Gb-Ethernet)	1x RJ-45 PHY (1 Gb-Ethernet)
Інтерфейс зв'язку з ЦП	PCI-Express x16 Gen.3	PCI-Express x8 Gen.3	PCI-Express x8 Gen.3	PCI-Express x16 Gen.2	USB 2.0
Форм-фактор	Dual-slot PCI-E card стандартної висоти 3/4 довжини	Single-slot PCI-E card 1/2 висоти 1/2 довжини	Single-slot PCI-E card стандартної висоти та довжини	Single-slot PCI-E card стандартної висоти 3/4 довжини	Окрема плата 120 x 133 мм
Охолодження	Рідинне	Активне	Пасивне	Відсутнє	Пасивне
Вартість, $	8995	5995	4995	1499	490

Дослідження щодо перспектив розвитку РУО в майбутньому наведено в [131].

1.6. Висновки до розділу 1

1. Проведено аналіз проблем технічного захисту інформації, які вирішуються сигнатурними засобами.

2. Сформульовано задачу множинного розпізнавання рядків.
3. Розглянуті питання апаратного прискорення ресурсоємних задач приєднаними уніфікованими обчислювачами.
4. Проведено порівняльний аналіз апаратних платформ для вирішення задачі множинного розпізнавання патернів. З'ясовано, що складній та динамічній природі задач інформаційного захисту якнайкраще відповідає реконфігуровна апаратна база, яка випереджає конкуруючи технології за гнучкістю та економічністю. В результаті порівняння предметом дослідження обрано РАСЗТЗІ – реконфігуровні апаратні сигнатурні засоби технічного захисту інформації.
5. Проведено дослідження технічних можливості сучасних ПЛІС та пристроїв на їх основі щодо вирішення задач технічного захисту інформації.
6. Розглянуто властивості реконфігуровних уніфікованих обчислювачів, досліджено наявні реалізації, з'ясовано, що внаслідок більш тривалого, ніж у традиційної комп’ютерної техніки, терміну морального старіння РУО, в експлуатації одночасно знаходяться дуже різні за можливостями обчислювачі, параметри яких відрізняються на декілька порядків.

РОЗДІЛ 2. ВИРІШЕННЯ ЗАДАЧ ЗАХИСТУ ІНФОРОМАЦІЇ РЕКОНФІГУРОВНИМИ ЗАСОБАМИ

2.1. Основи побудови та функціонування реконфігуровних РАСЗТЗІ

Для успішного досягнення мети даного дослідження необхідно розглянути основні принципи побудови та функціонування РАСЗТЗІ.

Як було вказано в підр. 1.1.2 "Системи виявлення вторгнень", історично першими реконфігуровними засобами, що широко застосовувалися для інформаційного захисту, були мережеві системи виявлення вторгнень. Тому, не втрачаючи загальності міркувань, розглянемо особливості побудови та функціонування РАСЗТЗІ на прикладі саме таких пристроїв.

2.1.1. Узагальнена структура МСВВ на ПЛІС

В публікації [132] проведено дослідження та узагальнення основних принципів побудови МСВВ на базі програмованої логіки за результатами накопиченого в світі досвіду застосування ПЛІС типу FPGA. Розглянуто основні параметри подібних систем і вимоги до них. На рис. 2.1 наведено структуру апаратної реалізації на базі ПЛІС системи виявлення вторгнень, точніше, її основний функціональний компонент – аналізатор, отриману в результаті аналізу та узагальнення існуючих розробок відомих дослідників в цій галузі. Розглянемо призначення компонентів, що входять до складу схеми [133].

Модуль прийому пакетів здійснює низькорівневе захоплення мережевих пакетів та їх перетворення в більш зручний для внутрішньосхемної обробки тип кодування, наприклад, з формату XAUI (10 Gigabit Attachment Unit Interface) у формат XGMII (10 Gigabit Media Independent Interface).

Аналізатор пакетів відокремлює заголовки пакетів від тіл на відповідних рівнях мережевої моделі OSI в залежності від способу, яким

аналізується мережевий трафік: raw чи stateful (див. підр. 1.1.2.6 "Функціонування сигнатурної МСВВ").

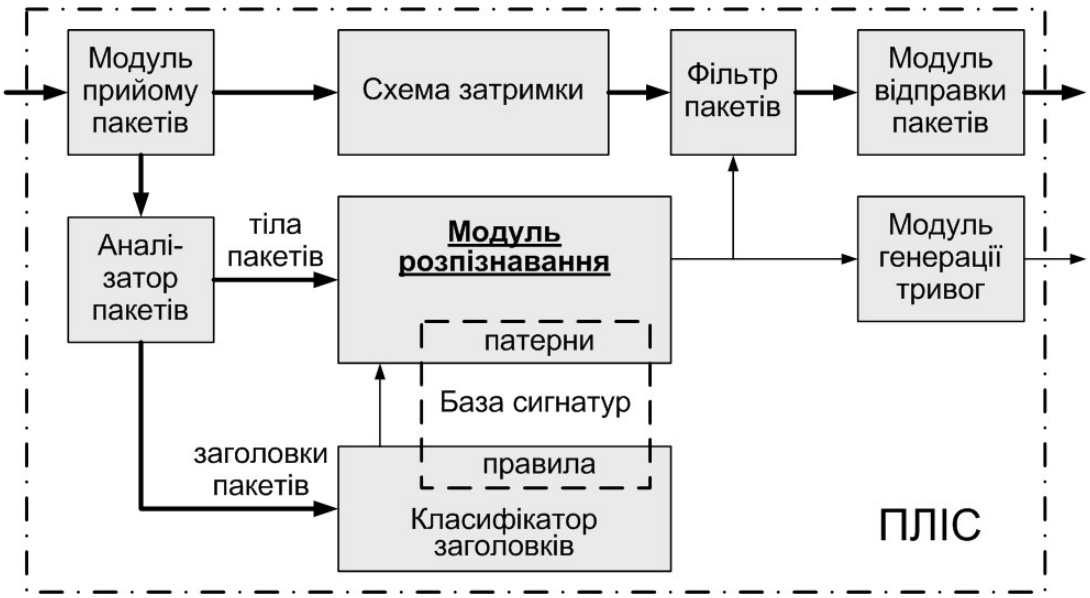


Рис. 2.1. Узагальнена структурна схема аналізатора МСВВ на базі ПЛІС

Класифікатор заголовків витягає інформацію з полів заголовків пакетів щодо сигнатур, які виявляються (див. підр. 1.1.2.9 "Поняття сигнатури та патерну. Склад сигнатури МСВВ"), та передає її модулю розпізнавання, внаслідок чого ідентифікуються шкідливі пакети. Приклади подібного зв'язку між класифікатором заголовків та модулем розпізнавання можна подивитися на відповідних схемах в роботах [134] (Fig. 1) та [135] (Fig. 1).

Модуль генерації тривог формує повідомлення про виявлені вторгнення.

Наступні три компоненти – схема затримки, фільтр пакетів і модуль видачі пакетів – не є необхідними компонентами МСВВ і присутні в структурі тільки у випадку роботи системи в режимі запобігання вторгнень.

Схема затримки синхронізує потік пакетів з роботою модуля розпізнавання.

Фільтр пакетів призначений для припинення шкідливого трафіку шляхом відкидання підозрілих пакетів.

Модуль видачі пакетів реалізує перетворення, зворотне такому, що виконується в модулі прийому пакетів.

Модуль розпізнавання є центральним функціональним блоком МСВВ, який вирішує в реальному масштабі часу обчислювально складну задачу ЗМРП, тобто перевіряє зміст мережевих пакетів на наявність патернів.

База даних сигнатур типового сучасного засобу технічного захисту містить від десятків тисяч до мільйонів патернів. Специфіка апаратних ЗТЗІ така, що ця база не зберігається окремо на якомусь носії інформації, а фактично "вшита" у обчислювальну структуру аналізатора МСВВ. При цьому всі патерни, що входять до складу бази, містяться (явно чи неявно) у модулі розпізнавання, а заголовкова частина сигнатур записів (правил) бази – протоколи, адреси, порти тощо – в класифікаторі заголовків.

2.1.2. Критичні компоненти РАСЗТЗІ

Аналіз численних досліджень та розробок в галузі реконфігуровних засобів технічного захисту інформації свідчить, що модуль розпізнавання (МР) є ключовим компонентом будь-якого РАСЗТЗІ. Його технічні характеристики безпосередньо впливають на властивості засобу ТЗІ в цілому. По різним оцінкам у програмних реалізаціях МСВВ вирішення ЗМРП складає 40-70% загального часу обробки (включаючи файлові операції) або 60-85% обчислювальних ресурсів центрального процесора за кількістю виконаних інструкцій [136], [55].

Тобто модуль розпізнавання є найважливішим компонентом МСВВ зокрема та будь-якого РАСЗТЗІ взагалі, від характеристик якого вирішальною мірою залежать показники ефективності системи в цілому. Інші часові та ресурсні витрати МСВВ розподілені між рештою компонентів, внесок кожного з котрих є забагато меншим.

Назвемо *критичними компонентами* такі складові РАСЗТЗІ, внесок яких в загальну ефективність технічного засобу в цілому є вирішальним.

Наведений вище аналіз узагальненої структури МСВВ як типового РАСЗТЗІ свідчить, що його критичним компонентом в сенсі ефективності є модуль розпізнавання.

Виявлення критичного компоненту дозволяє зосередити зусилля під час проведення дослідження саме на його вдосконаленні та не займатися поліпшенням складових, доробка яких не призведе до підвищення ефективності всієї системи.

Отже в подальшому досліджені в якості уточненого предмету дослідження будемо розуміти модуль розпізнавання реконфігуровного апаратного сигнатурного засобу технічного захисту інформації (МР РАСЗТЗІ).

2.1.3. Поняття процедури оперативного оновлення

Специфічною особливістю застосування реконфігуровних засобів для апаратного прискорення систем виявлення вторгнень та інших сигнатурних засобів захисту інформації є необхідність повторного синтезу деяких апаратних компонентів, яка регулярно виникає під час їх функціонування.

Назвемо таку операцію *процедурою оперативного оновлення* (ПОО).

Причиною потреби в подібній процедурі може бути, по-перше, поява описів нових атак, що не були відомі раніше, по-друге – зміна умов роботи інформаційної системи, що захищається (модифікація локальної мережі, зміна її складу або структури, переустановлення програмного забезпечення тощо). В обох випадках виникає потреба в оновленні бази даних сигнатур. Наприклад, якщо в локальної обчислювальної мережі встановлюється програмний додаток, якого раніше не було, скажімо, веб-сервер, в базу даних сигнатур потрібно додати описи (сигнатури) всіх відомих атак на цій додаток.

У зв'язку з тим, що в процесі оперативного оновлення змінюється незначна частка обладнання, виконання повного циклу створення цифрової схеми в ПЛІС не потрібно. Більшу частину роботи можна виконати завчасно

та уникнути під час проведення ПОО, що спрощує процес повторного синтезу пристрою РАСЗТЗІ. Фактично, весь технологічний ланцюжок синтезу цифрової схеми розпізнавання зводиться при цьому лише до двох етапів. На першому етапі згідно вхідного набору патернів формуються обчислювальні структури оновлених компонентів у вигляді тексту на мові опису апаратури. На другому отримані описи в автоматичному режимі перетворюються в конфігурації для ПЛІС. Більш детально ці етапи розглянуто в розділі 7.

2.1.4. Змінні та постійні складові РАСЗТЗІ

Аналізуючи структурну схему аналізатора МСВВ можна зробити висновок, що під час виконання ПОО не потрібно пересинтезувати всі компоненти аналізатора МСВВ.

Назвемо *змінною складовою* такі компоненти, що потрібно наново синтезувати при виконанні ПОО, та *постійною складовою* такі, що пересинтезувати не потрібно. Оскільки потреба у ПОО виникає при зміні складу бази даних сигнатур, змінна складова містить компоненти, безпосередньо пов'язані з обробкою сигнатур. В результаті аналізу узагальненої структурної схеми аналізатора (рис. 2.1) находимо, що у випадку МСВВ до змінної складової належать модуль розпізнавання та класифікатор заголовків.

Виокремлення змінних складових від постійних дозволяє спростити та прискорити процедуру оперативного оновлення за рахунок повторного синтезу лише змінних складових.

2.2. Показники ефективності РАСЗТЗІ

Щоб оцінювати результативність запропонованих методів, потрібно визначитися з показниками ефективності. Такі показники крім використання для оцінки властивостей розробок сигнатурних реконфігуровних засобів захисту інформації та їх компонентів (як відомих, так і отриманих в

результаті застосування запропонованих в даному дослідженні методів) також нададуть змогу порівнювати різні підходи до побудови РАСЗТЗІ та окремі технічні рішення.

Спираючись на аналіз відомих розробок реконфігуровних засобів захисту інформації розглянемо на загальному (якісному) рівні класифікацію таких показників (рис. 2.2).

Рис. 2.2. Ієрархія показників ефективності РАСЗТЗІ

Всі показники ефективності РАСЗТЗІ можна поділити на наступні головні категорії:

- вартісні;
- швидкісні або показники продуктивності;
- функціональні.

Крім головних існують такі показники, що не потрапляють до їх числа, але також мають важливе значення. Умовно назвемо їх *проміжними*

показниками. До таких показників відносяться масштабованість, яка пов'язує швидкісні характеристики з вартісними, показник використання надлишковості словнику патернів та низка похідних показників, що формуються з кількох інших.

Наведена класифікація не є повною, оскільки дозволяє додавання нових показників, зокрема – функціональних, внаслідок висування спеціальних вимог до РАСЗТЗІ в деяких випадках їх практичного застосування.

Проте, незважаючи на неповноту, показники, що наведені в запропонованій класифікації, дозволяють успішно оцінювати рівень ефективності реалізацій РАСЗТЗІ та їх компонентів, підвищення якого є метою даного дослідження.

Нижче більш детально розглянуті показники ефективності відповідно до запропонованої класифікації.

2.2.1. Вартісні показники

До вартісних показників належать: обсяги *логічних ресурсів* програмованої логіки, які задіяні для створення цифрової схеми, *витрати на пам'ять* (як зовнішню відносно кристалу ПЛІС, так і внутрішню – блочну пам'ять BRAM та розподілену у вигляді тригерів логічних комірок), а також *інші витрати*, що можуть бути оцінені в вартісних одиницях, аж до загальної вартості володіння системою, яка включає витрати на розробку, виготовлення, програмування та експлуатацію системи. Зауважимо, що останній показник дозволяє оцінувати розробки РАСЗТЗІ в цілому.

2.2.2. Показники продуктивності (швидкісні)

До показників продуктивності (або швидкісних показників) відносять, з одного боку – *об’єм словнику патернів* (тобто кількість різних патернів, що розпізнає засіб), з іншого – *швидкодію* засобу, яка характеризується або часом обробки даних пристроєм, або пропускною здатністю. До даної групи показників можна віднести й *латентність* – затримку розповсюдження

даних від входу до виходу, яка може не співпадати з часом обробки у випадку використання в системі конвеєрного методу обробки інформації.

Важливим швидкісним показником РАСЗТЗІ є також *передбачуваність пропускної здатності* (Deterministic performance) [56]. Деякі алгоритми одиночного або множинного розпізнавання рядків, наприклад, Бойєра–Мура чи Ву–Манбера, що використовують певні евристики з метою підвищити їх пропускну здатність у середньому, в залежності від вхідних даних змінюють швидкість обробки в широких межах [137], [136]. При цьому інші алгоритми, наприклад, Ахо–Корасік [63], мають детерміновану продуктивність. Передбачуваність пропускної здатності спрощує оперування іншими швидкісними характеристиками системи, полегшує інтеграцію РАСЗТЗІ з іншими технічними системами, покращує зручність використання (usability) засобів технічного захисту.

Передбачуваність пропускної здатності як швидкісний показник не треба плутати з функціональним показником спроможності протидіяти атакам на РАСЗТЗІ, з яким вони тісно пов’язані (див. нижче, підр. 2.2.3.1 "Спроможність протидіяти атакам на РАСЗТЗІ").

2.2.3. Функціональні показники

Розглянемо деякі з показників ефективності РАСЗТЗІ, що відносяться до функціональних. Найбільш поширеними серед них є:

- спроможність протидіяти цілеспрямованим атакам на РАСЗТЗІ або інші засоби захисту;
- здатність до оновлення словнику патернів без припинення процесу розпізнавання (динамічна реконфігурація);
- спроможність під керуванням ззовні обирати підмножину патернів, що розпізнаватимуться (селективне розпізнавання);
- здатність МСВВ працювати у режимі запобігання вторгнень.

Зауважимо, що останній показник зі згаданих притаманний лише системам виявлення вторгнень.

Розглянемо докладніше перелічені вище функціональні показники.

2.2.3.1. Спроможність протидіяти атакам на РАСЗТЗІ

Важливим наслідком передбачуваності пропускної здатності (див. підр. 2.2.2 "Показники продуктивності (швидкісні)") є незалежність швидкодії від змісту вхідних даних, що робить відповідні РАСЗТЗІ стійкими до деяких видів атак на засоби захисту інформації. Прикладом може бути так звана атака алгоритмічної складності – навмисне насичення порушником мережевого трафіку хибними пакетами спеціального змісту, розпізнавання яких призводить до лавиноподібного зростання числа обчислювальних операцій, які виконуються згідно алгоритму розпізнавання в засобі захисту, що унеможливлює його коректну роботу [138], [139]. Але, оскільки існують інші різновиди атак на РАСЗТЗІ, передбачуваність пропускної здатності не є достатньою ознакою захищеності від атак на засоби захисту інформації.

2.2.3.2. Здатність до динамічної реконфігурації

Як вказувалося раніше, перевагою обчислювальних засобів на базі ПЛІС є здатність створювати в одному й тому ж обладнанні дуже різні за функціональністю пристрої. Зміна поведінки реконфігуровної апаратури здійснюється шляхом перепрограмування, тобто завантаження в ПЛІС іншого файлу конфігурації. Цій процес в залежності від моделі НВІС програмованої логіки триває від кількох мілісекунд до кількох секунд. Але для засобу захисту, що опрацьовує інтенсивний потік даних, наприклад 10-гігабітний мережевий канал зв'язку, затримка навіть в кілька секунд може виявитися непридатною.

Спроможність оновлення словнику патернів без призупинення функціонування РАСЗТЗІ назвемо *здатністю до динамічної реконфігурації* (Fast dynamic updates) [56].

Технічно таку можливість забезпечують або використанням в системі декількох ПЛІС [140], або застосуванням відповідного алгоритму або підходу до побудови РАСЗТЗІ. Наприклад скінченний автомат Ахо–Корасік,

реалізований на ПЛІС, таблиця переходів якого зберігається в зовнішній пам'яті, дозволяє змінити набір патернів перезаписом змісту ОЗП (або перемиканням на іншу область пам'яті) без реконфігурації програмованої логіки [141].

Зауважимо, що не слід плутати динамічну реконфігурацію з процедурою оперативного оновлення. Остання повинна обов'язково виконуватися всіма сигнатурними ЗТЗІ, не тільки реконфігуровними. З іншого боку, для не реконфігуровних засобів поняття динамічної реконфігурації взагалі не існує.

2.2.3.3. Селективне розпізнавання

Назвемо здатними до *селективного розпізнавання* (Pattern subset) такі РАСЗТЗІ, що дозволяють відповідними керуючими сигналами змінювати підмножину патернів, що розпізнаватимуться [56]. Пристрої з такою властивістю можуть, наприклад, в залежності від типу протоколу перемикатися між підгрупами патернів, притаманних атакам з використанням саме цього протоколу.

2.2.3.4. Режим запобігання вторгнень

В підр. 1.1.2.3 "Системи запобігання вторгнень" вказувалося, що система виявлення вторгнень опціонально може функціонувати в режимі запобігання вторгнень, коли відповідь на ідентифіковані атаки здійснюється в автоматичному режимі. Здатність функціонувати як МСЗВ накладає на МСВВ суворі вимоги як до швидкодії, так і стосовно точності розпізнавання, оскільки у випадку автоматичного запобігання атакам помилки другого роду можуть призвести до важких наслідків аж до призупинення функціонування мережі, що захищається.

Здатність МСВВ працювати у режимі запобігання вторгнень потребує наявності в її складі додаткових апаратних компонентів, а саме: схеми затримки, фільтра пакетів і модуля відправки пакетів (див. рис. 2.1 у підр. 2.1.1 "Узагальнена структура МСВВ на ПЛІС ")

2.2.3.5. Інші функціональні показники

Як вказувалося вище, при створенні деяких РАСЗТЗІ висуваються спеціальні вимоги до їх функціональності, що призводить до появи додаткових якісних функціональних показників, крім розглянутих вище.

2.2.4. Використання надлишковості

Важливий проміжний показник – здатність використання надлишковості словнику патернів з метою покращення швидкісних та ресурсних характеристик. Завдяки ефекту самоподоби (див. підр. 1.2.3 "Ефект самоподоби. Префікси, інфікси, суфікси") інформація, що міститься в базі даних сигнатур є надмірною. Здатність засобу захисту задіяти цю надмірність для підвищення своєї ефективності є його корисною властивістю.

2.2.5. Показники масштабованості

Важливе значення для оцінки ефективності реконфігуровних обчислювальних засобів також мають показники масштабованості, які пов'язують швидкісні характеристики з вартісними. *Масштабованість* (Scalability) – це здатність технічного рішення нарощувати характеристики продуктивності без занадто великих додаткових ресурсних витрат [56].

Розрізняють масштабованість трьох типів:

- за пропускною здатністю (Speed scalability) [142];
- за об’ємом словнику патернів (Rule-set scalability) [143];
- за довжиною патернів (Pattern length scalability) [142].

Добра масштабованість обумовлює здатність РАСЗТЗІ зберігати показники ефективності з часом на задовільному рівні по мірі розвитку інформативних об’єктів, що захищаються.

2.2.6. Похідні показники

Багатий досвід створення та використання РАСТЗІ світовою спільнотою висвітлив потребу універсальних та узагальнюючих показниках,

які поєднували би в собі властивості як вартісних, так і швидкісних показників. Відповідно було запропоновано низку показників, які в кількісному вимірі мають вигляд певної функціональної залежності (математичного виразу) від декількох кількісних показників (як швидкісних, так і вартісних в загальному випадку), таких як пропускна здатність, об'єм задіяних ресурсів ПЛІС, загальна кількість всіх символів, що містяться в патернах бази даних сигнатур тощо.

Назвемо такі показники ефективності *похідними*.

Зауважимо, що похідні показники якнайкраще підходять для кількісної оцінки ефективності технічних рішень та розробок РАСЗТЗИ.

Більш детально похідні показники та засновані на них кількісні технічні параметри розглянуті в підр. 5.1.4 "Похідні параметри".

2.2.7. Питання енергоспоживання

Оскільки в даному дослідженні розглядаються РАСЗТЗІ, що вирішують складну обчислювальну задачу ЗМРП, для їх побудови передбачається використання високопродуктивної стаціонарної обчислювальної техніки. Тому показники споживання електроенергії не розглядаються.

2.2.8. Взаємозв’язок показників. Поняття трейдофу

При оперуванні показниками ефективності та технічними характеристиками цифрових пристроїв важливу роль відіграє явище, яке полягає у здатності обчислювальних схем чи цілих блоків варіювати одними показниками за рахунок інших. Наприклад, якщо продуктивність певного вузла може бути підвищено за рахунок збільшення апаратних витрат на його побудову, і навпаки [56]. В англомовній літературі цю здібність пристрою позначають терміном trade-off. На жаль, у вітчизняних наукових текстах немає повноцінного аналога даному позначенню. Найближчий до нього термін "компроміс" не виконує дану функцію, оскільки співвідноситься зі згаданим поняттям як точка на графіку функціональної залежності співвідноситься з самою залежністю.

В якості зразка властивості трейдофу можна навести, наприклад, здатність 16-розрядного бібліотечного суматора lpm_mult фірмового пакету пакета MAX+plus компанії Altera (власником якої зараз є корпорація Intel) за допомогою зміни параметра MAXIMIZE_SPEED (який має 5 градацій) бути сконфігурованим, з одного боку, так, щоб на його "виготовлення" знадобилося 163 логічних блоків, і при цьому він мав час затримки від входу до виходу 52 нс; а при іншому крайньому значенні даного параметра, ці ж показники будуть мати значення відповідно 121 логічний блок і 80 нс [144].

2.3. Попередній аналіз відомих підходів до побудови МР РАСЗТЗІ

Вивчання світового досвіду побудови реконфігуровних засобів захисту інформації свідчить, що розробники, бажаючи досягнути більш високих показників ефективності, використовують різноманітні підходи щодо побудови критичних компонентів РАСЗТЗІ, які засновані на дуже різних за своєю природою схемотехнічних рішеннях. Випробувавши той чи інший напрям і стикнувшись з певними труднощами, дослідники застосовують численні техніки, методики і прийоми для модифікації базових схем з метою подолання проблем та вдосконалення базових рішень. При цьому, незважаючи на доволі тривалий термін досліджень, в цьому процесі постійного пошуку найкращих рішень не було виявлено чітких закономірностей, не сформовано єдиного математичного апарату чи методики, які дозволяли б систематизувати процес створення питомих пристроїв. Інакше кажучи, наявний досвід в даному напряму досліджень свідчить про наявність об'єктивних чинників або обставин, що ускладнюють формалізацію та застосування наукового методу для узагальнення та побудови універсальних аналітичних моделей процесу створення оптимальних в сенсі ефективності модулів розпізнавання РАСЗТЗІ. Крім того, велика кількість розробок (десятки колективів зі всього світу публікують сотні статей та матеріалів конференцій) ускладнюють аналіз та систематизацію наявних знань.

З огляду на виявлені обставини для досягнення мети даного наукового дослідження було здійснено наступні кроки:

1. Виокремлено декілька найбільш ефективних підходів до побудови МР РАСЗТЗІ (нижче, в даному підрозділі).

2. Шляхом аналізу та систематизації лише обраних підходів і численних технік їх вдосконалення спочатку на якісному рівні виявлено та формалізовано особливості, переваги та недоліки кожного з них, враховуючи специфіку реалізації реконфігуровними засобами (підр. 3.1-3.3).

3. Виокремленні підходи порівняно між собою з метою спроби виявити найкращій по більшості показників (підр. 3.4).

4. Розроблено методи кількісної оцінки різнорідних технічних рішень в єдиному метричному просторі та застосовано їх для виявлення залежностей кількісних показників ефективності від властивостей словнику патернів та характеристик РУО для базових схем обраних підходів та їх модифікацій (розділ 4).

5. Створено методи об'єднання в одному пристрої декількох підходів, максимально задіявши переваги кожного з них для досягнення найвищої ефективності (розділ 5).

На рис. 2.3 схематично подано взаємозв'язки між основними підходами до побудови модулів розпізнавання РАСЗТЗІ, технологіями, на яких вони базуються, та техніками покращення показників ефективності створюваних розробок. (Тут ТСАМ – трійкова асоціативна пам'ять, ДСА – детерміновані скінченні автомати, НСА – недетерміновані скінченні автомати, Геш+ПЗП+ЦК – каскадна схема розпізнавання на базі геш-функції, постійного запам'ятовуючого пристрою та цифрових компараторів).

Насправді складових ієрархії підходів, технологій та технік в даної галузі, а також зв'язків між ними набагато більше. На наведеному рисунку з міркувань наочності їх більшу частину не відображено.

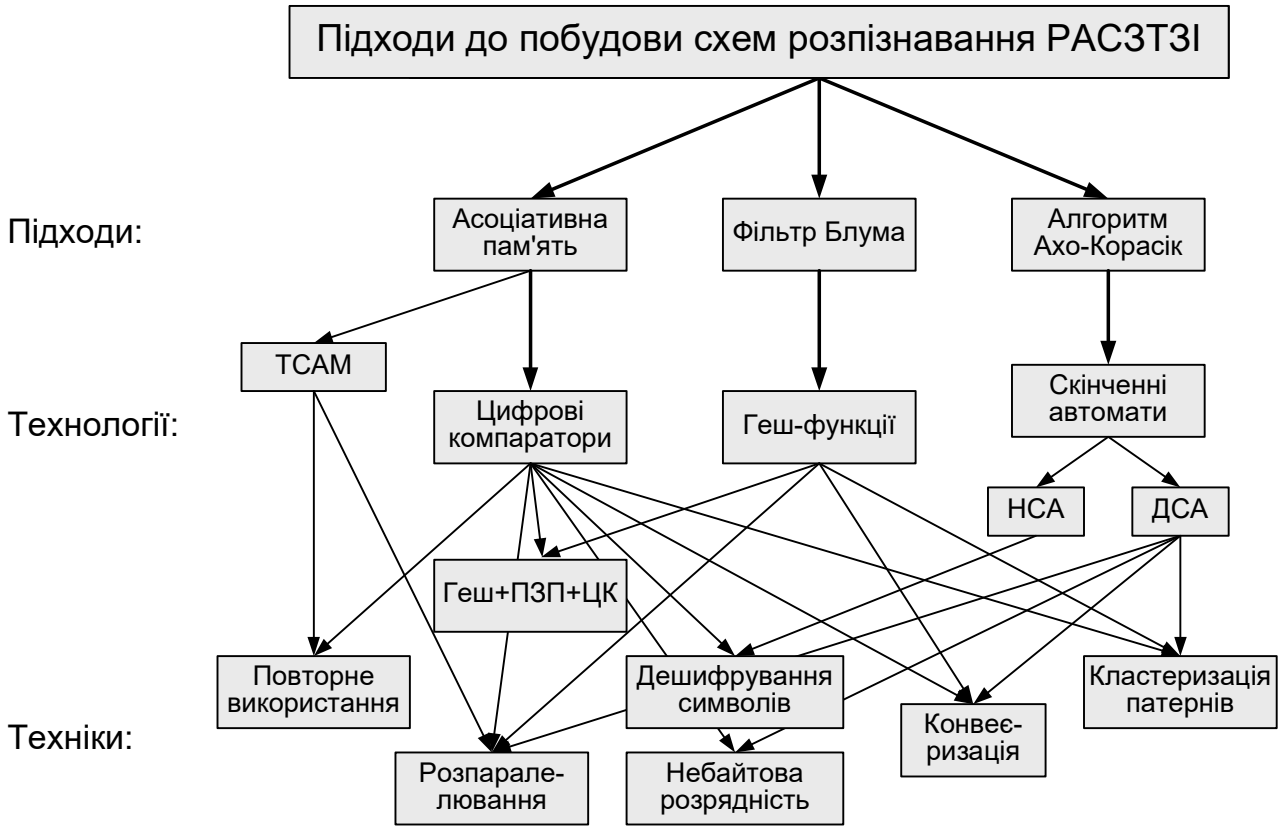


Рис. 2.3. Основні підходів, технологій, та техніки для побудови МР РАСЗТЗІ

2.3.1. Гнучке розпізнавання

На рис. 2.3 не відображено моменти, пов'язані з гнучким розпізнаванням, яке вперше згадувалося в підр. 1.1.2.9 "Поняття сигнатури та патерну. Склад сигнатури МСВВ". Майже кожний сучасний засіб СРАСЗІ містить певну кількість сигнатур, до складу яких входять регулярні вирази – RegEx, які на відміну від патернів, що описують фіксовані послідовності символів, дозволяють розпізнавати широко коло варіацій та комбінацій символів. Потужний апарат RegEx дозволяє за допомогою так званих символів-джокерів (таких як "*", "?", "+", "$", "^","|" та ін.) формувати шаблони, які припускають наявність будь-якого символу (або довільної кількості символів) у певній позиції, варіанти підрядків, відстань між фіксованими патернами у довільну кількість знаків та багато інших

різновидів варіацій, що можуть складатися з фрагментів рядків та окремих символів [145], [83], [146], [147], [148].

Гнучке розпізнавання надає багато можливостей для виявлення найбільш складних та витончених атак на комп'ютерні системи та мережі. Тому гнучкому розпізнаванню розробники РАСЗТЗІ також приділяють увагу вже багато років. Але на відміну від використання патернів, серед підходів до побудови реконфігуровних засобів гнучкого розпізнавання є виражений привалюючий напрям, що випереджає інші підходи – це скінченні автомати [149]. Тому питання реалізації на базі ПЛІС алгоритмів RegEx випадають за межи даного дослідження, мета якого пов'язана з комбінуванням.

2.3.2. Найбільш ефективні підходи до побудови МР РАСЗТЗІ

Як свідчить проведений автором аналіз (розпочатий у 2010 році [150]), найкращі здібності при побудові апаратних схем множинного розпізнавання патернів продемонстрували наступні три підходи (та їх модифікації):

- асоціативна пам'ять на базі цифрових компараторів [151];
- фільтр Блума на базі геш-функцій [152];
- алгоритм Ахо–Корасік на базі скінченних автоматів [141].

В наступних підрозділах проведено формалізацію на якісному рівні кожного з даних напрямів внаслідок ретельного дослідження, в процесі якого головна увага була спрямована на:

- особливості (переваги та недоліки) кожного підходу в сенсі показників ефективності, сформульованих у підр. 2.2 "Показники ефективності РАСЗТЗІ";
- специфіку реалізації підходу на ПЛІС;
- складнощі та проблеми, що виникають під час створення реконфігуровних засобів розпізнавання на базі цих підходів, і шляхи їх подолання.

2.4. Висновки до розділу 2

1. Досліджено принципи побудови та функціонування РАСЗТЗІ на прикладі реконфігуровних МСВВ.
2. Введено поняття критичних компонентів РАСЗТЗІ. З'ясовано, що критичним компонентом РАСЗТЗІ в сенсі ефективності є модуль розпізнавання. Уточнено предмет дослідження як модуль розпізнавання РАСЗТЗІ – МР РАСЗТЗІ.
3. Введено поняття процедури оперативного оновлення, яка повинна регулярно виконуватися на будь-якому сигнатурному засобі технічного захисту інформації внаслідок появи нових сигнатур та зміни умов роботи інформаційної системи, що захищається.
4. Введене поняття змінних та постійних складових РАСЗТЗІ, які відповідно потрібно чи не потрібно пересинтезувати під час виконання процедури оперативного оновлення.
5. Сформульовано та розглянуто показники ефективності РАСЗТЗІ.
6. Проведено попередній огляд відомих підходів до побудови МР РАСЗТЗІ. Виокремлено три найбільш ефективних підходи в сенсі сформульованих показників ефективності.

РОЗДІЛ 3. ФОРМАЛІЗАЦІЯ ВЛАСТИВОСТЕЙ НАЙБІЛЬШ ЕФЕКТИВНИХ ПІДХОДІВ ДО ПОБУДОВИ СХЕМ РОЗПІЗНАВАННЯ

3.1. Використання асоціативної пам'яті на базі цифрових компараторів

Асоціативна пам'ять (АП), відповідний англомовний термін – Content Addressable Memory (CAM), є класом пристроїв, що від самого початку створювалися для швидкого розпізнавання патернів [153], [154], [155]. В одному режимі вони працюють як стандартні схеми пам'яті довільного доступу, Random Access Memory (RAM), і можуть використовуватися для зберігання бінарних даних. Але на відміну від останніх, АП здатна забезпечити також потужний паралельний режим розпізнавання. Цей режим дозволяє шукати всі дані в пристрої паралельно, за одиниці наносекунд [156]. При цьому асоціативна пам'ять, фактично виконує функцію, протилежну традиційному оперативному запам'ятовуючому пристрою. Якщо ОЗП, приймаючи на вхід адресну інформацію, видає зміст відповідної комірки пам’яті, то АП навпаки, за змістом відшукує місце розташування питомих даних або сигналізує про їх відсутність.

На протязі десятиліть АП використовувалася переважно для керування процесом доступу до кеш-пам'яті. Ключовим питанням у цьому застосуванні, яке потрібно якомога скоріше вирішувати, є визначення, чи відбувся "хіт" або "промах". Саме здатність АП до паралельного апаратного пошуку обумовлює можливість її використання для керування кеш-пам'яттю, де критичне значення має швидкодія.

Але згодом АП як апаратне рішення почала знаходити застосування в мережевих технологіях [157]. Обробка пакетів даних, що надходять у мережевий маршрутизатор, залежить від адреси призначення мережі пакета. Через велику кількість потенційних адрес і зростаючі вимоги до

продуктивності, АП почала поширюватися на обробку інформації про мережеві адреси.

Третя, найбільш новітня сфера застосування асоціативної пам’яті пов’язана з докладним аналізом пакетів з метою викриття зловмисної активності у системах захисту інформації.

3.1.1. Асоціативна та трійкова пам'ять з фіксованою логікою

Активне використання АП у мережевих додатках призвело до підвищення інтересу до цієї ніші виробників комерційних пристроїв. В результаті на ринку з’явилися спеціалізовані мікросхеми АП. Наприклад, компанії Net Logic [158] та Music Semiconductor [159] почали виробництво подібних розробок спеціально для мережевого ринку [160].

На жаль спеціалізовані мікросхеми АП мають ті ж самі недоліки, що й будь-які апаратні вироби з фіксованою структурою. Вони забезпечують максимальну швидкодію, але жорстка архітектура обмежує гнучкість використання. Частково подолати це обмеження дозволяє такий різновид асоціативної пам’яті як трійкова асоціативна пам’ять (ТАП), відповідний англомовний термін – Ternary Content Addressable Memory (TCAM). На відміну від традиційної, бінарної АП, у трійковій асоціативній пам’яті до значень "нуль" та "одиниця", які може приймати кожен біт, додається третє значення для порівняння, яке має сенс "не важливо" [82], [81]. Таке розширення надає додаткову гнучкість пошуку, зокрема, спрощує розпізнавання на одному й тому ж апаратному пристрою з фіксованою організацією пам’яті патерни з різною довжиною, а також виконувати найпростіші варіанти гнучкого розпізнавання (див. підр. 2.3.1 "Гнучке розпізнавання").

В роботі [81] розглянуто розробку апаратної системи розпізнавання сигнатур на базі мікросхеми промислової ТАП типу 5512GLQ ємністю 240 Kb виробництва фірми NetLogic Microsystems. Рішення дозволяє розпізнавати 1768 патернів довжиною від 6 до 2189 символів зі словнику

антивірусної системи ClamAV або 1836 патернів довжиною від 10 до 100 символів бази сигнатур МСВВ Snort, забезпечуючи пропускну здатність в 2 Гб/сек. При цьому реалізовані такі види гнучкого пошуку, як зв’язані патерни та операція заперечення.

Крім обмежень, що накладає фіксована структура, спеціалізовані мікросхеми АП страждають також від відсутності поширеної платформи, що спрощувала б їх практичне використання, подібної РУО для ПЛІС. Частково подолати цю проблему дозволили мережеві процесори (див. підр. 1.3.5.2 "Мережеві процесори"). Відомо кілька розробок для сигнатурних систем інформаційного захисту, виконаних на базі NP, які містять в своєму складі спеціалізовані мікросхеми АП [161], [75], [162]. Але за останні 15 років інтерес розробників РАСЗТЗІ до пристроїв асоціативної пам’яті з фіксованою логікою знизився.

Тому на сьогоднішній день найкращою платформою для побудови АП в складі РАСЗТЗІ виявляється програмована логіка.

3.1.2. Асоціативна та трійкова пам'ять на базі ПЛІС

Приклади побудови АП на реконфігуровній логіці можна знайти в роботах [160], [135], [25], [163], [125], [164], [134] та багатьох інших. В публікації [165] наведений приклад використання розробки ТАП на базі ПЛІС.

В переважній більшості розробок асоціативної пам'яті на ПЛІС її швидкодіючою основою, що забезпечує головні властивості, є дискретні або *цифрові компаратори* (ЦК). Тому в даному дослідженні ці два поняття – АП та ЦК – використовуватимуться, фактично, як синоніми.

Перш ніж перейти до розглядання побудови АП на ПЛІС, нагадаємо, що під патерном ми розуміємо фіксовану кількість окремих символів, представлених у вигляді байтів (тобто код символу може приймати довільне значення зі всього діапазону можливих двійкових комбінацій з восьми бітів), під словником патернів, що підлягає розпізнаванню, маємо на увазі всю

сукупність патернів, наявну в базі даних сигнатур, а задача розпізнавання патерну полягає у пошуку відповіді на питання – чи присутній у вхідній послідовності символів такий фрагмент, що повністю збігається з будь-яким зразком зі словнику патернів.

3.1.3. Базова схема на цифрових компараторах

Безпосередньою реалізацією функції виявлення збігу вхідних символів з патерном є паралельний набір цифрових компараторів, кожен з яких порівнює вхідний байт з наперед заданим символом (див. Fig. 2 (a) у [135], та Fig. 2 у [25]). Назвемо схему, що діє в такий спосіб *Базовою схемою на АП або схемою **BsCAM***.

На рис. 3.1 наведена спрощена схема, яка пояснює принцип дії базової схеми розпізнавання на ЦК. Вона містить конвеєр, складений з 8-розрядних регістрів RG_i, компаратори CMP_1 … CMP_3, кожен з яких виконує функцію порівняння з якимось певним символом, та логічний елемент "I", що об'єднує виходи всіх компараторів. Один набір компараторів відповідає якомусь з патернів, що підлягають розпізнаванню. На вхід конвеєру подається послідовність символів, яка має бути проаналізована. В разі збігу фрагменту вхідної послідовності з патерном "ABC" на виході Match з'являється активний сигнал.

Щоб уникнути подальших непорозумінь, слід звернути увагу, що перший символ патерну (який передує інших) порівнюється зі змістом останнього щабля конвеєру.

Найважливіша перевага схеми – її висока швидкодія. В граничному випадку результат на виході з'являється за один такт синхронізації після подачі на вхід послідовності символів, що аналізується. К плюсам схеми слід віднести також регулярність структури та простоту її синтезу.

Найсуттєвіший недолік даної схеми порівняно з іншими підходами – значне споживання ресурсів, а саме логіки та розподіленої пам'яті (тригерів

логічних комірок ПЛІС). Недоліком також є високе енергоспоживання та погана масштабованість за об'ємом словнику патернів.

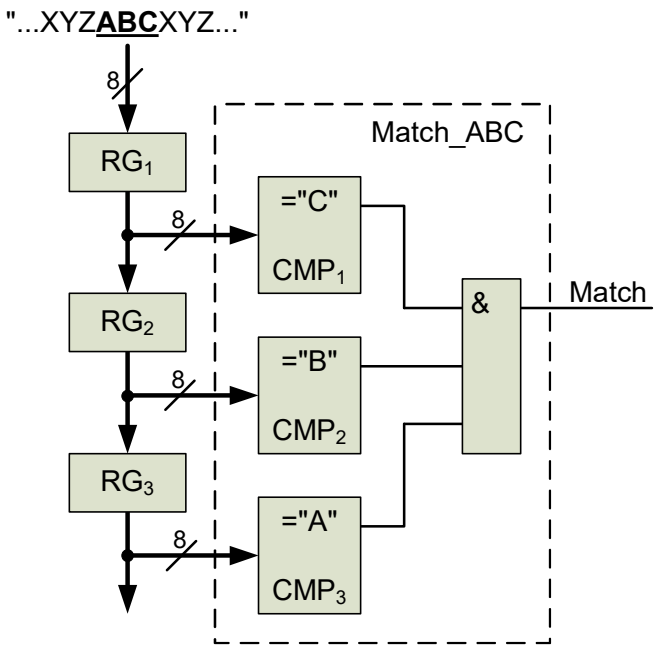


Рис. 3.1. Безпосереднє розпізнавання компараторами

Причини походження вказаних проблем а також шляхи їх подолання розглянуті нижче.

3.1.4. Особливості реалізації схеми з цифрових компараторів на ПЛІС

Схема на рис. 3.1 виглядає досить простою. Щоб краще зрозуміти труднощі її практичного застосування реконфігуровними засобами розглянемо наступний факт. Вихідні лінії регістра RG_1 подаються на компаратори, що відповідають останньому символу кожного з патернів словнику сигнатур, виходи регістра RG_2 – на компаратори передостаннього символу кожного з патернів і т.д. Оскільки словник сигнатур сучасних МСВВ містить велику кількість патернів, здатність навантаження ($fan\text{-}out_{max}$) виходів цих регістрів, синтезованих на штатних компонентах ПЛІС, виявляється недостатньою. (Для ПЛІС Xilinx ця величина дорівнює 16). Якщо вона перевищується, різко зніжується максимальна частота

функціонування схеми [125]. З іншого боку, довжина цифрових ліній, що зв’язують велику кількість логічних елементів, розподілених по площині кристалу ПЛІС, призводить до затримок розповсюдження сигналу, внаслідок чого тактова частота також зніжується.

Для вирішення проблеми створюють конвеєр з декількох щаблів, на кожному з котрих вихідні сигнали розгалужуються на множину входів D-тригерів наступного щаблю (див. рис. 3.2, або, наприклад, Fig. 4 у [163].

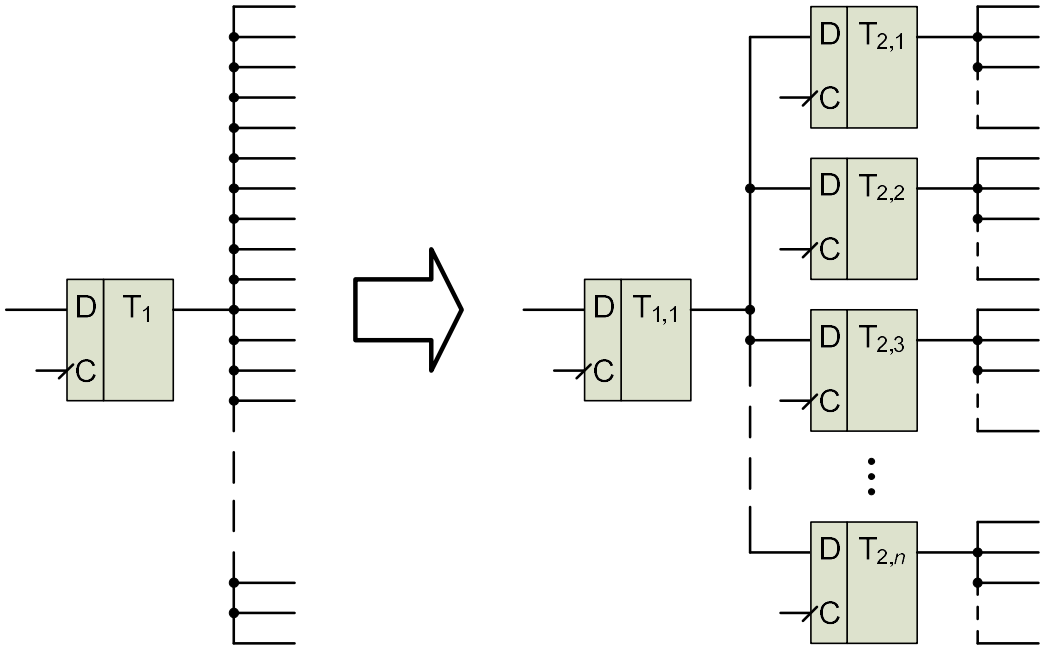


Рис. 3.2. Вирішення проблеми навантаження виходів

Таке рішення дозволяє розподілити виходи регістрів RG_i на довільну кількість компараторів без зниження тактової частоти.

Інша проблема базової схеми на компараторах пов'язана з надвеликою кількістю входів логічних елементів "І", оскільки довжина патернів може сягати десятків символів, а ЛТ сучасних ПЛІС, на яких реалізуються ці елементи, мають лише від 4 до 8 входів. Дана складність також вирішується шляхом конвеєризації (див. рис. 3.3, Fig. 2 (a) у [135], або Fig. 5 у [163]).

В результаті аналізу згаданих складнощів можна побачити, що при збільшенні кількості та довжини патернів, які мають бути розпізнані, розглянуті проблеми, по-перше, призводять до нелінійного зростання

апаратних витрат, тобто обумовлюють погану масштабованість підходу за об’ємом словнику патернів та довжиною патернів.

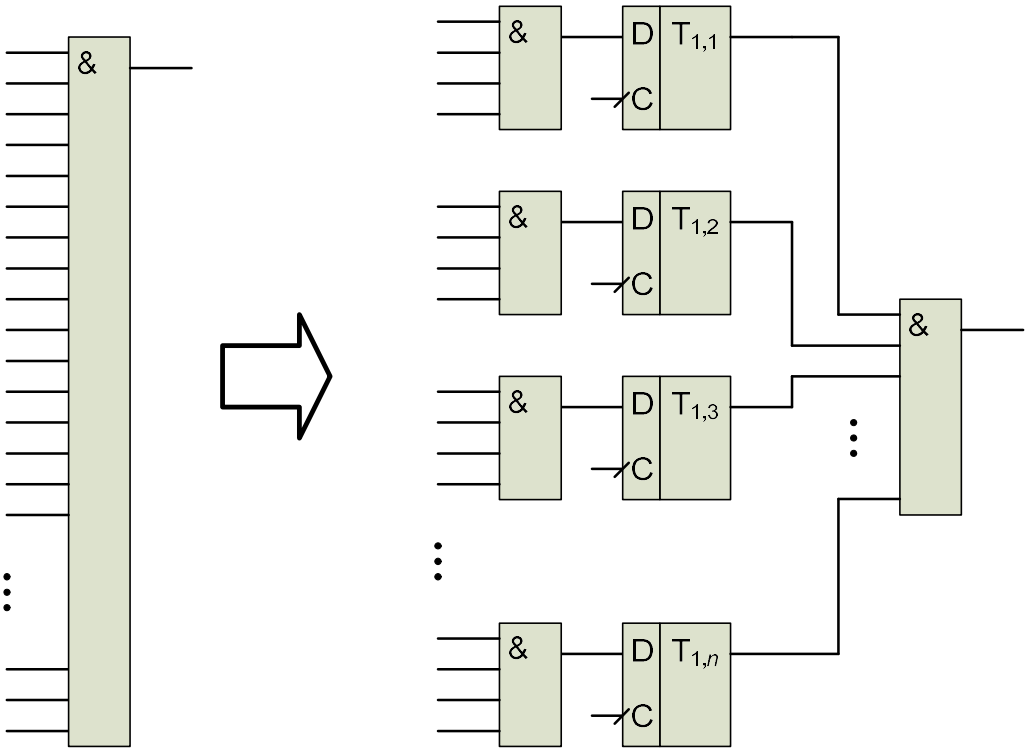


Рис. 3.3. Вирішення проблеми завеликого числа входів логічних елементів "І"

По-друге, конвеєризація додає затримку до поширення сигналу вздовж цифрової схеми на кількість тактів, що дорівнює кількості щаблів конвеєру, тобто підвищують латентність схеми з ростом числа та довжини патернів.

По-третє, конвеєризація підсилює проблему високого енергоспоживання базової схеми на ЦК.

Тому дослідниками даного підходу було запропоновано низку прийомів та технічних рішень, які дозволили суттєво знизити ресурсоємність та покращити інші показники ефективності даного напрямку побудови реконфігуровних засобів розпізнавання.

3.1.5. Зменшення ресурсних витрат

Головна ідея скорочення апаратних витрат компараторної схеми полягає у повторному використанні одних й тих самих компараторів для

розпізнавання різних патернів. В схемі на рис. 3.4 компаратор CMP_3 задіяний для розпізнавання одночасно двох патернів: "AB" та "AC" (див. Fig. 2 (a) у [125]).

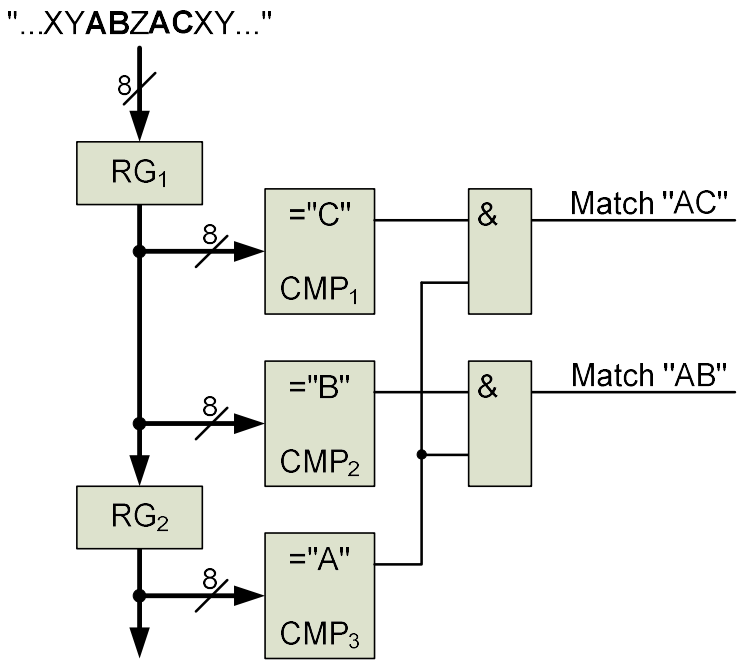


Рис. 3.4. Сумісне використання компаратора

Подальший розвиток ідеї повторного використання призводить до граничного рішення, у якому для кожного символу словника сигнатур використовується лише один компаратор, а всі комбінації, потрібні для розпізнавання кожного з патернів, формуються за допомогою відповідних цифрових схем затримки (ЦСЗ) та схем "І", як подано на рис. 3.5).

Тут DEL(1) і DEL(2) – ЦСЗ на один і на два такти відповідно.

Таке рішення отримало окрему назву *Схема розпізнавання на базі декодованої асоціативної пам'яті* або *схема* ***DCAM*** (Decoded Content-Addressable Memory), тому що повний комплект з 256 компараторів для кожної можливої комбінації бітів у вхідному байті, фактично, являє собою дешифратор (decoder) конфігурацією 8/256 [125].

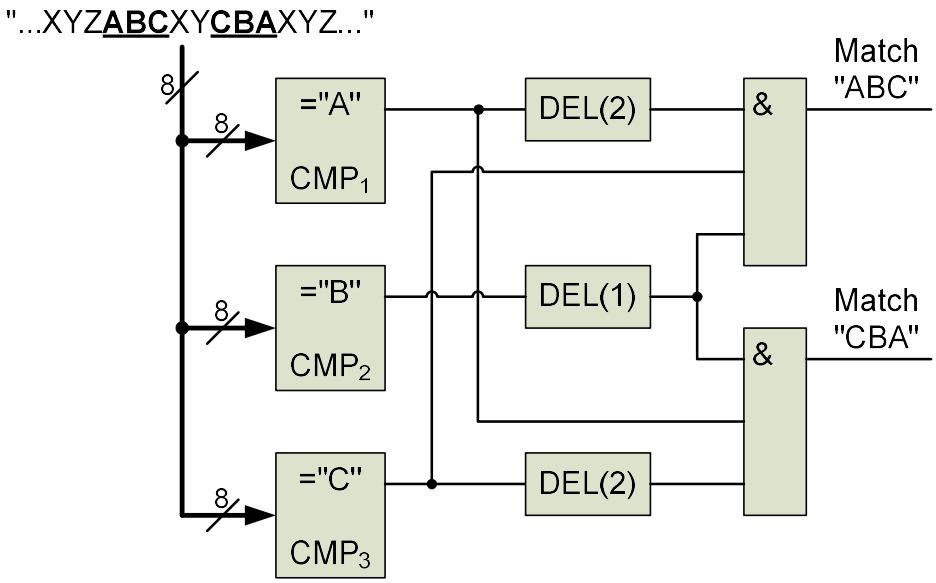


Рис. 3.5. Схемне рішення DCAM

Очевидними перевагами рішення DCAM, по-перше, є скорочення числа 8-розрядних компараторів з величини, що чисельно дорівнює сумарної кількості символів у словнику сигнатур до 256 одиниць; по-друге, відпадає потреба в 8-розрядному конвеєрі довжиною, що дорівнює розміру найдовшого патерну в словнику. Негативною рисою, проте, виявляється необхідність синтезу великої кількості ЦСЗ, яких, теоретично, також потрібна кількість, що дорівнює сумарному числу символів у словнику сигнатур. Але на практиці цю кількість можна скоротити, зокрема, за рахунок повторного використання однакових символів, що займають ту ж саму позицію у різних патернах (наприклад, затримка DEL(1) в схемі на рис. 3.5).

Іншим фактором, що сприяє використанню техніки DCAM, є той факт, що для більшості сучасних ПЛІС в наявності є бібліотечні елементи, які надають можливість економно синтезувати ЦСЗ. Наприклад, в САПР фірми Xilinx існують типові компоненти FD, SRL16 та SRL32 які дозволяють використовуючи відповідно один тригер, 4-розрядну та 6-розрядну (або 8-розрядну) логічну таблицю створити ЦСЗ на один такт або на довільну величину від 1 до 16 / 32 тактів [166], [167].

Зауважимо, що використання техніки повного дешифрування вхідних символів у однодротові лінії використовується не лише в схемах на цифрових компараторах та асоціативної пам'яті. Наприклад, в роботі [168] та сама техніка застосована при реалізації гнучкого розпізнавання з використанням недетермінованих скінченних автоматів.

3.1.6. Подальше зменшення ресурсних витрат

Подальшим напрямом зменшення складності схеми розпізнавання на базі DCAM є використання техніки часткового розпізнавання.

Її суть полягає у наступному. Щоб зменшити витрати апаратних ресурсів схемою DCAM, довгі патерни розбиваються на коротші фрагменти, які розпізнаються послідовно. При цьому не потрібно затримувати дані з дешифратора на велику кількість тактів. Натомість достатньо лише затримувати сигнал часткового збігу.

На рис. 3.6 наведена схема розпізнавання 31-символьного патерну, що побудована за подібним принципом (див. Fig. 3 у [164]).

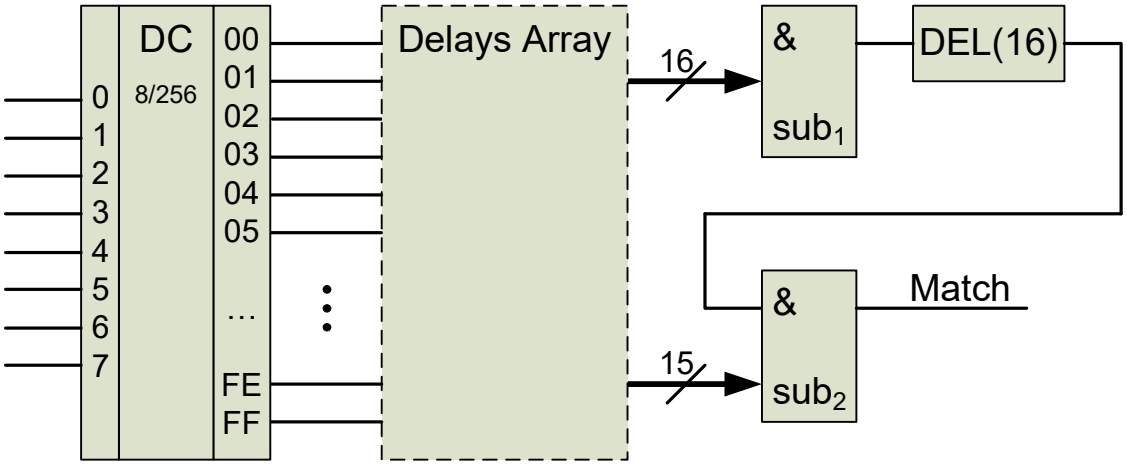


Рис. 3.6. Схемне рішення DpCAM

Таке рішення згідно пропозиції авторів публікації [164] отримало назву *Схема розпізнавання на базі частково декодованої асоціативної пам'яті* або *схема **DpCAM*** (Decoded partial Content-Addressable Memory).

Послідовність символів, що аналізується, подається на вхід дешифратора DC. Перетворені на однодротові лінії символи поступають на масив цифрових схем затримки Delays Array. Лінії, що відповідають першим 16 символам питомого патерну с виходів відповідних ЦСЗ подаються на логічний елемент sub_1. Лінії, що відповідають останнім 15 символам питомого патерну с виходів відповідних схем затримки подаються на логічний елемент sub_2. На цей же елемент подається сигнал з виходу елемента sub_1 після затримки на 16 тактів схемою DEL(16). Кожен з логічних елементів "I" на шістнадцять входів може бути синтезований на чотирьох 4-входових ЛТ або на двох 8-входових.

Отже за рахунок додавання однієї ЦСЗ на 16 тактів ця схема дозволяє на кожному патерні довжиною від 17 до 31 символу зберігати 15 ліній затримки довжиною від 17 до 31 тактів. В загальному випадку розпізнавання патерну довжиною L можна здійснити шляхом K-кратного каскадування по S символів, де $K = \lfloor L/S \rfloor$. Для цього потрібно додатково K схем затримки на S тактів замість звільнених ЦСЗ кількістю від 1 до $L - K \cdot S$ та довжиною від $S + 1$ до L тактів.

Іншим шляхом скорочення вживаних ресурсів схемою DpCAM є повторне використання фрагментів патернів (префіксів та інфіксів) довжиною в 16 або 32 символи, які збігаються та розташовані на одній і тій самій позиції. Тобто, для прикладу на схемі рис. 3.6 всі патерні, що мають однаковими перші 16 символів, можуть сумісно використовувати схему sub_1, тобто крім схем затримки економляться також логічні ресурси для створення схем "I".

К недолікам техніки DpCAM можна віднести незначну залежність її ефективності від змісту словника сигнатур.

3.1.7. Розпаралелювання

Як було показано, асоціативної пам'яті на цифрових компараторах в якості засобу розпізнавання притаманна погана масштабованість за об’ємом

словнику патернів. Розглянемо, як масштабується цій підхід стосовно швидкодії.

Для прискорення роботи базової схеми розпізнавання (рис. 3.1) в m разів за допомогою розпаралелювання на m потоків необхідно об'єднати m таких схем та подати на кожну з них вхідну послідовність зі зсувом на один символ. На рис. 3.7 наведений приклад для $m = 4$ (див. Fig. 2 (b) у [135]). Тут кожен з блоків SCH_i містить схему Match_ABC, що охоплена пунктиром на рис. 3.1.

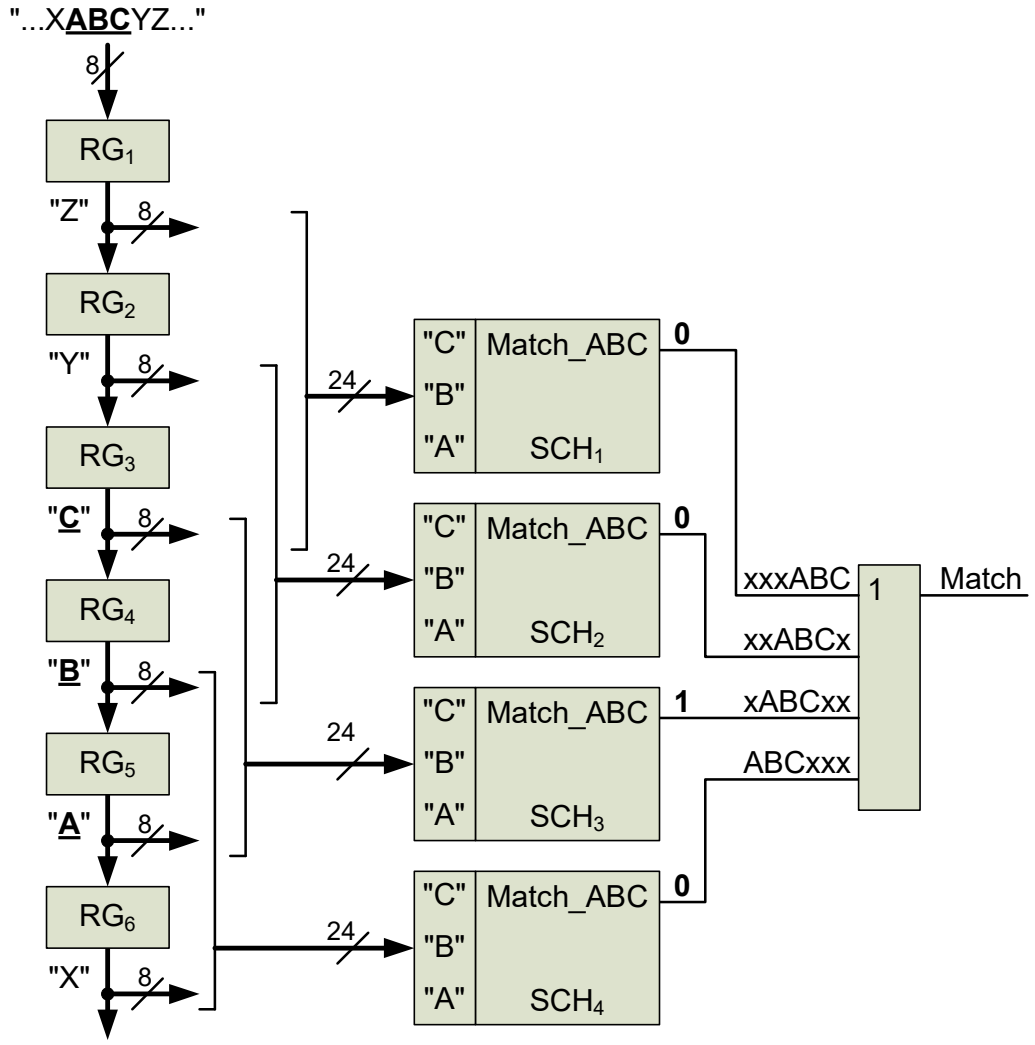


Рис. 3.7. Схема 4-потокового розпаралелювання

Назвемо технічне рішення на базі паралельної роботи блоків АП *схемою **ParCAM***.

Темп пересування інформації у конвеєрі в чотири рази вищий за швидкість блоків розпізнавання. Тобто вхідна послідовність просувається на чотири символи вперед на кожному циклі обробки інформації блоками SCH_i.

Як можна побачити, *m*-кратне прискорення призводить до *m*-кратного збільшення апаратних ресурсів при несуттєвих додаткових витратах. Подальше дослідження свідчить, що використання описаних вище технік зменшення витрат у компараторних схемах розпізнавання в разі їх розпаралелювання призводить до більш ефективної економії ресурсів порівняно з лінійним ростом. Тому можна стверджувати, що схеми на ЦК має добру масштабованість за пропускною здатністю.

На рис. 3.8 наведений приклад схеми розпаралелювання рішення DCAM для $m = 2$. Її прототипом є Fig. 5 у роботі [125].

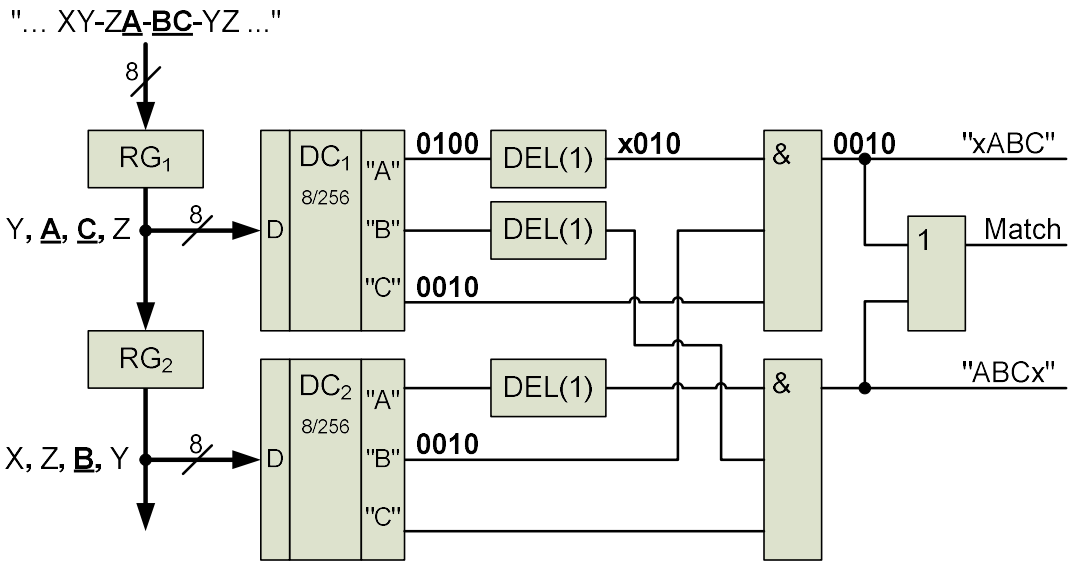


Рис. 3.8. Схема двократного розпаралелювання DCAM

Назвемо технічне рішення на базі паралельної роботи декодованої АП *схемою* ***ParDCAM***.

Вхідна послідовність просувається на два символи вперед на кожному циклі розпізнавання. Вихід Match сигналізує про збіг вхідної послідовності з патерном. Додаткові виходи xABC та ABCx дозволяють з’ясувати, з яким

зсувом з’явилася питома комбінація символів – парним або непарним, що заздалегідь не відомо.

Як можна бачити, при довжині патерна в три символи у схемі присутні всього три ЦСЗ по одному такту. Притому в відповідній однопотоковій схемі (дивиться частину рис. 3.5, що працює на вихід Match_ABC), яка вдвічі повільніша, задіяна одна затримка на один такт и одна – на два такти, тобто, фактично, така сама кількість ресурсів. Цей ефект зменшення кількості та довжин ЦСЗ обумовлений тим фактом, що вхідна послідовність додатково зсувається між тактами розпізнавання. Отже схема двократного прискорення, що наведена на рис. 3.8, збільшує потрібні ресурси не вдвічі, а трохи менше, тобто по швидкодії вона масштабується за сублінейним законом.

3.1.8. Небайтова розрядність обробки даних

Ще одним напрямком досліджень стосовно зменшення апаратних ресурсів при синтезі блоків розпізнавання взагалі та при побудові схем на ЦК зокрема є відхід від побайтового принципу обробки інформації.

В роботі [163] висловлена та досліджена ідея, яка базується на використанні так званих півбайтових компараторів (Half-Byte Comparator – HBC). При цьому кожен символ, що аналізується, розглядається як складений з двох півбайтів – старшого та молодшого.

Назвемо таке рішення *схемою **Hbc***.

На рис. 3.9 наведений приклад схеми розпізнавання патерну "ABCA", в якої одночасно використані дві техніки: Hbc та DCAM.

Назвемо таке технічне рішення *схемою **HbcDCAM***.

Старший півбайт (розряди з 7-го по 4-й) вхідної послідовності подається на півбайтовий компаратор HBC_1; молодший півбайт (розряди з 3-го по 0-й) – на півбайтові компаратори HBC_2 – HBC_4; виходи компараторів після певних затримок об’єднуються схемою "I".

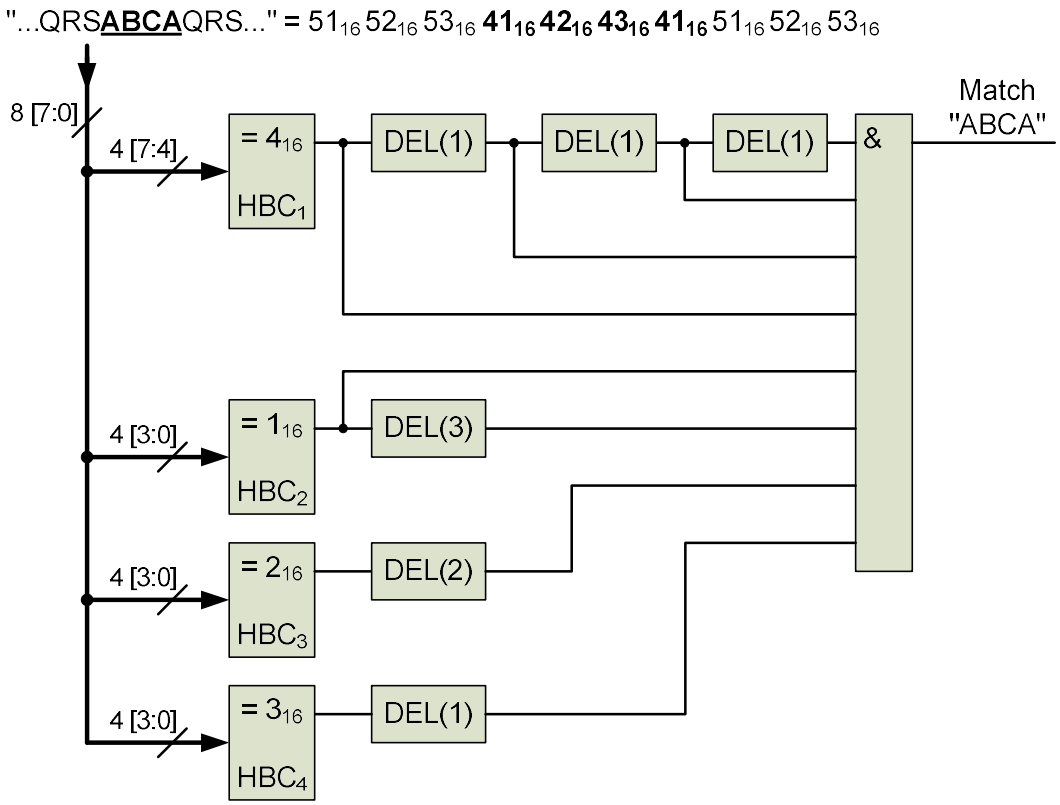


Рис. 3.9. Схемне рішення HbcDCAM

Як можна бачити, замість трьох повнобайтових компараторів в схемі використані всього чотири півбайтові компаратори. Але головною перевагою техніки виявляється той факт, що замість одного дешифратора на 8 входів та 256 виходів для розпізнавання всіх 4-бітних комбінацій півбайтів достатньо двох дешифраторів організацією 4/16. Тобто на вхід масиву ЦСЗ подаються всього 32 дроти замість 256.

Зменшенню задіяних ресурсів сприяє також той факт, що символами, які входять до складу патернів словників сигнатур систем виявлення вторгнень, у більшості випадків є літери англійського алфавіту в стандартному кодуванні ASCII. Це означає, що для половини алфавіту (окремо для малих та великих літер) старші півбайти мають однакові значення (4_{16} – для літер від "A" до "P", 5_{16} – для літер від "Q" до "Z", 6_{16} – для літер від "a" до "p" та 7_{16} – для літер від "q" до "z"). Отже велика кількість ЦСЗ може бути зекономлена за рахунок їх сумісного використання.

Також до переваг схеми слід віднести можливість синтезувати півбайтові компаратори на одній ЛТ, навіть на 4-входовій.

Можна перевірити, що техніка використання півбайтових компараторів цілком сумісна с технікою розпаралелювання з метою прискорення розпізнавання. При цьому ефект зменшення кількості та довжини ЦСЗ також спостерігається.

На рис. 3.10 наведена схема розпізнавання патерну "ABC" з двопотоковим розпаралелюванням, яка використовує півбайтові компаратори та декодовану АП.

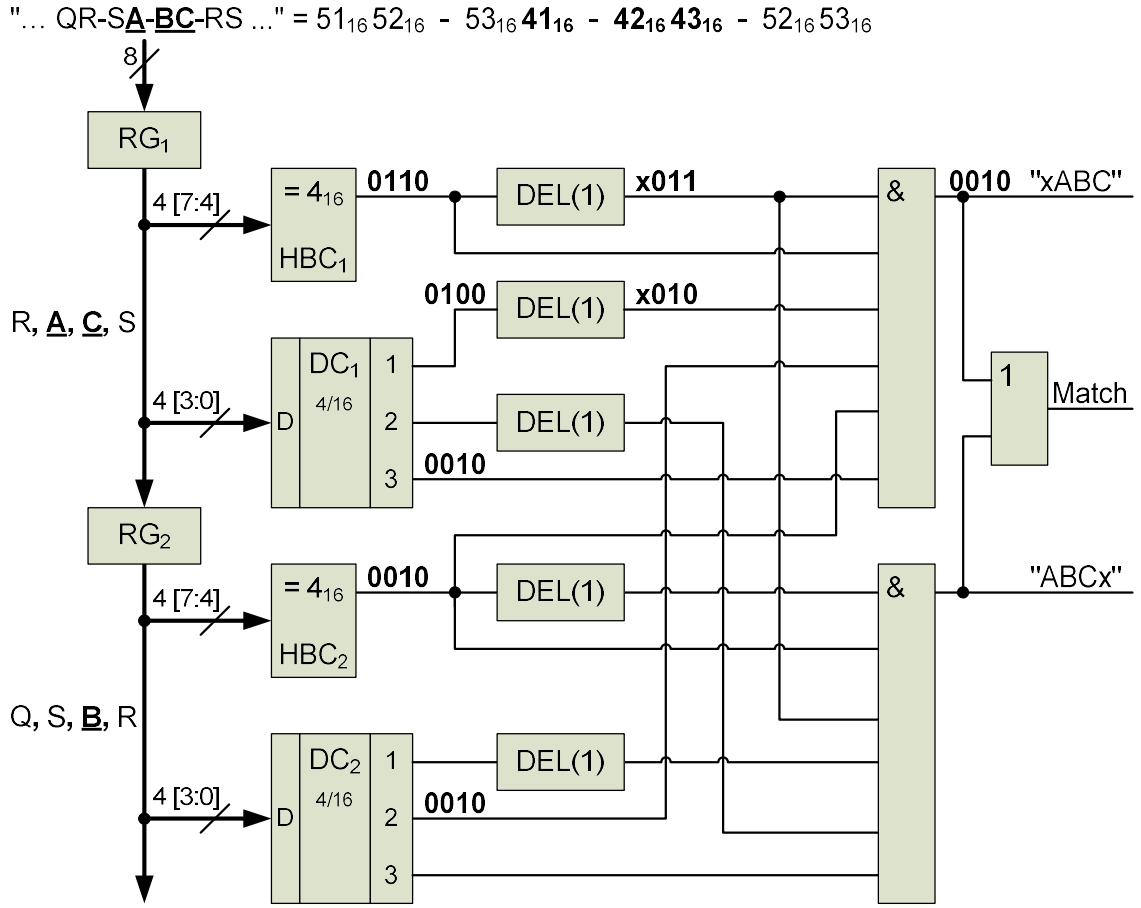


Рис. 3.10. 2-потокове розпаралелювання схеми HbcDCAM

Назвемо таке технічне рішення *схемою* ***ParHbcDCAM***.

Як і в схемі на рис. 3.8 вхідна послідовність тут просувається на два символи вперед на кожному циклі розпізнавання. При цьому апаратні

ресурси збільшені менш, чим вдвічі. Так, якщо дешифраторів 4/16 потрібно рівно два комплекти, то на один такт схема потребує також п’ять ЦСЗ, як і базова схема на рис. 3.9, яка вдвічі повільніша. Тобто паралельна схема на півбайтових компараторах масштабується стосовно швидкодії також за сублінейним законом.

В загальному випадку інформація, що обробляється в схемі розпізнавання, може бути згрупована не тільки по 8 або по 4 біти. Кількість двійкових розрядів, що утворюють елементарну частку інформації, в граничному випадку може дорівнювати одиниці. Тобто елементарні символи, з яких складаються патерни, можуть кодуватися одним бітом.

В роботі [169] пропонується реалізація на ПЛІС схеми розпізнавання, принцип дії якої ґрунтується саме на бітовому рівні. В її основу закладений математичний апарат так званих бінарних діаграм рішень (БДР), відповідний англомовний термін – Binary Decision Diagram (BDD).

БДР – це механізм представлення булевої функції $f(x_1, x_2, \dots x_n)$ у вигляді орієнтованого ациклічного графа, який складається з декількох вузлів рішень і термінальних вузлів двох типів (0-термінал і 1-термінал). Кожен внутрішній вузол рішень на рівні i позначений булевою змінною x_i та має по два нащадки – так звані молодший нащадок і старший нащадок. Перехід від внутрішнього вузла x_i до молодшого або старшого нащадка виконується в залежності від поточного значення змінної x_i – 0 або 1 відповідно. Якщо для заданого набору значень $x_1, x_2, \dots x_n$ шлях від кореневого вузла призводить до 1-терміналу або 0-терміналу, це означає, що функція f приймає для цього набору змінних значення 1 / 0 відповідно [170]. На рис. 3.11 наведений приклад БДР для булевої функції $f = (x_1 \vee x_2) \wedge x_3$ [171].

Але найбільш цікавою для застосування є так звана скорочена впорядкована бінарна діаграма рішень (СВБДР), яка здобувається зі звичайної впорядкованої БДР шляхом злиття ізоморфних підграфів та вилучення вузлів, у яких обидва нащадки ізоморфні. Користь скороченої БДР полягає в тому, що вона однозначно представляє конкретну функцію для

заданого порядку змінних. СВБДР має тільки два термінальних вузла: один 1-термінал та один 0-термінал. На рис. 3.12 наведена СВБДР для БДР на рис. 3.11.

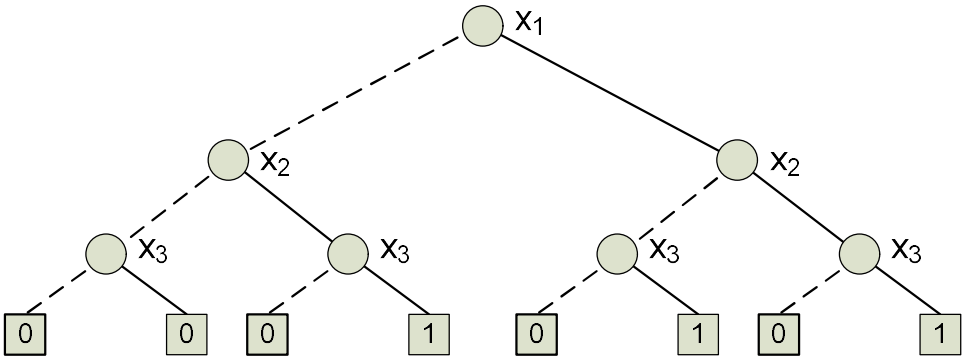


Рис. 3.11. Приклад бінарної діаграми рішень

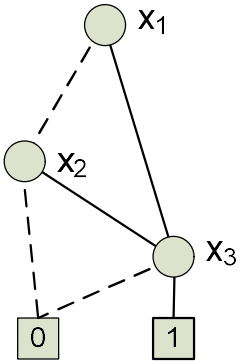


Рис. 3.12. Скорочена впорядкована бінарна діаграма рішень

Тобто СВБДР дозволяє представити бінарну діаграму рішень булевої функції в однозначному та компактному вигляді.

Якщо представити схему розпізнавання у вигляді чорної скриньки, на вхід якої побітно подається фрагмент вхідної послідовності символів, а вихідна бітова змінна приймає значення 1 або 0 в залежності от того, чи збігається вхідний рядок з будь-яким з паттернів, закладених в булеву функцію чорної скриньки, то БДР цієї функції відобразить логіку роботи такої схеми на бітовому рівні.

В роботі [169] запропонована 5-крокова техніка побудови пристроїв на базі БДР для її реалізації на ПЛІС. В процесі виконання техніки з вхідної бази

даних сигнатур (на прикладі МСВВ Snort) витягуються патерни, для кожного символу котрих створюються БДР, які потім зливаються в єдину БДР рядка, далі операція повторюється для всіх патернів словнику. Надлишковість множини патернів призводить до того, що окремі гілки діаграми дублюють одна одну. Автори техніки використовують цей ефект наступним чином. Вони виокремлюють одну так звану *основну БДР*, яка об'єднує спільні для всіх патернів біти, а також низку *неосновних БДР*, які відображають відмінності кожного патерну. Внаслідок надмірності словнику патернів кількість неосновних БДР зазвичай менша за кількість патернів. Далі по здобутим бінарним діаграмам рішень будуються компаратори. Для основної БДР компаратор не відрізняється від звичайних, які використовуються в асоціативної пам'яті. По неосновним діаграмам компаратори будуються, використовуючи ЛТ в якості мультіплексорів.

На рис. 3.13 наведений приклад схеми розпізнавання, побудованої з використанням механізму БДР, що розпізнає патерни довжиною по чотири символи (32 біти) [169].

В роботі [169] таке рішення отримало назву *асоціативної пам'яті на базі БДР* – ***BCAM*** (BDD-based CAM).

Вхідна послідовність представлена бітовими сигналами, з яких на входи компараторів (одного основного Com.CMP та чотирьох неосновних N/com.CMP$_i$) подається потрібна кількість бітів. Виходи неосновних компараторів, які підтверджені сигналом с основного, свідчать про присутність у вхідній послідовності відповідного питомого рядка.

На наступному кроці техніки неосновні компаратори подальше модифікуються подібним чином. Тобто кожен з них перетворюється в основний та кілка неосновних компараторів. Процедура повторюється рекурсивно до повної оптимізації схеми. В результаті з початкової БДР створюється компактна та ефективна апаратна схема BCAM.

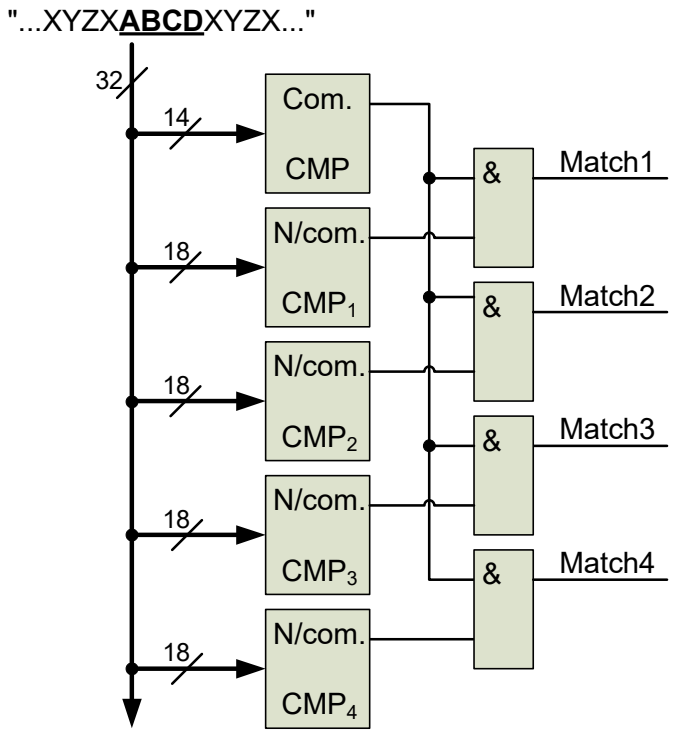


Рис. 3.13. Схемне рішення BCAM

За результатами практичної реалізації та тестових випробувань автори розробки стверджують, що оптимізована на бітовому рівні схема, побудована з використанням розробленої техніки, потребує в два рази менше ресурсів ПЛІС порівняно з рішенням ParDCAM.

До недоліків розглянутої техніки слід віднести її специфічність і як слід несумісність з іншими техніками побудови схем на базі ЦК, а також складність розпізнавання патернів різної довжини. Тому вона здобула більш значне поширення для побудови реконфігуровних засобів класифікації пакетів ніж для схем розпізнавання рядків.

3.1.9. Кластеризація словника патернів

Розглянуті в попередніх підрозділах техніки дозволяють покращити показники швидкодії та економії ресурсів порівняно з базовою схемою. Але при їх практичної реалізації одна з головних негативних рис асоціативної пам'яті – погана масштабованість по об'єму словника сигнатур – в деяких випадках виявляється критичною. В роботі [125] розглянутий приклад

практичної розробки модуля розпізнавання для МСВВ на базі ЦК, коли множина патернів містила загалом близько 100000 символів, закодованих 8-бітним кодом. При цьому зі всіх можливих 255 комбінацій бітів були задіяні лише 250. Здатність навантаження виходів дешифратора, яка з'явилася потрібною, досягла значення 40. Довжина ліній з'єднань також виявилася непридатне великою.

Єдиним рішенням, яке здатне принципово здолати проблему поганої масштабованості за об’ємом словнику патернів, виявляється кластеризація, тобто розділення множини патернів на більш дрібні групи (за якимось принципом). Тоді єдина велика схема розпізнавання розпадається на кілька підсхем меншого розміру, на вхід кожної з яких одночасно подаються питомі символи. В результаті, по-перше, за рахунок зменшення розміру підсхем знижається вимога до здатності навантаження на виходи дешифраторів, по-друге, кожна окрема підсхема займає меншу площину на кристалі ПЛІС, що призводе до скорочення довжини провідникових з’єднань [125].

Недоліком такої техніки є необхідність дублювати дешифратори, тобто синтезувати для кожної підгрупи окремий перетворювач вхідних символів в унітарний код. Частково пом'якшити ситуацію дозволяє використання такого алгоритму кластеризації, який призводе до скорочення алфавіту підмножин патернів в підсхемах, що зменшує витрати на дешифратори. Найпростішім прикладом подібного алгоритму виявилося звичайне сортування патернів. Запропонований в роботі [125] більш складний алгоритм не надав суттєвих переваг ані по економії ресурсів, ані по швидкодії.

Підтвердженням ефективності техніки кластеризації можна зарахувати результати, отримані в згаданій вище практичної розробці. Множину патернів розміром біля 100000 символів вдалося розбити на 24 групи по 54 символи в кожної в середньому, внаслідок чого розрядність дешифраторів в підсхемах скоротилася до 6.

3.1.10. Здатність до динамічної реконфігурації

Як бачимо по розглянутих рішеннях, при створенні засобів розпізнавання на цифрових компараторах інформація про патерни фактично "прошивається" у апаратну схему на самому нижчому рівні. З цього випливає ще один важливий недолік асоціативної пам’яті – занадто висока складність, тобто фактична неможливість реалізації режиму динамічної реконфігурації.

В роботі [160] розроблений оригінальний метод синтезу цифрових компараторів на логічних таблицях LUT. Метод дозволяє, використовуючи власну розробку авторів JBit [172] на базі фірмової бібліотеки XBI (Xilinx Bitstream Interface) та здатність деяких виробів фірми Xilinx здійснювати так звану часткову реконфігурацію (partial reconfiguration), модифікувати зміст ЛТ під час функціонування ПЛІС, не заважаючи роботі інших областей інтегральної схеми. Крім того автори стверджують, що метод дозволяє синтезувати схеми асоціативної пам’яті на ПЛІС у 3-4 рази більш ефективно порівняно з іншими підходами. Але за минулі майже 20 років метод не набув поширення та практичного застосування.

3.1.11. Висновки щодо підходу на основі асоціативної пам'яті

Результати формалізації опису властивостей схем розпізнавання на базі АП та ЦК зведені в табл. 3.1.

Головна перевага АП та ЦК при побудові реконфігуровних засобів інформаційної безпеки – це *висока пропускна здатність*.

Найсуттєвіший недолік – *значне споживання ресурсів*, а саме логіки та розподіленої пам'яті (тригерів логічних комірок ПЛІС). Як наслідок – високе енергоспоживання.

Важлива особливість реалізації АП на ПЛІС – занадто високі вимоги щодо здатності навантаження компонентів програмованої логіки та надмірна довжина цифрових ліній. Конвеєризація, яка є вирішенням вказаних проблем, призводить до нелінійного зростання апаратних витрат при збільшенні

кількості патернів, що обумовлює другий важливий недолік підходу – *погану масштабованість за об'ємом словнику патернів.*

Таблиця 3.1

Формалізований опис властивостей підходу на базі АП/ЦК

№	Показник		Якісна оцінка властивості
1.	Витрати логіки		Суттєвий недолік
2.	Витрати пам'яті	розподіленої	Суттєвий недолік
3.		блочної	Суттєва перевага
4.		зовнішньої	Суттєва перевага
5.	Швидкодія		Суттєва перевага
6.	Передбачуваність пропускної здатності		Суттєва перевага
7.	Функціональні показники	здатність протидіяти атакам на РАСЗТЗІ	Суттєва перевага
8.		динамічна реконфігурація	Суттєвий недолік
9.		селективне розпізнавання	Суттєвий недолік
10.		режим запобігання вторгнень (для МСВВ)	Суттєва перевага
11.	Масштабованість	за пропускною здатністю	Помірна перевага
12.		за об'ємом словнику патернів	Помірний недолік
13.		за довжиною патернів	Помірний недолік
14.	Використання надлишковості		Суттєва перевага
15.	Суттєвий недолік, який зводить нанівець головні переваги підходу		Завелике споживання ресурсів

Розробники доклали значних зусиль щодо подолання вказаних проблем. Техніки попереднього дешифрування DCAM та часткового дешифрування DpCAM дозволяють дещо знизити апаратні витрати підходу, ефективно використовуючи *надмірність словнику патернів*. Ще скоротити ресурси дозволяє використання півбайтових компараторів. Оптимізація на бітовому рівні (з використанням апарату БДР) зменшує витрати майже вдвічі, але конфліктує з іншими техніками, зокрема DCAM. Але, не зважаючи на докладені зусилля, значних успіхів щодо зниження вартісних показників досягти не вдалося.

Проте досліджений підхід *гарно масштабується за пропускною здатністю*. Розпаралелювання на *m* потоків призводить до *m*-кратного прискорення в наслідок майже лінійного зростання витрат, а поєднання з деякими зі згаданих вище технік апаратні витрати збільшуються навіть за сублінейним законом.

Проблему поганої масштабованості за об’ємом словника сигнатур вирішують за допомогою кластеризації множини патернів, внаслідок чого велика схема розпізнавання розпадається на кілька підсхем меншого розміру. Застосування спеціального методу кластеризації дозволяє додатково скоротити сукупні апаратні витрати.

Важливим фактом є те, що зміст вхідних даних ніяким чином не впливає на характер роботи схеми розпізнавання на основі АП. Це, по-перше, зумовлює високий рівень такого показника продуктивності, як *передбачуваність пропускної здатності*, по-друге, робить системи, які використовують даний підхід, *невразливими до атак на РАСЗТЗІ*, що є одним з функціональних показників. *Динамічну реконфігурацію*, на жаль, впровадити в схему на ЦК практично неможливо.

3.2. Використання фільтра Блума на базі геш-функцій

Операція гешування (від англ. "to hash" – рубати м'ясо) багато років використовується в техніці. В галузі захисту інформації ця процедура найчастіше застосовується в криптографічних засобах. Але останнім часом гешування використовується також для розпізнавання сигнатур відомих атак в якості базової технології для певної схеми обробки даних.

В 1970 році Бёртон Говард Блум опублікував роботу [173], метою якої було забезпечити більш компактне зберігання гешованих даних в застосуваннях, коли є прийнятною деяка частка хибних спрацювань. Винахід виявилася вдалим и знайшов багато практичних впроваджень спочатку в сфері традиційного застосування гешування, а згодом і в суміжних галузях техніки. Запропонована схема обчислень отримала назву по імені свого

автора і стала відома розробникам як *фільтр Блума* (ФБ), відповідна англомовна назва – Bloom Filter (BF).

Перша публікація про застосування цієї схеми обробки даних для розпізнавання тексту з використанням апаратного прискорювача на базі ПЛІС типу FPGA, згідно відомостям автора, з'явилася в 1993. То була робота [174], в якій втім власне назва "фільтр Блума" не згадувалася. Згодом подібні дослідження почали проводитися все частіше. Причому більшість з них була присвячена мережевим застосуванням. Огляд різноманітних застосувань ФБ галузі комп'ютерних мереж можна знайти, наприклад, в публікації [175].

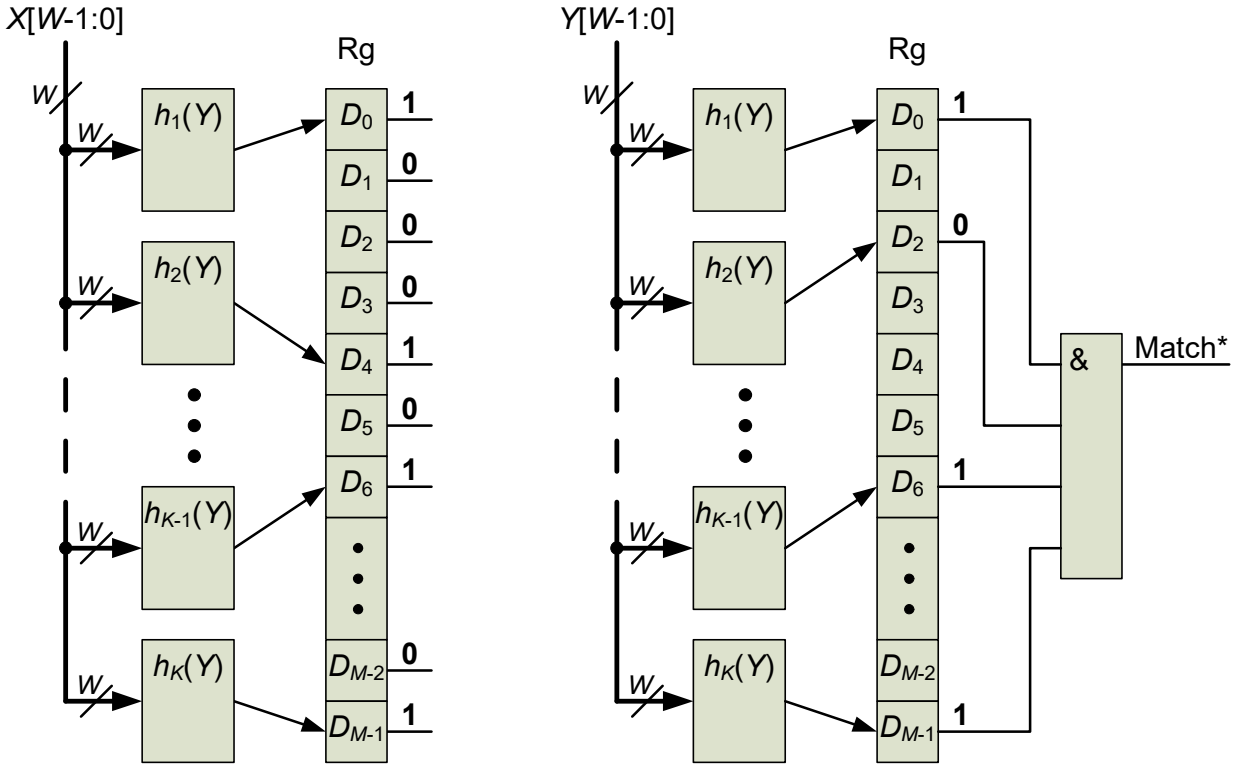
Починаючі с 2004 року, фільтр Блума активно використовується в галузі інформаційної безпеки, зокрема для побудови модуля розпізнавання МСВВ, чому посприяла низка відомих робот проф. Дж. Локвуда з Вашингтонського університету. Численні приклади використання ФБ в галузі захисту інформаційної висвітлені в вичерпному дослідженні [176].

3.2.1. Основні принципи побудови та використання фільтра Блума

Фільтр Блума – це абстрактний пристрій, який дозволяє визначити чи належить певний фрагмент даних до заданого набору – зокрема, чи збігається задана послідовність символів с якимось з патернів, що входять до складу словнику патернів [177].

ФБ складається з двох ключових компонент (рис. 3.14): комплекту з K блоків, що обчислюють геш-функції $h_1(x)$, $h_2(x)$, $h_3(x)$, …, $h_K(x)$ та масиву з M бітових комірок (компонент Rg на рисунку). Назвемо цей масив регістром бітів (РБ) В початковому стані він заповнений нулями.

На етапі програмування (або навчання) на входи всіх геш-функцій послідовно подається кожен з N елементів словнику патернів (довжиною W бітів). Для кожного елемента обчислюються значення всіх K геш-функцій, отримані значення яких інтерпретуються як номер (адреса) комірки у РБ. У відповідні комірки заносяться значення "1".



а) програмування б) розпізнавання

Рис. 3.14. Принцип дії фільтра Блума.

В процесі функціонування фільтра Блума на його вхід подається фрагмент вхідної послідовності символів (також довжиною W бітів). Для нього обчислюються значення всіх K геш-функцій. По отриманим адресам здійснюється звернення до комірок РБ. Якщо у всіх позиціях, на які вкажуть геш-функції, містяться одиниці, вважається, що вхідна комбінація символів з певною вірогідністю співпадає з одним з патернів, що приймали участь у програмуванні ФБ. Але якщо хоча б одна геш-функція вкаже на комірку з нульовим значенням, це гарантовано свідчить про відсутність відповідної послідовності символів серед навчальної множини патернів.

Отже, ФБ функціонує з деякою вірогідністю виникання помилки розпізнавання другого роду (false positive), проте без помилок першого роду (false negative). Тобто інколи він може сигналізувати про наявність збігу, якого насправді немає, але ніколи не пропустить справжній збіг. Значення вірогідності помилки другого роду для заданих значень кількості патернів N,

розміру M регістру бітів, та кількості геш-функцій K знаходиться згідно виразу [176]:

$$p_{пом.2} = \left(1 - \left(1 - \frac{1}{M}\right)^{KN}\right)^{K} \approx \left(1 - e^{-KN/M}\right)^{K} .$$

При цьому значення M покладається досить великим, що відповідає дійсності на практиці.

Як можна бачити, збільшення кількості патернів призводить до збільшення системної помилки ФБ, в той час як збільшення розміру РБ – до зменшення помилки. Проте залежність від кількості геш-функцій K не настільки наочна. Доведено [176], що для певних значень N та M вірогідність помилки мінімальна при

$$K = \frac{M}{N} \ln 2 \approx 0{,}6931 \frac{M}{N} .$$

При цьому вірогідність помилки

$$p_{пом.2} = 2^{-K} \approx 0{,}6185^{M/N} .$$

З останнього виразу випливає, що для утримання помилки на певному рівні потрібно забезпечувати певне співвідношення між параметрами M та N.

При використанні наведених співвідношень для вибору параметрів побудови ФБ слід мати на увазі той факт, що оскільки геш-функції обчислюються цифровими схемами у двійковій системі числення, немає сенсу використовувати значення параметру розміру M регістру бітів, яке не дорівнює ступеню двійки.

Аналізуючи наведені вище відомості щодо основ функціонування фільтра Блума можна зробити попередню оцінку його можливостей в якості підходу щодо створення реконфігуровних засобів розпізнавання на базі реконфігуровної логіки.

По-перше, ФБ дозволяє дуже економно використовувати ресурси пам’яті. Розмір словнику патернів не впливає напряму на розмір РБ. Додання нових патернів до вже присутніх у цієї структурі даних, призводить лише до підвищення вірогідності помилки розпізнавання, але не до збільшення об’єму запам’ятовуючого пристрою. По-друге, кількість та складність геш-функцій, які потрібно обчислювати в процесі розпізнавання, тобто, апаратурна витратність та продуктивність при даному підході теж не залежать від об’єму словника патернів. Нарешті, довжина патернів також не впливає напряму ні на кількість ресурсів пам’яті, ані на продуктивність.

Отже можна зробити висновок, що ФБ добре масштабується по двох напрямках: за об’ємом словнику патернів та за довжиною патернів, що відшукуються. Остання властивість має важливе значення: навіть дуже довгі рядки після перетворення геш-функціями потребують для зберігання ті ж самі K комірок РБ.

На жаль, фільтр Блума має дуже важливий недолік. Цьому підходу притаманна особливість усіх рішень, що використовують гешування: розмір вхідної послідовності символів, тобто довжина патерну має бути фіксованою для обраного набору геш-функції. Інакше кажучи, один ФБ здатен розпізнавати патерни тільки однакової довжини.

3.2.2. Кластерізація

Єдиною можливістю подолання згаданого вище недоліку фільтра Блума – здатності розпізнавати тільки патерни однакової довжини – виявляється побудова структури, яка містить кілька одночасно працюючих ФБ, кожен з яких розрахований на розпізнавання патернів однієї з можливих довжин. На рис. 3.15 наведена така схема з чотирьох фільтрів Блума BF_1, …

BF_4, які розпізнають патерни довжиною від 2 до 5 символів відповідно [178]. Вхідна послідовність подається на конвеєр, складений з 8-розрядних регістрів RG_i. До кожного ФБ надходить відповідна кількість 8-розрядних шин.

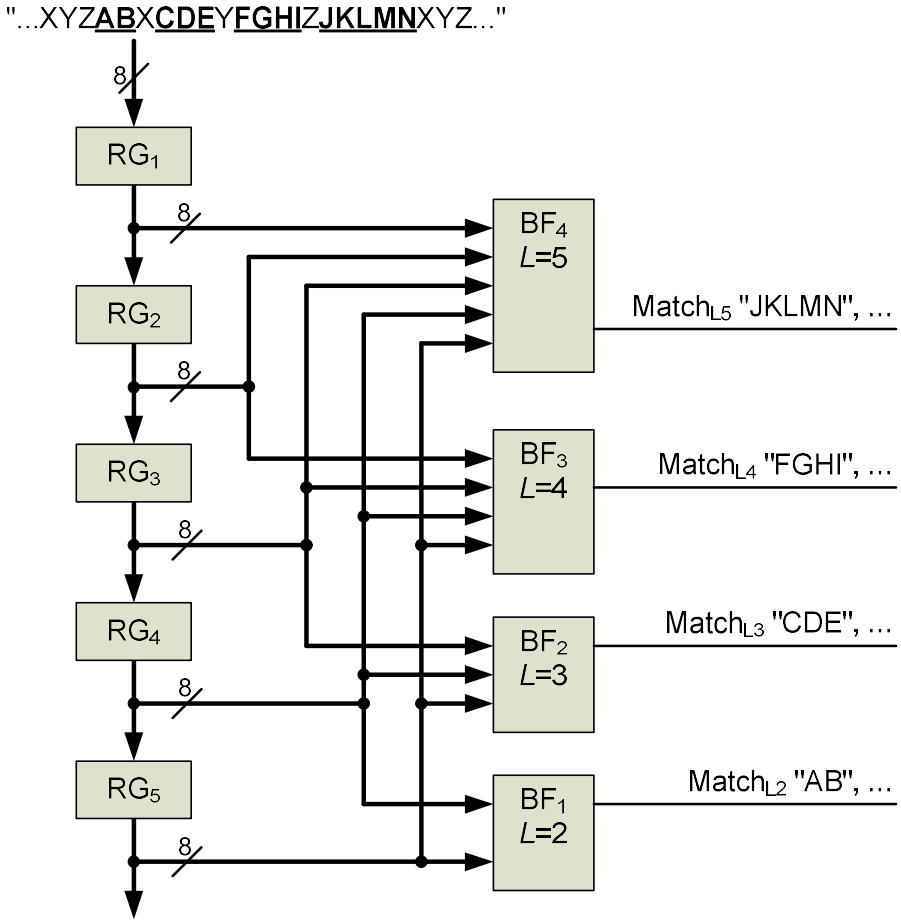


Рис. 3.15. Спільне підключення різних фільтрів Блума

Для реалізації розглянутої структури необхідно провести кластеризацію набору патернів за розміром. У загальному випадку модуль розпізнавання, побудований на засадах фільтра Блума, має стільки окремих фільтрів, скільки різних довжин мають патерни, що входять до складу бази даних сигнатур. Це призводить до великих витрат ресурсів, що значно погіршує вартівні показники ефективності ФБ. Але це погіршення не стосується швидкісних показників, про що свідчить той факт, що певна кількість успішних практичних розробок МСВВ на базі фільтра Блума

фактично є системами запобігання вторгнень, для яких висуваються значно жорсткіші вимоги щодо продуктивності [178], [179], [177], [180].

Розглянемо особливості реалізації фільтра Блума на ПЛІС стосовно генерації геш-функцій та організації роботи з РБ.

3.2.3. Особливості реалізації геш-функцій на ПЛІС

Вибір геш-функцій має важливе значення для ефективного створення фільтра Блума [175]. Ці функції мають, по-перше, однозначно відображати множину комбінацій вхідних символів на множину вихідних значень, тобто результати обчислень геш-функції не повинні повторюватися для різних вхідних слів. По-друге, цифрова схема реалізації функцій $h_1(x)$, $h_2(x)$, $h_3(x)$, …, $h_K(x)$ повинна мати якомога простішу апаратну структуру, щоб забезпечити високу швидкодію при низьких ресурсних витратах.

Слід також зауважити, що діапазон вихідних значень геш-функції повинен відповідати розміру M регістру бітів, оскільки має вказувати на довільну комірку регістру Rg. Але параметр M може приймати доволі різні значення з метою забезпечення оптимальності параметрів фільтра Блума. Отже, вихідний діапазон результатів обчислення геш-функцій також має легко змінюватися.

Крім того, у випадку використання фільтра Блума в складі РАСЗТЗІ, множина геш-функцій має легко реалізовуватися на реконфігуровній платформі.

Сформульованим вимогам задовольняють так звані геш-функції класу H_3 [181], [182], [183], які останніми роками успішно використовуються при створенні фільтрів Блума [178], [177]. Стосовно реалізації на ПЛІС такі геш-функції можливо обчислювати рекурсивно: результати обробки перших і бітів можуть використовуватися для пошуку перших $(i + 1)$ бітів. Ця властивість геш-функції призводить до регулярності цифрової схеми та дозволяє знизити потрібні ресурси [178].

У праці [184] розглянуто ще кілька варіантів геш-функцій, які доцільно використовувати у фільтрах Блума.

Розглянемо більш докладно, як обчислюються згадані вище геш-функції класу H_3.

Згідно визначення функцій класу H_3 вхідний бітовий вектор $a = \overline{a_1 a_2 a_3 ... a_b ... a_B}$, де B – кількість бітів у ньому, керує побітовим складанням по модулю 2 рядків з матриці $D = [d_{b,g}]$ випадкових двійкових значень таким чином, що у складанні приймають учать тільки рядки, для яких у відповідній позиції вектору X присутні одиниці. При цьому індекс g = 1, 2, 3, …, G , де G – розрядність питомих геш-функцій. Нижче розглянутий приклад обчислення геш-функції $h(x)$ для B = 8 та G = 4 [177]:

$$a = 00011010; \quad D = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}; \quad h(x) = h(00011010) =$$

$$= \left(a_1 \wedge \overline{d_{1,1}\, d_{1,2}\, d_{1,3}\, d_{1,4}}\right) \oplus \left(a_2 \wedge \overline{d_{2,1}\, d_{2,2}\, d_{2,3}\, d_{2,4}}\right) \oplus \ldots \oplus \left(a_8 \wedge \overline{d_{8,1}\, d_{8,2}\, d_{8,3}\, d_{8,4}}\right) =$$

$$= \overline{d_{4,1}\, d_{4,2}\, d_{4,3}\, d_{4,4}} \oplus \overline{d_{5,1}\, d_{5,2}\, d_{5,3}\, d_{5,4}} \oplus \overline{d_{7,1}\, d_{7,2}\, d_{7,3}\, d_{7,4}} =$$

$$= 1110 \oplus 1001 \oplus 1101 = 1010 .$$

Якщо потрібно отримати групу з K геш-функцій (l = 1, 2, 3, …, K), обчислення виконуються аналогічно, але для кожного l обираються різні матриці D.

Оскільки в нашому випадку фільтр Блума використовується для розпізнавання послідовностей символів, а не окремих розрядів, на вхід генераторів геш-функції дані подаються згрупованими по 8 бітів (у випадку байтового кодування) відповідно до байтового кодування. Положимо, до $p = \overline{p_1 p_2 p_3 ... p_i ... p_j}$ – це вхідне слово (рядок), що складається з окремих символів $p_i = \overline{a_{i,1} a_{i,2} a_{i,3} ... a_{i,b} ... a_{i,8}}$, які в свою чергу складаються з окремих бітів $a_{i,b}$. При цьому весь рядок p як єдиний бітовий вектор взаємодіє з матрицею D = [$d_{b,g}$], де b = 1, 2, 3, … 8, 9, … 16, 17, … 8j; g = 1, 2, 3, …, G.

Спрощену схему генерування геш-функцій за розглянутим вище алгоритмом можна знайти, наприклад в роботі [182].

3.2.4. Організація роботи з регістром бітів фільтра Блума на ПЛІС

Особливістю базової схеми фільтра Блума (рис. 3.14) є необхідність декільком компонентам одночасно доступатися до регістру бітів Rg. Якщо реалізувати його на єдиному запам’ятовуючому пристрої, виникатимуть колізії. Помірні вимоги до розміру РБ в схемі фільтра Блума дозволяють реалізовувати його на базі внутрішнього ресурсу пам'яті ПЛІС – вбудованої блокової пам'яті BRAM. Бібліотечні компоненти більшості сучасних виробників реконфігуровної логіки надають можливість синтезувати двопортові пристрої пам'яті. Такі пристрої дозволяють одночасний доступ до комірок РБ двох геш-функцій. Але оскільки реальна кількість геш-функцій в практичних додатках перевищує значення два (може сягати десяті та більше), колізії неминучі.

Рішення полягає у розбитті РБ на декілька запам'ятовуючих пристроїв залежно від числа геш-функції та здатності одночасного доступу схеми пам'яті. Для цього потрібно, по-перше, розподілити виходи блоків реалізації геш-функцій між відповідними ОЗП, по-друге, на функціональність цих

блоків накласти додаткові обмеження, щоб їх вихідний діапазон укладався у зменшений об'єм запам'ятовуючого пристрою. В роботі [178] показано, що такі обмеження і підвищують вірогідність помилок другого роду фільтра Блума, але не суттєво.

В роботі [179] розглянуто практичну схему побудови РБ для ФБ на базі блокової пам'яті (Fig. 5 – Fig. 7). Її структура складається з елементарних компонентів на базі BRAM, так званих часткових ФБ (partial Bloom filter), які ієрархічно поєднуються в міні-ФБ (mini-Bloom filter), а потім – у великий ФБ (large Bloom filter).

Більш детально ця схема розглянута у підр. 4.6.2.4 "Функція оцінки схеми повнорозмірного фільтра Блума LBF. Ресурсна складова" при підрахунку ресурсних витрат при побудові фільтра Блума. Але зараз зауважимо, що схему міні-ФБ в деяких випадках можна використовувати окремо в якості складової частини модуля розпізнавання РАСЗТЗІ. Назвемо таку схему *Спрощеним фільтром Блума* або ***SBF*** (Simplified Bloom Filter). Натомість великий ФБ (як більш загальний варіант реалізації роботи з РБ) при його використанні в якості компонента модуля розпізнавання РАСЗТЗІ будемо називати *Повнорозмірним фільтром Блума* або ***LBF*** (Large Bloom Filter).

Відомі також інші техніки вирішення проблеми одночасно доступу до запам'ятовуючого пристрою у ФБ. Наприклад, в роботі [185] запатентовано модифікацію фільтра Блума, яка забезпечує одночасне звернення кількох геш-функцій до масиву бітів, використовуючи однопортову пам'ять. А в роботі [186] використовується попереднє хешування вхідної послідовності, за результатами якого дані комутуються на потрібний ФБ.

3.2.5. Розпаралелювання

Підвищити продуктивність рішення на основі фільтра Блума можливо за рахунок паралельного підключення із зсувом кількох однакових блоків, кожен з яких містить потрібний комплект ФБ. На рис. 3.15 такий блок

(обведений пунктиром) має назву BF_Engine. На рис. 3.16 наведений приклад паралельного використання трьох подібних блоків.

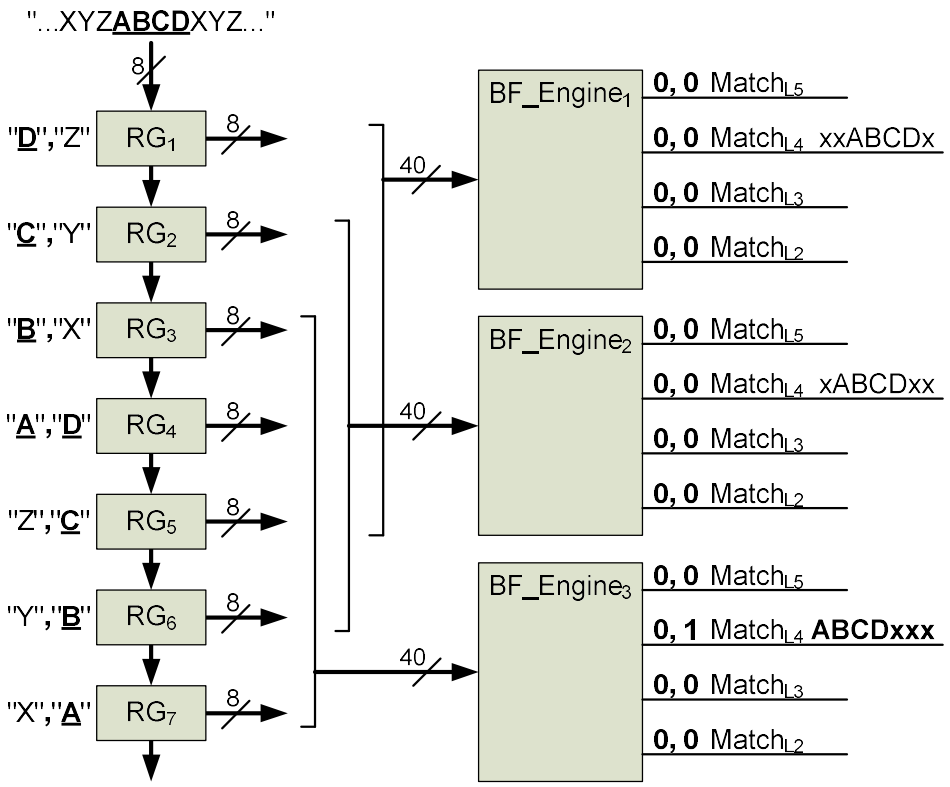


Рис. 3.16. Паралельне підключення кількох блоків розпізнавання на базі фільтрів Блума

Як і в випадку розпаралелювання схем на цифрових компараторах, вхідна послідовність просувається швидше, аніж в базовій схемі (рис. 3.15), в даному прикладі – на три символи вперед на кожному циклі обробки інформації блоками BF_Engine$_i$.

Також як і в випадку з цифровими компараторами можна бачити, що трикратне прискорення призводить до трикратного збільшення апаратних ресурсів. Тобто розпаралелювання базовий схеми на фільтрах Блума також має добру масштабованість з швидкодії.

Але на відміну від цифрових компараторів, властивості схем на фільтрах Блума не дозволяють досягнути сублінейного закону. Тобто

апаратні витрати розпаралеленої схеми строго пропорційні досягнутому прискоренню.

3.2.6. Уточнення результатів

Як було вказано в підрозділі 2.3.2.1, Фільтру Блума притаманні системні помилки розпізнавання другого роду. Тому результати розпізнавання потрібно уточнювати, тобто переконатися, чи насправді виявлена послідовність символів збігається з патерном. Операція уточнення може бути виконана програмно на хост-комп'ютері, апаратними засобами із використанням частки ресурсів програмованої логіки [179] або за допомогою вбудованих в ПЛІС процесорних ядер універсальної архітектурі [177]. У двох останніх випадках цю операцію зазвичай виконують із застосуванням вторинної геш-таблиці, яка зберігається в бортовому ОЗП реконфігуровного обчислювача, зовнішньому по відношенню до кристалу ПЛІС.

Слід зауважити, що операція уточнення виконується значно повільніше, ніж процедура розпізнавання. У випадках коли ознаки зловмисних дій в мереженому трафіку зустрічаються рідко, це не призводить до порушення коректності функціонування модуля розпізнавання. Але якщо зловмисник здійснює атаку на систему виявлення вторгнень шляхом насичення мережевого трафіку пакетами, що імітують напад (тобто співпадають з патернами бази даних сигнатур МСВВ), це може ускладнити роботу системи аж до втрати працездатності.

Отже необхідність уточнення результатів розпізнавання призводить до непередбачуваності пропускної здатності, що робить фільтр Блума вразливим до атак на МСВВ (див. підр. 2.2.3.1 "Спроможність протидіяти атакам на РАСЗТЗІ"). Вирішення проблеми можливо через апаратний захист від переповнення черги запитів на уточнення. У цьому випадку в разі атаки схема уточнення не припиняє функціонування, проте подає на вихід частково неперевірені дані. Як наслідок система виявлення вторгнень в цілому починає робити помилки розпізнавання другого роду. Втім слід пам’ятати,

що за рахунок належного вибору параметрів K та M вірогідність хибного спрацьовування фільтра Блума може бути знижена до несуттєвого значення.

3.2.7. Подальше підвищення швидкодії

Як згадувалося вище, фільтру Блума притаманна висока продуктивність. Багато дослідників намагалися підвищити швидкісні характеристики цього підходу. Але крім розпаралелювання (яке для ФБ на відміну від підходу на цифрових компараторах не дає сублінейного прискорення – див. розділ 2.3.2.5) суттєвих успіхів досягнуто не було.

Багато розробок присвячено скороченню надлишковості та оптимізації часу обчислення геш-функцій [187], [188], [189]. Більшість таких рішень зв'язана з каскадуванням. Суть техніки полягає у наступному. Оскільки в фільтрі Блума для отримання позитивного висновку про розпізнавання патерна з бази сигнатур необхідна наявність одиниць в усіх відповідних комірках РБ, достатньо одного нуля, щоб зробити негативний висновок. Тому, якщо обчислювати геш-функції послідовно, можна позбавитися відпрацьовування більшості з них. Але організувати обчислювальний процес таким чином, щоб послідовна обробка випереджала паралельну, складно. В результаті більшість схем каскадування призводять до позитивного результату в сенсі енергозбереження, але не прискорення.

В роботі [189] вдалося досягнути загального підвищення продуктивності системи МСВВ у середньому, але за рахунок маніпулювань із запитами на обробку вхідних даних, причому на рівні пакетної обробки інформації, не змінюючи власне структуру фільтра Блума.

В роботі [180] запропонована техніка, при якої швидкодія збільшується за рахунок одночасного доступу до двох бітів під час звернення до ОЗП. Таку можливість надає особливий режим роботи деяких запам'ятовуючих пристроїв, зокрема механізм burst-type I/O синхронної пам'яті типу SDRAM. За рахунок зменшення кількості звернень до РБ вдалося досягти певного прискорення. Але це призвело до підвищення вірогідності помилок другого

типу, хоча й незначного. На жаль дана техніка не може бути використана при реалізації фільтра Блума на ПЛІС.

3.2.8. Здатність до динамічної реконфігурації

Класична схема фільтра Блума не дозволяє в процесі функціонування вилучати патерні, додані в схему під час його програмування. Якщо ця властивість потрібна, застосовують такий різновид підходу, як *фільтр Блума з лічильниками*, або *схема* ***CBF*** (Counting Bloom Filter). Вперше подібна модифікація була запропонована в роботі [190]. В схему додається *M* лічильників. В процесі програмування фільтра вихідні сигнали з геш-функцій подаються на входи відповідних лічильників у режимі додавання. В разі потреби вилучити патерн активні значення виходів геш-функцій керують лічильниками в режимі віднімання. Перехід лічильника зі стану "0" в стан "1" призводить до запису "1" у відповідну комірку РБ фільтра Блума, перехід лічильника зі стану "1" в стан "0" – до запису "0". В іншому принцип функціонування схеми не змінюється.

Отже техніка використання лічильників надає можливість динамічної реконфігурації фільтра Блума в реальному часі без перепрограмування ПЛІС та тимчасового припинення функціонування РАСЗТЗІ.

Недоліком техніки фільтра Блума з лічильниками є додаткові витрати ресурсів ПЛІС. Причому точне значення розрядності лічильників складно визначити, тому що кількість "влучень" різних геш-функцій в одну й ту саму комірку РБ є випадковою величиною. Якщо використати лічильники недостатньої розрядності, їх переповнення призводитиме до виникнення помилок першого роду (false negative), тобто, схема буде розпізнавання не всі патерни, що були запрограмовані.

В роботах [190] та [175] показано, що 4-розрядних лічильників достатньо для більшості практичних застосувань, наведено оцінку вірогідності помилки першого роду.

Зважаючи на те, що потреба в вилучанні патернів зі словнику виникає

досить рідко, можливий варіант реалізації фільтра Блума з лічильниками програмними засобами [178]. Такий прийом не потребує додаткових апаратних витрат порівняно з базовою схемою, але в цьому випадку необхідно задіяти спеціальний режим конфігурації, при якому відновлюється тільки зміст блокової пам'яті BRAM, а інші частини ПЛІС залишаються незмінними.

Отже техніка використання лічильників дозволяє динамічно реконфігурувати фільтр Блума в реальному часі без перепрограмування ПЛІС та тимчасового припинення функціонування сигнатурної системи захисту інформації незалежно від типу його реалізації.

3.2.9. Висновки щодо підходу на основі фільтра Блума

Результати формалізації опису властивостей схем розпізнавання на базі АП та ЦК зведені в табл. 3.2.

Таблиця 3.2

Формалізований опис властивостей підходу на базі ФБ

№	Показник		Якісна оцінка властивості
1.	Витрати логіки		Помірна перевага
2.	Витрати пам'яті	розподіленої	Помірна перевага
3.		блочної	Помірна перевага
4.		зовнішньої	Суттєва перевага
5.	Швидкодія		Помірна перевага
6.	Передбачуваність пропускної здатності		Суттєвий недолік
7.	Функціональні показники	здатність протидіяти атакам на РАСЗТЗІ	Суттєвий недолік
8.		динамічна реконфігурація	Помірна перевага
9.		селективне розпізнавання	Помірна перевага
10.		режим запобігання вторгнень (для МСВВ)	Помірний недолік
11.	Масштабованість	за пропускною здатністю	Помірний недолік
12.		за об’ємом словнику патернів	Суттєва перевага
13.		за довжиною патернів	Суттєва перевага
14.	Використання надлишковості		Суттєвий недолік
15.	Суттєвий недолік, який зводить нанівець головні переваги підходу		Фіксована довжина патернів

Головні переваги фільтра Блума при побудові реконфігуровних засобів інформаційної безпеки – це *економне споживання ресурсів* та відносно *висока пропускна здатність*.

ФБ добре масштабується стосовно швидкодії, а особливо – об’єму словнику та довжині патернів.

Його регулярна та логічна структура обумовлює добру здатність до реалізації на ПЛІС. Єдина складність полягає в одночасному зверненні множини геш-функцій до багаторозрядного масиву бітів. Але ця проблема вирішується шляхом розбиття запам’ятовуючого пристрою на декілька окремих модулів та незначної корекції обчислювачів геш-функцій.

Також позитивною рисою підходу є можливість реалізації режиму динамічної реконфігурації без припинення функціонування системи розпізнавання. Схема ФБ з лічильниками за рахунок невеликих додаткових витрат дозволяє здійснити таку функцію.

Головний недолік підходу на базі фільтра Блума полягає у низької гнучкості, тому що він здатен розпізнавати тільки патерни однакової довжини. Тобто в загальному випадку для розпізнавання множини патернів потрібно стільки окремих ФБ, скільки різних довжин ці патерни мають, що призводить до суттєвих ресурсних витрат, тобто зводить нанівець одну з головних переваг підходу.

Більш відомим, але менш важким за наслідками недоліком ФБ є наявність системних помилок розпізнавання другого роду, що призводить до необхідності виконання додаткового обчислювального етапу уточнення результатів роботі схеми. Сама по собі ця властивість не є критичною. Але, оскільки додаткова операція виконується значно повільніше аніж процедура розпізнавання, фільтр Блума стає вразливим до специфічних атак на МСВВ, якщо не вжити запобіжних заходів.

3.3. Використання алгоритму Ахо–Корасік на базі скінченних автоматів

Для опису поведінки та створення складних обчислювальних систем вже багато років успішно використовують математичний апарат цифрових (або дискретних) автоматів [191], [192], [193].

Абстрактний або цифровий автомат (ЦА), відповідний англомовний термін – state machine (SM), у загальному випадку задається шісткою елементів:

$$A = \{X, Y, S, S_0, f_s, f_y\}, \tag{3.1}$$

де X – множина вхідних сигналів;

Y – множина вихідних сигналів;

S – множина станів автомата;

S_0 – початковий стан автомата;

f_s – функція переходів з одного стану в інший;

f_y – функція виходів автомата.

Класичний ЦА в початковий момент часу перебуває у стані S_0; на кожному такті в залежності від вхідних сигналів переходить з одного стану в інший згідно функції переходів f_s; при цьому його вихідні сигнали формуються згідно функції виходів f_y. Якщо функція виходів залежить тільки від поточного стану, ЦА називають автоматом Мура, якщо – від поточного стану та вхідних сигналів – автоматом Мілі. Звернемо увагу, що в обох випадках вихід змінюється при кожному переході. Автомат такого класу, називають *перетворювачем* або *трансдуктором* (transducer) [194]. Подібні ЦА були давно відомі та активно використалися при створенні різноманітних цифрових пристроїв та систем в роки становлення та розвитку цифрової обчислювальної техніки.

Але останнім часом теоретичні дослідження та практичні розробки частіше використовують інші класи ЦА. Так автомат – *акцептор* (acceptor, recognizer), також відомий як *виявлювач послідовностей* (sequence detector) видає на виході активний сигнал тільки в разі, якщо на його вхід надійшла певна послідовність певних *символів* тобто комбінацій вхідних сигналів. На практиці функція акцептора полягає в виявленні (розпізнаванні) певного слова (послідовності символів) у вхідному потоці даних. В термінах теорії обчислень на строках [61] акцептор виявляє входження підрядка в рядок. Технічно акцептори реалізуються таким чином, що в результаті надходження на вхід автомата питомої послідовності символів він переходить у так званий *прийнятний* стан. Інакше кажучи, сигнал на виході автомата відсутній поки здійснюються переходи між неприйнятними станами та з'являється тільки в разі досягнення прийнятного стану.

Коли потрібно виявляти у вхідному потоці даних замість одиничного підрядку певну множину зразкових слів (*патернів*), тобто необхідно виконувати задачу множинного розпізнавання рядків (див. підр. 1.2 "Задача множинного розпізнавання патернів"), множина станів автомата містить декілька прийнятних станів. В разі досягнення будь-якого з них автомат сигналізує про розпізнавання відповідного патерну. Такий автомат називають *розпізнавачем* (classifier). Саме такі автомати використовуються при створенні РАСЗТЗІ, включаючи МСВВ на базі ПЛІС.

Автомат – розпізнавач є різновидом класичного цифрового автомату (3.1). Відмінність полягає у відсутності множини вихідних сигналів Y та функції виходів f_y, замість яких з'являється множина прийнятних станів F:

$$A = \{X, S, S_0, f_s, F\}. \tag{3.2}$$

Отже автомат для множинного розпізнавання рядків описується не шісткою, а п’ятіркою відповідних елементів.

Якщо у виразах (3.1) та (3.2) елементи X, Y, S та F є скінченними множинами, відповідний автомат називають *скінченним автоматом* (СА). Формально це стосується будь-якого дискретного автомата. Але за спостереженням автора в публікаціях останніх років, особливо в англомовних, у переважній більшості випадків словосполука *скінченний автомат* (Finite State Machine – FSM або Finite Automaton – FA) замість виразу *цифровий автомат* (state machine – SM) використовується саме для посилання на автомати, що розпізнають рядки, тобто на автомати – розпізнавачі. Тому в даному дослідженні для позначення автоматів, на яких будуються РАСЗТЗІ, також вживатиметься цей термін – "скінченний автомат".

Оскільки в автоматах – розпізнавачах на відміну від класичних ЦА виду (3.1) замість вхідних сигналів вживаються символи, тобто групи сигналів (кількістю у вісім одиниць, якщо символ кодується одним байтом, або іншого розміру, кратного восьми, якщо кодування багатобайтове), то множина X набуває сенсу *вхідного алфавіту* автомата (множини символів, які можуть бути подані на вхід).

За функціонуванням СА дещо відрізняється від цифрового автомата (3.1). В початковий момент часу СА перебуває у стані S_0; на кожному такті отримує вхідний символ (з алфавіту X) і в залежності від його значення переходить з одного стану в інший згідно функції переходів f_s; досягнувши одного з прийнятних станів, відає сигнал про розпізнавання відповідного слова (підрядка), після чого продовжує роботу з поточного стану.

Скінченні автомати бувають *детермінованими* (ДСА) та *недетермінованими* (НСА). Відповідні англомовні терміни – *deterministic finite automaton* (DFA) та *nondeterministic finite automaton* (NFA). У детермінованому автоматі на кожному такті можливий лише один перехід лише до одного стану. Як наслідок в кожний окремий момент часу ДСА може перебувати тільки в одному стані. Для НСА вказані обмеження не виконуються. Тобто НСА може переходити за один такт з одного стану

одразу до кількох станів та находитися у кількох станах одночасно. Теоретично довільний НСА може бути трансформований в еквівалентний за функціональністю ДСА (який проте має значно більшу кількість станів) та навпаки. На практиці ДСА та НСА в наслідок особливостей технічної реалізації мають різні сфери застосування.

3.3.1. Алгоритм Ахо–Корасік і його реалізація у вигляді кінцевого автомату

Алгоритм Ахо–Корасік (АК) [63] є зразком класичного засобу множинного розпізнавання рядків на відміну від алгоритмів одиночного розпізнавання (див. підр. 1.2.2 "Алгоритми одиночного розпізнавання" та підр. 1.2.4 "Алгоритми множинного розпізнавання"). Його суть полягає у тому, що на початковому етапі з наданого набору патернів за певними правилами створюється ДСА – розпізнавач, який потім, на етапі функціонування, розпізнає відповідні патерни. В подальшому такий ДСА буде зватися скінченним автоматом Ахо–Корасік (СА-АК).

Алгоритм має гарні кількісні показники: число станів автомату СА-АК має складність O(n), де n – загальна кількість символів у словнику патернів, а час обробки вхідного потоку даних має складність O(m), де m – кількість символів у вхідних даних, тобто автомат витрачає один такт на обробку одного вхідного символу [55]. Мабуть саме тому алгоритм набув значного поширення серед розробників сигнатурних систем інформаційного захисту, зокрема, РАСЗТЗІ. Не зважаючи на те, що він був в числі перших алгоритмів, запропонованих для множинного розпізнавання, він досі залишається одним з найзатребуваніших інструментів сигнатурного аналізу. В англомовній літературі навіть сформувався усталений термін – абревіатура AC-DFA (Aho-Corasick deterministic finite automaton).

Існує багато джерел, в яких докладно описані як сам автомат СА-АК, так і особливості поводження з ним, включаючи реалізацію на ПЛІС [55], [97], [195], [196], [197], [56], [140], [198], [98], [199]. Але, як буде пояснено

нижче, певні особливості стосовно даного алгоритму є суттєвими при порівнянні показників ефективності різних підходів до побудови схем апаратного розпізнавання. Тому розглянемо алгоритм Ахо–Корасік, звертаючи увагу на такі особливості.

Рис. 3.17 ілюструє процес створення ДСА згідно алгоритму АК на прикладі розпізнавання трьох патернів: "SHIP", "HIS" та "IN" [55].

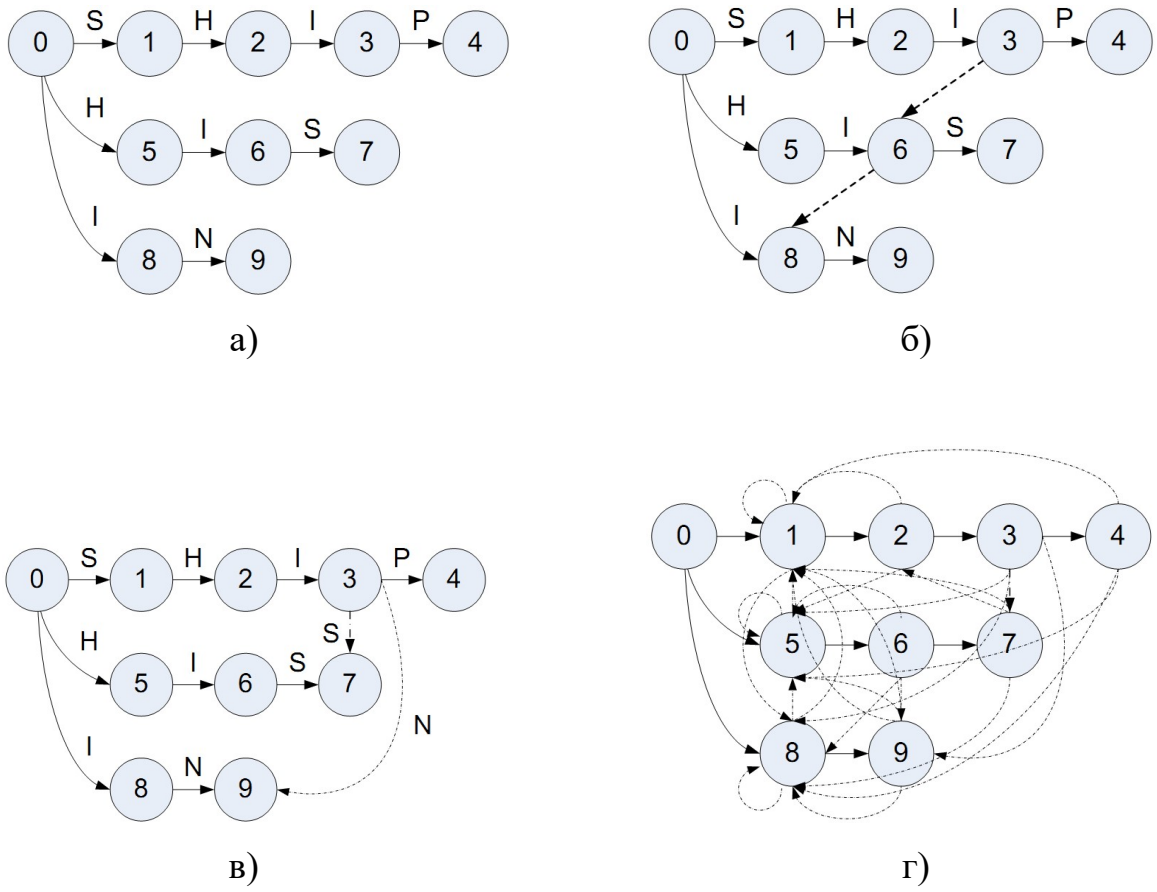


Рис. 3.17. Формування ДСА згідно алгоритму АК
для множини патернів {SHIP, HIS, IN}

Процедура створення автомата згідно алгоритму АК складається з двох кроків.

На першому кроці (рис. 3.17, а) будується ациклічний орієнтовний граф у вигляді дерева, коренем якого є початковий стан S_0. Кожен з патернів, що підлягають розпізнаванню, додає до графу ланцюжок вершин, що

відповідають станам автомату, таким чином, що кожний символ відзначає дугу до наступної вершини – стану, починаючи з кореня аж до кінця патерну.

Фактично, вже зображений на рис. 3.17, а автомат здатен розпізнавати відповідні патерни, якщо його функція переходів f_s визначена наступним чином: в разі надходження на вхід автомата символу, який відповідає дузі, він переходить до стану, на який ця дуга вказує, а у випадку надходження будь-якого іншого символу, здійснює перехід до початкового стану (ці зворотні переходи не відображені на рисунку, щоб занадто не ускладнювати його). Стани S_4, S_7 та S_9 цього скінченого автомату є прийнятними станами; досягнення будь-якого з них свідчить про наявність у вхідній послідовності підрядка "SHIP", "HIS" або "IN" відповідно.

Але, як було вказано в підр. 1.2.3 "Ефект самоподоби. Префікси, інфікси, суфікси", базам даних сигнатур МСВВ та антивірусів притаманна властивість самоподоби, суть якої полягає у тому, що багато патернів мають фрагменти (префікси, інфікси або суфікси), що збігаються. У наведеному прикладі, зокрема, інфікс "HI" патерну "SHIP" співпадає з префіксом "HI" патерну "HIS".

З іншого боку, якщо не враховувати ефект самоподоби, автомат згідно рис. 3.17, а не буде здатним функціонувати коректно. Дійсно, якщо на вхід автомату надійде послідовність символів "SHIS", цій автомат зі стану S_3 перейде у початковий стан S_0, тобто не розпізнає патерн "HIS", який міститься у вхідних даних.

Щоб автомат коректно опрацьовував подібність патернів "SHIP" та "HIS", потрібно його доробити, а саме – змінити функцію переходів таким чином, щоб перебуваючи у стані S_3 при надходженні символу, відмінного від "P", автомат не робив перехід у початковий стан, а опинявся у стані S_6 (див. рис. 3.17, в). У загальному випадку нове правило переходів формулюється наступним чином: якщо розпізнавання поточного патерну переривається, треба перевірити, чи не є вже розпізнаний фрагмент префіксом іншого патерну. Якщо таких патернів виявиться більш одного, обирається той, з

префіксом якого збігається найбільша кількість вже розпізнаних символів. В розглянутому прикладі в стані S_3 автомат частково розпізнав не тільки послідовність "HI", яка збігається с префіксом патерну "HIS", але й символ "I", що є префіксом патерну "IN". Але фрагмент "HI" довше фрагменту "I", тому перехід слід робити у стан S_6. Але стрибок зі стану S_3 у стан S_6 повинен здійснюватися одночасно з переходом зі стану S_6 у наступний стан в залежності від вхідного символу, який поступає за розпізнаним тактом раніше символом "I", тобто автомат повинен здійснювати два переходи у відповідь на прийом одного вхідного символу. У детермінованому автоматі це неприпустимо. Тому СА-АК об'єднує перехід на ланцюжок подібного патерну з тактом розпізнавання наступного символу. В результаті перехід зі стану S_3 замість стану S_6 здійснюється у стан S_7, але за умови подання на вхід автомату символу "S" (див. рис. 3.17, в). За таким принципом можна використати також частково розпізнаний фрагмент "I" в разі надходження символу "N", тобто додати в функцію переходів дугу зі стану S3 в стан S9. Саме пошук таких переходів виконується на другому кроці побудови автомату СА-АК.

Насправді властивість самоподоби для обраних трьох патернів не обмежується спільними фрагментами "HI" та "I". Наприклад, якщо у вхідній послідовності за префіксом якогось патерну слідує знов такій же префікс, здійснюється перехід в початок ланцюжку даного патерну. На рис. 3.17, г наведені всі можливі переходи, які повністю реалізують всі ефекти самоподоби набору з трьох патернів "SHIP", "HIS" та "IN". При цьому задля більшої наочності поряд з дугами опущені відповідні літери, а також опущені хибні переходи у початковий стан. Як можна бачити, навіть для досить простого прикладу загальна кількість переходів у повністю побудованому СА-АК доволі велика.

3.3.2. Класифікація типів переходів автомата СА-АК

Наразі проаналізуємо різновиди переходів в автоматі Ахо–Корасік та проведемо їх класифікацію.

У сформованому автоматі СА-АК можна розрізнити чотири типи переходів:

1. *Прямі* (direct, basic або goto) *переходи* – такі, що закладаються в граф на першому кроці побудови автомату (рис. 3.17, а).

2. *Перехресні* (cross) *переходи*, які відпрацьовують явище самоподоби патернів (рис. 3.17, в).

3. *Хибні* (failure) *переходи* – переходи у початковий стан.

4. *Післястартові* (restartable [97]) *переходи* – перехресні переходи, що здійснюються на стани, які безпосередньо слідують за початковим станом (на рис. 3.17, в це переходи з будь-якого стану в стани S_1, S_5 та S_8).

Наявність четвертого типу переходів обумовлена тим фактом, що остання літера (односимвольний суфікс) майже кожного з патернів збігається з першою літерою (односимвольним префіксом) якогось іншого патерну. Відокремлення переходів даного типу від інших перехресних переходів, як буде пояснено пізніше, обумовлено особливостями технічної реалізації автоматів СА-АК.

З метою підвищення зрозумілості викладеного матеріалу іноді будемо називати переходи другої групи *основними перехресними*, щоб краще відрізняти їх від перехресних післястартових переходів четвертої групи.

3.3.3. Базова схема скінченного автомата Ахо–Корасік

Узагальнена структура апаратної реалізації СА-АК подана на рис. 3.18.

Його основу складає запам'ятовуючий пристрій (ЗП), в якому функція f_s зберігається у вигляді так званої таблиці переходив скінченого автомата [195], [196]. Кожна комірка такої таблиці містить номер наступного стану та вектор збігу. До значення номеру наступного стану, що витягується з пам'яті, шляхом конкатенації додається код символу із вхідній послідовності.

Отримане значення подається на адресний вхід запам'ятовуючого пристрою та обирає відповідний рядок інформації. Для рядків з прийнятними станами вектор збігу (ВЗ) містить одиницю у позиції, що позначає відповідний патерн, яка після пересилання змісту ВЗ до вихідного регістру ініціює відповідний сигнал збігу. Керує роботою автомата керуючий пристрій (КП).

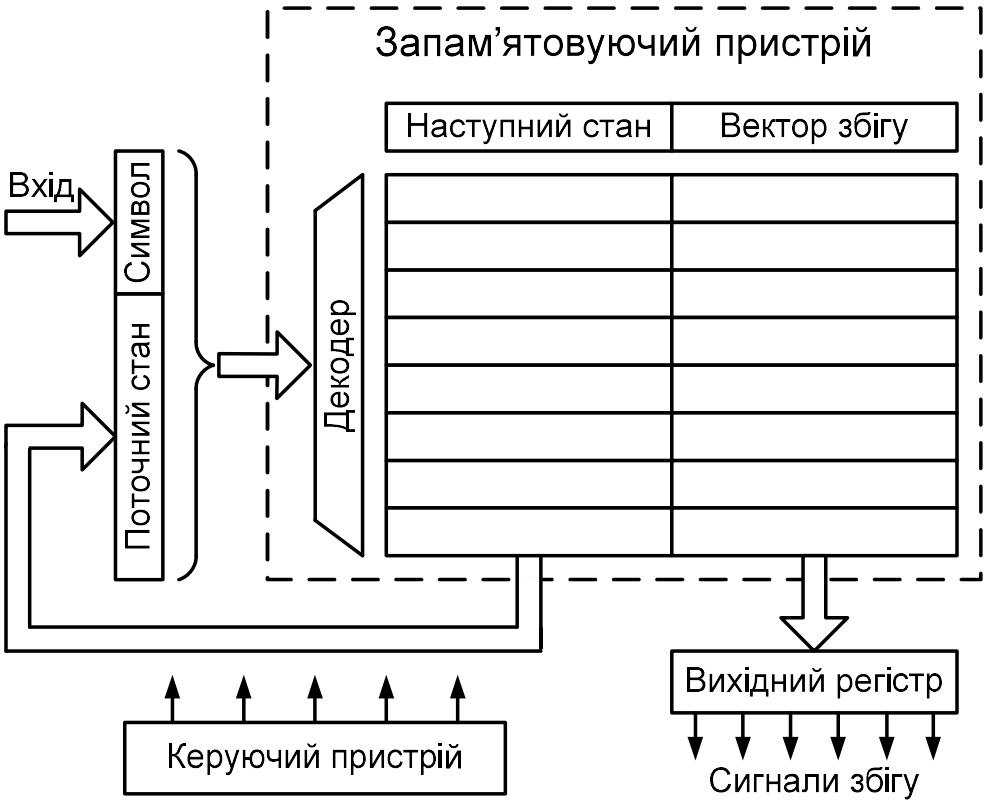


Рис. 3.18. Узагальнена структурна схема типової реалізації скінченого автомата Ахо–Корасік

Якщо при реалізації даної схеми використовується зовнішня стосовно кристалу ПЛІС пам'ять, називатимемо таке рішення *Базовою схемою СА-АК із зовнішньою пам'яттю* або *схемою* ***ACRAM***.

Рішення на базі внутрішній блокової пам'яті ПЛІС будемо називати *Базовою схемою СА-АК із блоковою пам'яттю* або *схемою* ***ACBRAM***.

3.3.4. Основні властивості базових схем СА-АК

Найважливіша перевага рішень на базі скінченного автомату Ахо-Корасік – *передбачуваність пропускної здатності* а також незалежність пропускної здатності від об’єму словнику та його складу, зокрема від довжин патернів. Теоретично, скінченний автомат за кожен такт приймає один символ зі вхідної послідовності. Але на практиці якщо розмір таблиці переходив завеликий, і для її зберігання потрібно використовувати зовнішню відносно ПЛІС пам'ять, за певних обставин кожне звернення до ЗП буде займати декілька тактів. Крім того, зовнішня пам'ять сама по собі повільніша за внутрішню. Отже, зворотна сторона цієї властивості – відносно низька швидкодія.

Стосовно споживання ресурсів, як можна бачити, СА-АК потребує незначну кількість ресурсів логіки, обмежуючись, фактично, лише схемою керування та контролером ЗП. Проте об’єм запам'ятовуючого пристрою може сягати дуже великих значень. Оскільки в загальному випадку кількість станів дорівнює загальній кількості символів у словнику патернів, а кількість переходів з кожного стану в інший в граничному випадку може сягати 256 (при байтовому кодуванні), то безпосередня реалізація схеми згідно рис. 3.18 споживатиме наступну кількість байтів пам’яті:

$$M = n \cdot m \cdot 256 \cdot (1 + n / 8), \tag{3.3}$$

де n – кількість патернів у базі даних сигнатур, а m – їх середня довжина. При розмірах словнику в десятки та сотні тисяч патернів та їх середній довжині в десятки символів розмір ЗП сягатиме порядків сотень гігабайтів – десятків терабайтів.

Отже, підхід характеризується високою ресурсоємністтю, а саме – *значними витратами на пам'ять*, що вважається його головним недоліком. Тому переважна більшість досліджень щодо застосування СА-АК та

відповідних технічних рішень спрямована саме на зменшення об’єму вживаної пам’яті.

Слід зауважити, що квадратична залежність (3.3), яка призводить до лавиноподібного зросту необхідних ресурсів при збільшенні кількості патернів, означає *погану масштабованість за об’ємом словнику патернів*.

Стабільна, проте відносно *низька пропускна здатність* СА-АК також вважається її недоліком.

Але більш суттєвим недоліком є складність підвищення швидкодії базової схеми автомата. Тобто підхід також має *погану масштабованість за пропускною здатністю*.

З іншого боку, алгоритм Ахо–Корасік має *добру масштабованість за довжиною патернів*.

3.3.5. Особливості реалізації схеми СА-АК на ПЛІС

Як було вказано вище, значну частину скінченного автомата становить запам’ятовуючий пристрій, тому основні складнощі, що виникають при побудові на програмованій логіці СА-АК для розпізнавання рядків, полягають в створенні цього пристрою та організації ефективного обміну даними з ним. В разі використання реконфігуровних обчислювачів проблема загострюється в зв’язку з тим, що бортова пам'ять цих пристроїв має фіксовану структуру, не завжди зручну для побудови ЗП для скінченного автомата. В роботі [197] детально розглянуті питання організації роботи з пам’яттю при побудові засобів розпізнавання на основі алгоритму Ахо–Корасік для задач захисту інформації. Ретельно вивчена функціональність, яка потрібна для ефективного доступу до даних, враховуючі можливість повторного використання (кешування) інформації. Досліджені особливості взаємодії з пристроями динамічної пам’яті різних поколінь. Доведена можливість практичного досягнення високих значень пропускної здатності.

3.3.6. Модифікації базової схеми СА-АК

Як свідчить аналіз світового досвіду, підхід з використанням алгоритму Ахо-Корасік на базі цифрових автоматів набув найбільшу кількість модифікацій, варіантів, доробок та вдосконалень порівняно з конкуруючими напрямами. Нижче розглянути найбільш значні та ефективні з них, що суттєво покращили кількісні характеристики і виявилися прототипами для подальшої модернізації. Більшість модифікацій апаратною структурою принципово не відрізняється від базової схеми, а відмінності полягають у принципах створення таблиці переходів. Тому не для кожної з розглянутих модифікацій було надано окрему назву. З іншого боку, на кількісні показники ефективності реалізацій СА-АК більш значуще впливає тип використаної пам'яті – зовнішня чи блокова. Тому майже кожне рішення матиме пару назв – для кожного з типів ЗП.

3.3.7. Зменшення ресурсних витрат

Як слідство головного недоліку підходу на основі скінченних автоматів, більшість досліджень щодо апаратного застосування алгоритму Ахо–Корасік, присвячена проблемі зменшення вживання ресурсів пам’яті.

Перш за все багато розробників пропонують різноманітні техніки зменшення об'єму ЗП за рахунок кодування таблиці переходів. Цій шлях нібито дозволяє досягнути мети, але за рахунок виконання додаткових операцій, тобто не відрізняється ефективністю. Більш результативними виявляються інші напрями досліджень.

В роботах [195] та [196] запропонована модифікація алгоритму АК таким чином, щоб використовувати збіг не тільки суфіксів одних патернів с префіксами інших, а також подібних інфіксів (середніх частин) різних патернів. Експериментальна перевірка підтвердила реалізованість даної ідеї, показав можливість скорочення пам’яті на 21%.

Але більш суттєво зменшувати потрібні ресурси дозволяє врахування та поводження з типами переходів СА-АК, перелічених в підр. 3.3.2 "Класифікація типів переходів автомата СА-АК".

Відомо, що значну частку переходів, а отже, і пам'яті, займають хибні переходи у початковий стан, а також післястартові переходи. Спеціальна техніка на основі використання пріоритетів дозволила автору дослідження [56] скоротити кількість переходів цих двох класів до 256 одиниць для довільного за розміром словника патернів. (Ось чому було доцільно виокремити післястартові переходи з загального числа перехресних переходів). Оскільки скоротити прями переходи неможливо за визначенням (які до того ж складають найменшу частку), остання численна група переходів, яка залишається неоптимізованою – основні перехресні переходи. Згідно дослідженню [97] в автоматі СА-АК, побудованому по базі даних сигнатур відомої відкритої системи МСВВ Snort, частка таких переходів на момент проведення дослідження становила 79,2%; для антивірусній системи ClamAV ця доля була ще більша – 95,9%. Автори цієї роботи модифікували схему СА-АК, створивши так званий кешований детермінований скінченний автомат Cached Deterministic Finite Automaton –CDFA. Вони стверджують, що розробка дозволяє видалити до 90% переходів в апаратних додатках інформаційного захисту мережевих об’єктів. В результаті авторам роботи вдалося скоротити ресурси пам’яті до 81 Kb для бази сигнатур Snort об’ємом в 1800 записів, та до 29 Kb для бази ClamAV розміром в 50000 записів. (станом на 8 жовтня 2006 р.).

3.3.8. Конвеєризація

В роботі [199] була вперше запропонована, а потім розвинута в дослідженні [55] та інших публікаціях техніка застосування конвеєризації при побудові СА-АК. Традиційно в обчислювальній техніці конвеєр використовується для підвищення пропускної здатності (за рахунок збільшення параметру латентності, який в багатьох використаннях не є

критичним). Але у випадку алгоритму Ахо–Корасік цей метод обробки інформації дозволив суттєво скоротити потрібні ресурси пам'яті. Автори роботи [55] формально довели, що лінійний конвеєр з *H* щаблів дозволяє позбавитися всіх основних перехресних переходів в структурі СА-АК від початкового стану до рівня *H*.

На рис. 3.19 наведено схематичне зображення конвеєрної схеми СА-АК (з використанням блокової пам'яті), запропонованої в [55].

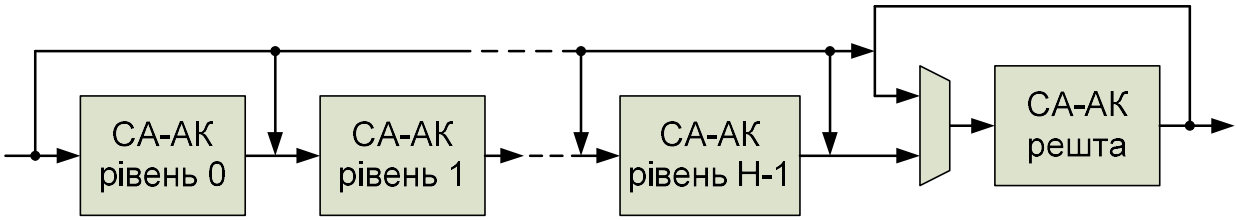


Рис. 3.19. Схема конвеєрного розпізнавання на базі СА-АК

Оскільки значне скорочення об'єму ЗП, що використовує конвеєрна схема, дозволяє в багатьох застосуваннях обходиться виключно внутрішньою пам'яттю одного кристалу ПЛІС, назвемо таке рішення *схемою **PipACBRAM***.

Зауважимо, що в цієї публікації також запропонований варіант прискорення даної схеми шляхом розпаралелювання (див. підр. 3.3.9 "Розпаралелювання"). Особливістю рішення є застосування блоків двопортової пам'яті BRAM одночасно в двох паралельних каналах конвеєру. Таку модифікацію природно назвати схемою ***ParPipACBRAM***.

3.3.9. Розпаралелювання

Окрім надмірного споживання ресурсів пам'яті СА-АК характеризується не дуже гарними швидкісними показниками. Як будь-який скінченний автомат він обробляє вхідну інформацію принципово послідовно – символ за символом. У багатьох дослідженнях (наприклад, в [200]) робляться спроби прискорити роботу алгоритму АК шляхом обробки більш, ніж одного байту за такт. При цьому схема розглядає декілька байтів разом

як один символ алфавіту *X*. В іншому принцип роботі СА-АК не змінюється. Оскільки заздалегідь невідомо, з яким зсувом опиниться питомий патерн у вхідному потоці даних, потрібно організувати паралельну роботу відповідної кількості автоматів, які розглядають скомбіновані патерни с різними зсувами.

На рис. 3.20 наведено схему, яка обробляє по три символи за такт.

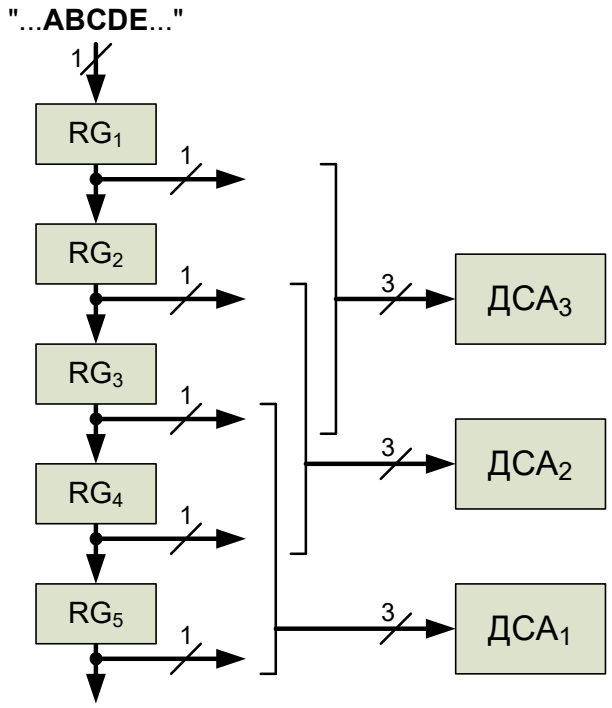


Рис. 3.20. Паралельна схема багатосимвольного розпізнавання

Тут ширина шин позначена в кількості символів.

В залежності від типу пам'яті називатиме це рішення або *схемою* ***ParACRAM*** або *схемою* ***ParACBRAM***.

Вхідна інформація пересувається у конвеєрі, що складається з регістрів RG_i. Кожен з $ДСА_j$ отримує на вході по три символи за такт. Очевидно, що така схема не дуже ефективна. По-перше, кожен з автоматів потребує значно більше ресурсів пам'яті, оскільки його алфавіт складається не з 256, а з 2^{24} символів. По-друге, таких автоматів потрібно три. Тобто масштабованість за пропускною здатністю виявляється дуже поганою. Авторам роботи [200] вдалося за рахунок запропонованих методів суттєво покращити ситуацію та

досягнути трикратного прискорення для модифікованої схеми, але загальний результат виявився не дуже оптимістичним.

3.3.10. Небайтова розрядність обробки даних

Оригінальне рішення щодо скорочення ресурсних витрат СА-АК було запропоновано в роботі [140], яке було назване авторами алгоритмом *Bit-split*. Його суть полягає в заміні одного автомата, що обробляє 8-бітні символи на декілька однакових паралельно працюючих підавтоматів (tiny automata), які аналізують або по 1, або по 2 або по 4 біти. За рахунок зменшення "символів", що обробляються, суттєво скорочується розмір алфавіту. Наприклад для 4-бітної обробки кількість допустимих символів в алфавіті скорочується до 16 (замість 256), а для 2-бітної – до 4. Відповідно скоротилася кількість можливих переходів з кожного стану. Ця техніка виявилася вдалою, в неї з’явилися послідовники, які допрацювали та розвили її. В роботі [140] було доведено, що оптимальним розбиттям є ділення по 2 біти. В дослідженні [198] за допомогою попередньої обробки фільтром Блума вдалося позбавитися зайвих переходів в підавтоматах.

На жаль рішення Bit-split не покращує масштабованість підходу за об’ємом словнику патернів. Тому для розпізнавання великої кількості патернів потрібне розбиття їх множини на підгрупи. В роботі [98] досліджена оптимальна кластеризація бази даних сигнатур між Bit-split підавтоматами.

3.3.11. Здатність до динамічної реконфігурації

Використання зовнішнього запам'ятовуючого пристрою призводить до зниження швидкодії СА, але надає можливість шляхом перезапису його змісту змінювати склад переходів автомату, тобто повністю змінювати алгоритм його роботи. При цьому не потрібно зупиняти процес функціонування РАСЗТЗІ [141]. За рахунок цього підхід на базі СА-АК має таку перевагу перед іншими, як здатність до динамічної реконфігурації. Як було зазначено в підр. 3.2.8 "Здатність до динамічної реконфігурації", фільтр Блума для реалізації цієї функції потребує ускладнення його структури

шляхом заміни комірок пам’яті на лічильники або задіяти спеціальний режим конфігурації, а асоціативна пам'ять на цифрових компараторах взагалі не дає змоги здійснювати динамічну реконфігурацію (див. підр. 3.1.10 "Здатність до динамічної реконфігурації").

Єдиним обмеженням при здійсненні динамічної реконфігурації СА-АК є вимога до незмінності розрядності ЗП, тобто, щоб об’єм ресурсів, потрібних для алгоритму, не перевищував розмір запам'ятовуючого пристрою. З метою забезпечення цієї вимоги є доцільним при побудові цифрового автомата передбачувати певний запас по об’єму пам'яті.

Використання в схемі СА-АК замість зовнішньої пам'яті вбудованих в ПЛІС блоків BRAM дещо ускладнює процедуру динамічної реконфігурації. В цьому випадку потрібно, як і при використанні ФБ, задіяти спеціальний режим конфігурації, при якому відновлюється тільки зміст блокової пам'яті BRAM, а інші частини ПЛІС залишаються незмінними.

Близьким за своєю природою до показника динамічної реконфігурації є інший функціональний показник – селективного розпізнавання. Спроможність обирати підмножину патернів, що розпізнаватимуться, під керуванням зовнішнього сигналу досить просто реалізується шляхом формування в пам'яті ЗП кількох таблиць переходів, які відповідають різним автоматам Ахо–Корасік, створеним для відповідних підмножин патернів. Дія зовнішнього сигналу курування в такому випадку зводиться до додавання певного зміщення до значення адреси на вході ЗП.

3.3.12. Висновки щодо підходу на основі алгоритму Ахо–Корасік

Результати формалізації опису властивостей схем розпізнавання на базі СА, що реалізують алгоритм АК, зведені в табл. 3.3.

Головні переваги підходу на основі алгоритму Ахо–Корасік – це *передбачуваність пропускної здатності*, *добра масштабованість за довжиною патернів* та *незначне споживання логічних ресурсів* ПЛІС, а також

здатність до селективного розпізнавання та динамічної реконфігурації словнику патернів без припинення процесу розпізнавання.

Таблиця 3.3

Формалізований опис властивостей підходу на базі СА-АК

№	Показник		Якісна оцінка властивості
1.	Витрати логіки		Суттєва перевага
2.	Витрати пам'яті	розподіленої	Суттєва перевага
3.		блочної	Суттєвий недолік
4.		зовнішньої	Суттєвий недолік
5.	Швидкодія		Помірний недолік
6.	Передбачуваність пропускної здатності		Суттєва перевага
7.	Функціональні показники	здатність протидіяти атакам на РАСЗТЗІ	Помірна перевага
8.		динамічна реконфігурація	Суттєва перевага
9.		селективне розпізнавання	Суттєва перевага
10.		режим запобігання вторгнень (для МСВВ)	Суттєвий недолік
11.	Масштабованість	за пропускною здатністю	Помірний недолік
12.		за об’ємом словнику патернів	Суттєвий недолік
13.		за довжиною патернів	Суттєва перевага
14.	Використання надлишковості		Помірна перевага
15.	Суттєвий недолік, який зводить нанівець головні переваги підходу		Вибухоподібний зріст об’єму пам'яті

Головні вади – *надмірне споживання ресурсів пам'яті* (як зовнішньої, так і внутрішньої блокової), *погана масштабованість за об’ємом словнику патернів* та відносно погана – *за пропускною здатністю*. Але численні модифікації та вдосконаленні базової схеми СА-АК дозволили дещо пом'якшити ці проблеми.

Недоліком також є відносно *невисока пропускна здатність* порівняно з асоціативною пам'яттю та фільтром Блума.

3.4. Порівняльний аналіз підходів до побудови МР РАСЗТЗІ

Проведені в попередніх підрозділах аналіз та формалізація на якісному

рівні трьох підходів, що пов’язані з використанням асоціативної пам'яті на базі цифрових компараторів, фільтра Блума на базі хеш-функцій та алгоритму Ахо–Корасік на базі скінченних автоматів, а також численних технік їх вдосконалення дозволили ґрунтовно оцінити переваги та недоліки кожного з них в сенсі сформульованих раніше показників ефективності. В табл. 3.4 наведені результати цього порівняльного аналізу.

Таблиця 3.4

Результати порівняння основних підходів до побудови РАСЗТЗІ

№	Показник		Підхід		
			Асоціативна пам'ять	**Фільтр Блума**	**Скінчен. автомат Ахо–Корасік**
1.	Витрати логіки		---	+	+++
2.	Витрати пам'яті	розподіленої	---	+	+++
3.		блочної	+++	+	---
4.		зовнішньої	+++	+++	---
5.	Швидкодія		+++	+	-
6.	Передбачуваність пропускної здатності		+++	---	+++
7.	Функціональні показники	здатність протидіяти атакам на РАСЗТЗІ	+++	---	+
8.		динамічна реконфігурація	---	+	+++
9.		селективне розпізнавання	---	+	+++
10.		режим запобігання вторгнень (для МСВВ)	+++	-	---
11.	Масшта-бованість	за пропускною здатністю	+	-	-
12.		за об’ємом словнику	-	+++	---
13.		за довжиною патернів	-	+++	+++
14.	Використання надлишковості		+++	---	+
15.	Суттєвий недолік, який зводить нанівець головні переваги підходу		Завелике споживання ресурсів	Фіксована довжина патернів	Вибухоподібний зріст об’єму пам'яті

Позначення: "+++" – суттєва перевага; "+" – помірна перевага; "---" – суттєвий недолік; "-" – помірний недолік;

Як бачимо, жоден з досліджених підходів (або жодна технологія, на яких вони ґрунтуються) не демонструє явних переваг перед іншими, кожен має власні позитивні риси та недоліки. Причому часто густо перевага по одному з показників обертається недоліком стосовно іншого. Як наслідок, кожен з підходів має суттєвий недолік, який, фактично, зводить нанівець головні переваги підходу.

Так ЦК та побудовані на їх основі різновиди АП забезпечують максимальну швидкодію, але витратніше інших рішень за споживанням апаратних ресурсів та електроенергії, вони також програють в плані масштабування. Фільтр Блума більш економічний та краще масштабується, але накладає обмеження на довжину патернів; він також вимагає додаткових витрат на доуточнення здобутих результатів через системну наявність помилок розпізнавання другого роду. Скінченні автомати більш економні щодо витрат на логіку, забезпечують стабільну, але відносно невисоку пропускну здатність, складні в побудові, призводять до "вибухового" зростання об’єму пам'яті для великих словників сигнатур.

Відсутність лідируючого напрямку, який би перевершував конкурентні рішення за всіма показниками, призводить до того, що розробники, по-перше, пропонують чисельні модифікації основних підходів, намагаючись позбутися їх недоліків, по-друге – комбінують підходи в різноманітних поєднаннях. Але ці спроби носять евристичний, несистемний характер. Відчувається брак формалізації та узагальнення, що не дозволяє перевести проблему з інженерного на науковий рівень.

Тому виникає потреба в методах, які дозволили б системно поєднати різні підходи в єдиному пристрої максимізуючи його ефективність за рахунок здійснення переваг кожного з них.

Але до того потрібно розробити засоби кількісної оцінки та порівняння різнорідних технічних рішень в єдиному метричному просторі та скласти відповідні розрахунки для кожного з напрямів. Даному питанню присвячено наступний розділ.

3.5. Висновки до розділу 3

1. Формалізовано на якісному рівні властивості наступних підходів до побудови схем розпізнавання, придатних для створення ефективних МР РАСЗТЗІ, та схемотехнічних технологій, на яких вони базуються:
   - асоціативна пам'ять на базі цифрових компараторів;
   - фільтр Блума на базі геш-функцій;
   - алгоритм Ахо–Корасік на базі скінченних автоматів.
2. Виявлено теоретичний потенціал кожного підходу та практичні розробки, в яких він був реалізований на численних прикладах модифікацій базових схем та використань різноманітних технік і прийомів щодо підвищення ефективності.
3. Для кожного з напрямів з'ясовано: переваги та недоліки в сенсі ефективності, можливості подальшого покращення показників, складнощі реалізації на ПЛІС і шляхи їх подолання.
4. Проведено порівняльний аналіз досліджених підходів. В результаті з'ясовано, що жоден з них не демонструє переваг перед іншими за більшістю показників.

РОЗДІЛ 4. КІЛЬКІСНА ОЦІНКА ПІДХОДІВ ДО ПОБУДОВИ МР РАСЗТЗІ

У попередніх підрозділах виявлені та формалізовані *на якісному рівні* найбільш ефективні підходи до побудови МР РАСЗТЗІ. Але для створення методів, які дозволили б системно об'єднати різні підходи в одному пристрої, максимізуючи його ефективність за рахунок здійснення переваг кожного з них, потрібно розробити інструмент *кількісної* оцінки та порівняння різнорідних технічних рішень в єдиному метричному просторі.

З розглянутих в підр. 2.2 "Показники ефективності РАСЗТЗІ" показників ефективності є три основні класи показників, придатних до кількісного вимірювання: 1) показники ресурсних витрат на створення технічного об’єкту, що розглядається; 2) часові характеристики об’єкту; 3) показники, які мають сенс виконаної роботи або об’єму обробленої інформації. Всі інші показники є або похідними від цих трьох, або принципово не можуть бути оцінені кількісно. Проте, показники, що мають сенс виконаної роботи, відображають не стільки властивості, притаманні створюваному пристрою обробки інформації, а скоріш зовнішні фактори або вхідні змінні, які можна задати однаковими для питомих об’єктів. Тому для кількісної оцінки та порівняння різних технічних рішень вистачає перших двох показників, тобто *ресурсів*, що мають бути витраченими для їх створення, та *часу*, який вони витрачають на обробку певних даних.

Тому запропонований в даному розділі метод прискореного обчислення технічних параметрів компонентів РАСЗТЗІ використовує так звані функції оцінки, які складаються саме з ресурсної та часової складових. При розрахунку ресурсної складової даним методом в якості універсальної умовної одиниці вимірювання використовується логічна таблиця (LUT), яка є найдрібнішим внутрішнім компонентом більшості сучасних ПЛІС. Всі інші ресурси зводяться до цієї одиниці за допомогою відповідних коефіцієнтів.

Згідно запропонованого метода в розділі також складені потрібні розрахунки та сформовані функції оцінки для базових схем кожного з основних підходів до побудови МР РАСЗТЗІ, а також для їх найбільш перспективних модифікацій.

Перед тим, як безпосередньо перейти до розрахунків технічних параметрів компонентів РАСЗТЗІ, розглянуті та вирішені додаткові потрібні питання. Введені визначення, розроблені допоміжні техніки поводження з патернами, що мають оброблятися створюваними технічними засобами.

4.1. Набір патернів

Оскільки ключову роль при реалізації методів побудови оптимізованих МР відіграють патерни, що підлягають розпізнаванню, розглянемо визначення та техніки, які дозволять формалізувати процеси поводження з ними.

Положимо $P = \{p_1, p_2, p_3, \dots p_k, \dots p_\sigma / \sigma, \Omega, m_{\min}, m_{\max}, \delta, \mu, \mu_z, \nu\}$ – це множина патернів, яка характеризується власне набором патернів $p_1, p_2, p_3, \dots p_k, \dots p_\sigma$, а також: потужністю множини σ, загальною кількістю символів Ω, довжиною найкоротшого $m_{\min}$ та найдовшого $m_{\max}$ патернів, функцією розподілу довжин δ, першою μ, першою частковою μ_z та другою ν функціями самоподоби.

Кожний патерн p_k є фіксованою послідовністю символів, код кожного з котрих належить до певного алфавіту Σ. Для випадку однобайтового кодування $\Sigma = \{00_{16}, 01_{16}, 02_{16}, \dots, FF_{16}\}$.

4.2. Техніка впорядкування патернів

В якості підґрунтя подальших розрахунків, пов’язаних з множиною патернів, пропонується наступна техніка впорядкування патернів у наборі.

Впорядкуємо всі патерни в множині P за зростанням їх довжини, як подано на рис. 4.1. Складені з квадратів стовпчики тут зображають складені з

символів рядки.

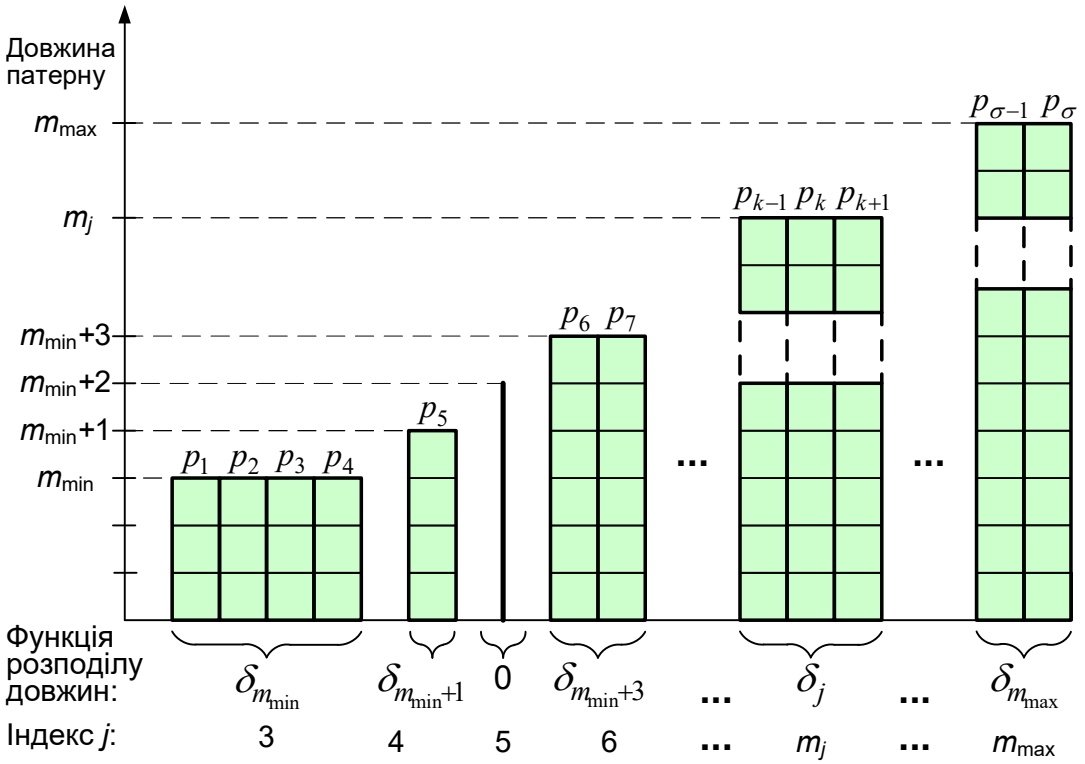


Рис. 4.1. Техніка впорядкування пакетами патернів однакової довжини

Назвемо *пакетом* сукупність патернів однакової довжини, не звертаючи уваги, як само впорядковані патерни всередині пакету. Введемо індекс j таким, що співпадає з довжиною патернів в пакеті $j = m_{min}$, $m_{min}+1$, $m_{min}+2$, … m_j, … m_{max}, де m_{min} – довжина найкоротшого патерну; m_{max} – довжина найдовшого патерну, причому кожне наступне значення цього індексу обов'язково на одиницю більше за попереднє, тобто в нумерації немає пропусків. Тоді довжина кожного патерну в наборі співпадатиме з індексом його пакету: $m_j = j$. Зворотне твердження хибне, тому що для деяких індексів j патерни відповідної довжини можуть бути відсутні. Тому загальна кількість пакетів ξ в наборі патернів P в загальному випадку менша за величину різниці між довжинами крайніх розмірів ($m_{max} - m_{min}$).

Функцію розподілу довжин δ як залежність від індексу j визначимо рівною кількості патернів у відповідному пакеті: $\delta(j) = \delta_j$. В прикладі на рис. 4.1 $j = 3, 4, 5, \ldots m_{max}$; $\delta(3) = 4$, $\delta(4) = 1$, $\delta(5) = 0$, $\delta(6) = 2$, $\delta(m_{max}) = 2$.

За допомогою функції розподілу довжин, зокрема, зручно обчислити кількість символів Ω в наборі патернів та кількість ненульових пакетів ξ.

Загальна кількість символів дорівнює сумі символів у кожному пакеті, яка в свою чергу дорівнює добутку довжини рядків на їх кількість в пакеті:

$$\Omega = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \, m_j = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \, j \, . \tag{4.1}$$

Для підрахунку кількості пакетів ξ в множині P потрібно визначити функцію нерівності нулю:

$$\mathrm{NotZ}(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \end{cases} . \tag{4.2}$$

Тоді кількість пакетів в множині *P:*

$$\xi = \sum_{j=m_{\min}}^{m_{\max}} \mathrm{NotZ}(\delta_j). \tag{4.3}$$

Для подальших розрахунків потрібно також визначити функції самоподоби множини патернів *P*.

4.3. Функції самоподоби набору патернів

Нагадаємо, що функції самоподоби μ, μ_z та ν – це кількісні параметри певного набору патернів, які характеризують ступінь подібності патернів між собою. В практичному сенсі ці функції визначають надлишковість, що присутня в множині патернів, яку можна використати для скорочення числа операцій порівняння під час вирішення ЗМРП, відповідно – для зменшення ресурсних витрат при створенні модулю розпізнавання РАСЗТЗІ.

Визначимо *першу функцію самоподоби* $\mu(s, j)$ *множини патернів P* як таку, що дорівнює сумарної кількості символів з кодом s ($s \in \Sigma$), розташованих на j-ої позиції всіх патернів цієї множини, причому нумерація позиції j символу в патерні здійснюється з кінця патерна до початку.

Наприклад, для P = {"SHIP", "HIS", "HER", "IN"} μ("S", 4) = 1; μ("H", 3) = 3; μ("I", 2) = 3; μ("P", 1) = 1; μ("S", 1) = 1; μ("N", 4) = 0; μ("N", 3) = 0; μ("N", 2) = 0, де зображення символу в лапках позначає його код у відповідному кодуванні.

Записати в аналітичному вигляді вираз для находження першої функції самоподоби μ складно, але не викликає труднощів підрахувати всі її значення алгоритмічним чином в програмному забезпеченні реалізації методів побудови оптимізованої структури МР.

Перша часткова функція самоподоби $\mu_z(s, j)$ *множини патернів P* дорівнює сумарної кількості символів з кодом s ($s \in \Sigma$), розташованих періодично на позиціях $(j - 1)\,k\,z$, $j = 2, 3, 4, \ldots (z - 1)$, $k = 1, 2, 3, \ldots \left| m_j / z \right|$ всіх патернів цієї множини, причому нумерація позиції j символу в патерні також здійснюється з кінця патерна до початку.

Наприклад, для $z = 4$ і P = {"43214321", "HGFA4EDA4CBA", "555515555", "277727} μ_z("4", 4) = 4; μ_z("3", 3) = 2; μ_z("2", 2) = 4; μ_z("1", 1) = 3; μ_z("A", 1) = 3; μ_z("A", 3) = 0, де зображення символу в лапках також позначає його код у відповідному кодуванні.

Друга функція самоподоби ν *множини патернів P* у кількісний спосіб визначає ступень збігу фрагментів одних патернів з іншим. Для функції ν також складно сформулювати аналітичний запис, але в неявному вигляді вона присутня й відіграє важливу роль при розрахунках кількісних характеристик схем на базі алгоритму Ахо–Корасік.

4.4. Основні кількісні технічні характеристики МР

Як було обґрунтовано на початку розділу, для кількісної оцінки та

порівняння різних технічних рішень достатньо використовувати лише ресурсні та часові показники. Наведемо формалізовані визначення технічних характеристик (або параметрів) створюваних пристроїв, які відповідають цім двом показникам ефективності.

4.4.1. Ресурсні параметри

Якщо цифрова схема, характеристики якої потрібно обчислити, створюється на базі реконфігуровного обчислювача, для її синтезу в загальному випадку можуть використовуватися ресурси різного типу. Щоб забезпечити можливість в методичному сенсі строго порівнювати різні технічні рішення, потрібно звести підрахунок задіяних ресурсів до якоїсь єдиної умовної одиниці. В якості такої одиниці доцільно використати деякій мінімальний структурний елемент внутрішньої структури ПЛІС, наприклад, логічну таблицю (LUT). Тобто здійснювати підрахунок в умовних логічних таблицях (УЛТ).

Використовуючи такій підхід, кількісне значення ресурсів, що потрібні для синтезу деякого компоненту обчислювальної структури, може бути подано у вигляді:

$$R = L + \alpha F + \beta B + \gamma M \ , \tag{4.4}$$

де L – кількість ресурсів логіки ПЛІС (кількість ЛТ),

F – кількість ресурсів розподіленої пам’яті ПЛІС (кількість тригерів),

B – кількість ресурсів блокової пам’яті ПЛІС (число блоків BRAM),

M – кількість ресурсів зовнішньої пам’яті – об’єм бортової пам’яті РУО (у Мбайтах),

α, β, γ – коефіцієнти нормалізації відносно ЛТ (умовних логічних таблиць) ресурсів різного типу, відповідно: кількості тригерів, числа блоків BRAM та об’єму бортової пам’яті РУО.

Звісно, що перелік ресурсів, що використовується при створенні реконфігуровних пристроїв, в загальному випадку може бути ширше. (Наприклад, нещодавно компанія Xilinx анонсувала нові мікросхеми Virtex UltraScale+, які містять компоненти вбудованої так званої HBM-пам'яті об'ємом від 4 до 16 Гбайт з пропускною здатністю до 460 Гбайт/с [201]. Але перелік ресурсів, задіяних у виразі (4.4) буде прийнятним для більшості ПЛІС, що зараз використовуються в галузі захисту інформації. З іншого боку, додати в разі потреби в цю формулу нові доданки та скорегувати розрахунки, що засновані на ній, не викликає труднощів. Тобто фіксований склад виразу (4.4) не обмежує сфери використання запропонованої техніки.

4.4.2. Часові параметри

Затримки в цифрових схемах, незалежно від того, на якій платформі вони побудовані, завжди вимірюються в одних і тих самих одиницях. Тому з використанням часових технічних характеристик при створенні компонентів РАСЗТЗІ складнощів не виникає.

4.5. Метод прискореного обчислення технічних параметрів РАСЗТЗІ

На практиці ресурсні та часові характеристики реконфігуровних компонентів можна знайти шляхом синтезу їх цифрових схем за допомогою інструментальних засобів створення конфігурацій для ПЛІС. Але цей процес потребує забагато часу (див. підр. 1.5.4 "Складності при використанні РУО та шляхи їх усунення"), що унеможливлює його використання в циклі процедури оптимізації при використанні методів створення ефективних РАСЗТЗІ.

Тому було запропоновано метод прискореного обчислення технічних характеристик цифрових пристроїв, призначених для реалізації на програмованій логіці без виконання повної процедури їх синтезу. Суть методу полягає у створенні для кожної цифрової схеми розпізнавання так

званої функції оцінки (ФО) θ. Така функція, маючи на вході заданий набір патернів та інші потрібні параметри, здатна з певною точністю обчислювати та видавати на виході в якості результату значення об'єму ресурсів *R*, що вживатиме питома схема, яка буде розпізнавати цей набір патернів, та чисельне значення часової затримки *T*, яку вона матиме після синтезу:

$$\theta = \{R, T\} = \{\theta_R(P, \Theta, \Psi), \theta_T(P, \Theta, \Psi)\}, \tag{4.5}$$

де *P* – набір патернів, що потрібно розпізнавати,

Θ – сукупність характеристик реконфігуровного прискорювача,

Ψ – сукупність вимог користувача системи захисту в залежності від обраного підходу до побудови МР РАСЗТЗІ.

Функції розрахунку ресурсів θ_R та часу θ_T, які називатимемо відповідно *ресурсна складова* та *часова складова*, залежать в загальному випадку від набору патернів, характеристик РУО, та специфічних вимог користувача, яки є зазвичай параметрами задіяного підходу до побудови схеми розпізнавання.

Набір патернів, як вказувалося в підр. 4.1 "Набір патернів", в свою чергу має такі параметри, як потужність σ, кількість символів Ω, довжина найкоротшого патерну $m_{\min}$, довжина та найдовшого патерну $m_{\max}$, функція розподілу довжин δ, перша функція самоподоби μ та друга функція самоподоби ν.

Властивості РУО, як буде розглянуто нижче, задаються множиною його характеристик, з яких більшість відноситься до ПЛІС, що використана в ньому:

$$\Theta = \{x, y, z, p, U, B_{\text{FPGA}}, L_{\text{FPGA}}, F_{\text{FPGA}}, T_{\text{FPGA}}, M_{\text{RA}}, \alpha, \beta, \gamma\}, \tag{4.6}$$

де *x* – число входів ЛТ мікросхеми ПЛІС, що задіяна в РУО,

y – здатність навантаження (fan-out$_{max}$) логічних елементів ПЛІС,

z – максимальне значення затримки ЦСЗ, створюваної на бази ЛТ,

p – кількість портів блокової пам’яті ПЛІС,

U – розмір блоків пам’яті BRAM в мікросхемі ПЛІС,

B_{FPGA} – кількість блоків блокової пам’яті BRAM в ПЛІС,

L_{FPGA} – кількість ЛТ в ПЛІС,

F_{FPGA} – кількість тригерів в ПЛІС,

T_{FPGA} – максимально можлива швидкодія ПЛІС,

M_{RA} – об’єм бортової пам’яті РУО,

α, β, γ – згадані вище коефіцієнти зведення ресурсів різного типу до УЛТ.

Після обчислення ФО для всіх компонентів МР, можна знайти об'єм ресурсів та часову затримку для всього пристрою.

Нижче розглянуті приклади побудови ФО для базових схем основних підходів та їх найбільш перспективних модифікацій, розглянутих в попередньому розділі.

4.6. Реалізація функцій оцінки для деяких схем розпізнавання

Нижче детально розглянуто декілька прикладів реалізації ФО для найбільш поширених підходів до побудови МР РАСЗТЗІ. Отримані описи ФО дозволять, з одного боку, експериментально перевірити теоретичні положення щодо запропонованих методів комбінування, з іншого – вони можуть бути використані розробниками РАСЗТЗІ при практичному використанні результатів дослідження.

В зв’язку з наявністю великої кількості модифікацій базових підходів до побудови РАСЗТЗІ та технік підвищення їх ефективності видається доцільним обмежитися розгляданням лише декількох прикладів створення ФО для найбільш поширених варіантів побудови блоків розпізнавання. Згідно проведеному в третьому розділі аналізу до таких варіантів відносяться:

- базова схема на АП BsCAM;
- модифікована схема на АП DCAM;
- модифікована схема на АП DpCAM;
- повнорозмірний фільтр Блума LBF;
- спрощений фільтр Блума SBF;
- базова схема СА-АК із зовнішньою пам'яттю ACRAM;
- базова схема СА-АК із блоковою пам'яттю ACBRAM;
- схема СА-АК за конвеєрною схемою PipACBRAM;
- двокаскадна схема HRCmp.

У зв’язку з різною природою підходів до побудови схем розпізнавання принципи складання функцій оцінки для них розрізняються. Тому для кожного з них пропонується окрема техніка створення.

4.6.1. Техніка створення функцій оцінки для асоціативної пам'яті

Для блоків розпізнавання, що будуються на базі АП, в зв’язку с прозорістю та регулярністю структур на цифрових компараторах ФО може бути заснована на прямому підрахунку потрібних ресурсів та часових затримок. В деяких випадках результати такого підрахунку можуть бути уточнені внаслідок виявлення та використання певних технічних особливостей реалізації задіяних підходів та технік. Отже пропонується *Техніка створення ФО для схем АП на базі ЦК*, яка в загальному випадку складається з етапу прямого підрахунку та декількох етапів уточнення отриманих співвідношень.

Оскільки схеми розпізнавання на АП не потребують блокової або зовнішньої пам'яті, вираз (4.4) для знаходження ресурсів схеми, побудованої за таким підходом спрощується до:

$$R^{*}_{\text{CAM}} = L_{\text{CAM}} + \alpha F_{\text{CAM}}, \tag{4.7}$$

де L_{CAM} і F_{CAM} – відповідно кількість ЛТ і тригерів в схемі.

В зв'язку с подальшими уточненнями значення споживаних схемою ресурсів, що передбачаються, розглядатиме величину R^*_{CAM} як таку, що отримана в *першому теоретичному наближенні*.

4.6.1.1. Функція оцінки базової схеми BsCAM. Ресурсна складова

Використовуючи формулу (4.7), знайдемо кількість обчислювальних ресурсів, що потребує базова схема розпізнавання на цифрових компараторах BsCAM.

Логічні таблиці LUT в схемі BsCAM використовуються, по-перше, для синтезу компараторів CMP (див. рис. 3.1), по-друге – для створення багатовходової схеми "I", що будується у вигляді конвеєра (див. рис. 3.3):

$$L_{\text{BSCAM}} = L_{\text{CMP}} + L_{\&}. \tag{4.8}$$

Особливістю схеми BsCAM є той факт, що вона потребує стільки компараторів, скільки символів сумарно міститься у всіх патернах, що мають розпізнаватися, тобто Ω.

Один цифровий компаратор CMP, що розпізнає символ у байтовому кодуванні, при побудові схеми АП для РАСЗТЗІ потребує дві ЛТ на 4 чи на 6 входів, або одну 8-входову ЛТ. Введемо функцію-кваліфікатор

$$\Lambda(x) = \begin{cases} 1, & x \geq 8 \\ 2, & x < 8 \end{cases}. \tag{4.9}$$

Тоді кількість ЛТ, що потрібна для створення всіх компараторів схеми BsCAM, дорівнюватиме

$$L_{\text{CMP}} = \Lambda(x)\Omega, \tag{4.10}$$

де x – число входів ЛТ для заданої ПЛІС.

Нескладно перевірити, що кількість ЛТ, потрібних для об'єднання j входів каскадною схемою "І" (рис. 3.3), з урахуванням того факту, що на x-входої ЛТ можна синтезувати логічну схему "І" не більш, чим на x входів, дорівнює для кожного патерну

$$L_j = L(j) = \left\lceil \frac{j-1}{x-1} \right\rceil. \tag{4.11}$$

Кількість ЛТ для складання всіх конвеєрів всіх патернів підраховується аналогічно (4.1):

$$L_{\&} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil. \tag{4.12}$$

Підставляючи (4.10) та (4.12) в (4.8) та враховуючи (4.1), отримуємо загальну кількість ЛТ для схеми BsCAM:

$$L_{\text{BSCAM}} = \Lambda(x)\Omega + \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil = \Lambda(x) \cdot \sum_{j=m_{\min}}^{m_{\max}} \delta_j \, j + \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil =$$

$$= \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\Lambda(x) j + \left\lceil \frac{j-1}{x-1} \right\rceil \right). \tag{4.13}$$

Кількість тригерів, потрібних для створення схеми BsCAM, складається з їх кількості FRG, потрібної для побудови вхідного конвеєра (рис. 3.1), кількості тригерів F_{fan} в конвеєрі розгалуження для підвищення здатності навантаження (fan-out$_{\max}$) виходів регістрів вхідного конвеєра (див. рис. 3.2) та кількості тригерів $F_{\&}$ в конвеєрі для об'єднання по "І" виходів всіх компараторів для всіх патернів (рис. 3.3):

$$F_{\mathrm{BSCAM}} = F_{\mathrm{RG}} + F_{\mathrm{fan}} + F_{\&} . \tag{4.14}$$

Довжина вхідного конвеєру, по якому просувається потік символів, що розпізнаються, дорівнює найдовшому патерну з підгрупи $m_{\max}$, а ширина – одному байту (у випадку байтового кодування), тобто восьми бітам. Тому для його побудови потрібно тригерів:

$$F_{\mathrm{RG}} = 8\, m_{\max} . \tag{4.15}$$

Підрахунок числа F_{fan} виявляється дещо складнішим порівняно з попередніми розрахунками. Навантаження на вихідні ланцюги регістрів вхідного конвеєру нерівномірне. Якщо з його перших $m_{\min}$ щаблів сигнали використовуються для розпізнавання всіх патернів підгрупи, то з останнього – тільки для розпізнавання пакету з найдовших патернів, як можна бачити на рис. 4.2.

Тут складені з квадратів стовпчики (крім вхідного конвеєру) зображають складені з компараторів схеми розпізнавання патернів $p_1, p_2, p_3, \ldots p_k, \ldots p_\sigma$, що входять до множини P.

Сигнали з виходів кожного з перших $m_{\min}$ каскадів конвеєра подаються на всі σ патернів підгрупи, тобто мають навантаження:

$$O_1 = \sum_{i=m_{\min}}^{m_{\max}} \delta_i = \sigma .$$

Вихід кожного j-го з решти каскадів конвеєра, починаючи з $i = m_{\min} + 1$, подається на патерни, що входять у пакети з номерами від j по $m_{\max}$, тобто повинен розгалужуватися на кількість входів

$$O_j = \sum_{i=j}^{m_{\max}} \delta_i \, .$$

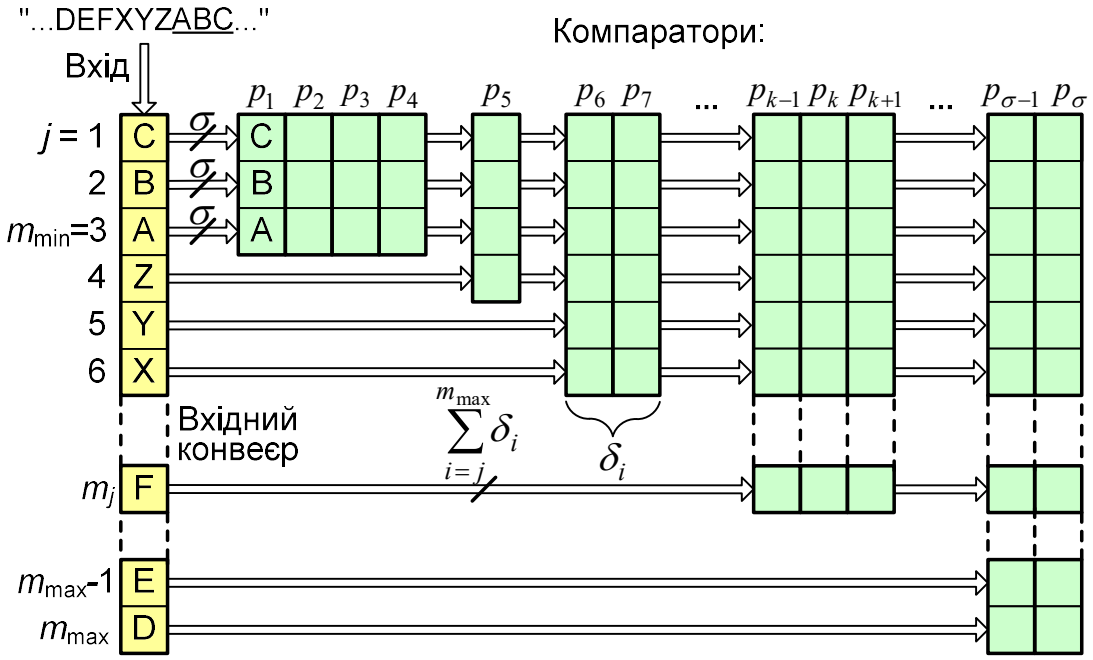


Рис. 4.2. Розподіл навантаження на вхідний регістр у схемі BsCAM

Кількість тригерів, потрібних для розгалуження одного виходу на O входів конвеєром згідно рис. 3.2 може бути обчислена за формулою аналогічною (4.11):

$$F(O) = \left\lceil \frac{O-1}{y-1} \right\rceil, \tag{4.16}$$

де y – здатність навантаження виходів тригерів для заданої ПЛІС.

Тоді загальна кількість тригерів, потрібних для розвантаження всіх виходів вхідного конвеєру – як перших $m_{\min}$, так і решти, з урахуванням (4.11):

$$F_{\text{fan}} = m_{\min} \left\lceil \frac{O_1 - 1}{y-1} \right\rceil + \sum_{j=m_{\min}+1}^{m_{\max}} \left\lceil \frac{O_j - 1}{y-1} \right\rceil =$$

$$= m_{\min}\left\lceil \frac{\sigma-1}{y-1} \right\rceil + \sum_{j=m_{\min}+1}^{m_{\max}} \left\lceil \frac{\sum_{i=j}^{m_{\max}} \delta_i - 1}{y-1} \right\rceil. \qquad (4.17)$$

Кількість тригерів в конвеєрі для об'єднання по "І" (рис. 3.3) для кожного патерну відрізняється на одиницю від кількості ЛТ $L_{\&}$ у цьому ж конвеєру згідно (4.12), тому що після останнього каскаду схеми "І" тригер не потрібен:

$$F_{\&} = L_{\&} - 1 = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil - 1. \qquad (4.18)$$

Отже, згідно (4.14), враховуючи (4.15), (4.17) та (4.18), отримуємо загальну кількість тригерів у схемі BsCAM:

$$F_{\text{BSCAM}} = 8\, m_{\max} + m_{\min}\left\lceil \frac{\sigma-1}{y-1} \right\rceil + \sum_{j=m_{\min}+1}^{m_{\max}} \left\lceil \frac{\sum_{i=j}^{m_{\max}} \delta_i - 1}{y-1} \right\rceil + \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil - 1. \qquad (4.19)$$

Нарешті, загальна кількість УЛТ у базовій схемі BsCAM розпізнавання на базі ЦК згідно виразів (4.7), (4.13) та (4.19):

$$R^*_{BSCAM} = L_{BSCAM} + \alpha F_{BSCAM} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\Lambda(x)\right) \cdot j + \left\lceil \frac{j-1}{x-1} \right\rceil +$$

$$+\alpha\left(8m_{\max}+m_{\min}\left\lceil\frac{\sigma-1}{y-1}\right\rceil+\sum_{i=m_{\min}+1}^{m_{\max}}\left\lceil\frac{\sum_{i=j}^{m_{\max}}\delta_i-1}{y-1}\right\rceil+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil-1\right). \quad (4.20)$$

Отже, формула (4.20) в першому теоретичному наближенні дає вихідний результат ресурсної частини ФО для схеми BsCAM.

Ця формула може бути уточнена, згідно наступних міркувань. Структура більшості сучасних ПЛІС така (див. підр. 1.4.2.4 "Логічні комірки"), що кожна логічна комірка одночасно володіє обчислювальними ресурсами двох типів – логічними та розподіленої пам’яті – тому що містить як ЛТ, так і тригері (див. рис. 1.2). Якщо в силу особливості схеми при її синтезі вдається використати обидва ресурси з однієї й тій самій логічній комірці, підрахунки ресурсів кожного типу окремо призведуть до дещо завищених результатів. Наприклад, якщо роздивитися конвеєрну схему багатовходового логічного елементу "І" (рис. 3.3), можна зрозуміти, що тригери та ЛТ, потрібні для його синтезу, можуть бути використані сумісно з однієї логічної комірки ПЛІС, тому порахована кількість, скажімо, тригерів може бути зменшена на величину $F_{\&}\approx L_{\&}$, і доданок $F_{\&}$ у виразі (314) можна прийняти рівним нулю. В результаті чого отримуємо питому залежність у другому теоретичному наближенні:

$$R^{**}_{\mathrm{BSCAM}}=\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left(\Lambda(x)j+\left\lceil\frac{j-1}{x-1}\right\rceil\right)+$$

$$+\alpha\left(8m_{\max}+m_{\min}\left\lceil\frac{\sigma-1}{y-1}\right\rceil+\sum_{j=m_{\min}+1}^{m_{\max}}\left\lceil\frac{\sum_{i=j}^{m_{\max}}\delta_i-1}{y-1}\right\rceil\right). \quad (4.21)$$

Проаналізуємо здобутий результат.

Вираз (4.21) є ресурсною складовою ФО, яка залежить від таких вхідних змінних, як параметри множини патернів, що розпізнаватимуться, та деяких параметрів ПЛІС, що використовуватимуться: $\theta_R = \theta_R(P, \Theta) = R^{**}_{\text{BSCAM}}(\sigma, m_{\min}, m_{\max}, \delta, x, y)$. Зауважимо, що в даному виразі відсутні змінні μ та ν, що є функціями самоподоби множини патернів. Це свідчить про те, що базова схема BsCAM не використовує надмірність, яка присутня в словнику сигнатур.

4.6.1.2. Функція оцінки базової схеми BsCAM. Часова складова

Побудуємо тепер часову складову ФО θ_T для схеми розпізнавання на основі схеми BsCAM.

Перевагою підходу на ЦК є висока швидкодія. Порівняння вхідного байта з символом патерну здійснюється з один такт, тобто за один період тактової частоти:

$$T = \frac{1}{f_T}, \tag{4.22}$$

де f_T – максимальна тактова частота, на якій здатна функціонувати синтезована в ПЛІС схема. Ця частота в досить широких межах коливається в залежності від властивостей проекту, і, як слідство, від обчислювальної структури, що створюється на кристалі в результаті його синтезу. максимальну частоту проекту f_T можна знайти лише шляхом компіляції проекту, виконуючи всі етапи синтезу аж до останнього - розміщення та трасування (Place & Route) (див. підр. 1.5.4 "Складності при використанні РУО та шляхи їх усунення"). Але, як було вказано раніше, через велику часову ємність цього процесу його неможливо використовувати безпосередньо в циклі оптимізації.

Тому пропонується використовувати наступну техніку. Для даного компоненту емпіричним шляхом заздалегідь відшукується апроксимаційна функціональна залежність від ресурсних витрат на побудову цифрової схеми у вигляді, наприклад, кривої другого порядку, яка з певною похибкою дозволяє передбачити майбутню максимальну частоту без її компіляції:

$$f_T = A_T + B_T R + C_T R^2 .$$

Таким чином отримуємо в першому емпіричному наближенні значення часової затримки для питомого компонента в якості складової частини ФО:

$$T^* = \frac{1}{A_T + B_T R + C_T R^2} .$$

У співвідношенні (4.22) ми припускали, що вся синтезована в ПЛІС схема функціонує на одній частоті. Але існують техніки, завдяки яким складна цифрова структура, що створюється в реконфігуровному середовищі, поділяється на так звані часові зони, всередині яких використовується різна тактова частота [125]. Подібне рішення експлуатує той факт, що інформація вздовж тракту її проходження по схемі, обробляється з різною інтенсивністю: вхідні ланцюги розгалужуються довгими лініями зв’язку на чисельні але більш компактні ділянки, в яких частота синтезу може бути істотно підвищена. Тому функціонування різних зон на різних частотах дозволяє підвищити максимально можливу тактову частоту f_T і, як слідство, прискорити роботу обчислювальної структури в цілому. Наслідком такого прийому стосовно підходу на базі АП є той факт, що цифрові компаратори працюють на більш високій частоті, ніж вхідні та вихідні ланцюги. Ця різниця визначається коефіцієнтом ділення частоти S, який вказує, в скільки раз відрізняється тактова частота в "швидких" та "повільних" зонах. Але швидкодія блоку розпізнавання обумовлюється темпом функціонування саме

зовнішніх елементів. Тому, з урахуванням розглянутої техніки, в загальному випадку значення часової затримки в другому емпіричному наближенні ФО приймає наступний вигляд:

$$T^{**} = \frac{S}{A_T + B_T R + C_T R^2}. \tag{4.23}$$

Розглядаючи питання швидкодії слід згадати про інший часовий показник – латентність, що є затримкою розповсюдження даних від входу до виходу, яка може в декілька разів перевищувати час обробки одиниці даних, але не призводить до втрати продуктивності. Така ситуація виникає при використанні конвеєрного методу обробки інформації. В розглянутій вище схемі BsCAM конвеєризація використовується, по-перше, при розгалуженні сигналів на входи чисельних компараторів, по-друге, при об'єднанні по "І" їх виходів. Вхідний конвеєр не впливає на латентність, тому що послідовний принцип обробки вхідних символів є особливістю всіх пристроїв розпізнавання рядків – шуканий патерн не може бути розпізнаним поки відповідний фрагмент вхідної послідовності повністю не введений в схему.

Обчислити значення латентності схеми DSCAM, користуючись наведеними вище міркуваннями, не складно, але в даному дослідженні воно не знадобиться.

4.6.1.3. Функція оцінки схеми DCAM. Ресурсна складова

Схема DCAM (див. рис. 3.5) на відміну від BsCAM не потребує компараторів, проте містить один дешифратор, який у випадку байтового кодування має організацію 8/256 та певну кількість цифрових схем затримки DEL(j) на j тактів кожна, які будуються на ЛТ. Виходи цих ЦСЗ об’єднуються багатовходовими схемами "І" відповідно до патернів, що мають розпізнаватися.

ЛТ в схемі DCAM використовуються, по-перше, для синтезу дешифратора, по-друге – для цифрових схем затримки DEL, и по-третє – для створення багатовходових схем "I", що будуються у вигляді конвеєра (див. рис. 3.3):

$$L_{\mathrm{DCAM}} = L_{\mathrm{DC}} + L_{\mathrm{DEL}} + L_{\&}. \tag{4.24}$$

Дешифратор 8/256 складається з 256 компараторів, аналогічних компонентам CMP в схемі BsCAM. Тому згідно (4.9):

$$L_{\mathrm{DC}} = 256\,\Lambda(x). \tag{4.25}$$

де x – число входів ЛТ для заданої ПЛІС.

Підрахунок кількості ЦСЗ для схеми DCAM не є тривіальним завданням.

На рис. 4.3 наведено схематичне зображення обчислювальної структури схеми розпізнавання DCAM.

Тут стовпчики з позначкою "&" зображають багатовходові схеми "I" для відповідних патернів $p_1, p_2, p_3, \ldots p_k, \ldots p_\sigma$, що входять до множини P; прямокутники DEL(j – 1) – комплекти однакових схем затримки на (j – 1) тактів кожна, входи яких (одиночні дроти) підключені до потрібних виходів дешифратора, а виходи (також одиночні) – до входів відповідних схем "I".

Щоб не заплутатися в подальших розрахунках, звернемо увагу на наступні два факти.

По-перше, оскільки фрагменти рядків, що збігаються з патернами, в складі вхідної послідовності подаються на вхід схеми починаючи з їх перших символів (які потребують найбільшої затримки), найнижчі частини стовпчиків на рисунку відповідають першим символам патернів, а найвищі – останнім. Щоб нумерація позиції символу в патерні не була протилежною

значенню затримки у відповідних комплектах ЦСЗ, пронумеруємо символи у патернах у зворотному порядку – зверху к низу: $j = 1, 2, 3, \ldots m_{\max}$. Тобто у кожному патерні останній символ матиме номер 1, а перший – j, де j – довжина цього патерну. Перші символи найдовших патернів в наборі відповідно матимуть номер $m_{\max}$.

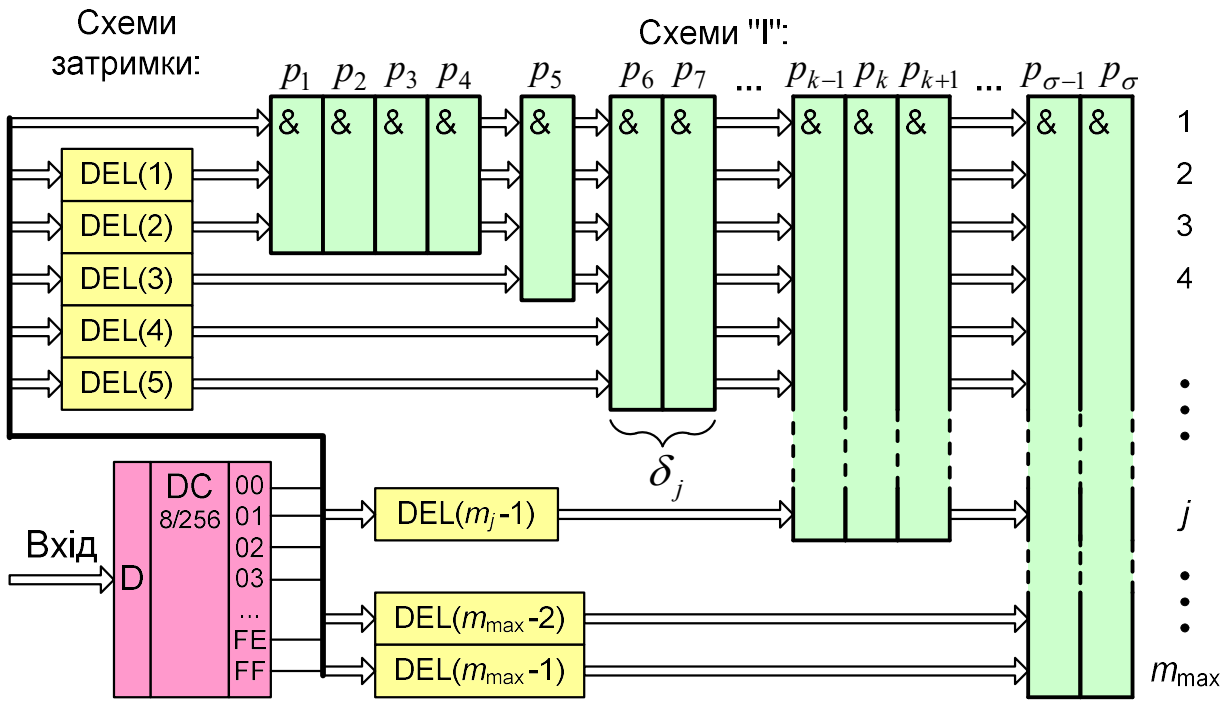


Рис. 4.3. Обчислювальна структура схеми розпізнавання DCAM

По-друге, оскільки верхні тобто останні символи патернів (для яких $j = 1$) розпізнаються без затримки (інакше кажучи, з затримкою в нуль тактів), кожній позиції символу в патерні номер j відповідає затримка довжиною в $(j - 1)$ тактів DEL$(j - 1)$ = DEL$(m_j - 1)$, де m_j - довжина патерну, що складається з j символів.

Кількість ЦСЗ в комплекті не може бути більшою за значення 256 (при кодуванні одним байтом). Як наслідок у випадку, коли число патернів суттєво перевищує 256, значна кількість одних й тих самих символів обов'язково будуть знаходитися на однакових позиціях у різних патернах. Нагадаємо, що саме це значення – загальна кількість символів з кодом s на

j-ої позиції всіх патернів – визначається першою функцією самоподоби $\mu(s, j)$, $s = 0, 1, \ldots 255$ (для байтового кодування), $j = 1, 2, 3, \ldots m_{\max}$.

Як можна побачити на рис. 4.3, на j-й вхід кожної схеми "I" подається сигнал з виходу якоїсь з схем затримки на $(j - 1)$ тактів з відповідного комплекту DEL(j – 1). Очевидно, що кількість коротших ЦСЗ в кожному комплекті буде наближатися до 256, оскільки серед останніх символів всіх патернів словнику з високою вірогідністю можуть знайтися майже всі символи алфавіту. З іншого боку, в останньому комплекті DEL($m_{\max}$ – 1), який відповідає першим символам найдовших патернів, кількість ЦСЗ буде не більшою за число патернів в останньому пакеті (в наведеному на рисунку прикладі – лише два). У найгіршому випадку якщо всі перші символи різні, ця кількість буде дорівнювати кількості найдовших патернів. В найкращому випадку, якщо всі перші символи найдовших патернів однакові, комплект DEL($m_{\max}$ – 1) буде містити лише одну схему затримки (довжиною в ($m_{\max}$ - 1) тактів).

В загальному випадку довільний символ алфавіту s може бути або наявним на j-ої позиції хоча б одного з патернів, або відсутнім взагалі.

Факт наявності символу s на j-ої позиції хоча б одного з патернів словнику буде визначатися виразом $\mathrm{NotZ}(\mu(s, j))$, де NotZ( ) – функція нерівності нулю згідно (4.2). А загальна кількість схем затримки у комплекті на (j – 1) тактів *Dj-1* буде дорівнювати

$$D_{j-1} = \sum_{s=0}^{255} \mathrm{NotZ}(\mu(s, j)). \tag{4.26}$$

На той факт, що ЛТ можуть бути використані для побудови ЦСЗ, вже вказувалося в підр. 3.1.5 "Зменшення ресурсних витрат". Розглянемо цю техніку докладніше.

Різні сімейства ПЛІС дозволяють будувати ЦСЗ з різним максимальним значенням затримки. Наприклад, САПР фірми Xilinx пропонують розробникам бібліотечні компоненти FD, SRL16 та SRL32, що використовують відповідно: один тригер логічної комірки для створення ЦСЗ на 1 такт, одну 4-входову ЛТ для створення ЦСЗ на довільну величину від 1 до 16 тактів та одну 6-входову ЛТ для створення ЦСЗ на довільну величину від 1 до 32 тактів [166], [167].

Назвемо *показником z* максимальне можливе значення затримки ЦСЗ на базі ЛТ для даної ПЛІС. (В наведеному вище прикладі z = 16 для мікросхем ПЛІС фірми Xilinx, що містять 4-входові ЛТ, та z = 32 для моделей мікросхем, в яких ЛТ мають більше входів).

Слід зауважити, що до складу кожної схеми ЦСЗ обов'язково має входити принаймні одна однотактна схема затримки FD на базі тригеру. Більш того, якщо складна схема ЦСЗ будується з послідовності кількох компонентів SRL16 або SRL32, між цими компонентами також необхідно вставляти елементи FD [116].

Отже кількість ЛТ (рівно як і тригерів), потрібних для синтезу однієї ЦСЗ DEL(j – 1) в залежності від значення затримки (j – 1):

$$L_{j-1} = \left\lceil \frac{j-1}{z+1} \right\rceil. \tag{4.27}$$

Тоді загальна кількість ЛТ, потрібних для створення всіх ЦСЗ:

$$L_{\mathrm{DEL}} = \sum_{j=2}^{m_{\max}} L_{j-1} D_{j-1} = \sum_{j=2}^{m_{\max}} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \mathrm{NotZ}(\mu(s, j)) \right). \tag{4.28}$$

Кількість ЛТ, потрібних для створення багатовходової схеми "І", що будується у вигляді конвеєра (рис. 3.3) для схеми DCAM, дорівнює тій самій величині $L_{\&}$, що й для базової схеми на компараторах BsCAM, яка

обчислюється згідно виразу (4.12), тому що число схем "І" та кількість входів кожної з них не змінюються.

Отже, загальна кількість ЛТ для всієї схеми DCAM здобувається шляхом підстановки (4.25), (4.28) та (4.12) у (4.24):

$$L_{\mathrm{DCAM}} = L_{\mathrm{DC}} + L_{\mathrm{DEL}} + L_{\&} = 256\,\Lambda(x) +$$

$$+ \sum_{j=2}^{m_{\max}} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \mathrm{NotZ}(\mu(s,j)) \right) + \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil, \qquad (4.29)$$

де, нагадаємо, δ_j – функція розподілу довжин патернів.

Тригери в схемі DCAM потрібні: в складі схем затримки, для побудови конвеєрів розгалуження як дешифратора, так і виходів схем затримки, а також в конвеєрах багатовходового об'єднання по "І" для всіх патернів:

$$F_{\mathrm{DCAM}} = F_{\mathrm{DEL}} + F_{\mathrm{fanDC}} + F_{\mathrm{fanDEL}} + F_{\&}. \qquad (4.30)$$

Кількість тригерів, потрібних для створення всіх ЦСЗ дорівнює кількості ЛТ (4.28), потрібних для створення всіх ЦСЗ:

$$F_{\mathrm{DEL}} = L_{\mathrm{DEL}} = \sum_{j=2}^{m_{\max}} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \mathrm{NotZ}(\mu(s,j)) \right) \qquad (4.31)$$

Кількість ЦСЗ, що підключені до кожного виходу s дешифратора, можна обчислити шляхом перебору всіх позицій j в патернах, враховуючи при цьому, чи є хоча б один патерн, в якому присутній символ з кодом s в даній позиції. Тобто, коефіцієнт розгалуження з виходу s дорівнює числу

ненульових значень першої функції самоподоби $\mu(s, j)$ для всіх позицій j в патерні для даного символу:

$$O_s = \sum_{j=2}^{m_{\max}} \mathrm{NotZ}(\mu(s, j)).$$

Кількість тригерів, потрібних для розгалуження одного виходу дешифратора на O_s входів за допомогою конвеєра згідно рис. 3.2 обчислюється за формулою (4.16):

$$F_{s,\mathrm{DC}} = \left\lceil \frac{O_s - 1}{y - 1} \right\rceil = \left\lceil \frac{\sum_{j=2}^{m_{\max}} \mathrm{NotZ}(\mu(s, j)) - 1}{y - 1} \right\rceil$$

де y – здатність навантаження виходів тригерів для заданої ПЛІС.

Загальна кількість тригерів, потрібних для розгалуження всіх виходів дешифраторів:

$$F_{\mathrm{fanDC}} = \sum_{s=0}^{255} F_{s,DC} = \sum_{s=0}^{255} \left\lceil \frac{\sum_{j=2}^{m_{\max}} \mathrm{NotZ}(\mu(s, j)) - 1}{y - 1} \right\rceil. \qquad (4.32)$$

Вихід кожної окремої схеми затримки DEL(j – 1) в залежності від коду s символу, що розпізнається, розгалужується рівно на $\mu(s, j)$ входів багатовходових логічних елементів "І". Кількість тригерів, потрібних для конвеєру його розгалуження, обчислюється також за формулою (4.16):

$$F_{s,\mathrm{DEL}j} = \left\lceil \frac{\mu(s,j)-1}{y-1} \right\rceil.$$

Тоді загальна кількість тригерів для розгалуження виходів схем затримки для всього алфавіту для кожній позиції j в патернах:

$$F_{\mathrm{fanDEL}j} = \sum_{s=0}^{255} F_{s,\mathrm{DEL}j} = \sum_{s=0}^{255} \left\lceil \frac{\mu(s,j)-1}{y-1} \right\rceil.$$

А загальна кількість тригерів, потрібних для розгалуження всіх виходів ЦСЗ:

$$F_{\mathrm{fanDEL}} = \sum_{j=2}^{m_{\max}} F_{\mathrm{fanDEL}j} = \sum_{j=2}^{m_{\max}} \sum_{s=0}^{255} \left\lceil \frac{\mu(s,j)-1}{y-1} \right\rceil. \qquad (4.33)$$

Зауважимо, що у формулі (4.33) додавання по j здійснюється не до ($m_{\max}$ - 1), а на одиницю більше тому, що для умовно нульових затримок навантаження з виходів ЦСЗ перекладається на виходи дешифратора.

Кількість тригерів в конвеєрі для багатовходового об'єднання по "І" для схеми DCAM, як і в випадку з DSCAM, відрізняється на одиницю від кількості ЛТ $L_{\&}$ у цьому ж конвеєру (4.18).

Отже, підставляючи (4.31), (4.32), (4.33) та (4.18) у (4.30), отримуємо загальну кількість тригерів, потрібних для побудови всієї схеми DCAM:

$$F_{\mathrm{DCAM}} = F_{DEL} + F_{\mathrm{fanDC}} + F_{\mathrm{fanDEL}} + F_{\&} = \sum_{j=2}^{m_{\max}} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \mathrm{NotZ}(\mu(s,j)) \right) +$$

$$\sum_{s=0}^{255}\left\lceil\frac{\sum_{j=2}^{m_{\max}}\mathrm{NotZ}(\mu(s,j))-1}{y-1}\right\rceil+\sum_{s=0}^{255}\left\lceil\frac{\mu(s,j)-1}{y-1}\right\rceil+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil-1=$$

$$=\sum_{j=2}^{m_{\max}}\left(\left\lceil\frac{j-1}{z+1}\right\rceil\sum_{s=0}^{255}\mathrm{NotZ}(\mu(s,j))\right)+$$

$$=\sum_{s=0}^{255}\left\lceil\frac{\sum_{j=2}^{m_{\max}}\mathrm{NotZ}(\mu(s,j))+\mu(s,j)-2}{y-1}\right\rceil+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil-1. \qquad (4.34)$$

Знайдемо, нарешті, для схеми DCAM ресурсну складову ФО в першому теоретичному наближенні за формулою (4.7), використовуючи вирази (4.29) та (4.34):

$$R^*_{\mathrm{DCAM}}=L_{\mathrm{DCAM}}+\alpha F_{\mathrm{DCAM}}=256\Lambda(x)+$$

$$+\sum_{j=2}^{m_{\max}}\left(\left\lceil\frac{j-1}{z+1}\right\rceil\sum_{s=0}^{255}\mathrm{NotZ}(\mu(s,j))\right)+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil+$$

$$+\alpha\left(\sum_{j=2}^{m_{\max}}\left(\left\lceil\frac{j-1}{z+1}\right\rceil\sum_{s=0}^{255}\mathrm{NotZ}(\mu(s,j))\right)\right)+$$

$$+\alpha\left(\sum_{s=0}^{255}\left\lceil\frac{\sum_{j=2}^{m_{\max}}\mathrm{NotZ}(\mu(s,j))+\mu(s,j)-2}{y-1}\right\rceil+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil-1\right)=$$

$$=256\Lambda(x)+(1+\alpha)\sum_{j=2}^{m_{\max}}\left(\left\lceil\frac{j-1}{z+1}\right\rceil\sum_{s=0}^{255}\mathrm{NotZ}(\mu(s,j))\right)+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil+$$

$$+\alpha\left(\sum_{s=0}^{255}\left\lceil\frac{\sum_{j=2}^{m_{\max}}\mathrm{NotZ}(\mu(s,j))+\mu(s,j)-2}{y-1}\right\rceil+\sum_{j=m_{\min}}^{m_{\max}}\delta_j\left\lceil\frac{j-1}{x-1}\right\rceil-1\right)= \qquad (4.35)$$

Формула (4.35) може бути уточнена так само, як і для схеми BsCAM. Оскільки кожна логічна комірка ПЛІС містить як ЛТ, так і тригері, які можуть бути використані сумісно при створенні конвеєрної схеми багатовходового логічного елементу "І", доданок $F_{\&}$ з виразу (4.30) можна вилучити. Але при створенні ЦСЗ так само можуть бути використані сумісно ЛТ і тригери, що містяться в однієї логічній комірці. Тому доданок F_{DEL} також можна вилучити. Тоді отримуємо питому залежність у другому теоретичному наближенні в наступному вигляді:

$$R^{**}_{\mathrm{DCAM}}=256\Lambda(x)+\sum_{j=2}^{m_{\max}}\left(\left\lceil\frac{j-1}{z+1}\right\rceil\sum_{s=0}^{255}\mathrm{NotZ}(\mu(s,j))\right)+$$

$$+\sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil + \alpha \sum_{s=0}^{255} \left\lceil \frac{\sum_{j=2}^{m_{\max}} \mathrm{NotZ}(\mu(s,j)) + \mu(s,j) - 2}{y-1} \right\rceil. \qquad (4.36)$$

Проаналізуємо здобутий результат.

Вираз (4.36) є ресурсною складовою ФО для схеми DCAM на базі ЦК: $\theta_R = R^{**}_{\mathrm{DCAM}}(m_{\min}, m_{\max}, \delta, \mu, x, y, z, \alpha)$, яка може бути використана при розрахунку цільової функції для оптимізації структури МР, в склад якої входить блок розпізнавання, побудований на базі даної схеми. Даний вираз залежить як від параметрів патернів, що мають розпізнаватися (мінімальної $m_{\min}$ та максимальної $m_{\max}$ довжини патернів, функцій розподілу довжин δ та самоподоби μ), так і від параметрів ПЛІС, що використовується (числа x входів ЛТ, здатності навантаження логічних елементів у максимального розміру схеми затримки z, побудованої на базі ЛТ), а також від коефіцієнту нормалізації α. На відміну від схеми BsCAM, кількість ресурсів в схемі DCAM залежить від принаймні однієї функції самоподоби, що дозволяє використовувати надмірність, притаманну базі даних сигнатур.

4.6.1.4. Функція оцінки схеми DCAM. Часова складова

Часова складова ФО θ_T для компоненту на основі схеми DCAM не відрізняється від такої для схеми BsCAM. Всі твердження та розрахунки подібні, включаючи емпіричну корекцію робочої частоти шляхом апроксимації другого порядку.

Оскільки техніка поділу проекту вздовж простору кристалу на "часові зони" також може бути застосовано для схеми декодованої АП, остаточне значення часової складовій ФО схеми DCAM в другому емпіричному наближенні також обчислюється за формулою (4.23).

4.6.1.5. Функція оцінки схеми DpCAM. Ресурсна складова

Схема DpCAM (див. рис. 3.6) здебільшого подібна до схеми DCAM, що розглянута вище. Відміна полягає у тому, що багатовходові схеми "І" замінюються на комбінації більш коротших схем "І" та ЦСЗ, синтезованих на базі ЛТ з максимально можливою затримкою.

При розпізнаванні патернів, що потребують більше двох каскадів, схема дещо змінюється порівняно з рис. 3.6. В зв'язку з необхідністю виділяти один вхід схем "І" на перенос з попереднього каскаду необхідно або використовувати ЦСЗ не на максимальну затримку, на яку здатні ЛТ обраної ПЛІС, або задіяти схеми "І" на непарну кількість входів, що перешкоджає ефективному використанню ЛТ для її синтезу. Перший варіант виявляється більш ефективним. На рис. 4.4 подано приклад обчислювальної структури схеми DpCAM, побудований саме за цим варіантом.

Приклад розраховано на використання 4-входових ЛТ, на базі яких створюються ЦСЗ тривалістю до 16 тактів (тобто параметр РУО z = 16), і здатен розпізнавати патерні довжиною 60 символів.

Дана схема, як и структура на рис. 4.3, містить дешифратор організацію 8/256. Але комплекти однакових ЦСЗ (компоненти DEL(1), DEL(2), … , DEL(14)) мають затримки лише від 1 до 14, в той час як в схемі DCAM комплекти ЦСЗ забезпечують затримки аж до ($m_{\max}$ - 1), тобто майже до величини, що дорівнює довжині найдовшого патерну. Натомість багатовходові схеми "І" тут заміщуються схемами, які крім логіки ще містять ЦСЗ на 15 тактів.

Частина схеми, що обведена на рис. 4.4 штриховою лінією, позначає компонент p_k, (k = 1, 2, … , σ), що будується *окремо для кожного патерну*, в той час як комплекти схем затримки DEL(1), DEL(2), … , DEL(15) використовуються сумісно для розпізнавання всіх патернів, тобто є *загальною частиною* схеми. Виходи останніх (одиночні дроти) в залежності від змісту патернів підключені до входів схем "І" з номерами 1, 2, 3, 4 наступнім чином: виходи DEL(1) – до потрібних входів 2 схем "І", виходи

DEL(2) – до входів 3 схем "І", … виходи DEL(14) – до входів 15 схем "І". Вхід 16 останньої схеми "І", який залишається вільним, також можна було б задіяти, додавши до комплектів однакових ЦСЗ групу затримок DEL(15), але це порушить регулярність структури та призведе до неефективних витрат ресурсів

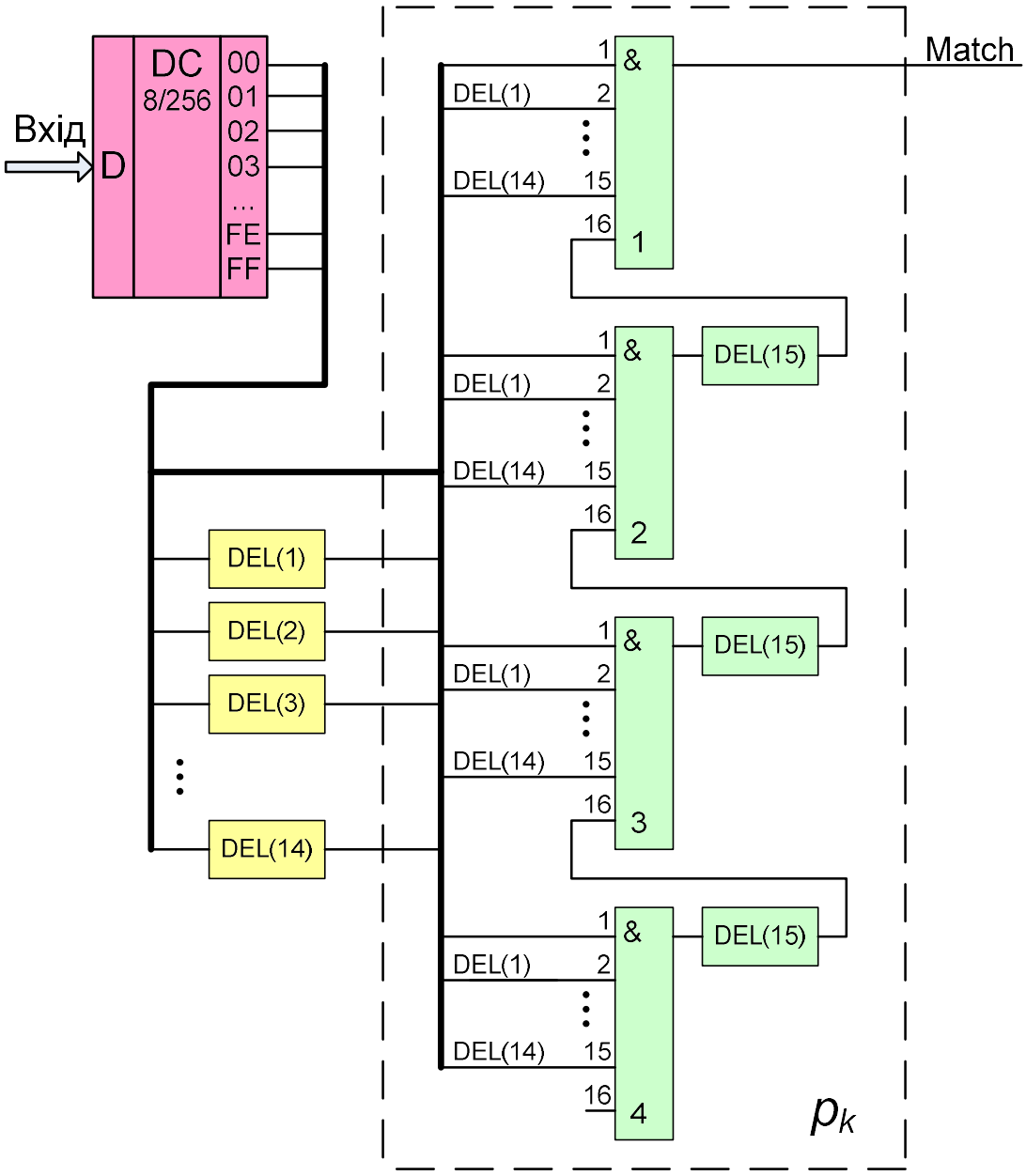


Рис. 4.4. Обчислювальна структура схеми розпізнавання DpCAM

Як і для схеми DCAM, символи у патернах нумеруються у зворотному порядку – зверху к низу. Тобто вхід 1 схеми "1" з номером 1 відповідає

останньому символу патерну p_k (для нього $j = 1$), в той час як вхід 15 схеми "1" з номером 4 – першому, що має розпізнаватися з максимальною затримкою (для нього $j = m_{\max}$). Зауважимо, що при такій нумерації номер j позиції символу в патерні зростає пропорційно значенню затримки DEL(j - 1) відповідної ЦСЗ загальної частини схеми.

В довільному випадку загальна частина схеми містить комплекти однакових ЦСЗ, що мають затримки від 1 до (z – 2) тактів. Окремі компоненти p_k складаються зі схем "І" на z входів кожна (остання з яких може мати довільну кількість входів від 2 до (z – 1)), та ЦСЗ на (z – 1) тактів.

Порахуємо ресурси, що потрібні для створення схеми DpCAM на базі ПЛІС, яка має параметр максимального значення затримки ЦСЗ, створюваної на бази ЛТ рівним z та x-входові логічні таблиці.

Логічних ресурсів для схеми DpCAM потрібно:

$$L_{\text{DpCAM}} = L_{\text{DC}} + L_{\text{DEL}} + L_{\&\,pk} + L_{\text{DEL}pk}, \tag{4.37}$$

де LDC – ресурси для синтезу дешифратора,

L_{DEL} – ресурси для синтезу ЦСЗ загального використання,

$L_{\&pk}$ – ресурси для синтезу схем "І" окремо для патернів,

$L_{\text{DEL}pk}$ – ресурси для синтезу ЦСЗ окремо для патернів.

Ресурси L_{DC} для синтезу дешифратора потрібні такі ж самі, як і для схеми DCAM, тому визначаються виразом (4.25).

Кількість L_{DEL} логічних таблиць, потрібних для створення всіх ЦСЗ, загальної частини розраховується подібно до схеми DCAM (за формулою (4.28)), але, по-перше, не для ($m_{\max}$ - 1), а для (z - 2) ЦСЗ (індекс (j - 1) змінюється від 2 до (z - 2)), по-друге – із застосуванням замість першої функції самоподоби першої часткової функції самоподоби (яку було введено саме для розрахунків витрат схеми DpCAM):

$$L_{\mathrm{DEL}} = \sum_{j=2}^{z-1}\left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \mathrm{NotZ}\left(\mu_{z-1}(s,j)\right)\right). \tag{4.38}$$

Оскільки кожний повний (крім останнього) елемент "I" в складі схеми DpCAM, що має z входів, може бути складений з $\left\lceil \frac{z}{x} \right\rceil + 1$ x-входових ЛТ, всього таких повних елементів для патерну довжиною j потрібно $\left\lfloor \frac{j}{z-1} \right\rfloor$, а останній елемент "I" в довільному випадку має j mod (z - 1) входів, то кількість $L_{\&j}$ логічних таблиць для створення всіх елементів "I" схеми p_k для розпізнавання одного патерну довжиною j дорівнює:

$$L_{\&\,j} = \left\lfloor \frac{j}{z-1} \right\rfloor \left(\left\lceil \frac{z}{x} \right\rceil + 1 \right) + \left\lceil \frac{j \bmod (z-1)}{x} \right\rceil + 1. \tag{4.39}$$

Для патернів всіх довжин від $m_{\min}$ до $m_{\max}$ з урахуванням (4.39), а також того факту, що патернів довжини j в словнику міститься δ_j, загальна кількість ЛТ для створення всіх схем "I":

$$L_{\&\,pk} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lfloor \frac{j}{z-1} \right\rfloor \left(\left\lceil \frac{z}{x} \right\rceil + 1 \right) + \left\lceil \frac{j \bmod (z-1)}{x} \right\rceil + 1 \right). \tag{4.40}$$

Кількість $L_{\mathrm{DEL}j}$ логічних ресурсів для синтезу всіх ЦСЗ схеми p_k для розпізнавання одного патерну довжиною j дорівнює кількості всіх елементів "I", включаючи неповний:

$$L_{\mathrm{DEL}j} = \left\lceil \frac{j}{z-1} \right\rceil. \tag{4.41}$$

Для всіх патернів схеми DpCAM кількість ЛТ розраховується подібно до (4.40):

$$L_{\mathrm{DEL}pk} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j}{z-1} \right\rceil . \tag{4.42}$$

Тригери в схемі DpCAM потрібні: 1) в складі ЦСЗ як загального використання, так і окремо для патернів; 2) з метою зниження навантаження на виходи дешифратора та ЦСЗ загального використання шляхом побудови конвеєрів розгалуження:

$$F_{\mathrm{DpCAM}} = F_{\mathrm{DEL}} + F_{\mathrm{DEL}pk} + F_{\mathrm{fanDC}} + F_{\mathrm{fanDEL}} . \tag{4.43}$$

Кількості тригерів, потрібних для створення ЦСЗ як загальної частини, так і окремо для патернів, дорівнюють відповідним кількостям ЛТ (4.38) і (4.42), потрібних для їх синтезу:

$$F_{\mathrm{DEL}} = L_{\mathrm{DEL}} = \sum_{j=2}^{z-1} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \mathrm{NotZ}\left(\mu_{z-1}(s,j)\right) \right) , \tag{4.44}$$

$$F_{\mathrm{DEL}pk} = L_{\mathrm{DEL}pk} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lceil \frac{j}{z-1} \right\rceil \right) . \tag{4.45}$$

Кількості тригерів, потрібних для розгалуження виходів дешифраторів розраховуються також подібно до схеми DCAM (за формулами (4.32) та (4.33)), але для (z - 2) схем затримки замість ($m_{\max}$ - 1), та використовуючи першу часткову функцію самоподоби:

$$F_{\text{fanDC}} = \sum_{s=0}^{255} \left\lceil \frac{\sum_{j=2}^{z-1} \text{NotZ}(\mu_{z-1}(s,j)) - 1}{y-1} \right\rceil. \tag{4.46}$$

$$F_{\text{fanDEL}} = \sum_{j=2}^{m_{\max}} F_{\text{fanDEL}j} = \sum_{j=2}^{z-1} \sum_{s=0}^{255} \left\lceil \frac{\mu_{z-1}(s,j) - 1}{y-1} \right\rceil. \tag{4.47}$$

Отже, загальна кількість ресурсів, потрібних для створення схеми DpCAM, тобто ресурсна складова ФО для схеми DCAM, що складається тільки з витрат на логіку та тригери, в першому теоретичному наближенні з урахуванням (4.25), (4.37), (4.38), (4.40), (4.42), (4.42), (4.43), (4.44) (4.45), (4.46) та (4.47) дорівнює:

$$R^*_{\text{DpCAM}} = L_{\text{DpCAM}} + \alpha F_{\text{DpCAM}} =$$

$$= L_{\text{DC}} + L_{\text{DEL}} + L_{\&pk} + L_{\text{DEL}pk} + \alpha(F_{\text{DEL}} + F_{\text{DEL}pk} + F_{\text{fanDC}} + F_{\text{fanDEL}}) =$$

$$= 256\,\Lambda(x) + \sum_{j=2}^{z-1} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \text{NotZ}(\mu_{z-1}(s,j)) \right) + \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lfloor \frac{j}{z-1} \right\rfloor \left(\left\lceil \frac{z}{x} \right\rceil + 1 \right) + \right.$$

$$\left. + \left\lceil \frac{j \bmod (z-1)}{x} \right\rceil + 1 \right) + \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lceil \frac{j}{z-1} \right\rceil \right) + \sum_{j=2}^{z-1} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \text{NotZ}(\mu_{z-1}(s,j)) \right) +$$

$$+ \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lceil \frac{j}{z-1} \right\rceil \right) + \sum_{s=0}^{255} \left\lceil \frac{\sum_{j=2}^{z-1} \text{NotZ}(\mu_{z-1}(s,j)) - 1}{y-1} \right\rceil + \sum_{j=2}^{z-1} \sum_{s=0}^{255} \left\lceil \frac{\mu_{z-1}(s,j) - 1}{y-1} \right\rceil =$$

$$= L_{\mathrm{DC}} + (1+\alpha)L_{\mathrm{DEL}} + L_{\&\,pk} + (1+\alpha)L_{\mathrm{DEL}pk} + \alpha\, F_{\mathrm{fanDC}} + \alpha\, F_{\mathrm{fanDEL}} =$$

$$= 256\,\Lambda(x) + (1+\alpha)\sum_{j=2}^{z-1}\left(\left\lceil \frac{j-1}{z+1}\right\rceil \sum_{s=0}^{255} \mathrm{NotZ}(\mu_{z-1}(s,j))\right) +$$

$$+ \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lfloor \frac{j}{z-1}\right\rfloor \left(\left\lceil \frac{z}{x}\right\rceil + 1\right) +\right.$$

$$\left. + \left\lceil \frac{j \bmod (z-1)}{x}\right\rceil + 1\right) + (1+\alpha)\sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lceil \frac{j}{z-1}\right\rceil\right) +$$

$$+ \alpha \sum_{s=0}^{255}\left\lceil \frac{\sum_{j=2}^{z-1} \mathrm{NotZ}(\mu_{z-1}(s,j)) - 1}{y-1}\right\rceil + \alpha \sum_{j=2}^{z-1}\sum_{s=0}^{255}\left\lceil \frac{\mu_{z-1}(s,j)-1}{y-1}\right\rceil =$$

$$= 256\,\Lambda(x) + \sum_{j=2}^{z-1}\left(\sum_{s=0}^{255}(1+\alpha)\left\lceil \frac{j-1}{z+1}\right\rceil \mathrm{NotZ}(\mu_{z-1}(s,j)) + \alpha\left\lceil \frac{\mu_{z-1}(s,j)-1}{y-1}\right\rceil\right) +$$

$$+ \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lfloor \frac{j}{z-1}\right\rfloor \left(\left\lceil \frac{z}{x}\right\rceil + 2 + \alpha\right) + \left\lceil \frac{j \bmod (z-1)}{x}\right\rceil + 1\right) +$$

$$+ \alpha \sum_{s=0}^{255}\left\lceil \frac{\sum_{j=2}^{z-1} \mathrm{NotZ}(\mu_{z-1}(s,j)) - 1}{y-1}\right\rceil. \qquad (4.48)$$

Формула (4.48) може бути уточнена. Оскільки кожна логічна комірка ПЛІС містить як ЛТ, так і тригері, які при створенні ЦСЗ можуть бути використані сумісно, доданки F_{DEL} та $F_{\mathrm{DEL}pk}$ можна вилучити. В результаті отримуємо питому залежність у другому теоретичному наближенні:

$$R^{**}_{\mathrm{DpCAM}} = 256\,\Lambda(x) + \sum_{j=2}^{z-1}\left(\sum_{s=0}^{255}\left\lceil \frac{j-1}{z+1}\right\rceil \mathrm{NotZ}(\mu_{z-1}(s,j)) + \left\lceil \frac{\mu_{z-1}(s,j)-1}{y-1}\right\rceil\right) +$$

$$+ \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lfloor \frac{j}{z-1}\right\rfloor \left(\left\lceil \frac{z}{x}\right\rceil + 1\right) + \left\lceil \frac{j \bmod (z-1)}{x}\right\rceil + 1\right) +$$

$$+ \sum_{s=0}^{255}\left\lceil \frac{\sum_{j=2}^{z-1} \mathrm{NotZ}(\mu_{z-1}(s,j)) - 1}{y-1}\right\rceil. \qquad (4.49)$$

Проаналізуємо здобутий результат.

Вираз (4.49) є ресурсною складовою ФО для схеми DpCAM на базі ЦК: $\theta_R = R^{**}_{\mathrm{DpCAM}}(m_{\min}, m_{\max}, \delta, \mu_{z-1}, x, y, z, \alpha)$, яка може бути використана при розрахунку цільової функції для оптимізації структури МР, в склад якої входить блок розпізнавання, побудований на базі даної схеми. Даний вираз залежить від тих самих параметрів, що і R^{**}_{DCAM}, за винятком того, що перша функція самоподоби підмінена на першу часткову функцію самоподоби.

4.6.1.6. Функція оцінки схеми DpCAM. Часова складова

Часова складова ФО блоку розпізнавання на основі схеми DpCAM не відрізняється від такої для схем BsCAM та DCAM. Всі твердження та розрахунки подібні, включаючи емпіричну корекцію робочої частоти шляхом

апроксимації другого порядку згідно формули (4.23). Численні ЦСЗ, які містить рішення DpCAM, насправді не вносять затримки в роботу схеми. Натомість вони заміщують собою конвеєр, складений з 8-розрядних регістрів, по якому просувається вхідна послідовність символів в схемі BsCAM.

4.6.2. Техніка створення функцій оцінки для фільтрів Блума

Принципи побудови блоків розпізнавання з використанням фільтра Блума розглянуті в підр. 3.2 "Використання фільтра Блума на базі геш-функцій". Згідно визначення ФБ складається (див. рис. 3.14) з генератора геш-функцій (ГГФ), що містить K схем, кожна з котрих з одного й того самого фрагменту вхідної послідовності обчислює свою геш-функцію, та з регістру бітів довжиною в M розрядів. Результати обчислень геш-функції інтерпретуються як номери комірок у регістрі. С початку всі розряди РБ обнуляються. Потім, під час програмування ФБ, на його вхід по черзі подаються всі патерни з словнику, при цьому в комірки регістру, на які вказують виходи геш-функцій, записуються одиниці. Під час розпізнавання на вхід ФБ замість патернів подається фрагмент реальної вхідної послідовності. Якщо в усіх комірках, на які вкажуть геш-функції, виявляться одиниці, робиться припущення, що фрагмент послідовності на вході ФБ збігається с відповідним патерном. Остаточна перевірка для усунення помилок розпізнавання другого роду здійснюється шляхом виконання процедури уточнення.

Оскільки розмір РБ відносно невеликий, для його створення реконфігуровними засобами добре підходить блокова пам'ять ПЛІС. Тому згідно базової формули (4), до ресурсів будь-якої реконфігуровної схеми ФБ крім логіки та тригерів додаються витрати на модулі блокової пам'яті BRAM, що входять до складу ПЛІС:

$$R_{\mathrm{BF}} = L_{\mathrm{BF}} + \alpha\, F_{\mathrm{BF}} + \beta\, B_{\mathrm{BF}}\,.$$

Витрати на блокову пам'ять входять тільки до складу РБ. Але крім ГГФ (який містить ЛТ та тригери) та РБ до складу ФБ входять допоміжні схеми, витрати на які також потрібно враховувати. Отже більш детально вираз для витрат на ФБ виглядає наступним чином:

$$R_{\text{BF}} = R_{\text{ГФ}} + R_{\text{БР}} + R_{\text{доп}} = L_{\text{ГФ}} + \alpha\, F_{\text{ГФ}} + \beta\, B_{\text{БР}} + L_{\text{доп}} + \alpha\, F_{\text{доп}} =$$

$$= L_{\text{ГФ}} + L_{\text{доп}} + \alpha\,(F_{\text{ГФ}} + F_{\text{доп}}) + \beta\, B_{\text{БР}}, \qquad (4.50)$$

де $L_{\text{ГФ}}$ і $L_{\text{ДОП}}$ – витрати ЛТ відповідно на ГГФ та на допоміжні схеми,

$F_{\text{ГФ}}$ і $F_{\text{ДОП}}$ – витрати тригерів відповідно на ГГФ та на допоміжні схеми,

$B_{\text{РБ}}$ – витрати на блокову пам'ять для РБ, які вимірюються кількістю задіяних блоків BRAM кристалу ПЛІС.

Перш ніж складати ФО для конкретних реалізацій ФБ, розглянемо окремо, як підраховуються ресурсні витрати на ГГФ, допоміжні схеми та РБ.

4.6.2.1. Витрати ресурсів на генератор геш-функцій для ФБ

Нагадаємо, що згідно визначення функцій класу H_3 вхідний бітовий вектор $\overline{a} = \overline{a_1 a_2 a_3 ... a_b ... a_B}$, де B – кількість бітів у ньому, керує побітовим складанням по модулю 2 рядків з матриці $D = [d_{b,g}]$ випадкових двійкових значень таким чином, що у складанні приймають учать тільки рядки, для яких у відповідній позиції вектору $\overline{a}$ присутні одиниці (див. підр. 3.2.3 "Особливості реалізації геш-функцій на ПЛІС"). При цьому індекс g = 1, 2, 3, …, G, де G – розрядність питомих геш-функцій. Дуже спрощену схему генерування геш-функцій за даним алгоритмом можна знайти, наприклад в роботі [182]. Але для підрахування використаних ресурсів необхідно більш детально розібратися з реалізацією на ПЛІС цього алгоритму.

На рис. 4.5 наведено схему побудови в ПЛІС генератора геш-функцій класу H_3 з використанням ЛТ.

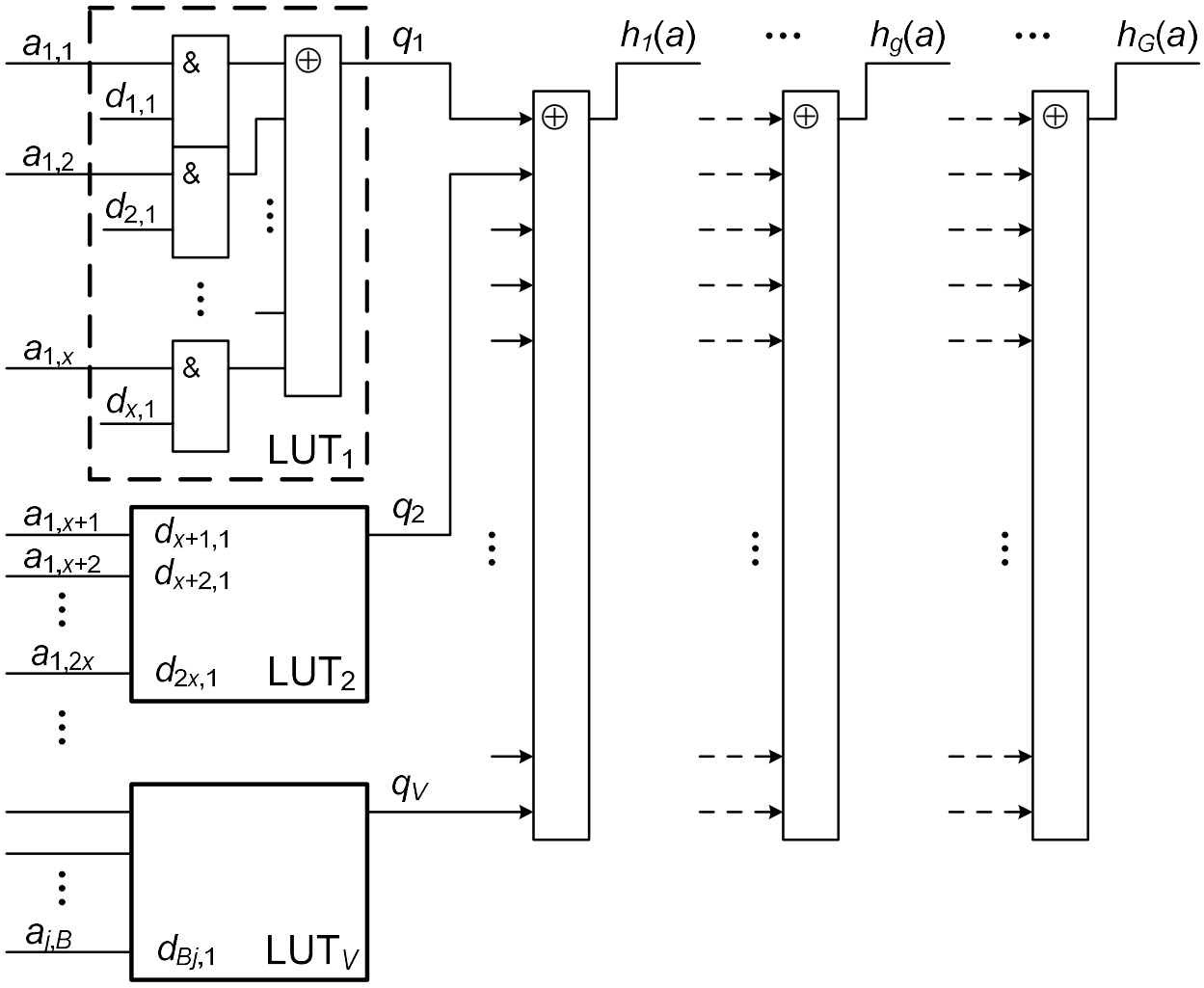


Рис. 4.5. Схема генератора геш-функцій

За даним підходом в кожній ЛТ синтезується схема, яка виконує логічне перемноження бітів $a_{i,b}$ вхідного рядка на елементи матриці D, а також часткове складання по модулю 2 отриманих здобутків. В результаті кожен елементарний компонент LUT_V обробляє стільки вхідних бітів, скільки входів мають логічні таблиці в наданої ПЛІС. Оскільки значення $d_{b,g}$ випадкової матриці D відомі заздалегідь, на етапі синтезу схеми в ПЛІС, вони вводяться в структуру обчислювальної структури в неявній формі (в процесі створення логічної функції, що виконуватиме ЛТ), не потребуючи для себе окремих вхідних ліній.

Подвійна індексація бітів вхідного слова $a_{i,b}$ означає (див. підр. 3.2.3 "Особливості реалізації геш-функцій на ПЛІС"), що, з одного боку, ці біти

групами по 8 (для байтового кодування) входять до складу патерну, що розпізнається, довжина якого дорівнює j символам (i = 1, 2, … j). З іншого – вони поділяються між *x-входовими ЛТ* (LUT_V) групами по x бітів (b = 1, 2, … B, де B – кількість бітів у всьому патерні).

Для обробки всього вхідного патерну, що складається з j символів, тобто, містить 8j бітів, потрібно V логічних таблиць:

$$V = \left\lceil \frac{8j}{x} \right\rceil, \tag{4.51}$$

де x – число входів ЛТ в наданої ПЛІС.

Виходи q_v (v = 1, 2, … V) елементарних компонентів LUT_V остаточно складаються по модулю 2 у відповідній багатовходової схемі, яка синтезується з ЛТ та тригерів шляхом побудови конвеєру, подібного тому, що використовувався для багатовходових схем "І" в схемах на ЦК (див. рис. 3.3). Згідно формули (4.11) для створення такого конвеєру потрібна кількість логічних таблиць:

$$L_V = \left\lceil \frac{V-1}{x-1} \right\rceil. \tag{4.52}$$

Вихід багатовходової схеми складання по модулю 2, що об'єднує виходи q_v одного стовпчику компонентів LUT_V згідно рис. 4.5, видає значення одного розряду $h_g(x)$ геш-функції. Для отримання всіх G розрядів потрібно мати відповідно G таких стовпчиків.

Оскільки для кожного з G розрядів одна логічна таблиця використовуються як елементарний компонент LUT_V (котрих у кожному стовпчику рівно V) та L_V таблиць використовується для створення конвеєру багатовходової схеми складання по модулю 2, кількість ЛТ, потрібних для

створення всіх розрядів однієї геш-функції, можна знайти, використовуючи вирази (4.51) і (4.52), наступним чином:

$$L_G = G\left(V + L_V\right) = G\left(V + \left\lceil \frac{V-1}{x-1} \right\rceil\right) = G\left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil\right), \qquad (4.53)$$

де j – довжина патернів, що розпізнаються,

x – число входів ЛТ в наданої ПЛІС.

Оскільки ФБ потребує обчислення K окремих геш-функцій, для знаходження остаточної кількості ЛТ в складі повнофункціонального генератору геш-функцій класу $H3$ вираз (4.53) потрібно ще помножити на K:

$$L_{H_3} = K\,L_G = K\,G\left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil\right). \qquad (4.54)$$

Тригері в ГГФ потрібні тільки у конвеєрі багатовходового логічного схеми складання по модулю 2. Їх кількість на одиницю менша за кількість ЛТ в даної схемі (4.52):

$$F_V = \left\lceil \frac{V-1}{x-1} \right\rceil - 1.$$

Для всього ГГФ їх також потрібно в $K{\cdot}G$ разів більше:

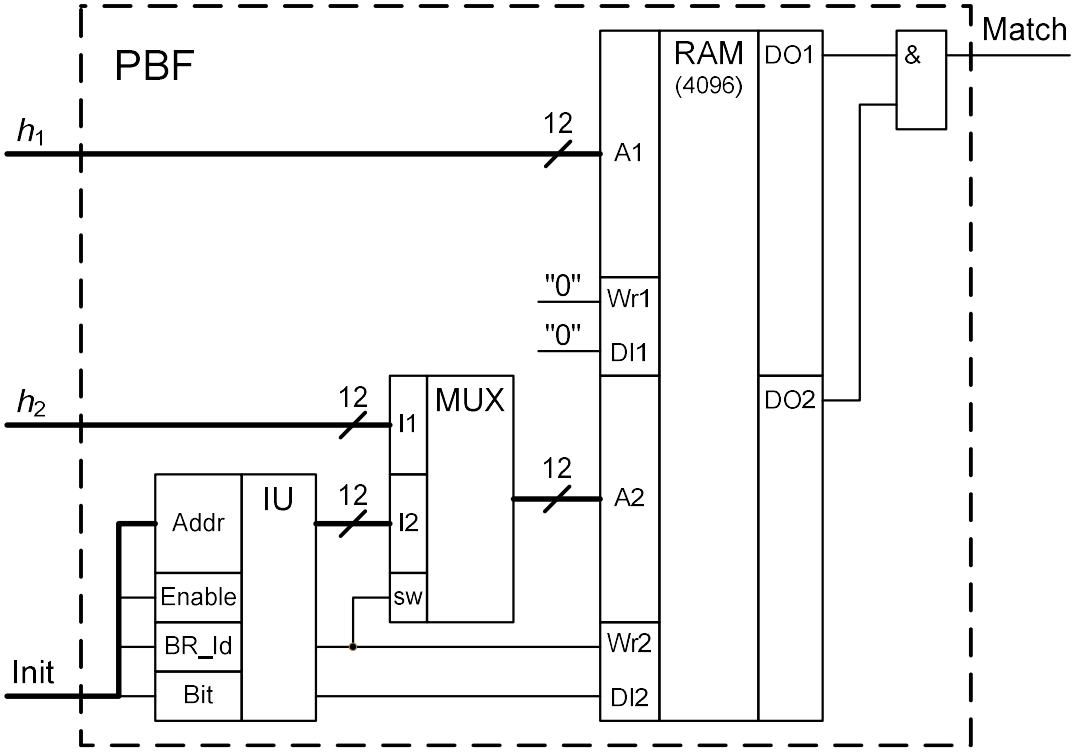
$$F_{H_3} = K\,G\,F_V = K\,G\left(\left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil - 1\right). \qquad (4.55)$$

Отже, для формулі (4.50) у випадку використання геш-функції класу H_3 ми вже маємо вираз (4.54) для величини $L_{ГФ} = L_{H_3}$ та вираз (4.55) для величини $F_{ГФ} = F_{H_3}$.

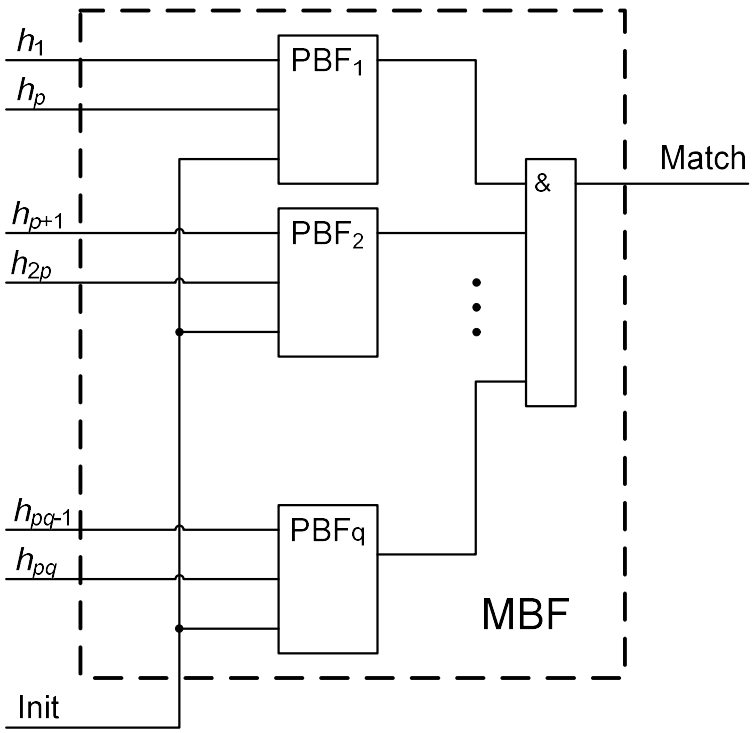
Перейдемо до розрахунків ресурсів для РБ.

4.6.2.2. Витрати ресурсів на регістр бітів ФБ

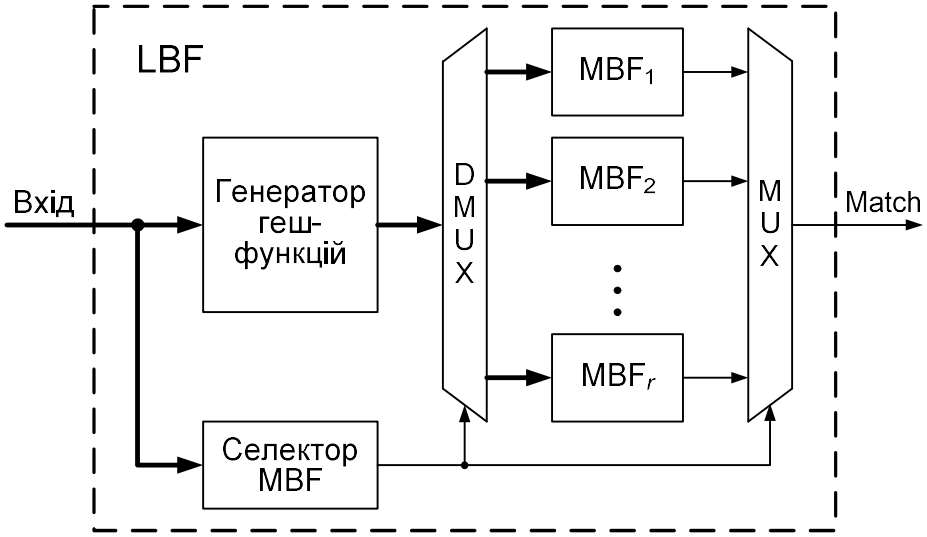
Крім ГГФ споживачем ресурсів в ФБ є також РБ. В роботі [179] (див. Fig. 5 – Fig. 7) розглянуто приклад створення такого компоненту з використанням модулю блокової пам'яті ПЛІС, що сконфігурована як двопортовий ОЗП шириною в один біт (рис. 4.6, а). Використана в прикладі модель ПЛІС дозволяє побудувати на базі BRAM регістр бітів довжиною в 4096 бітів.



а



б



в

Рис. 4.6. Схеми регістрів бітів та фільтра Блума: а) частковий ФБ (PBF); б) міні-ФБ (MBF); в) укрупнений ФБ (LBF)

В загальному випадку кількість p портів ОЗП, побудованого з блокової пам’яті різних моделей ПЛІС, може бути різною.

Дотримуючись нотації авторів схеми [179] назвемо початкову схему на одному блоці BRAM (рис. 4.6, а) частковим ФБ (Partial Bloom Filter – PBF),

маючи на увазі, що насправді то є схема тільки РБ, а не усього фільтра Блума.

Як вказувалося в підр. 3.2.1 "Основні принципи побудови та використання фільтра Блума", оптимальне співвідношення між кількістю K геш-функцій, довжиною M регістра бітів та кількістю N патернів, на розпізнавання яких запрограмований ФБ, виглядає наступним чином:

$$K = \frac{M}{N} \ln 2 . \tag{4.56}$$

В розглянутому вище прикладі $K = 2$, $M = 4096$. Тоді $N \approx 1420$ патернів. Але вірогідність помилки фільтра Блума другого роду в даному випадку буде $\rho_{пом.2} = 2^{-K} = 0{,}25$, що є неприпустимо великим значенням. Тому у розробці пропонується паралельно об’єднати кілька регістрів, наприклад, п’ять, збільшуючи тим самим кількість задіяних геш-функцій вп’ятеро, тобто до 10 одиниць. Тоді вірогідність помилки зменшиться до $\rho_{пом.2} = 2^{-10} \approx 0{,}00089$. Оскільки сумарний розмір бітового масиву при цьому збільшується пропорційно K, параметр N залишиться незмінним (≈ 1420 патернів).

На рис. 4.6, б наведено подібну модифікацію, в якої об’єднані п’ять PBF. Назвемо цю схему міні-ФБ (Mini Bloom Filter – MBF) притому, що це також лише схема РБ.

Кількість патернів, які можуть бути внесені у ФБ, побудований на базі схеми MBF, як і для PBF дорівнює 1420 одиницям. Якщо потрібно, щоб ФБ розпізнавав більше патернів, необхідно використовувати схему укрупненого ФБ (Large Bloom Filter – LBF), наведену на рис. 4.6, в. Тут паралельне об’єднання MBF (в кількості r) дозволяє збільшити кількість патернів, тому що їх геш-відбитки з виходу ГГФ рівномірно розподіляються по загальному об’єму блокової пам’яті, який в r разів перевищує розмір РБ однієї схеми MBF. Процесом розподілення керує спеціальний пристрій – селектор MBF, який також за допомогою гешування в залежності від вхідної інформації

через демультиплексор DMUX перемикає виходи геш-функцій на той чи інший модуль MBF. Результат розпізнавання подається на вихід пристрою через мультиплексор MUX.

Отже, об'єднання елементарних схем PBF у MBF дозволяє збільшити кількість геш-функцій та знизити вірогідність помилки другого роду, а MBF у LBF – збільшити кількість патернів, яки здатен розпізнавати ФБ.

Зауважимо, що на відміну від PBF і MBF схема LBF є повноцінним фільтром Блума, оскільки вже містить ГГФ.

Підрахуємо витрати ресурсів блокової пам'яті на створення РБ.

Схема PBF містить один модуль BRAM: $B_{\text{PBF}} = 1$.

Кількість схем PBF в складі MBF залежить від допустимого рівня помилок розпізнавання другого роду, який є прийнятним для створюваного ФБ. Введемо показник E – фактор хибного розпізнавання, фізичний сенс якого – ступень двійки, обернена величина до якої чисельно дорівнює дозволеної вірогідності помилки розпізнавання, яку не має перевищувати Фільтр Блума:

$$\rho_{дозв.} = 2^{-E},$$

звідки

$$E = -\lfloor \log_2 \rho_{дозв.} \rfloor . \tag{4.57}$$

Зауважимо, що оскільки вірогідність помилкового розпізнавання $\rho_{дозв.}$ – величина дуже маленька, показник E згідно (4.57) здатен приймати лише позитивні цілі значення.

При такому формулюванні показник E чисельно дорівнює кількості K геш-функцій ФБ при оптимальному співвідношенні змінних N та M (4.56). Тоді кількість q схем PBF (кожна з яких містить один модуль BRAM) в

складі MBF в залежності від кількості p портів ОЗП, який може буди синтезована у даній ПЛІС дорівнює кількості блоків BRAM в складі схеми MBF:

$$B_{\mathrm{MBF}} = q = \left\lceil \frac{E}{p} \right\rceil = \left\lceil \frac{K}{p} \right\rceil, \tag{4.58}$$

Число схем MBF в складі LBF залежить від кількості патернів, які повинен розпізнавати ФБ. Нагадаємо, що один окремий фільтр Блума (з одним ГГФ) здатен розпізнавати тільки патерни однакової довжини. Згідно запропонованої вище техніки впорядкування патернів (див. підр. 4.2 "Техніка впорядкування патернів") розмір пакету патернів однакової довжини у складі множини P всіх патернів, що розпізнаються, визначається функцією розподілу довжин $\delta(j) = \delta_j$.

З співвідношення (4.56) випливає кількість патернів, яку може вмістити ФБ, в даному випадку – схема MBF:

$$N_{\mathrm{MBF}} = \left\lfloor \frac{M}{K} \ln 2 \right\rfloor = \left\lfloor \frac{U}{E} \ln 2 \right\rfloor, \tag{4.59}$$

де U – максимальний розмір ОЗП бітової розрядності, що може бути синтезований в одному модулі блокової пам'яті ПЛІС.

Тоді кількість r схем MBF в складі LBF для розпізнавання δ_j патернів довжиною j:

$$r = \left\lceil \frac{\delta_j}{N_{\mathrm{MBF}}} \right\rceil = \left\lceil \frac{\delta_j \, E}{U \ln 2} \right\rceil. \tag{4.60}$$

Отже, загальна кількість модулів BRAM, що споживає схема LBF як функція від довжини патерну (дорівнює δ_j), фактору хибного розпізнавання E, розміру U блока пам'яті BRAM та числа p її портів, враховуючи (4.58) та (4.60):

$$B_{\mathrm{LBF}} = r\,q = \left\lceil \frac{\delta_j\,E}{U\ln 2} \right\rceil q = \left\lceil \frac{\delta_j\,E}{U\ln 2} \right\rceil \cdot \left\lceil \frac{E}{p} \right\rceil. \qquad (4.61)$$

4.6.2.3. Витрати ресурсів на допоміжні схеми для ФБ

Ієрархічна структура LBF дещо ускладнює підрахунок витрат на логіку та тригери допоміжних компоненти ФБ.

Допоміжні схеми, що входять до складу часткового ФБ – схеми PBF – призначені для керування введенням-виведенням інформації з BRAM (див. рис. 4.6, а).

Зауважимо, що процес програмування ФБ, описаний в підр. 3.2 "Використання фільтра Блума на базі геш-функцій", на практиці здійснюється програмними засобами перед побудовою апаратної схеми. І вже відомий зміст регістру бітів (після завантаження в ПЛІС синтезованої схеми ФБ) записується у відповідні пристрої блокової пам'яті. Для цього використовується один з портів ОЗП на базі BRAM (порт 2 на рис. 4.6, а), який також потрібен в процесі розпізнавання для вводу значення відповідної геш-функції. Інтерфейсний блок IU керує процесом ініціювання РБ, використовуючи сигнали вибору адресу Addr, сигнал дозволу Enable, сигнал BR_Id вибору BRAM та інформаційну лінію Bit. Мультиплексор MUX перемикає джерело інформації для другого порту ОЗП.

Отже, до складу допоміжного обладнання PBF входять: інтерфейсний блок IU (потребує три ЛТ і G тригерів, де G – розрядність геш-функцій, яка дорівнює довжині адресу блоків BRAM), мультиплексор MUX (потребує

$\lceil G / \lfloor (x-1)/2 \rfloor \rceil$ логічних таблиць) та один логічний елемент "І" (потребує одну ЛТ). Що потребує логіки:

$$L_{\text{допPBF}} = 3 + \left\lceil \frac{G}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 1 = \left\lceil \frac{G}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4 \tag{4.62}$$

та тригерів:

$$F_{\text{допPBF}} = G, \tag{4.63}$$

Зауважимо, що кількість геш-функцій залежить від розміру РБ. У випадку застосування блокової пам'яті

$$G = \log 2\ U. \tag{4.64}$$

Для побудови MBF (див. рис. 4.6, б) крім ресурсів для кожної схеми PBF потрібні ще ЛТ для створення багатовходової схеми "І":

$$L_{\text{допMBF}} = \left\lceil \frac{q-1}{x-1} \right\rceil. \tag{4.65}$$

До складу схеми LBF укрупненого ФБ (див. рис. 4.6, в) входять наступні допоміжні схеми: демультиплексор DMUX, мультиплексор MUX, селектор MBF (подібний за складом до схем обчислення геш-функції, що входять до складу ГГФ). Але загальна кількість ресурсів, що потрібні для створення всієї схеми LBF, крім витрат на згадані компоненти містить також ресурси допоміжних схем у складі MBF, які в свою чергу містять допоміжні

схеми PBF. Тому розглянемо спочатку витрати на допоміжні компоненти власне схеми LBF, а потім – витрати загалом.

– для створення демультиплексора DMUX та мультиплексора MUX потрібна лише логіка:

$$L_{\mathrm{DMUX}} = \left\lceil \frac{K\,G\,r}{\left\lfloor \frac{x - \lceil \log_2 r \rceil}{2} \right\rfloor} \right\rceil, \tag{4.66}$$

$$L_{\mathrm{MUX}} = \left\lceil \frac{r}{\left\lfloor \frac{x - \lceil \log_2 r \rceil}{2} \right\rfloor} \right\rceil. \tag{4.67}$$

Витрати на побудову селектора MBF знаходяться подібно до розрахунків ГГФ згідно виразів (4.54) та (4.55), але замість комплекту з K геш-функцій на G розряди в селекторі задіяна лише одна розрядністю $\lceil \log_2 r \rceil$:

$$L_{\mathrm{Sel}} = \lceil \log_2 r \rceil \left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil \right), \tag{4.68}$$

$$F_{\mathrm{Sel}} = \lceil \log_2 r \rceil \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil - 1. \tag{4.69}$$

Загальні витрати на допоміжні схеми, що потрібні для створення всієї схеми LBF укрупненого ФБ (див. рис. 4.6, в) складаються з витрат на

демультиплексор DMUX, на мультиплексор MUX, на селектор MBF та на допоміжні схеми у складі MBF (яких налічується r одиниць), що рекурсивно містять допоміжні схеми PBF (у кількості q одиниць):

$$R_{\text{допLBF}} = R_{\text{DMUX}} + R_{\text{MUX}} + R_{\text{Sel}} + r\left(R_{\text{допMBF}} + qR_{\text{допPBF}}\right), \tag{4.70}$$

де літерами R позначені витрати відповідних компонентів як у ЛТ, так і в тригерах.

Згідно (4.70), враховуючи (4.62), (4.63), (4.65), (4.66), (4.67) та (4.68), отримуємо:

$$L_{\text{допLBF}} = L_{\text{DMUX}} + L_{\text{MUX}} + L_{\text{Sel}} + r\left(L_{\text{допMBF}} + qL_{\text{допPBF}}\right) =$$

$$= \left\lceil \frac{KG+1}{\left\lfloor \frac{x - \lceil \log_2 r \rceil}{2} \right\rfloor} \right\rceil + \lceil \log_2 r \rceil \left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil \right) +$$

$$+ r\left(\left\lceil \frac{q-1}{x-1} \right\rceil + q\left(\left\lceil \frac{G}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4 \right) \right), \tag{4.71}$$

$$F_{\text{допLBF}} = F_{\text{Sel}} + r\,q\,F_{\text{допPBF}} = \lceil \log_2 r \rceil \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil - 1 + r\,q\,G\ . \tag{4.72}$$

Отже, маючи розрахунки витрат на всі компоненти фільтрів Блума, можемо складати ФО для конкретних реалізацій ФБ

4.6.2.4. Функція оцінки схеми повнорозмірного фільтра Блума LBF. Ресурсна складова

Підрахувати загальну кількість ресурсів всіх видів (в УЛТ), що споживає модуль розпізнавання, побудований по так званої *схемі повнорозмірного фільтру Блума* або по *схемі LBF* (на базі геш-функцій класу H_3) та блокової пам'яті BRAM). Вона складається згідно виразу (4.50) з формул (4.54), (4.55), (4.61), (4.71), (4.72):

$$R_{\mathrm{LBF}} = L_{\text{ГФ}} + L_{\text{доп}} + \alpha\,(F_{\text{ГФ}} + F_{\text{доп}}) + \beta\, B_{\text{БР}} =$$

$$= L_{H_3} + L_{\text{допLBF}} + \alpha\,(F_{H_3} + F_{\text{допLBF}}) + \beta\, B_{\mathrm{LBF}} =$$

$$= K\,G\left(\left\lceil\frac{8j}{x}\right\rceil + \left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil\right) + \left\lceil\frac{K\,G+1}{\left\lfloor\frac{x-\lceil\log_2 r\rceil}{2}\right\rfloor}\right\rceil +$$

$$+\lceil\log_2 r\rceil\left(\left\lceil\frac{8j}{x}\right\rceil + \left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil\right) + r\left(\left\lceil\frac{q-1}{x-1}\right\rceil + q\left(\left\lceil\frac{G}{\left\lfloor\frac{x-1}{2}\right\rfloor}\right\rceil + 4\right)\right) +$$

$$+\alpha\left(K\,G\left(\left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil - 1\right) + \lceil\log_2 r\rceil\left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil - 1 + r\,q\,G\right) + \beta\left\lceil\frac{\delta_j\,E}{U\ln 2}\right\rceil q\,. \quad (4.73)$$

Щоб краще розуміти, від яких змінних залежить ресурсна складова повнорозмірного фільтра Блума, нагадаємо, що кількість геш-функцій K дорівнює фактору хибного розпізнавання E, а значення змінних G, q і r, що входять до складу формулі (4.73) можна розкрити згідно виразам (4.64), (4.58) та (4.60) відповідно:

$$R_{\mathrm{LBF}} = E\log_2 U\left(\left\lceil\frac{8j}{x}\right\rceil + \left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil\right) +$$

$$+\left\lceil\frac{E\log_2 U + 1}{\left\lfloor\frac{x - \left\lceil\log_2\left\lceil\frac{\delta_j\ E}{U\ln 2}\right\rceil\right\rceil}{2}\right\rfloor}\right\rceil + \left\lceil\log_2\left\lceil\frac{\delta_j\ E}{U\ln 2}\right\rceil\right\rceil\left(\left\lceil\frac{8j}{x}\right\rceil + \left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil\right) +$$

$$+\left\lceil\frac{\delta_j\ E}{U\ln 2}\right\rceil\left(\left\lceil\frac{\left\lceil\frac{E}{p}\right\rceil - 1}{x-1}\right\rceil + \left\lceil\frac{E}{p}\right\rceil\left(\left\lceil\frac{\log_2 U}{\left\lfloor\frac{x-1}{2}\right\rfloor}\right\rceil + 4\right)\right)$$

$$+\alpha\left(E\log_2 U\left(\left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil - 1\right) +\right.$$

$$\left. +\left\lceil\log_2\left\lceil\frac{\delta_j\ E}{U\ln 2}\right\rceil\right\rceil\left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil - 1}{x-1}\right\rceil - 1 + \left\lceil\frac{\delta_j\ E}{U\ln 2}\right\rceil\left\lceil\frac{E}{p}\right\rceil\log_2 U\right) + \beta\left\lceil\frac{\delta_j\ E}{U\ln 2}\right\rceil\left\lceil\frac{E}{p}\right\rceil. \quad (4.74)$$

Проаналізуємо здобутий результат.

Вираз (4.74) є ресурсною складовою ФО для схеми LBF: $\theta_R = R_{\mathrm{LBF}}(\delta, x, p, U, \alpha, \beta, E)$. Даний вираз залежить від довжини j патернів, що

розпізнаються; функції розподілу довжин δ; а також від низки параметрів ПЛІС, що використовується: числа x входів ЛТ, кількості портів блокової пам’яті p, розміру U (в бітах) модулю BRAM блокової пам'яті та коефіцієнтів нормалізації α і β. На відміну від схеми DCAM, кількість ресурсів в схемі LBF не залежить від функцій самоподоби, що не дозволяє використовувати надмірність, притаманну базі даних сигнатур. Характерною рисою ФБ є також залежність ресурсної складової його ФО від фактору помилки E, який задає прийнятну вірогідність хибного розпізнавання другого роду ФБ. Зауважимо, що цей параметр E належить до сукупності вимог користувача системи захисту Ψ у виразі (4.5).

Слід зробити наступне важливе зауваження. Формули (4.73) та (4.74) визначають витрати на побудову ФБ, який розпізнає патерни тільки одного розміру – довжиною в j символів. Щоб розпізнавати патерни різних довжин, необхідно згідно рис. 3.15 синтезувати окремі ФБ для патернів кожної довжини, яка присутня у питомій бази даних сигнатур. Це є суттєвим недоліком підходу на базі фільтрів Блума.

Важливим (хоча й менш критичним) недоліком є також необхідність доуточнення результату розпізнавання, що видає ФБ. У випадку низької вірогідності виявлення підозрілих послідовностей символів цю процедуру можна виконувати навіть програмними засобами.

4.6.2.5. Функція оцінки схеми спрощеного фільтра Блума SBF. Ресурсна складова

У зв'язку с відносно великою ємністю модулів BRAM блокової пам'яті сучасних ПЛІС розглянута вище схема дозволяє розпізнавати багато патернів однакової довжини. У наведеному в підр. 4.6.2.2 "Витрати ресурсів на регістр бітів ФБ" прикладі співвідношення основних параметрів для блоків BRAM розміром в 4096 бітів схема ФБ з придатною вірогідністю помилки здатна розпізнавати до 1420 патернів. Але в базах даних сигнатур реальних додатків зустрічається не дуже багато великих пакетів патернів однакової довжини. В

більшості випадків об’єми пакетів невеликі, отже для їх розпізнавання в схемі ФБ достатньо мати лише одну схему MBF (міні-ФБ) та ГГФ. Такий пристрій є сенс розглядати як окремий компонент для використання в модулі розпізнавання РАСЗТЗІ. Назвемо його *спрощеною схемою на базі Фільтра Блума* або *схемою SBF* (Simplified Bloom Filter).

Оскільки РБ такої схеми містить лише q підсхем PBF та логічну схему "І", що містить $L_{\text{допMBF}}$ ЛТ (див. рис. 4.6, в), витрати на допоміжну логіку складатимуть:

$$L_{\text{доп}} = q\, L_{\text{допPBF}} + L_{\text{допMBF}}, \tag{4.75}$$

а на допоміжні тригери:

$$F_{\text{доп}} = q\, F_{\text{допPBF}}, \tag{4.76}$$

Тоді, враховуючи (4.50), (4.54), (4.55), (4.58), (4.62), (4.63), (4.65), (4.75) та (4.76) знаходимо загальну кількість ресурсів, що споживає схема SBF:

$$R_{\text{SBF}} = L_{\text{ГФ}} + L_{\text{доп}} + \alpha\,(F_{\text{ГФ}} + F_{\text{доп}}) + \beta\, B_{\text{БР}} =$$

$$= L_{H_3} + \left(q\, L_{\text{допPBF}} + L_{\text{допMBF}}\right) + \alpha\left(F_{H_3} + q\, F_{\text{допPBF}}\right) + \beta\, B_{\text{MBF}} =$$

$$= K\, G\left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil\right) + q\left(\left\lceil \frac{G}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4\right) + \left\lceil \frac{q-1}{x-1} \right\rceil +$$

$$+\alpha\left(K\,G\left(\left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil-1}{x-1}\right\rceil-1\right)+q\,G\right)+\beta\,q=$$

$$=K\,G\left(\left\lceil\frac{8j}{x}\right\rceil+(\alpha+1)\left\lceil\frac{\left\lceil\frac{8j}{x}\right\rceil-1}{x-1}\right\rceil-\alpha\right)+$$

$$+q\left(\alpha\,G+\beta+\left\lceil\frac{G}{\left\lfloor\frac{x-1}{2}\right\rfloor}\right\rceil+4\right)+\left\lceil\frac{q-1}{x-1}\right\rceil. \qquad (4.77)$$

Зауважимо, що значення розрядності геш-функцій (тобто кількість бітів, потрібних для адресації всіх комірок блоку BRAM) знаходиться як $G = \log2\ U$ лише у випадку, якщо задіяний вісь наявний об’єм BRAM. Але на практиці кількість N_j патернів довжиною j, розпізнавати які має ФБ, може бути меншою за ту, що теоретично здатен вмістити ФБ, а відповідна кількість M_j комірок BRAM, що будуть насправді використані, менша за значення U ємності блоків BRAM.

Тому, враховуючи, що кількість патернів N_j довжиною j дорівнює δ_j, з урахуванням (4.56) реально потрібна розрядність геш-функцій (адреси BRAM) дорівнює:

$$\mathrm{G}=\left\lceil\log_2 \mathrm{M}_j\right\rceil=\left\lceil\log_2\frac{K\cdot N_j}{\ln 2}\right\rceil=\left\lceil\log_2\frac{E\cdot\delta_j}{\ln 2}\right\rceil. \qquad (4.78)$$

Щоб більш наочно представити, від яких змінних залежить ресурсна складова спрощеного фільтра Блума SBF, перерахуємо формулу (4.77) з

урахуванням виразів (4.58) та (4.78) для q та G відповідно, а також того факту, що $K = E$:

$$R_{\mathrm{SBF}} = E\left\lceil \log_2 \frac{E \cdot \delta_j}{\ln 2} \right\rceil \left(\left\lceil \frac{8j}{x} \right\rceil + (\alpha + 1) \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x - 1} \right\rceil - \alpha \right) +$$

$$+ \left\lceil \frac{E}{p} \right\rceil \left(\alpha \left\lceil \log_2 \frac{E \cdot \delta_j}{\ln 2} \right\rceil + \beta + \left\lceil \frac{\left\lceil \log_2 \frac{E \cdot \delta_j}{\ln 2} \right\rceil}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4 \right) + \left\lceil \frac{\left\lceil \frac{E}{p} \right\rceil - 1}{x - 1} \right\rceil. \qquad (4.79)$$

Проаналізуємо здобутий результат.

Порівняно з формулою (4.74), яка визначає ресурсну складовою ФО для повної схеми LBF, вираз (4.79) для спрощеної схеми SBF значно простіше, але також залежить від залежить від довжини патернів j, одного параметру множини патернів – функції розподілу довжин δ і від параметрів ПЛІС, що використовується. Цей вираз також не залежить від функцій самоподоби та залежить від фактору хибного розпізнавання E, який задає прийнятну вірогідність хибного розпізнавання другого роду ФБ. Знов нагадаємо, що цей параметр E належить до сукупність вимог користувача системи захисту Ψ у виразі (4.5).

4.6.2.6. Функції оцінки схем повнорозмірного фільтра Блума LBF та спрощеного фільтра Блума SBF. Часова складова

Швидкодія схем розпізнавання на основі фільтра Блума нижча за швидкодію рішень на базі АП, хоча також є досить високою. Головну затримку в просування інформації вносить блокова пам'ять. Вибір класу геш-функцій теж впливає на швидкодію даного підходу.

Роботу схеми LBF також уповільнюють селектор MBF, демультиплексор та мультиплексор (див. рис. 4.6, в), які відсутні в схемі SBF, тому остання є більш швидкою.

Але, як і в випадку підходу на ЦК пропускна здатність схем на базі фільтра Блума може бути підвищена (за рахунок збільшення латентності) шляхом застосування конвеєризації.

З урахуванням згаданих особливостей нескладно сформувати часову складову для конкретних умов. Але слід мати на увазі потребу в додатковому механізмі уточнення результатів роботи ФБ в зв'язку з системною помилкою розпізнавання другого роду. Цей додатковий етап обробки даних в деяких обставинах може зменшити швидкодію пристрою на декілька порядків. Тому складати універсальний вираз для часової складової ФО схем на базі ФБ не є доцільним.

4.6.3. Техніка створення функцій оцінки для скінчених автоматів Ахо–Корасік

На відміну від схем на ЦК та фільтрах Блума при створенні ФО для СА-АК через їх складність безпосередній підрахунок витрат ресурсних компонентів є практично неможливим. Тому для складання функціональної залежності потрібних ЛТ, тригерів та ресурсів пам'яті від вхідного словнику патернів пропонується інший підхід, який базується на двох техніках, розглянутих нижче.

З типової схеми СА-АК (див. рис. 3.18) зрозуміло, що при створенні модуля розпізнавання на базі СА-АК кількість потрібних ресурсів складається з незначних витрат на логічні компоненти (керуючого пристрою та регістрів поточного стану і вхідного символу), але суттєвих – на запам'ятовуючий пристрій. В разі використання зовнішньої пам'яті – бортового ОЗП РУО – ще потрібна помірна кількість ресурсів для створення контролера доступу до нього (якщо до складу ПЛІС вже не входить потрібний контролер, реалізований жорсткою логікою). Отже при

використанні підходу на основі СА-АК в загальному випадку потрібні ресурси:

$$R_{\text{СА-АК}} = R_{\text{КП}} + R_{\text{КОЗП}} + R_{\text{ОЗП}}, \tag{4.80}$$

де $R_{\text{КП}}$ – регістрові ресурси реалізації скінченного автомату,

$R_{\text{КОЗП}}$ – ресурси для створення контролера доступу до зовнішнього ОЗП,

$R_{\text{ОЗП}}$ – ресурси власне запам'ятовуючого пристрою.

Як було з’ясовано у підр. 3.3.4 "Основні властивості базових схем СА-АК", головними недоліками підходу на базі СА-АК є значні витрати на пам'ять та погана масштабованість за об’ємом словнику патернів внаслідок лавиноподібного зросту запам'ятовуючого пристрою при збільшенні кількості патернів. Тому розробниками РАСЗТЗІ запропоновано багато методів та технічних рішень, які спрямовані на зменшення розміру ЗП. В підр. 3.3.1 "Алгоритм Ахо–Корасік і його реалізація у вигляді кінцевого автомату" вказано, що кількість перехресних переходів у СА-АК (яких найскладніше позбутися) залежить від властивості самоподоби словнику сигнатур. Як слідство, кількість ресурсів пам'яті, потрібних для створення модулю розпізнавання на базі СА-АК, залежить не тільки від техніки, обраної для реалізації скінченого автомату Ахо–Корасік та розміру словнику патернів, але й від властивостей патернів теж.

Натомість кількість ресурсів, що потрібні для побудови логічних схем $R_{\text{КП}}$ та контролеру $R_{\text{КОЗП}}$, як свідчить досвід, залежить від кількості та змісту патернів незначною мірою. Фактично ця залежність зводиться до зміні розрядності регістрів відповідних пристроїв, яка в свою чергу залежить лише від об’єму потрібної пам'яті RЗП.

Отже, для розрахунку потрібних апаратних ресурсів для побудови СА-АК найголовніше – визначитися з розміром ЗП, достатнім для зберігання потрібного об'єму переходів автомата. Якщо обрана техніка потребує для

своєї реалізації декілька запам'ятовуючих пристроїв, знайти достатній розмір кожного з них.

4.6.3.1. Техніка підрахунку ресурсів пам'яті для функцій оцінки СА-АК

Техніка обчислення ресурсів пам'яті $R_{ЗП}$ для ФО апаратних витрат на СА-АК заснована на включенні в її склад реалізованого програмними засобами функціонального перетворення ProcAC(P, $\mathrm{Tr_{dr}}$, $\mathrm{Tr_{cr}}$, $\mathrm{Tr_{fl}}$, $\mathrm{Tr_{rs}}$), яке для кожного заданого вхідного набору патернів швидко будує віртуальний (розташований у пам'яті комп'ютера) скінчений автомат Ахо–Корасік. На вхід даного засобу в якості змінної подається P – вхідна множина патернів, а розрахунковими величинами, що отримуються на виході, є кількості прямих ($\mathrm{Tr_{dr}}$), перехресних ($\mathrm{Tr_{cr}}$), хибних ($\mathrm{Tr_{fl}}$) та післястартових ($\mathrm{Tr_{rs}}$) переходів (див. підр. 3.3.2 "Класифікація типів переходів автомата СА-АК").

Як можна бачити зі схеми СА-АК (див. рис. 3.18), в ЗП скінченого автомату зберігаються дані двох типів: переходи автомату та вектори збігу. Тому загальний об’єм ресурсів ЗП складається з об’ємів пам'яті для переходів кожного типу та для векторів збігу:

$$B_{ЗП} = B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}} =$$

$$= f_{\mathrm{dr}}(\mathrm{Tr_{dr}}) + f_{\mathrm{cr}}(\mathrm{Tr_{cr}}) + f_{\mathrm{fl}}(\mathrm{Tr_{fl}}) + f_{\mathrm{rs}}(\mathrm{Tr_{rs}}) + f_{\mathrm{MV}}, \qquad (4.81)$$

де $f_{\mathrm{dr}}()$, $f_{\mathrm{cr}}()$, $f_{\mathrm{fl}}()$, $f_{\mathrm{rs}}()$ та f_{MV} – функціонали, які визначають, скільки ресурсів пам'яті (в байтах) потрібно для зберігання переходів кожного типу та векторів збігу відповідно.

Перші чотири функціонали у більшості випадків являють собою множення на коефіцієнт, який дорівнює кількості байтів, що використовують задіяні кодування та техніка скорочення запам’ятовуючих ресурсів. Останній функціонал в залежності від ступеню ущільнення варіюється від добутку суми перших чотирьох доданків (тобто всіх переходів) на кількість бітів, що

дорівнює числу патернів у питомому наборі, у найгіршому випадку до величини, пропорційної логарифму числа патернів у наборі для найкращих технік.

Приклади використання даної техніки підрахунку ресурсів пам'яті розглянуті нижче, у підр. 4.6.3.4 – 4.6.3.8.

4.6.3.2. Техніка підрахунку ресурсів керуючих пристроїв для функцій оцінки СА-АК

Суть запропонованої техніки полягає в тому, що створюються декілька тестових варіантів схеми пристрою, для якого потрібно знайти функціональну залежність потрібних ресурсів від розрядності шини адреси.

Ресурси керуючого пристрою (КП) та контролера зовнішнього запам’ятовуючого пристрою (КОЗП) складаються тільки з ЛТ та тригерів (4.4):

$$R_{\text{КП}} = L_{\text{КП}} + \alpha\, F_{\text{КП}}, \tag{4.82}$$

$$R_{\text{КОЗП}} = L_{\text{КОЗП}} + \alpha\, F_{\text{КОЗП}}. \tag{4.83}$$

Як вказувалося вище, ці витрати майже незмінні, але незначно залежать від розрядності зовнішньої пам’яті. Припустимо що об’єм потрібної зовнішньої пам’яті дорівнює значенню R_{RAM}, заданому в мегабайтах (1 Мбайт = 2^{20} байтів). Тоді розрядність в бітах, що потрібна для адресації такого об’єму дорівнює

$$\text{Bit}(R_{\text{RAM}}) = \lceil \log_2 R_{\text{RAM}} \rceil + 20. \tag{4.84}$$

Отже витрати на ЛТ та тригери можуть бути із задовільною точністю апроксимовані лінійними функціями від розрядності R_{RAM}:

$$L_{\text{КП}} = f_{\text{ЛКП}}(R_{\text{RAM}}) = A_{\text{ЛКП}} + B_{\text{ЛКП}}(\lceil \log_2 R_{\text{RAM}} \rceil + 20), \quad (4.85)$$

$$F_{\text{КП}} = f_{\text{ТКП}}(R_{\text{RAM}}) = A_{\text{ТКП}} + B_{\text{ТКП}}(\lceil \log_2 R_{\text{RAM}} \rceil + 20), \quad (4.86)$$

$$L_{\text{КОЗП}} = f_{\text{ЛКОЗП}}(R_{\text{RAM}}) = A_{\text{ЛКОЗП}} + B_{\text{ЛКОЗП}}(\lceil \log_2 R_{\text{RAM}} \rceil + 20), \quad (4.87)$$

$$F_{\text{КОЗП}} = f_{\text{ТКОЗП}}(R_{\text{RAM}}) = A_{\text{ТКОЗП}} + B_{\text{ТКОЗП}}(\lceil \log_2 R_{\text{RAM}} \rceil + 20), \quad (4.88)$$

де коефіцієнти $A_{\text{ЛКП}}$, $B_{\text{ЛКП}}$, $A_{\text{ТКП}}$, $B_{\text{ТКП}}$, $A_{\text{ЛКОЗП}}$, $B_{\text{ЛКОЗП}}$, $A_{\text{ТКОЗП}}$, $B_{\text{ТКОЗП}}$ – константи, які можуть бути знайдені емпіричним шляхом в результаті синтезу відповідних схем для ПЛІС з використанням інструментального засобу для декількох значень R_{RAM} та апроксимації отриманих значень витрачених ресурсів.

Розглянемо використання техніки на прикладі контролера зовнішньої пам'яті типу DDR2, синтезованого на пристрої Virtex-5 (xc5vlx20t-2ff323) за допомогою засобу Memory Interface Generator ver.2.1 (Xilinx ISE design suite 10.1) [202]. Виробник пам'яті (в даному випадку – фірма Micron Technology) не має значення, тому що пристрої динамічної пам’яті стандартизовані. В табл. 4.1 наведені дані щодо витрат ресурсів логіки та тригерів на створення контролера в залежності від об'єму ОЗП. Тут розрядність в бітах Bit(R_{RAM}), що потрібна для адресації об’єму R_{RAM}, обчислюється за формулою (4.84).

Таблиця 4.1

Розрахунки витрат ЛТ та тригерів на синтез контролера ОЗП

№	Пристрій пам’яті DDR2	Об’єм ОЗП (R_{RAM}), Мбайт	Розрядність Bit(R_{RAM})	Кількість ЛТ ($L_{\text{КОЗП}}$)	Кількість тригерів ($F_{\text{КОЗП}}$)	Помилка розрахунку $L_{\text{КОЗП}}$	Помилка розрахунку $F_{\text{КОЗП}}$
1.	MT47H32M8	32	25	982	1117	0	0
2.	MT47H64M8	64	26	991	1128	–1	0
3.	MT47H128M8	128	27	997	1139	+1	0
4.	MT47H256M8	256	28	1006	1150	0	0

Підставивши R_{RAM} та $L_{\text{КОЗП}}$ у вираз (4.87), можна знайти приблизні значення $A_{\text{ЛКОЗП}}$ = 782 та $B_{\text{ЛКОЗП}}$ = 8, які дозволяють обчислити $L_{\text{КОЗП}}$ по заданому R_{RAM} згідно формулі (4.87) з помилкою не більше однієї ЛУТ (< 0,1%).

Так само можна використати вираз (4.88) для тригерів, але в даному випадку питомі значення отримуються точні: $A_{\text{ТКОЗП}}$ = 842, $B_{\text{ТКОЗП}}$ = 11.

Наведемо формальний опис створюваної техніки підрахунку ресурсів керуючих пристроїв для ФО СА-АК.

Крок 1. Створюються та синтезуються за допомогою відповідного інструментального засобу декілька варіантів схеми пристрою для різних значень розрядності Bit_i.

Крок 2. Для кожного варіанта знаходяться кількості R_i спожитих ресурсів (окремо для кожного виду ресурсів – ЛТ та тригерів).

Крок 3. Окремо для ЛТ та тригерів складаються системи лінійних алгебраїчних рівнянь (СЛАР) виду $R_i = A_i + B_i\ \text{Bit}_i$, в результаті вирішення яких знаходяться питомі коефіцієнти A_i та B_i. Якщо система рівнянь виявиться несумісною, застосовується якийсь відомий апроксимаційний метод.

Дана техніка використовується як для побудови контролера зовнішнього ОЗП (як було розглянуто у прикладі вище), так і для логічних компонентів модулю розпізнавання (регістрів, що реалізують функціональність СА, та пристрою керування).

Нижче наведено приклади використання розглянутих технік для створення ФО для схем: СА-АК із зовнішньою пам'яттю ACRAM, СА-АК із блоковою пам'яттю ACBRAM та СА-АК, побудованого за конвеєрною схемою PipACBRAM.

4.6.3.3. Функція оцінки СА-АК із зовнішньою пам'яттю ACRAM. Ресурсна складова

Згідно (4.80) скінчений автомат із зовнішньою пам'яттю ACRAM (див. рис. 3.18) потребує ресурси ПЛІС для створення керуючого пристрою та регістрів $R_{\text{КП}}$, контролера зовнішнього запам’ятовуючого пристрою $R_{\text{КОЗП}}$, а також ресурси R_{RAM} зовнішнього ОЗП (бортового запам’ятовуючого пристрою РУО):

$$\theta^{R}_{\text{ACRAM}} = R_{\text{ACRAM}} = R_{\text{КП}} + R_{\text{КОЗП}} + R_{\text{RAM}} . \qquad (4.89)$$

Оскільки ресурси зовнішнього ОЗП містять тільки один вид ресурсів – бортову пам’ять РУО *M*РУО з урахуванням (4.82) і (4.83) та згідно (4) отримуємо:

$$\theta^{R}_{\text{ACRAM}} = L_{\text{КП}} + \alpha F_{\text{КП}} + L_{\text{КОЗП}} + \alpha F_{\text{КОЗП}} + \gamma M_{\text{РУО}} =$$

$$= L_{\text{КП}} + L_{\text{КОЗП}} + \alpha \left(F_{\text{КП}} + F_{\text{КОЗП}} \right) + \gamma M_{\text{РУО}} . \qquad (4.90)$$

Ресурси КП та КОЗП знаходяться за допомогою техніки підрахунку ресурсів керуючих пристроїв для ФО СА-АК згідно формул (4.82), (4.83), (4.85) – (4.88) (див. підр. 4.6.3.2 "Техніка підрахунку ресурсів керуючих пристроїв для функцій оцінки СА-АК").

Об'єм ресурсів *M*РУО зовнішнього ОЗП знаходяться за допомогою Техніки підрахунку ресурсів пам'яті для ФО СА-АК (див. підр. 4.6.3.1 "Техніка підрахунку ресурсів пам'яті для функцій оцінки СА-АК"). При цьому відповідно до обраної техніки модифікації СА-АК програмними засобами розраховуються кількості переходів автомату (Tr_{dr}, Tr_{cr}, Tr_{fl}, Tr_{rs}) та функціонали $f_{\text{dr}}()$, $f_{\text{cr}}()$, $f_{\text{fl}}()$, $f_{\text{rs}}()$ і f_{MV}, після чого за формулою (4.81) обчислюється об’єм зовнішнього ЗП.

Як бачимо, схема ACRAM є типовим об’єктом для застосування згаданих технік для випадку коли ЗП скінченного автомату розташовується в бортовому ОЗП РУО, тобто *М*РУО=*М*ОЗП.

Отже згідно (4.81) та (4.90) отримуємо остаточний вираз для ресурсної складової ФО СА-АК із зовнішньою пам'яттю ACRAM:

$$\theta^{R}_{\text{ACRAM}} = A_{\text{ЛКП}} + A_{\text{ЛКОЗП}} + \left(B_{\text{ЛКП}} + B_{\text{ЛКОЗП}}\right)\left(\lceil \log_2 M_{\text{РУО}} \rceil + 20\right) +$$

$$+ \alpha \left(A_{\text{ТКП}} + A_{\text{ТКОЗП}} + \left(B_{\text{ТКП}} + B_{\text{ТКОЗП}}\right)\left(\lceil \log_2 M_{\text{РУО}} \rceil + 20\right)\right) +$$

$$+ \gamma \left(f_{\text{dr}}\left(\text{Tr}_{\text{dr}}\right) + f_{\text{cr}}\left(\text{Tr}_{\text{cr}}\right) + f_{\text{fl}}\left(\text{Tr}_{\text{fl}}\right) + f_{\text{rs}}\left(\text{Tr}_{\text{rs}}\right) + f_{\text{MV}}\right). \qquad (4.91)$$

Аналізуючи здобутий результат, звернемо увагу на те, що ресурсна складова схеми ACRAM (4.91) залежить, з одного боку, від набору знайдених емпіричним шляхом констант, на які майже не впливають властивості словнику патернів, що розпізнаються, з іншого боку – від кількості переходів автомата, які, навпаки, залежать від набору патернів *P* дуже сильно, зокрема, від такої його властивості, як друга функція самоподоби ν.

4.6.3.4. Функція оцінки СА-АК із зовнішньою пам'яттю ACRAM. Часова складова

Об’єм бортових ОЗП сучасних реконфігуровних прискорювачів сягає десятків гігабайтів, що дозволяє розпізнавати за допомогою СА-АК словники патернів великого обсягу. Але відносно низка (у порівнянні з внутрішніми швидкостями на кристалі) пропускна здатність каналу обміну між ПЛІС та зовнішньою пам’яттю (яка до того ж майже завжди є однопортовою) драматично обмежує швидкодію модулю розпізнавання.

У випадку використання підходу на СА-АК, реалізованого схемою ACRAM, часова складова ФО, задіяних в алгоритмах оптимізації, може бути складена як добуток затримки доступу к зовнішньої пам’яті *T*RAM та

середньої кількості звернень до ОЗП за такт роботи скінченного автомату *N*RAM, якої вимагає задіяна техніка:

$$\theta^{T}_{\text{ACRAM}} = T_{\text{ACRAM}} = T_{\text{RAM}} \cdot N_{\text{RAM}} \,. \tag{4.92}$$

Значення *T*RAM є постійною величиною, яка визначається виключно характеристиками бортового ОЗП, і тому не залежить ні від розміру, ні від складу словнику патернів. Її значення береться з технічного опису відповідного електронного виробу – модулю DIMM/SoDIMM, або безпосередньо мікросхем ОЗП, якщо пам’ять інтегрована в друковану плату РУО. При цьому слід враховувати, що звернення автомату Ахо–Корасік до комірок пам’яті практично завжди відбувається не послідовно, а випадковим чином у довільному порядку, отже необхідно використовувати відповідні технічні показники.

Натомість значення *N*RAM в загальному випадку залежить від властивостей набору патернів. Це обумовлено тим фактом, що, з одного боку, різни типи переходів в автоматі в загальному випадку кодуються різною кількістю байтів, з іншого, алгоритми реалізації переходів також відрізняються. Тому середню кількість звернень до ОЗП можна обчислити як

$$N_{RAM} = \frac{N_{\text{dr}} R_{\text{dr}} + N_{\text{cr}} R_{\text{cr}} + N_{\text{fl}} R_{\text{fl}} + N_{\text{rs}} R_{\text{rs}} + N_{\text{MV}} R_{\text{MV}}}{5 R_{\text{ОЗП}}}, \tag{4.93}$$

де N_{dr}, N_{cr}, N_{fl}, N_{rs} та N_{MV} – кількості звернень до ОЗП за такт роботи СА у випадку здійснення прямого, перехресного, хибного, післястартового переходу та отримання коду прийнятного стану відповідно,

R_{dr}, R_{cr}, R_{fl}, R_{rs} та R_{MV} – об’єми пам'яті, які виділені відповідно під прямі, перехресні, хибні, післястартові переходи та вектори збігу, що знаходяться згідно (4.81),

$R_{\text{ОЗП}}$ – сумарний об’єм пам’яті, потрібної для всіх переходів автомату та векторів збігу.

Отже, остаточний результат для часової складової ФО схеми СА-АК із зовнішньою пам'яттю ACRAM згідно (4.92) та (4.93):

$$\theta^{T}_{\text{ACRAM}} = T_{\text{ACRAM}} = T_{\text{RAM}} \frac{N_{\text{dr}}R_{\text{dr}} + N_{\text{cr}}R_{\text{cr}} + N_{\text{fl}}R_{\text{fl}} + N_{\text{rs}}R_{\text{rs}} + N_{\text{MV}}R_{\text{MV}}}{5R_{\text{ОЗП}}}. \quad (4.94)$$

4.6.3.5. Функція оцінки СА-АК із блоковою пам'яттю ACBRAM. Ресурсна складова

Використання в схемі модуля розпізнавання вбудованої в ПЛІС блокової пам’яті надає низку переваг. По-перше, швидкодія BRAM вище порівняно з бортовим ОЗП. По-друге, вона не потребує додаткового контролеру пам’яті: сформована керуючим пристроєм СА адреса може бути безпосередньо подана на адресний вхід блокової пам’яті. По-третє, гнучкість BRAM дозволяє формувати з неї пристрої пам’яті довільної конфігурації, що сприяє більш ефективному використанню ресурсів. По-четверте пам'ять BRAM зазвичай багатопортова, що дозволяє прискорити схему за рахунок паралельного підключення декількох автоматів (див. підр. 3.3.9 "Розпаралелювання").

Головним недоліком BRAM є її відносно висока вартість і, як наслідок, значно менший об’єм порівняно із зовнішньою пам’яттю. Але цей показник постійно зростає з розвитком ПЛІС. В сучасних пристроях програмованої логіки сумарна ємність BRAM складає десятки мегабайтів (див. табл. 1.4 "Основні характеристики деяких РУО" в підр. 1.5.2 "Існуючі реалізації"). З іншого боку, як вказувалось раніше, розробниками запропоновано багато модифікацій та технік щодо скорочення об’єму пам’яті, потрібної для схем на базі СА-АК.

Отже при побудові схеми ACBRAM не потрібно витрачати ресурси на контролер зовнішньої ОЗП. Натомість, оскільки пам’ять BRAM поділена між багатьма окремими блоками, для її об’єднання в єдиний адресний простір необхідно синтезувати додатковий дешифратор DC, а також мультиплексор MUX для об'єднання виходів модулів BRAM (рис. 4.7). Як буде показано нижче, дешифратор потребує тільки логічні ресурси, а мультиплексор – як логіку, так і тригери.

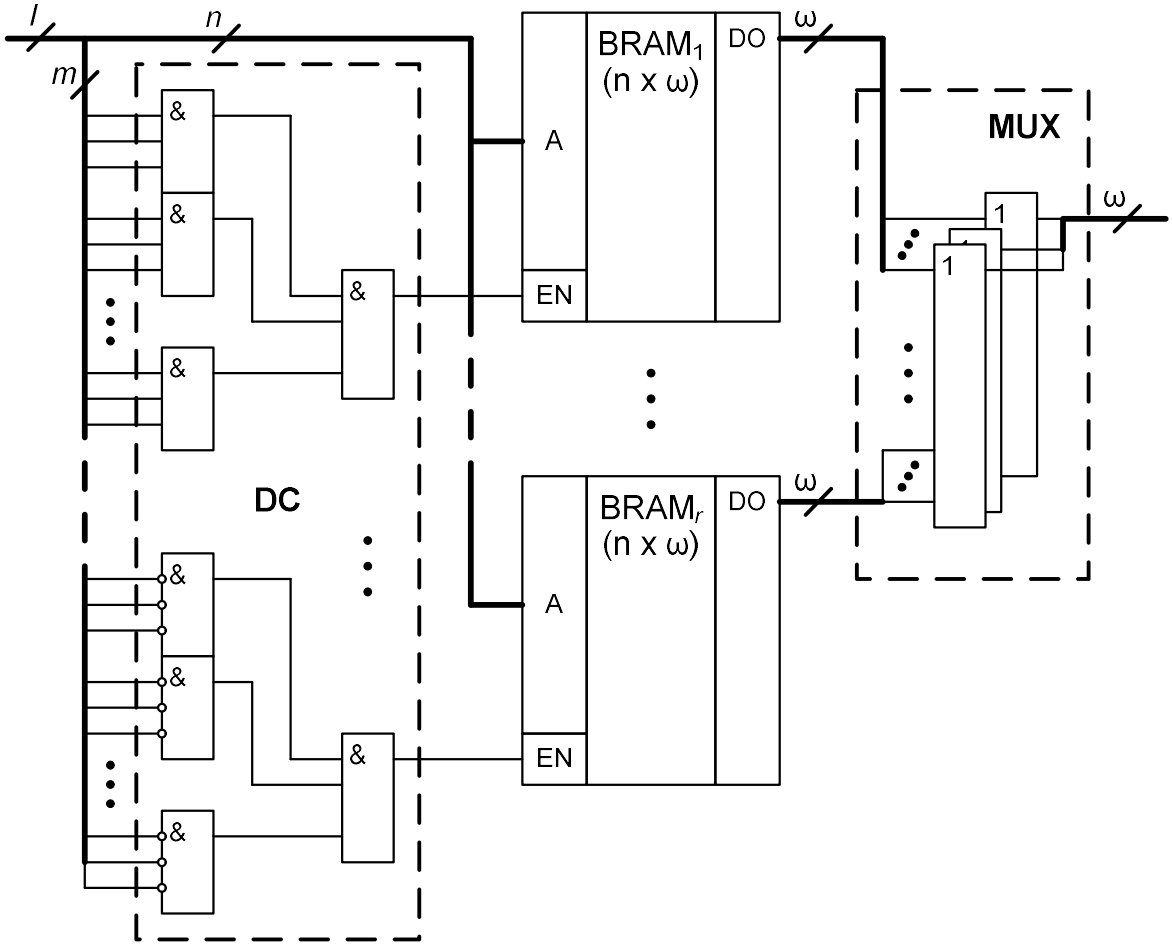


Рис. 4.7. Схема об'єднання блокової пам'яті в єдиний пристрій ОЗП

Незважаючи на те, що в процесі функціонування скінченного автомата в пам'яті переходів виконується тільки операція читання даних, ці данні необхідно занести в пам'ять перед початком роботи. Для цього ще потрібна схема запису WR, яка також потребує як логіку, так і тригери, хоча і в незначної кількості.

Отже, згідно (4) та (4.82) ресурсна складова ФО для схеми ACBRAM:

$$\theta^{R}_{\text{ACBRAM}} = R_{\text{ACBRAM}} = R_{\text{КП}} + R_{\text{DC}} + R_{\text{MUX}} + R_{\text{BRAM}} =$$

$$= L_{\text{КП}} + L_{\text{DC}} + L_{\text{MUX}} + L_{\text{WR}} + \alpha\left(F_{\text{КП}} + F_{\text{MUX}} + F_{\text{WR}}\right) + \beta\,\Xi(B), \qquad (4.95)$$

де LКП – кількість ЛТ, потрібних для побудови керуючого пристрою скінченого автомату,

LDC – кількість ЛТ, потрібних для побудови дешифратора,

L_{MUX} та F_{MUX} – відповідно кількість ЛТ та тригерів, потрібних для побудови мультиплексора,

L_{WR} та F_{WR} – відповідно кількість ЛТ та тригерів, потрібних для побудови схеми запису,

B – кількість потрібних ресурсів блокової пам’яті у Кбітах,

$\Xi(B)$ – функція перерахування об'єму потрібної блокової пам'яті у потрібну кількість блоків BRAM, оскільки ресурси блокової пам'яті на відміну від зовнішнього ОЗП підраховуються як кількість блоків BRAM, а не в одиницях об’єму пам'яті.

Величини LКП та FКП знаходяться так само, як і для схеми ACRAM – за допомогою Техніки підрахунку ресурсів керуючих пристроїв для ФО СА-АК (див. підр. 4.6.3.2 "Техніка підрахунку ресурсів керуючих пристроїв для функцій оцінки СА-АК").

Об’єм потрібної блокової пам’яті B (у Мбітах) як і для схеми ACRAM обчислюється шляхом підстановки у формулу (4.81) ресурсів блокової пам’яті, потрібної для здійснення кожного типу переходів автомата:

$$B = B_{\text{dr}} + B_{\text{cr}} + B_{\text{fl}} + B_{\text{rs}} + B_{\text{MV}}. \qquad (4.96)$$

де B_{dr}, B_{cr}, B_{fl}, B_{rs} та B_{MV} – об’єми блокової пам'яті (у Мбітах), які виділені відповідно під прямі, перехресні, хибні, післястартові переходи та вектори збігу.

Кількість r потрібних блоків BRAM знаходиться як:

$$r = \left\lceil \frac{B}{N} \right\rceil = \left\lceil \frac{B}{1024\left(\frac{U}{\omega}\right)} \right\rceil. \tag{4.97}$$

де N – кількість комірок даних в одному блоку BRAM

U – об'єм пам'яті (у Кбітах) в одному блоку BRAM (не враховуючі біти парності, якщо вони використовуються),

ω – ширина (розрядність) даних, що зберігаються в BRAM, в бітах, яка залежить виключно від застосованої модифікації СА-АК.

В наведеній на рис. 4.7 схемі об'єднання декількох модулів блокової пам'яті в єдиний пристрій ОЗП, потрібний для роботи СА-АК, розрядність питомого пристрою l складається з розрядності $n = \lfloor \log_2 N \rfloor$ адресного простору модулю BRAM та $m = \lceil \log_2 r \rceil$ додаткових розрядів, потрібних для однозначного вибору потрібного модулю. Дешифратор складається з r однакових схем "І" на m входів, причому на кожен вхід сигнал подається або прямо, або з інверсією в залежності від номера блоку BRAM. Оскільки число блоків BRAM навіть у найсучасніших ПЛІС не перевищує декількох тисяч ($m < 12$), а число входів ЛТ в них не менше за 6 (а у більш старих ПЛІС з 4-входовими ЛТ кількість BRAM-блоків менше сотні ($m < 7$)), то для кожної підсхеми дешифратора достатньо двох ярусів логічних елементів, а загальна кількість ЛТ, потрібних для створення всього дешифратора:

$$L_{DC} = r\left(\left\lceil \frac{m}{x} \right\rceil + 1\right). \tag{4.98}$$

де x – число входів ЛТ.

Конфігурація блокової пам'яті в ПЛІС дозволяє керувати подачею даних на вихідні лінії. Тому мультиплексор MUX зводиться до комплекту з ω r-входових схем "АБО". Оскільки кількість r задіяних модулів BRAM може сягати досить великих значень (декількох тисяч у граничному випадку), схеми "АБО" потрібно будувати у вигляді конвеєру, аналогічно тому, як це робилось в схемах на цифрових компараторах для багатовходових схем "І" (див. рис. 3.3 у підр. 3.1.4 "Особливості реалізації схеми з цифрових компараторів на ПЛІС"). Тому кількість ЛТ та тригерів для схеми ACBRAM обчислюється за формулами, подібними (4.11) та (4.18):

$$L_{\mathrm{MUX}} = \omega \left\lceil \frac{r-1}{x-1} \right\rceil, \tag{4.99}$$

$$F_{\mathrm{MUX}} = \omega \left(\left\lceil \frac{r-1}{x-1} \right\rceil - 1 \right). \tag{4.100}$$

Схема занесення даних в пам'ять СА-АК для ACBRAM будується аналогічно такої у Фільтрі Блума (див. у підр. 4.6.2.4 "Функція оцінки схеми повнорозмірного фільтра Блума LBF. Ресурсна складова" рис. 4.6, а та коментарі щодо підрахунку ресурсів міні-ФБ – MBF). Тому кількість ЛТ та тригерів для схеми запису WR знаходяться так само, як вони враховувалися у формулах розрахунку допоміжного обладнання схеми PBF (4.62) та (4.63):

$$L_{\mathrm{WR}} = \left\lceil \frac{n}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4, \tag{4.101}$$

$$F_{\mathrm{WR}} = n. \tag{4.102}$$

Отже, підставляючи (4.96) – (4.102) у (4.95) з урахуванням (4.85) та (4.86), отримуємо загальну кількість УЛТ, потрібних для побудови всієї схеми ACBRAM:

$$R_{\mathrm{ACBRAM}} = L_{\text{КП}} + L_{\mathrm{DC}} + L_{\mathrm{MUX}} + L_{\mathrm{WR}} + \alpha\left(F_{\text{КП}} + F_{\mathrm{MUX}} + F_{\mathrm{WR}}\right) + \beta\,\Xi(B) =$$

$$= A_{\text{ЛКП}} + B_{\text{ЛКП}}\left(\lceil \log_2 B \rceil + 20\right) + r\left(\left\lceil \frac{m}{x} \right\rceil + 1\right) + \omega \cdot \left\lceil \frac{r-1}{x-1} \right\rceil + \left\lceil \frac{n}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4 +$$

$$+ \alpha\left(A_{\text{ТКП}} + B_{\text{ТКП}}\left(\lceil \log_2 B \rceil + 20\right) + \omega \cdot \left(\left\lceil \frac{r-1}{x-1} \right\rceil - 1\right) + n \right) + \beta\,\Xi(B) =$$

$$= A_{\text{ЛКП}} + B_{\text{ЛКП}}\left(\lceil \log_2 \left(B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}}\right) \rceil + 20\right) +$$

$$+ \left\lceil \frac{B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}}}{1024\left(\frac{U}{\omega}\right)} \right\rceil \left(\left\lceil \frac{\left\lceil \log_2 \left\lceil \frac{B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}}}{1024\left(\frac{U}{\omega}\right)} \right\rceil \right\rceil}{x} \right\rceil + 1 \right) +$$

$$+\omega\left\lceil\frac{\left\lceil\frac{B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}}{1024\left(\frac{U}{\omega}\right)}\right\rceil-1}{x-1}\right\rceil+$$

$$+\alpha\left(A_{\text{ТКП}}+B_{\text{ТКП}}\left(\left\lceil\log_2\left(B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}\right)\right\rceil+20\right)+\right.$$

$$+\omega\left(\left\lceil\frac{\left\lceil\frac{B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}}{1024\left(\frac{U}{\omega}\right)}\right\rceil-1}{x-1}\right\rceil-1\right)+\left\lfloor\log_2\left(\frac{U}{\omega}\right)+10\right\rfloor\Bigg)+$$

$$+\beta\,\Xi\left(B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}\right). \qquad (4.103)$$

Аналізуючи здобутий результат, нагадаємо, що у виразі (4.103):

$A_{\text{ЛКП}}$, $B_{\text{ЛКП}}$, $A_{\text{ТКП}}$, $B_{\text{ТКП}}$ – коефіцієнти лінійної апроксимації ресурсів керуючого пристрою СА-АК, які знаходяться емпірично шляхом вирішення СЛАР,

x – число входів ЛТ використаної ПЛІС,

α,β – коефіцієнти нормалізації ресурсів різного типу відносно логічного ресурсу (логічних таблиць),

$\Xi(B)$ – функція перерахування об'єму потрібної блокової пам'яті у кількість блоків BRAM.

Як і у випадку з ACRAM, ресурсна складова схеми ACBRAM (4.97) залежить, з одного боку, від набору знайдених емпіричним шляхом констант, на які майже не впливають властивості словнику патернів, що розпізнаються, з іншого боку – від кількості переходів автомата, які, навпаки, дуже залежать від набору патернів P, зокрема, від такого його параметра, як друга функція самоподоби ν.

4.6.3.6. Функція оцінки СА-АК із блоковою пам'яттю ACBRAM. Часова складова

У випадку застосування схеми ACBRAM, що використовує блокову пам'ять, часова складова ФО як і у випадку ACRAM є добутком затримки доступу к пам’яті T_{BRAM} та середньої кількості звернень до ОЗП за такт роботи скінченного автомату N_{BRAM}:

$$\theta^{T}_{\mathrm{ACBRAM}} = T_{\mathrm{ACBRAM}} = T_{\mathrm{BRAM}} \cdot N_{\mathrm{BRAM}}. \tag{4.104}$$

Другий співмножник тут обчислюється так само, як и для ACRAM, по формулі подібній (4.93):

$$N_{\mathrm{BRAM}} = \frac{N_{\mathrm{dr}}B_{\mathrm{dr}} + N_{\mathrm{cr}}B_{\mathrm{cr}} + N_{\mathrm{fl}}B_{\mathrm{fl}} + N_{\mathrm{rs}}B_{\mathrm{rs}} + N_{\mathrm{MV}}B_{\mathrm{MV}}}{5\left(B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}}\right)}, \tag{4.105}$$

де N_{dr}, N_{cr}, N_{fl}, N_{rs} та N_{MV} – кількості звернень до BRAM за такт роботи СА у випадку здійснення прямого, перехресного, хибного, післястартового переходу та отримання коду прийнятного стану відповідно,

B_{dr}, B_{cr}, B_{fl}, B_{rs} та B_{MV} – об’єми блокової пам'яті (у Мбітах), які виділені відповідно під прямі, перехресні, хибні, післястартові переходи та вектори збігу.

Перший співмножник у (4.104) також має сенс часу доступу до пристрою пам'яті, але на відміну від зовнішнього ОЗП приймає суттєво

менші значення в абсолютних величинах. Для випадку BRAM цей час складається (рис. 4.7) із затримки дешифратора T_{DC} (не більше одного такту сінхросерії) та мультиплексора T_{MUX} (декілька тактів). Власне схема ОЗП вносить незначну затримку T_{BR} в один такт.

Затримка в схемі MUX обумовлена її конвеєрною структурою. Кількість щаблів конвеєру k залежить від кількості r задіяних блоків BRAM та числа x входів ЛТ у вигляді наступної залежності: $k = \lceil \log_x r \rceil$. При $x = 6$ число щаблів конвеєру не може перевищувати 5 для будь-якої з існуючих ПЛІС. А для $x = 8$ це число буде не більше за 4.

Отже навіть у найгіршому випадку час доступу до пам'яті BRAM для схеми ACBRAM:

$$T_{\mathrm{BRAM}} = T_{\mathrm{DC}} + T_{\mathrm{BR}} + T_{\mathrm{MUX}} \le (1+1+5)\, T_{CLK} = 7 \cdot T_{CLK}\,, \tag{4.106}$$

але для типових випадків ця складова дорівнює (4..5) T_{CLK}.

Отже, згідно (4.97), (4.104), (4.105) і (4.106) часова складова ФО для схеми СА-АК із блоковою пам'яттю ACBRAM:

$$\theta^{T}_{\mathrm{ACBRAM}} = T_{\mathrm{BRAM}} \cdot N_{\mathrm{BRAM}} = \left(2 + \left\lceil \log_x \left\lceil \frac{(B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}})}{1024\left(\dfrac{U}{\omega}\right)} \right\rceil \right\rceil \right) \times$$

$$\times \left(\frac{N_{\mathrm{dr}} B_{\mathrm{dr}} + N_{\mathrm{cr}} B_{\mathrm{cr}} + N_{\mathrm{fl}} B_{\mathrm{fl}} + N_{\mathrm{rs}} B_{\mathrm{rs}} + N_{\mathrm{MV}} B_{\mathrm{MV}}}{5(B_{\mathrm{dr}} + B_{\mathrm{cr}} + B_{\mathrm{fl}} + B_{\mathrm{rs}} + B_{\mathrm{MV}})} \right). \tag{4.107}$$

Порівняно зі схемою ACRAM (4.94) остаточний результат для схеми ACBRAM (4.107) за фактичними значеннями параметрів демонструє кращі характеристики швидкодії.

4.6.3.7. Функція оцінки СА-АК за конвеєрною схемою PipACBRAM. Ресурсна складова

Витрати на конвеєрну модифікацію скінченного автомату Ахо-Корасік з використанням блокової пам'яті PipACBRAM, розраховуються майже так само, як и на схему ACBRAM (4.103). Відмінність полягає лише у наявності кількох КП для кожного шаблю конвеєра та для залишкового автомату (див. рис. 3.19) та в розбитті єдиного запам’ятовуючого пристрою для зберігання переходів скінченного автомату на кілька локальних пристроїв пам'яті щаблів конвеєру. Тому ресурсна складова ФО для цієї схеми складається з витрат на H+1 керуючий пристрій, витрат на H+1 комплект допоміжних схем організації ОЗП з блокової пам'яті та витрат на власне, блокову пам'ять, об'єм якої обчислюється по відповідному алгоритму:

$$R_{\text{PipACBRAM}} = (H+1)\left(L_{\text{КП}} + L_{\text{DC}} + L_{\text{MUX}} + L_{\text{WR}} + \alpha\left(F_{\text{КП}} + F_{\text{MUX}} + F_{\text{WR}}\right)\right) +$$

$$+ \beta\,\Xi(B) = (H+1)\left(A_{\text{ЛКП}} + B_{\text{ЛКП}} + \left(\left\lceil \log_2\left(B_{\text{dr}} + B_{\text{cr}} + B_{\text{fl}} + B_{\text{rs}} + B_{\text{MV}}\right)\right\rceil + 20\right) +$$

$$+ \left\lceil \frac{B_{\text{dr}} + B_{\text{cr}} + B_{\text{fl}} + B_{\text{rs}} + B_{\text{MV}}}{1024\left(\frac{U}{\omega}\right)} \right\rceil \left(\left\lceil \frac{\left\lceil \log_2 \left\lceil \frac{B_{\text{dr}} + B_{\text{cr}} + B_{\text{fl}} + B_{\text{rs}} + B_{\text{MV}}}{1024\left(\frac{U}{\omega}\right)} \right\rceil \right\rceil}{x} \right\rceil + 1 \right) +$$

$$+\omega\left\lceil\frac{\left\lceil\frac{B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}}{1024\left(\frac{U}{\omega}\right)}\right\rceil-1}{x-1}\right\rceil+$$

$$+\alpha\left(A_{\text{ТКП}}+B_{\text{ТКП}}\left(\left\lceil\log_2\left(B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}\right)\right\rceil+20\right)+\right.$$

$$\left.+\omega\left(\left\lceil\frac{\left\lceil\frac{B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}}{1024\left(\frac{U}{\omega}\right)}\right\rceil-1}{x-1}\right\rceil-1\right)+\left\lfloor\log_2\left(\frac{U}{\omega}\right)+10\right\rfloor\right)\right)+$$

$$+\beta\,\Xi\left(B_{\mathrm{dr}}+B_{\mathrm{cr}}+B_{\mathrm{fl}}+B_{\mathrm{rs}}+B_{\mathrm{MV}}\right),\qquad(4.108)$$

де H – кількість рівнів СА, що підлягають конвеєризації,

$A_{\text{ЛКП}}$, $B_{\text{ЛКП}}$, $A_{\text{ТКП}}$, $B_{\text{ТКП}}$ – коефіцієнти лінійної апроксимації ресурсів керуючих пристроїв,

B_{dr}, B_{cr}, B_{fl}, B_{rs} та B_{MV} – об’єми блокової пам'яті (у Мбітах), які виділені відповідно під прямі, перехресні, хибні, післястартові переходи та вектори збігу.

U – розмір (у Кбітах) одного блока BRAM (не враховуючі біти парності, якщо вони використовуються),

ω – ширина (розрядність) даних, що зберігаються в BRAM, в бітах,

x – число входів ЛТ використаної ПЛІС,

α, β – коефіцієнти нормалізації ресурсів різного типу відносно логічного ресурсу (логічних таблиць),

$\Xi(B)$ – функція перерахування об'єму потрібної блокової пам'яті у кількість блоків BRAM.

Незважаючи на те, що вираз (4.108) виглядає складнішім порівняно з виразом для ACBRAM (4.103), за рахунок суттєвого скорочення перехресних переходів та, як наслідок, потреби в блоках пам'яті, схема PipACBRAM споживає значно менше ресурсів, ніж схема ACBRAM.

4.6.3.8. Функція оцінки СА-АК за конвеєрною схемою PipACBRAM. Часова складова

Часова складова ФО для схеми СА-АК за конвеєрною схемою PipACBRAM знаходиться з тим же виразом (4.107), що й для схеми ACBRAM. Але за рахунок значного скорочення перехресних переходів фактичні значення тактової частоти виявляються меншими. З іншого боку, латентність даної модифікації зростає на H тактів, але для пристроїв РАСЗТЗІ це збільшення не є важливим.

4.6.4. Техніка створення функцій оцінки для двокаскадної схеми на базі геш-функції, ПЗП та цифрових компараторів

У підр. 5.4.1 "Структура методу МПсКс" буде розглянуто метод побудови структури модуля розпізнавання РАСЗТЗІ, що використовує каскадування відомих технологій. В даному підрозділі розглянуто приклад реалізації даного методу у вигляді двокаскадної схеми, що використовує геш-функцію, постійний запам'ятовуючий пристрій та цифрові компаратори.

Узагальнену структуру такої схеми наведено на рис. 4.8.

Регістри RG_1, RG_2, … , RG_L, RG_{L+1}, … , RG_{Mmax} створюють конвеєр (розрядністю 8 бітів для байтового кодування), по якому просуваються вхідні символи. Зміст перших L щаблів конвеєра. тобто префікси потенційних патернів подаються на генератор геш-функції (ГГф), який формує на своєму виході адресу в постійному запам’ятовуючому пристрої, що зберігає суфікси

патернів. Значення ймовірного патерну з виходу ПЗП подається на лінійку з ($m_{\max} - L$) компараторів CMP одночасно з виходами щаблів RG_{L+1}, … , RG_{Mmax} вхідного конвеєру. Якщо перші L та наступні ($m_j - L$) байтів вхідного потоку виявляються такими, що належать до одного з питомих патернів (довжиною m_j), на виході багатовходової схеми "І", що об'єднує виходи компараторів, виникає сигнал збігу.

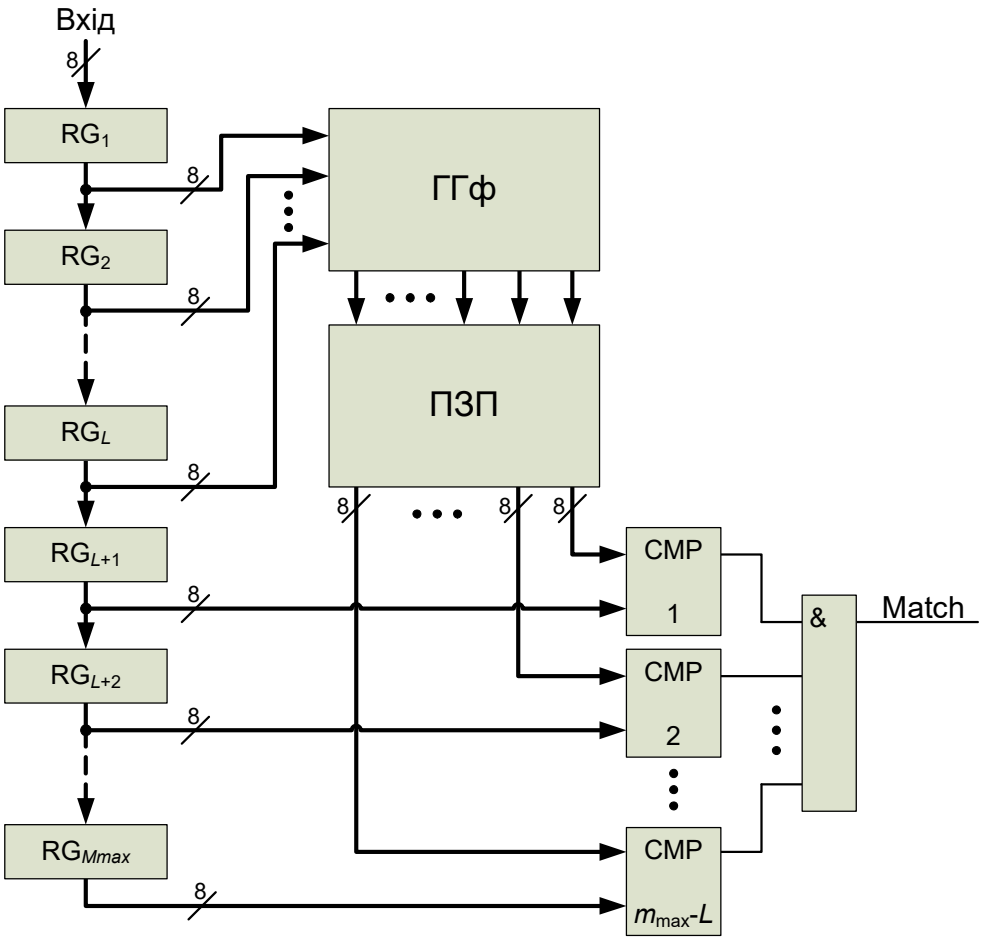


Рис. 4.8. Обчислювальна структура схеми розпізнавання HRCmp

Назвемо таке рішення *Схемою розпізнавання на базі геш-функції, ПЗП та цифрових компараторів* або *схемою* ***HRCmp*** (Hash-function + Read Only Memory + digital Comparators).

Особливістю ГГф для схеми HRCmp є здатність генерувати унікальні вихідні значення для всіх префіксів патернів.

Схема, що подана на рис. 4.8, є базовим варіантом класу технічних рішень, які можна умовно позначити як *Hash-function + read only memory + digital comparators*. У відомих практичних реалізаціях здійснено деякі модифікації та вдосконалення даного рішення [203], [204], [164]. Зокрема, запропоновані різні методи створення геш-функції. Проміжний блок непрямої адресації між ГГФ та ПЗП дозволяє ущільнити розташування патернів у постійній пам'яті. Також використовується кластеризація та розпаралелювання.

Відомі також двокаскадні рішення, подібні розглянутим вище, але побудовані за схемою *digital comparators + read only memory + digital comparators*, тобто в перших каскадах яких використані також компаратори [134], [205], [206], [207]. Втім, витрати ресурсів на такі схеми стають надмірними при наявності в словнику патернів великої кількості довгих однакових префіксів [203].

4.6.4.1. Функція оцінки схеми HRCmp. Ресурсна складова

Підрахуємо ресурси, що потрібні для створення схеми HRCmp.

Схема потребує логічні таблиці, тригери та блокову пам’ять:

$$R_{\text{HROMCmp}} = L_{\text{HROMCmp}} + \alpha\, F_{\text{HROMCmp}} + \beta\, B_{\text{HROMCmp}}. \qquad (4.109)$$

Логічні ресурси потрібні для побудови ГГф, в якості допоміжних компонентів ПЗП (див. підр. 4.6.3.5 "Функція оцінки СА-АК із блоковою пам'яттю ACBRAM. Ресурсна складова"), для побудови компараторів та багатовходової схеми "І":

$$L_{\text{HROMCmp}} = L_{\text{ГГф}} + L_{\text{DC}} + L_{\text{MUX}} + L_{\text{WR}} + L_{\text{CMP}} + L_{\&}, \qquad (4.110)$$

де LDC – кількість ЛТ, потрібних для побудови дешифратора в склади ПЗП,

L_{MUX} – кількість ЛТ, потрібних для побудови мультиплексора в складі ПЗП,

L_{WR} – кількість ЛТ, потрібних для побудови схеми запису в складі ПЗП.

Якщо в ГГф використовувати геш-функцію такого ж класу, до якого належали геш-функції розглянутих вище схем на базі фільтра Блума, то, подібно до (4.53):

$$L_{\mathrm{suf}} = G\left(\left\lceil \frac{8L}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8L}{x} \right\rceil - 1}{x-1} \right\rceil\right). \tag{4.111}$$

де L_{suf} – витрати на логіку для геш-функції, яка однозначно ідентифікує всі суфікси, що належать патернам довжиною від L+1 до $m_{\max}$ та зберігаються в ПЗП,

L – довжина префіксів (що подаються на вхід ГГф) тих самих патернів довжиною від L+1 до $m_{\max}$, що зберігаються в ПЗП,

G – вихідна розрядність ГГф (у бітах), якої має бути достатньо для адресації суфіксів у ПЗП.

У зв’язку з тим, що на практиці створити ідеальну геш-функцію неможливо, заповнити ПЗП кодами символів суфіксів патернів без прогалин не вдається навіть завдяки використанню непрямої адресації. Тому адресний простір виявляється більшим за ідеальний на певний коефіцієнт надмірності заповнення ПЗП корисною інформацію θ (більший за одиницю), який залежить від того, наскільки вдало організований розподіл суфіксів по блокам пам’яті в залежності від конкретної реалізації схем ГГф. Тому вихідна розрядність ГГф має бути рівною:

$$G = \left\lceil \log_2\left(\theta\,\sigma_{\text{ПЗП}}\right)\right\rceil, \tag{4.112}$$

де $\sigma_{\text{ПЗП}}$ – кількість суфіксів патернів, що містяться у ПЗП, яка дорівнює кількості патернів довжиною більш за L:

$$\sigma_{\text{ПЗП}} == \sum_{j=L+1}^{m_{\max}} L_j \,, \tag{4.113}$$

де δ_j – функція розподілу довжин патернів в наборі, що розпізнається,

$m_{\max}$ – довжина найдовшого патерну.

Вираз (4.111) визначає логічні витрати на геш-функцію, що ідентифікує всі суфікси патернів довжиною більш за L. Але коротші патерни (які в даному сенсі складаються лише з префіксів) також потрібно розпізнавати. Для всіх таких патернів с довжинами $j = m_{\min}, m_{\min} +1, \ldots , L$ логічні витрати визначаються виразом:

$$L_j = G_j \left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil \right), \tag{4.114}$$

де L_j – витрати на логіку для геш-функції, що розпізнає префікси довжиною j,

G_j – розрядність ГГф (у бітах), якої має бути достатньо для однозначної ідентифікації префіксу довжиною j:

$$G_j = \left| \log_2 \delta_j \right|, \tag{4.115}$$

де δ_j – j-те значення функції розподілу довжин, тобто кількість патернів довжиною j.

Тоді витрати на логіку для всіх геш-функцій, потрібних для розпізнавання префіксів:

$$L_{\text{prf}} = \sum_{j=m_{\min}}^{L} L_j = \sum_{j=m_{\min}}^{L} \lceil \log_2 \delta_j \rceil \left(\left\lceil \frac{8j}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil \right), \qquad (4.116)$$

Загальний об’єм логічних витрат на ГГф:

$$L_{\text{ГГф}} = L_{\text{prf}} + L_{\text{suf}}, \qquad (4.117)$$

Значення LDC, L_{MUX} та L_{WR} обчислюються за формулами, подібними до таких, що використовувалися для схеми ACBRAM (див. рис. 4.7 та формули (4.98), (4.99) і (4.101)), але з урахуванням того, що, по-перше корисна інформація потребує в θ раз більший об’єм пам'яті, по-друге, розрядність даних в ПЗП варіюється в широких межах в залежності від довжини суфіксу.

Вхідна адресна шина ПЗП, яка має ширину в G бітів, поділяється на n бітів адресу всередині блока пам’яті BRAM, та номер лінійки блоків BRAM (що мітить декілька суфіксів), який складається з m бітів.

Розрядність адресного простору всередині модуля BRAM можна знайти як

$$n = \left\lceil \log_2 1024 \left(\frac{U}{8} \right) \right\rceil = \lceil \log_2 128U \rceil = \lceil 7 + \log_2 U \rceil, \qquad (4.118)$$

де U – розмір (у Кбітах) одного блока BRAM (не враховуючі біти парності, якщо вони використовуються).

Якщо для створення схеми потрібно більше одного блока BRAM, тоді решта бітів адресної шини:

$$m = G - n. \tag{4.119}$$

Витрати логіки на дешифратор залежать лише від кількості m бітів в номері лінійки блоків BRAM (та числа входів ЛТ)

$$L_{\mathrm{DC}} = 2^m \left(\left\lceil \frac{m}{x} \right\rceil + 1 \right), \tag{4.120}$$

тоді як витрати на мультиплексор – від розрядності вихідної шини ПЗП та кількості лінійок BRAM:

$$L_{\mathrm{MUX}} = \left(m_{\max} - L \right) \left\lceil \frac{2^m - 1}{x - 1} \right\rceil. \tag{4.121}$$

Витрати на схему запису розраховуються так само, як и для схеми ACBRAM:

$$L_{\mathrm{WR}} = \left\lceil \frac{n}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4, \tag{4.122}$$

Нескладно перевірити, що для створення 2-входового 8-розрядного компаратора потрібно або п’ять 4-входових ЛТ, або три ЛТ на 6 чи на 8 входів. Введемо функцію-кваліфікатор

$$\Pi(x) = \begin{cases} 3, & x > 4 \\ 5, & x \le 4 \end{cases}. \tag{4.123}$$

Тоді

$$L_{CMP} = \Pi(x)(m_{\max} - L), \tag{4.124}$$

де x – число входів ЛТ.

Кількість ЛТ для багатовходової схеми "І" обчислюється, як звичайно:

$$L_{\&} = \left\lceil \frac{m_{\max} - L - 1}{x - 1} \right\rceil. \tag{4.125}$$

Тригери при створенні схеми HRCmp потрібні для побудови конвеєру, ГГф, в якості допоміжних компонентів ПЗП та для побудови багатовходової схеми "І":

$$F_{\text{HROMCmp}} = F_{\text{RG}} + F_{\text{ГГф}} + F_{\text{MUX}} + F_{\text{WR}} + F_{\&}, \tag{4.126}$$

де F_{MUX} –кількість тригерів, потрібних для побудови мультиплексора ПЗП,

F_{WR} –кількість тригерів, потрібних для побудови схеми запису ПЗП.

Для байтового кодування кількість тригерів для створення конвеєру довжиною $m_{\max}$ щаблів:

$$F_{\text{RG}} = 8\, m_{\max}, \tag{4.127}$$

Загальні витрати на тригери для побудови ГГф, як і для ЛТ, складаються з двох компонент:

$$F_{\text{ГГф}} = F_{\text{prf}} + F_{\text{suf}}, \tag{4.128}$$

де витрати на розпізнавання префіксів:

$$F_{\text{prf}} = \sum_{j=m_{\min}}^{L} \lceil \log_2 \delta_j \rceil \left\lceil \frac{\left\lceil \frac{8j}{x} \right\rceil - 1}{x-1} \right\rceil, \tag{4.129}$$

а витрати на розпізнавання суфіксів:

$$F_{\text{suf}} = G\left(\left\lceil \frac{\left\lceil \frac{8L}{x} \right\rceil - 1}{x-1} \right\rceil - 1 \right). \tag{4.130}$$

Значення F_{MUX} та F_{WR} також обчислюються подібно до схеми ACBRAM (див. рис. 4.7 та формули (4.100) і (4.102)) з урахуванням тих самих особливостей, що й для ЛТ:

$$F_{\text{MUX}} = (m_{\max} - L)\left(\left\lceil \frac{2^m - 1}{x-1} \right\rceil - 1 \right), \tag{4.131}$$

$$F_{\text{WR}} = n. \tag{4.132}$$

Вираз для $F_{\&}$ також типовий:

$$F_{\&} = \left\lceil \frac{m_{\max} - L - 1}{x-1} \right\rceil - 1. \tag{4.133}$$

Для підрахунку блоків BRAM потрібно спочатку знайти об’єм корисної інформації (у байтах для байтового кодування), що має містити ПЗП.

теоретично, цей об'єм складається з символів суфіксів усіх патернів різної довжини:

$$\Omega_{\text{ПЗП}} = \sum_{j=L+1}^{m_{\max}} \delta_j \left(m_j - L\right) = \sum_{j=L+1}^{m_{\max}} \delta_j \left(j - L\right), \tag{4.134}$$

де δ_j – функція розподілу довжин патернів,

m_j – довжина кожного j-го патерна.

(Тут L – довжина префіксів питомих патернів, що подаються на вхід ГГф, а $(m_j - L)$ – довжини їх суфіксів, що зберігаються в ПЗП).

Але таке значення близьке до реального лише для великих пакетів патернів (однакових за розміром), для яких вдається заповнювати блоки BRAM майже без прогалин. У випадку, коли патерни в наборі суттєво відрізняються за довжиною, а саме, коли кількість символів в поточному пакеті патернів менша за кількість символів, що містяться у одному блоці BRAM, яка дорівнює $128U$ (де U – об’єм блоку BRAM у Кбітах), здійснити щільне заповнення блокової пам'яті не вдається. До того ж в даному випадку на ситуацію починає негативно впливати факт відмінності довжини патерна від ступеня двійки. Тобто крім впливу неідеальності роботи ГГф, який коректується введенням коефіцієнту надмірності θ, також потрібно уточнити вираз (4.134) для обчислення величини $\Omega_{\text{ПЗП}}$.

Введемо функцію округлення у більший бік до найближчого ступеню двійки та її можливу реалізацію:

$$\text{UpPw2}(a) = 2^{\lceil \log_2 a \rceil}, \tag{4.135}$$

та функцію вибору найбільшого значення змінної

$$\mathrm{Max}(a,b)=\begin{cases} a, & a \ge b \\ b, & a < b \end{cases}. \tag{4.136}$$

Тоді вираз (4.134) з урахуванням (4.135) та (4.136) перетворюється на:

$$\Omega_{\text{ПЗП}} = \sum_{j=L+1}^{m_{\max}} \mathrm{Max}\left(\delta_j \,\mathrm{UpPw2}(j-L), 128U\right). \tag{4.137}$$

Кількість блоків BRAM, потрібних для створення ПЗП, мають вмістити такий об'єм даних $\Omega_{\text{ПЗП}}$ з урахуванням коефіцієнту надмірності:

$$B_{\mathrm{HROMCmp}} = \left\lceil \frac{M}{N} \right\rceil = \left\lceil \frac{\theta\,\Omega_{\text{ПЗП}}}{1024\left(\frac{U}{8}\right)} \right\rceil = \left\lceil \frac{\theta\,\Omega_{\text{ПЗП}}}{128U} \right\rceil. \tag{4.138}$$

де $M = \theta\,\Omega_{\text{ПЗП}}$ – фактичний об’єм ПЗП (у байтах), що потребується,

θ – коефіцієнт надмірності заповнення ПЗП корисною інформацію,

N – кількість символів (у байтовому кодуванні), яку може вмістити один блок BRAM,

U – розмір (у Кбітах) одного блока BRAM (не враховуючі біти парності, якщо вони використовуються).

Отже, ресурсна складова функції оцінки схеми HRCmp, тобто загальна кількість ресурсів, потрібних для її створення, враховуючи (4.109), (4.110), (4.111), (4.112), (4.113), (4.116), (4.117), (4.118), (4.119), (4.120), (4.121), (4.122), (4.124), (4.125), (4.126), (4.127), (4.128), (4.129), (4.130), (4.131), (4.132), (4.133), (4.137) та (4.138) дорівнює:

$$\theta^{R}_{\mathrm{HROMCmp}} = R_{\mathrm{HROMCmp}} = L_{\mathrm{HROMCmp}} + \alpha\, F_{\mathrm{HROMCmp}} + \beta\, B_{\mathrm{HROMCmp}} =$$

$$= L_{\Gamma\Gamma\text{ф}} + L_{\text{DC}} + L_{\text{MUX}} + L_{\text{WR}} + L_{\text{CMP}} + L_{\&} +$$

$$+ \alpha \left(F_{\text{RG}} + F_{\Gamma\Gamma\text{ф}} + F_{\text{MUX}} + F_{\text{WR}} + F_{\&} \right) + \beta\, B_{\text{HROMCmp}} =$$

$$= G \left(\left\lceil \frac{8L}{x} \right\rceil + \left\lceil \frac{\left\lceil \frac{8L}{x} \right\rceil - 1}{x-1} \right\rceil \right) + 2^m \left(\left\lceil \frac{m}{x} \right\rceil + 1 \right) + \left(m_{\max} - L \right) \left\lceil \frac{2^m - 1}{x-1} \right\rceil +$$

$$+ \left\lceil \frac{n}{\left\lfloor \frac{x-1}{2} \right\rfloor} \right\rceil + 4 + \Pi(x) \left(m_{\max} - L \right) + \left\lceil \frac{m_{\max} - L - 1}{x-1} \right\rceil +$$

$$+ \alpha \left(8 \cdot m_{\max} + G \left(\left\lceil \frac{\left\lceil \frac{8L}{x} \right\rceil - 1}{x-1} \right\rceil - 1 \right) + \left(m_{\max} - L \right) \left(\left\lceil \frac{2^m - 1}{x-1} \right\rceil - 1 \right) \right) +$$

$$+ \alpha \left(n + \left\lceil \frac{m_{\max} - L - 1}{x-1} \right\rceil - 1 \right) + \beta \left\lceil \frac{\theta \cdot \Omega_{\text{ПЗП}}}{128U} \right\rceil =$$

$$= G \left(\left\lceil \frac{8L}{x} \right\rceil + (\alpha + 1) \left\lceil \frac{\left\lceil \frac{8L}{x} \right\rceil - 1}{x-1} \right\rceil - 1 \right) +$$

$$+ \left(m_{\max} - L \right) \left((\alpha + 1) \left\lceil \frac{2^m - 1}{x-1} \right\rceil + \Pi(x) - \alpha \right) + (\alpha + 1) \left\lceil \frac{m_{\max} - L - 1}{x-1} \right\rceil +$$

$$+2^{m}\left(\left\lceil\frac{m}{x}\right\rceil+1\right)+\left\lceil\frac{n}{\left\lfloor\frac{x-1}{2}\right\rfloor}\right\rceil+\alpha\left(8\,m_{\max}+n-1\right)+\beta\left\lceil\frac{\theta\cdot\Omega_{\text{ПЗП}}}{128U}\right\rceil+4=$$

$$=\left\lceil\log_2\left(\theta\sum_{j=L+1}^{m_{\max}}\delta_j\right)\right\rceil\left(\left\lceil\frac{8L}{x}\right\rceil+(\alpha+1)\left\lceil\frac{\left\lceil\frac{8L}{x}\right\rceil-1}{x-1}\right\rceil-1\right)+$$

$$+\left(m_{\max}-L\right)\left((\alpha+1)\left\lceil\frac{\left\lceil\frac{\theta\sum_{j=L+1}^{m_{\max}}\delta_j}{128U}\right\rceil-1}{x-1}\right\rceil+\Pi(x)-\alpha\right)+(\alpha+1)\left\lceil\frac{m_{\max}-L-1}{x-1}\right\rceil+$$

$$+\left\lceil\frac{\theta\sum_{j=L+1}^{m_{\max}}\delta_j}{128U}\right\rceil\left(\left\lceil\frac{\left\lceil\log_2\left(\theta\sum_{j=L+1}^{m_{\max}}\delta_j\right)\right\rceil-\lceil 7+\log_2 U\rceil}{x}\right\rceil+1\right)+\left\lceil\frac{\lceil 7+\log_2 U\rceil}{\left\lfloor\frac{x-1}{2}\right\rfloor}\right\rceil+$$

$$+\alpha\left(8m_{\max}+\lceil 7+\log_2 U\rceil-1\right)+\beta\left\lceil\frac{\theta\sum_{j=L+1}^{m_{\max}}\text{Max}\left(\delta_j\,\text{UpPw2}(j-L),128U\right)}{128U}\right\rceil+4\,.\quad(4.139)$$

Проаналізуємо здобутий результат.

Відмінністю формули ресурсної складової функції оцінки схеми HRCmp (4.139) є наявність коефіцієнту надмірності заповнення ПЗП корисною інформацію θ, функції-кваліфікатора $\Pi(x)$, що визначає потрібні для створення компараторів логічні ресурси, функції округлення до найближчого ступеню двійки $\mathrm{UpPw2}(a)$, та функції вибору найбільшого значення змінної $\mathrm{Max}(a,b)$. В решті вона не відрізняється від ФО інших підходів та технологій – також має певні змінні, що залежать від набору патернів, та константи, що залежать від характеристик ПЛІС.

4.6.4.2. Функція оцінки схеми схема HRCmp. Часова складова

Швидкодія двокаскадної схеми розпізнавання HRCmp на базі ГГф, ПЗП та ЦК завдяки наявності геш-функції та блокової пам’яті близька до швидкодії схем на базі фільтра Блума. але на відміну від схеми MBF ця схема не містить селектор, демультиплексор та мультиплексом. З іншого боку для неї не потрібне додаткове уточнення результатів роботи, як для ФБ в зв'язку з системною помилкою розпізнавання другого роду. Тому схема HRCmp більш швидкодіюча.

Єдиним чинником, який теоретично міг би впливати на швидкість, є відносно великий об’єм пам’яті ПЗП. Але на практиці вдається за рахунок вибору належним чином геш-функції відтермінувати момент порівняння виходів сигналів ПЗП з відповідними щаблями конвеєру. За рахунок цього всі операції звернення до ПЗП та отримання від нього потрібних даних можна конвеєризувати. Тобто при практичному застосуванні вдається запобігти впливу даного чинника на швидкодію схеми HRCmp.

4.6.5. Способи знаходження коефіцієнтів нормалізації ресурсів РУО різних типів

Задіяні в розглянутих вище розрахунках коефіцієнти нормалізації α, β

та γ, які дозволяють звести кількісні значення ресурсів різного типу до єдиної умовної одиниці УЛТ, можуть бути знайдені шляхом використання різних підходів. Це може бути, наприклад, метод експертних оцінок чи принцип рівноцінності (ПРЦ) або їх комбінація. Принцип ПРЦ передбачає, що ресурси РУО кожного типу мають однакову "вагу". Тобто сумарна кількість всіх ЛТ у реконфігуровному виробі коштує так само, як і сумарна кількість всіх тригерів і т.д.:

$$L_{\text{FPGA}} = \alpha F_{\text{FPGA}} = \beta B_{\text{FPGA}} = \gamma M_{\text{RA}}, \tag{4.140}$$

де L_{FPGA} – кількість ЛТ в мікросхемі ПЛІС,

F_{FPGA} – кількість тригерів в ПЛІС,

B_{FPGA} – кількість блоків блокової пам’яті BRAM в ПЛІС,

M_{RA} – об’єм бортової пам’яті РУО,

α, β, γ –коефіцієнти нормалізації відповідних ресурсів відносно ЛТ.

З (4.140) безпосередньо випливають вирази для знаходження питомих коефіцієнтів:

$$\alpha = \frac{L_{\text{FPGA}}}{F_{\text{FPGA}}}, \quad \beta = \frac{L_{\text{FPGA}}}{B_{\text{FPGA}}}, \quad \gamma = \frac{L_{\text{FPGA}}}{M_{\text{RA}}}. \tag{4.141}$$

4.6.6. Технічні характеристики РУО, потрібні для використання методів прискорення

В підр. 4.1, 4.2 та 4.3 розглянуті властивості набору патернів та техніки поводження з ними, які потрібні для використання методів прискореного обчислення на базі функцій оцінки. Але до складу отриманих виразів для ФО входять також кількісні характеристики реконфігуровних прискорювачів.

В табл. 1.4 "Основні характеристики деяких РУО" в підр. 1.5.2 "Існуючі реалізації" наведено відомості щодо деяких РУО, обраних в якості прикладів

апаратної платформи для створення РАСЗТЗІ. Відображені в цієї таблиці характеристики є типовими параметрами, що містяться у специфікаціях виробів, і якими зазвичай оперують технічні працівники. Але деякі складові ФО не є типовими, і щоб здобути чисельні значення для них, потрібно докласти додаткових зусиль. (Наприклад, така характеристика, як здатність навантаження (fan-out) виходів логічних комірок в технічній документації ПЛІС взагалі не наводиться, тому її необхідно знаходити експериментальним шляхом).

Тому в табл. 4.2 для тих самих пристроїв, що й в табл. 1.4, наведені специфічні характеристики, які потрібні для використання згаданих методів прискорення, а саме – ті, що входять до виразів ФО та обмежень, які накладаються. Розробники РАСЗТЗІ, які застосовуватимуть розроблені в даному дослідженні методи комбінування, можуть використовувати дані цієї таблиці на практиці.

В табл. 4.2 також додано стовпчик з параметрами ПЛІС, що була застосована в роботі [164], з яких у підр. 6.7.4 "Експеримент № 4. Порівняння з опублікованими даними" проводитимуться порівняльні розрахунки.

Зауважимо, що в табл. 4.2 коефіцієнти нормалізації обчислено згідно виразів (4.141).

4.7. Висновки до розділу 4

1. Проаналізовано склад та основні властивості набору патернів сигнатурних систем захисту інформації. Зроблено необхідні визначення та виявлено технічні характеристики множини патернів сучасних РАСЗТЗІ.
2. Запропоновано техніки поводження з наборами патернів та їх характеристиками.
3. Виявлено основні кількісні технічні характеристики МР РАСЗТЗІ та їх компонентів, які потрібні для оцінки та порівняння їх ефективності.

Таблиця 4.2

Характеристики РУО, необхідні для розрахунків ФО методів прискорення

Параметр	**Позн.**	**Nallatech**	**Nallatech**	**Xilinx**	**Digilent**	**Digilent**	**[164]**
Модель	–	XUP-VV8	385A	VC709 Kit	NetFPGA-1G-CML	Atlys	–
Тип ПЛІС	–	Xilinx Virtex UltraScale+ VU13P	Intel (Altera) Arria 10 GX 1150	Xilinx Virtex-7 VX690T	Xilinx Kintex-7 XC7K325T	Xilinx Spartan-6 LX45	Xilinx Virtex2-800
Кількість входів LUT	x	6	8	6	6	6	4
Здатність навантаження*	y	10	10	10	10	10	10
Затримка ЦСЗ на базі ЛТ	z	32	32	32	32	32	16
Кількість Портів BRAM	p	2	2	2	2	2	2
Розмір блоку BRAM, Кбітів (без парності)	U	32	20	32	32	16	16
Кількість блоків BRAM	B_{FPGA}	2 688	2 713	1 470	445	116	168
Об’єм пам'яті BRAM, Мбітів (без парності)	N_{FPGA}	86,016	54,260	47,040	14,240	1,856	2,688
Кількість ЛТ	L_{FPGA}	1 728 000	427 200	433 200	203 800	27 288	93 184
Кількість тригерів	F_{FPGA}	3 456 000	1 708 800	866 400	407 600	54 576	93 184
Наявність контролерів ОЗП	$K_{\text{ОЗП}}$	+	+	+	+	+	–
Бортовий ОЗП: організація, розрядність	M_{RA}, організація	4x 128 Гб, 64-розр.	2x 16 Гб, 64-розр.	2x 4 Гб, 64-розр.	1x 512 Мб, 32-розр.	1x 128 Мб, 16-розр.	–
Коеф. нормал. тригерів	α	0,5	0,25	0,5	0,5	0,5	1,0
Коеф. нормал. блоків BRAM	β	643	157	295	458	235	554
Коеф. нормал. пам’яті РУО	γ	3,4	13,4	54,2	400	213	–

*потребує уточнення

4. Розроблено методи прискореного обчислення технічних параметрів РАСЗТЗІ, заснований на побудові функцій оцінки (функціональних залежностей кількісних показників ефективності від властивостей словнику патернів та характеристик РУО), який не потребує виконання

повної процедури виконання проєкту щодо створення реконфігуровного пристрою на базі ПЛІС.

5. Досліджено приклади побудови функцій оцінки для наступних найбільш перспективних схем розпізнавання РАСЗТЗІ:
   - базова схема на АП BsCAM;
   - модифікована схема на АП DCAM;
   - модифікована схема на АП DpCAM;
   - повнорозмірний фільтр Блума LBF;
   - спрощений фільтр Блума SBF;
   - базова схема СА-АК із зовнішньою пам'яттю ACRAM;
   - базова схема СА-АК із блоковою пам'яттю ACBRAM;
   - схема СА-АК за конвеєрною схемою PipACBRAM;
   - двокаскадна схема HRCmp.
6. Розглянуті технічні характеристики РУО, що потрібні для використання методів прискорення, а також питання знаходження коефіцієнтів нормалізації ресурсів РУО різних типів.

РОЗДІЛ 5. МЕТОДИ ПОБУДОВИ ОПТИМІЗОВАНОЇ СТРУКТУРИ МОДУЛЯ РОЗПІЗНАВАННЯ РАСЗТЗІ

Дослідженнями в галузі побудови складних цифрових пристроїв, включаючи реконфігуровні засоби, займаються численні вчені. Відомі та успішно використовуються на практиці процедури формалізації процесу проектування проблемно-орієнтовних реконфігуровних пристроїв. Наприклад, в недавній роботі вітчизняних вчених з Інституту кібернетики ім. В.М. Глушкова НАН України [208] розглядається наступна методика. Згідно [91] та [209] модель пристрою, що проектується, можна подати як $S = < M, A, B, P >$, де M – множина математичних методів, A – множина алгоритмів реалізації методу, $B = \{b\}$ – алфавіт конструктивів з котрих будується структура, P – процедура опису проєкту. Процес проектування полягає у вирішенні задачі синтезу структури на базі конструктивів $\{b\}$ алфавіту B для виконання алгоритму A за методом M. Результатом процедури P є опис проекту у вхідному мові САПР.

Процес створення МР РАСЗТЗІ, розглянутий в даному досліджені, повністю узгоджується зі згаданим вище формальним описом процесу проектування реконфігуровних пристроїв. Уточнення, які внесені в даний процес, полягають лише у тому, що множина методів M ділиться на дві підмножини: MСТ – методи синтезу незмінних компонентів РАСЗТЗІ (в якості котрих можуть використовуватися будь-які відомі методи побудови цифрових схем на ПЛІС), та MЗМ – методи синтезу змінних компонентів РАСЗТЗІ (розглянути нижче у даному розділі). В якості бібліотечних компонентів $\{b\}$ використовуються блоки розпізнавання, побудовані на базі найбільш ефективних підходів до побудови МР РАСЗТЗІ та їх модифікацій, які були виокремлені в розділі 2 та формалізовані: на якісному рівні – в розділі 3, на кількісному – в розділі 4.

Розглянемо методи, що базуються на ідеї комбінування (сумісного використання) різних підходів до побудови РАСЗТЗІ з метою максимізації їх ефективності за рахунок використання переваг кожного з цих підходів [210].

5.1. Технічні параметри модулю розпізнавання та його компонентів

В підр. 2.2 "Показники ефективності РАСЗТЗІ" були сформульовані показники, за якими можна оцінювати ефективність РАСЗТЗІ. Для досягнення максимальної ефективності створення МР запропоновані в даному розділі методи використовують процедури оптимізації за певним критерієм. Розглянемо, які технічні параметри можуть бути використані в якості цільових функцій для виконання цих процедур, маючи метою підвищення ефективності РАСЗТЗІ в сенсі сформульованих показників.

Як було з'ясовано в розділі 3, є три основні класи показників ефективності, придатних до кількісного вимірювання: 1) показники ресурсних витрат; 2) часові характеристики; 3) показники, які мають сенс виконаної роботи або об'єму обробленої інформації. Всі інші показники є або похідними від цих трьох, або принципово не можуть бути оцінені кількісно. Тому розглянуті нижче технічні параметри для цільових функцій оптимізації або належать до цих трьох груп, або певним чином з них походять.

Модуль розпізнавання РАСЗТЗІ, побудований з використанням методів комбінування, складається з декількох компонентів, побудованих на базі різних підходів, кожен з яких здатен самостійно або частково виконувати задачу множинного розпізнавання патернів. Назвемо такі компоненти ***блоками розпізнавання*** (БР). Розглянемо також, яким чином технічні параметри модуля розпізнавання РАСЗТЗІ можуть бути обчислені для використання в якості цільових функцій для процедур оптимізації.

5.1.1. Ресурсні параметри

Будь-яка цифрова схема в загальному випадку складається з декількох компонентів. Ресурси R, потрібні для її синтезу, складаються з ресурсів R_i,

потрібних для створення кожного *i*-го компоненту:

$$R = \sum_{i=1}^{n} R_i \,, \tag{5.1}$$

де n – кількість компонентів.

5.1.2. Часові параметри

Кожен компонент обробляє дані з певною швидкодією, одним з параметрів якої є час T розповсюдження сигналу від входу до виходу. В разі паралельного з'єднання компонентів, кожен з котрих має свій такий параметр T_i, час затримки цифрової схеми в цілому визначається найповільнішим з них:

$$T = \max(T_i) \,. \tag{5.2}$$

У випадку послідовного з'єднання компонентів час затримки цифрової схеми в цілому визначається як сума затримок кожного з них:

$$T = \sum_{i=1}^{n} T_i \,. \tag{5.3}$$

5.1.3. Пропускна здатність

Пропускна здатність модулю розпізнавання визначається як кількість інформації, що обробляється за одиницю часу. Цей параметр формується з аналогічних параметрів кожного з компонентів. В разі паралельного з'єднання компонентів пропускна здатність МР обчислюється як сума:

$$C = \sum_{i=1}^{n} C_i \tag{5.4}$$

де C_i – пропускна здатність кожного з компонентів, яка знаходиться як

$$C_i = \frac{M_i}{T_i},$$

де M_i – об'єм інформації, що обробляється i-м компонентом.

У випадку послідовного з’єднання компонентів пропускна здатність МР визначається найменш продуктивним з них:

$$C = \min(C_i). \tag{5.5}$$

5.1.4. Похідні параметри

З метою здатності більш ретельно порівнювати технічні рішення численними дослідниками та розробниками РАСЗТЗІ крім відносно простих виразів (5.1) – (5.5) були запропоновані та використані більш складні параметри, які також можуть бути задіяні в якості цільовій функції при створенні критеріїв оптимізації.

До останніх належить, зокрема, *ефективність пропускної здатності* (Throughput efficiency) [211]:

$$Q = \frac{C}{R} \tag{5.6}$$

приведена ефективність пропускної здатності (a new Performance Efficiency Metric (nPEM) в [164], Efficiency1 в [55], Memory efficiency в [196]):

$$E = \frac{C \cdot \Omega}{R}, \tag{5.7}$$

де Ω – сумарна кількість всіх символів в словнику патернів, що розпізнається,

або *ефективність ресурсів* (Memory efficiency) [211]:

$$F = \frac{R}{S}, \tag{5.8}$$

де S – кількість сигнатур в базі даних РАСЗТЗІ.

В разі потреби можуть бути використані й більш складні та витончені похідні технічні параметри, якщо потрібно врахувати ті чи інші особливості конкретного застосування РАСЗТЗІ.

Вирази (5.1) – (5.7) можуть бути задіяними в якості цільовій функції при створенні критеріїв оптимізації для методів підвищення ефективності РАСЗТЗІ.

5.2. Критерії оптимізації

Критерієм оптимізації, що здійснюється при застосуванні методів підвищення ефективності РАСЗТЗІ, є досягнення мінімуму або максимуму певної цільової функції, в якості якої виступає чисельне значення деякого технічного параметру при наданих умовах та накладених обмеженнях Constr. Критерій оптимізації з урахуванням (5.1) – (5.5)(5.5) може бути заданий суто ресурсний

$$K_R: \quad R \to \min \Big/ Constr,$$

суто часовий

$$K_T: \quad T \to \min \Big/ Constr,$$

або як критерій продуктивності

$$K_C: \quad C \to \max \Big/ Constr.$$

Згадані вище параметри ефективності пропускної здатності (5.6), приведеної ефективності пропускної здатності (5.7) та ефективності ресурсів (5.8) також можуть бути використані в якості цільових функцій для відповідних критеріїв оптимізації:

$$K_Q: \quad Q \to \max \Big/ Constr,$$

$$K_E: \quad E \to \max \Big/ Constr.$$

$$K_F: \quad F \to \max \Big/ Constr.$$

Умови та обмеження Constr є сукупністю наступних чинників:

$$Constr = \{\Theta, \Psi, \chi\},$$

де Θ – сукупність характеристик реконфігуровного прискорювача, розшифровку яких наведено в попередньому розділі у формули (4.6):

$$\Theta = \{x, y, z, p, U, B_{\text{FPGA}}, L_{\text{FPGA}}, F_{\text{FPGA}}, T_{\text{FPGA}}, M_{\text{RA}}, \alpha, \beta, \gamma\},$$

Ψ – кількісні обмеження, які накладає користувач системи захисту, в залежності від обраного критерію, підходу до побудови РАСЗТЗІ, або з інших міркувань, яки зазвичай мають сенс порогового значення для певної технічної характеристики (розглянуті нижче),

χ – множина опцій, що задає користувач, в якості яких використовуються якісні показники ефективності РАСЗТЗІ, такі як можливість динамічної реконфігурації Din, опція протидії спрямованим на МСВВ атакам $AtDef$, спроможність системи функціонувати в режимі запобігання вторгнень Ips, опція незалежності швидкодії від вхідних даних $Tstab$ тощо:

$$\chi = \{Din, AtDef, Ips, Tstab, \ldots\}.$$

Використання реконфігуровної елементної бази накладає певну специфіку на процес створення обчислювального засобу порівняно з традиційною задачею розробки цифрових пристроїв. Якщо при компоновці дискретної системи з "розсипної логіки" розробник використовує довільну кількість довільних компонентів, керуючись вимогами щодо швидкодії та економії комплектуючих, то при використанні реконфігуровного прискорювача об’єм наявних ресурсів є фіксованим і обумовлюється переважно характеристиками мікросхеми ПЛІС та бортового ОЗП, що входять до складу використаного РУО.

Тому, якщо заданий критерій, що заснований на характеристиках швидкодії, пропускної здатності або їх похідних, використовується обмеження виду:

$$R < R_{\mathrm{FPGA}}\,(1 - \Delta/100)$$

або

$$R < R_{FPGA} - \Delta_R ,$$

де R_{FPGA} – об'єм ресурсів ПЛІС, що входить до чинника Fpga, згаданого вище,

Δ – порогове значення (у відсотках) невикористаної частки ресурсів ПЛІС введення якої обумовлено тим фактом, що коли розмір цифрової схеми, що синтезуються, наближається до максимуму ресурсів, це призводить до вибухоподібному зросту складності та витрат часу на синтез,

Δ_R – порогове значення (в абсолютних умовних одиницях) невикористаної частки ресурсів ПЛІС.

В разі завдання ресурсного критерію оптимізації, тобто коли необхідно мінімізувати витрати, то вимоги щодо швидкодії задаються у вигляді обмеження виду:

В разі використання ресурсного критерію оптимізації, тобто коли необхідно мінімізувати витрати, обмеженнями є вимоги щодо швидкодії, які задаються у наступному вигляді:

$$T_{FPGA} < T < T_{USR},$$

де T_{FPGA} – максимальне значення показника швидкодії ПЛІС – одна з характеристик РУО,

T_{USR} – порогове значення швидкодії, задане користувачем – один з елементів множини кількісних обмежень Ψ.

Отже, критерії оптимізації сформульовані, тому можна перейти до розглядання власне методів побудови оптимізованої структури модуля розпізнавання РАСЗТЗІ.

5.3. Метод паралельного комбінування

Суть ***Методу паралельного комбінування*** (МПрКмб) полягає в поділі набору патернів, які повинні відшукуватися модулем розпізнавання, на

підгрупи та синтезі в складі модулю розпізнавання такої ж кількості різнорідних блоків розпізнавання, кожен з яких найбільш результативно відшукує патерни відповідної підгрупи, максимально використовуючи при цьому переваги задіяного в ньому підходу. Найвища ефективність при цьому досягається за рахунок охоплення обох процесів – поділу та підбору БР – загальною процедурою оптимізації.

Підґрунтям методу є той факт, що патерни, які входять до словнику сигнатур, відміні за довжиною та властивостями самоподоби, в наслідок чого ефективність, з якою вони відшукуються, також відмінна та залежить від підходу до побудови засобу розпізнавання.

Іншим чинником на користь даного методу є обмеженість ресурсів РУО, на базі якого побудований РАСЗТЗІ. Використання лише однієї з можливих схем розпізнавання призводить до того, що частина ресурсів реконфігуровного обчислювача (наприклад, логічних) використовуються майже повністю, а інша (ресурси блокової чи бортової пам'яті) – не задіяна взагалі. Як наслідок, втрачається ефективність.

5.3.1. Структура методу МПрКмб

На рис. 5.1. подано схематичне зображення структури МПрКмб. *На етапі синтезу* набір патернів P, що підлягає розпізнаванню, розбивається на n підгруп P_i, $i = 1, 2, \ldots n$. Після синтезу *в процесі функціонування* кожний блок розпізнавання БР$_i$ в складі МР здійснює розпізнавання патернів відповідної підгрупи в потоці вхідних даних, що подаються одночасно на входи всіх БР. В разі виявлення збігу фрагменту вхідних даних з якимось патерном активується відповідний сигнал розпізнавання.

5.3.2. Функціонування методу МПрКмб

Для досягнення максимальної ефективності процес розбиття патернів на підгрупи та формування структури МР доцільно здійснювати під управлінням загальної *процедури оптимізації*.

Змінними параметрами при цьому є, з одного боку, варіанти поділу патернів, тобто кількість підгруп n та склад кожної з них P_i, з іншого боку – варіанти реалізації кожного з БР$_i$, які вибираються з бібліотеки готових компонентів, створених з використанням підходів та модифікацій, розглянутих в розділі. 3.

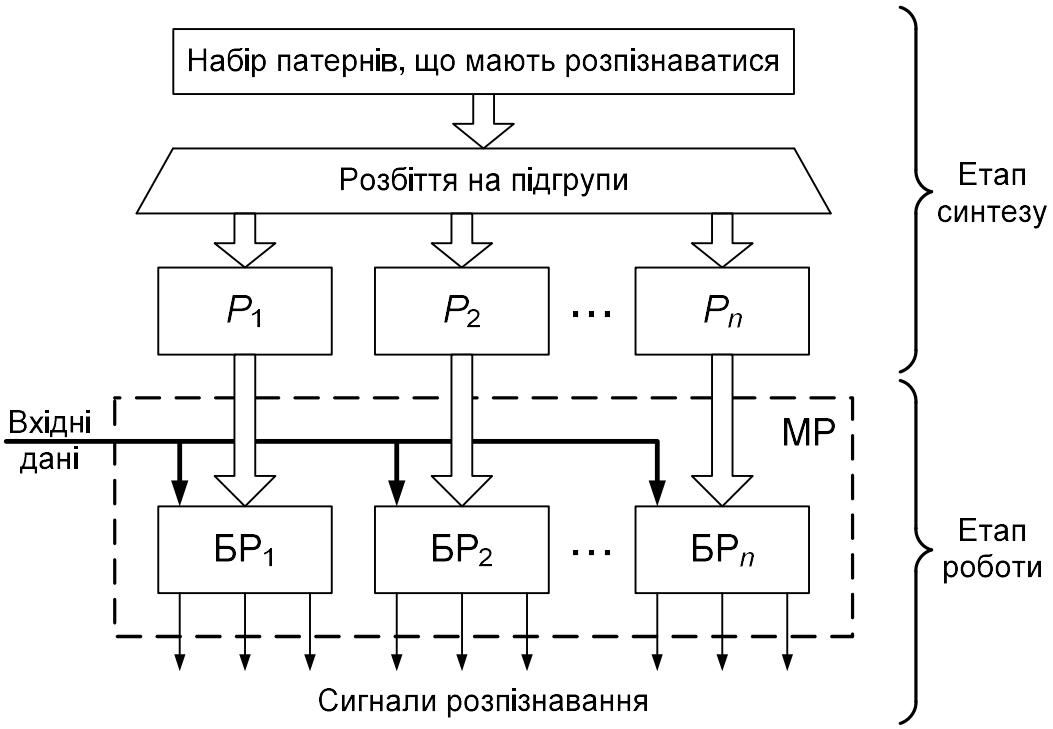


Рис. 5.1. Схематичне подання структури методу паралельного комбінування

Критерії оптимізації при реалізації МПрКмб можуть використовуватися різними в залежності від от потреб користувача та показнику ефективності РАСЗТЗІ, який потрібно максимізувати.

5.3.3. Реалізація методу МПрКмб

Алгоритм, що реалізує метод МПрКмб, в процесі своєї роботи варіює змінні параметри, на кожному кроці знаходячи методом прискореного обчислення, розглянутого в попередньому розділі, функції оцінки θ_i ресурсних та часових характеристик кожного з БР$_i$ та МР в цілому.

На кожному кроці ітераційного процесу для кожного блоку розпізнавання алгоритм отримує на вході заданий набір патернів підгрупи P_i,

та потрібні технічні параметри РУО, на виході видає здобуті оцінки чисельних значень, по-перше, об'єму ресурсів R_i, потрібних для синтезу БР$_i$, по-друге – часової затримки T_i, яку матиме БР$_i$.

Як випливає з постановки оптимізаційної задачі для МПрКмб, алгоритм його реалізації повинен знайти оптимальний за відповідним критерієм варіант побудови МР із загального числа комбінацій, що містить:

1) всі можливі варіанти вибору схемотехнічних рішень цих БР$_i$, що вибираються з бібліотеки готових компонентів;

2) всі можливі варіанти поділу патернів між БР$_i$.

Тобто діапазон пошуку оптимального рішення охоплює одночасно як множину комбінацій блоків розпізнавання, так і множину комбінацій поділу між ними патернів. Причому, ці дві множини комбінаторно перемножуються між собою.

Перша підзадача – перебір варіантів технічних рішень – є типовою для процесу створення складного реконфігуровного пристрою, тому докладно досліджена та пророблена. Відповідні методики можна знайти, наприклад, в згаданих вище публікаціях [208] та [209].

Друга підзадача – охоплення всіх можливих варіантів поділу патернів словнику між різнорідними блоками розпізнавання – є комбінаторно складною і потребує неприйнятно багато часу, якщо вирішувати її методом повного перебору. Тому в наступному розділі запропоновані техніки, що дозволяють уникнути опрацьовування повного числа комбінацій та знизити часові витрати до прийнятних значень.

Стосовно існуючих розробок, в яких метод МПрКмб використовувався на практиці – в наукових публікаціях автору вдалося знайти лише один приклад, коли дії розробників МП РАСЗТЗІ можна кваліфікувати, як використання ідеї методу паралельного комбінування. В роботі [164] автори, розглядаючи два технічних рішення – DpCAM (див. підр. 3.1.6 "Подальше зменшення ресурсних витрат") і PHmem (яке є реалізацією методу послідовного каскадування, що розглядатиметься в підр. 5.4 "Метод

послідовного каскадування"), помітили, що перша схема порівняно с другою більш ефективно обробляє довгіші патерни, тому вирішили поділити словник на дві частини: патерні довжиною до 50 символів включно розпізнавати технікою PHmem, а решту – технікою DpCAM. В результаті комбінована схема (яку вони назвали PHmem+DpCAM) продемонструвала зріст показника приведеної ефективності пропускної здатності (див. підр. 5.1.4 "Похідні параметри") на 10-25% порівняно з окремими рішеннями PHmem та DpCAM в залежності від розміру словника патернів. Цей приклад свідчить, що, по-перше, метод МПрКмб має підтверджену на практиці спроможність покращення показників ефективності компонентів РАСЗТЗІ, по-друге, розробники реконфігуровних засобів інформаційної безпеки поки ще нечасто використовують цей метод, і його формалізація та вдосконалення, що здійснені в даної роботі, мають посприяти більш активному його застосуванню.

5.3.4. Узгодження швидкісних характеристик блоків розпізнавання

При об'єднанні методом МПрКм різних за своєю природою підходів до побудови схем розпізнавання виникає потреба в узгодженні швидкісних характеристик відповідних БР$_i$. З цією метою для повільнішої технології застосовується прискорення, наприклад, шляхом розпаралелювання або за рахунок обробки більш, ніж одного байту за такт.

5.4. Метод послідовного каскадування

Недоліком розглянутого вище методу МПрКмб є вимога того, щоб кожен з патернів був розпізнаним повністю, інакше відповідний сигнал тривоги не активується. Дійсно, для виявлення збігу фрагменту вхідних даних з якимось патерном потрібно, щоб співпали всі символи. Але щоб зробити висновок, що поточний фрагмент вхідних даних не співпадає з патерном, достатньо виявити розбіжність всього в одному символі. Тоді порівняння решти рядку стає зайвим. Бази даних сигнатур сучасних РАСЗТЗІ

містять патерни довжиною в кілька десятків символів (деякі – більш ніж 150 символів [55]). виявлення розбіжності лише в маленькому фрагменті патерну може помітно знизити об'єм роботи, потрібної для його розпізнавання.

Метод послідовного каскадування (МПсКс) використовує цей факт для підвищення ефективності процесу розпізнавання. Його суть полягає у розбитті операції порівняння вхідних даних з патерном на послідовні етапи, на кожному з котрих здійснюється часткове розпізнавання відповідного фрагменту патерну за умови, що виявлено збіг на усіх попередніх етапах.

Внаслідок особливостей методу МПсКс до характеристик різних каскадів висуваються різні вимоги. Тому використання в них компонентів, різних за принципами побудови, створює передумови для підвищення ефективності рішення в цілому.

5.4.1. Структура методу МПсКс

На рис. 5.2. подано схематичне зображення структури МПсКс. *На етапі синтезу* кожний патерн всередині вхідного набору P, ділиться на n фрагментів, в результаті чого сам набір також ділиться на n частин Q_i, $i = 1, 2, \ldots n$. Після синтезу *в процесі функціонування* кожний каскад $КС_i$ в складі МР виконує розпізнавання відповідних фрагментів патернів в потоці вхідних даних, що подаються одночасно на входи всіх КС, але лише у випадку виявлення збігу для відповідного фрагменту з попереднього каскаду. Сигнали про частковий збіг передаються з попереднього на наступний каскад. В разі виявлення збігу для якогось фрагменту в останньому каскаді $КС_n$, активується потрібний сигнал розпізнавання.

Виходячи з припущення, що збіг вхідних даних з довшим фрагментом патерну є менш вірогідною подією, ніж з коротшим, можна очікувати, що потреба в використанні кожного наступного каскаду буде виникати рідше порівняно з попереднім, внаслідок чого вимоги до швидкодії (та, як наслідок – споживаних ресурсів) каскадів знижатимуться за зростанням номеру каскаду аж до можливості реалізації останніх каскадів програмними

засобами. Водночас швидкодія перших каскадів може бути досягнута вища, за рахунок розпізнавання більш коротких підрядків. (Граничним випадком є двокаскадна схема, в якої кожен патерн розбивається на короткий префікс, що розпізнається швидкодіючим КС, та довгий суфікс, який обробляється в разі потреби повільнішим, за рахунок чого – менш ресурсоємним блоком). В результаті побудова МР за каскадною схемою може призвести одночасно як до підвищення швидкодії МСВВ в цілому, так і до скорочення потрібних ресурсів.

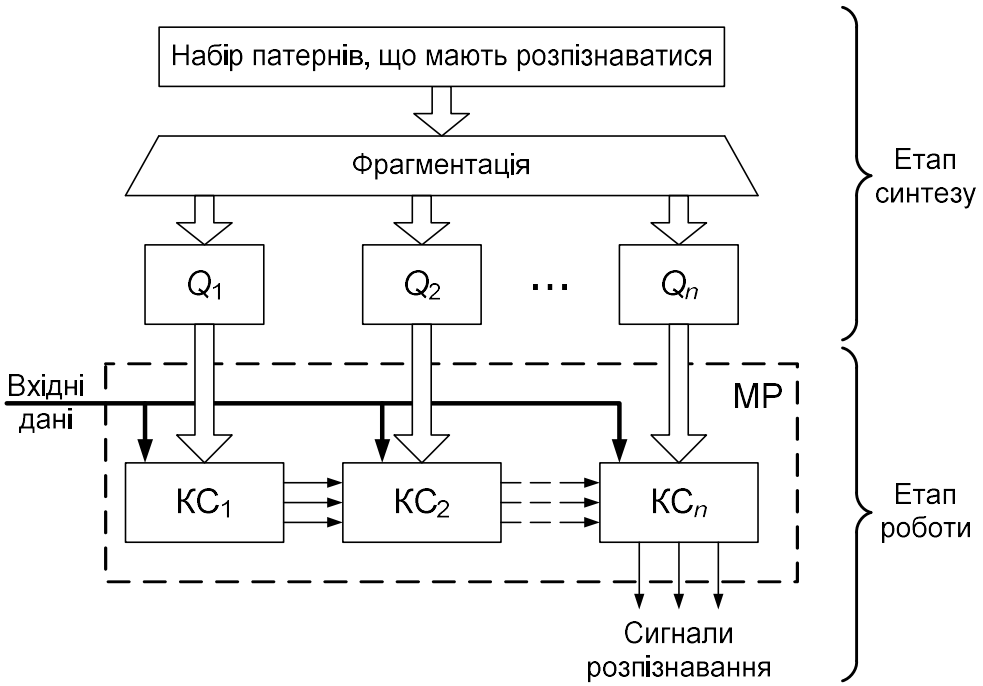


Рис. 5.2. Схематичне подання структури методу послідовного каскадування

Особливістю методу МПсКс є те, що всі каскади, починаючи з другого, тобто КС$_i$ для $i = 2, 3, \ldots n$, повинні бути здатними до селективного розпізнавання (див. підр. 2.2.3.3 "Селективне розпізнавання"), тобто по зовнішньому керуючому сигналу перемикатися на розпізнавання різних підмножин патернів, які вони розпізнають. Це потрібно тому, що перший фрагмент загального патерну, що розпізнається в першому каскаді, в загальному випадку може виявитися спільним префіксом певної кількості патернів, які розрізняються фрагментами, що розпізнаються в решти

каскадів. У граничних випадках кожен патерн, який не має в попередньому каскаді спільних префіксів з іншими патернами, складає всю множину словнику патернів, що мають розпізнаються в поточному каскаді. Тоді поточний каскад виконує задачу не розпізнавання патернів, а виявлення однієї послідовності, тобто вирішує задачу подібну такої, що виконує скінченний автомат – акцептор (див. підр. 3.3 "Використання алгоритму Ахо–Корасік на базі скінченних автоматів"). Фактично, в таких випадках попередній каскад видає гіпотезу щодо появи однієї окремої послідовності символів у вхідному потоці даних, а наступний каскад має її перевірити.

Функція виявлення послідовності (sequence detection) простіша за функцію розпізнавання множини патернів (multiple pattern matching). Цей факт можна використати, замінивши в поточному каскаді один блок розпізнавання на низку блоків – акцепторів та один або також низку блоків розпізнавання, робота кожного з яких ініціюється з попереднього каскаду сигналом о виявленні відповідного фрагменту.

З викладеного випливає, що до другого і наступних каскадів висуваються менш жорсткі вимоги не тільки за швидкодією, але й за функціональністю. Тому до їх побудови можуть застосовуватися більш прості підходи, ніж ті, що розглянуті в розділі 3, що також є перевагою методу МПсКс.

5.4.2. Функціонування методу МПсКс

Для знаходження найкращого значення кількості фрагментів n та максимально ефективного варіанту розбиття множини патернів P на фрагменти також доцільно використовувати процедуру оптимізації.

Критерії оптимізації можуть бути такими ж, як і для методу МПрКмб. *Змінними параметрами* є кількість фрагментів тобто число каскадів n, співвідношення довжин фрагментів для кожного каскаду $КС_i$ та варіанти реалізації кожного з каскадів, які також вибираються з бібліотеки готових компонентів.

Особливістю методу МПсКс є необхідність крім фіксованих властивостей питомій підгрупи патернів враховувати статистичні характеристики вхідного інформаційного потоку, зокрема, у випадку застосування МСВВ – імовірнісні характеристики мережевого трафіку. Це обумовлено тим фактом, що частота запуску наступних каскадів і, як наслідок, вимоги до їх швидкодії, залежать від властивостей вхідних даних.

Слідством цієї особливості є головний недолік методу МПсКс – непередбачуваність пропускної здатності (див. підр. 2.2.2 "Показники продуктивності (швидкісні)"), що призводить до втрати такого якісного показника, як здатність протидії атакам, спрямованим на засоби захисту інформації (див. підр. 2.2.3.1 "Спроможність протидіяти атакам на РАСЗТЗІ"). Цей важливий факт треба враховувати при застосуванні даного методу при побудові МР РАСЗТЗІ.

5.4.3. Реалізація методу МПсКс

Алгоритм, що реалізує метод МПсКс, в процесі своєї роботи варіює змінні параметри, тобто варіанти розбиття множини патернів P на фрагменти, на кожному кроці знаходячи методом прискореного обчислення функції оцінки ресурсних та часових характеристик кожного з КС$_i$ та МР в цілому.

Підзадача розбиття патернів на фрагменти з метою оптимізації порівняно з охопленням всіх комбінацій поділу патернів між БР$_i$ в методі МПрКмб, є значно менш складною. Фактично, потрібно лише визначитися з кількістю фрагментів та визначити межи, що відділяють довжини фрагментів один від одного. Тому для вирішення цієї підзадачі застосування методу повного перебору виявляється прийнятним.

Завершуючи розглядання основних положень щодо методу МПсКс, розглянемо існуючі розробки, в яких він використовується на практиці.

Застосування ідеї каскадування в існуючих розробках зустрічається значно частіше у порівнянні з методом паралельного комбінування. Але

відсутність системного підходу та формалізованого опису метода послідовного каскадування обмежували можливості розробників щодо досягнення максимального ефекту від його використання. Зокрема, в деяких рішеннях схеми розпізнавання всіх каскадів будуються за одними й тими ж підходами, тобто різниця у вимогах до різних каскадів не використовується повною мірою.

Найчастіше у подібних розробках застосування каскадування зустрічається підхід на основі фільтрів Блума. Наприклад, в роботі [212] з метою зменшення кількості ФБ довгі патерни розділяють на невеликі підрядки. Для розпізнаних в першому каскаді префіксів зберігаються спеціальні індекси, які разом з залишками рядків, що були помічені як частково розпізнані, ідентифікуються спеціальною схемою в другому каскаді. Назвемо це рішення *каскадованим фільтром Блума* або *схемою* ***CasBF*** (Cascaded Bloom Filter).

В роботі [213] в обох каскадах використовуються фільтри Блума. На етапі програмування ФБ результати обчислення хеш-функцій у первинному ФБ підключаються як дзеркальне зображення вставленого патерну. Це дзеркальне зображення гешується в наступний ФБ як масив. В результаті, як стверджують автори, драматично знижується вірогідність помилок розпізнавання другого роду. Також зменшується потрібна пам'ять. Ця розробка отримала назву *гешовано-каскадований фільтр Блума* або *схема* ***CHBF*** (Cascade Hash Bloom Filter).

Запропоновані в роботах [187], [188], [214], [215] схеми каскадування ФБ взагалі не підвищують швидкодію й не знижують ресурсні витрати, а використовуються виключно для зниження енергоспоживання.

Кращі результати при використанні методу МПсКс продемонстрували розробки, в яких запропоновані двокаскадні схеми, що використовують різні технології в першому і другому каскаді. В роботі [164] в них задіяні відповідно геш-функції та цифрові компаратори. Цієї двокаскадній схемі автори дали назву PHmem. Реалізація рішення споживає ЛТ, тригери та

блокову пам'ять. Порівняно з однокаскадною схемою DpCAM схема PHmem забезпечує близькі показники швидкодії та дещо кращі характеристики споживання ресурсів. Згідно нотації, прийнятої в даному дослідженні, ця схема отримала назву HRCmp (див. підр. 4.6.4 "Техніка створення функцій оцінки для двокаскадної схеми на базі геш-функції, ПЗП та цифрових компараторів").

5.5. Метод вертикального об’єднання

Принцип дії *методу вертикального об’єднання* (МВрОб) полягає в сполучанні кількох підходів або їх модифікацій, а також технологій та технік в одному модулі таким чином, що жоден з них не може бути відокремленим від інших. Зазвичай при такому об'єднанні один з підходів / технологій виконує основну функцію, під яку будується схема в цілому, а інші вирішують допоміжні задачі.

5.5.1. Структура методу МВрОб

На рис. 5.3. подано схематичне зображення структури МВрОб.

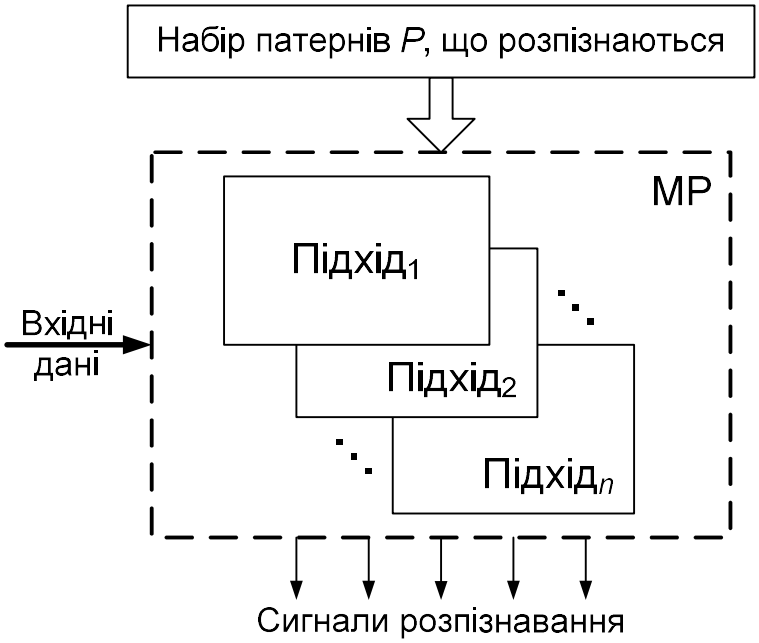


Рис. 5.3. Схематичне подання структури методу вертикального об’єднання

Як можна бачити, при використанні даного методу набор патернів, що розпізнається, ніяким чином не діліться на підгрупи. Тобто, побудований за методом МВрОб модуль діє як єдиний функціональний блок.

5.5.2. Таблиця сумісності для МВрОб

В зв’язку з тісною взаємодією підходів, їх модифікацій та допоміжних технік різноманітної природи, що використовуються при створенні МП РАСЗТЗІ методом МВрОб, теоретична формалізація цього методу виявляється неможливою. Для систематизації наявного досвіду та з метою спрощення процесу розробки компонентів РАСЗТЗІ, пропонується розвиток методу МВрОб, суть якого полягає в створенні та наповненні за визначеними правилам певної інформаційної структури, призначеної для зберігання та зручного використання відомостей, отриманих емпіричним шляхом. Назвемо цю структуру багатовимірною таблицею сумісності вертикального об'єднання (БТСВО).

Технічно БТСВО організована у вигляді багатовимірної таблиці (рис. 5.4), вздовж кожного виміру якої розташовані всі відомі підходи, їх модифікації та технології, які теоретично здатні бути складовими частинами для сполучення в єдину схему розпізнавання за принципом МВрОб, сформульованому на початку підр. 5.5 "Метод вертикального об’єднання5.5".

Кожна комірка, що розташована на перехресті всіх вимірів, містить повну інформацію стосовно об'єднання відповідної комбінації аргументів, а саме:

– чи можуть обрані техніки/технології функціонувати спільно, тобто, чи існує принципова можливість об'єднати відповідні складові в один модуль розпізнавання методом МВрОб;

– якщо так, в якому порядку відбудовуватиметься їх взаємодія, тобто, який напрям буде основним, а які допоміжними;

– які характеристики основного рішення та на який коефіцієнт покращуватимуться допоміжним техніками (мультипликаторами) в

результаті об'єднання, якщо такі коефіцієнти можуть бути розраховані для даної комбінації аргументів.

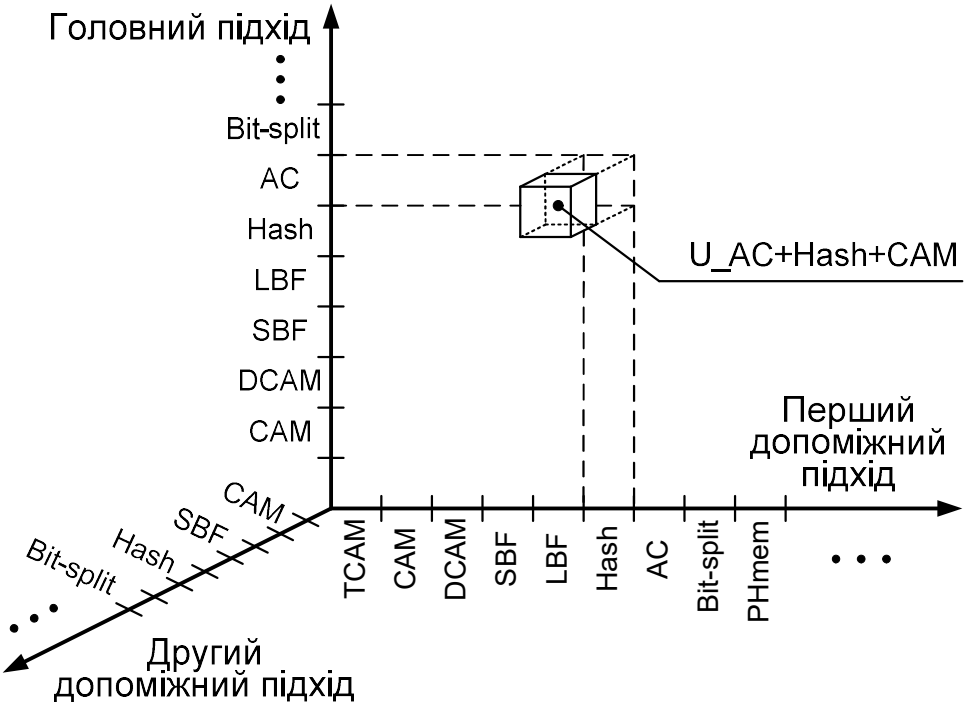


Рис. 5.4. Схематичне подання структури багатовимірної таблиці сумісності вертикального об'єднання

Звичайно, значна частина комірок БТСВО після її заповнення наявної інформацією, залишиться порожніми, тобто в них не буде жодних напрацьованих розробниками відомостей про можливість об'єднання складових, що в ній перетинаються. Проте це явище призводить до прискорення процесу обробки таблиці методом повного перебору.

З іншого боку, кожна порожня комірка таблиці БТСВО означає, що відповідна комбінація досі не використовувалася жодним розробником для побудови МП РАСЗТЗІ, що робить її потенційним джерелом для експериментів з метою пошуку нового ефективного сполучення відомих складових в єдину схему методом МВрОб.

5.5.3. Функціонування методу МВрОб

Використання БТСВО разом з функціями оцінки, розрахованими для

кожного аргументу, дозволяє охопити процес створення МР за методом МВрОб процедурою оптимізації з метою максимізації обраної цільової функції для заданого набору патернів. Змінними параметрами при цьому є технічні рішення – аргументи таблиці БТСВО. Результат оптимізації – певна комірка цієї таблиці, яка містить відомості, що потрібні для створення найефективнішого за наданими умовами компоненту РАСЗТЗІ методом МВрОб.

5.5.4. Реалізація методу МВрОб

Порівняно з МПрКм та МПсКс метод МВрОб, точніше, його принцип дії, розробниками реконфігуровних засобів технічного захисту інформації використовуються найбільш активно. Розглянемо декілька характерних прикладів, присвоюючи назви групам, до яких вони належать згідно складу та взаємозв’язку рішень, які об’єднуються.

5.5.4.1. Об’єднання на базі підходу СА-АК

В роботах [142] та [216] та для підвищення швидкодії скінченного автомату вхідні дані блоками по декілька байтів (які обробляються як окремі символи) розпізнаються фільтрами Блума, в той час як схема в цілому функціонує за алгоритмом Ахо–Корасік. Назвемо подібне об’єднання групою *U_AC+BF* (Union: Aho–Corasik + Bloom Filter), підкреслюючи первинність саме підходу Ахо–Корасік.

В розробці [217] основним також є підхід СА-АК, а асоціативна пам’ять використовується для зберігання переходів автомату, що виконуються найчастіше. Такий варіант поєднання доцільно назвати *U_AC+CAM* (Union: Aho–Corasik + Content-Addressable Memory).

Але частіше об'єднуються не підходи та їх модифікації, а технології, на яких підходи базуються. Наприклад розробка [218] подібна до *U_AC+BF*, але замість ФБ в неї використовуються окремі хеш-функції в поєднанні з асоціативною пам'яттю, тобто, методом МВрОб об’єднуються два

повноцінних підходи (СА-АК та АП) та технологічна основа третього (геш-функції): *U_AC+Hash+CAM*.

Проблема підвищення швидкодії підходу Ахо–Корасік займає багатьох дослідників. У спробах її вирішення вони звертаються в тому числі й до методу МВрОб. В роботі [219] як і в схемі *U_AC+BF* вхідні дані також обробляється блоками по декілька символів, але для їх швидкого розрізняння замість ФБ тут використовується апаратна трійкова асоціативна пам’ять: *U_AC+TCAM* (Union: Aho–Corasik + Ternary Content-Addressable Memory).

Зменшення об’єму ЗП в автоматі АК – також актуальний напрям. В роботі [220] для більш щільного зберігання переходів використовується модифікована АП – так звана mCAM. Відповідно маємо рішення *U_AC+mCAM*.

5.5.4.2. Технічні рішення – мультиплікатори

Як випливає з опису методу МВрОб, при його використанні можуть об'єднуватись ні тільки різні підходи, але й різні модифікації одного підходу. дуже плодючим в цьому сенсі виявився підхід Ахо–Корасік.

В підр. 3.3.10. "Небайтова розрядність обробки даних" була розглянута одна з модифікацій скінченного автомату АК – так званий алгоритм Bit-split. Використовуючи небайтову обробку даних, ця техніка за рахунок зменшення алфавіту символів, що розпізнаються, дозволила певною мірою знизити кількість переходів автомату і, відповідно, об’єм потрібної пам’яті. Але цікавою та корисною властивістю цієї модифікації виявилась ортогональність до інших різновидів алгоритму АК, тобто здатність зменшення потрібної пам’яті на майже однаковий відсоток при сумісному використанні з ними.

Наприклад, автори розробки [196], яка згадувалася в тому ж підрозділі, дослідили сумісність з технікою Bit-split своєї модифікації алгоритму АК, яка за рахунок виявлення збігу інфіксів деяких патернів скорочує потрібну

пам'ять на 21%. Виявилося, що об’єднане цієї модифікації з технікою Bit-split дозволило здобути додатково ще 24% економії ресурсів пам'яті.

В сенсі застосування до методу МВрОб назвемо подібні рішення, що дозволяють в разі застосування до інших рішень зменшити на певний коефіцієнт споживані ресурси, *мультиплікаторами* (тому що вони нібито помножують один з показників ефективності на певний множник).

Введемо також спеціальну нотацію для позначення факту застосування відповідного мультиплікатора до інших рішень метода МВрОб. Наприклад, згадану техніку небайтової обробки позначимо як =Bit-split. Відповідно сумісне використання цього рішення з базовою технологією АК буде мати назву *U_AC=Bit-split*.

Прикладом застосування мультиплікатора =Bit-split може служити розробка [221], в якої для скорочення потрібної пам’яті крім використання АП застосовано також цю модифікацію небайтової обробки. Здобуте рішення згідно введеної нотації має назватися *U_AC+CAM=Bit-split*.

Прикладом сумісного використання технологій АК і ФБ та мультиплікатора є розробка [222], в якої ФБ використовуються для попередньої фільтрації символів вхідних даних з метою запобігання непотрібних переходів в автоматах, що поділені на бітовому рівні. Таке об'єднання має називатися *U_AC+BF=Bit-split*.

Нагадаємо, що відомості про можливість використання деяких технік в якості мультиплікаторів також містяться у комірках БТСВО разом з іншою інформацією о сумісності технічних рішень, що об’єднуються методом МВрОб.

5.6. Комбінування методів комбінування

Аналізуючи три досліджені вище методи комбінування, можна виявити суттєву різницю між їх природою, тобто між способами комбінування технічних рішень. У кожного з цих методів є власні переваги та недоліки, що

наштовхує на природну думку про можливості їх комбінування один з одним, тобто про комбінування методів комбінування (КМК).

На рис. 5.5 наведено приклад комбінування всіх трьох методів: МПрКмб, МПсКс, МВрОб.

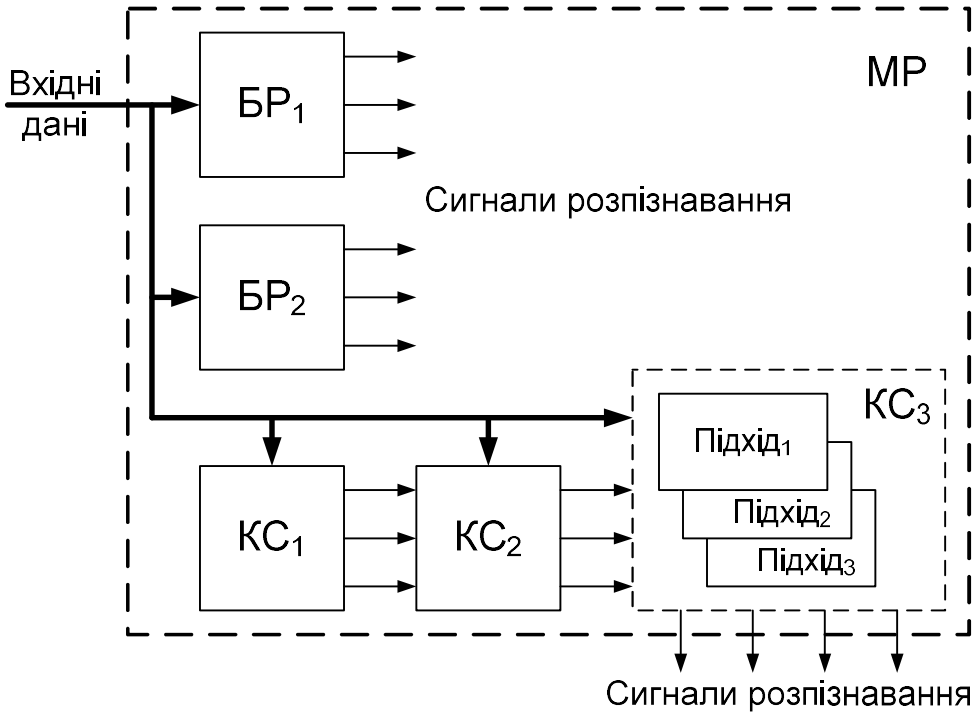


Рис. 5.5. Приклад комбінування методів комбінування

Як бачимо, загальна структура отриманої комбінації в цілому нагадує метод паралельного комбінування, але один з блоків розпізнавання замінено на структуру методу послідовного каскадування, один с каскадів якого в свою чергу є реалізацією метода вертикального об'єднання. При практичній реалізації принципу КМК не виникає труднощів, тому що відповідні алгоритми просто вкладаються один в одного згідно заданої ієрархії.

Завдяки здатності додатково підвищувати ефективність принцип КМК широко застосовується на практиці, в тому числі неявно. Наприклад, фільтр Блума, який потребує уточнення внаслідок системної помилки розпізнавання другого роду, може розглядатися як застосування МПсКс, відтак участь блоку на базі ФБ в будь-якому комбінуванні, фактично є використанням

принципу КМК. Або техніка кластеризації, що застосовується з метою зменшення витрат пам'яті в СА-АК, внаслідок чого один скінченний автомат замінюється низкою інших, більш дрібних, може вважатися неявним використанням методу МПрКм всередині підходу.

Перспективність підходу КМК та його здатність до підвищення ефективності підтверджують реальні розробки, що його використовують. Вже згадуване раніше дослідження [164] насправді є комбінацією МПрКм і МПсКс. Один з паралельно об'єднаних БР тут реалізований як модифікація DpCAM, інший – комбінує методом послідовного каскадування технології гешування та ЦК.

Оскільки будь-який компонент будь-якого методу може рекурсивно містити в себе довільну ієрархію методів комбінування довільної глибини, стає зрозумілим, що в загальному випадку даний процес є нескінченим. Але на практиці ієрархічна структура методів комбінування глибша за три рівні майже не застосовується.

5.7. Висновки до розділу 5

1. Розглянуто критерії та технічні параметри, які можуть бути використані в якості цільових функцій для виконання процедур оптимізації в складі методів комбінування.
2. Сформульовано та досліджено наступні методи комбінування:
   - Паралельного комбінування;
   - Послідовного каскадування;
   - Вертикального об'єднання.
3. Для кожного методу розглянуто:
   - структура методу;
   - функціонування методу;
   - реалізація методу в існуючих розробках;
   - особливості методу.
4. Розглянуто принцип комбінування методів комбінування.

РОЗДІЛ 6. АЛГОРИТМИ ПОБУДОВИ ОПТИМІЗОВАНОЇ СТРУКТУРИ МОДУЛЯ РОЗПІЗНАВАННЯ РАСЗТЗІ

6.1. Алгоритм реалізації методу МПрКм

Згідно визначенню (див. підр. 4.3 "Метод паралельного комбінування") суть метода МПрКм полягає в поділі набору патернів P на n підгруп P_i та вибору з бібліотеки готових компонентів такої ж кількості різних блоків розпізнавання БР$_i$ таким чином, щоб при їх з'єднанні по схемі МПрКм функція оцінки, розрахована для отриманого МР згідно заданому критерію оптимізації, досягла значення, максимально наближеного до оптимального.

Алгоритм вирішення задачі побудови МР за методом МПрКм в такої постановці має містити три вкладених цикли:

1. Цикл кількості БР – варіюється кількість n блоків розпізнавання БР$_i$ в діапазоні від 2 до $n_{\max}$, де $n_{\max}$ – максимальне значення n, задане користувачем.

2. Цикл вибору БР – з бібліотеки компонентів обираються схеми БР$_i$, побудовані за різними підходами, модифікаціями та техніками.

3. Розбиття патернів – варіюються різні комбінації розбиття набору патернів P на n підгруп.

Для кожної комбінації на кожному циклі для кожного БР$_i$ обчислюються ФО, з яких згідно заданого критерію оптимізації формується відповідна цільова функція. Кінцева мета алгоритму – знайти комбінацію, для якої значення цільової функції є найбільшим або найменшим в залежності від критерію.

6.1.1. Проблема розбиття патернів у методі МПрКм

Перший та другий цикли з вищеперелічених нескладно реалізувати методом повного перебору внаслідок невеликого числа змінних. З третім циклом ситуація виявляється складнішою.

Кількість варіантів розбиття множини патернів P, що налічує σ патернів $p_1, p_2, \ldots p_k, \ldots\ p_\sigma$ між блоками розпізнавання в кількості n одиниць БР$_1$, БР$_2$, … БР$_i$, … БР$_n$ в загальному випадку дорівнює

$$W_\sigma = n^\sigma .$$

Для баз даних сигнатур сучасних РАСЗТЗІ, які налічують до сотень тисяч патернів, кількість варіантів поділу W_σ навіть між двома БР$_i$, є комбінаторно великим числом, що виключає можливість виконання повного перебору. дещо знизити його дозволяє оперування не окремими патернами, а пакетами патернів однакової довжини. (Якщо знехтувати ефектом самоподоби, розпізнавання патернів однакової довжини призводить до однакових значень функції оцінки). Тоді кількість варіантів розбиття множини патернів P, що налічує ξ пакетів по δ_j патернів однакової довжини, між блоками розпізнавання в кількості n одиниць знизиться до

$$W_\xi = n^\xi .$$

Але це зменшує ступень числа варіантів розподілу лише на один – два десяткові порядки. Тобто число W_ξ також є завеликим для практичної реалізації процедури оптимізації методом повного перебору.

Використання інших методів оптимізації, наприклад, покоординатного спуску, унеможливлюється в зв’язку з відсутністю будь-якої метрики в просторі варіантів, які є комбінаторною комбінацією змінних.

6.1.2. Метод прискорення процедури оптимізації для МПрКм

З метою надання множині комбінацій метричних здібностей та скорочення часу розрахунків до прийнятних значень, пропонується ***Метод прискорення процедури оптимізації***, суть якого полягає у сортуванні

патернів в наборі за певними параметрами. Тоді замість повного перебору можна використати послідовне збільшення значення такої метрики від найменшої до найбільшої межі, внаслідок чого складність процесу обчислень зменшується від комбінаторного до лінійного закону. Звичайно, такий підхід не гарантує досягнення глобального екстремуму, проте за рахунок скорочення об'єму обчислень з комбінаторного до лінійного закону призводить до реалізації процедури оптимізації за прийнятний час.

Реалізація цього підходу призводить до наступного алгоритму.

Алгоритм прискореного розбиття набору патернів для методу МПрКм. Для $n = 2$ на першому кроці блокові $БР_1$ призначуються всі патерни набору P, тоді як блокові $БР_2$ – жодного, тобто пусту множину патернів. На наступному кроці блокові $БР_1$ призначуються всі патерни набору крім першого за обраним параметром сортування патерну, який надається для розпізнавання блокові $БР_2$. На наступному кроці вже два патерни – перший та другий за обраним параметром сортування – переходять від $БР_1$ до блоку $БР_2$. процедура повторюється до ситуації, коли в підмножині, яку розпізнає блок $БР_1$ не залишається жодного патерну, а блок $БР_2$ отримує для розпізнавання весь набір патернів P. У випадку, коли кількість блоків n більша за 2, алгоритм можна ускладнити рекурсивно, розглядаючи один з блоків як такий, що складається з двох інших блоків, і так далі.

Під час проведення експериментів в даному розділі застосовувалися три види впорядкування, засновані на використанні функції розподілу довжин δ_j:

- попатерне сортування за довжиною патернів *Pattern_L*;
- попакетне сортування за кількістю патернів у пакеті *Delta_j*;
- попакетне сортування за "площею" пакету (тобто здобутку кількості патернів у пакеті на їх довжину) *Pat_x_Del*,

хоча алгоритм дозволяє використовувати й більш складні види сортування, включаючи урахування функцій самоподоби.

6.2. Алгоритм реалізації методу МПсКс

Алгоритм реалізації методу МПсКс включає такі ж три цикли, що й МПрКм. Але на третьому циклі поділ набору патернів для даного методу здійснюється по довжіні патернів. Тобто до першого каскаду $КС_1$ відходять для розпізнавання префікси всіх патернів, що містять перши L_1 символи довжиною від $m_{\min}$ до $(m_{\min} + L_1 - 1)$, а також патерни довжиною менше ніж L_1; до другого каскаду $КС_1$ – інфікси довжиною в L_2 символи, а саме, з $(m_{\min} + L_1)$-го по $(m_{\min} + L_1 + L_2 - 1)$-й символи, а також суфікси патернів довжиною менш ніж L_2, і так далі … ; до останнього каскаду $КС_n$ відходять суфікси довжиною $L_{n\text{-}1}$ символи, а саме, з $(m_{\max} - L_{n\text{-}1})$-го по $m_{\max}$ -й символи, а також суфікси патернів довжиною від $(m_{\max} - L_{n\text{-}1})$ до $m_{\max}$.

В обчислювальному сенсі цикл розбиття патернів для методу МПсКс виявляється істотно простішим, тому що не потребує перебору комбінаторно великої кількості варіантів. Для знаходження оптимуму для даного методу потрібно знайти в загальному випадку всього $n - 1$ питому змінну – довжини L_1, L_2, … $L_{n\text{-}1}$ фрагментів патернів, на які відбуватиметься розбиття патернів. У найпоширенішому на практиці випадку, коли $n = 2$, метод вироджується в пошук тільки однієї величини L, по якої патерні поділяються на префікси та інфікси між першим та другим каскадами відповідно.

6.3. Алгоритм реалізації методу МВрОб

Строго кажучи, МВрОб не є повноцінним методом комбінування в тому сенсі, що він не передбачає розподіл набору патернів на підгрупи. Тому алгоритм його реалізації складається лише з двох перших циклів, подібних до тих, що згадані у підр. 6.1 "Алгоритм реалізації методу МПрКм". На першому з них обирається розмірність n методу, тобто кількість підходів / модифікацій / допоміжних технік, що будуть об'єднуватися даним методом. На другому циклі з таблиці БТСВО виокремлюється підмножина комірок відповідної розмірності, з яких обирається готова реалізація блоку

розпізнавання, оптимальна за заданим критерієм.

6.4. Алгоритм реалізації принципу КМК

Принцип комбінування методів комбінування при його практичній реалізації передбачає два вкладених цикли – вибір глибини ієрархічної структури на першому та перебір варіантів комбінування на другому. Як свідчить досвід дослідників, глибина КМК більша за три рівні не надає значних переваг, проте суттєво ускладнює вирішення задачі оптимізації.

6.5. Загальна структура використання методів комбінування

На рис. 6.1 наведено загальне схематичне подання використання розглянутих в даному дослідженні методів та алгоритмів.

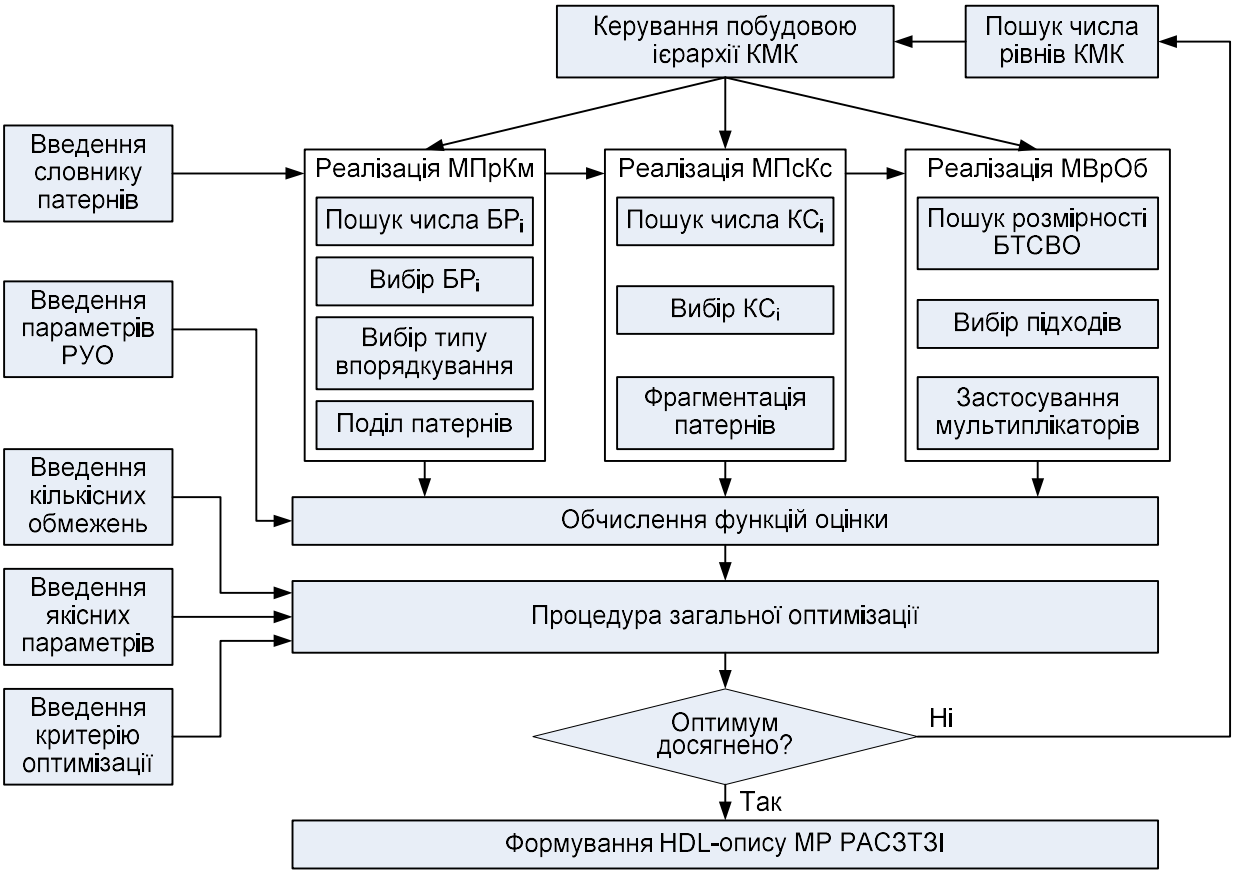


Рис. 6.1. Загальна структура використання методів комбінування

Первинною інформацією для побудови МП РАСЗТЗІ є набір патернів, які потрібно розпізнавати. До вхідних даних належать також тип прискорювача, що використовується, кількісні обмеження та якісні вимоги до системи захисту, що проектується, а також критерій, за яким потрібно досягти оптимуму.

Процедура загальної оптимізації на найвищому рівні охоплює цикл вибору глибини ієрархії методів комбінування за принципом КМК та формування її конкретної структури. На наступному рівні окремо розраховуються кожний з методів комбінування, включаючи всі цикли, що потрібні для кожного з них. Всередині кожного методу для обраних бібліотечних блоків згідно поточному варіанту розбиття набору патернів знаходяться ФО, зі значень яких згідно поточного варіанту КМК обчислюються характеристики результуючої схеми. Після знаходження оптимального рішення згідно заданого критерію формується специфікація обчислювальної структури на якоїсь з мов опису апаратури, тобто, HDL-опис модулю розпізнавання.

Нагадаємо, що весь комплекс алгоритмів, наведений на рис. 6.1, необхідно виконувати при проведенні кожної процедури оперативного оновлення. Тому важливе значення набувають методи прискореного обчислення, запропоновані в даному дослідженні.

6.6. Експериментальне програмне забезпечення

Для проведення обчислювальних експериментів щодо кількісної оцінки запропонованих методів і засобів шляхом проведення обчислювальних експериментів було розроблено Експериментальний програмний комплекс FPGA_Pattern_Matching (розпізнавання патернів на базі ПЛІС) – *ЕПК FPM*.

На рис. 6.2 наведено загальний вигляд головного вікна програмного комплексу.

Перелік основних функцій ЕПК FPM включає наступні пункти.

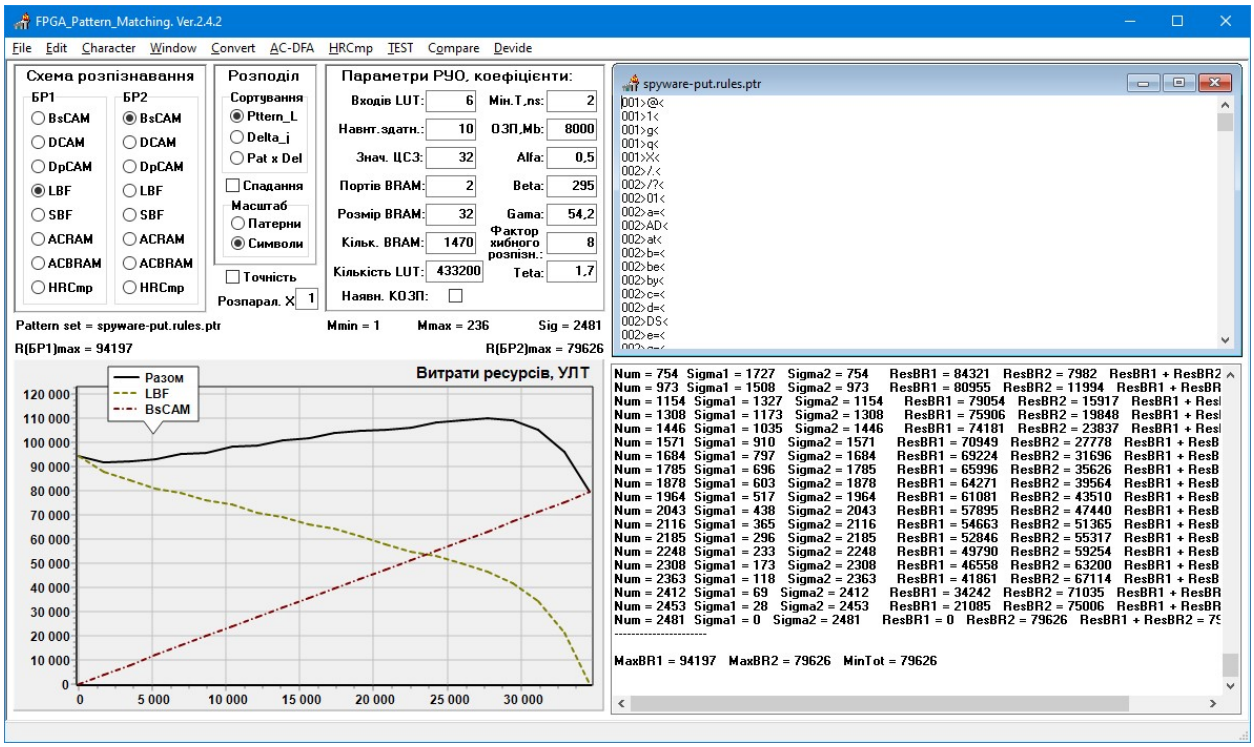


Рис. 6.2. Експериментальний програмний комплекс FPGA_Pattern_Matching

1. Вилучення, аналіз та сортування патернів з баз даних сигнатур найбільш поширених відкритих сигнатурних засобів технічного захисту.
2. Введення значень характеристик реконфігуровного прискорювача, який використовується в складі РАСЗТЗІ:
   - число входів ЛТ мікросхеми ПЛІС, що задіяна в РУО;
   - здатність навантаження (fan-outmax) логічних елементів ПЛІС;
   - максимальне значення затримки ЦСЗ, створюваної на бази ЛТ;
   - кількість портів блокової пам’яті ПЛІС;
   - розмір блоків пам’яті BRAM в мікросхемі ПЛІС;
   - кількість блоків блокової пам’яті BRAM в ПЛІС;
   - кількість ЛТ в ПЛІС;
   - максимально можлива швидкодія (мінімальна затримка) ПЛІС;
   - об’єм бортової пам’яті РУО;
   - опція наявності вбудованого контролера ОЗП.
3. Введення також наступних величин:

- коефіцієнти зведення ресурсів різного типу до УЛТ;
- деякі кількісні параметри схем розпізнавання, а саме: фактор хибного розпізнавання для підходів на базі фільтра Блума та коефіцієнт надмірності заповнення ПЗП для схеми HRCmp.

4. Завдання типу сортування патернів:
   - за довжиною патернів Pattern_L;
   - за кількістю патернів у пакеті Delta_j;
   - за "площею" пакету Pat_x_Del,

   як за зростанням, так і за спаданням.
5. Завдання розпаралеленого режиму обчислень для $БР_2$ із зазначенням кратності розпаралелювання.
6. Завдання режиму підвищеної точності розрахунків.
7. Обчислення ФО двох блоків розпізнавання ($БР_1$ і $БР_2$) для найбільш поширених підходів до побудови апаратних схем РАСЗТЗІ та їх найефективніших модифікацій:
   - базова схема на АП BsCAM;
   - модифікована схема на АП DCAM;
   - модифікована схема на АП DpCAM;
   - повнорозмірний фільтр Блума LBF;
   - спрощений фільтр Блума SBF;
   - базова схема СА-АК із зовнішньою пам'яттю ACRAM;
   - базова схема СА-АК із блоковою пам'яттю ACBRAM;
   - двокаскадна схема HRCmp.
8. Дослідження властивостей та аналіз ресурсного складу БР, побудованих згідно підходів та їх модифікацій з вищенаведеного переліку.
9. Порівняння поведінки ПЕ питомих підходів попарно та декількох разом.
10. Вирішення оптимізаційної задачі для методу паралельного комбінування двох схем, побудованих за довільним підходом з

вищенаведеного переліку.

11. Вирішення оптимізаційної задачі для методу послідовного каскадування на прикладі двокаскадної схеми HRCmp.
12. Відображення:
    - основних кількісних характеристик набору патернів;
    - змісту набору патернів у відсортованому вигляді;
    - проміжних та остаточних результатів розрахунків в цифровому вигляді;
    - результатів розрахунків в графічному вигляді.
13. Вибір типу масштабування результатів розрахунків у графічному вигляді по осі абсцис.

Експериментальний комплекс ЕПК FPM створено в середовищі програмування Embarcadero Delphi 10.3 Version 26.0.34749.6593 Community Edition.

Нижче наведено опис експериментів та результатів, отриманих під час їх проведення.

6.7. Проведення експериментів

6.7.1. Низка експериментів № 1. Комбінування методом МПрКм

Розглянемо результати декількох розрахунків з метою перевірки працездатності та особливостей застосування методу МПрКм.

На рис. 6.3 наведено результати розрахунків для прикладу мінімізації ресурсних витрат методом МПрКм за наступних умов:

<u>Експеримент 1.1 (рис. 6.3):</u>

- $n = 2$;
- реалізація БР1 – за схемою BsCAM;
- реалізація БР2 – за схемою LBF;
- впорядкування – Pat_x_Del (за спаданням);
- набір патернів P використаний з вільно розповсюджуваної бази даних сигнатур "Community Ruleset" (rev:50 від 12.07.18) відкритої

МСВВ Snort версії 3.0, що містить $\sigma = 4208$ патернів довжиною від $m_{\min} = 1$ до $m_{\max} = 364$ загальною кількістю $\Omega = 82081$ символ;

– в якості РУО використано модель VC709 Kit оціночного набору (Evaluation Kit) від фірми Xilinx на базі ПЛІС Virtex-7 VX690T від Xilinx (див. табл. 1.4 і табл. 4.2).

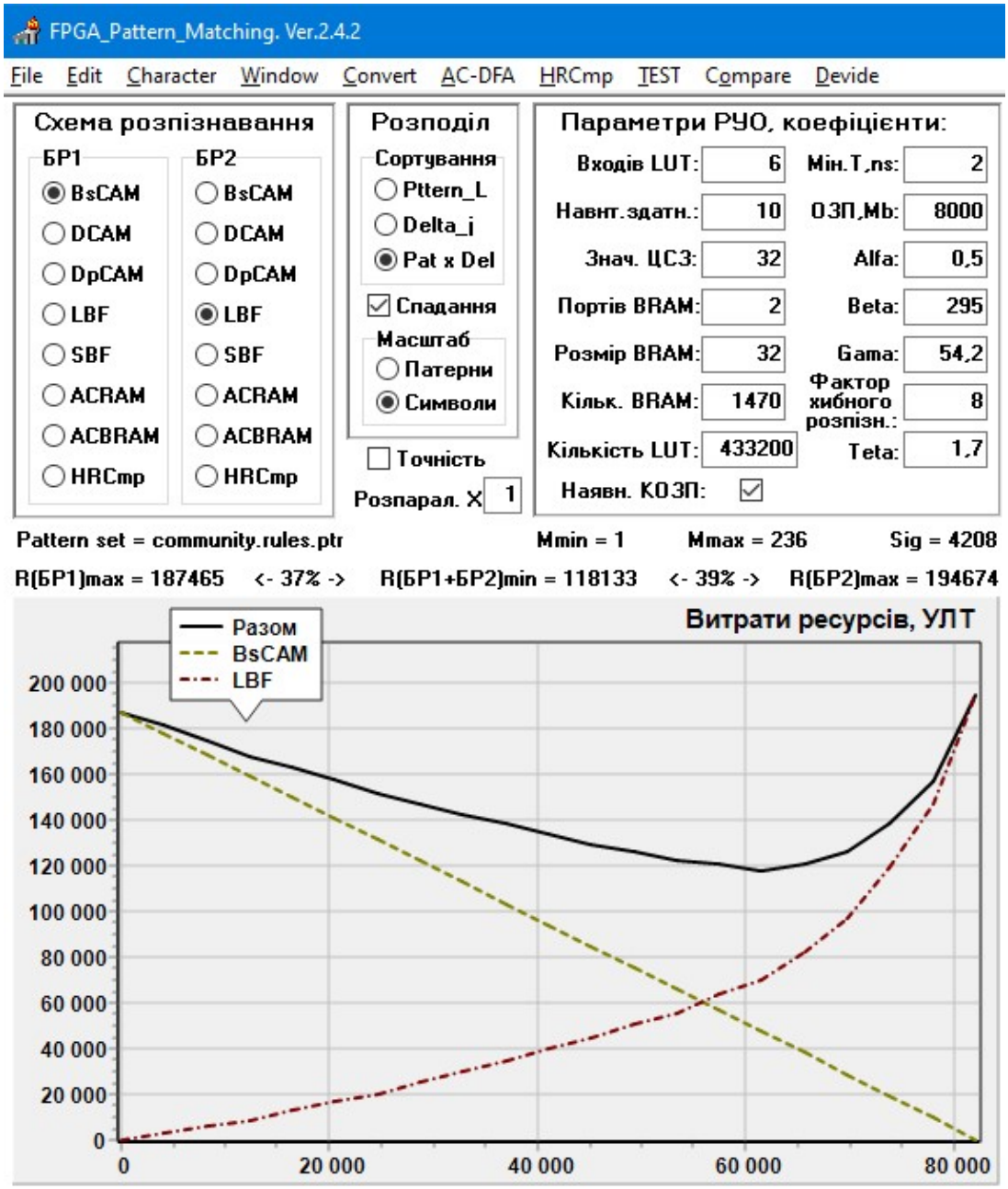


Рис. 6.3. Вдале комбінування методом МПрКм базових схем розпізнавання на основі асоціативної пам'яті BsCAM та на фільтрах Блума LBF

(Всі потрібні для розрахунків відомості щодо обраних для $БР_1$ та $БР_2$ схем розпізнавання, типу сортування, параметрів реконфігуровного прискорювача та набору патернів (Pattern set) можна побачити у верхній частині вікна ЕПК).

По осі абсцис тут відображається кількість символів у патернах, які передані на розпізнавання з блоку $БР_1$ до блоку $БР_2$. По осі ординат – витрати в умовних ЛТ на створення кожного з БР та разом – на весь МР, що з них складається.

Спадаюча крива штриховою лінією відображає зниження витрат на створення блоку $БР_1$ по мірі зменшення числа патернів у підгрупі, яку він розпізнає. Її значення у крайній лівій точці (187465 УЛТ) означає, що саме стільки ресурсів потрібно, якщо базова схема на асоціативній пам'яті АП обробляє усі патерни набору.

Зростаюча крива штрих-пунктирною лінією відображає зріст витрат на створення блоку $БР_1$ по мірі збільшення числа патернів у підгрупі, яку розпізнає він. Її значення у крайній правій точці (194674 УЛТ) показує, скільки ресурсів потрібно, якщо усі патерни набору базова обробляє базова схема на фільтрі Блума.

Суцільна лінія показує сумарні витрати на обидва блоки розпізнавання. Як можна бачити, крива витрат на весь МР має виражений мінімум, при якому сумарні витрати (118133 УЛТ) на 37% менші за витрати на $БР_1$, та на 39% – на $БР_2$, якщо б вони розпізнавали весь набір патернів поодинці. Тобто паралельне комбінування цих двох блоків призводить до підвищення показника ефективності за ресурсними витратами на 37% або 39% порівняно з одиночним використанням кожного з підходів.

Наведений на рис. 6.3 приклад підтверджує працездатність методу МПрКм, оскільки демонструє вдалий результат паралельного комбінування. Але в даному прикладі використовуються базові схеми відповідних підходів, які не є найефективнішими у відповідних напрямках. Тому виникає слушне запитання – чи можливо досягти настільки ж вдалого результату, якщо

використовувати більш ефективні модифікації підходів? Чи не є зниження витратності комбінованої схеми результатом недосконалості використаних в неї блоків розпізнавання?

На рис. 6.4 наведено результати розрахунків для прикладу мінімізації ресурсних витрат методом МПрКм за наступних умов:

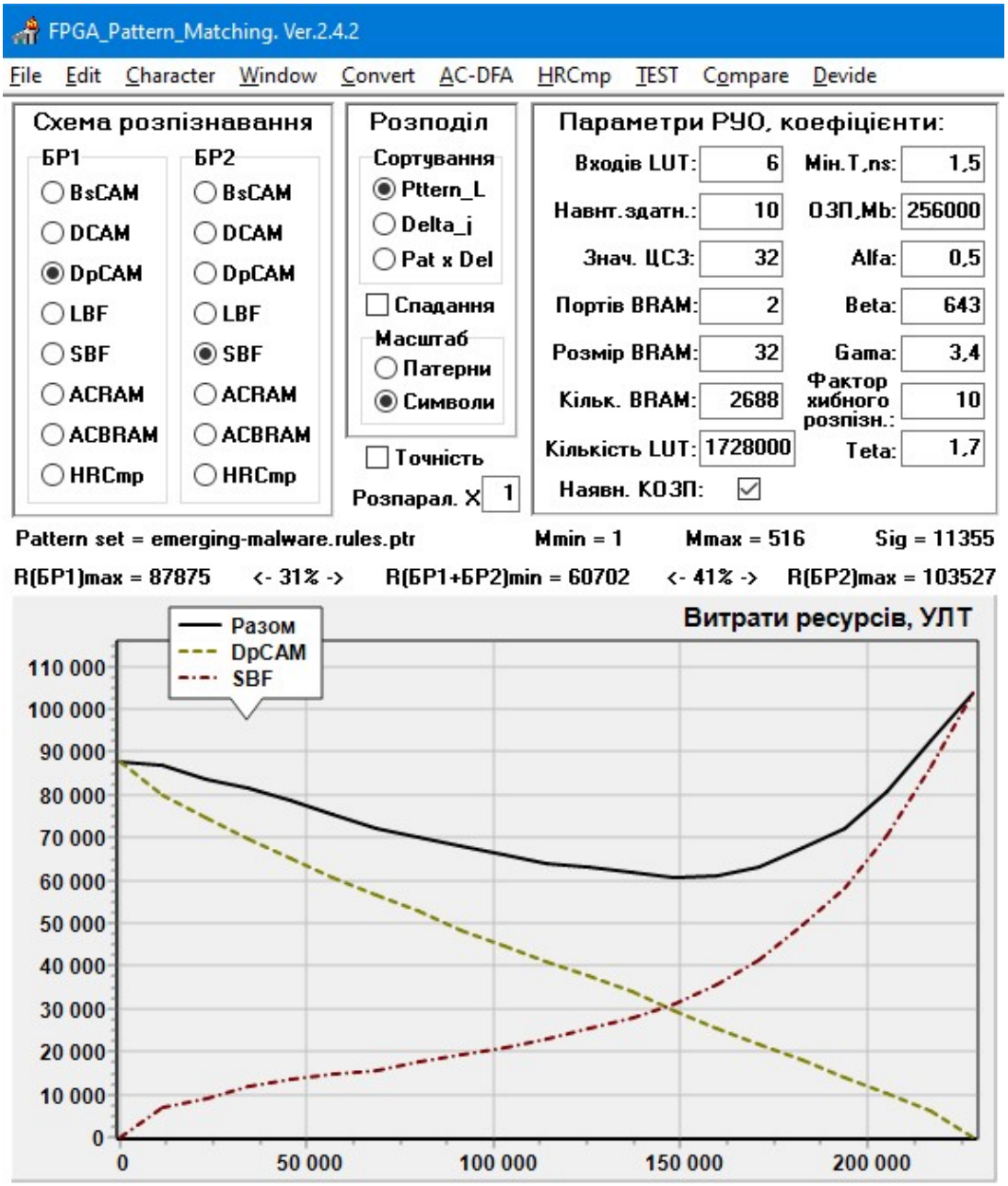


Рис. 6.4. Вдале комбінування методом МПрКм модифікації підходу на основі асоціативної пам'яті DpCAM та спрощеної схеми на фільтрах Блума SBF

<u>Експеримент 1.2 (рис. 6.4):</u>

- n = 2;
- реалізація $БР_1$ – за схемою DpCAM;
- реалізація $БР_2$ – за схемою SBF;
- впорядкування – Pattern_L;
- набір патернів P використаний з вільно розповсюджуваної бази даних сигнатур "Malware Rules" відкритої МСВВ Suricata версії 5.0, що містить σ = 11355 патернів довжиною від $m_{\min} = 1$ до $m_{\max} = 236$ загальною кількістю Ω = 228206 символів;
- в якості прискорювача тут використано повноцінний РУО XUP-VV8 від фірми Nallatech на базі ПЛІС Virtex UltraScale+ VU13P від Xilinx.

Як можна бачити, схеми, що використані в даному прикладі розпізнають значно більшу кількість патернів, витрачаючи при цьому в декілька разів менший об'єм ресурсів. Але в даному випадку крива сумарних витрат на весь МР також має виражений мінімум, при якому комбінована схема потребує ресурсів на 31% менше за схему DpCAM (яка є модифікацією базової схеми на основі асоціативної пам'яті) та на 41% менше за схему SBF (яка є спрощеною модифікацією базової схеми на фільтрах Блума). Отже, метод МПрКм працює для ефективних модифікацій також.

Але на практиці вдалий результат комбінування, подібний до наведених на рис. 6.3 та на рис. 6.4, виходить рідко. Набагато частіше один з блоків, що комбінуються, вздовж всього виміру варіювання демонструє кращі значення окремо, ніж сумісно з іншим. Подібний приклад наведено на рис. 6.5.

Тут також задіяні схеми розпізнавання DpCAM та SBF. Але решта умов інші:

- РУО типу 385A від фірми Nallatech на базі ПЛІС Arria 10 GX 1150 від Altera (зараз – Intel);
- впорядкування – Pat_x_Del (за спаданням);

– набір патернів P – з бази даних сигнатур "Protocol-FTP rules" МСВВ Snort версії 3.0, яку складено лише з сигнатур атак на FTP-сервери локальних обчислювальних мереж, що містить $\sigma = 97$ патернів довжиною від $m_{\min} = 1$ до $m_{\max} = 75$ загальною кількістю $\Omega = 726$ символів.

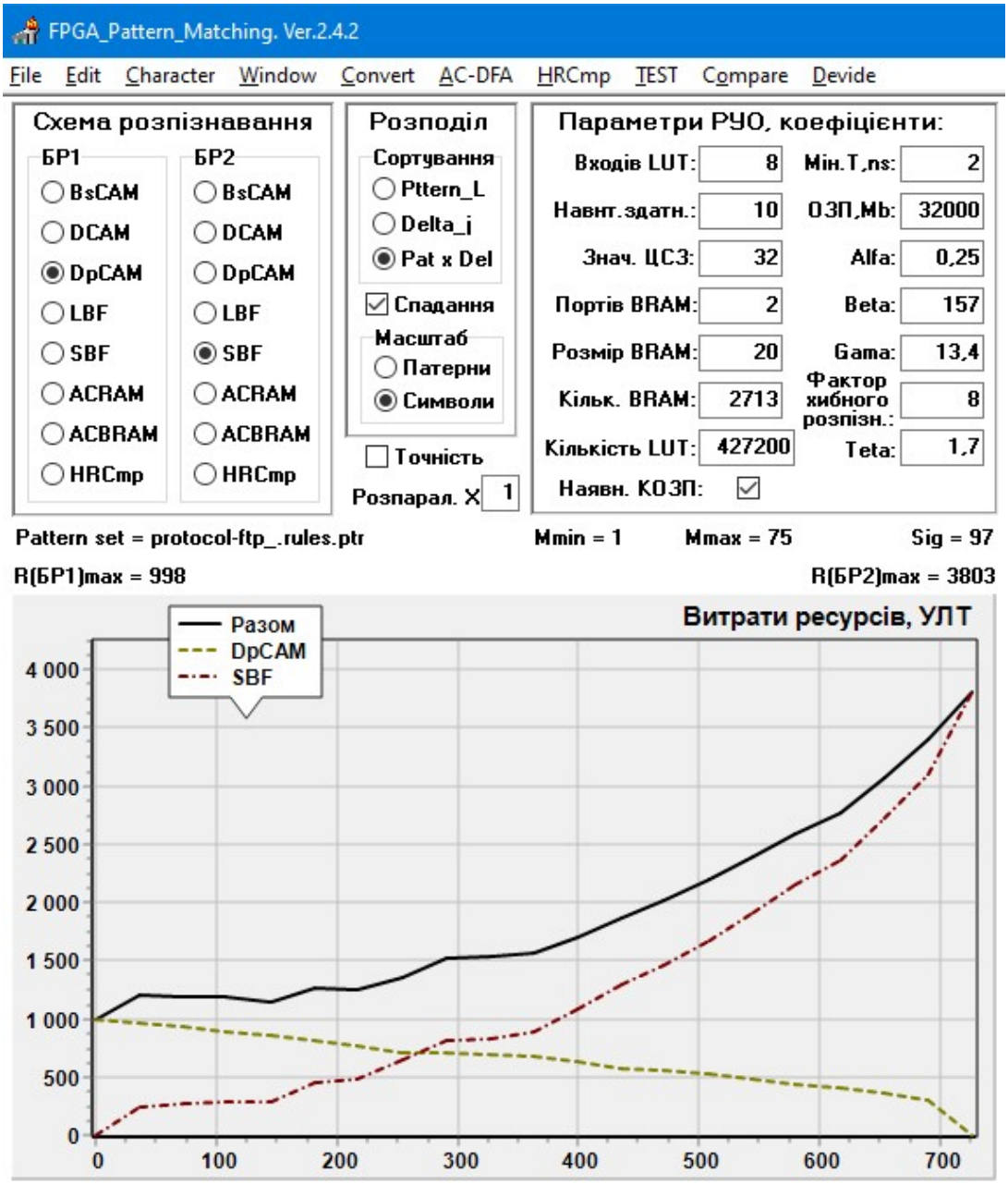


Рис. 6.5. Невдале комбінування методом МПрКм двох схем розпізнавання

За таких умов витрати на створення $БР_1$ у граничному випадку, тобто коли він розпізнає весь набір патернів, значно менші, ніж на створення $БР_2$ в

разі розпізнавання того ж самого набору патернів. Як наслідок, крива сумарних витрат не має мінімуму. Тобто для будь-якої комбінації поділу патернів між двома БР, схема DpCAM виявляється кращою за комбіновану схему.

Численні експерименти з різними підходами, РУО та базами сигнатур підтверджують, що ситуація, яку відображено на рис. 6.5 при практичному застосуванні РАСЗТЗІ зустрічається значно частіше за таку, яку наведено на рис. 6.3 та рис. 6.4.

Цей факт можна розглядати як такий, що пояснює, чому на практиці метод МПрКм рідко використовується окремо від інших методів комбінування. З'ясуємо, чи існують інші причини, що ускладнюють використання МПрКм.

Аналізуючи та узагальнюючи ситуації, подібні до наведених на рис. 6.3 та рис. 6.4 прикладів, можна сформулювати відповідний чинник вдалого комбінування методом МПрКм наступним чином.

Чинник 1 вдалого комбінування методом МПрКм: чисельні значення показників ефективності БР$_i$, що комбінуються, мають бути достатньо близькими за абсолютними значеннями. Це стосується як ресурсних, так и часових ПЕ. Інакше кажучи, для паралельного комбінування потрібно узгоджувати швидкісні характеристики задіяних блоків розпізнавання, побудованих за різними підходами. На практиці це інколи вдається досягти шляхом розпаралелювання більш повільних схем розпізнавання.

Але сформульований вище чинник не є єдиною причиною невдалого паралельного комбінування. Тобто від є необхідною, але не достатньою умовою. Продовжуючи дослідження в цьому напрямку, можна сформулювати ще одну необхідну умову запобігання невдачі для МПрКм, яка формулюється наступним чином.

Чинник 2 вдалого комбінування методом МПрКм: функціональної залежності показників ефективності БР$_i$, що комбінуються, від поділу патернів мають бути подібними.

Пояснимо це твердження також на наведених вище прикладах вдалого комбінування. Якщо придивитися до характеру поведінки кривих вдовж осі абсцис на рис. 6.4, можна звернути увагу на те, що спадаюча та зростаюча крива поводяться по-іншому. Графік витрат на схему DpCAM майже лінійний, в той час як графіку витрат на схемі SBF притаманне значне викривлення. Зрозуміло, що якби обидві криві поводилися однаково, їх сума не мала би ніякого екстремуму.

На рис. 6.6 наведено порівняння двох кривих ресурсних витрат – для модифікації DCAM підходу на базі АП та ЦК та спрощеного варіанту реалізації схеми на базі фільтра Блума SBF (для бази даних сигнатур " Community Ruleset " при впорядкуванні за типом Pat_x_Del). Тут вздовж осі абсцис кількість та склад патернів, що підлягають розпізнаванню, змінюються однаково для обох підходів.

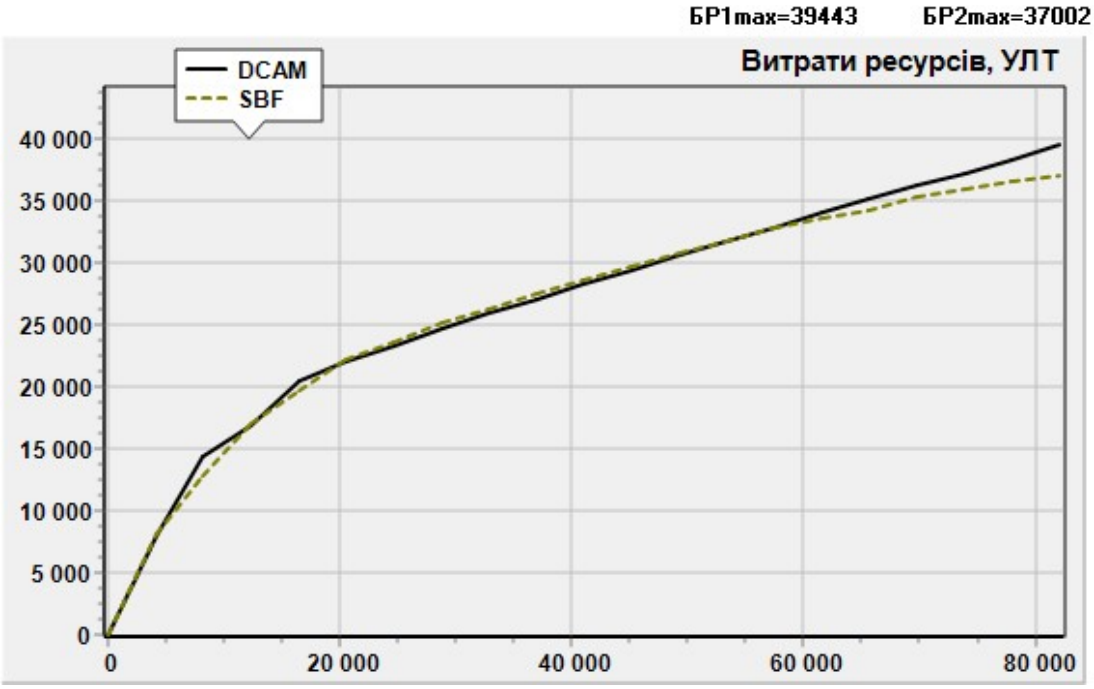


Рис. 6.6. Порівняння поведінки схем, побудованих за різними підходами: DCAM і SBF (впорядкування Pat_x_Del)

Як можна бачити, характер поведінки обох графіків вздовж всього виміру варіювання майже не відрізняється. Тому можна впевнено

стверджувати, що їх паралельне комбінування за наданих умов не призведе до позитивного ефекту в сенсі зменшення витрат. Незважаючи на близькість абсолютних значень, тобто на наявність Чиннику 1 вдалого комбінування методом МПрКм.

Зауважимо, що цей експериментальний результат розбігається з теоретичним припущенням щодо наявності суттєвих відмінностей поведінки схем розпізнавання, побудованих за різними за своєю природою підходами, яке було покладено в основу дослідження методів комбінування.

Більш того, також всупереч цьому теоретичному припущенню виявляється, що з іншого боку – навпаки, різні модифікації одного й того ж підходу можуть суттєво розрізнятися. На рис. 6.7 зведено разом за однакових умов характеристики модифікацій DCAM та DpCAM єдиного підходу до побудови МР РАСЗТЗІ – на базі АП та ЦК.

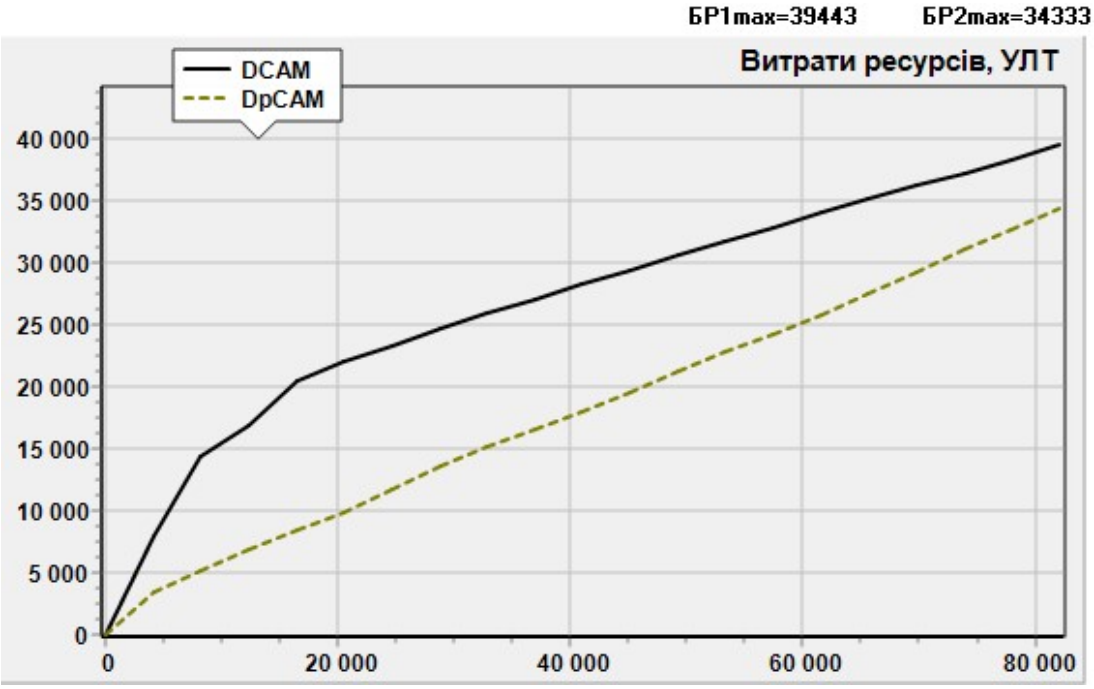


Рис. 6.7. Порівняння поведінки схем, побудованих за спорідненими підходами: DCAM і DpCAM (впорядкування Pat_x_Del)

Як можна бачити, попри близькість внутрішній структури схем DCAM та DpCAM (які є відповідно декодованою та частково декодованою

модифікаціями базової схеми BsCAM, чиї внутрішні структури майже не відрізняються), поводяться вони на одних й тих самих наборах даних суттєво по-різному. Тому їх об'єднання дає позитивний ефект при паралельному комбінуванні.

Ці факти розбіжності фактичних результатів з теоретичними припущеннями свідчать про важливість експериментальних досліджень для виконання даної роботи.

Розглянемо ще два розрахунки, мета проведення яких не пов'язана з аналізом результативності методу МПрКм.

На рис. 6.8 та рис. 6.9 наведено пари характеристик тих самих схем, що й на рис. 6.6 та рис. 6.7, за тими ж самими умовами, але впорядковані за іншим типом сортування – Pattern_L.

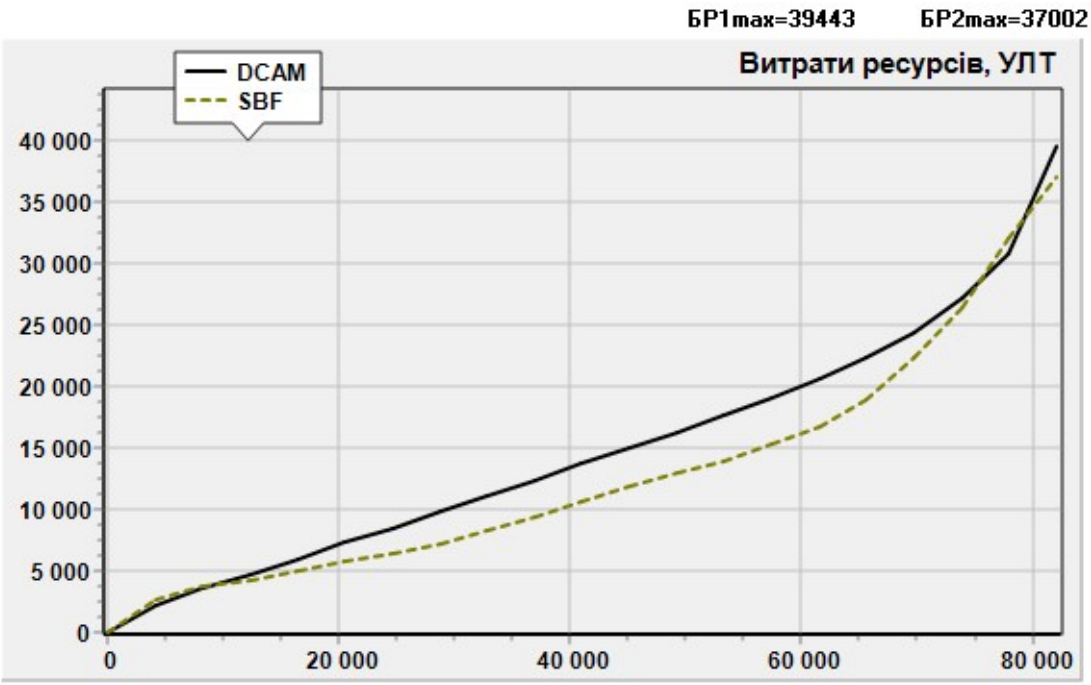


Рис. 6.8. Порівняння поведінки схем, побудованих за різними підходами: DCAM і SBF (впорядкування Pattern_L)

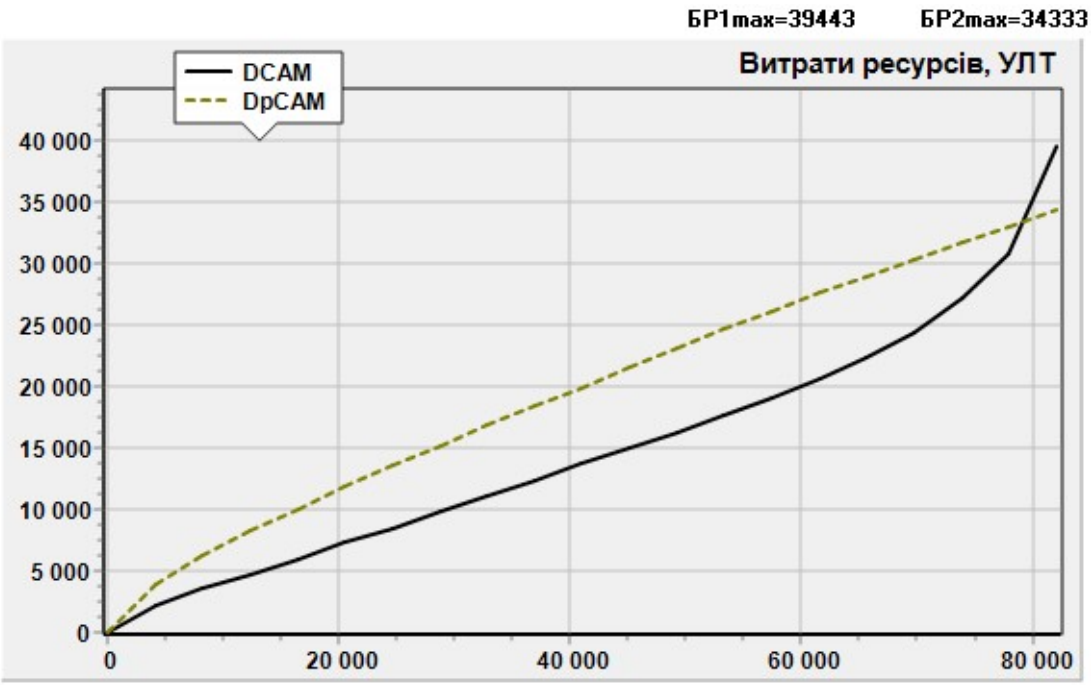


Рис. 6.9. Порівняння поведінки схем, побудованих за спорідненими підходами: DCAM і DpCAM (впорядкування Pattern_L)

Якщо для першої пари зміна типу впорядкування не вплинула на близькість поведінки відповідних схем, то для пари DCAM і DpCAM, якщо порівняти з кривими на рис. 6.7, зміна типу впорядкування призвела до принципової різниці: для "площинної" метрики Pat_x_Del схема DCAM значно гірше схеми DpCAM за витратами протягом всього діапазону варіювання числа патернів у підгрупі, тоді як при сортуванні за довжиною патернів (тип Pattern_L) перша схема, навпаки, демонструє меншу витратність майже на всьому діапазоні.

Висновок, який можна зробити за результатами аналізу кривих на рис. 6.6-6.9, полягає у тому, що вибір типу впорядкування впливає на результат застосування методу МПрКм. Цей факт можна враховувати при використанні паралельного комбінування.

Після тимчасового відхилення від проблеми результативності паралельного комбінування знов повернемося до даної теми.

Врахування розглянутого вище Чинника 2 вдалого комбінування методом МПрКм дозволяє прискорити процедуру побудови оптимальної

схеми МР РАСЗТЗІ за рахунок виключення з переліку варіантів, що аналізуються в процесі виконання процедури оптимізації таких комбінацій, які гарантовано не призведуть до позитивного ефекту. З цією метою доцільно заздалегідь порівняти між собою поведінку наявних в бібліотеці компонентів БР$_i$ за різними типами впорядкування патернів та виявити пари блоків, що мають подібні характеристики.

На рис. 6.10 наведено характеристики п'яті різних БР, обчислених, використовуючи впорядкування типу Pattern_L за спаданням. Як можна бачити, вони всі мають різні поведінки, тому комплект відповідних блоків розпізнавання можна використовувати для методу МПрКм, не втрачаючи часу на гарантовано невдалі варіанти.

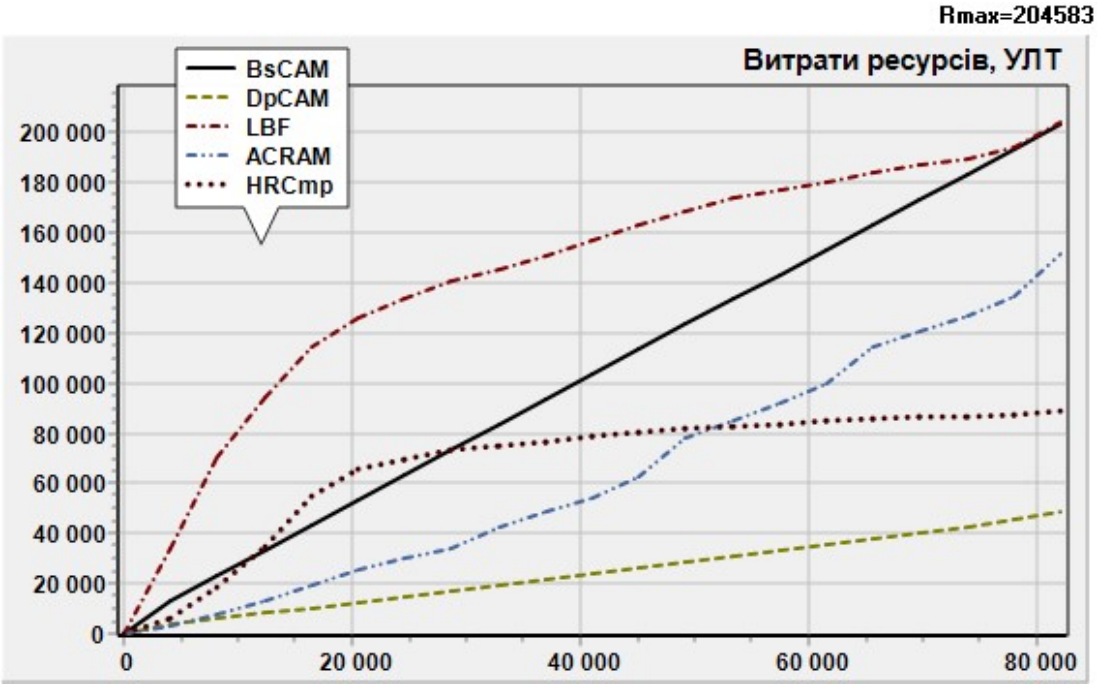


Рис. 6.10. Попереднє порівняння різних схем розпізнавання (впорядкування типу Pattern_L за спаданням)

Завершуючи розгляд експериментального дослідження методу МПрКм, зауважимо, що у випадках, коли по результатах виконання розрахунків не вдається отримати підвищення ефективності від паралельного комбінування декількох різнорідних схем, результат є також прийнятним, оскільки

досягнуто головну мету застосування методів комбінування – виявлено найбільш ефективну за наданих умов схему розпізнавання для МР РАСЗТЗІ. Фактично, отриманий результат є виродженим випадком МПрКм для $n = 1$, швидко здобутим завдяки застосуванню розроблених в даному дослідженні методів прискореного обчислення технічних характеристик схем розпізнавання.

6.7.2. Експеримент № 2. Комбінування методом МПсКс

Як було вказано раніше, алгоритм реалізації методу МПсКс в обчислювальному сенсі значно простіше за МПрКм, тому що не потребує перебору комбінаторно великої кількості варіантів.

Експериментальні дослідження підтвердили, що оптимум для методу МПсКс впевнено знаходиться для тих комбінацій схем розпізнавання, що застосовуються на практиці при його використанні. На рис. 6.11 наведено графік залежності від довжини префіксу L ресурсних витрат (з розшифровкою їх складу) на побудову двокаскадної послідовної схеми комбінування HRCmp. Ця схема використовує в першому каскаді технологію хешування без створювання повноцінної схеми ФБ, а в другому – ПЗП і ЦК без створення повної схеми АП (див. рис. 4.8 у підр. 4.6.4 "Техніка створення функцій оцінки для двокаскадної схеми на базі геш-функції, ПЗП та цифрових компараторів").

Тут по осі абсцис відкладено довжину L префіксів патернів (у символах), що розпізнаватимуться першим каскадом. Літерами GGf та штриховою лінією позначена зростаюча крива витрат на генератор геш-функції, літерами ROM+Cmp та штрих-пунктирною лінією – сумарні витрати на ПЗП та компаратори, літерами RG та тонкою суцільною лінією – витрати на конвеєр вхідної послідовності символів.

Умови проведення розрахунку схеми послідовного каскадування наступні:

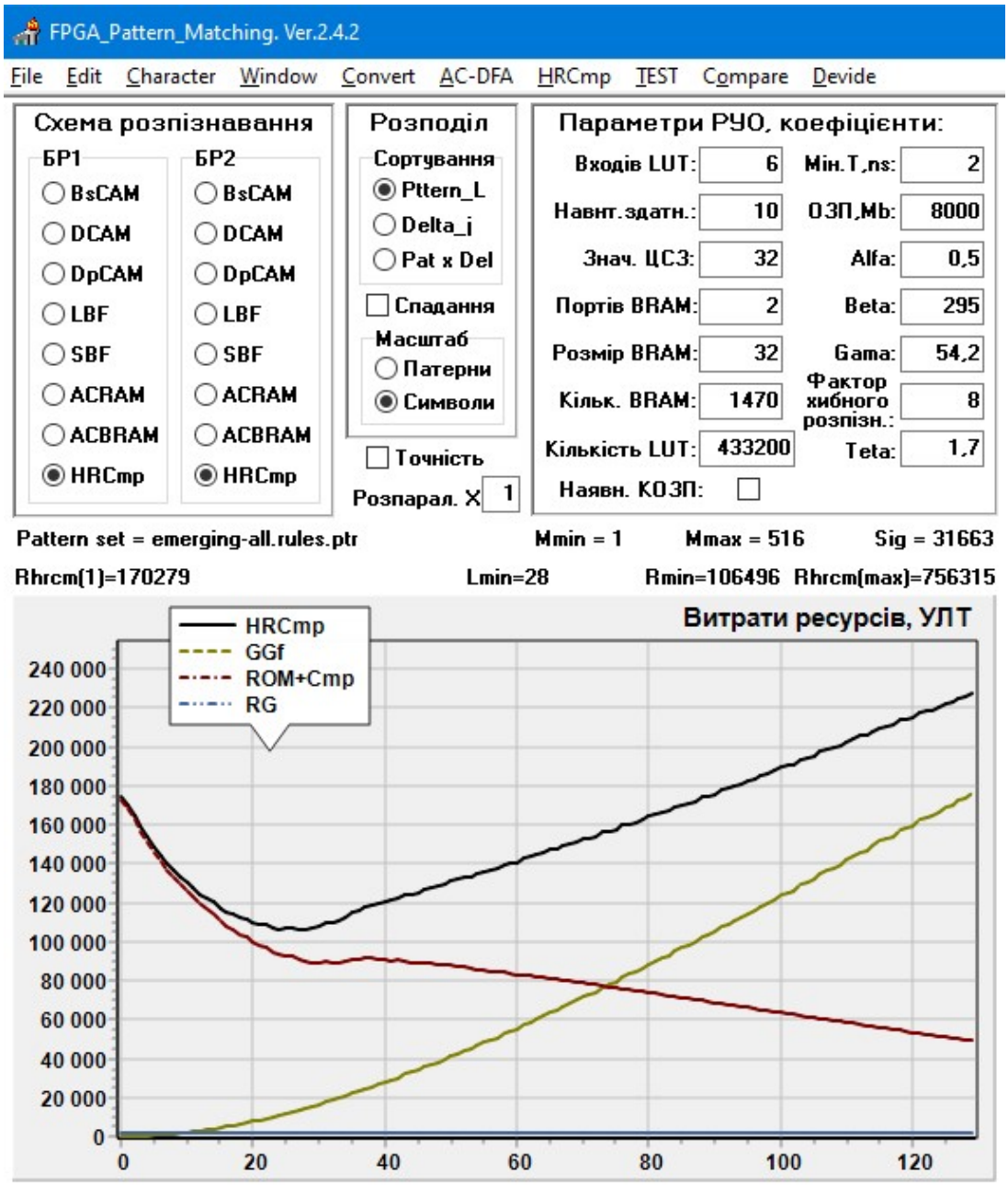


Рис. 6.11. Склад ресурсних витрат схеми HRCmp послідовного каскадування

- $n = 2$;
- реалізація $КС_1$ – генератору геш-функції класу H_3 (див. пункт 3.2.3 "Особливості реалізації геш-функцій на ПЛІС");
- реалізація $КС_2$ – за схемою ПЗП+ЦК (див. підр 4.6.4.1 "Функція оцінки схеми HRCmp. Ресурсна складова");
- набір патернів P використаний з повної бази даних сигнатур "All Rules" MCBB Suricata версії 5.0, що містить $\sigma = 31663$ патернів

довжиною від $m_{\min} = 1$ до $m_{\max} = 516$ загальною кількістю $\Omega = 675883$ символи;

- модель РУО – VC709 від фірми Xilinx на базі ПЛІС Virtex-7 VX690T від Xilinx.

Впорядкування набору патернів в даному випадку не потрібне, тому що природну метрику створює довжина префіксів патернів L (довжина суфіксів патернів при цьому дорівнює $(m_j - L)$, де m_j – довжина j-го патерну).

Як можна бачити, крива витрат на схему HRCmp в цілому має виражений мінімум. Це не є випадковим результатом. Справа в тому, що по мірі зростання довжини префіксів витрати на генератор геш-функції (який ці префікси обробляє) збільшуються з прискоренням. З іншого боку, по мірі скорочення суфіксів патернів витрати на ПЗП та компаратори (які мають справу з суфіксами) зменшуються, причому також нелінійно.

Оптимум досягається при значенні $L = 28$ символів.

Зауважимо, що в наведеному прикладі ефективно задіяні переваги технології гешування завдяки тому, що в першому каскаді розпізнається велика кількість патернів (точніше префіксів) однакової довжини, що, як відомо, є вузьким місцем технічних рішень на базі геш-функцій.

Оскільки при застосуванні методу МПсКс в більшості розробок використовуються не повноцінні схеми розпізнавання, а лише базові технології, коректно оцінити його ефективність складно за відсутністю рівноцінних аналогів. Але порівняння з повноцінними схемами розпізнавання, побудованими за тими ж технологіями, що використані при комбінуванні за методом МПсКс демонструє перевагу в зниженні ресурсних витрат на 40-45% при збереженні швидкодії.

Втім, як свідчать розрахунки, наведені вище значення зниження ресурсних витрат досягаються лише на відносно коротких патернах. На наборах, що містять велику кількість довгих патернів крайня ліва точка на графіках, подібних до наведених на рис. 6.11, піднімається вгору. Внаслідок чого мінімум зсувається вправо, причому відповідне чисельне значення

оптимальних витрат зростає. Тобто, двокаскадні модулі розпізнавання, подібні до схеми HRCom, погано поводяться з довгими патернами.

Тому виникає природний інтерес до об'єднання в одному пристрої методів МПсКс та МПрКм за принципом КМК, розглянутому в попередньому розділі.

6.7.3. Низка експериментів № 3. Принцип КМК

Метою проведення наступної низки експериментів було тестування обчислювальної структури, побудованої з використанням принципу КМК. В складі модуля розпізнавання, що аналізувався, було паралельно скомбіновано схему DpCAM (модифікацію підходу на базі АП і ЦК) та розглянуту вище схему HRCmp, яка в свою чергу є результатом послідовного комбінування. В даних експериментах використовувалися різні за характеристиками РУО та набори патернів, властивості яких дуже розрізняються.

На рис. 6.12-6.14 наведено результати розрахунків за наступних умов.

<u>Експеримент 3.1 (рис. 6.12):</u>

- впорядкування – Pattern_L (за спаданням);
- набір патернів P – з бази даних сигнатур "ActiveX Rules" MCBB Suricata версії 5.0, що містить $\sigma = 2387$ патернів довжиною від $m_{\min} = 3$ до $m_{\max} = 230$ загальною кількістю $\Omega = 212064$ символи;
- модель РУО – проектна плата (Development Board) NetFPGA-1G-CML від фірми Digilent на базі ПЛІС Kintex-7 XC7K325T від Xilinx.

<u>Експеримент 3.2 (рис. 6.13):</u>

- впорядкування – Delta_J (за спаданням);
- набір патернів P – з бази даних сигнатур "Trojan Rules" MCBB Suricata версії 5.0, що містить $\sigma = 7008$ патернів довжиною від $m_{\min} = 1$ до $m_{\max} = 516$ загальною кількістю $\Omega = 145742$ символи;
- модель РУО – VC709 від фірми Xilinx на базі ПЛІС Virtex-7 VX690T від Xilinx.

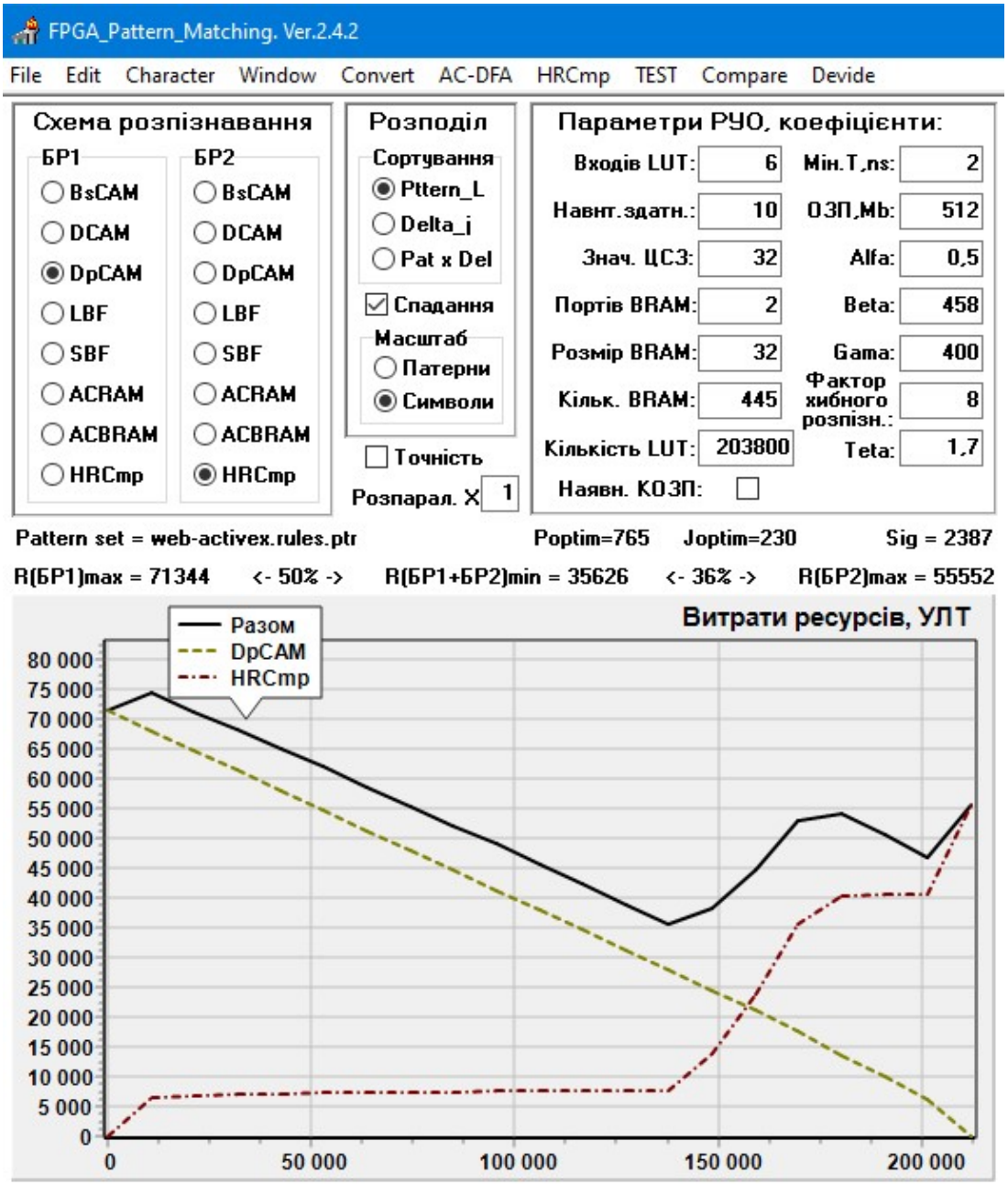


Рис. 6.12. DpCAM + HRCmp, РУО: NetFPGA-1G-CML (Kintex-7), *P*: "ActiveX Rules" (2387 патернів, 212064 символи)

<u>Експеримент 3.3 (рис. 6.14):</u>

- впорядкування – Delta_J (за спаданням);
- набір патернів *P* – з повної бази даних сигнатур "All Rules" MCBB Suricata версії 5.0, що містить $\sigma = 31663$ патернів довжиною від $m_{\min} = 1$ до $m_{\max} = 516$ загальною кількістю $\Omega = 675883$ символи;
- модель РУО – XUP-VV8 від фірми Nallatech на базі ПЛІС Virtex

UltraScale+ VU13P від Xilinx.

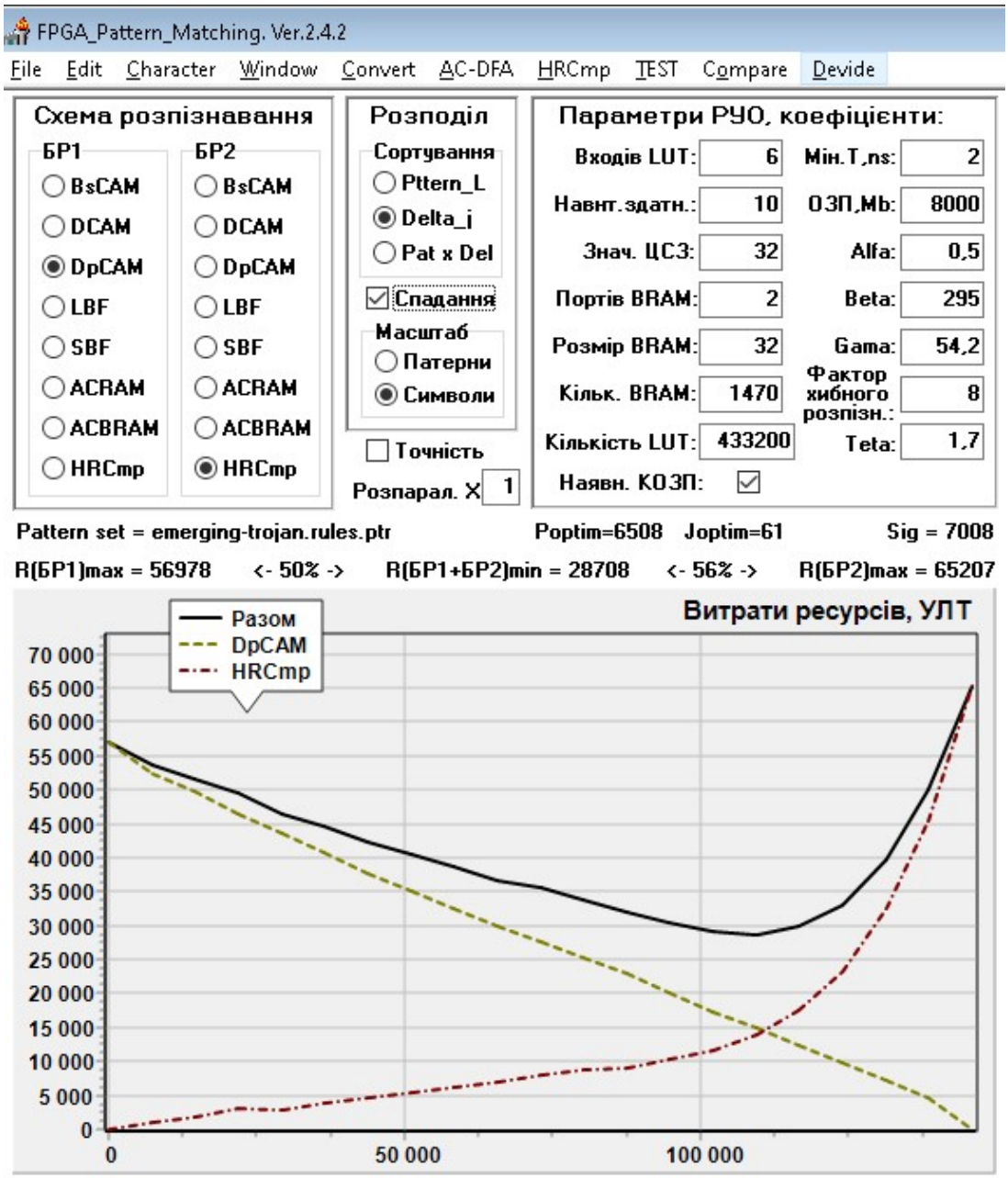


Рис. 6.13. DpCAM + HRCmp, РУО: VC709 (Virtex-7),

P: "Trojan Rules" (7008 патернів, 145742 символи)

Як можна бачити, незважаючи на варіювання умов розрахунків в широких межах, принцип КМК демонструє впевнене підвищення ресурсного ПЕ: порівняно з окремим рішенням DpCAM – на 50-70%, порівняно з двокаскадною схемою HRCmp – на 36-63%.

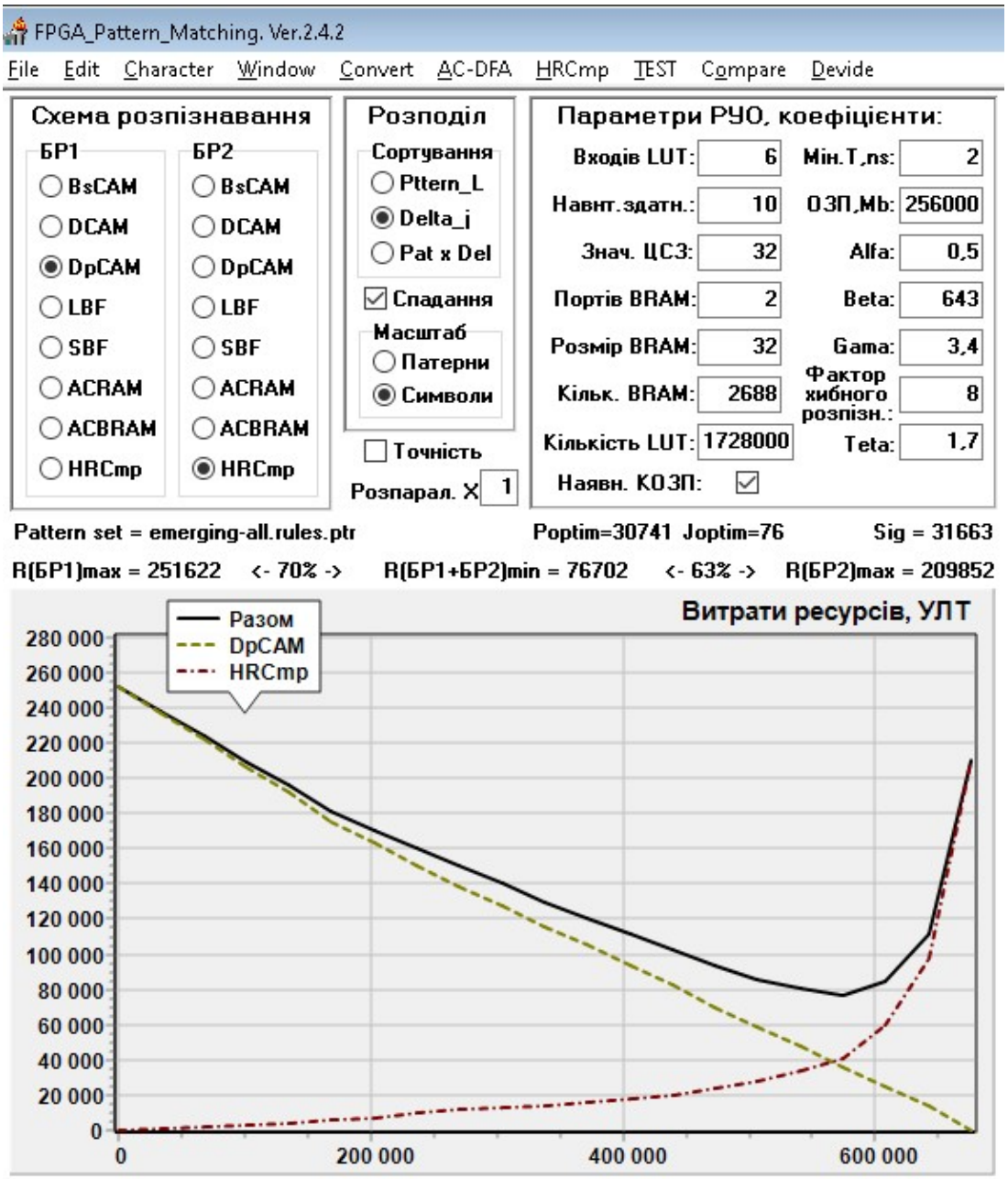


Рис. 6.14. DpCAM + HRCmp, РУО: XUP-VV8 (UltraScale+),
P: "All Rules" (31663 патерни, 675883 символи)

Якщо порівняти мінімальне покращення в низці експериментів з МПрКм (~30%) з мінімальним покращенням в Експерименті 3.1 (36%), а найкращій результат, притаманний МПсКс (45%) з найбільшім покращенням в Експерименті 3.3 (70%), то виявляється, що принцип КМК дозволяє підвищити ПЕ на 20-55% порівняно з використанням окремих методів комбінування.

Ще один висновок, який можна зробити з результатів

Експерименту 3.1, враховуючи нетипову поведінку сумарного графіку загальних апаратних витрат, полягає в наступному. Склад та властивості патернів в наборі суттєво впливають на технічні характеристики, а відтак – на загальні показники ефективності обчислювальних структур, що створюються. Це фактично унеможливлює розробку таких методів підвищення ПЕ РАСЗТЗІ, які були б незалежними від наборів патернів, що розпізнаються, та не враховували їх властивостей під час обчислень. Отже, створені в даному дослідженні методи та засоби, що засновані на ретельному поводженні з патернами, мають важливе теоретичне та практичне значення.

6.7.4. Експеримент № 4. Порівняння з опублікованими даними

Метою наступного експерименту є співставлення результатів експериментів, проведених на базі запропонованих методів, з даними з літературних джерел щодо реальних розробок РАСЗТЗІ.

Колектив авторів публікацій [135], [125] та [164] створив низку експериментальних реконфігуровних пристроїв сигнатурного розпізнавання, використовуючи підходи на базі АП та ЦК, зокрема, DpCAM, з одного боку, та на базі двокаскадної схеми HRCmp – з іншого. Під час порівняння ефективності цих двох розробок вони звернули увагу, що схема DpCAM краще поводиться з довгими патернами, ніж HRCmp, при загальної переваги другої схеми над першою. Це наштовхнуло їх на думку об'єднати в одному пристрою обидві схеми та поділити патерні між ними за довжиною. Вони "застосували схему HRCmp для розпізнавання патернів довжиною до 50 символів та DpCAM для довших патернів … що призвело до збільшення показника ефективності на 10-25%." [164]. Тобто, не володіючи формальним апаратом методів комбінування, автори даної публікації інтуїтивно реалізували прийоми, які призвели до результату, подібному до застосовування принципу КМК до паралельного комбінування схеми DpCAM з послідовним каскадуванням схеми гешування та цифрових компараторів.

На рис. 6.15 наведено результати розрахунку апаратних витрат на схему розпізнавання, побудовану та розраховану за умов, максимально наближених до таких, що використовувалися в згаданих вище публікаціях (див. останній стовпчик з параметрами ПЛІС у табл. 4.2):

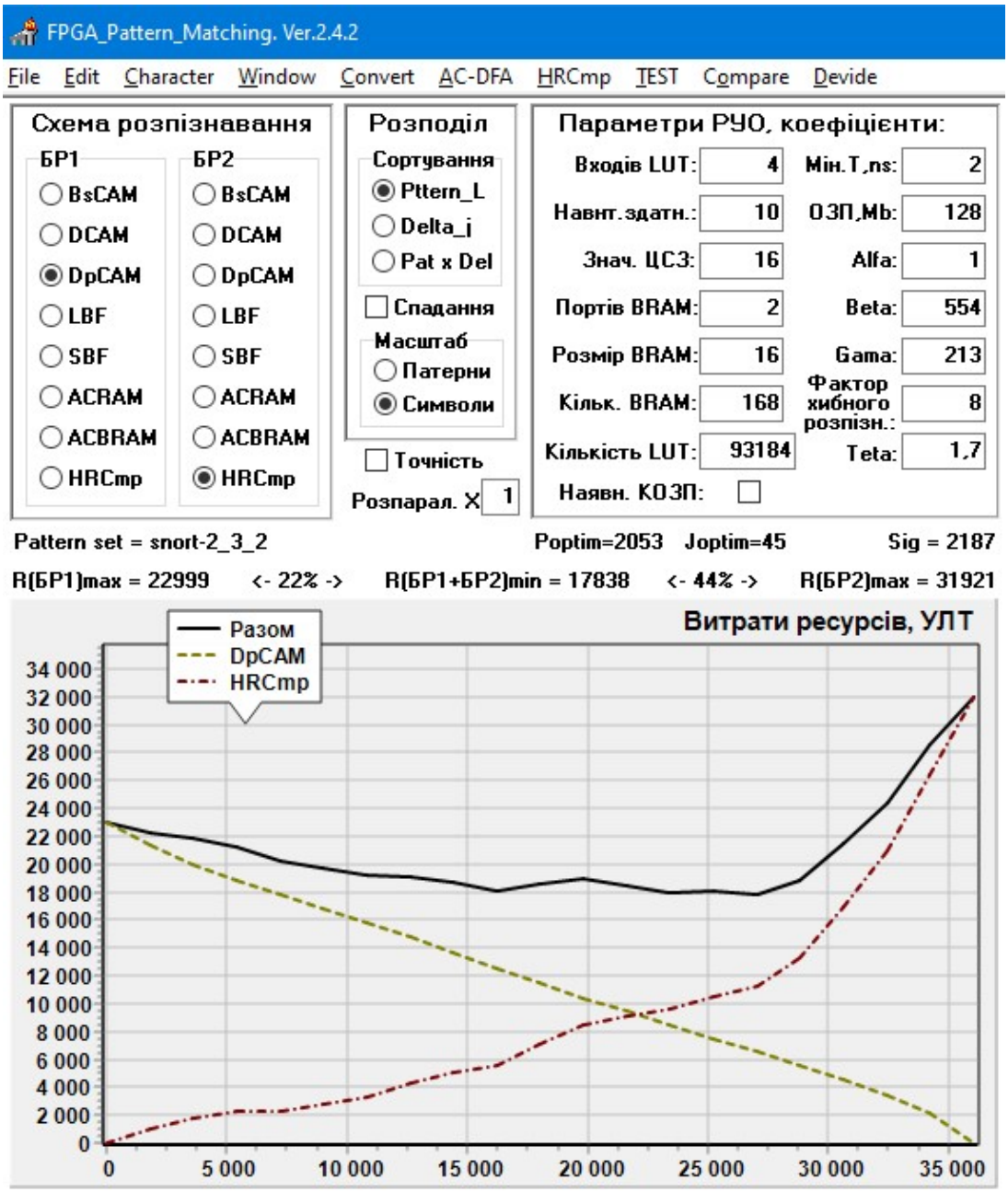


Рис. 6.15. Комбінування методом МПсКс схем розпізнавання DpCAM та HRCmp

– n = 2;

- реалізація $БР_1$ – за схемою DpCAM;
- реалізація $БР_2$ – за схемою HRCmp;
- впорядкування – Pattern_L (за довжиною патернів, за зростанням);
- набір патернів P використаний з бази даних сигнатур MCBB Snort 2007 року версії 2.3.2, що містить $\sigma = 2188$ патернів довжиною від $m_{\min} = 1$ до $m_{\max} = 122$ загальною кількістю $\Omega = 33618$ символів;
- тип РУО не визначено, використовується ПЛІС типу Virtex 2-8000 фірми Xilinx.

Як можна бачити, крива сумарних витрат на апаратуру для розрахованої схеми має мінімум, якому відповідає поділ по довжіні патернів на значенні в 45 символів. При цьому здобуто покращення на 22% та на 44% порівняно з окремими схемами DpCAM та HRCmp відповідно.

Аналізуючи отримані результати можна зробити наступний висновок. За рахунок використання формалізованих методів підвищення ефективності, заснованих на використанні прискореного обчислення для вирішення задачі оптимізації, було з'ясовано, що, якщо поділити патерни на значенні довжини в 45 символів, а не 50, як запропонували автори публікації, то можна отримати результати на 12-19% краще тих, що були здобуті в згаданій роботі (10-25%).

6.8. Висновки до розділу 6

1. Розроблено алгоритми реалізації запропонованих методів комбінування та принципу комбінування методів комбінування.
2. Розроблено програмне забезпечення, що дозволяє проводити експерименти з розробленими в даному дослідженні методами та засобами.
3. В результаті проведення експериментів виявлено два чинники, які є необхідними але не достаніми умовами вдалого комбінування методом МПрКм.
4. В результаті проведення експериментів з'ясовано, що:

- метод паралельного комбінування дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ в окремих випадках на 35-40% порівняно з блоками розпізнавання, побудованими з використанням одного з підходів;
- метод послідовного каскадування дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ в окремих випадках на 40-45% порівняно з блоками розпізнавання, побудованими з використанням одного з підходів;
- метод вертикального об’єднання за рахунок використання багатовимірної таблиці сумісності дозволяє формалізувати наявний досвід розробників РАСЗТЗІ та спростити процедуру пошуку найбільш ефективної комбінації технічних рішень.
- принцип комбінування методів комбінування дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ в окремих випадках на 20-55% порівняно з використанням окремих методів комбінування.
- порівняння з опублікованими практичними результатами показало, що запропоновані методи дозволяють перевершити відомі розробки в окремих випадках на 12-19%.

РОЗДІЛ 7. СИСТЕМА ЦЕНТРАЛІЗОВАНОГО СИНТЕЗУ КОМПОНЕНТІВ РАСЗТЗІ

Процес розробки обчислювальної структури реконфігуровних пристроїв є складною та ресурсномісткою задачею, що вимагає від розробника володіння на високому рівні засобами САПР, знання специфіки проектування цифрових пристроїв, а від комп'ютерної техніки – високої продуктивності. Користувачі РАСЗТЗІ (системні адміністратори, персонал, відповідальний за інформаційну безпеку) не мають умов для самостійного вирішення цієї задачі. Тому актуальними є рішення, за якими процес синтезу реконфігуровних засобів організований таким чином, що складні та ресурсномісткі процедури виконуються не локально на кожної окремої системі, а централізовано, з використанням високопродуктивних комп'ютерних технологій, таких як грід-мережі та хмарні обчислення.

Вперше принципи централізованого створення компонентів РАСЗТЗІ було викладено автором у 2015 році в публікації [223].

7.1. Принцип централізованого створення обчислювальних структур реконфігуровних пристроїв

На рис. 7.1 наведено схематичне зображення сервісу щодо створення обчислювальних структур реконфігуровних засобів, який функціонує за централізованим принципом.

При такому підході єдиний центр обробки запитів від великого числа користувачів виконує наступні функції:

- оперативно поповнює бази сигнатур актуальною інформацією про виявлені факти зловмисної активності;
- збирає дані про поточні параметри безпеки кожного з об'єктів, що захищаються;

– здійснює синтез цифрових схем з урахуванням особливостей кожної клієнтської системи РАСЗТЗІ, генерацію і оперативну доставку користувачам файлів конфігурацій для завантаження в ПЛІС.

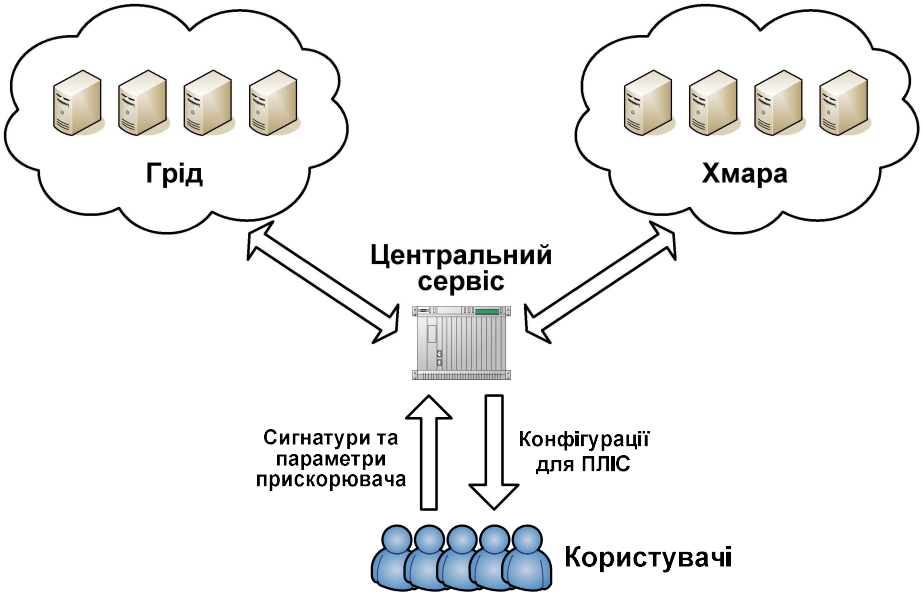


Рис. 7.1. Принцип централізованого синтезу

Зауважимо, що в якості базису для реалізації перерахованих функцій крім грід-системи і хмарного сервісу можуть бути задіяні і інші високопродуктивні платформи.

Використання принципу централізованого створення обчислювальних структур реконфігуровних пристроїв дозволило створити на його основі веб-сервіс, що реалізує запропоновані в даному дослідженні методи та засоби побудови комбінованих обчислювальних структур для підвищення ефективності реконфігуровних сигнатурних засобів захисту інформації в комп'ютерних системах і мережах.

7.2. Функціонування централізованого веб-сервісу

Схема внутрішніх взаємозв'язків веб-сервісу STRAGS (рис. 7.2) в процесі створення реконфігуровних структур МР РАСЗТЗІ та автоматизованого синтезу конфігурацій під час виконання процедури

оперативного оновлення побудована за відомими принципами [224]. Як було розглянуто у розділі 2, під час проведення ПОО технологічний ланцюжок синтезу МР РАСЗТЗІ зводиться до двох етапів: першого, більш наукоємного, на якому формуються машинні описи змінних компонентів, та другого, більш витратного обчислювально – генерації конфігурації для ПЛІС.

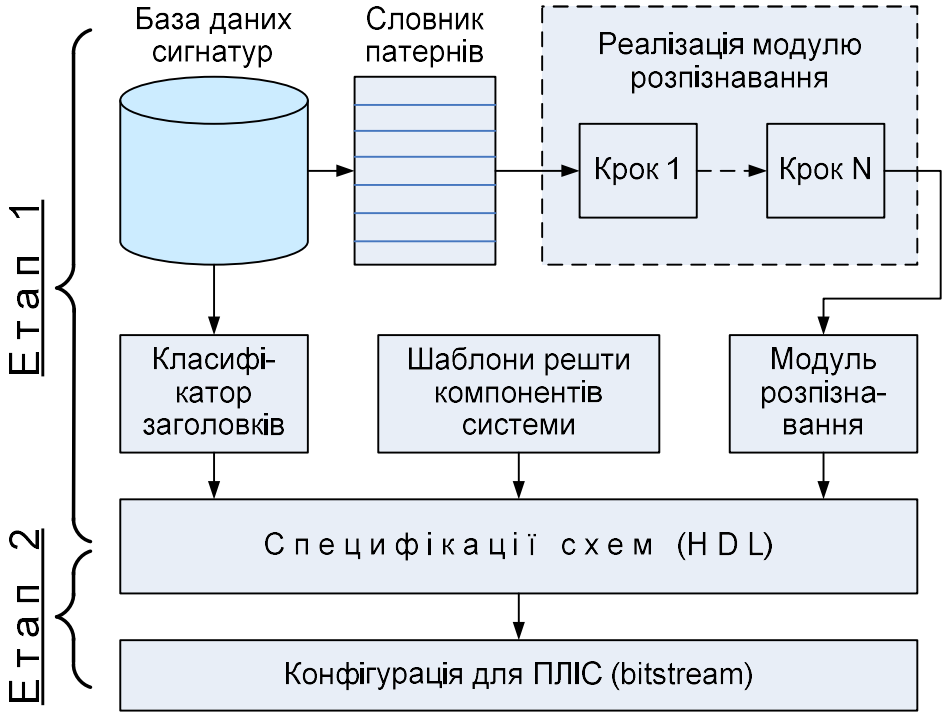


Рис. 7.2. Етапи автоматизованого синтезу реконфігуровної структури МСВВ

7.2.1. Створення машинних описів компонентів РАСЗТЗІ

Перший етап починається з аналізу сигнатур, які повинні розпізнаватися системою захисту, що створюється, та генерації опису схеми класифікатора заголовків (див. підр. 1.1.2.9 "Поняття сигнатури та патерну. Склад сигнатури МСВВ"). Одночасно с тим формується актуальний словник патернів. Потім до згенерованих блоків додаються фіксовані схемні заготовки інших компонентів, що не залежать від набору сигнатур. Для отриманих описів обчислювальних структур складаються специфікації схем, що створюються, на одній з HDL-мов (опису апаратури).

На завершальній стадії HDL-опис доповнюється файлами обмежень (ucf-файлами) та іншими компонентами проекту створення цифрового пристрою, після чого запускається процедура автоматичної генерації конфігурацій для ПЛІС, в даному сервісі – із застосуванням фірмового пакета САПР виробника мікросхем програмованої логіки, що використовуються.

Процедура реалізації модулю розпізнавання на даному етапі виконується із застосуванням методів та засобів, створених в результаті виконання даного дослідження (див. рис. 6.1 з підр. 6.5 "Загальна структура використання методів комбінування").

7.2.2. Генерація конфігурацій, що завантажуються в ПЛІС

Після того, як обчислювальну структуру цифрової схеми створено та подано на мові опису апаратури, ініціюється процедура генерації файлів конфігурацій. Даний етап синтезу цифрової схеми крім власне процедури створення послідовностей бітів, що завантажуються в ПЛІС (Bitstream Generating), включає передуючу йому операцію компіляції проекту, яка в свою чергу складається з ряду обчислювально ємних процедур, таких як:

- синтез (Synthesize);
- трансляція (Translate);
- відображення (Map);
- розміщення та трасування (Place & Route).

Кожний з перерахованих кроків в свою чергу може включати різну кількість підкроків. Рис. 7.3 ілюструє ієрархію таких підзадач на прикладі процесу розробки цифрової схеми, що відображується графічним інтерфейсом САПР WebPack ISE фірми Xilinx. (Тут процедури Translate, Map та Place & Route структурно згруповані в єдину операцію реалізації проекту Implement Design).

Всі стадії процесу, починаючи з рядка "Synthesize – XST" по "Programming File Generation Report" виконуються в автоматичному режимі без участі розробника (в разі відсутності синтаксичних і логічних помилок в

проекті).

В залежності від складності схеми, що синтезується, і типу ПЛІС процес генерації файлів конфігурацій може сягати від одиниць хвилин до кількох годин.

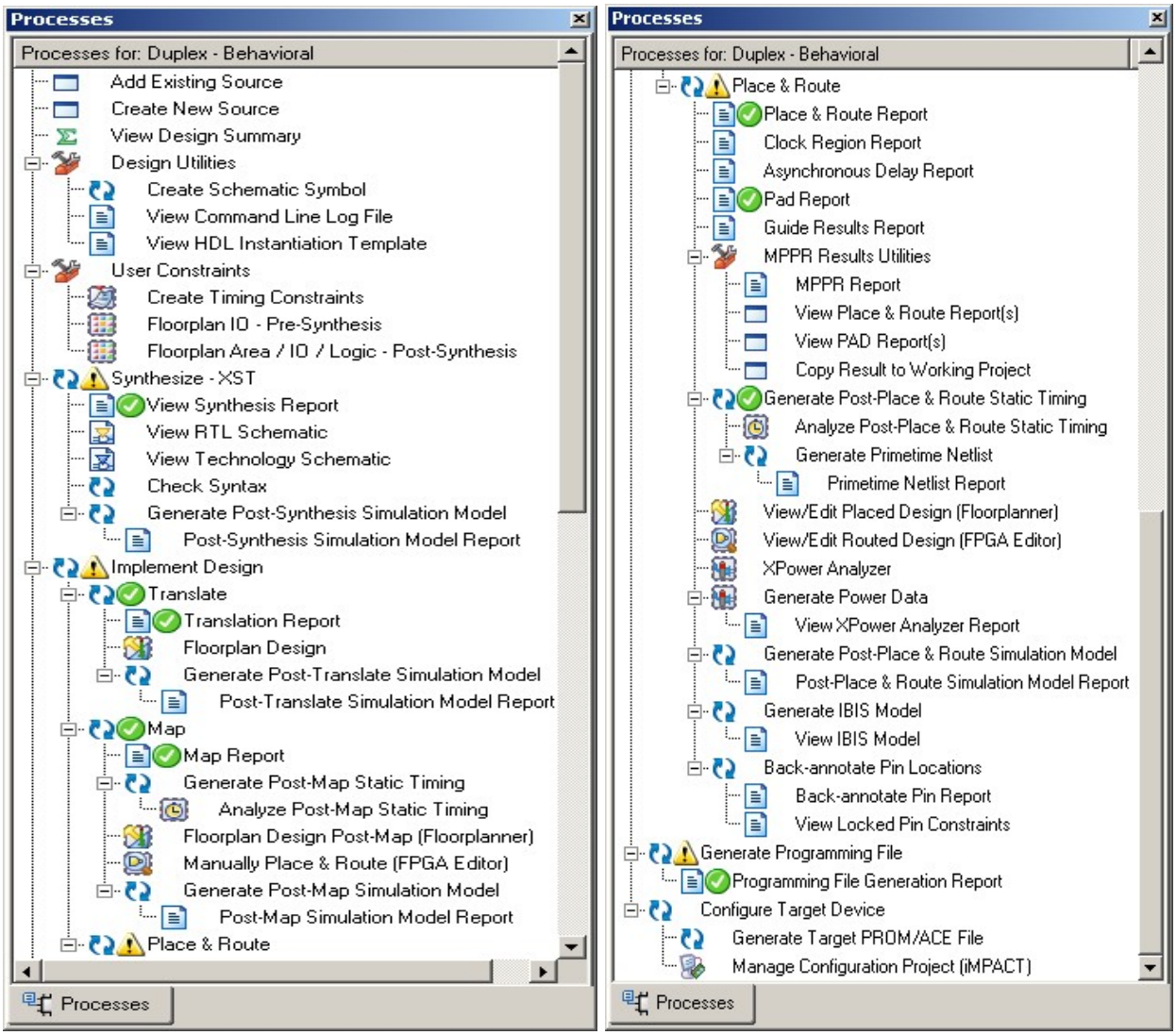


Рис. 7.3 Процес генерації конфігурації, що завантажується в ПЛІС

Особливо велику трудомісткість компіляція набуває, коли задіяні ресурси майже повністю займають площу кристала програмованої НВІС. В цьому випадку процедура розміщення і трасування перетворюється складне завдання перебору комбінаторно великого числа варіантів.

Додаткові деталі щодо функціонування реалізації процедури ПОО в складі сервісу можна знайти в публікації [225].

7.3. Особливості реалізації веб-сервісу

Сервіс було створено у вигляді макетного зразка на базі обчислювального кластера та грід-вузла Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України [226]. Розробка отримала назву STRAGS (Security Tasks Reconfigurable Accelerators Grid-Service – грід-сервіс реконфігуровних прискорювачів завдань інформаційної безпеки) Зовнішній вигляд стартової сторінки сервісу наведено на рис. 7.4.

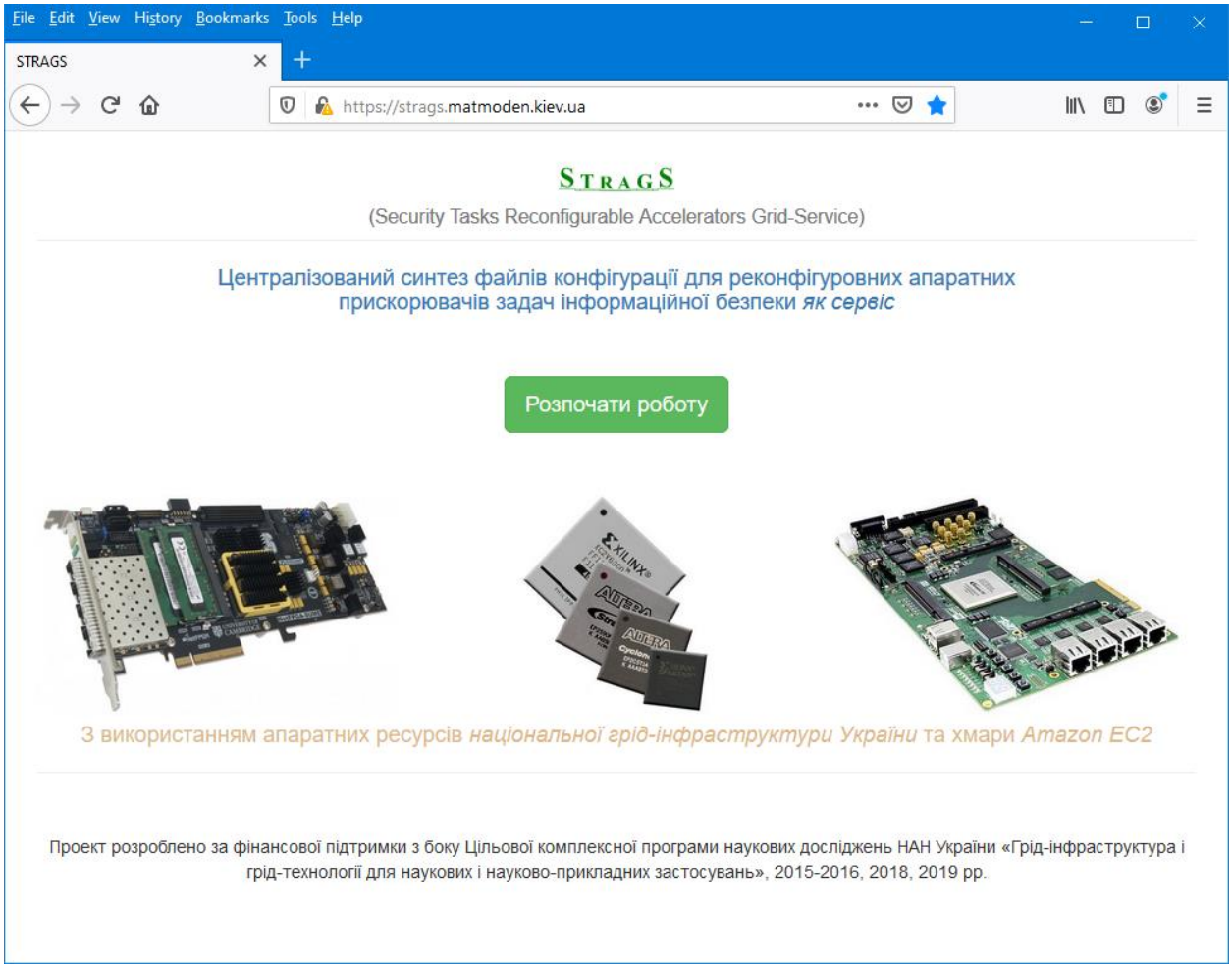


Рис. 7.4. Веб-сервіс STRAGS

В якості функціонального базису грід-складової сервісу використовується вітчизняна грід-технологія Rainbow ("ARC in the Cloud") [227], яка спочатку створювалася для запуску в середовищі Українського національного гріду (УНГ) спеціалізованого ПЗ moldyngrid для віртуальної

організації medgrid. Розробка виявилася вдалою і знайшла більш широке застосування. Технологія дозволяє запускати на грід-вузлах віртуальні машини з необхідним програмним забезпеченням і забезпечує інтерактивний доступ до них.

В процесі функціонування веб-сервіс STRAGS ініціює роботу на віддалених вузлах грід-середовища УНГ декількох агентів – віртуальних машин з попередньо встановленим і налаштованим програмним забезпеченням, необхідним для синтезу реконфігуровних пристроїв і генерації конфігурацій. Отримуючи запити від клієнтів, сервіс формує завдання та розподіляє їх між активними агентами, підтримуючи їх число достатнім для забезпечення готовності на необхідному рівні. Отримавши завдання у вигляді грід-завдання, агент ініціює процеси автоматичного створення необхідної цифрової схеми та синтезу відповідної конфігурації для ПЛІС, після чого повертає результат роботи сервісу.

В якості експериментальної платформи для хмарної складової системи STRAGS було використано сервіс Elastic Compute Cloud (EC2) інфраструктури хмарних веб-сервісів Amazon Web Services (AWS) в безкоштовному демонстраційному режимі. Віртуальні машини – агенти запускаються на рівні IaaS (Infrastructure as a Service) цього хмарного сервісу.

Інтерфейс сервісу дозволяє користувачеві завантажити завдання зі своїми даними (переліком сигнатур, типом використаного РУО та додатковими вимогами щодо процесу розпізнавання). По завершенню виконання завдання клієнт сервісу отримує результати обчислень у вигляді файлів конфігурації для ПЛІС та лог-файли с даними щодо процесу виконання процедури синтезу, або, в разі невдачі, докладну діагностичну інформацію.

Завдяки використанню технології Rainbow користувачі мають можливість спостерігати за процесом виконання робіт центральним сервісом. На рис. 7.5 наведено одне з вікон грід-сервісу, яке відображає прогрес виконання запущеного на синтез завдання. Наведено момент виконання

кроку трасування процедури PAR (Place and Route).

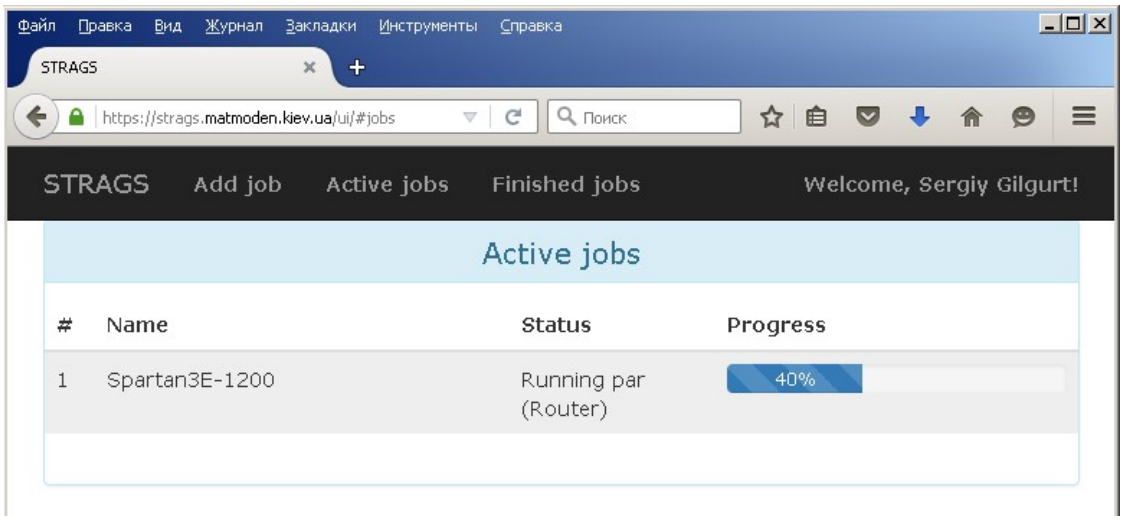


Рис. 7.5. Екран активних завдань веб-сервісу STRAGS

Інтерфейс адміністратора сервісу дозволяє контролювати роботу системи, збільшувати кількість агентів, що чергують, вручну запускати та термінувати завдання, отримувати службову інформацію щодо завдань, що виконуються та робочих параметрів сервісу.

7.4. Висновки до розділу 7

5. Розглянуто принцип централізованого створення обчислювальних структур реконфігуровних пристроїв.
6. Розглянуто веб-сервіс, що реалізує даний принцип. Процедура синтезу МР РАСЗТЗІ складається до двох етапів, на першому формуються машинні описи змінних компонентів, на другому автоматично створюються конфігурації для ПЛІС.
7. Експерименти, проведені з розробленим веб-сервісом, підтвердили перспективність ідеї перенесення ресурсномістких операцій з локальних систем інформаційного захисту до високопродуктивного обчислювального середовища за принципом централізованого виконання.

ВИСНОВКИ

В дисертаційній роботі, на основі проведених досліджень, вирішена важлива науково-технічна проблема, що полягає в розробці та розвитку методів побудови комбінованих обчислювальних структур для підвищення ефективності реконфігуровних сигнатурних технічних засобів захисту інформації.

Під час дослідження були отримані наступні наукові та практичні результати:

1. Проведено аналіз проблем технічного захисту інформації, які вирішуються сигнатурними засобами, що дозволило сформулювати задачу множинного розпізнавання рядків та ввести поняття процедури оперативного оновлення.
2. Проведено дослідження технічних можливості сучасних ПЛІС та пристроїв на їх основі в якості апаратної бази для вирішення задач технічного захисту інформації, що дозволило порівняти її з іншими апаратними платформами.
3. Проведено аналіз принципів функціонування та побудови РАСЗТЗІ як технічних систем. Сформульовано та класифіковано показники їх ефективності, що дало можливість оцінювати та порівнювати ефективність як окремих компонентів створюваних засобів захисту інформації, так и системи в цілому.
4. В результаті вивчення та дослідження наявного світового досвіду побудови реконфігуровних сигнатурних засобів захисту, виокремлено три найбільш ефективні підходи до побудови модуля розпізнавання РАСЗТЗІ; шляхом дослідження цих підходів та їх модифікацій на структурному рівні формалізовано властивості кожного з них, враховуючи специфіку реалізації реконфігуровними засобами, що дозволило сформулювати у вигляді наукових методів ідею комбінування (сумісного використання) різних підходів до побудови

компонентів РАСЗТЗІ з метою використання переваг кожного з підходів.

5. Розроблено метод прискореного обчислення технічних характеристик на основі функцій оцінки, який на відміну від відомих методів проектування РАСЗТЗІ не потребує виконання витратних за часом процедур синтезу цифрових схем, що дозволило здійснювати швидку оцінку за заданими показниками ефективності окремих компонентів і РАСЗТЗІ в цілому.
6. Розроблено метод прискорення процедури оптимізації паралельного комбінування, суть якого полягає у впорядкуванні патернів за певним параметром, що дозволило за рахунок послідовної зміни параметру впорядкування пришвидшити поділ патернів між блоками, що комбінуються, від повного перебору комбінаторно великої кількості варіантів до витрат часу за лінійним законом. Метод не гарантує знаходження глобального оптимуму, проте зменшує обчислювальну складність процедури оптимізації до прийнятних для практичного використання значень від десятків секунд до десятків хвилин.
7. На основі розроблених методів прискорення обчислень, виявлено функціональні залежності кількісних характеристик найбільш ефективних схем розпізнавання від параметрів реконфігуровного обчислювача та властивостей заданого набору патернів, що дозволило здійснити експериментальну перевірку методів комбінування.
8. Сформульовано та вдосконалено метод паралельного комбінування, який за рахунок паралельного з'єднання різних за принципами побудови блоків розпізнавання та оптимізації поділу між ними набору патернів, використовуючи розроблені методи прискорення, дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ в окремих випадках на 35-40% порівняно з блоками розпізнавання, побудованими з використанням одного з підходів.

9. Сформульовано та вдосконалено метод послідовного каскадування, який за рахунок послідовно з'єднання різних за принципами побудови блоків розпізнавання та оптимізації поділу між ними питомих патернів по довжині, використовуючи розроблені методи прискорення, дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ в окремих випадках на 40-45% порівняно з блоками розпізнавання, побудованими з використанням одного з підходів.
10. Отримав подальший розвиток метод вертикального об’єднання, який за рахунок тісного сполучання в одному блоці кількох підходів або технічних рішень дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ, а саме, запропоновано модифікацію метода, яка за рахунок використання багатовимірної таблиці сумісності дозволяє формалізувати наявний досвід розробників РАСЗТЗІ та спростити процедуру пошуку найбільш ефективної за наданих умов комбінації задіяних даним методом підходів або технічних рішень.
11. Сформульовано та вдосконалено принцип комбінування методів комбінування, який за рахунок ієрархічного використання сформульованих методів комбінування дозволяє підвищити показники ефективності модулю розпізнавання РАСЗТЗІ в окремих випадках на 20-45% порівняно з використанням окремих методів комбінування.
12. Розроблено алгоритми реалізації вдосконалених та розвинутих методів комбінування, що дозволило створити програмні засоби їх перевірки та аналізу.
13. Розроблено програмні засоби для перевірки та аналізу вдосконалених та розвинутих методів, що дозволило шляхом проведення обчислювальних експериментів порівняти їх з відомими розробками, в яких застосовуються методи комбінування, та з'ясувати, що запропоновані та вдосконалені методи за рахунок прискорення обчислень та використання процедур оптимізації дозволяють перевершити відомі розробки в окремих випадках на 12-19%.

14. Досліджено та розвито принцип централізованого створення обчислювальних структур реконфігуровних пристроїв, що дозволило розробити на його основі веб-сервіс, що реалізує запропоновані методи та засоби, використовуючи розподілені та хмарні середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] С. Я. Гильгурт, "Анализ применения аппаратного ускорения информационной защиты в автоматизированных системах энергетической отрасли," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 83, pp. 154-164, 2018.

[2] Y. Yang, H. Xu, L. Gao, Y. Yuan, K. McLaughlin, and S. Sezer, "Multidimensional Intrusion Detection System for IEC 61850-Based SCADA Networks," *IEEE Transactions on Power Delivery*, vol. 32, no. 2, pp. 1068-1078, 2017, doi: 10.1109/TPWRD.2016.2603339.

[3] В. А. Лужецький, А. Д. Кожухівський, and О. П. Войнович, *Основи інформаційної безпеки. Навчальний посібник*. Вінниця: ВНТУ, 2009.

[4] E. G. Amoroso, Fundamentals of Computer Security Technology. NJ: Prentice Hall PTR, 1994, p. 404.

[5] К. Касперски, Техника сетевых атак. М.: Солон-Р, 2001.

[6] Г. Ф. Конахович, В. П. Климчук, С. М. Паук, and В. Г. Потапов, *Защита информации в телекоммуникационных системах*. К.: «МК-Пресс», 2005.

[7] А. Г. Микитишин, М. М. Митник, and П. Д. Стухляк, *Комплексна безпека інформаційних мережевих систем: навчальний посібник*. Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016, p. 256.

[8] Ю. М. Коростиль, А. Н. Давиденко, С. Я. Гильгурт, and М. М. Панченко, "Анализ угроз и опасностей в компьютерных системах на предмет защиты цифровыми реконфигурируемыми устройствами," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 56, pp. 10-17, 2010.

[9] С. В. Сальник, А. С. Сторчак, and А. Є. Крамський, "Аналіз вразливостей та атак на державні інформаційні ресурси, що обробляються в інформаційно-телекомунікаційних системах," *Системи обробки інформації,* no. 2(157), pp. 121-128, 2019.

[10] S. Anwar *et al.*, "From Intrusion Detection to an Intrusion Response System: Fundamentals, Requirements, and Future Directions," *Algorithms,* Review vol. 10, no. 2, p. 24, Jun 2017, Art no. 39, doi: 10.3390/a10020039.

[11] С. Я. Гильгурт and А. К. Гиранова, "О применении реконфигурируемых вычислителей для решения задач защиты информации," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 46, pp. 93-99, 2008.

[12] С. Я. Гильгурт and А. К. Гиранова, "О применении реконфигурируемых вычислителей для решения вопросов защиты информации, пересылаемой в грид-среде," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 51, pp. 65-72, 2009.

[13] С. Я. Гильгурт and А. К. Гиранова, "Программно-аппаратная защита данных в распределенных интеллектуальных системах," *Штучний інтелект,* no. 3, pp. 706-711, 2010.

[14] А. В. Лукацкий, *Обнаружение атак*. СПб.: БХВ-Петербург, 2001, p. 624.

[15] С. Я. Гильгурт, "Аппаратное распознавание строк в интеллектуальных системах защиты информации," in *Тез. докл. международной научно-технической конференции «Искусственный интеллект. Интеллектуальные системы ИИ-2011»*, 2011, vol. 1: Донецк: ИПИИ «Наука і освіта», pp. 271-274.

[16] В. Евдокимов, А. Давиденко, and С. Гильгурт, "Централизованный синтез реконфигурируемых аппаратных средств информационной безопасности на высокопроизводительных платформах," *Захист інформації,* vol. 20, no. 4, pp. 247-258, 2018, doi: 10.18372/2410-7840.20.13426.

[17] D. Stiawan, A. Y. I. Shakhatreh, M. Y. Idris, A. B. Kamarulnizam, and H. A. Abdul, "Intrusion prevention system: A survey," *Journal of Theoretical and Applied Information Technology,* vol. 40, no. 1, pp. 44-54, 2012.

[18] S. Kumar. "Survey of Current Network Intrusion Detection Techniques." https://www.cse.wustl.edu/~jain/cse571-07/ids.htm (accessed 15.09, 2019).

[19] U. A. Sandhu, S. Haider, S. Naseer, O. U. Ateeb, and S. Z. A. Bhutto, "A Survey of Intrusion Detection & Prevention Techniques," in *2011 International Conference on Information Communication and Management*, Singapore, 2011, vol. 16: IACSIT Press, pp. 66-71.

[20] M. Bijone, "A Survey on Secure Network: Intrusion Detection & Prevention Approaches," *American Journal of Information Systems,* vol. 4, no. 3, pp. 69-88, 2016.

[21] С. В. Казмірчук, А. О. Корченко, and Т. І. Паращук, "Аналіз систем виявлення вторгнень," *Захист інформації,* vol. 20, no. 4, pp. 259-276, 2018.

[22] А. Корченко, О. Заріцький, Т. Паращук, and В. Бичков, "Програмне забезпечення формування еталонів параметрів," *Захист інформації,* vol. 20, no. 3, pp. 133-148, 2018.

[23] Ю. М. Коростиль and С. Я. Гильгурт, "Принципы построения сетевых систем обнаружения вторжений на базе ПЛИС," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 57, pp. 87-94, 2010.

[24] С. Я. Гильгурт, "Применение реконфигурируемых вычислителей для аппаратного ускорения сигнатурных систем защиты информации," in *Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2018»*, Київ, 2018: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, pp. 107-110.

[25] Y. H. Cho and W. H. Mangione-Smith, "Deep packet filter with dedicated logic and read only memories," *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Proceedings,* Proceedings Paper pp. 125-134, 2004, doi: 10.1109/fccm.2004.25.

[26] P. C. Lin, Y. D. Lin, T. H. Lee, and Y. C. Lai, "Using string matching for deep packet inspection," *Computer,* Article vol. 41, no. 4, pp. 23-28, Apr 2008, doi: 10.1109/mc.2008.138.

[27] S. Y. Hilhurt, "Application of FPGA-based reconfigurable accelerators for network security tasks," *Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine,* no. 73, pp. 17-26, 2014.

[28] С. Я. Гильгурт, "Сравнительный анализ межсетевых экранов, систем обнаружения и предотвращения вторжений, контроля целостности," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 60, pp. 116-120, 2011.

[29] О. С. Потенко, "Аналіз систем захисту веб-додатків від хакерских атак," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 89, pp. 50-63, 2019.

[30] "Zeek. An Open Source Network Security Monitoring Tool." Version 3.0.0. https://zeek.org (accessed 15.10, 2019).

[31] "Welcome to Hogwash." Version 0.5. http://hogwash.sourceforge.net/oldindex.html (accessed 15.09, 2018).

[32] "SNORT." Version 2.9.3.1. http://www.snort.org (accessed 15.09, 2018).

[33] "Suricata. Open Source IDS." Version 4.1.0. http://suricata-ids.org (accessed 15.09, 2018).

[34] "SNORT Users Manual. The Snort Project." Cisco. Version 2.9.16. http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node2.html (accessed 17.08, 2019).

[35] А. Н. Давиденко, С. Я. Гильгурт, and В. И. Сабат, "Аппаратное ускорение алгоритмов сигнатурного обнаружения вторжений в открытой системе информационной безопасности Snort," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. ГЄ Пухова НАН України,* pp. 94-103, 2012.

[36] Ю. М. Коростиль, С. Я. Гильгурт, and О. М. Назаренко, "Анализ базы данных системы информационной безопасности Snort и вопросы быстродействия," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. ГЄ Пухова НАН України,* pp. 77-84, 2012.

[37] А. Красоткин, "Красоткин А. Обнаружение сетевых так – Snort," vol. Мир ПК, no. 6, 2003.

[38] Н. Н. Безруков, Компьютерная вирусология: Справ. руководство. Киев: Укр. Сов. энцикл., 1991, p. 414.

[39] F. Cohen, "On the implications of computer viruses and methods of defense," *C&S,* vol. 7, no. 2, pp. 167-184, 1988.

[40] M. Al-Asli and T. A. Ghaleb, "Review of signature-based techniques in antivirus products," in *2019 International Conference on Computer and Information Sciences, ICCIS 2019*, 2019: Institute of Electrical and Electronics Engineers Inc., doi: 10.1109/ICCISci.2019.8716381.

[41] D. Pao, X. Wang, X. R. Wang, C. Cao, and Y. S. Zhu, "String Searching Engine for Virus Scanning," *IEEE Transactions on Computers,* Article vol. 60, no. 11, pp. 1596-1609, Nov 2011, doi: 10.1109/tc.2010.250.

[42] J. T. L. Ho and G. G. F. Lemieux, "PERG: A Scalable FPGA-based Pattern-matching Engine with Consolidated Bloomier Filters," *Proceedings of the 2008 International Conference on Field-Programmable Technology,* Proceedings Paper pp. 73-80, 2008, doi: 10.1109/fpt.2008.4762368.

[43] "ClamAV is an open source antivirus engine for detecting trojans, viruses, malware & other malicious threats." Version 0.101.2. http://www.clamav.net (accessed 09.04, 2019).

[44] U. Savagaonkar, R. Sahita, G. Nagabhushan, P. Rajagopal, and D. Durham. (2005) An OS Independent Heuristics-Based Worm-Containment System. *Technology@Intel Magazine*. 1-8.

[45] B. Madhusudan and J. W. Lockwood, "A hardware-accelerated system for real-time worm detection," *IEEE Micro,* Article; Proceedings Paper vol. 25, no. 1, pp. 60-69, Jan-Feb 2005, doi: 10.1109/mm.2005.1.

[46] M. Faezipour, M. Nourani, and R. Panigrahy, "A Hardware Platform for Efficient Worm Outbreak Detection," *ACM Transactions on Design Automation of Electronic Systems,* Article vol. 14, no. 4, p. 29, Aug 2009, Art no. 49, doi: 10.1145/1562514.1562517.

[47] С. Я. Гильгурт, "Особенности применения реконфигурируемых вычислителей для аппаратной защиты информационных систем," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 38, pp. 36-41, 2007.

[48] S. Singh, C. Estan, G. Varghese, S. Savage, and Usenix, "Automated worm fingerprinting," *Usenix Association Proceedings of the Sixth Symposium on Operating Systems Design and Implementation (Osde '04),* Proceedings Paper pp. 45-60, 2004.

[49] И. Ашманов, "Борьба со спамом техническими средствами," *BYTE/Россия,* vol. 65, no. 1, pp. 70-74, 2004.

[50] M. Takesue, "Personalized Filtering of Polymorphic E-mail Spam," *2009 Third International Conference on Emerging Security Information, Systems, and Technologies,* Proceedings Paper pp. 249-254, 2009, doi: 10.1109/securware.2009.45.

[51] J. Yan and P. L. Cho, "Enhancing collaborative spam detection with bloom filters," in *22nd Annual Computer Security Applications Conference*, Miami Beach, FL, Dec 11-15 2006, LOS ALAMITOS: Ieee Computer Soc, in Annual Computer Security Applications Conference – Proceedings, 2006, pp. 414-+, doi: 10.1109/acsac.2006.26.

[52] D. Gusfield, lgorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge: Cambridge University Press, 1997.

[53] Д. Гасфилд, Гасфилд Д. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. СПб.: Невский Диалект; БХВ-Петербург, 2003, p. 654.

[54] T. AbuHmed, A. Mohaisen, and D. Nyang, "Deep Packet Inspection for Intrusion Detection Systems: A Survey," *Computer Networks,* vol. 57, no. 18, pp. 4047-4064, 2012.

[55] W. Jiang, Y. H. E. Yang, and V. K. Prasanna, "Scalable multi-pipeline architecture for high performance multi-pattern string matching," in *24th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2010*, Atlanta, GA, 2010, doi: 10.1109/IPDPS.2010.5470374.

[56] J. Lunteren, "High-performance pattern-matching for intrusion detection," *25th IEEE International Conference on Computer Communications, Vols 1-7, Proceedings IEEE Infocom 2006,* Proceedings Paper pp. 1409-1421, 2006.

[57] D. E. KNUTH, J. H. MORRIS, and P. V.R., "Fast pattern matching in strings," *SIAM J. COMPUT.,* vol. 6, no. 2, 1977.

[58] R. S. Boyer and J. S. Moore, "A Fast String Searching Algorithm," *Communications of the ACM,* vol. 20, no. 10, pp. 762-772, 1977, doi: 10.1145/359842.359859.

[59] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *Ibm Journal of Research and Development,* Article vol. 31, no. 2, pp. 249-260, Mar 1987, doi: 10.1147/rd.312.0249.

[60] B. Smyth, *Computing Patterns in Strings*. Essex: Pearson Addison Wesley, 2003.

[61] Б. Смит, Методы и алгоритмы вычислений на строках. Теоретические основы регулярных вычисленийСмит. М.: Вильямс (in Пер. с англ.), 2006.

[62] С. Я. Гильгурт, "Задача множественного распознавания строк в интенсивном потоке данных и методы ее аппаратного ускорения," in *Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2016»*, 2016: К.: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, pp. 166-169.

[63] A. V. Aho and M. J. Corasick, "Efficient String Matching: An Aid to Bibliographic Search," *Communications of the ACM,* vol. 18, no. 6, pp. 333-340, 1975, doi: 10.1145/360825.360855.

[64] B. Commentz-Walter, "A string matching algorithm fast on the average," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol. 71 LNCS, ed, 1979, pp. 118-132.

[65] S. Wu and U. Manber, "A fast algorithm for multi-pattern searching," University of Arizona, 1994.

[66] Б. Домёлки, "Алгоритмы для распознавания свойств последовательностей символов," *Журнал вычислительной математики и математической физики,* vol. 5, no. 1, pp. 77-97, 1965.

[67] B. Dömölki, "A universal compiler system based on production rules," *BIT Numerical Mathematics,* vol. 8, no. 4, pp. 262-275, 1968, doi: 10.1007/BF01933436.

[68] M. Fisk and G. Varghese, "Applying fast string matching to intrusion detection," University of California, San Diego, 2002.

[69] P. C. Lin, Z. X. Li, Y. D. Lin, Y. C. Lai, and F. C. Lin, "Profiling and accelerating string matching algorithms in three network content security applications," *IEEE Communications Surveys and Tutorials,* Article vol. 8, no. 2, pp. 24-36, 2006, doi: 10.1109/comst.2006.315851.

[70] Z. Khan, R. K. Pateriya, and M. Azad, "Multiple Pattern String Matching Methodologies: A Comparative Analysis," *International Journal of Scientific and Research Publications,* vol. 2, no. 7, pp. 1-7, 2012.

[71] S. Yusuf, W. Luk, M. Sloman, N. Dulay, E. C. Lupu, and G. Brown, "Reconfigurable architecture for network flow analysis," *IEEE Transactions on Very Large Scale Integration (Vlsi) Systems,* Article; Proceedings Paper vol. 16, no. 1, pp. 57-65, Jan 2008, doi: 10.1109/tvlsi.2007.912115.

[72] Г. В. Ройзензон, "Выбор вычислительных кластеров на основе анализа количественной и качественной информации," *Искусственный интеллект,* no. 2, pp. 73-79, 2004.

[73] С. Я. Гильгурт, "Анализ применения унифицированных вычислителей в интеллектуальных системах," *Штучний інтелект,* no. 1, pp. 144-148, 2009.

[74] С. Гильгурт, "Реконфигурируемые вычислители. Аналитический обзор," *Электронное моделирование,* 2013.

[75] H. Bos and K. Huang, "A network instruction detection system on IXP1200 network processors with support for large rule sets," 2004.

[76] M. Aldwairi, T. Conte, and P. Franzon, "Configurable string matching hardware for speeding up intrusion detection," *ACM SIGARCH Computer Architecture News,* vol. 33, no. 1, pp. 99-107, 2005.

[77] Markatos, "An active splitter architecture for intrusion detection and prevention," *IEEE Transactions on Dependable and Secure Computing,* Article vol. 3, no. 1, pp. 31-44, Jan-Mar 2006, doi: 10.1109/tdsc.2006.6.

[78] P. Piyachon and Y. Luo, "Efficient memory utilization on network processors for deep packet inspection," in *2nd ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2006*, San Jose, CA, 2006, pp. 71-80, doi: 10.1145/1185347.1185358.

[79] J. Ni, C. Lin, Z. Chen, and P. Ungsunan, "A Fast Multi-pattern Matching Algorithm for Deep Packet Inspection on a Network Processor," in *36th Annual International Conference on Parallel Processing (ICPP 2007)*, Xian, PEOPLES R CHINA, Jul 10-Sep 14 2007, NEW YORK: Ieee, in International Conference on Parallel Processing Workshops, 2007, pp. 131-138.

[80] Y. N. Lin, Y. C. Chang, Y. D. Lin, and Y. C. Lai, "Resource allocation in network processors for network intrusion prevention systems," *Journal of Systems and Software,* Article vol. 80, no. 7, pp. 1030-1036, Jul 2007, doi: 10.1016/j.jss.2007.01.032.

[81] F. Yu, R. H. Katz, and T. V. Lakshman, "Gigabit rate packet pattern-matching using TCAM," *12th IEEE International Conference on Network Protocols – Proceedings,* Proceedings Paper pp. 174-183, 2004, doi: 10.1109/icnp.2004.1348108.

[82] J. S. Sung, S. M. Kang, Y. Lee, T. G. Kwon, and B. T. Kim, "A multi-gigabit rate deep packet inspection algorithm using TCAM," in *IEEE Global Telecommunications Conference (GLOBECOM 05)*, St Louis, MO, Nov 28-Dec 02 2005, NEW YORK: Ieee, in IEEE Global Telecommunications Conference (Globecom), 2005, pp. 453-457.

[83] J. Moscola, J. Lockwood, R. P. Loui, and M. Pachos, "Implementation of a content-scanning module for an Internet firewall," *Fccm 2003: 11th Annual Ieee Symposium on Field-Programmable Custom Computing Machines, Proceedings,* Proceedings Paper pp. 31-38, 2003.

[84] J. S. Sung, S. M. Kang, and T. G. Kwon, "A fast pattern-matching algorithm for network intrusion detection system," *Networking 2006: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems,* Article; Proceedings Paper vol. 3976, pp. 1157-1162, 2006.

[85] K. Xinidis, I. Charitakis, S. Antonatos, K. G. Anagnostakis, and E. P. C. C. Xu, S. H. Chen, J. S. Su, S. M. Yiu, and L. C. K. Hui, "A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, and Hardware Platforms," *IEEE Communications Surveys and Tutorials,* Article vol. 18, no. 4, pp. 2991-3029, 2016, doi: 10.1109/comst.2016.2566669.

[86] "Support. Intel Xeon Phi Processors." Intel. https://www.intel.com/content/www/us/en/support/products/75557/processors/intel-xeon-phi-processors.html (accessed 09.04, 2019).

[87] D. P. Scarpazza, O. Villa, and F. Petrini, "High-speed string searching against large dictionaries on the Cell/BE processor," *2008 IEEE*

International Symposium on Parallel & Distributed Processing, Vols 1-8, Proceedings Paper pp. 1729-+, 2008.

[88] Y. H. E. Yang and V. K. Prasanna, "Optimizing regular expression matching with SR-NFA on multi-core systems," in *Parallel Architectures and Compilation Techniques – Conference Proceedings, PACT*, 2011, pp. 424-433, doi: 10.1109/PACT.2011.73.

[89] J. D. Owens *et al.*, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum,* Review vol. 26, no. 1, pp. 80-113, 2007, doi: 10.1111/j.1467-8659.2007.01012.x.

[90] А. В. Боресков and А. А. Харламов, *Основы работы с технологией CUDA*. М.: ДМК Пресс, 2010, p. 232.

[91] А. В. Палагин and В. Н. Опанасенко, Реконфигурируемые вычислительные системы: Основы и приложения. К.: «Просвіта», 2006, p. 280.

[92] А. О. Мельник and В. А. Мельник, *Персональні суперкомп'ютери*. Львів: Видавництво Національного університету «Львівська політехніка», 2012, p. 600.

[93] "FPGA Boards and Systems." http://www.fpga-faq.com/FPGA_Boards.shtml (accessed 15.09, 2019).

[94] V. F. Evdokimov, A. M. Davydenko, S. Y. Hilgurt, and O. R. Yarema, "Hardware platform for reconfigurable information security tools," (in Ukrainian), *Modeling and information technologies. Collection of scientific works of PIMEE NAS of Ukraine,* no. 86, pp. 3-11, 2019, doi: 10.5281/zenodo.3610626.

[95] R. Abdulhammed, M. Faezipour, and K. M. Elleithy, "Network Intrusion Detection Using Hardware Techniques: A Review," *2016 IEEE Long Island Systems, Applications and Technology Conference (Lisat),* Proceedings Paper p. 7, 2016.

[96] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis, "Gnort: High Performance Network Intrusion Detection Using Graphics Processors," *Recent Advances in Intrusion Detection, Raid 2008,* Proceedings Paper vol. 5230, pp. 116-134, 2008.

[97] T. Song, W. Zhang, D. S. Wang, and Y. B. Xue, "A memory efficient multiple pattern matching architecture for network security," *27th IEEE Conference on Computer Communications (Infocom), Vols 1-5,* Proceedings Paper pp. 673-681, 2008.

[98] H. J. Jung, Z. K. Baker, and V. K. Prasanna, "Performance of FPGA implementation of bit-split architecture for intrusion detection systems," in *20th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2006*, 2006, vol. 2006: IEEE Computer Society, doi: 10.1109/IPDPS.2006.1639434.

[99] Y. H. E. Yang and V. K. Prasanna, "Memory-Efficient Pipelined Architecture for Large-Scale String Matching," *Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines,* Proceedings Paper pp. 104-111, 2009, doi: 10.1109/fccm.2009.17.

[100] T. Huffmire *et al.*, "Designing secure systems on reconfigurable hardware," *Acm Transactions on Design Automation of Electronic Systems,* Article; Proceedings Paper vol. 13, no. 3, p. 24, Jul 2008, Art no. 44, doi: 10.1145/1367045.1367053.

[101] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis, "Regular Expression Matching on Graphics Hardware for Intrusion Detection," *Recent Advances in Intrusion Detection, Proceedings,* Proceedings Paper vol. 5758, pp. 265-283, 2009.

[102] Y. Kaneta, S. Minato, and H. Arimura, "Fast Bit-Parallel Matching for Network and Regular Expressions," *String Processing and Information Retrieval,* Proceedings Paper vol. 6393, pp. 372-384, 2010.

[103] H. Chen, Y. Chen, and D. H. Summerville, "A Survey on the Application of FPGAs for Network Infrastructure Security," *IEEE Communications Surveys and Tutorials,* Article vol. 13, no. 4, pp. 541-561, 2011, doi: 10.1109/surv.2011.072210.00075.

[104] U. Yuichiro, M. Inagi, S. Wakabayashi, and S. Nagayama, "A GPGPU Implementation of Approximate String Matching with Regular Expression Operators and Comparison with Its FPGA Implementation," in *The 2012*

International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'12), 2012, pp. 115-120.

[105] R. Nehemia, C. Lim, M. Galinium, and A. R. Widianto, "Implementation of Multipattern String Matching Accelerated with GPU for Intrusion Detection System," in *3rd IAES International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Univ Islam Sultan Agung, Semarang, INDONESIA, Nov 23-25 2016, vol. 190, BRISTOL: Iop Publishing Ltd, in IOP Conference Series-Materials Science and Engineering, 2017, doi: 10.1088/1757-899x/190/1/012023.

[106] V. Jyothi, S. K. Addepalli, and R. Karri, "DPFEE: A High Performance Scalable Pre-Processor for Network Security Systems," *IEEE Transactions on Multi-Scale Computing Systems,* Article vol. 4, no. 1, pp. 55-68, Jan-Mar 2018, doi: 10.1109/tmscs.2017.2765324.

[107] С. Н. Шипулин, Д. А. Губанов, В. Б. Стешенко, and В. Ю. Храпов, "Тенденции развития ПЛИС и их применение для цифровой обработки сигналов," *Электронные компоненты,* no. 5, pp. 42-45, 1999.

[108] С. Я. Гильгурт, "Анализ применений программируемых ИС для решения задач вычислительной техники," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 19, pp. 79-85, 2003.

[109] "Xilinx." Xilinx. http://www.xilinx.com (accessed 15.09, 2019).

[110] "Intel FPGAs and programmable devices." https://www.intel.com/content/www/us/en/products/programmable.html (accessed 15.09, 2019).

[111] В. Ю. Зотов, Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPACK ISE. М.: Горячая линия-Телеком, 2003, p. 624.

[112] В. Б. Стешенко, ПЛИС фирмы Altera: элементная база, система проектирования и языки описания аппаратуры. М.: Издательский дом "Додэка-XXI", 2002, p. 576.

[113] В. В. Соловьев and А. Г. Васильев, *Программируемые логические интегральные схемы и их применение*. Минск: Беларуская навука, 1998, p. 270.

[114] С. Н. Шипулин and В. Ю. Храпов, "Основные тенденции развития ПЛИС," *Электронные компоненты,* no. 3-4, pp. 26-27, 1996.

[115] С. Я. Гильгурт, "Обзор возможностей реконфигурируемых устройств для применения в компьютерной безопасности," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 55, pp. 117-124, 2010.

[116] В. Ф. Гузик, И. А. Каляев, and И. И. Левин, *Реконфигурируемые вычислительные системы: учебное пособие*. Ростов-на-Дону: Южный федеральный университет, Московский государственный университет имени М. В. Ломоносова, 2016, p. 472.

[117] Д. С. Потехин and И. Е. Тарасов, *Разработка систем цифровой обработки сигналов на базе ПЛИС*. М.: Горячая линия – Телеком, 2007, p. 248.

[118] В. Н. Опанасенко, "Высокопроизводительные реконфигурируемые компьютеры на базе FPGA," *Проблеми інформатизації та управління. – К.: НАУ,* vol. 27, no. 3, pp. 114-118, 2009.

[119] Р. И. Грушвицкий, А. Х. Мурсаев, and Е. П. Угрюмов, *Проектирование систем на микросхемах программируемой логики*. СПб.: БХВ-Петербург, 2002, p. 608.

[120] Е. А. Суворова and Ю. Е. Шейнин, *Проектирование цифровых систем на VHDL*. СПб.: БХВ-Петербург, 2003, p. 576.

[121] А. М. Сергиенко, *VHDL для проектирования вычислительных устройств*. К.: ЧП "Корнейчук", ООО "ТИД "ДС", 2003, p. 208.

[122] А. М. Сергиенко and В. И. Корнейчук*, Микропроцессорные устройства на программируемых логических ИС*. К.: «Корнійчук», 2005, p. 108.

[123] К. Максфилд, *Проектирование на ПЛИС. Курс молодого бойца*. М.: Издательский дом «Додэка-XXI», 2007, p. 408.

[124] И. А. Каляев, И. И. Левин, Е. А. Семерников, and В. И. Шмойлов, *Реконфигурируемые мультиконвейерные вычислительные структуры*. Ростов-на-Дону: Издательство ЮНЦ РАН, 2008, p. 393.

[125] I. Sourdis and D. Pnevmatikatos, "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching," *12th Annual IEEE Symposium on*

Field-Programmable Custom Computing Machines, Proceedings, Proceedings Paper pp. 258-267, 2004, doi: 10.1109/fccm.2004.46.

[126] В. Ю. Зотов, "Особенности архитектуры нового поколения ПЛИС с архитектурой FPGA фирмы Xilinx," *Компоненты и технологии,* no. 12, 2010.

[127] С. Я. Гильгурт, "Применение типовых устройств на базе программируемой логики для решения вычислительных задач," in *Труды II Междунар. конф.«Параллельные вычисления и задачи управления» PACO'2004 памяти Е.Г. Сухова*, 2004: Институт проблем управления им. В.А. Трапезникова РАН, pp. 514-530.

[128] С. Я. Гильгурт, "Анализ типовых режимов обмена данными с реконфигурируемыми вычислителями," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 59, pp. 113-121, 2011.

[129] С. Я. Гильгурт, "Обзор современных реконфигурируемых унифицированных вычислителей," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 49, pp. 17-24, 2008.

[130] "Bitware. A Molex company." https://www.bittware.com (accessed 15.09, 2019).

[131] Ю. М. Коростиль and С. Я. Гильгурт, "Перспективы развития реконфигурируемых вычислителей для выполнения ресурсоемких расчетов," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 68, pp. 84-91, 2013.

[132] В. Ф. Евдокимов, А. Н. Давиденко, and С. Я. Гильгурт, "Централизованный синтез вычислительных структур реконфигурируемых ускорителей задач информационной безопасности," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 82, pp. 3-11, 2018.

[133] T. Katashita, Y. Yamaguchi, A. Maeda, and K. Toda, "FPGA-based intrusion detection system for 10 Gigabit Ethernet," *Ieice Transactions on Information and Systems,* Article vol. E90D, no. 12, pp. 1923-1931, Dec 2007, doi: 10.1093/ietisy/e90-d.12.1923.

[134] Y. H. Cho, S. Navab, and W. H. Mangione-Smith, "Specialized hardware for deep network packet filtering," *Field-Programmable Logic and Applications, Proceedings: Reconfigurable Computing Is Going Mainstream,* Article; Proceedings Paper vol. 2438, pp. 452-461, 2002.

[135] I. Sourdis and D. Pnevmatikatos, "Fast, large-scale string match for a 10Gbps FPGA-based network Intrusion Detection System," *Field-Programmable Logic and Applications, Proceedings,* Article; Proceedings Paper vol. 2778, pp. 880-889, 2003.

[136] S. Antonatos, K. G. Anagnostakis, and E. P. Markatos, "Generating realistic workloads for network intrusion detection systems," in *Proceedings of the Fourth International Workshop on Software and Performance, WOSP'04*, 2004, pp. 207-215, doi: 10.1145/974043.974078.

[137] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks-the International Journal of Computer and Telecommunications Networking,* Article vol. 31, no. 23-24, pp. 2435-2463, Dec 1999, doi: 10.1016/s1389-1286(99)00112-7.

[138] G. S. Shenoy, J. Tubella, and A. Gonzalez, "Hardware/Software Mechanisms for Protecting an IDS Against Algorithmic Complexity Attacks," *2012 Ieee 26th International Parallel and Distributed Processing Symposium Workshops & Phd Forum (Ipdpsw),* Proceedings Paper pp. 1190-1196, 2012, doi: 10.1109/ipdpsw.2012.145.

[139] С. Гильгурт, Б. Дурняк, and Ю. Коростиль, "Противодействие атакам алгоритмической сложности на системы обнаружения вторжений," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 71, pp. 3-12, 2014.

[140] L. Tan and T. Sherwood, "A high throughput string matching architecture for intrusion detection and prevention," in *32nd International Symposium on Computer Architecture*, Madison, WI, Jun 04-08 2005, LOS ALAMITOS: Ieee Computer Soc, in Conference Proceedings Annual International Symposium on Computer Architecture, 2005, pp. 112-122.

[141] С. Гільгурт, "Побудова скінченних автоматів реконфігуровними засобами для вирішення задач інформаційної безпеки," *Захист*

інформації, vol. 21, no. 2, pp. 111-120, 2019, doi: 10.18372/2410-7840.21.13768.

[142] S. Dharmapurikar and J. Lockwood, "Fast and scalable pattern matching for content filtering," in *2005 Symposium on Architectures for Networking and Communications Systems (ANCS)*, Princeton, NJ, USA, 26-28 Oct. 2005 2005: IEEE, pp. 183-192, doi: 10.1145/1095890.1095916.

[143] J. Lunteren and T. Engbersen, "Fast and scalable packet classification," *Ieee Journal on Selected Areas in Communications,* Article; Proceedings Paper vol. 21, no. 4, pp. 560-571, May 2003, doi: 10.1109/jsac.2003.810527.

[144] С. Я. Гильгурт, "Применение реконфигурируемых вычислений для решения задач распознавания," *Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 16, pp. 41-47, 2002.

[145] R. Sidhu and V. K. Prasanna, "Fast Regular Expression Matching Using FPGAs," in *9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2001*, 2001: Institute of Electrical and Electronics Engineers Inc., pp. 227-238.

[146] F. Yu, Z. Chen, Y. Diao, T. V. Lakshman, and R. H. Katz, "Fast and memory-efficient regular expression matching for deep packet inspection," in *2nd ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2006*, San Jose, CA, 2006, pp. 93-102, doi: 10.1145/1185347.1185360.

[147] J. Bispo, L. Sourdis, J. M. P. Cardoso, and S. Vassiliadis, "Regular expression matching for reconfigurable packet inspection," *2006 IEEE International Conference on Field Programmable Technology, Proceedings,* Proceedings Paper pp. 119-126, 2006, doi: 10.1109/fpt.2006.270302.

[148] J. Kim and J. Park, "FPGA-based memory efficient shift-and algorithm for regular expression matching. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)," in *14th International Symposium on Applied Reconfigurable Computing, ARC 2018* vol. 10824 LNCS, N. Voros, G. Keramidas, C. Antonopoulos, M. Huebner, P. C. Diniz, and D. Goehringer, Eds., ed: Springer Verlag, 2018, pp. 132-141.

[149] S. Prithi and S. Sumathi, "A survey on recent DFA compression techniques for deep packet inspection in network intrusion detection system," *Journal of Electrical Engineering,* 2014.

[150] А. Н. Давиденко and С. Я. Гильгурт, "Алгоритмы распознавания строк в системах обнаружения вторжений на ПЛИС," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 58, pp. 103-109, 2010.

[151] С. Я. Гільгурт, "Побудова асоціативної пам'яті на цифрових компараторах реконфігуровними засобами для вирішення задач інформаційної безпеки," *Электронное моделирование,* vol. 41, no. 3, pp. 59-80, 2019, doi: 10.15407/emodel .41.03.059.

[152] С. Гільгурт, "Побудова фільтрів Блума реконфігуровними засобами для вирішення задач інформаційної безпеки," *Безпека інформації,* vol. 35, no. 1, pp. 53-58, 2019, doi: 10.18372/2225-5036.25.13594.

[153] A. G. Hanlon, "Content-Addressable and Associative Memory Systems a Survey," *IEEE Transactions on Electronic Computers,* vol. EC-15, no. 4, pp. 509-521, 1966, doi: 10.1109/PGEC.1966.264358.

[154] I. N. Robinson, "Pattern-addressable memory," *IEEE Micro,* Article vol. 12, no. 3, pp. 20-30, Jun 1992, doi: 10.1109/40.141600.

[155] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE Journal of Solid-State Circuits,* Article vol. 41, no. 3, pp. 712-727, Mar 2006, doi: 10.1109/jssc.2005.864128.

[156] T. Kohonen, *Content-Addressable Memories*, 2 ed. (Springer Series in Information Sciences, no. 1). New York: Springer-Verlag Berlin Heidelberg, 1987.

[157] R. Neale, "Is content addressable memory (CAM) the key to network success?," *Electronic Engineering,* Editorial Material vol. 71, no. 865, pp. 9-12, Feb 1999.

[158] "Net Logic Microsystems." http://www.netlogicmicro.com (accessed 03.05, 2009).

[159] "Music-IC." https://music-ic.com (accessed 15.09, 2019).

[160] S. A. Guccione, D. Levi, and D. Downs, "A reconfigurable content addressable memory," *Parallel and Distributed Processing, Proceedings,* Article; Proceedings Paper vol. 1800, pp. 882-889, 2000.

[161] M. Iliopoulos and T. Antonakopoulos, "Reconfigurable network processors based on field programmable system level integrated circuits," in *10th International Conference on Field-Programmable Logic and Applications, FPL 2000* vol. 1896, H. Grunbacher and R. W. Hartenstein, Eds., ed: Springer Verlag, 2000, pp. 39-47.

[162] K. Xinidis, K. G. Anagnostakis, and E. P. Markatos, "Design and implementation of a high-performance network intrusion prevention system," in *IFIP TC11 20th International Information Security Conference, IFIP/SEC2005* vol. 181, ed. Chiba, 2005, pp. 359-374.

[163] J. Huang, Z. K. Yang, X. Du, and W. Liu, "FPGA based high speed and low area cost pattern matching," in *IEEE Region 10 Conference (TENCON 2005)*, Melbourne, AUSTRALIA, Nov 21-24 2005, NEW YORK: Ieee, in TENCON IEEE Region 10 Conference Proceedings, 2006, pp. 2693-2697.

[164] I. Sourdis, D. N. Pnevmatikatos, and S. Vassiliadis, "Scalable multigigabit pattern matching for packet inspection," *IEEE Transactions on Very Large Scale Integration (Vlsi) Systems,* Article vol. 16, no. 2, pp. 156-166, Feb 2008, doi: 10.1109/tvls1.2007.912036.

[165] M. Gokhale, D. Dubois, A. Dubois, M. Boorman, S. Poole, and V. Hogsett, "Granidt: Towards gigabit rate network intrusion detection technology," *Field-Programmable Logic and Applications, Proceedings: Reconfigurable Computing Is Going Mainstream,* Article; Proceedings Paper vol. 2438, pp. 404-413, 2002.

[166] "Spartan-6 FPGA Configurable Logic Block User Guide," in *UG384*, 1.0 ed: Xilinx, 2009, p. 50.

[167] "7 Series FPGAs Configurable Logic Block User Guide," in *UG474*, 1.8 ed: Xilinx, 2016, p. 74.

[168] C. R. Clark and D. E. Schimmel, "Scalable pattern matching for high speed networks," *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Proceedings,* Proceedings Paper pp. 249-257, 2004, doi: 10.1109/fccm.2004.50.

[169] S. Yusuf and W. Luk, "Bitwise optimised cam for network intrusion detection systems," in *2005 International Conference on Field Programmable Logic and Applications, FPL*, Tampere, 2005, vol. 2005, pp. 444-449, doi: 10.1109/FPL.2005.1515762.

[170] Д. Э. Кнут, Искусство программирования, том 4, А. Комбинаторные алгоритмы. Москва: ООО "И.Д. Вильямс", 2013, p. 960.

[171] S. Hazelhurst, A. Fatti, and A. Henwood, "Binary decision diagram representations of firewall and router access lists," Johannesburg , South Africa, 1998.

[172] S. A. Guccione and D. Levi, "XBI: A Java-based interface to FPGA hardware," in *Conference on Configurable Computing – Technology and Applications*, Boston, Ma, Nov 02-03 1998, vol. 3526, BELLINGHAM: Spie-Int Soc Optical Engineering, in Proceedings of the Society of Photo-Optical Instrumentation Engineers (Spie), 1998, pp. 97-102, doi: 10.1117/12.327023.

[173] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM,* Article vol. 13, no. 7, pp. 422-426, 1970, doi: 10.1145/362686.362692.

[174] D. V. Pryor, M. R. Thistle, and N. Shirazi, "Text searching on Splash 2," in *IEEE Workshop on FPGAs for Custom Computing Machines*, April 1993 1993, 1993, pp. 172-177.

[175] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics,* vol. 1, no. 4, pp. 485-509, 2004, doi: 10.1080/15427951.2004.10129096.

[176] S. Geravand and M. Ahmadi, "Bloom filter applications in network security: A state-of-the-art survey," *Computer Networks,* Article vol. 57, no. 18, pp. 4047-4064, Dec 2013, doi: 10.1016/j.comnet.2013.09.003.

[177] J. Harwayne-Gidansky, D. Stefan, and I. Dalal, "FPGA-based SoC for Real-Time Network Intrusion Detection using Counting Bloom Filters," presented at the Proceedings of the IEEE Southeastcon 2009, Technical Proceedings, 2009, Proceedings Paper.

[178] S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull, and J. W. Lockwood, "Deep packet inspection using parallel bloom filters," *IEEE Micro,* Article;

Proceedings Paper vol. 24, no. 1, pp. 52-61, Jan-Feb 2004, doi: 10.1109/mm.2004.1268997.

[179] S. Dharmapurikar, M. Attig, and J. Lockwood, "Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based Bloom Filters," in "All Computer Science and Engineering Research," Washington University in St. Louis, WUCSE-2004-12, 2004-03-25 2004.

[180] Y. Chen, A. Kumar, and J. Xu, "A new design of bloom filter for packet inspection speedup," in *IEEE Global Telecommunications Conference (GLOBECOM 07)*, Washington, DC, Nov 26-30 2007, NEW YORK: Ieee, in IEEE Global Telecommunications Conference (Globecom), 2007, pp. 1-5.

[181] L. Carter and M. Wegman, "Universal Classes of Hashing Functions," *Computer and System Sciences,* vol. 18, no. 2, pp. 143-154, 1979.

[182] M. Ramakrishna, E. Fu, and E. Bahcekapili, "A Performance Study of Hashing Functions for Hardware Applications," in *6th Int'l Conf. Computing and Information ICCI'94*, 1994, pp. 1621-1636.

[183] M. V. Ramakrishna, E. Fu, and E. Bahcekapili, "Efficient hardware hashing functions for high performance computers," *Ieee Transactions on Computers,* Article vol. 46, no. 12, pp. 1378-1381, Dec 1997, doi: 10.1109/12.641938.

[184] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and Practice of Bloom Filters for Distributed Systems," *IEEE Communications Surveys and Tutorials,* Article vol. 14, no. 1, pp. 131-155, 2012, doi: 10.1109/surv.2011.031611.00024.

[185] M. Breternitz, Y. Wu, P. G. Sassone, J. P. Rupley, W. Attrot, and B. Black, "Efficient Bloom filter," USA Patent US2008147714 (A1) Patent Appl. 11/643,314, 06/19/2008, 2008.

[186] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Method and apparatus for detecting predefined signatures in packet payload using Bloom filters," USA Patent US2005086520 (A1) Patent Appl. 10/640,513, 04/21/2005, 2005.

[187] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection," *IEEE Communications Letters,* Article vol. 10, no. 3, pp. 210-212, Mar 2006, doi: 10.1109/lcomm.2006.1603387.

[188] H. Kaya and T. Kocak, "A low power lookup technique for multi-hashing network applications," *IEEE Computer Society Annual Symposium on Vlsi, Proceedings: Emerging Vlsi Technologies and Architectures,* Proceedings Paper pp. 179-+, 2006.

[189] N. S. Artan, K. Sinkar, J. Patel, and H. J. Chao, "Aggregated bloom filters for intrusion detection and prevention hardware," in *IEEE Global Telecommunications Conference (GLOBECOM 07)*, Washington, DC, Nov 26-30 2007, NEW YORK: Ieee, in IEEE Global Telecommunications Conference (Globecom), 2007, pp. 349-354.

[190] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol," *IEEE ACM Transactions on Networking,* Article vol. 8, no. 3, pp. 281-293, Jun 2000, doi: 10.1109/90.851975.

[191] К. Г. Самофалов, А. М. Ромлинкевич, В. Н. Валуйский, Ю. С. Каневский, and М. М. Пиневич, *Прикладная теория цифровых автоматов*. Київ: Вища шк. Головное изд-во, 1987.

[192] Ю. Г. Карпов, *Теория автоматов*. СПб.: Питер, 2002, p. 206.

[193] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of the Computations*, 2nd ed. Prentice-Hall 1998.

[194] R. M. Keller, Computer Science: Abstraction to Implementation. Harvey Mudd College, 2001.

[195] C. H. Lin, Y. T. Tai, and S. C. Chang, "Optimization of pattern matching algorithm for memory based architecture," in *3rd ACM/IEEE Symposium on Architecture for networking and communications systems, ANCS 2007*, Orlando, Florida, USA, 2007, 1323551: ACM, pp. 11-16, doi: 10.1145/1323548.1323551.

[196] C. H. Lin and S. C. Chang, "Efficient Pattern Matching Algorithm for Memory Architecture," *IEEE Transactions on Very Large Scale Integration (VlSI) Systems,* Article vol. 19, no. 1, pp. 33-41, Jan 2011, doi: 10.1109/tvlsi.2009.2028346.

[197] Q. B. Wang and V. K. Prasanna, "Multi-Core Architecture on FPGA for Large Dictionary String Matching," *Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines,* Proceedings Paper pp. 96-103, 2009, doi: 10.1109/fccm.2009.43.

[198] K. Huang and D. F. Zhang, "A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection," *Proceedings of the 9th International Conference for Young Computer Scientists, Vols 1-5,* Proceedings Paper pp. 2073-2078, 2008.

[199] D. Pao, W. Lin, and B. Liu, "Pipelined architecture for multi-string matching," *IEEE Computer Architecture Letters,* vol. 7, no. 2, pp. 33-36, 2008, doi: 10.1109/L-CA.2008.5.

[200] H. Lu, K. Zheng, B. Liu, X. Zhang, and Y. H. Liu, "A memory-efficient parallel string matching architecture for high-speed intrusion detection," *IEEE Journal on Selected Areas in Communications,* Article vol. 24, no. 10, pp. 1793-1804, Oct 2006, doi: 10.1109/jsac.2006.877221.

[201] "16 Гбайт памяти HBM в ПЛИС Virtex® UltraScale+™ от Xilinx (24.09.2019 15:15) – Новости." Макрогрупп. https://www.macrogroup.ru/news/2019/16-gbayt-pamyati-hbm-v-plis-virtexr-ultrascaletm-ot-xilinx (accessed 24.09, 2019).

[202] "Memory Interface Solutions User Guide," in *UG086*, 3.6 ed: Xilinx, 2010, p. 598.

[203] G. Papadopoulos and D. Pnevmatikatos, "Hashing + Memory = low cost, exact pattern matching," in *Proceedings – 2005 International Conference on Field Programmable Logic and Applications, FPL*, 2005, vol. 2005, pp. 39-44, doi: 10.1109/FPL.2005.1515696.

[204] I. Sourdis, D. Pnevmatikatos, S. Wong, and S. Vassiliadis, "A reconfigurable perfect-hashing scheme for packet inspection," in *Proceedings – 2005 International Conference on Field Programmable Logic and Applications, FPL*, 2005, vol. 2005, pp. 644-647, doi: 10.1109/FPL.2005.1515804.

[205] Z. K. Baker and V. K. Prasanna, "A methodology for synthesis of efficient intrusion detection systems on FPGAs," *12th Annual IEEE Symposium on*

Field-Programmable Custom Computing Machines, Proceedings, Proceedings Paper pp. 135-144, 2004, doi: 10.1109/fccm.2004.6.

[206] Y. H. Cho and W. H. Mangione-Smith, "Fast reconfiguring deep packet filter for 1+gigabit network," *FCCM 2005: 13th Annual Ieee Symposium on Field-Programmable Custom Computing Machines, Proceedings,* Proceedings Paper pp. 215-224, 2005.

[207] Y. H. Cho and W. H. Mangione-Smith, "Deep network packet filter design for reconfigurable devices," *Acm Transactions on Embedded Computing Systems,* Article; Proceedings Paper vol. 7, no. 2, p. 26, 2008, Art no. 21, doi: 10.1145/1331331.1331345.

[208] В. Опанасенко, С. Крывый, and С. Завьялов, "Формализация процесса проектирования проблемно-ориентированных устройств на базе FPGA," *International Journal "Information Technologies & Knowledge,* vol. 13, no. 1, pp. 3-16.

[209] А. В. Палагин and В. Н. Опанасенко, "Проектирование реконфигурируемых систем на ПЛИС," *Технология и конструирование в электронной аппаратуре,* no. 3, pp. 15-19, 2007.

[210] С. Гільгурт, "Методи побудови оптимальних схем розпізнавання для реконфігуровних засобів інформаційної безпеки," *Безпека інформації,* vol. 25, no. 2, pp. 74-81, 2019, doi: 10.18372/2225-5036.25.13824.

[211] O. Erdem, H. Le, and V. K. Prasanna, "Clustered hierarchical search structure for large-scale packet classification on FPGA," in *Proceedings – 21st International Conference on Field Programmable Logic and Applications, FPL 2011*, 2011, pp. 201-206, doi: 10.1109/FPL.2011.44.

[212] M. J. Ye, K. Xu, J. P. Wu, and Y. Cui, "A high performance and scalable packet pattern-matching architecture," *2008 the International Conference on Information Networking,* Proceedings Paper pp. 425-429, 2008.

[213] S. H. Zhang, Z. Qin, Y. Zhao, and X. L. Peng, "A Cascade Hash Design of Bloom Filter for Signature Detection," *2009 International Forum on Information Technology and Applications, Vol 2, Proceedings,* Proceedings Paper pp. 559-+, 2009, doi: 10.1109/ifita.2009.129.

[214] M. Paynter and T. Kocak, "Fully Pipelined Bloom Filter Architecture," *IEEE Communications Letters,* Article vol. 12, no. 11, pp. 855-857, Nov 2008, doi: 10.1109/lcomm.2008.081176.

[215] M. Arun, A. Krishnan, and S. Periasamy, "Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based low power multiple-hashing Bloom Filters," *International Journal Of Computer Science And Applications,* vol. 1, no. 3, pp. 186-189, 2008.

[216] S. Dharmapurikar and J. W. Lockwood, "Fast and scalable pattern matching for network intrusion detection systems," *Ieee Journal on Selected Areas in Communications,* Article vol. 24, no. 10, pp. 1781-1792, Oct 2006, doi: 10.1109/jsac.2006.877131.

[217] V. Dimopoulos, I. Papaefstathiou, and D. Pnevmatikatos, "A memory-efficient reconfigurable Aho-Corasick FSM implementation for intrusion detection systems," *Ic-Samos: 2007 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, Proceedings,* Proceedings Paper pp. 186-193, 2007.

[218] Y. H. Cho and W. H. Mangione-Smith, "A pattern matching co-processor for network security," in *42nd Design Automation Conference*, Anaheim, CA, Jun 13-17 2005, LOS ALAMITOS: Ieee Computer Soc, in Design Automation Conference DAC, 2005, pp. 234-239.

[219] M. Alicherry, M. Muthuprasanna, and V. Kumar, "High speed pattern matching for network IDS/IPS," *Proceedings of the 2006 IEEE International Conference on Network Protocols,* Proceedings Paper pp. 187-196, 2006.

[220] C. L. Hayes and Y. Luo, "DPICO: A high speed deep packet inspection engine using compact finite automata," in *3rd ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2007*, Orlando, FL, 2007, pp. 195-203, doi: 10.1145/1323548.1323579.

[221] P. Piyachon and Y. Luo, "Design of high performance pattern matching engine through compact deterministic finite automata," in *45th ACM/IEEE Design Automation Conference*, Anaheim, CA, Jun 08-13 2008, NEW YORK: IEEE, in Design Automation Conference, 2008, pp. 852-857.

[222] K. Huang, D. F. Zhang, and Z. Qin, "Accelerating the bit-split string matching algorithm using Bloom filters," *Computer Communications,* Article vol. 33, no. 15, pp. 1785-1794, Sep 2010, doi: 10.1016/j.comcom.2010.04.043.

[223] С. Я. Гильгурт, "Организация вычислительного процесса синтеза файлов конфигураций для аппаратных ускорителей при решении задач информационной безопасности," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 74, pp. 29-33, 2015.

[224] A. Mitra, W. Najjar, and L. Bhuyan, "Compiling PCRE to FPGA for accelerating SNORT IDS," in *3rd ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2007*, Orlando, FL, 2007, pp. 127-135, doi: 10.1145/1323548.1323571.

[225] В. Ф. Евдокимов, А. Н. Давиденко, and С. Я. Гильгурт, "Дополнительные этапы процедуры оперативной реконфигурации аппаратных ускорителей задач информационной безопасности," *Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України,* no. 85, pp. 3-11, 2018.

[226] В. Ф. Євдокимов, А. М. Давиденко, and С. Я. Гільгурт, "Створення на базі грід-сайту ІПМЕ ім. Г.Є. Пухова НАНУ системи централізованого синтезу апаратних прискорювачів для вирішення задач інформаційної безпеки в енергетичній галузі," *Моделювання та інформаційні технології,* no. 79, pp. 3-8, 2017.

[227] А. А. Сальников, В. В. Вишневский, and А. Ф. Борецкий, "«Платформа как сервис» в грид для интерактивного анализа медицинских данных," *Математичні машини і системи,* no. 1, pp. 53-64, 2015.

ДОДАТКИ

Додаток А. Список публікацій здобувача за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Гильгурт С.Я. Применение реконфигурируемых вычислений для решения задач распознавания / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2002. – Вип. 16. – С.41-47.

2. Гильгурт С.Я. Анализ применений программируемых ИС для решения задач вычислительной техники / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2003. – Вип. 19. – С.79-85.

3. Гильгурт С.Я. Особенности применения реконфигурируемых вычислителей для аппаратной защиты информационных систем / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2007. – Вип. 38. – С.36-41.

4. Гильгурт С.Я. О применении реконфигурируемых вычислителей для решения задач защиты информации / С.Я. Гильгурт, А.К. Гиранова // Зб. наук. праць ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2008. – Вип. 46. – С.93-99.

5. Гильгурт С.Я. Обзор современных реконфигурируемых унифицированных вычислителей / С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2008. – Вип. 49. – С.17-24.

6. Гильгурт С.Я. Анализ применения унифицированных вычислителей в интеллектуальных системах / С.Я. Гильгурт // Искусственный интеллект. – 2009. – № 1. – С.144-148.

7. Гильгурт С.Я. О применении реконфигурируемых вычислителей для решения вопросов защиты информации, пересылаемой в грид-среде /

С.Я. Гильгурт, А.К. Гиранова // Зб. наук. праць ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2009. – Вип. 51. – С.65-72.

8. Коростиль Ю.М. Анализ угроз и опасностей в компьютерных системах на предмет защиты цифровыми реконфигурируемыми устройствами / Ю.М. Коростиль, А.Н. Давиденко, С.Я. Гільгурт, М.М. Панченко // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 56. – С.10-17.

9. Гильгурт С.Я. Обзор возможностей реконфигурируемых устройств для применения в компьютерной безопасности / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 55. – С.117-124.

10. Коростиль Ю.М. Принципы построения сетевых систем обнаружения вторжений на базе ПЛИС / Ю.М. Коростиль, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 57. – С.87-94.

11. Давиденко А.Н. Алгоритмы распознавания строк в системах обнаружения вторжений на ПЛИС / А.Н. Давиденко, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2010. – Вип. 58. – С.103-109.

12. Гильгурт С.Я. Программно-аппаратная защита данных в распределенных интеллектуальных системах / С.Я. Гильгурт, А.К. Гиранова // Искусственный интеллект. – 2010. – № 3. – С.706-711.

13. Гильгурт С.Я. Анализ типовых режимов обмена данными с реконфигурируемыми вычислителями / С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2011. – Вип. 59. – С.113-121.

14. Гильгурт С.Я. Аппаратное распознавание строк в интеллектуальных системах защиты информации / С.Я. Гильгурт // Искусственный интеллект. – 2012. – № 1. – С.259-266.

15. Гильгурт С.Я. Сравнительный анализ межсетевых экранов, систем обнаружения и предотвращения вторжений, контроля целостности /

С.Я. Гильгурт // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2011. – Вип. 60. – С.116-120.

16. Давиденко А.Н. Аппаратное ускорение алгоритмов сигнатурного обнаружения вторжений в открытой системе информационной безопасности Snort / А.Н. Давиденко, С.Я. Гильгурт, В.И. Сабат // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2012. – Вип. 65. – С.94-103.

17. Коростиль Ю.М. Анализ базы данных системы информационной безопасности Snort и вопросы быстродействия / Ю.М. Коростиль, С.Я. Гильгурт, О.М. Назаренко // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2012. – Вип. 66. – С.77-84.

18. Коростиль Ю.М. Перспективы развития реконфигурируемых вычислителей для выполнения ресурсоемких расчетов / Ю.М. Коростиль, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2013. – Вип. 68. – С.84-91.

19. Гильгурт С.Я. Реконфигурируемые вычислители. Аналитический обзор / С.Я. Гильгурт // Электронное моделирование. – 2013. – Т. 35, № 4. – С.49-72.

20. Гильгурт С.Я. Противодействие атакам алгоритмической сложности на системы обнаружения вторжений / С.Я. Гильгурт, Б.В. Дурняк, Ю.М. Коростиль // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2014. – Вип. 71. – С.3-12.

21. Hilhurt S. Ya. Application of FPGA-based Reconfigurable Accelerators for Network Security Tasks / S. Hilhurt // Collection of scientific works. Simulation and informational technologies. – PIMEE NAS of Ukraine. – Kyiv, 2014. – Vol. 73. – P. 17-26.

22. Гильгурт С.Я. Организация вычислительного процесса синтеза файлов конфигураций для аппаратных ускорителей при решении задач информационной безопасности / С.Я. Гильгурт // Моделювання та

інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2015. – Вип. 74. – С.29-33.

23. Євдокимов В.Ф. Створення на базі грід-сайту ІПМЕ ім. Г.Є. Пухова НАНУ системи централізованого синтезу апаратних прискорювачів для вирішення задач інформаційної безпеки в енергетичній галузі / В.Ф. Євдокимов, А.М. Давиденко, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2017. – Вип. 79. – С.3-8.

24. Гильгурт С.Я. Анализ применения аппаратного ускорения информационной защиты в автоматизированных системах энергетической отрасли / С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2018. – Вип. 83. – С.154-164.

25. Евдокимов В.Ф. Дополнительные этапы процедуры оперативной реконфигурации аппаратных ускорителей задач информационной безопасности / В.Ф. Евдокимов, А.Н. Давиденко, С.Я. Гильгурт // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова НАН України. – Київ, 2018. – Вип. 85. – С.3-11.

26. Евдокимов В. Централизованный синтез реконфигурируемых аппаратных средств информационной безопасности на высокопроизводительных платформах / В. Евдокимов, А. Давиденко, С. Гильгурт // Захист інформації. – 2018. – Т. 20, № 4. – С.247-258.

27. Гільгурт С.Я. Побудова асоціативної пам'яті на цифрових компараторах реконфігуровними засобами для вирішення задач інформаційної безпеки / С.Я. Гільгурт // Електронне моделювання. – 2019. – Т. 41, № 3. – С.59-80.

28. Гільгурт С. Побудова фільтрів Блума реконфігуровними засобами для вирішення задач інформаційної безпеки / С. Гільгурт // Безпека інформації. – 2019. – Т. 25, № 1. – С.53-58.

29. Гільгурт С. Побудова скінченних автоматів реконфігуровними засобами для вирішення задач інформаційної безпеки / С. Гільгурт // Захист інформації. – 2019. – Т. 21, № 2. – С.111-120.

30. Гільгурт С. Методи побудови оптимальних схем розпізнавання для реконфігуровних засобів інформаційної безпеки / С. Гільгурт // Безпека інформації. – 2019. – Т. 25, № 2. – С.74-81.

Наукові праці, які засвідчують апробацію матеріалів дисертації

1. Гильгурт С.Я. Применение типовых устройств на базе программируемой логики для решения вычислительных задач / С.Я. Гильгурт // Труды II международной конф. «Параллельные вычисления и задачи управления» РАСО’2004 памяти Е.Г. Сухова. Москва, 4-6 окт. 2004 г. – М.: Институт проблем управления им. В.А. Трапезникова РАН, 2004. – С.514-530.

2. Гильгурт С.Я. К вопросу о применении реконфигурируемых вычислителей для решения ресурсоемких задач / С.Я. Гильгурт // Информационные технологии в управлении энергетическими системами (ИТУЭС-2005): тез. докл. Междунар. конф. (18-19 октября 2005 р., г. Киев), К.: Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, 2005. – С.72-74.

3. Гильгурт С.Я. Решение задач распознавания с применением реконфигурируемых вычислителей / С.Я. Гильгурт, А.Н. Давиденко // Искусственный интеллект. Интеллектуальные и многопроцессорные системы – 2006: Тез. докл. Международной научно-технической конференции. Т. 2. – Таганрог: Изд-во ТРТУ, 2006. – С.76-79.

4. Гильгурт С.Я. Применение типовых реконфигурируемых устройств на базе ПЛИС для ресурсоемких вычислений / С.Я. Гильгурт // Матеріали Міжнар. наук.-практ. конф. «Інтелектуальні системи прийняття рішень та інформаційні технології», м. Чернівці, 17-19 травня 2006 р. – Чернівці: "Рута", 2006. – С.278-279.

5. Гильгурт С.Я. О применении реконфигурируемых унифицированных вычислителей для решения научно-технических задач / С.Я. Гильгурт // Параллельные вычислительные технологии (ПаВТ'2008): Труды международной научной конференции (Санкт-Петербург, 28 января – 1 февраля 2008 г.). – Челябинск: Изд. ЮУрГУ, 2008. – С.358-363.

6. Гильгурт С.Я. Некоторые вопросы аппаратного ускорения в современных информационных технологиях / С.Я. Гильгурт // Материалы Международной научно-технической конференции «Многопроцессорные вычислительные и управляющие системы» (МВУС-2009), Дивноморское, 28 сентября – 03 октября 2009 г. Т. 1. – Таганрог: Изд-во ТТИ ЮФУ, 2009. – С.26-29.

7. Гильгурт С.Я. Анализ применения реконфигурируемых устройств в системах обнаружения вторжений / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2010». Т. 1. – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2010. – С.260-267.

8. Гильгурт С.Я. Аппаратное распознавание строк в интеллектуальных системах защиты информации / С.Я. Гильгурт // Тез. докл. международной научно-технической конференции «Искусственный интеллект. Интеллектуальные системы ИИ-2011». – Донецк: ИПИИ «Наука і освіта». – 2011. – Т. 1. – С.271-274.

9. Гільгурт С.Я. Програмно-апаратна система захисту даних для інфраструктури грід / С.Я. Гільгурт, А.К. Гіранова // Сучасні комп’ютерні системи та мережі: розробка та використання: матеріали 5-ої Міжнар. наук.-техн. конф. ACSN-2011, 29 вересня – 1 жовтня 2011, Львів, Україна. – Л.: Вид-во Нац. ун-ту «Львів. політехніка», 2011. – С.50-53.

10. Гільгурт С.Я. Множинне розпізнавання рядків у системах виявлення вторгнення на базі реконфігурованих обчислювачів / С.Я. Гільгурт // Сучасні комп’ютерні системи та мережі: розробка та використання: матеріали 5-ої Міжнар. наук.-техн. конф. ACSN-2011, 29 вересня – 1 жовтня

2011, Львів, Україна. – Л.: Вид-во Нац. ун-ту «Львів. політехніка», 2011. – С.54-56.

11. Гильгурт С.Я. Аппаратное ускорение задач информационной безопасности на базе ПЛИС с применением грид-вычислений / С.Я. Гильгурт // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тез. доп. VII Міжнар. наук.-прак. конф. (17-19 вересня 2014 р., м. Запоріжжя), Запоріжжя: ЗНТУ, 2014. – С.329-330.

12. Гильгурт С.Я. Задача множественного распознавания строк в интенсивном потоке данных и методы ее аппаратного ускорения / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2016». – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2016. – С.166-169.

13. Гильгурт С.Я. Применение реконфигурируемых вычислителей для аппаратного ускорения сигнатурных систем защиты информации / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2018». – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2018. – С.107-110.

14. Hilgurt S.Ya. A method to construct the signature-based secure tools on reconfigurable accelerators / S.Ya. Hilgurt // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тез. доп. IX Міжнар. наук.-прак. конф. (03-05 жовтня 2018 р., м. Запоріжжя) [Електронний ресурс] / Редкол.: Д.М. Піза, С.В. Морщавка. Електрон. дані. – Запоріжжя: ЗНТУ, 2018. – 1 електрон. опт. диск (DVD-ROM); 12 см. – Назва з тит. екрана, 2018. – С.200-201.

15. Гильгурт С.Я. Централизованный синтез в грид-среде реконфигурируемых средств защиты информации / С.Я. Гильгурт // Тез. доп. Міжнар. наук.-техн. конф. «Високопродуктивні обчислення» (HPC-UA 2018) – Київ, 2018. – С.40-47.

16. Гильгурт С.Я. Применение реконфигурируемых устройств на базе ПЛИС при построении сетевых систем обнаружения вторжений /

С.Я. Гильгурт // Перспективні напрями захисту інформації: матеріали IV всеукр. наук.-пр. конф. – м. Одеса, 02-06 вересня 2018 р.: збірник тез. – Одеса: ОНАЗ ім. О.С. Попова, 2018. – С.37-42.

17. Гільгурт С.Я. Підвищення ефективності реконфігуровних систем виявлення вторгнень / С.Я. Гільгурт // Безпека інформаційних технологій: матеріали IX Міжнар. наук.-техн. конф. ITSec-2019, 22-27 березня 2019, м. Шарм-ель-Шейх, Єгипет. – К.: НАУ, 2019. – С.10-11.

18. Давыденко А.Н. Применение грид-сети для синтеза промышленных систем защиты информации на базе ПЛИС / А.Н. Давыденко, С.Я. Гільгурт // «Цифровые технологии в промышленности»: материалы республиканской научно-практической конференции, г. Актау, Казахстан, 28 марта 2019 г. – Актау, КГУТИ им. Ш. Есенова, 2019. – С.15-20.

19. Hilgurt S. Method for constructing reconfigurable multi-pattern matching modules for information security systems / S Hilgurt // Захист інформації і безпека інформаційних систем: матеріали VII Міжнар. наук.-техн. конф, м. Львів, 30-31 травня 2019. – Львів: Видавництво Львівської політехніки, 2019. – С.134-135.

20. Гільгурт С.Я. Апаратне рішення задач кібербезпеки в електроенергетичній галузі / С.Я. Гільгурт // Кібербезпека енергетики: Збірка праць конференції, м. Одеса, 28 травня – 1 червня 2019. – К.: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, 2019. – С.11-14.

21. Давиденко А.М., Політучий О.О. Методи підвищення ефективності мережевих систем виявлення вторгнень на базі ПЛІС / А.М. Давиденко, С.Я. Гільгурт, О.О. Політучий // Матеріали V Всеукр. наук.-практич. конф. «Перспективні напрями захисту інформації 2019», смт Затока Одеської обл., 31 серпня – 5 вересня 2019. – Одеса: 2019. – С.52-54.

Наукові праці, які додатково відображають наукові результати дисертації

1. Заявка u 2019 09353; G06F17/27; Апаратно-програмний комплекс підтримки прийняття рішень при проведенні державних експертиз комплексних систем захисту інформації / Давиденко А.М., Гільгурт С.Я., Шабан М.Р.; Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова Національної академії наук України. – заявл. 16.08.2019 р. (Рішення про видачу деклараційного патенту на корисну модель № 25510/ЗУ/19 від 28.10.2019 р.)

Додаток Б. Документи, що підтверджують впровадження результатів дисертаційної роботи, та лист підтримки

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
ІМ. М. Є. ЖУКОВСЬКОГО «ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Вул. Чкалова,17, м. Харків, 61070, тел. (057) 315-10-56, факс (057) 315-11-31
E-mail: khai@khai.edu, код ЄДРПОУ 02066769

503-дов/120:19 № від 06.12.2019р.

На № _____ від _____

ДОВІДКА

про впровадження результатів дисертаційної роботи старшого наукового співробітника Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України Гільгурта Сергія Яковича на тему «Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж» на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти

Результати дисертаційної роботи Гільгурта С.Я. на тему «Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж» в частині досліджень щодо підвищення ефективності апаратних засобів захисту інформації на базі ПЛІС в комп'ютерних системах і мережах було використано в рамках проекту «Інтернет речей: нова навчальна програма для потреб промисловості та суспільства», шифр ALIOT, номер проекту: 573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP (термін виконання: 15 жовтня 2016 – 14 квітня 2019 р.). Проект спрямовано на розроблення навчальних курсів з Інтернету Речей та програмно-апаратних засобів і систем для магістрів та аспірантів вищих навчальних закладів та наукових установ.

Матеріали дисертації Гільгурта С.Я., зокрема, методи створення реконфігуровних сигнатурних засобів захисту інформації шляхом комбінування різних підходів до побудови схем розпізнавання, увійшли у магістерську (MSc) програму навчання «Основи Інтернету речей, Інтернету Всього та Вебу Речей», і висвітлено в книзі Інтернет речей для індустріальних і гуманітарних застосунків. У трьох томах. Том 1. Основи і технології / За ред. В. С. Харченка. – Міністерство освіти і науки України, Національний аерокосмічний університет ХАІ, 2019. – 605 с.

Національний координатор проекту TEMPUS ALIOT
завідувач кафедри комп'ютерних систем мереж і кібербезпеки
Національного аерокосмічного університету
ім. М.Є. Жуковського «ХАІ»,
заслужений винахідник України
доктор технічних наук, професор В.С. Харченко

Менеджер проекту TEMPUS-ALIOT
доцент кафедри комп'ютерних систем, мереж і кібербезпеки
кандидат технічних наук О.О. Ілляшенко

06 грудня 2019 року

Підписи професора Харченка Вячеслава Сергійовича та Ілляшенка Олега Олександровича засвідчую.
Декан факультету радіоелектроніки,
комп'ютерних систем та інфокомунікацій ХАІ
кандидат технічних наук О.В. Одокієнко

Національний аерокосмічний університет «Харківський авіаційний інститут» Факультет №5 №02066769

ГЕРАКС-К

Україна, 01030, м. Київ, вул. Михайла Коцюбинського, будинок 4Б, офіс 17/1
код ЄДРПОУ 40405813, тел./факс: +380 (44) 461-90-48; 467-72-07; 483-11-70
e-mail: info@geraks.com; ppgeraks@gmail.com; http://geraks.com

А К Т

про використання результатів дисертаційної роботи «Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж» Гільгурта Сергія Яковича, яка подається на здобуття наукового ступеня доктора технічних наук

Даний акт складено про те, що результати дисертаційної роботи ст. наук. співр. відділу № 6 Математичного та економетричного моделювання Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України Гільгурта Сергія Яковича на тему «Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж», яка подається на здобуття наукового ступеня доктора технічних наук, впроваджено у виробничий процес ТОВ «ГЕРАКС-К».

Методи централізованого створення та супроводу засобів захисту інформації, розвинені Гільгуртом С.Я в дисертаційному дослідженні, були використані в процесі експлуатації інформаційної мережі підприємства та при виконанні комерційного проеку по створенню Системи автоматики магістрального нафтопродуктопроводу "Полоцьк-Фаніполь".

Впровадження результатів наукової роботи Гільгурта С.Я. у виробничу діяльність ТОВ «ГЕРАКС-К» дозволило знизити сукупну вартість володіння та підвищити рівень захисту інформації комп'ютерної системи підприємства та створюваної автоматизованої системи.

Директор ТОВ «ГЕРАКС-К»
канд. техн. наук М.П. Рижук
09.10.19

Україна * Товариство з обмеженою відповідальністю * м. Київ
"ГЕРАКС-К"
Ідентифікаційний код 40405813

ЗАТВЕРДЖУЮ:
Проректор з наукової роботи
Національного авіаційного
університету
д-р техн. наук, професор
_____ В.П. Харченко
"___" _____11_____ 2019 р.

АКТ

впровадження результатів дисертаційної роботи Гільгурта Сергія Яковича на тему «Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж» на здобуття наукового ступеня доктора технічних наук у навчальний процес кафедри комп'ютерних систем та мереж Національного авіаційного університету

Комісія у складі: голова – заступник завідувача кафедри комп'ютерних систем та мереж (КСМ) кандидат технічних наук, доцент Проценко М.М. та членів комісії - доцент кафедри КСМ Гузій М.М. та професор кафедри КСМ Антонов В.К. склали даний акт про те, що результати дисертаційної роботи Гільгурта Сергія Яковича «Методи та засоби створення реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж» впроваджені у навчальний процес та використовуються на кафедрі КСМ з 2017-2018 навчального року при викладанні навчальних дисциплін "Реконфігуровані комп'ютерні системи" та "Комп'ютерна схемотехніка".

№ з/п	Назва роботи, що впроваджується	Форма впровадження	Ефективність, від впровадження
	1	2	3
1.	Методика побудови сигнатурних засобів захисту інформації на базі ПЛІС	Розділи навчальних програм дисциплін, лекційні та лабораторні завдання	Ознайомлення студентів з принципами побудови, класифікацією та основними показниками продуктивності пристроїв прискорення обчислень на базі ПЛІС
2.	Методика підвищення ефективності реконфігуровних сигнатурних засобів захисту інформації комп'ютерних систем і мереж	Розділи навчальних програм дисциплін, лекційні та лабораторні завдання	Ознайомлення студентів з підходами до створення реконфігуровних апаратних засобів прискореного вирішення ресурсномісткої задачі множинного розпізнавання рядків у сигнатурних системах захисту інформації

Голова комісії:
заступник завідувача кафедри комп'ютерних систем та мереж
кандидат технічних наук, доцент М.М. Проценко

Члени комісії:
доцент кафедри комп'ютерних систем та мереж
кандидат технічних наук, доцент М.М. Гузій

професор кафедри комп'ютерних систем та мереж
доктор технічних наук В.К. Антонов

ЗАТВЕРДЖУЮ
Декан факультету радіофізики,
електроніки та комп'ютерних
систем Київського національного
університету ім. Тараса Шевченка

_____ І.О. Анісімов

_____ 11 _____ 2019 р.

А К Т

про впровадження результатів дисертаційних досліджень співробітників тематичної групи «Моделювання систем захисту інформації в енергетиці» відділу № 6 Математичного та економетричного моделювання Інституту проблем моделювання ім. Г.Є. Пухова НАН України Давиденка Анатолія Миколайовича, Гільгурта Сергія Яковича та Шабана Максима Радуйовича

Даний акт складено про те, що результати дисертаційних робіт співробітників тематичної групи «Моделювання систем захисту інформації в енергетиці» відділу № 6 Математичного та економетричного моделювання Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України Давиденка Анатолія Миколайовича, Гільгурта Сергія Яковича та Шабана Максима Радуйовича впроваджено у навчальний процес Кафедри радіотехніки та радіоелектронних систем Факультету радіофізики, електроніки та комп'ютерних систем Київського національного університету ім. Тараса Шевченка.

Авторами були розроблені лекції для навчальної дісціпліни «Випробування та експертиза засобів захисту інформації», які використовують науково-практичні результати, що отримані Давиденко А.М., Гільгуртом С.Я. та Шабаном М.Р., а саме: методи опису та аналізу критичної інформації на основі використання штучних нейроних мереж, моделі параметрів функціонального профілю захисту в комп'ютерних системах, принципи побудови систем підтримки прийняття рішень орієнтованих на інформаційне забезпечення процедур аналізу кібербезпеки, а також методи створення та аналізу реконіфгуровних засобів захисту інформації на базі ПЛІС.

Завідувач кафедри радіотехніки
та радіоелектронних систем
Київського національного університету
імені Тараса Шевченка
к. т. н., доцент М.І. Рєзніков

LUNDS UNIVERSITET

Pukhov Institute for Modelling
in Energy Engineering
National Academy of Sciences of Ukraine
15, General Naumov Str.
03164, Kyiv, Ukraine

Recommendation letter

The series of scientific research works devoted to the grid and cloud services development authored by Anatolii Davydenko (Pukhov Institute for Modeling in Energy Engineering, Kyiv, Ukraine; davidenkoan@gmail.com), Serhii Hilhurt (Pukhov Institute for Modeling in Energy Engineering, Kyiv, Ukraine; hilgurt@ipme.kiev.ua) and Andrii Salnikov (National Taras Shevchenko University of Kyiv, Ukraine; manf@grid.org.ua) is well known by the NorduGrid international community as it was presented on the series of annual NorduGrid Conferences.

In particular, the report "Web-service for FPGA synthesis using ARC-powered grid infrastructure" (https://indico.lucas.lu.se/event/573/session/8/contribution/9) took a worthy place among proceedings of the annual NorduGrid Conference "From Data Factories to Insight as a Service" that took place in Tromsø, Norway on 28-30 June 2017.

In general, the work of the team is of undoubted interest for the e-Science community as a basis for building services of such type. I had discussed the results of this work with the authors and recommend to support further development in this area in Ukraine.

Dr. Balázs Kónya 5/3/2019 LUND

Technical Coordinator
NorduGrid Collaboration

www.nordugrid.org

Lund University
Department of Physics
BOX 118, S - 221 00 LUND,
Sweden

balazs.konya@hep.lu.se
phone: +46 46 222 8049
fax: +46 46 222 4015

Додаток В. Експериментальне програмне забезпечення

Експериментальний програмний комплекс FPGA_Pattern_Matching призначений для проведення обчислювальних експериментів щодо дослідження методів комбінування підходів до побудови модулів розпізнавання реконфігуровних апаратних сигнатурних засобів технічного захисту інформації. Комплекс створено в середовищі програмування Embarcadero Delphi 10.3 Version 26.0.34749.6593 Community Edition. Нижче наведено окремі фрагменти коду, в яких реалізовано найважливіші програмні компоненти.

```
unit FPGA_Pattern_Matching;

{ Ver.2.4.2 - 2019-11-21 }

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  System.UITypes, Menus, StdCtrls, Grids, Math,
  AhoCorasick, Vcl.ComCtrls, VclTee.TeeGDIPlus, VCLTee.TeEngine, VCLTee.Series,
  Vcl.ExtCtrls, VCLTee.TeeProcs, VCLTee.Chart, Vcl.Mask;

const MmaxLimit = 520;  //// Максимальная возможная максимальная длина паттерна (370)
      Razrezh = 20;     //// Число отсчетов по X на графике // 20  // 50 // 1000
      MaxRazr = 1000;   //// Макс. число отсчетов по X на графике // 1000
      ShowListLim = 100;//// Лимит патернов, чтобы выводить для отладки // 20
      MaxZz = 32;       //// Макс. значение задержки ЦСЗ на базе LUT // 32

      Ftrans =   1;     //// Число байтов памяти на один переход А-К // 2
      FnRam  = 270;     //// Коэф. эффективности алгоритма AC(B)RAM  // 270
      Alkp   =  25;     //// Коэф. Алкп - для устройства управления А-К // 25
      Blkp   =   2;     //// Коэф. Влкп - для устройства управления А-К // 2
      Afkp   =  16;     //// Коэф. Аткп - для устройства управления А-К // 16
      Bfkp   =   3;     //// Коэф. Вткп - для устройства управления А-К // 3
      Alkoz  = 782;     //// Коэф. Алкозп - для контроллера памяти // 782
      Blkoz  =   8;     //// Коэф. Влкозп - для контроллера памяти // 8
      Afkoz  = 842;     //// Коэф. Аткозп - для контроллера памяти // 842
      Bfkoz  =  11;     //// Коэф. Вткозп - для контроллера памяти // 11

type
  TFrameForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    New1: TMenuItem;
    Open1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    Window1: TMenuItem;
    Tile1: TMenuItem;
    Cascade1: TMenuItem;
    Arrangeicons1: TMenuItem;
    Convert1: TMenuItem;
    Devide1: TMenuItem;
```

```
    DFAAC1: TMenuItem;
    Test11: TMenuItem;
    TestBR2: TMenuItem;
    BR1BR2: TMenuItem;
    TestHRCmp1: TMenuItem;
    CmprAll1: TMenuItem;
    Compare51: TMenuItem;
    OpenFileDialog: TOpenDialog;
    Label1,  Label2,  Label3,  Label4,  Label5,  Label6,  Label7,  Label8,
    Label9,  Label10, Label11, Label12, Label13, Label14, Label15, Label16,
    Label17, Label18, Label19, Label20, Label21, Label22, Label23, Label24,
    Label25, Label26, Label27, Label28, Label29, Label30, Label31, Label32,
    Label33, Label34, Label35, Label36, Label37, Label38, Label39, Label40: TLabel;
    Memo1: TMemo;
    StatusBar1: TStatusBar;
    Chart1: TChart;
    Series1,  Series2,  Series3,  Series4,  Series5,  Series6,  Series7,  Series8:
TLineSeries;
    Edit1, Edit2, Edit3, Edit4, Edit5, Edit6, Edit7, Edit8, Edit9, Edit10,
    Edit11, Edit12, Edit13, Edit14, Edit15: TEdit;
    RadioGroup1: TRadioGroup;
    RadioGroup2: TRadioGroup;
    RadioGroup3: TRadioGroup;
    RadioGroup4: TRadioGroup;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;

    function GetBR1Title: string;
    function GetBR2Title: string;
    function CheckParameters: boolean;
    function IntTo0000Str(Inp: integer): string;
    function IntTo00000Str(Inp: integer): string;
    function LAMBDA(x: integer): integer;
    function PI(x: integer): integer;
    function UpFrac(x, y: integer): integer;
    function NotZ(x: integer): integer;
    function UpPw2(x: real): integer;

    procedure ChartInit(NumberOfCurvs: integer; S1, S2, S3, S4, S5, S6, S7, S8:
string);
    procedure ChartDraw(NumberOfCurvs, TheLastPoint, X1, X2, X3, V1, V2, V3, V4, V5,
V6, V7, V8: integer);
    procedure FormInitiation(Sender: TObject);

    procedure ShowParameters;
    procedure ReadParameters;
    procedure GetParameters;

    procedure CountMiMaDelOm(Sigma: integer; var List: TStringList; var Mmin, Mmax,
OMEGA: integer);
    procedure CountMju(Mmax, Sigma: integer; var List: TStringList);
    procedure CountMjuZ(DelMax, Sigma: integer; var List: TStringList);
    procedure SortPatSet(Sigma: integer; var InpList: TStringList; var OutList:
TStringList; var DeltaMax: integer);

    procedure BsCAM(Mmin, Mmax, Sigma, OMEGA: integer; var Rbscam: integer);
    procedure DCAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var Rdcam:
integer);
    procedure DpCAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var Rdpcam:
integer);
    procedure LBF(Mmin, Mmax: integer; Beta: real; var Rbf: integer);
    procedure SBF(Mmin, Mmax: integer; Beta: real; var Rbf: integer);
    procedure ACRAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var Racram:
integer);
    procedure ACBRAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var Racbram:
integer);
```

```
  procedure HROMCmp(Mmin, Mmax: integer; var Rmin: integer; var Lmin: integer);
  procedure OptimHROMC(L, Mmin, Mmax: integer; var Rhrcmp, Rggf, Rrom, Rrg,
              G, nn, mm, m2m, Bhrcmp, SIGrom, OMGrom,
              Lhrcmp, Lggf, Lprf, Lsuf, Ldc, Lmux, Lwr, Lcmp, Land,
              Fhrcmp, Frg, Fggf, Fprf, Fsuf, Fmux, Fwr, Fand: integer);

  procedure Exit1Click(Sender: TObject);
  procedure New1Click(Sender: TObject);
  procedure Tile1Click(Sender: TObject);
  procedure Cascade1Click(Sender: TObject);
  procedure Arrangeicons1Click(Sender: TObject);
  procedure Open1Click(Sender: TObject);
  procedure Convert1Click(Sender: TObject);
  procedure Devide1Click(Sender: TObject);
  procedure DFAAC1Click(Sender: TObject);
  procedure Test11Click(Sender: TObject);
  procedure BR1BR2Click(Sender: TObject);
  procedure TestBR2Click(Sender: TObject);
  procedure TestHRCmp1Click(Sender: TObject);
  procedure CmprAll1Click(Sender: TObject);
  procedure Compare51Click(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
end;

IntParRec = record
  ParNam: string;  //// Название параметра
  UpLim : integer; //// Верхний предел
  IntPar: integer; //// Значение параметра
  LoLim : integer; //// Нижний предел
end;
ReaParRec = record
  ParNam: string;  //// Название параметра
  UpLim : real;    //// Верхний предел
  ReaPar: real;    //// Значение параметра
  LoLim : real;    //// Нижний предел
end;

TpIntArr = array [1..9] of IntParRec;
TpReaArr = array [1..6] of ReaParRec;

TpStrArr = array [0..6] of string;

var
FrameForm: TFrameForm;

Xx, Yy, Zz, Pr, Br, Lfpga, Bfpga, Er, ParBR2: integer;
Tfpga, Mra, Alfa, Beta, Gama, Teta: real;

Delta: array [1..MmaxLimit] of integer;
Mju: array [32..127, 1..MmaxLimit] of integer;
MjuZ: array [32..127, 1..MaxZz] of integer;
Marks: array [1..MaxRazr+1] of integer;

IntArr: TpIntArr = (
  (ParNam:'числа входів LUT'; UpLim:8; IntPar:6; LoLim:4),
  (ParNam:'здатності навантаження'; UpLim:50; IntPar:10; LoLim:5),
  (ParNam:'значення затримки ЦСЗ на LUT'; UpLim:64; IntPar:32; LoLim:16),
  (ParNam:'кількості портів пам`яті BRAM'; UpLim:8; IntPar:2; LoLim:1),
  (ParNam:'розміру блоків BRAM'; UpLim:64; IntPar:32; LoLim:2),
  (ParNam:'кількості блоків BRAM в ПЛІС'; UpLim:3000; IntPar:1470; LoLim:100),
  (ParNam:'кількості LUT в ПЛІС'; UpLim:2000000; IntPar:433200; LoLim:20000),
  (ParNam:'фактору хибного розпізнавання'; UpLim:20; IntPar:8; LoLim:3),
  (ParNam:'коефіцієнту розпаралелювання БР2'; UpLim:20; IntPar:1; LoLim:1)
);
ReaArr: TpReaArr = (
  (ParNam:'мінімальної тактової частоти'; UpLim:1000.0; ReaPar:2.0; LoLim:0.5),
```

```
    (ParNam:'об`єму    бортової    пам`яті    РУО';    UpLim:512000.0;    ReaPar:8000.0;
LoLim:128.0),
    (ParNam:'коеф. приведення тригерів до LUT'; UpLim:2.0; ReaPar:0.5; LoLim:0.25),
    (ParNam:'коеф.  приведення  блокової  пам`яті  до  LUT';  UpLim:1000.0;  ReaPar:295.0;
LoLim:50.0),
    (ParNam:'коеф.  приведення  зовнішньої  пам`яті  до  LUT';  UpLim:500.0;  ReaPar:54.2;
LoLim:0.1),
    (ParNam:'коеф. надмірності заповнення ПЗП'; UpLim:5.0; ReaPar:1.7; LoLim:1.0)
  );

implementation

uses MDIEdit, ChartUnit;

{$R *.dfm}


function TFrameForm.GetBR1Title: string;
begin
  case RadioGroup1.ItemIndex of
    0: GetBR1Title:= 'BsCAM';
    1: GetBR1Title:= 'DCAM';
    2: GetBR1Title:= 'DpCAM';
    3: GetBR1Title:= 'LBF';
    4: GetBR1Title:= 'SBF';
    5: GetBR1Title:= 'ACRAM';
    6: GetBR1Title:= 'ACBRAM';
    7: GetBR1Title:= 'HRCmp';
  end;
end;

function TFrameForm.GetBR2Title: string;
begin
  case RadioGroup2.ItemIndex of
    0: GetBR2Title:= 'BsCAM';
    1: GetBR2Title:= 'DCAM';
    2: GetBR2Title:= 'DpCAM';
    3: GetBR2Title:= 'LBF';
    4: GetBR2Title:= 'SBF';
    5: GetBR2Title:= 'ACRAM';
    6: GetBR2Title:= 'ACBRAM';
    7: GetBR2Title:= 'HRCmp';
  end;
end;

function TFrameForm.CheckParameters: boolean;
var i, TmpI: integer; TmpR: real;
begin
  ReadParameters;
  CheckParameters:= true;
  for i:= 1 to 9 do begin
    TmpI:= IntArr[i].IntPar;
    if TmpI < IntArr[i].LoLim then begin
      ShowMessage('Значення ' + IntArr[i].ParNam +
        ' не повинно бути менш за ' + IntToStr(IntArr[i].LoLim) + '!');
      CheckParameters:= false;
    end;
    if TmpI > IntArr[i].UpLim then begin
      ShowMessage('Значення ' + IntArr[i].ParNam +
        ' не повинно бути більш за ' + IntToStr(IntArr[i].UpLim) + '!');
      CheckParameters:= false;
    end;
  end;
  for i:= 1 to 6 do begin
    TmpR:= ReaArr[i].ReaPar;
    if TmpR < ReaArr[i].LoLim then begin
      ShowMessage('Значення  '  +  ReaArr[i].ParNam  +  '  не  повинно  бути  менш  за  '  +
FloatToStr(ReaArr[i].LoLim) + '!');
      CheckParameters:= false;
    end;
    if TmpR > ReaArr[i].UpLim then begin
```

```
      ShowMessage('Значення ' + ReaArr[i].ParNam + ' не повинно бути більш за ' +
FloatToStr(ReaArr[i].UpLim) + '!');
      CheckParameters:= false;
    end;
  end;
end;

function TFrameForm.LAMBDA(x:integer):integer;
begin
  if x < 8 then result:=2 else result:=1;
end;

function TFrameForm.PI(x:integer):integer;
begin
  if x > 4 then result:=3 else result:=5;
end;

function TFrameForm.UpFrac(x, y:integer):integer;
var Tmp: real;
begin
  Tmp:= x / y; result:= Ceil(Tmp);
end;

function TFrameForm.NotZ(x: integer): integer;
begin
  if x = 0 then result:= 0 else result:= 1;
end;

function TFrameForm.UpPw2(x: real): integer;
var Tmp: real;
begin
  result:= 1 shl Ceil(log2(real(x)));
end;


procedure TFrameForm.CountMiMaDelOm(Sigma: integer; var List: TStringList; var Mmin,
Mmax, OMEGA: integer);
var i, j, Cur, Prv: integer;
begin
  Mmin:=StrToInt(Copy(List[0], 1, 3));         ////  Вычисление Mmin
  Mmax:=StrToInt(Copy(List[Sigma - 1], 1, 3)); ////  Вычисление Mmax
  for j:= 1 to MmaxLimit do Delta[j]:= 0;      ////  Вычисление функции Delta[j]
  Prv:=Mmin; j:=Mmin; Delta[j]:= 1;
  if Sigma > 1 then for i:= 1 to Sigma - 1 do begin
    Cur:=StrToInt(Copy(List[i], 1, 3));
    if Cur = Prv then begin
      Delta[j]:=Delta[j]+1; Prv:=Cur;
    end else begin
      j:=j+1;
      while j < Cur do j:=j+1;
      Delta[j]:=1;
      Prv:=Cur;
    end;
  end;
  OMEGA:= 0;
  for j:= Mmin to Mmax do OMEGA:= OMEGA + Delta[j] * j;  ////  Вычисление OMEGA
end;

procedure TFrameForm.CountMju(Mmax, Sigma: integer; var List: TStringList);
var i, j, s, FstSymPos: integer;
begin
  for s:= 32 to 127 do for j:= 1 to Mmax do Mju[s, j]:= 0;
  for s:= 32 to 127 do begin         //// 0 to 255
    for i:= 0 to Sigma - 1 do begin  //// Цикл по всем паттернам
      FstSymPos:= length(List[i]) - 1;
      for j:= 1 to FstSymPos - 4 do begin   //// Цикл по всем символам текущего
паттерна
        if s = Byte(List[i][FstSymPos + 1 - j]) then Inc(Mju[s, j]);
      end;
    end;
  end;
```

```
end;

procedure TFrameForm.CountMjuZ(DelMax, Sigma: integer; var List: TStringList);
var i, j, s, FstSymPos: integer;
begin
  for s:= 32 to 127 do for j:= 1 to DelMax do MjuZ[s, j]:= 0;
  for s:= 32 to 127 do begin         //// 0 to 255
    for i:= 0 to Sigma - 1 do begin  //// Цикл по всем паттернам
      FstSymPos:= length(List[i]) - 1 {- 5};
      for j:= 1 to FstSymPos - 4 do begin   //// Цикл по всем символам текущего
паттерна
        if s = Byte(List[i][FstSymPos +1 -j]) then Inc(MjuZ[s, ((j-1) mod DelMax)+1]);
      end;
    end;
  end;
end;

procedure TFrameForm.SortPatSet(Sigma: integer; var InpList: TStringList; var OutList:
TStringList; var DeltaMax: integer);
var i, TmDel: integer;  StmJ: string;
begin
  if Sigma < ShowListLim then Memo1.Lines.Add('======================'); ///
  DeltaMax:= 0;
  for i:= 0 to Sigma - 1 do begin
    StmJ:= Copy(InpList[i], 1, 3);  //// StmJ = j - длина текущего патерна
    TmDel:= Delta[StrToInt(StmJ)];  //// TmDel = Delta[j] - размер текущего пакета
    if TmDel > DeltaMax then DeltaMax:= TmDel; //// DeltaMax - максимум Delta[j]

    if RadioGroup3.ItemIndex = 0 then begin //// Сортировка по длине паттерна -
Pattern_L
      if CheckBox1.Checked then begin //// с инверсией:
        OutList.Add(IntTo00000Str(i+1) + '-' + InpList[Sigma-1-i]);
      end else begin //// - без инверсии:
        OutList.Add('00' + Copy(InpList[i], 1, 3) + '-' + InpList[i]);
      end;
    end;

    if RadioGroup3.ItemIndex = 1 then begin //// Сортировка по размеру пакета -
Delta_J
      if CheckBox1.Checked then begin //// с инверсией:
        OutList.Add(IntTo00000Str(99999 - TmDel)+'-'+StmJ+Copy(InpList[i], 4,
Length(InpList[i])-1));
      end else begin //// - без инверсии:
        OutList.Add(IntTo00000Str(TmDel)+'-'+StmJ+Copy(InpList[i], 4,
Length(InpList[i])-1));
      end;
    end;

    if RadioGroup3.ItemIndex = 2 then begin //// По произведению пакета на длину
      if CheckBox1.Checked then begin //// с инверсией:
        OutList.Add(IntTo00000Str(99999 - TmDel*StrToInt(StmJ))+'-
'+StmJ+Copy(InpList[i], 4, Length(InpList[i])-1));
      end else begin //// - без инверсии:
        OutList.Add(IntTo00000Str(TmDel*StrToInt(StmJ))+'-'+StmJ+Copy(InpList[i], 4,
Length(InpList[i])-1));
      end;
    end;

  end;
  if Sigma < ShowListLim then begin
    Memo1.Lines.Add('OutList:'); ///
    for i:= 0 to OutList.Count-1 do Memo1.Lines.Add(OutList[i]); ///
  end;
end;


procedure TFrameForm.BsCAM(Mmin, Mmax, Sigma, OMEGA: integer; var Rbscam: integer);
var i, j, Tmpj, Tmp1, Lbscam, Fbscam, Lcmp, Land, Frg, Ffan, Fand: integer;
begin
  Land:= 0;
  for j:= Mmin to Mmax do Land:= Land + Delta[j] * UpFrac(j-1,Xx-1);  ////(4.12)
```

```
  Lcmp:= LAMBDA(Xx) * OMEGA;                                          ////(4.10)
  Lbscam:= Lcmp + Land;                                               ////(4.8)
  Fand:= Land - 1;                                                    ////(4.18)
  Tmp1:= 0;
  for j:= Mmin+1 to Mmax do begin
    Tmpj:= 0; for i:= j to Mmax do Tmpj:= Tmpj + Delta[i];
    Tmp1:= Tmp1 + UpFrac(Tmpj-1,Yy-1);
  end;
  Ffan:= Mmin * UpFrac(Sigma - 1,Yy-1) + Tmp1;                        ////(4.17)
  Frg:= 8 * Mmax;                                                     ////(4.15)
  Fbscam:= Frg + Ffan {+ Fand};    //// во втором теоретич. приближении - (4.14)
  Rbscam:= Lbscam + trunc(Alfa * Fbscam);
end;


procedure TFrameForm.DCAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var
Rdcam: integer);
var j, s, Tmp, Ldcam, Ldc, Ldel, Land, Fdcam, Fdel, Ffandc, Ffandel, Fand: integer;
begin
  CountMju(Mmax, Sigma, List);          //// Вычисление 1-й функции самоподобия
  Ldc:= 256 * LAMBDA(Xx);                                             ////(4.25)
  Ldel:= 0;
  for j:= 2 to Mmax do begin
    Tmp:= 0; for s:= 32 to 127 do Tmp:= Tmp + NotZ(Mju[s, j]);
    Ldel:= Ldel + UpFrac(j-1,Zz+1) * Tmp;                             ////(4.28)
  end;
  Land:= 0;
  for j:= Mmin to Mmax do Land:= Land + Delta[j] * UpFrac(j-1,Xx-1);  ////(4.12)
  Ldcam:= Ldc + Ldel + Land;                                          ////(4.24)
  Fdel:= Ldel;                                                        ////(4.31)
  Ffandc:= 0;
  for s:= 32 to 127 do begin
    Tmp:= 0; for j:= 2 to Mmax do Tmp:= Tmp + NotZ(Mju[s, j]);
    Ffandc:= Ffandc + UpFrac(Tmp-1,Yy-1);                             ////(4.32)
  end;
  Ffandel:= 0;
  for j:= 2 to Mmax do begin
    Tmp:= 0; for s:= 32 to 127 do Tmp:= Tmp + UpFrac(Mju[s, j]-1,Yy-1);
    Ffandel:= Ffandel + Tmp;                                          ////(4.33)
  end;
  Fand:=Land - 1;                                                     ////(4.18)
  Fdcam:={Fdel +} Ffandc + Ffandel {+ Fand};   //// во втором теор.прибл. (4.30)
  Rdcam:= Ldcam + trunc(Alfa * Fdcam);
end;


procedure TFrameForm.DpCAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var
Rdpcam: integer);
var j, s, Tmp, Ldpcam, Ldc, Ldel, Ldelpk, Landpk, Fdpcam, Fdel, Fdelpk, Ffandc,
Ffandel: integer;
begin
  CountMjuZ(Zz-1, Sigma, List);          //// Вычис-е 1-й част.функ.самоподобия
  Ldc:= 256 * LAMBDA(Xx);                                             ////(4.25)
  Ldel:= 0;
  for j:= 2 to Zz-1 do begin
    Tmp:= 0; for s:= 32 to 127 do Tmp:= Tmp + NotZ(MjuZ[s, j]);
    Ldel:= Ldel + UpFrac(j-1,Zz+1) * Tmp;                             ////(4.38)
  end;
  Landpk:= 0;
  for j:= Mmin to Mmax do Landpk:= Landpk + Delta[j] * (Floor(j/(Zz-1))*
    (UpFrac(Zz,Xx)+1)+UpFrac(j mod (Zz-1),Xx)+1);                     ////(4.40)
  Ldelpk:= 0;
  for j:= Mmin to Mmax do Ldelpk:= Ldelpk + Delta[j] * UpFrac(j,Zz-1);////(4.42)
  Ldpcam:= Ldc + Ldel + Landpk + Ldelpk;                              ////(4.37)
  Fdel:= Ldel;                                                        ////(4.44)
  Fdelpk:= Ldelpk;                                                    ////(4.45)
  Ffandc:= 0;
  for s:= 32 to 127 do begin
    Tmp:= 0; for j:= 2 to Zz - 1 do Tmp:= Tmp + NotZ(MjuZ[s, j]);
    Ffandc:= Ffandc + UpFrac(Tmp-1,Yy-1);                             ////(4.46)
  end;
```

```
  Ffandel:= 0;
  for j:= 2 to Zz - 1 do begin
    Tmp:= 0; for s:= 32 to 127 do Tmp:= Tmp + UpFrac(MjuZ[s, j]-1,Yy-1);
    Ffandel:= Ffandel + Tmp;                                          ////(4.47)
  end;
  Fdpcam:={Fdel + Fdelpk +} Ffandc+Ffandel;     //// во втором теор.прибл.(4.43)
  Rdpcam:= Ldpcam + trunc(Alfa * Fdpcam);                            /////(4.48)
end;


procedure TFrameForm.LBF(Mmin, Mmax: integer; Beta: real; var Rbf: integer);
var j, q, Rsbf, Fh3, G, LdopPBF, FdopPBF, LdopMBF, Ldop, Fdop, Lh3: integer;
begin
  Rbf:= 0;
  for j:= Mmin to Mmax do begin
    if Delta[j] <> 0 then begin
      G:= Ceil(log2(Er * Delta[j]/Ln(2)));                            ////(4.78)
      q:= UpFrac(Er,Pr);                                              ////(4.58)
      Lh3:= Er * G * (2*LAMBDA(Xx)+1);     ///(65) сокращ.схема!!! логики меньше
      Fh3:= Er * G * j;              ///(66) сокращ.схема!!! триггеров меньше
      LdopPBF:= UpFrac(G, Floor((Xx-1)/2)) + 4;                       ////(4.62)
      FdopPBF:= G;                                                    ////(4.63)
      LdopMBF:= UpFrac(q-1,Xx-1);                                     ////(4.65)
      Ldop:= q * LdopPBF + LdopMBF;                                   ////(4.75)
      Fdop:= q * FdopPBF;                                             ////(4.76)
      Rsbf:= Lh3 + Ldop + trunc(Alfa * (Fh3+Fdop)) + trunc(Beta * q); ////(4.77)
      Rbf:= Rbf + Rsbf;
    end;
  end;
end;


procedure TFrameForm.ACRAM(Mmin, Mmax, Sigma: integer; var List: TStringList; var
Racram: integer);
var i, j: integer;
  Ch: AnsiChar; Stmp: AnsiString;
  Alphabet: array[0..255] of boolean;
  FAhoCorasick: TAhoCorasick;
  InpString, OutString: AnsiString;
  Dir, Crs, Fal, Rst, Depth: cardinal;
  Ram, LogRam, Rkp, Rkoz, Rram, Lkp, Fkp, Lkoz, Fkoz: integer;
begin
  for i:=0 to 255 do Alphabet[i]:= false;
  Fal:= 0;
  FAhoCorasick:= TAhoCorasick.Create;
  for i:= 0 to Sigma - 1 do begin          //// Построение автомата Ахо-Корасик
    Stmp:=Copy(List[i], 5, (Length(List[i])-5));
    FAhoCorasick.AddKeyword(Stmp, Dir);    //// Подсчет прямых переходов
    for Ch in Stmp do begin
      if not Alphabet[Ord(Ch)] then begin
        Alphabet[Ord(Ch)]:= true;
        inc(Fal);                          //// Подсчет ошибочных переходов
      end;
    end;
  end;
  FAhoCorasick.SetCrossNodes(0, Rst, Crs); //// Рестартовые и перекрест. переходы
  Ram:= (Ftrans + Sigma) * (Dir+Crs+Fal+Rst);                         ////(4.81)
  LogRam:= trunc((Ceil(log2(Ram))+20));
  Lkp:= Alkp + Blkp * LogRam;                                         ////(4.85)
  Fkp:= Afkp + Bfkp * LogRam;                                         ////(4.86)
  Lkoz:= Alkoz + Blkoz * LogRam;                                      ////(4.87)
  Fkoz:= Afkoz + Bfkoz * LogRam;                                      ////(4.88)
  Rkp:= Lkp + trunc(Alfa * Fkp);                                      ////(4.82)
  Rkoz:= Lkoz + trunc(Alfa * Fkoz);                                   ////(4.83)
  Rram:= trunc(Gama * Ram);                                           //// (4.4)
  if CheckBox4.Checked then Rkoz:= 0;
  Racram:= Rkp + Rkoz + Rram;                                         ////(4.89)
  FAhoCorasick.Destroy;
end;
```

```
procedure TFrameForm.HROMCmp(Mmin, Mmax: integer; var Rmin: integer; var Lmin:
integer);
var j, L, Rhrcmp, Lhrcmp, Lggf, Lprf, Lsuf, Ldc, Lmux, Lwr, Lcmp, Land,
  Fhrcmp, Frg, Fggf, Fprf, Fsuf, Fmux, Fwr, Fand, Bhrcmp,
  G, SIGrom, OMGrom, nn, mm, m2m, Rrg, Rggf, Rrom: integer;
begin
  Rmin:= 2147483646;
  for L:= 0 to (Mmax - 1) do begin
    OptimHROMC(L, Mmin, Mmax, Rhrcmp, Rggf, Rrom, Rrg, G, nn, mm, m2m, Bhrcmp,
    SIGrom, OMGrom, Lhrcmp, Lggf, Lprf, Lsuf, Ldc, Lmux, Lwr, Lcmp, Land,
    Fhrcmp, Frg, Fggf, Fprf, Fsuf, Fmux, Fwr, Fand);
    if Rhrcmp < Rmin then begin Rmin:= Rhrcmp; Lmin:= L; end;
  end;
end;


procedure TFrameForm.Convert1Click(Sender: TObject);
var PatList, OutList: TStringList;
  i, j, Tmp, PatPos: integer;
  S, Stmp: string;
begin
  PatList:= TStringList.Create; PatList.LoadFromFile(OpenFileDialog.FileName);
  OutList:= TStringList.Create; OutList.Sorted:=True; OutList.Duplicates:=dupIgnore;

  Label2.Caption:= 'Pattern set = ' + ExtractFileName(OpenFileDialog.FileName);
  Label4.Caption:=IntToStr(PatList.Count);
  Label6.Caption:= PatList[1];
  Label8.Caption:= PatList[2];
  Label10.Caption:= PatList[3];

  for i:= 0 to PatList.Count-1 do begin
    PatPos:= Pos('content:"', PatList[i]);
    while PatPos <> 0 do begin
      S:='';
      for j:=PatPos+9 to Length(PatList[i])-1 do begin
        Stmp:=Copy(PatList[i], j, 1);
        if Stmp = '"' then break;
        S:=S+Stmp;
      end;
      Label10.Caption:= S;
      Stmp:=PatList[i];
      Delete(Stmp, PatPos, 8);
      PatList[i]:=Stmp;
      Tmp:= length(S); Stmp:=IntToStr(tmp);
      if Tmp <100 then Stmp:= '0'+Stmp; if Tmp < 10 then Stmp:= '0'+Stmp;
      Memo1.Lines.Add(Stmp+'>'+S+'<'); ///
      OutList.Add(Stmp+'>'+S+'<');
      PatPos:= Pos('content:"', PatList[i]);
    end;
  end;
  //// Запись в файл
  OutList.SaveToFile(OpenFileDialog.FileName+'.ptr');
  OutList.Free;
  PatList.Free;
end;

procedure TFrameForm.Devide1Click(Sender: TObject);
var PatList, SortList, List_1, List_2: TStringList;
  Sigma, Ymax, Mmin, Mmax, Step, Rest, OMEGA, Num, Ltmp, Lmin, Popt, Jopt: integer;
  i, k, DeltaMax, ResBR1, ResBR2, s: integer;
  Sigma1, Sigma2: integer;
  MaxBR1, MaxBR2, MinTot: integer;
  OmgCur, Mcur: integer;
  PerCent1, PerCent2: real;
  ToCount, ToShow: boolean;
begin
  if not CheckParameters then exit;
  GetParameters;

  PatList:= TStringList.Create; PatList.LoadFromFile(OpenFileDialog.FileName);
```

```
  SortList:= TStringList.Create; SortList.Sorted:=True;
SortList.Duplicates:=dupAccept;
  Sigma:=PatList.Count;

  ChartInit(3, 'Разом', GetBR1Title, GetBR2Title, '', '', '', '', '');
  Chart1.BottomAxis.Maximum:= Sigma;
  Ymax:= 1;

  CountMiMaDelOm(Sigma, PatList, Mmin, Mmax, OMEGA);   //// Mmin, Mmax, Delta[j]
  SortPatSet(Sigma, PatList, SortList, DeltaMax);
  PatList.Free;

  MaxBR1:= 0; MaxBR2:= 0; MinTot:= 2147483646;

  if Sigma > Razrezh then Step:= trunc(Sigma / Razrezh) else Step:= 1;
  Rest:= Sigma mod Step;
  for k:= 0 to Razrezh+1  do Marks[k]:= trunc(k * OMEGA / Razrezh);
  if (RadioGroup4.ItemIndex = 1) then Step:= trunc(OMEGA / Razrezh);
  k:= 0; OmgCur:=0;

  Memo1.Lines.Add('----------------------'); ///
  Memo1.Lines.Add('Sigma = ' + IntToStr(Sigma) + '  OMEGA = ' + IntToStr(OMEGA) +
       '  Razrezh = ' + IntToStr(Razrezh) + '  Step = ' + IntToStr(Step) +
       '  Rest = ' + IntToStr(Rest)); ///

  for Num:=0 to Sigma do begin          //// Г Л А В Н Ы Й   Ц И К Л

    ToShow:= (Num = 0) or (Num = Sigma) or
      ((RadioGroup4.ItemIndex = 1) and (OmgCur > Marks[k])) or
      ((RadioGroup4.ItemIndex = 0) and (Num mod Step = Rest));
    ToCount:= ToShow or CheckBox2.Checked;

    //// Прореживание для ускорения:
    if ToCount then begin

      //// Р А З Б И В К А  на два списка - List_1 и List_2:
      List_1:= TStringList.Create; List_1.Sorted:=True; List_1.Duplicates:=dupAccept;
      List_2:= TStringList.Create; List_2.Sorted:=True; List_2.Duplicates:=dupAccept;
      for i:= 0 to Num - 1 do begin //// начало сортированого списка - в List_2
        List_2.Add(Copy(SortList[i], 7, Length(SortList[i])-1));
      end;
      Sigma2:=List_2.Count;
      for i:= Num to Sigma - 1 do begin //// остаток сортированого списка - в List_1
        List_1.Add(Copy(SortList[i], 7, Length(SortList[i])-1));
      end;
      Sigma1:=List_1.Count;

      //// Просчет BR1 ////
      if Sigma1 > 0 then begin
        CountMiMaDelOm(Sigma1, List_1, Mmin, Mmax, OMEGA);
        case RadioGroup1.ItemIndex of
          0: begin //// BsCAM
             BsCAM(Mmin, Mmax, Sigma1, OMEGA, ResBR1);
          end;
          1: begin //// DCAM
             DCAM(Mmin, Mmax, Sigma1, List_1, ResBR1);
          end;
          2: begin //// DpCAM
             DpCAM(Mmin, Mmax, Sigma1, List_1, ResBR1);
          end;
          3: begin //// LBF
             LBF(Mmin, Mmax, Beta, ResBR1);
          end;
          4: begin //// SBF
             SBF(Mmin, Mmax, Beta, ResBR1);
          end;
          5: begin //// ACRAM
            ACRAM(Mmin, Mmax, Sigma1, List_1, ResBR1);
          end;
          6: begin //// ACBRAM
            ACBRAM(Mmin, Mmax, Sigma1, List_1, ResBR1);
```

```
        end;
        7: begin //// HROMCmp
           HROMCmp(Mmin, Mmax, ResBR1, Ltmp);
        end;
      end;
    end else ResBR1:= 0;

    //// Просчет BR2 ////
    if Sigma2 > 0 then begin
      CountMiMaDelOm(Sigma2, List_2, Mmin, Mmax, OMEGA);
      case RadioGroup2.ItemIndex of   //// просчет BR2
        0: begin //// BsCAM
           BsCAM(Mmin, Mmax, Sigma2, OMEGA, ResBR2);
        end;
        1: begin //// DCAM
           DCAM(Mmin, Mmax, Sigma2, List_2, ResBR2);
        end;
        2: begin //// DpCAM
           DpCAM(Mmin, Mmax, Sigma2, List_2, ResBR2);
        end;
        3: begin //// LBF
           LBF(Mmin, Mmax, Beta, ResBR2);
        end;
        4: begin //// SBF
           SBF(Mmin, Mmax, Beta, ResBR2);
        end;
        5: begin //// ACRAM
          ACRAM(Mmin, Mmax, Sigma2, List_2, ResBR2);
        end;
        6: begin //// ACBRAM
          ACBRAM(Mmin, Mmax, Sigma2, List_2, ResBR2);
        end;
        7: begin //// HROMCmp
           HROMCmp(Mmin, Mmax, ResBR2, Ltmp);
        end;
      end;
      ResBR2:= ResBR2 * ParBR2; ////
    end else ResBR2:= 0;

    //// Подсчет крайних значений
    if ResBR1 > MaxBR1 then MaxBR1:= ResBR1;
    if ResBR2 > MaxBR2 then MaxBR2:= ResBR2;
    if (ResBR1 + ResBR2) < MinTot then MinTot:= (ResBR1 + ResBR2);
    if (ResBR1 + ResBR2) = MinTot then if ((RadioGroup1.ItemIndex = 7) or
      (RadioGroup2.ItemIndex = 7)) and (Sigma2 > 0) then begin
        Popt:= Sigma2;                  //// номер птерна опт. деления
        Jopt:= StrToInt(Copy(List_2[Popt-1], 1, 3)); //// длина патерна
      end;
    List_2.Free;
    List_1.Free;
  end;

  //// Прореживание для ускорения - вывод информации:
  if ToShow then begin
    if (RadioGroup4.ItemIndex = 1) and (Num <> Sigma) then Inc(k);

    //// Вывод на отладочный экран
    Memo1.Lines.Add('Num = ' + IntToStr(Num) + ///
      '  Sigma1 = ' + IntToStr(Sigma1) + '   Sigma2 = ' + IntToStr(Sigma2) +
      '     ResBR1 = ' + IntToStr(ResBR1) + '   ResBR2 = ' + IntToStr(ResBR2) +
      '   ResBR1 + ResBR2 = ' + IntToStr(ResBR1 + ResBR2)
    );
    //// Вывод на график
      if ResBR1 > Ymax then Ymax:= ResBR1;
      if ResBR2 > Ymax then Ymax:= ResBR2;
      if ResBR1 + ResBR2 > Ymax then Ymax:= ResBR1 + ResBR2;
      Chart1.LeftAxis.Maximum:= trunc(Ymax * 1.12); //// Запас по оси Y - 12%
      ChartDraw(3, Sigma, Num, Marks[k-1], Marks[k], ResBR1 + ResBR2, ResBR1,
ResBR2, 0, 0, 0, 0, 0);
  end;
```

```
    ///if (Num < Sigma) then Memo1.Lines.Add('OmgCur = ' + IntToStr(OmgCur) + '
Copy(SortList[Num], 7, 3) = ' + Copy(SortList[Num], 7, 3));
    ///Memo1.Lines.Add('');
    if (Num < Sigma) then OmgCur:= OmgCur + StrToInt(Copy(SortList[Num], 7, 3));
  end;
  Memo1.Lines.Add('----------------------'); Memo1.Lines.Add(''); ///
  Memo1.Lines.Add('MaxBR1 = '+IntToStr(MaxBR1) + '   MaxBR2 = '+ IntToStr(MaxBR2) +
    '   MinTot = '+ IntToStr(MinTot) ///
  );
  //// Вывод крайних значений
  Label30.Caption:= 'R(БР1)max = ' + IntToStr(MaxBR1);
  Label34.Caption:= 'R(БР2)max = ' + IntToStr(MaxBR2);
  if (MinTot < (MaxBR1)*0.97) and (MinTot < (MaxBR2)*0.97) then begin //// 0.97 = 3%
    Label32.Caption:= 'R(БР1+БР2)min = ' + IntToStr(MinTot);
    PerCent1:= (MaxBR1 - MinTot)*100/MaxBR1;
    PerCent2:= (MaxBR2 - MinTot)*100/MaxBR2;
    if MaxBR1 > MinTot then Label31.Caption:= '<- ' + IntToStr(Round(PerCent1)) + '% -
>' else Label31.Caption:= ' ';
    if MaxBR2 > MinTot then Label33.Caption:= '<- ' + IntToStr(Round(PerCent2)) + '% -
>' else Label33.Caption:= ' ';
    Memo1.Lines.Add('MinTot / BR1 = ' + FloatToStrF(PerCent1, ffGeneral, 5, 2) + '%' +
                    '    MinTot / BR2 = ' + FloatToStrF(PerCent2, ffGeneral, 5, 2) +
'%');
  end else begin
    Label31.Caption:= ''; Label32.Caption:= ''; Label33.Caption:= '';
    Label35.Caption:= 'Mmin = ' + IntToStr(Mmin); Label36.Caption:= 'Mmax = ' +
IntToStr(Mmax);
  end;
  //Label35.Caption:= ''; Label36.Caption:= '';
  if (RadioGroup3.ItemIndex <> 2) then //// если упорядочение - не Pat_x_Del
    if ((RadioGroup1.ItemIndex = 7) or (RadioGroup2.ItemIndex = 7)) then begin
      Label35.Caption:= 'Poptim=' + IntToStr(Popt); //// номер птерна опт. деления
      Label36.Caption:= 'Joptim=' + IntToStr(Jopt); //// длина птерна опт. деления
    end
  else begin
    Label35.Caption:= 'Mmin = ' + IntToStr(Mmin); Label36.Caption:= 'Mmax = ' +
IntToStr(Mmax);
  end;
  Label40.Caption:= 'Sig = ' + IntToStr(Sigma);
  SortList.Free;
end;


end.
```