МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна наукова
праця на правах рукопису

Мінзюк Вадим Володимирович

УДК 62-507

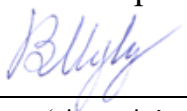
ДИСЕРТАЦІЯ

Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв

05.12.13 – Радіотехнічні пристрої та засоби телекомунікацій
(шифр і назва спеціальності)

05 «Технічні науки»
(галузь знань)

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


/Вадим МІНЗЮК/
(підпис, ім'я та прізвище здобувача)

Подається на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник –
Рицар Богдан Євгенович,
д.т.н., професор

Ідентичність усіх примірників дисертації
ЗАСВІДЧУЮ:
Учений секретар спеціалізованої вченої ради


/Микола БЕШЛЕЙ/

Львів – 2024

АНОТАЦІЯ

Мінзюк В. В. Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв. — Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.12.13 – «Радіотехнічні пристрої та засоби телекомунікацій» (172 – Телекомунікації та радіотехніка). — Національний університет «Львівська політехніка» МОН України, Львів, 2024.

У дисертації розв’язано актуальне науково-практичне завдання, яке полягає в удосконаленні та розробленні алгоритмів та методів мінімізації булових функцій для проектування цифрових комбінаційних схем радіотехнічних пристроїв та засобів телекомунікацій.

Запропоновано числове подання кон'юнктермів довільного рангу у вигляді пари чисел: двійкового зображення та двійкової маски літералів. Двійкове зображення отримуємо шляхом заміни символів поглинання «−» на нулі в псевдотрійковому зображенні. Щоб отримати маску літералів, спочатку замінюємо в трійковому зображенні нулі одиницями, а потім символи поглинання нулями. Оскільки склеювання можливі лише між кон'юнктермами з однаковою маскою літералів, об'єднуємо такі кон'юнктерми у множини. Тепер можна позначати кожен кон'юнктерм лише одним числом, а маску літералів вказати єдину для всієї множини. Це дає змогу використати в алгоритмах мінімізації булових функцій двійкові операції замість операцій над символами. Такий підхід полегшує комп'ютерну реалізацію методів мінімізації.

Розвинуто метод розчеплення кон'юнктермів. Запропоновано модифікацію процедури розчеплення кон'юнктермів на основі числового подання, що дозволяє зменшити витрати обчислювальних ресурсів, а використання додатково шістнадцяткової системи числення дає змогу розширити коло завдань, які можна розв’язати без застосування комп’ютера.

Розвинуто метод порозрядного вирощування простих кон'юнктермів. Використана розроблене числове подання замість псевдотрійкового, що дає змогу оперувати числами замість символів у підмножинах кон'юнктермів з

однаковою маскою літералів. Уведено поняття коду помітки множини кон'юнктермів, для усічення трійкового дерева вирощування простих кон'юнктермів. Уведено процедуру виявлення підмножини, елементи якої склеюються в один кон'юнктерм, шляхом підрахунку кількості елементів одразу на етапі сортування за заданим бітом. Розширено сферу застосування методу для недовизначених функцій. Запропоновані зміни дають змогу розширити коло розв'язуваних завдань за рахунок зменшення обчислювальних витрат та розширення сфери застосування методу.

Удосконалено метод мінімаксного покриття в теоретико-множинній формі. Запропоновано впорядковувати символьні диз’юнктерми щодо наростання їх потужностей, тобто починаючи з мінімальної. Це прискорює процедуру спрощення сформованої множини за рахунок скорочення шляху пошуку диз’юнктермів, що спрощуються.

Запропоновано процедуру ланцюгового покриття таблиці простих кон’юнктермів, яка враховує взаємне розташування кон’юнктермів у двійковому просторі, що дає змогу спростити циклічну частину таблиці простих кон’юнктермів. Це призводить до зменшення витрат машинних ресурсів. Для зниження обчислювальної складності завдання можна розрахувати коефіцієнт складності в околі розгалуження ланцюгів та прийняти рішення про розрив ланцюга у цьому місці. Процедура ланцюгового покриття множини простих кон’юнктермів дає змогу розбивати завдання покриття на кілька обчислювальних потоків. Запропонований підхід є евристичним.

Удосконалено метод побітового сортування множини цілих чисел, яка не містить тавтології. У процесі розбиття за заданим бітом з номером n виконують підрахунок елементів підмножини, що дає змогу виявити склеювання отриманої підмножини в кон'юнктерм і тоді замінити процедуру сортування по інших бітах простим перерахунком від 0 до $(2^n–1)$. Крім того, оскільки певні методи мінімізації потребують попереднього сортування множини вихідних кон’юнктермів, запропонована модифікація дає змогу отримати частину імплікантів вже на етапі сортування.

Уперше запропоновано метод мінімізації булових функцій побітовим розбиттям множини кон'юнктермів на основі розробленої модифікації побітового сортування зі склеюванням, що дає змогу знаходити прості кон'юнктерми низького рангу без проміжних склеювань. Цей метод позбавлений тавтології.

Для прикладної задачі виконано синтез перетворювача кодів запропонованими у роботі методами, серед яких метод мінімізації булових функцій побітовим розбиттям зі склеюванням показав найкращий результат.

Ключові слова: кон'юнктерм, мінімізація булових функцій, пошук простих кон'юнктермів, покриття таблиці простих кон'юнктермів, проектування цифрових комбінаційних схем.

ABSTRACT

Minziuk V. V. Development and research of optimal algorithms for minimizing Boolean functions in an arbitrary logical basis for the design of digital combinational devices. –Manuscript.

Dissertation for the degree of candidate of technical sciences on specialty 05.12.13 – "Radio engineering devices and means of telecommunications" (172 – Telecommunications and Radioengineering). — Lviv Polytechnic National University of Ministry of Education and Science of Ukraine, Lviv, 2024.

The dissertation solves an actual scientific and practical task, which is to impreove and to develop algorithms and methods of minimizing Boolean functions for designing digital combinational circuits of radio engineering devices and telecommunications equipment.

A numerical representation of conjuncterms of arbitrary rank is proposed in the form of a pair of numbers: a binary image and a binary mask of literals. The binary image is obtained by replacing the absorption symbols "–" with zeros in the pseudo-ternary image. To obtain a literal mask, we first replace zeros in the ternary image with ones, and then the absorption symbols "–" with zeros. Since gluing is possible only between conjuncterms with the same literal mask, we combine such conjuncterms into

sets. Now you can denote each conjunction with only one number, and specify the literal mask with a single number for the entire set. This will allow the use of binary operations in algorithms for minimizing Boolean functions instead of operations on symbols. This approach simplifies the computer implementation of minimization methods.

The splitting conjuncterms method has been developed. A modification of the procedure for splitting conjuncterms based on the proposed numerical representation, which makes it possible to reduce the amount of used computing resources, and the additional use of hexadecimal number system allows one to expand the range of problems that can be solved without the use of a computer.

The method of bitwise cultivation of primes has been developed. The developed numeric representation was used instead of the pseudo-ternary one, which made it possible to operate with numbers instead of symbols in subsets of conjuncterms with the same literal mask. The concept of a code for marking a set of conjuncterms has been introduced to truncate the ternary tree of growing primes. A procedure has been introduced for detecting a subset, the elements of which are glued together into one conjuncterm by simply counting the number of elements immediately at the stage of sorting by a given bit. The scope of application of the method for not fully defined functions has been expanded. The proposed changes make it possible to expand the range of problems to be solved by reducing computational costs and expanding the scope of the method.

The minimax coverage method in the set-theoretic form has been improved. It is proposed to arrange symbolic disjuncterms according to the increase in their powers, that is, starting from the minimum. This speeds up the procedure for simplifying the generated set by shortening the search path for simplifying disjuncterms.

The procedure for chain coverage of the table of primes is proposed. This procedure takes into account the relative position of conjuncterms in binary space, which makes it possible to simplify the cyclic part of the table of simple conjuncterms. This leads to a reduction in the amount of used computing resources. To reduce the computational complexity of the problem, you can calculate the complexity coefficient in the vicinity of the chain branch and make a decision about breaking the chain at this

place. The chain covering procedure for a set of simple conjuncterms makes it possible to split the covering problem into several computational threads. The proposed approach is heuristic.

The method of bitwise sorting of integers for a set of numbers that does not contain tautologies has been improved. In the process of partitioning by a given bit with number n, the elements of the subset are counted, which makes it possible to identify the gluing of the resulting subset into a conjuncterm and replace the sorting procedure by the remaining bits by simply calculating from 0 to 2n-1. In addition, since certain minimization methods require preliminary sorting of a set of initial conjuncterms, the proposed modification makes it possible to obtain part of the implicants already at the sorting stage.

The method of minimizing Boolean functions by bitwise partitioning of a set of conjuncterms has been developed. The method makes it possible to find low rank primes without intermediate gluings.

For the applied problem, the synthesis of the code converter was performed using the methods proposed in the paper, among which the method of minimizing Boolean functions by bitwise partitioning with gluing showed the best result.

Keywords: conjuncterm, minimization of Boolean functions, search for primes, coverage of the table of primes, design of digital combinational circuits.

Список публікацій здобувача:

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Рицар Б.Є., Мінзюк В.В. "Теоретико-множинна модифікація мінімаксного методу покриття булових функцій", *Управляющие системы и машины*, сентябрь-октябрь 2005, № 5, С. 43-47. (Scopus)

2. Рицар Б.Є., Мінзюк В.В. "Один спосіб зображення кон'юнктермів", Вісник Державного університету "Львівська політехніка" "*Радіоелектроніка і телекомунікації*". – Львів, 1999. – № 367. – С. 138-142.

3. Мінзюк В.В. "Спосіб синтезування кон'юнктермів булових функцій", Вісник Національного університету "Львівська політехніка" "*Радіоелектроніка та телекомунікації*". – Львів, 2004. – № 508. – С. 256-262.

4. Мінзюк В.В. "Спосіб спрощення задачі покриття булових функцій", Вісник Національного університету "Львівська політехніка" "*Радіоелектроніка та телекомунікації*". – Львів, 2005. – № 534. – С. 24-28.

5. Мінзюк В.В. "Спосіб сортування цілих чисел для задач мінімізації бульових функцій", Вісник Національного університету "Львівська політехніка" "*Радіоелектроніка та телекомунікації*". – Львів, 2011. – № 705. – С. 135-137.

6. Мінзюк В.В. "Модифікація методу порозрядного вирощування простих кон'юнктивних термів булових функцій", *Моделювання та інформаційні технології*, зб. наук. пр. ІПМЕ НАН України. – К., 2012. – Вип. 65. – С.129-134.

7. Мінзюк В.В. "Метод пошуку простих кон'юнктивних термів булових функцій побітовим розбиттям множини", *Моделювання та інформаційні технології*, зб. наук. пр. ІПМЕ НАН України. – К., 2013. – Вип. 66. – С.95-103.

8. Шклярський В. І., Матієшин Ю. М., Баланюк Ю. В., Мінзюк В. В. "Алгоритмічне забезпечення роботи телевізійного сканувального оптичного мікроскопа під час дослідження динамічних мікрооб'єктів", Вісник Національного університету "Львівська політехніка" "*Радіоелектроніка та телекомунікації*". – Львів, 2017. – № 885. – С. 15-21.

9. Мінзюк В. "Метод формування розгортки телевізійного сканувального оптичного мікроскопа", *Інфокомунікаційні технології та електронна інженерія*. – Львів, 2022. – Вип. 2, № 2. – С. 88-95.

10. Мінзюк В. "Метод мінімізації булевих функцій для проєктування цифрових комбінаційних схем", *Інфокомунікаційні технології та електронна інженерія*. – Львів, 2023. – Вип. 3, № 1. – С. 146-152.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

11. Rytsar, B., Minziuk, V. "The set-theoretical modification of Boolean functions minimax covering method", *Modern Problems of Radio Engineering, Telecommunications and Computer Science*. Proceedings of the International Conference TCSET'2004, 2004, pp. 46-48. (Scopus)

12. Minzyuk, V. "Modification of conjuncterms splitting method of boolean functions minimization", *Modern Problems of Radio Engineering,*

Telecommunications and Computer Science. Proceedings of International Conference TCSET'2006, 2006, pp. 81-82. (Scopus)

13. Minzyuk, V. "Integers sorting method for boolean functions minimization", *Modern Problems of Radio Engineering, Telecommunications and Computer Science.* Proceedings of the International Conference TCSET'2012, 2012, p. 61. (Scopus)

14. Matiieshyn, Y., Minziuk, V., Mankovskyy, S. "Algorithmic support of the television scanning optical microscope in the study of microobjects", *The Fourth International Conference on Information and Telecommunication Technologies and Radio Electronics*. Proceedings of the International Conference UkrMiCo'2019, 2019, 9165398, pp. 368-373. (Scopus)

15. Rytsar B. Ye., Minzyuk V. V., Sizonov Yu. V. "Minimization Method of Boolean Functions System", *Сучасні проблеми засобів телекомунікацій, комп'ютерної інженерії та підготовки спеціалістів*. Матеріали конференції TCSET'2000, 2000, Державний університет "Львівська політехніка", pp. 86-87.

16. Vadym Minzyuk "Technique of Boolean Functions Conjuncterm Synthesis", *Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці.* Матеріали VIII міжнародної науково-технічної конференції CADSM'2005. – Львів: Видавництво Національного університету "Львівська політехніка", 2005 – 578 с., ISBN 966-553-431-9.

17. Мінзюк В.В. "Модифікація методу порозрядного вирощування простих кон’юнктивних термів булових функцій", *Молодежь и современные проблемы радиотехники и телекоммуникаций РТ-2006* [Текст] : материалы междунар. молодеж. науч.-технич. конф. студ., аспирантов и ученых, 17-21 апреля 2006 г. / науч. ред. Ю. Б. Гимпилевич ; Севастопольский национальный технический ун-т. – Севастополь : Изд-во СевНТУ, 2006. – 309 с.

18. Пат. 149637 Україна. МПК H04B1/00 H04B1/04. Радіохвильовий сенсор / В. І. Оборжицький, В. Г. Сторож, Ю. М. Матієшин, В. Г. Протасевич, В. В. Мінзюк; заявник та власник патенту Національний університет “Львівська політехніка”. – № u202103668 ; заявл. 25.06.2021 ; опубл. 24.11.2021, Бюл. № 47.

ЗМІСТ

ВСТУП

Невпинний розвиток технологій сприяє появі нових цифрових великих інтегральних схем щораз вищого ступеня інтеграції, у тому числі програмованих логічних інтегральних схем (ПЛІС, англ. programmable logic device, PLD). Як їх розроблення так і застосування [1-20] у сфері проектування радіоелектронних цифрових комбінаційних пристроїв призводить до зростання розмірності розв'язуваних завдань. Через це одним з найскладніших стає етап функційно-логікового проектування, теоретичним підґрунтям якого є математичний апарат мінімізації булових функцій.

Ціль мінімізації — одержати такий аналітичний запис булової функції, що має найменший коефіцієнт складності. Коефіцієнт складності логікового виразу — це кількість термів плюс сума рангів усіх термів. При переході до схемотехнічного проектування кожен терм реалізується логіковим елементом, кількість входів якого дорівнює рангу терма. Таким чином чим менший аналітичний запис булової функції тим менше потрібно мікросхем та зв’язків між ними чи/і місця на кристалі для її апаратної реалізації.

Із середини XX століття почалося активне застосування булової алгебри для проектування пристроїв керування та комп’ютерів. Тоді ж з'явились перші наукові праці з мінімізації булових функцій. Зокрема, теоретичний фундамент заклали Блейк (Blake), Карно (Karnaugh), Квайн (Quine), МакКласкі (McCluskey), Шеннон (Shannon) та інші. Їх здобутки розвинули у своїх працях Куртіс (Curtis), Дітмаєр (Dietmeyer), Карп (Karp) Мореаль (Morreale), Рот (Roth), Кметь А. Б. та інші. Натепер точні методи мінімізації розробляють H. -G. Vu, N. -D. Bui, A. -T. Nguyen, ThanhBangLe, M. Garrido, M. Soeken, H. Riener, W. Haaswijk, A. Mishchenko, G. De Micheli та інші [21-24].

У подальшому мініатюризація електронних компонентів дала змогу будувати складніші цифрові пристрої. Це призвело до такого зростання

складності завдань мінімізації, що точні методи не могли дати результат за прийнятний час обчислень. Тому у 70-80-х роках XX століття з'явилася велика кількість евристичних методів мінімізації, зокрема у працях Брайтона (Brayton), МакМіллена (McMullen), Гонга (Hong), та інших. Натепер наближені методи мінімізації розробляють C. D. Murray, R. R. Williams, A. Bernasconi, V. Ciriani, G. Trucco, T. Villa, S. Cimato, M. C. Molteni, L. Amarú, A. Kagliwal, S. Balachandran, M. Nazemi, H. Kanakia and M. Pedram та інші [25-32].

На основі відомих методів мінімізації [33] розроблено програмне забезпечення для автоматизації проектування цифрових комбінаційних пристроїв. Застосування комп’ютерів розширило можливості проектування, але разом з тим спричинило появу складніших завдань, розв’язувати які старими методами неефективно, і навіть недоцільно удосконалювати їх. З’явилась потреба знайти нові методи розв’язання задач мінімізації. У такому ракурсі варто відзначити праці Брайтона (Brayton), МакҐіра (McGeer), Cacao (Sasao), Санджаві (Sanghavi), Санджованні-Вінчентеллі (Sangiovanni-Vincentelli), Рицара Б. Є., Соломко М. Т. та інших [34-89].

Актуальність теми. З появою мікросхем складність задач мінімізації булових функцій невпинно зростає разом із розвитком інтегральних технологій. З іншого боку зростають можливості обчислювальної техніки. Зокрема, прослідковується тенденція до збільшення кількості ядер та обчислювальних потоків при розробці нових процесорів. Для комп'ютерних обчислень важливі витрати обчислювальних ресурсів (обсяг пам'яті та машинний час, витрачений на обчислення). Можна зменшити витрати машинного часу за рахунок розпаралелювання обчислювальних процесів, але це переважно призводить до зростання витрат пам'яті. І навпаки, можливо зменшити витрати пам'яті за рахунок послідовного виконання обчислень в одному потоці. Комп'ютерна пам'ять та й взагалі апаратне забезпечення обчислень у порівнянні з програмним забезпеченням вже не є таким дороговартісним ресурсом як колись. Натепер вартість програмного забезпечення може в сто і більше разів перевищувати вартість апаратного забезпечення комп'ютера. Тому способи побудови

алгоритмів комп’ютерних обчислень розвиваються у бік швидкого розпаралелювання процесів. І навіть якщо закінчується вільна пам’ять, незалежні задачі можна поставити у чергу. Таким чином можна комбінувати розпаралелювання процесів і постановку у чергу залежно від наявності вільної пам’яті. Крім того переважна більшість методів мінімізації розроблена з позицій зручності сприйняття людиною у символьному поданні. У той час як процесор оперує числами і двійковими операціями. Це призводить до ускладнення алгоритмів комп’ютерної реалізації, що тягне за собою додаткові обчислювальні витрати. Усе це викликає потребу нових підходів до розв'язання задач мінімізації булових функцій. Тому не згасає інтерес вчених до цієї проблеми.

На даний час при проектуванні цифрових комбінаційних схем радіотехнічних пристроїв та засобів телекомунікацій актуальним **науково-практичним завданням** є удосконалення та розроблення алгоритмів та методів мінімізації булових функцій.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційну роботу виконано у рамках таких держбюджетних тем Національного університету “Львівська політехніка”: НДР ДБ/ВЕРБАЛЬ “Розробка макромоделей відмовостійких радіоелектронних засобів” (№ держреєстрації 0104U002291) Інституту телекомунікацій, радіоелектроніки та електронної техніки та НДР ДБ\Шум “Алгоритми екстракції даних методами ієрархічної кластеризації для важко вирішуваних комбінаторних задач великої розмірності” (№ держреєстрації 0108U000326) Інституту комп'ютерних наук та інформаційних технологій. У цих науково-дослідних роботах автор розробив числове зображення кон’юнктермів булових функцій, удосконалив метод мінімізації булових функцій розчепленням кон’юнктермів, удосконалив метод побітового вирощування простих кон’юнктермів булових функцій, розробив модифікацію методу побітового сортування цілих чисел для мінімізації логікових функцій, розробив метод мінімізації булових функцій побітовим розбиттям зі склеюванням.

Мета і завдання дослідження. *Метою* дисертаційного дослідження є удосконалення відомих та розроблення нових методів логікового синтезу комбінаційних схем, які дають змогу підвищити ефективність проектування цифрових пристроїв.

Для досягнення поставленої мети сформульовано такі основні *завдання*:

1) проаналізувати сучасні методи мінімізації булових функцій для виявлення таких, що можуть бути удосконалені;

2) удосконалити сучасний точний метод пошуку простих кон'юнктермів булової функції з метою розширення кола розв'язуваних задач;

3) удосконалити сучасний евристичний метод пошуку простих кон'юнктермів булової функції з метою розширення кола розв'язуваних задач;

4) удосконалити сучасний метод покриття таблиці простих кон'юнктермів булової функції;

5) на базі відомих методів сортування чисел розробити модифікацію, оптимізовану для задач мінімізації булових функцій;

6) розробити метод мінімізації булових функцій з меншою складністю реалізації порівняно з відомими методами, у тому числі за рахунок унеможливлення появи тавтології в процесі мінімізації.

Об'єкт дослідження — логіковий синтез цифрових комбінаційних пристроїв.

Предмет дослідження — методи мінімізації булових функцій логікового синтезу комбінаційних схем, що визначають ефективність процесу проектування цифрових радіотехнічних пристроїв та засобів телекомунікацій.

Методи дослідження. Для досягнення поставленої мети було використано: теорію цифрових автоматів, математичну логіку, булову алгебру, теорію множин і комбінаторику для побудови теоретичних основ запропонованих методів мінімізації булових функцій для різних форм задання;

Наукова новизна одержаних результатів:

1) уперше запропоновано метод мінімізації булових функцій побітовим розбиттям множини кон'юнктермів на основі розробленої модифікації

побітового сортування зі склеюванням, що на відміну від існуючих дає змогу виявляти прості кон’юнктерми низького рангу без проміжних склеювань простим підрахунком кількості кон’юнктермів. Для пошуку тупикових диз’юнктивних нормальних форм запропоновано процедуру ланцюгового покриття множини простих кон’юнктермів, що дає змогу розбивати задачу покриття на декілька обчислювальних потоків;

2) удосконалено метод порозрядного вирощування простих кон’юнктермів. Для цього використано розроблене маскове зображення замість псевдотрійкового, що дало змогу оперувати числовим поданням кон'юнктермів у підмножинах з однаковою маскою, введено поняття коду помітки множини кон’юнктермів для усічення трійкового дерева вирощування простих кон’юнктермів, розроблено процедуру виявлення підмножин, елементи яких склеюються в один кон’юнктерм простим підрахунком кількості елементів одразу на етапі сортування по заданому біту, розширено область застосування методу на недовизначені функції;

3) удосконалено метод побітового сортування множини цілих чисел, що позбавлена тавтології, шляхом підрахунку елементів підмножини у процесі сортування по заданому біту з номером n, що дає змогу виявити склеювання одержаної підмножини у кон'юнктерм і замінити процедуру сортування по решті бітів простим перерахунком від 0 до 2^n-1. Крім того, оскільки певні методи мінімізації потребують попереднього сортування множини вихідних кон'юнктермів, запропонована модифікація дає змогу одержати деякі імпліканти вже на етапі сортування;

4) набув подальшого розвитку метод мінімаксного покриття у теоретико-множинній формі, а саме запропоновано упорядковувати символьні диз’юнктерми за наростанням їх потужностей, тобто починаючи з мінімальної. Це пришвидшує процедуру спрощення сформованої множини за рахунок скорочення шляху пошуку спрощуваних диз’юнктермів.

Практичне значення отриманих результатів. Отримані у дисертаційній роботі наукові результати утворюють теоретичну базу для розроблення

алгоритмів і програмних засобів логікового синтезу цифрових комбінаційних схем та на їх основі реалізації процедур комп'ютерного проектування та розроблення цифрових радіотехнічних пристроїв і засобів телекомунікацій, у тому числі на базі ПЛІС. Практична цінність отриманих результатів така:

1) за допомогою розробленого у дисертаційній роботі методу мінімізації побітовим розбиттям множини кон'юнктермів розв'язано практичне завдання проектування цифрової комбінаційної схеми на базі програмованої логікової інтегральної схеми для керування мережею радіохвильових сенсорів. У Львівському центрі ІКД НАН та ДКА України було виготовлено прототип спроектованої цифрової комбінаційної схеми і виконано його експериментальне дослідження, яке підтвердило стовідсоткову точність реалізації заданих логікових функцій, що підтверджено відповідним актом. Розроблений у дисертаційній роботі удосконалений метод розчеплення кон'юнктермів для мінімізації булових функцій використано ТзОВ «Міта-Техніка» для проектування універсального сигнального перетворювача промислової автоматики на основі мережі первинних перетворювачів із оцифрованими вихідними сигналами. Порівняно з іншими методами мінімізації вдалося зменшити на 5% кількість комірок ПЛІС, які необхідні для реалізації заданих булових функцій, що дало змогу використати вивільнені ресурси для реалізації інших функцій, зокрема функції самоконтролю, що підтверджено відповідним актом (додаток А);

2) результати дисертаційної роботи використано для виконання держбюджетних науково-дослідних робіт ДБ/ВЕРБАЛЬ (держреєстр. № 0104U002291) та ДБ\Шум (держреєстр. № 0108U000326) НУ „Львівська політехніка", що підтверджено відповідним актом (додаток А);

3) удосконалено метод розчеплення кон'юнктермів на основі розробленого маскового зображення, що дає змогу зменшити витрати обчислювальних ресурсів, а використання додатково шістнадцяткової системи числення дає змогу розширити коло задач, які можна розв'язати без застосування комп'ютера;

4) результати дисертаційної роботи використовуються у навчальному процесі у Національному університеті „Львівська політехніка”, що підтверджено відповідним актом (додаток А).

Особистий внесок здобувача. Усі перелічені вище наукові результати дисертаційної роботи автор отримав самостійно. У співавторстві опубліковано праці [90] (у якій дисертантові належить процедура перетворення (кодування) довільно зображеного імпліканта), [91] (у якій дисертантові належить ідея покривати матриці розчеплення функцій системи однойменними векторами а не найдовшими), [92, 93] (у якій дисертантові належить ідея упорядковувати символьні диз’юнктерми за наростанням їх потужностей, тобто починаючи з мінімальної для пришвидшення процедури спрощення сформованої множини за рахунок скорочення шляху пошуку спрощуваних диз’юнктермів), [94, 95] (у якій дисертантові належить ідея здійснювати процес сканування шляхом послідовної зміни значень кодів лічильника положень світного елемента у рядку і лічильника рядків у кадрі, що дає змогу реалізувати цифрове керування за допомогою цифрової комбінаційної схеми перетворювача кодів на базі програмованої логікової інтегральної схеми для підвищення швидкодії), [96] (у якій дисертантові належить ідея цифрового керування радіохвильовим сенсором, що дає можливість застосувати цифрову комбінаційну схему на основі програмованої логікової інтегральної схеми для керування мережею давачів). Окрім того, автором здійснено 11 одноосібних публікацій [97-107].

Апробація результатів. Основні результати дисертаційної роботи апробовано на таких наукових конференціях і семінарах: Міжнар. конф. TCSET’2000, TCSET’2004, TCSET’2006 і TCSET’2012 „Сучасні проблеми радіоелектроніки, телекомунікацій, комп’ютерної інженерії” (Львів-Славське, 2000, 2004, 2006, 2012 рр.), 8-й Міжнар. наук.-техн. конф. „Досвід розробки та застосування приладо-технологічних САПР у мікроелектроніці” CADSM'2005 (Львів-Поляна, 2005), Міжнародна молодіжна науково-технічна конференція студентів, аспірантів та учених "Молодёжь и современные проблемы радиотехники и телекоммуникаций РТ-2006", Міжнародна конференція з

інформаційно-телекомунікаційних технологій та радіоелектроніки (УкрМіКо’2019/UkrMiCo’2019).

Публікації. Основні положення та результати дисертаційної роботи опубліковані у 18 наукових працях, а саме: 10 статей у наукових фахових виданнях згідно з переліком МОН України [90, 92, 94, 97-103] (з них 1 у SCOPUS [92], 7 написано одноосібно), 1 патент України на корисну модель [96], а також 7 праць у матеріалах міжнародних науково-технічних конференцій [91, 93, 95, 104-107] (з них 4 у SCOPUS, 4 написано одноосібно).

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. Повний обсяг роботи становить 201 сторінку, з яких 167 сторінок основного тексту. Робота містить 51 рисунок, 3 додатки. Список використаних джерел містить 112 найменувань.

1 ПЕРСПЕКТИВНІ МЕТОДИ МІНІМІЗАЦІЇ БУЛОВИХ ФУНКЦІЙ ДЛЯ ПРОЕКТУВАННЯ ЦИФРОВИХ КОМБІНАЦІЙНИХ ПРИСТРОЇВ

1.1 Порівняння довільних булових базисів за критерієм складності реалізації логікових функцій

Апаратні витрати (кількість логікових елементів та зв’язків між ними), що необхідні для побудови цифрових комбінаційних пристроїв, визначаються складністю булових функцій, які мають реалізовувати ці цифрові пристрої. У свою чергу складність булової функції — це фактично кількість кроків, необхідних для обчислення результату для заданого набору аргументів, при реалізації функції формулою (суперпозицією).

Базисом простору булових функцій будемо називати довільну скінченну повну систему булових функцій [108]. Прикладами базисів є множини функцій {І, ЧИ, НЕ}, {І, ЧИ, НЕ, сума за модулем 2} тощо. Задача порівняння булових базисів полягає у визначенні співвідношення складності реалізації булових функцій формулами у різних базисах. Складністю формули називають кількість літералів у ній, тобто кількість входжень у формулу символів змінних.

Одну й ту ж функцію можна подати формулами у різних базисах. Причому ці формули можуть бути різної складності. Потрібно знати, як може змінитися складність при переході з одного базису в інший. У [109] поставлена задача дослідження залежності реалізації булових функцій формулами від базису і доведено, що складність майже усіх булових функцій у класі формул асимптотично не залежить від базису, але відомо, що існує послідовність функцій, які реалізуються в одних базисах простіше ніж в інших. Наприклад, складність реалізації лінійних функцій від n змінних дорівнює n у базисі {І, ЧИ, НЕ, сума за модулем 2}, у той час як у базисі {І, ЧИ, НЕ} їх складність не менша за $n^{1,5}$. У зв'язку з цим постала задача порівняння базисів у загальному вигляді, а саме встановити, що деякий базис B не гірший за базис B`, якщо відношення

складності довільної функції у базисі *B* до складності тієї ж функції у базисі *B*` не більше деякої константи, яка залежить тільки від базисів *B* і *B*`.

У [108] сформульовано ознаку порівняння довільних базисів і на її основі показано, що базис {І, ЧИ, НЕ} є найгіршим (найбільшим), дано неалгоритмічний критерій порівняння довільних базисів, а також його застосовано до деяких пар базисів.

У подальших дослідженнях [110] доведено існування нескінченної множини попарно непорівнюваних базисів. Іншими словами структура базисів є нескінченно широкою. Але при цьому будь-який базис є порівнюваний із буловим базисом {І, ЧИ, НЕ}. Також виявлено, що структура базисів є нескінченно довгою, тобто існує нескінченний строго спадний ланцюжок базисів. А це у свою чергу свідчить про відсутність найкращого базису, тобто не існує мінімального базису.

1.2 Огляд сучасних методів мінімізації логікових функцій

Теоретичною базою проектування цифрових пристроїв є апарат математичної логіки, який винайшов Дж.Бул (G.Boole). З 30-х років XX-го століття Блейк (Blake A.), Шеннон К. (Shannon C. E.) почали використовувати алгебру Дж. Була для проектування цифрових пристроїв. У середині XX-го століття їх ідеї розвинули Куртис (Curtis H. A.), Карп (Karp R. M.), МакКласкі (McCluskey E. J.), Морель (Morreale E.), Мюлер (Muller D. E.), Квайн (Quine W. V.), Рот (Roth J. P.), та інші. З кінця 70-х років нові засади сформували Брайтон (Brayton R. K.), Кайн (Cain R.), Хачел (Hachtel G. D.), Гонґ (Hong S.), МакМюлен (McMullen C.), Остапко (Ostapko D.), Санджованні-Вінценеллі (Sangiovanni-Vincentelli A. L.), Кметь А. Б. та інші. Пізніше їх удосконалили Сасао (Sasao), Кудер (Coudert O.), Мадре (Madre J.) та ін. У XXI-му столітті над

цією проблемою працюють Сасао (Sasao T.), Кметь А. Б., Рицар Б. Є., Соломко М. Т., Тадеєв П. О. та інші [21-89].

Перші розроблені методи мінімізації булових функцій були точними, але через складність їх обчислювальних процедур вони мали обмеження практичного застосування. Вже при кількості змінних понад шість процес мінімізації різко ускладнювався. Згодом з'явились певні удосконалення, як наприклад у [21-24], що дали змогу дещо розширити коло розв'язуваних задач, але не суттєво. З появою програмованих логікових матриць зростає потреба у методах проєктування цифрових комбінаційних пристроїв із великою кількістю входів. На цей запит з'явились наближені (евристичні) методи мінімізації [29-31], які не гарантували знаходження глобального мінімуму, проте забезпечували прийнятний результат меншим обчислювальним коштом.

1.2.1 Аналіз проблеми мінімізації булових функцій

З виникненням практичної задачі проектування оптимальних цифрових пристроїв у середині минулого століття постала проблема мінімізації булових функцій, що спричинило появу праць Блейка (Blake A.), Карно (Karnaugh M.), Квайна (Quine W. V.), МакКласкі (McCluskey E. J.), Петріка (Petrick S. R.), Рота (Roth J. P.), Шеннона (Shannon C. E.). Для мінімізації вручну дотепер використовують класичні методи Квайна- МакКласкі та карт Карно. Результатом роботи цих методів є мінімальна диз'юнктивна нормальна форма (МДНФ) булової функції, безпосередньо з якої можна реалізувати функційну схему логікового пристрою. Зазначимо, що структура диз'юнктивної нормальної форми (ДНФ) є дворівневою. Використовуючи її як складову можна проектувати багаторівневі структури.

Класичний підхід до мінімізації булових функцій полягає у послідовному виконанні двох етапів, а саме обчислення скороченої ДНФ, а потім з неї –

мінімальної ДНФ. На першому етапі обчислюють множину всіх простих кон'юнктермів заданої функції від n змінних. Зауважимо, що потужність цієї множини може сягати дуже високих значень: $\frac{3^n}{n}$. Другий етап — це задача мінімального покриття, а значить є *NP*-повною. Тож навіть при невеликій кількості змінних на обох етапах потрібні значні обчислювальні ресурси. А з ростом n ми можемо взагалі зіткнутись із ситуацією, коли технічних можливостей забракне для розв'язання конкретної задачі. Через це з'явились наближені (евристичні) методи. Але при цьому від точних підходів не відмовились, оскільки активний розвиток обчислювальної техніки спонукав до удосконалення та розроблення нових точних методів. Зокрема, Рудел (Rudell R.) та Санджовані-Вінценелі (Sangiovanni-Vincentelli A. L.) на основі точного методу Квайна-МакКласкі розробили підхід [30]. Його програмною реалізацією є мінімайзер ESPRESSO-EXACT. Свій підхід до опрацювання множини простих кон'юнктермів запропонував Даґінес (реалізований у програмі MCBOOLE). Кудер (Coudert O.) та Мадр (Madre J.) розробили неявне подання простих кон'юнктермів, що ліг в основу алгоритму мінімізації [34]. Для мінімізації частинних функцій розроблено метод конкурентних інтервалів. Також відомо чимало підходів до розвитку методу Квайна-МакКласкі, зокрема. А з відмінних від класичних підходів, вирізняється підхід на основі багатозначної логіки [41]. Розроблена в [111] S_N - алгебра, лягла в основу інтегрованої струмової логіки, у якій і застосовується зазначений метод.

З появою великих інтегральних схем у семидесятих роках минулого століття починається активний розвиток евристичних підходів до мінімізації булових функцій, у тому числі на основі методу Квайна-МакКласкі. Першою з'явилась програма MINI від фірми IBM (автор підходу Гонґ (Hong S.)). У ній з'являлась менша кількість надлишкових простих кон'юнктермів, але залишалась потреба великих обчислювальних ресурсів для покриття. Під керівництвом професора Брайтона (Brayton R. K.) в університеті Берклі (США) розроблено програму *ESPRESSO*. В основу її евристичного підходу лягли роботи Мореаля (Morreale E.), Ройша (Reusch B.) та Cacao (Sasao T.). У перших

наближених алгоритмах мінімізації евристичні підходи застосовувалися до етапу покриття, а на першому етапі обчислювали точний розв'язок — множину усіх простих кон'юнктермів. Згодом було розроблено метод, у якому віднаходили лише частину простих кон'юнктермів. На ньому побудовано програму *ESPRESSO-SIGNATURES*.

Вихідний код програми *ESPRESSO* мовою *C* можна безкоштовно завантажити з сайту розробника. Також цей алгоритм використовується у середовищах розробки пристроїв на базі ПЛІС. Крім того формат файлів програми *ESPRESSO* підтримують деякі пакети для проектування цифрової техніки.

В основу алгоритму *ESPRESSO* покладено рекурсивний розклад Шеннона, що являє собою суперпозицію однорідних функцій , які фактично є монотонними функціями [112]. Недоліком алгоритму є залежність усунення чи наявності надлишковості у результуючій множині від обраної послідовності простих кон'юнктермів. Ця проблема розв'язується методом гілок та обмежень. За свідченням авторів метод *ESPRESSO* є ефективний для мінімізації булових функцій, що мають відносно невелику кількість змінних.

Із сучасних евристичних методів викликає інтерес метод розчеплення кон'юнктермів, що запропонував професор Рицар Б. Є. [52]. Автор розробив програму SPLIT, у якій реалізовано цей метод, і численними експериментами довів вищу точність власного методу, ніж в *ESPRESSO*. Процедура розчеплення кон'юнктермів дає змогу одержати скорочену ДНФ, для покриття якої запропоновано теоретико-множинну форму мінімаксного методу покриття [93].

Із сучасних точних методів пошуку множини простих кон'юнктермів викликає зацікавлення метод порозрядного вирощування [107], оскільки він має асимптотично мінімальну оцінку комбінаторної складності.

1.2.2 Метод розчеплення кон’юнктермів

У [52] подано метод мінімізації булових функцій, основою якого є процедура розчеплення кон’юнктермів (додаток Б).

Нехай задано булову функцію f від n змінних. Процедура розчеплення мінтерма утворює n кон’юнктермів рангу (n-1) шляхом заміни у цьому мінтермі по черзі кожного одного біта символом поглинання “-”. Наприклад, для мінтерма рангу 3 (101) процедура розчеплення дасть три кон’юнктерми рангу 2, а саме (– 01), (1–1), (10–). Для кон’юнктермів рангів нижчих ніж n у процедурі розчеплення відбувається заміна символом поглинання “–” тільки нулів та одиниць. Таким чином по відношенню до символу поглинання процедура розчеплення є інваріантною, розчепленню підлягають тільки булові літерали. Наприклад, розчеплення кон’юнктерма рангу 2 (1–1) дає два кон’юнктерми рангу 1, а саме (– –1) та (1– –).

Рівень мінімізації розчепленням q — це кількість поглинутих змінних у кон’юнктермі.

$$q = n - r, \tag{1.1}$$

де r — ранг кон’юнктерма, тобто кількість непоглинутих змінних.

Рівень мінімізації може набувати значення $q = 0,1,\ldots,(n-1)$. Незалежно від рівня q процедура розчеплення відбувається за одним і тим же алгоритмом, причому з ростом рівня q зменшується кількість новоутворених кон’юнктермів рівня (q+1).

Із N кон’юнктермів вихідної множини рівня q процедура розчеплення видає $(r \times N)$ кон’юнктермів рівня (q+1), що утворюють матрицю розчеплення $MP(q+1)$ відповідної розмірності. Кожен її рядок (вектор) має спільну для усіх кон’юнктермів цього вектора маску літералів (набір непоглинутих змінних).

Один і той самий кон’юнктерм q-рівня може бути одержаний заміною нуля чи одиниці символом поглинання з двох різних кон’юнктермів, що відрізняються лише одним бітом. Тож на одному векторі матриці розчеплення $MP(q+1)$ може з’явитись не більше двох копій одного кон’юнктерма рівня $(q+1)$. Причому поява копій підтверджує наявність у буловій функції кон’юнктерма, який одержано внаслідок розчеплення. Вектор матриці $MP(q+1)$, що містить кон’юнктерми-копії (хоча б одну пару), вважають покритим.

Усі кон’юнктерми рівня $(q+1)$, що одержані процедурою розчеплення з одного твірного кон’юнктерма рівня q, у матриці розчеплення $MP(q+1)$ утворюють стовпець. Якщо два стовпці на деякому векторі містять однакові кон’юнктерми копії, то відповідні їм твірні кон’юнктерми вважають неортогональними, інакше — ортогональними. Якщо всі кон’юнктерми у стовпці не мають копій на своїх векторах, то відповідний твірний вважають ізольованим. Звідси випливає, що ізольований кон’юнктерм є ортогональним відносно усіх кон’юнктермів заданої булової функції. Як наслідок, якщо у матриці немає покритих векторів, тоді усі твірні кон’юнктерми є ізольованими.

Ізольовані кон'юнктерми потрібно віднести до множини простих кон’юнктермів, а відповідні стовпці матриці розчеплення вилучити з подальшого розгляду. Тоді у матриці залишаться тільки стовпці, у яких є принаймні один кон’юнктерм, що має копію на відповідному векторі. Така матриця розчеплення покрита власними векторами. Множина кон’юнктермів-копій з покритих векторів є покриттям матриці розчеплення $MP(q+1)$. Кон’юнктерми одержаної множини покриттям є твірними для процедури розчеплення на наступному рівні. Процедура розчеплення застосовується до нових рівнів, якщо у твірній множині є більше одного кон’юнктерма.

Вектор матриці розчеплення, що складається виключно з кон’юнктермів копій називають повністю покритим, а матрицю розчеплення, що має хоча б один повністю покритий вектор, називають повністю покритою. Якщо матриця розчеплення містить тільки повністю покриті вектори, то вона є вироджена. Очевидно, що вироджена матриця може бути тільки за умови наявності 2^r

твірних кон'юнктермів. При цьому диз'юнкція усіх кон'юнктермів будь-якого вектора дорівнює одиниці.

Починаючи з мінтермів на будь-якому рівні q процедура розчеплення виконується однотипно, реалізовується відносно просто за допомогою маски літералів.

Можна скористатись процедурою розчеплення кон'юнктермів, щоб для довільної булової функції побудувати повне кон'юнктермове поле. Для цього починаючи з мінтермів необхідно виконати розчеплення послідовно на всіх рівнях $q = 1,2,\ldots,(n-1)$. Але при цьому починаючи з другого рівня з'явиться тавтологія так само як і в методі Квайна та подібних. Обсяг тавтології наростатиме за степеневою залежністю 2^{q-1}. Кількість кон'юнктермів N_q на кожному рівні буде визначатись комбінаторною залежністю [52]:

$$N_q = q \cdot C_q^n = \frac{q \cdot n!}{q!(n-q)!}. \qquad (1.2)$$

Щоб уникнути тавтології застосовують матрицю розчеплення літералів *MPLq*, яка за означенням являє собою матрицю стовпець. Її елементами є розчеплені маски літералів рівня (q+1). Твірна маска літералів рівня q утворюється з твірного кон'юнктерма шляхом заміни в ньому нулів та одиниць літерою *l*. Розчеплені маски літералів утворюють заміною в твірній масці літералів однієї з літер *l* на символ поглинання "–". Наприклад, твірному кон'юнктерму (1–0–) відповідає твірна маска літералів (*l*–*l*–), з якої утворюють розчеплені маски літералів (– –*l*–), (*l*– – –), що є елементами матриці розчеплення літералів *MPLq*.

Будь-який вектор матриці розчеплення літералів *MPLq* на певному рівні q еквівалентний повній множині розчеплених кон'юнктермів відповідної маски літералів. Таких векторів є C_n^q і вони разом складають повну множину розчеплених кон'юнктермів рівня (q+1) заданої булової функції. Звідси випливає, що матриця розчеплення літералів *MPLq* еквівалентна повній множині

розчеплених кон'юнктермів L_n^q рівня (q+1). Але на відміну від повного кон'юнктермового поля кількість розчеплених кон'юнктермів у $q!$ разів менша, а саме $C_n^q \times 2^{n-q}$. Таким чином процедура розчеплення літералів набуває вигляду:

$$L_n^0 \overset{S}{\Rightarrow} L_n^1 \overset{S}{\Rightarrow} L_n^2 \overset{S}{\Rightarrow} L_n^3 \overset{S}{\Rightarrow} \ldots \overset{S}{\Rightarrow} L_n^{n-2} \overset{S}{\Rightarrow} L_n^{n-1} \overset{S}{\Rightarrow} \emptyset. \qquad (1.3)$$

де $\overset{S}{\Rightarrow}$ — символ оператора розчеплення.

Для мінімізації булової функції методом розчеплення кон'юнктермів необхідно, починаючи з мінтермів, виконати на кожному рівні процедуру розчеплення, тобто сформувати матрицю *MP*(q+1), а потім виконати процедуру покриття цієї матриці. Після завершення розчеплення на усіх рівнях усунути надлишковість і таким чином отримати остаточний результат — мінімальне покриття заданої функції.

Щоб сформувати мінімальне покриття матриці розчеплення, необхідно після вилучення ізольованих кон'юнктермів знайти мінімальну кількість власних векторів цієї матриці, що покривають усі її стовпці.

Позначимо $E_n^{q\prime}$ покриті вектори матриці розчеплення певного рівня q, а E_n^{q+1} — матрицю розчеплення наступного рівня. Вектори $E_n^{q\prime}$ виступають твірними для E_n^{q+1}. У такому випадку розчеплення кон'юнктермів усіх рівнів набуває вигляду:

$$E_n^0 \overset{S}{\Rightarrow} E_n^1 \overset{C}{\Rightarrow} E_n^{1\prime} \overset{S}{\Rightarrow} E_n^2 \overset{C}{\Rightarrow} E_n^{2\prime} \overset{S}{\Rightarrow} E_n^3 \overset{C}{\Rightarrow} E_n^{3\prime} \overset{S}{\Rightarrow} \ldots \qquad (1.4)$$

де $\overset{C}{\Rightarrow}$ — символ оператора покриття матриці розчеплення.

Нехай $\alpha_i, \beta_i, \gamma_i, \ldots, \rho_i \in \{0,1,-\}$ для будь-якого $i \in \{0,1,\ldots,n-1\}$.

Тоді $K^{\alpha} = (\alpha_{n-1} \cdots \alpha_1 \alpha_0)$, $K^{\beta} = (\beta_{n-1} \cdots \beta_1 \beta_0)$, $K^{\gamma} = (\gamma_{n-1} \cdots \gamma_1 \gamma_0)$,…, $K^{\rho} = (\rho_{n-1} \cdots \rho_1 \rho_0)$ — кон'юнктерми у псевдотрійковому поданні.

Якщо на певному рівні q виконується умова:

$$K^{\alpha} \cap K^{\beta} \cap K^{\gamma} \cap \cdots \cap K^{\rho} = \varnothing, \tag{1.5}$$

тоді кон'юнктерми K^{α}, K^{β}, K^{γ},…, K^{ρ} є остаточні.

Також кон'юнктерми будуть остаточними, коли жоден елемент покриття не міститься в об'єднанні взаємних перетинів кон'юнктермів цього покриття.

На кожному рівні розчеплення ключову роль для мінімізації відіграє процедура пошуку мінімального покриття матриці розчеплення. Нехай k — потужність множини кон'юнктермів матриці $MP(q+1)$. Алгоритм усунення надлишковості сформулюємо так:

1) побудувати таблицю взаємних перетинів усіх пар кон'юнктермів за таким правилом:

$$K^{\alpha} \cap K^{\beta} = \begin{cases} \emptyset, \text{якщо існує перетин } \alpha_i \cap \beta_i = \emptyset, \\ K^{\alpha \cap \beta}, \text{якщо не існує перетин } \alpha_i \cap \beta_i = \emptyset, \end{cases} \tag{1.6}$$

де $K^{\alpha} = (\alpha_{n-1} \cdots \alpha_1 \alpha_0)$ та $K^{\beta} = (\beta_{n-1} \cdots \beta_1 \beta_0)$ — кон'юнктерми,

$$K^{\alpha \cap \beta} = ((\alpha_{n-1} \cap \beta_{n-1}) \cdots (\alpha_1 \cap \beta_1)(\alpha_0 \cap \beta_0)) \tag{1.7}$$

2) обчислити об'єднання k' непорожніх перетинів (k' ≤ k)

$$\bigcup_{\substack{i=1 \\ j=1}}^{k'} K_{ij}^{\alpha \cap \beta} \tag{1.8}$$

кожного кон'юнктерма в i-му рядку K_i (j — індекс перебору стовпців);

3) кон'юнктерм K_i з розгляду вилучають, якщо

$$K_i \setminus \bigcup_{\substack{i=1 \\ j=1}}^{k'} K_{ij}^{\alpha \cap \beta} = \emptyset \qquad (1.9)$$

інакше K_i відносять до множини остаточного результату.

Щоб змінімізувати систему з m булових функцій від n змінних методом розчеплення кон'юнктермів, згідно з [91] необхідно дотримуватися такого алгоритму:

1) для кожної функції системи виконати процедуру розчеплення кон'юнктермів. При цьому виконати мінімальне покриття матриць розчеплення цих функцій так, щоб у покритті було максимум векторів з однаковою маскою літералів. На кожному рівні (q+1) одержимо множину кон'юнктермів об'єднаного покриття системи функцій;

2) з множини об'єднаного покриття побудувати таблицю усіх кон'юнктермів. Виявити у цій таблиці кон'юнктерми, що однакові для різних функцій. Це будуть остаточні кон'юнктерми системи;

3) виявити та вилучити надлишкові кон'юнктерми у кожній функції.

Щоб здійснити мінімальне покриття матриці розчеплення, необхідно знайти розчеплені кон'юнктерми-копії у межах одного вектора матриці.

У розчепленні бере участь лише двійкова частина кон'юнктерма, але записується кон'юнктерм у псевдотрійковому зображенні, що ускладнює пошук копій, порівняно з опрацюванням числового подання.

1.2.3 Метод порозрядного вирощування

У [107] проаналізовано метод порозрядного вирощування простих кон'юнктермів. За твердженням автора методу його алгоритм не допускає надлишкових порівнянь термів і в цьому сенсі є непокращуваним.

Для деякої булової функції від *n* змінних, що задана множиною мінтермів, упорядковують в однаковий спосіб змінні в усіх мінтермах, а саме у порядку спадання індексів змінних зліва направо. Змінну без інверсії кодують одиницею, а з інверсією — нулем. Одержані у такий спосіб двійкові коди мінтермів розташовують за зростанням.

За окремою змінною множину всіх можливих кон'юнктермів заданої булової функції можна розбити на три підмножини:

— кон'юнктерми, що містять інверсне значення цієї змінної (у двійковому коді мінтермів її подано нулем "0");

— кон'юнктерми, що містять пряме значення цієї змінної (у двійковому коді мінтермів її подано одиницею "1");

— кон'юнктерми, у яких ця змінна поглинута (позначається символом поглинання "–" у псевдотрійковому коді).

Оскільки мінтерми впорядковані за зростанням двійкових кодів, то ті, що відрізняються лише молодшим бітом і, відповідно, можуть склеїтися по цьому біту, мають бути розташовані поруч. Тому спочатку шукають склеювання мінтермів за молодшим бітом. Для цього, починаючи з найменшого двійкового коду, порівнюють сусідні мінтерми впорядкованої множини. При цьому формують також множину мінтермів, що містять нуль у молодшому біті, та множину мінтермів, що містять одиницю в цьому біті, записуючи розглянуті мінтерми у відповідну множину. У такий спосіб одержують три множини кон'юнктермів.

Всі кон'юнктерми нижчих рангів, що містять нуль у розглянутому біті, можна одержати тільки склеюванням між собою мінтермів одержаної множини із нулем у цьому біті. Аналогічно, кон'юнктерми нижчих рангів із одиницею у розглянутому біті одержують тільки склеюванням мінтермів одержаної множини з одиницею в цьому біті. А кон'юнктерми нижчих рангів, у яких поглинута розглянута змінна, можна одержати тільки з кон'юнктермів множини, що одержана склеюванням мінтермів по розглянутому біту.

У кожній з трьох одержаних множин значення розглянутої змінної однакове для усіх кон’юнктермів відповідної множини, тому для здійснення подальших порівнянь праву змінну відкидають як неістотну для певної множини кон’юнктермів. При цьому ранг кожного терму зменшується на один, а в поданні залишаються тільки нулі й одиниці, що дає змогу оперувати двійковими кодами замість псевдотрійкових.

Описану процедуру пошуку склеювань по молодшому біту і формування за цим бітом трьох множин застосовують окремо для кожної з множин, що одержані на попередньому кроці. Пошук завершується тоді, коли в одержаних підмножинах залишається тільки по одному кон’юнктерму. Для відновлення подання кон’юнктерма необхідно справа у зворотному до одержання порядку дописати незалежні змінні з попередніх кроків.

У побудованому трійковому дереві присутні всі кон’юнктерми заданої булової функції. Щоб виокремити прості кон’юнктерми запропоновано двійковий код помітки. Його формують для кожного кон’юнктерма. Під час формування трьох множин за певним бітом код помітки встановлюють у такий спосіб:

— якщо кон’юнктерм брав участь у склеюванні по розглянутому біту, у новій множині у його коді помітки встановлюється нуль у відповідному біті, інакше – одиниця;

— якщо кон’юнктерм одержано склеюванням по розглянутому біту, то код помітки дорівнює побітовій диз’юнкції кодів помітки кон’юнктермів, що склеїлись. Якщо коди помітки відсутні у склеюваних кон’юнктермів, то цей код відсутній в одержаного склеюванням кон’юнктерма.

Простими є кон’юнктерми, у коді помітки яких немає нулів у жодному біті. Якщо в процесі пошуку в деякій з одержаних множин усі кон’юнктерми містять нуль у певному біті коду помітки, то цю множину вилучають з подальшого розгляду, оскільки з неї неможливо одержати прості кон’юнктерми.

У методі порозрядного вирощування відбувається швидке розбиття задачі на незалежні підзадачі, що можуть бути виконані у паралельних обчислювальних

процесах. Тоді верхня оцінка обсягу пам’яті для зберігання усіх кон’юнктермів булової функції від n змінних дорівнює $O(3^n)$. Проте для аналізу за кожною наступною змінною відбувається повне пересортування усіх наявних кон’юнктермів, починаючи з мінтермів. Це потребує додаткового обсягу пам’яті.

За твердженням автора методу його комбінаторна складність у загальному випадку збігається з нижньою оцінкою складності задачі про порівняння N чисел, що дорівнює $O(N \log N)$. Тому його можна вважати асимптотично оптимальним.

1.2.4 Мінімаксний метод покриття таблиці простих кон’юнктермів

Переважно множина усіх простих кон’юнктермів булової функції є надлишковою для реалізації заданої функції. Тому після знаходження усіх простих кон’юнктермів другим етапом мінімізації є пошук множини мінімальної потужності, достатньої для реалізації цієї функції, тобто задача пошуку найменшого покриття таблиці простих кон’юнктермів. Зростання розмірності булових функцій призвело до того, що точні методи покриття вимагають таких великих обчислювальних ресурсів, які часто складно реалізувати на практиці. Тому з’явились евристичні підходи. Одним з найхарактерніших є мінімаксний метод [93].

У мінімаксному методі пошук покриття виконується на основі евристичних оцінок. Здійснюються процедури прямого та комбінованого пошуку найкоротшого покриття матриці, а також процедура редукування матриці. Зазначимо, що комп’ютерна реалізація процедур із матрицями потребує досить великих обчислювальних витрат.

Професор Рицар Б. Є. запропонував теоретико-множинну модифікацію мінімаксного методу покриття [93]. Алгоритм цієї модифікації реалізується за допомогою операцій теорії множин над числами та символами, що дає змогу

простіше та швидше виконувати проміжні етапи пошуку мінімального покриття функції. Для цього задана булова функція *f* від *n* змінних подається у теоретико-множинній формі (ТМФ) [93]. Кожному простому кон'юнктерму скороченої ТМФ ставлять у відповідність символ (літеру). Такий символ є символьним простим кон'юнктермом заданої функції. Таким чином одержуємо подання булової функції у вигляді множини символьних простих кон'юнктермів.

Нехай булова функція від *n* змінних $f(x_{n-1},\dots,x_1,x_0)$ задана множиною з *N* мінтермів. У теоретико-множинній формі кожен мінтерм позначимо як m_i, де $i \in \{1,2,\dots,N\}$. Тоді досконала ТМФ цієї функції має вигляд:

$$Y^1 = \{m_1, m_2, \dots, m_N\}^1. \tag{1.10}$$

У псевдотрійковому коді кон'юнктерм рангу *r*, що одержаний склеюванням g мінтермів (де $g = 2^{n-r}$), являє собою *n*-позиційний код $(\sigma_{n-1},\dots,\sigma_j,\dots,\sigma_1,\sigma_0)$, де $\sigma_j \in \{0,1,-\}$, $j \in \{0,1,2,\dots,n-1\}$. У теоретико-множинній формі той самий кон'юнктерм є підмножиною цілих невід'ємних чисел $(m_1, m_2, \dots, m_g)$ від множини $\{0,1,2,\dots,2^n-1\}$.

Наприклад, терм функції від п'яти змінних $\bar{x}_4\bar{x}_3x_2x_1x_0 \equiv (00111) \equiv (7)$ — це кон'юнктерм рангу 5, тобто мінтерм, а терм $\bar{x}_4x_0 \equiv (0---1) \equiv (1,3,5,7,9,11,13,15)$ — кон'юнктерм рангу 2. Позначимо цей кон'юнктерм латинською літерою *K*. Тепер це символьний кон'юнктерм.

Таким чином символьний кон'юнктерм K_i певного рангу $r_i \in \{1,2,\dots,n\}$ являє собою множину з g_i мінтермів, що склеюються між собою, де $g_i = 2^{n-r_i}$. Цю множину мінтермів записуємо в круглих дужках:

$$K_i = (m_{i_1}, m_{i_2}, \dots, m_{g_i}). \tag{1.11}$$

Тоді скорочена ТМФ функції Y^1 являє собою множину $\{K_1, K_2, \dots, K_p\}$ простих кон'юнктермів, де *p* — кількість простих кон'юнктермів:

$$Y^1 = \{K_1, K_2, \ldots, K_p\}^1. \quad (1.12)$$

За означенням [93] множину всіх символьних простих кон'юнктермів, що містять деякий мінтерм m_j, позначимо символом D_j, та називатимемо символьним диз'юнктермом цієї функції:

$$D_j = \{K_{ji} | m_j \in K_{ji}\}, \quad (1.13)$$

де $K_{ji} \in \{K_1, K_2, \ldots, K_p\}$.

Таким чином символьний диз'юнктерм в аналітичній формі є диз'юнкцією символьних простих кон'юнктермів. Тоді за умови $D_k \subseteq D_l$ символьний диз'юнктерм D_l буде поглинутий символьним диз'юнктермом D_k. Звідси випливає, що задачу покриття можна спростити, якщо вилучити з розгляду ті символьні диз'юнктерми, що містять у собі інші диз'юнктерми.

Для покриття заданої функції спершу необхідно за умовою (1.13) сформувати усі символьні диз'юнктерми. У найпростішому випадку кожен символьний диз'юнктерм складається лише з одного символьного кон'юнктерма і жоден символьний диз'юнктерм не може бути поглинутий іншим. Тоді усі прості кон'юнктерми $K_1, K_2, \ldots, K_p$ є істотні, тобто для заданої функції мінімальною є вихідна скорочена ТМФ. Тож кінцевою метою алгоритму покриття є спростити скорочену ТМФ до множини одноелементних символьних диз'юнктермів, що разом містять усі мінтерми заданої булової функції. Одноелементний символьний диз'юнктерм фактично є тотожний істотному кон'юнктерму. У теоретико-множинному підході для досягнення зазначеної мети використовуються процедури спрощення та зважування.

Процедура спрощення одержаної множини символьних диз'юнктермів здійснюється операціями поглинання між множинами символьних кон'юнктермів, що відповідають певним символьним диз'юнктермам. Таким чином вдається вилучити ті символьні диз'юнктерми, що поглинають інші.

До неспрощуваної множини символьних диз'юнктермів на кожному кроці покриття функції застосовується процедура зважування. Для певного символьного диз'юнктерма D_h (1.13) зважування являє собою порівняння потужностей $P^s_{K_j}$ символьних кон'юнктермів K_j, що містяться в ньому.

$$P^s_{K_j} = \sum_{i=1}^{2^{n-r_j}} P^s_{m_i}, \qquad (1.14)$$

де $r_j \in \{1,2,3,\dots,n\}$,

$s = 1,2,3,4,\dots$ — номер кроку покриття,

$P^s_{m_i}$– потужність i-го мінтерма m_i на кроці s, який міститься в j-му кон'юнктермі r_i-рангу. $P^s_{m_i} = 1$ за умови, що мінтерм m_i не міститься в жодному істотному кон'юнктермі, інакше $P^s_{m_i} = 0$.

Результатом процедури зважування є виявлення кон'юнктерма найбільшої потужності, який потрібно віднести до множини результату.

Нехай маємо двоелементний диз'юнктерм $D_1 = (K_1, K_2)$, що складається з простих кон'юнктермів $K_1 = (m_{i_1}, m_{i_2}, m_{i_3}, \dots, m_{i_{2^{n-r_1}}})$ і $K_2 = (m_{i_1}, m_{i_2}, m_{i_3}, \dots, m_{i_{2^{n-r_2}}})$. Порівнюємо потужності цих кон'юнктермів $P^1_{K_1}$ і $P^1_{K_2}$. Якщо, потужність одного з них більша, наприклад $P^1_{K_1} > P^1_{K_2}$, то розглянутий символьний диз'юнктерм D_1 вилучаємо з подальшого розгляду. При цьому простий кон'юнктерм з більшою потужністю (в прикладі це K_1) заносимо у множину покриття як істотний кон'юнктерм мінімальної ТМФ. Після цього вилучаємо мінтерми, що належать кон'юнктерму K_1, з множин мінтермів інших простих кон'юнктермів, щоб не враховувати їх на подальших кроках покриття при розрахунку потужностей (1.14) символьних кон'юнктермів інших символьних диз'юнктермів. Після завершення процедури зважування на кожному кроці s виконуємо процедуру спрощення зредукованої множини. Наступним кроком беремо другий диз'юнктерма D_2 і застосовуємо до нього процедуру зважування і потім процедуру спрощення множини символьних

диз'юнктермів після вилучення D_2. Процес завершується, коли буде виконано зважування для усіх символьних диз'юнктермів.

Якщо деякий символьний диз'юнктерм D_j містить декілька символьних кон'юнктермів, що мають однакові потужності, то маємо точку розгалуження. Отримаємо декілька розв'язків покриття залежно від вибору символьного кон'юнктерма. Оскільки може бути багато таких розгалужень, тому обираємо будь-який з кон'юнктермів найбільшої потужності та заносимо його у множину покриття як істотний.

Розглянута теоретико-множинна модифікація мінімаксного методу покриття булових функцій вирізняється простотою реалізації процедур порівняно з відомими аналітичними та матричними [52] методами.

Оскільки в обчислювальній техніці подання символів здійснюється числами, то до описаних вище символьних процедур можна застосувати способи опрацювання множини чисел для зменшення обчислювальних витрат.

1.3 Висновок до розділу 1

1) Виконано аналіз літератури щодо питання порівняння довільних булових базисів за критерієм складності булових функцій у класі формул. Встановлено, що існує нескінченна множина попарно непорівнюваних базисів і нескінченний строго спадний ланцюжок базисів. Тобто структура базисів є нескінченно широкою і нескінченно довгою. Не існує мінімального базису, при цьому найбільшим за критерієм складності реалізації булових функцій є базис {І, ЧИ, НЕ}.

2) З аналізу літератури встановлено, що переважна більшість практичних задач мінімізації стосується реалізації булових функцій у базисі {І, ЧИ, НЕ}, у тому числі для проектування цифрових пристроїв на базі програмованих

логікових інтегральних схем. Враховуючи викладене вище для подальших досліджень із довільних булових базисів обрано базис {І, ЧИ, НЕ}.

3) У зв'язку із ростом ступеня інтеграції сучасних цифрових інтегральних схем, зростає розмірність задач проектування цифрових комбінаційних пристроїв і, відповідно, вимоги до обчислювальних ресурсів, що необхідні для розв'язання задач мінімізації булових функцій.

4) Точні методи мінімізації хоч і дають змогу знайти глобальний мінімум кошту функційної реалізації, але вимагають значних обчислювальних витрат, що для задач великої розмірності є практично нереалізовними. Евристичні методи не гарантують одержання глобального мінімуму, але можуть забезпечити прийнятний результат значно меншими обчислювальними витратами. Точні методи використовують для розв'язання задач малої розмірності, а евристичні - для задач великої розмірності.

5) Задачу мінімізації можна розбити на два етапи, а саме пошук множини простих кон'юнктермів та покриття цієї множини для усунення надлишковості. Причому другий етап має вищу складність, оскільки є *NP*-повною задачею.

6) Обрано для подальших досліджень як перспективні щодо оптимізації обчислювальних затрат один точний метод пошуку множини простих кон'юнктермів, один наближений метод пошуку множини простих кон'юнктермів, один наближений метод покриття множини простих кон'юнктермів, а саме: метод порозрядного вирощування, метод порозрядного вирощування, мінімаксний метод покриття.

7) Встановлено, що спільним недоліком переважної більшості методів є символьне подання кон'юнктермів, що ускладнює їхню комп'ютерну реалізацію. Таким чином, одним із шляхів оптимізації обчислювальних витрат є пошук числового подання кон'юнктермів [90], оскільки операції над числами в комп'ютері простіше реалізувати, ніж операції над наборами символів.

2 ТЕОРЕТИЧНІ ЗАСАДИ ОПТИМІЗАЦІЇ АЛГОРИТМІВ МІНІМІЗАЦІЇ БУЛОВИХ ФУНКЦІЙ

Одну й ту саму функцію можна подати різними формулами у базисі {І, ЧИ, НЕ}. Різні формули однієї функції можуть мати різний коефіцієнт складності, що призводить до різного кошту реалізації цифрових комбінаційних пристроїв. Завдання мінімізації булових функцій — знайти логіковий вираз із мінімальним коефіцієнтом складності. Це задача *NP*-складності, яка вимагає значних обчислювальних ресурсів, що у свою чергу обмежує коло розв'язуваних задач. У попередньому розділі розглянуто відомі підходи до задачі мінімізації, з яких обрано три методи для подальших досліджень щодо пошуку шляхів зменшення обчислювальних витрат.

2.1 Числові форми подання кон'юнктермів булових функцій

Ефективність методу мінімізації напряму залежить від форми подання булових функцій. Аналітична форма, що використовувалась у перших відомих методах, придатна для ручної мінімізації булових функцій невеликої кількості змінних. Те саме стосується графічних методів таких як метод карт Карно чи метод візерунків. У переважній більшості відомих методів мінімізації використовується символьне подання кон'юнктермів, зокрема псевдотрійкове, що ускладнює комп'ютерну реалізацію цих методів. Оскільки дані у сучасній обчислювальній техніці зберігаються і опрацьовуються у вигляді чисел, одним із шляхів зменшення обчислювальних витрат може бути подання кон'юнктермів числами.

2.1.1 Використання кортежу чисел для подання кон’юнктерма

З урахуванням недоліків псевдотрійкового подання у співавторстві з професором Рицарем Б.Є. для методу розчеплення кон'юнктермів було розроблено числове подання кон'юнктермів — R-формат [90]. Таке подання допоможе спростити процедуру пошуку однакових кон’юнктермів на векторі матриці розчеплення.

Особливість процедури розчеплення кон’юнктермів полягає у тому, що на будь-якому q-рівні розчеплення подальше розчеплення твірних відбувається тільки по двійкових літералах, а символ поглинання “–” не бере участі у цій процедурі. Тому можна вилучити символи поглинання з псевдотрійкового подання. Тоді залишаться тільки нулі й одиниці, які можна інтерпретувати як двійкове число. Для того, щоб можна було відновити аналітичне подання, достатньо встановити відповідність між номерами бітів двійкового зображення та змінними булової функції. Запропонований R-формат встановлює цю відповідність.

Нехай задано булову функцію від n змінних $f(x_1, x_2, x_3, \dots, x_{n-1}, x_n)$. Для кожної змінної $x_{l\ i}$ ($x_l \in \{x_1, x_2, x_3, \dots, x_{n-1}, x_n\}$) поставимо у відповідність деяке число i з ряду цілих чисел від 0 до n-1 ($i = 0,1,2,\dots, n-1$). Вважатимемо ці числа номерами бітів двійкового подання мінтерма. Тоді старший біт з номером $(n-1)$ відповідає змінній x_1 ($x_1 \to (n-1)$), біт $(n-2)$ — змінній x_2 ($x_2 \to (n-2)$), … , біт з номером 1 — змінній $x_{n\ -\ 1}$ ($x_{n-1} \to 1$) і молодший біт 0 — змінній x_n ($x_n \to 0$). Таким чином одержали спадну послідовність цілих чисел. Такий спосіб кодування ставить у відповідність кожному кон'юнктерму єдину спадну послідовність чисел. Тепер можна вилучити символи поглинання із псевдотрійкового подання кон'юнктерма. Тоді нулі й одиниці, що залишилися, можна інтерпретувати як двійкове число, біля якого у нижньому індексі запишемо у вигляді кортежу послідовність чисел (у трикутних дужках $\langle \dots \rangle$), що відображає номери літералів. Числа можна подавати як у двійковій так і у

десятковій системах числення. Наприклад, $10_{2\ \langle 3,0\rangle} = 2_{10\ \langle 3,0\rangle}$, що відповідає (1 – – 0).

У цьому контексті *R*-індекс змінної в кортежі — це номер літерала. *R*-формат кон'юнктерма — це числове його подання із кортежем *R*-індексів. Процедуру кодування кон'юнктерма називатимемо *R*-форматуванням і позначатимемо оператором $\overset{F_R}{\Rightarrow}$.

Проілюструємо викладене вище на прикладі кон'юнктерма булової функції від шести змінних $f(x_1, x_2, x_3, x_4, x_5, x_6)$. У аналітичному поданні маємо $x_1\bar{x}_3x_5$. Тоді у псевдотрійковому поданні маємо (1 – 0 – 1 –). Для R-форматування цієї функції сформуємо карту відповідності *R*-індексів змінним (рисунок 2.1).

i	5	4	3	2	1	0
x_i	x_1	x_2	x_2	x_4	x_5	x_6

Рисунок 2.1 – Карта відповідності *R*-індексів змінним для булової функції від шести змінних

Виконуючи описану вище процедуру *R*-форматування, одержимо:

$$x_1\bar{x}_3x_5 = (1 – 0 – 1 –) \overset{F_R}{\Rightarrow} 101_{2\ \langle 5,3,1\rangle} = 5_{10\ \langle 5,3,1\rangle}. \qquad (2.1)$$

Для переходу від *R*-форматованого подання назад до аналітичного використовуватимемо процедуру зворотного *R*-форматування, яку позначатимемо оператором $\overset{F_R^*}{\Rightarrow}$. Використовуючи карту відповідності *R*-індексів замінимо нулі й одиниці у *R*-форматованому поданні на значення відповідних змінних. Решта змінних відсутні, тому замість них запишемо символи поглинання "–". Розглянемо приклад зворотного *R*-форматування:

$$5_{10\,\langle 5,3,1\rangle} = 101_{2\,\langle 5,3,1\rangle} \overset{F_R^*}{\Rightarrow} (1\text{–}\,0\,\text{–}1\text{–}) = x_1\bar{x}_3x_5. \qquad (2.2)$$

У матриці розчеплення *MP*(q+1) кожен вектор є множиною кон'юнктермів, що мають спільну маску поглинутих бітів. Це означає, що на кожному векторі матриці розчеплення унаслідок процедури *R*-форматування кон'юнктермів одержують кон'юнктерми зі спільним *R*-форматом. Таким чином можна зазначити спільний *R*-формат для усіх кон'юнктермів вектора і використовувати його як індекс вектора у процедурі покриття матриці. Якщо процедури розчеплення і покриття виконувати вручну, то краще користуватися десятковим поданням кон'юнктермів, оскільки людина краще сприймає десяткові числа ніж двійкові.

Якщо змінна відсутня у заданому кон'юнктермі то, відповідно, і буде відсутній її індекс у *R*-форматованому поданні. Якщо у *R*-форматованому поданні є усі *R*-індекси заданої булової функції, то це мінтерм. Отже кон'юнктерм довільного рангу може бути поданий у *R*-форматі.

Розглянемо на прикладі використання *R*-форматування у методі розчеплення кон'юнктермів для мінімізації булової функції. Однакові кон'юнктерми на векторах матриці будемо підкреслювати.

Приклад 2.1 [90]

Нехай задано булову функцію від чотирьох змінних $f(x_1, x_2, x_3, x_4)$, досконала ТМФ якої:

$$Y^1 = \{2, 3, 4, 6, 7, 8, 9, 10, 11, 15\}^1.$$

Розв'язання. У двійковій формі задана функція набуває вигляду:

$$Y^1 = \{0010,\ 0011,\ 0100,\ 0110,\ 0111,\ 1000,\ 1001,\ 1010,\ 1011,\ 1111\}^1.$$

Складемо карту відповідності *R*-індексів (рисунок 2.2).

i	3	2	1	0
x_i	x_1	x_2	x_2	x_4

Рисунок 2.2 – Карта відповідності *R*-індексів змінним для булової функції від чотирьох змінних

$$Y^1 = [l \quad l \quad l \quad l] \overset{S}{\Rightarrow} \begin{bmatrix} l & l & l & - \\ l & l & - & l \\ l & - & l & l \\ - & l & l & l \end{bmatrix} =$$

$$= \begin{bmatrix} 001- & 001- & 010- & 011- & 011- & 100- & 100- & 101- & 101- & 111- \\ 00-0 & 00-1 & 01-0 & 01-0 & 01-1 & 10-0 & 10-1 & 10-0 & 10-1 & 11-1 \\ 0-10 & 0-11 & 0-00 & 0-10 & 0-11 & 1-00 & 1-01 & 1-10 & 1-11 & 1-11 \\ -010 & -011 & -100 & -110 & -111 & -000 & -001 & -010 & -011 & -111 \end{bmatrix} \overset{F_R}{\Rightarrow}$$

$$\overset{F_R}{\Rightarrow} \begin{bmatrix} [001 & 001 & 010 & 011 & 011 & 100 & 100 & 101 & 101 & 111]_{\langle 3,2,1\rangle} \\ [000 & 001 & 010 & 010 & 011 & 100 & 101 & 100 & 101 & 111]_{\langle 3,2,0\rangle} \\ [010 & 011 & 000 & 010 & 011 & 100 & 101 & 110 & 111 & 111]_{\langle 3,1,0\rangle} \\ [010 & 011 & 100 & 110 & 111 & 000 & 001 & 010 & 011 & 111]_{\langle 2,1,0\rangle} \end{bmatrix} =$$

$$= \begin{bmatrix} [\underline{1} & \underline{1} & 2 & \underline{3} & \underline{3} & \underline{4} & \underline{4} & \underline{5} & \underline{5} & 7]_{\langle 3,2,1\rangle} \\ [0 & 1 & \underline{2} & \underline{2} & 3 & \underline{4} & \underline{5} & \underline{4} & \underline{5} & 7]_{\langle 3,2,0\rangle} \\ [\underline{2} & \underline{3} & 0 & \underline{2} & \underline{3} & 4 & 5 & 6 & \underline{7} & \underline{7}]_{\langle 3,1,0\rangle} \\ [\underline{2} & \underline{3} & 4 & 6 & \underline{7} & 0 & 1 & \underline{2} & \underline{3} & \underline{7}]_{\langle 2,1,0\rangle} \end{bmatrix}_{\langle 3,2,1,0\rangle} \overset{C}{\Rightarrow}$$

Третій стовпець матриці покриває лише другий вектор, а останній стовпець покривають третій та четвертий вектори.

$$\overset{C}{\Rightarrow} \{[l \quad l \quad - \quad l], \begin{Bmatrix} [l & - & l & l] \\ [- & l & l & l] \end{Bmatrix}\} =$$

$$= \{\{2,4,5\}_{\langle 3,2,0\rangle}, \begin{Bmatrix}\{2,3,7\}_{\langle 3,1,0\rangle}\\ \{2,3,7\}_{\langle 2,1,0\rangle}\end{Bmatrix}\} =$$

$$= \{\{010,100,101\}_{\langle 3,2,0\rangle}, \begin{Bmatrix}\{010,011,111\}_{\langle 3,1,0\rangle}\\ \{010,011,111\}_{\langle 2,1,0\rangle}\end{Bmatrix}\} \overset{F_R^*}{\Rightarrow}$$

$$\overset{F_R^*}{\Rightarrow} \{\{(01-0),(10-0),(10-1)\}, \begin{Bmatrix}\{(0-10),(0-11),(1-11)\}\\ \{(-010),\ (-011),\ (-111)\}\end{Bmatrix}\} \overset{S}{\Rightarrow}$$

$$\overset{S}{\Rightarrow} \left\{ \begin{bmatrix} l & l & - & -\\ l & - & - & l\\ - & l & - & l \end{bmatrix}, \left\{ \begin{bmatrix} l & - & l & -\\ l & - & - & l\\ - & - & l & l \end{bmatrix} \right\} \right\} \left\{ \begin{bmatrix} l & l & - & -\\ l & - & - & l\\ - & l & - & l \end{bmatrix}, \left\{ \begin{bmatrix} l & - & l & -\\ l & - & - & l\\ - & - & l & l \end{bmatrix} \right\} \right\} =$$

$$= \{ \begin{bmatrix} 01-- & 10-- & 10--\\ 0--0 & 1--0 & 1--1\\ -1-0 & -0-0 & -0-1 \end{bmatrix}, \left\{ \begin{bmatrix} 0-1- & 0-1- & 1-1-\\ 0--0 & 0--1 & 1--1\\ --10 & --11 & --11 \end{bmatrix} \right\} \} \overset{F_R}{\Rightarrow}$$

$$\begin{bmatrix} -01- & -01- & -11-\\ -0-0 & -0-1 & -1-1\\ --10 & --11 & --11 \end{bmatrix}$$

$$\overset{F_R}{\Rightarrow} \{ \begin{bmatrix} [01 & 10 & 10]_{\langle 3,2\rangle}\\ [00 & 10 & 11]_{\langle 3,0\rangle}\\ [10 & 00 & 01]_{\langle 2,0\rangle} \end{bmatrix}, \left\{ \begin{bmatrix} [01 & 01 & 11]_{\langle 3,1\rangle}\\ [00 & 01 & 11]_{\langle 3,0\rangle}\\ [10 & 11 & 11]_{\langle 1,0\rangle} \end{bmatrix} \right\} \} =$$

$$\begin{bmatrix} [01 & 01 & 11]_{\langle 2,1\rangle}\\ [00 & 01 & 11]_{\langle 2,0\rangle}\\ [10 & 11 & 11]_{\langle 1,0\rangle} \end{bmatrix}$$

$$= \{ \begin{bmatrix} [1 & \underline{2} & \underline{2}]_{\langle 3,2\rangle}\\ [0 & 2 & 3]_{\langle 3,0\rangle}\\ [2 & 0 & 1]_{\langle 2,0\rangle} \end{bmatrix}, \left\{ \begin{bmatrix} [\underline{1} & \underline{1} & 3]_{\langle 3,1\rangle}\\ [0 & 1 & 3]_{\langle 3,0\rangle}\\ [2 & \underline{3} & \underline{3}]_{\langle 1,0\rangle} \end{bmatrix} \right\} \} \overset{C}{\Rightarrow}$$

$$\begin{bmatrix} [\underline{1} & \underline{1} & 3]_{\langle 2,1\rangle}\\ [0 & 1 & 3]_{\langle 2,0\rangle}\\ [2 & \underline{3} & \underline{3}]_{\langle 1,0\rangle} \end{bmatrix}$$

$$\overset{C}{\Rightarrow} \{2_{\langle 3,2,0\rangle}, [l \quad l \quad - \quad -], \begin{Bmatrix} \{[l & - & l & -], [- & - & l & l]\} \\ \{[- & l & l & -], [- & - & l & l]\} \end{Bmatrix}\} =$$

$$= \{2_{\langle 3,2,0\rangle}, 2_{\langle 3,2\rangle}, \begin{Bmatrix} \{1_{\langle 3,1\rangle}, 3_{\langle 1,0\rangle}\} \\ \{1_{\langle 2,1\rangle}, 3_{\langle 1,0\rangle}\} \end{Bmatrix}\} =$$

$$= \{2_{\langle 3,2,0\rangle}, 2_{\langle 3,2\rangle}, 3_{\langle 1,0\rangle}, \begin{Bmatrix} 1_{\langle 3,1\rangle} \\ 1_{\langle 2,1\rangle} \end{Bmatrix}\} = \{010_{\langle 3,2,0\rangle}, 10_{\langle 3,2\rangle}, 11_{\langle 1,0\rangle}, \begin{Bmatrix} 01_{\langle 3,1\rangle} \\ 01_{\langle 2,1\rangle} \end{Bmatrix}\} \overset{F_R^*}{\Rightarrow}$$

$$\overset{F_R^*}{\Rightarrow} \{(01-0), (10--), (--11), \begin{Bmatrix} (0-1-) \\ (-01-) \end{Bmatrix}^1 =$$

$$= \{\{4, 6\}, \{8, 9, 10, 11\}, \{3, 7, 11, 15\}, \begin{Bmatrix} \{2, 3, 6, 7\} \\ \{2, 3, 10, 11\} \end{Bmatrix}^1 .$$

У підсумку одержимо

$$Y^1 = \{(01-0), (10--), (--11), \begin{Bmatrix} (0-1-) \\ (-01-) \end{Bmatrix}^1 =$$

$$= \{\{4, 6\}, \{8, 9, 10, 11\}, \{3, 7, 11, 15\}, \begin{Bmatrix} \{2, 3, 6, 7\} \\ \{2, 3, 10, 11\} \end{Bmatrix}^1.$$

Запропонований *R*-формат подання кон’юнктермів дає змогу замінити псевдотрійкові зображення числами, що спрощує процес пошуку кон'юнктермів копій на векторах матриці розчеплення.

2.1.2 Подання кон’юнктермів парами чисел

Порівняно із псевдотрійковим (символьним) поданням кон’юнктермів *R*-формат (числове подання) є зручнішим для виконання обчислень, але все ще вимагає значних витрат комп'ютерної пам'яті. Хоча вже зберігаються тільки значимі літерали кон'юнктерма, але все ж у вигляді кортежу чисел. Ідея подання кон’юнктермів у числовому вигляді набула подальшого розвитку у [97, 106], де запропоновано числове подання кон’юнктермів у вигляді маскового зображення.

Маскове зображення — це пара чисел, а саме, числове зображення маски літералів і числове зображення кон'юнктерма з нулями замість символів поглинання "–".

За основу беремо псевдотрійкове зображення кон’юнктерма деякого рангу r (r = 0,1,2,...,n), у якому відображаємо кожну булову зміну x_i у вигляді окремої позиції a_i ($a_i \in \{0,1,-\}$). Замінимо у псевдотрійковому зображенні нулі одиницями, риски “–” нулями, а одиниці залишимо без змін. Одержали маску значимих позицій кон'юнктерма *M*. Наприклад, псевдотрійковому зображенню $(01-01-)$ відповідає маска значимих позицій *M* =110110. Таку маску можна вважати двійковим числом. Оскільки інформація про поглинуті біти уже збережена у масці, у псевдотрійковому поданні можна замінити символи поглинання нулями. Одержаний у такий спосіб набір нулів і одиниць так само вважатимемо двійковим числом. Цю пару чисел будемо називати масковим зображенням кон'юнктерма. Наприклад, псевдотрійкове подання $(01-01-)$ у масковому зображенні виглядатиме так $(010010)_{110110} = (18)_{54}$

Розглянемо приклад перетворення псевдотрійкового зображення кон’юнктерма $(01-1-)$ у маскове. У псевдотрійковому зображенні кон’юнктерма замінимо нулі “0” одиницями “1”, символи поглинання (риски “ – ”) замінимо нулями “0”. Одержимо двійкове число 11010. Це буде двійкове подання маски літералів заданого кон’юнктерма. У цій масці літералів одиниці стоять на тих позиціях, на яких у кон’юнктермі є нуль чи одиниця. Нулі у масці

літералів стоять на тих позиціях, на яких у кон’юнктермі є символи поглинання “ – ” (рисунок 2.3).

Кон’юнктерм	(0 1 – 1 –)
Маска літералів	[*l* *l* – *l* –]
Двійкова маска літералів	1 1 0 1 0
Маскове зображення кон’юнктерма	(0 1 0 1 0) $_{11010}$

Рисунок 2.3 – Формування маскового зображення кон’юнктерма

Друге число — двійкове зображення кон’юнктерма — одержуємо заміною у псевдотрійковому зображення кон’юнктерма риски “–” нулями “0”. Одержуємо двійкове число 01010. Як результат маскове зображення вихідного кон’юнктерма є числове зображення із маскою $(01010)_{11010}$. Переводимо числа з двійкової систему в десяткову і одержуємо $(10)_{26}$.

Для порівняння кон’юнктермів “вручну” зручніше перетворювати двійкові числа у вісімкові чи шістнадцяткові. Для наведеного на рисунку 2.3 прикладу одержимо у вісімковій системі числення маскове зображення (12_{32}), а в шістнадцятковій — (A_{1A}).

При здійсненні пошуку склеювань лише між кон’юнктермами, що мають однаковий набір поглинутих бітів, немає необхідності зберігати для кожного кон'юнктерма обидва числа його маскового зображення. Спільну маску літералів зберігатимемо як одне число для усієї розглянутої множини і називатимемо маскою літералів множини кон'юнктермів.

Маскове зображення кон’юнктермів використано для удосконалення методу розчеплення кон’юнктермів [104].

2.2 Процедура гіпотетичного синтезування кон’юнктермів

У [97, 106] розроблено процедуру гіпотетичного синтезування кон’юнктермів. За допомогою цієї процедури можна без проміжних склеювань шукати кон'юнктерми довільних рангів заданої булової функції. Це дає змогу уникнути появи тавтології в процесі пошуку простих кон'юнктермів. Як наслідок зменшуються обчислювальні витрати у задачах мінімізації булових функцій.

У цій процедурі використовується маскове подання кон'юнктерма. Для заданої булової функції f від n змінних розіб'ємо множину кон'юнктермів усіх рангів на підмножини з однаковими масками літералів. Позначатимемо ці підмножини S_M , де M — маска значимих позицій кон'юнктерма. Називатимемо їх *масковими множинами*.

Два кон’юнктерми булової функції можуть склеїтися лише за умови, що вони відрізняються вмістом одного значимого біту, що випливає із закону склеювання. Це означає що склеювання можливо лише між двома кон'юнктермами однієї маскової множини. Тобто у цих кон'юнктермів спільна маска літералів. При склеюванні пар кон'юнктермів деякої маскової множини рангу $(r + 1)$ з'являються кон'юнктерми меншого рангу r. На цьому кроці може утворитися $(r + 1)$ маскових множин рангу r. Для одержання маски нової множини достатньо замінити відповідну одиницю в масці M на нуль, а значить таких масок може бути $(r + 1)$. Маскову множину, яку ми використовуємо для утворення кон'юнктермів нижчих рангів називаємо *твірною масковою множиною*. *Похідна маскова множина* складається з кон'юнктермів нижчого рангу, що утворені з твірної маскової множини і мають спільну маску літералів. З іншого боку похідна множина рангу r є твірною для пошуку похідних множин рангу $(r – 1)$. Оскільки кон'юнктерми твірної множини рангу $(r + 1)$ містяться в усіх утворених із неї кон'юнктермах рангів менших за $(r + 1)$, вона є твірною для маскових підмножин, що утворюють ці кон’юнктерми нижчих рангів, а ці підмножини є, відповідно, її похідними масковими множинами.

Класичні підходи до мінімізації булових функцій викликають появу тавтології. Розглянемо причини її появи у класичних методах мінімізації булової функції від n змінних. Розіб'ємо множину кон'юнктермів на маскові множини. Кожен кон'юнктерм певного рангу r (від 0 до n-1), що входить до маскової множини S_M, може бути поданий множиною $2(n - r)$ кон'юнктермів рангу $(r + 1)$, з яких можна одержати розглянутий кон'юнктерм шляхом склеювання. Це є $(n - r)$ пар рангу $(r + 1)$. Наведемо приклад. Кон'юнктерм рангу 2 $(10 - - -)$ може бути поданий множиною кон'юнктермів рангу 3:

$$\{(10 - -0), (10 - -1), (10 - 0 -), (10 - 1 -), (100 - -), (101 - -)\}^1.$$

Ця множина розбивається на три пари кон'юнктермів:

$$(10 - -0) \vee (10 - -1) = (10 - - -),$$

$$(10 - 0 -) \vee (10 - 1 -) = (10 - - -),$$

$$(100 - -) \vee (101 - -) = (10 - - -).$$

Таким чином множина S_M рангу r є похідною для різних твірних маскових множин рангу $(r+1)$. І таких твірних маскових множин є $(n - r)$, а значить попарним склеюванням можна одержати стільки ж копій одного і того ж кон'юнктерма. Отже, якщо шукати усі можливі попарні склеювання, то це призведе до того, що у множині результатів склеювання кожен кон'юнктерм з'явиться $(n - r)$ разів.

Перешкодити появі тавтології можна у відомий спосіб. Для кон'юнктермів заданої функції створюємо спеціальний масив. У кожній комірці масиву розміщуємо числове зображення кон'юнктерма. Зв'язуємо числове зображення із адресою комірки за допомогою функції гешування. Коли з'являються однакові кон'юнктерми, вони потрапляють в одну і ту саму комірку. Таким чином

тавтологія не виникає. Оцінимо обсяги пам'яті потрібні такому масиву для мінімізації булової функції. Оскільки на кожному кроці опрацьовуємо повне кон’юнктермове поле кожного рангу *r* нам знадобиться $2^r \frac{n!}{r!(n-r)!}$ комірок. І це є дуже великий обсяг пам'яті, який можна було би використати раціональніше. При цьому не розв’язується інша проблема, а саме все одно витрачається машинний час на обчислення кон'юнктермів-копій. Тобто кожен кон’юнктерм будемо одержувати (*n* – *r*) разів.

У зв’язку із вище зазначеним запропоновано спосіб гіпотетичного пошуку кон'юнктермів кожен кон'юнктерм одержують лише одним склеюванням. А це у свою чергу усуває появу тавтології. Для цього введемо поняття гіпотетичного синтезування кон’юнктерма.

Розглядаємо булову функцію *f* від *n* змінних. Шукаємо кон’юнктерм рангу *r*, що містить деякий кон’юнктерм *K* рангу (*r*+*t*) заданої функції:

$$K = \left(a_{n-1} \cdots a_{j+t+1} a_{j+t} \cdots a_j a_{j-1} \cdots a_1 a_0\right), \qquad (2.3)$$

де $a_i \in \begin{cases} \{0,1,-\}, \text{ для } & i = 0,1,\ldots,j-1,j+t+1,\ldots,n-1 \\ \{0,1\}, \text{ для } & i = j,j+1,\ldots,j+t-1 \end{cases}$,

n, *j*, *t* — цілі невід’ємні числа.

Оскільки існує кон’юнктерм *K*, можна припустити (висунути гіпотезу), що у розглянутій функції є також кон’юнктерм рангу *r*, який має вигляд $\left(a_{n-1} \ldots a_{j+t} - \ldots - a_{j-1} \ldots a_2 a_1 a_0\right)$, у якого порівняно з *K* поглинуті змінні з індексами *j*,*j*+1,...,*j*+*t*-1. Тобто *K* міститься в цьому “уявному” кон’юнктермі.

Кон’юнктерм, який ми одержали внаслідок припущення, будемо називати гіпотетичним кон'юнктермом. А кон’юнктерм *K*, на основі якого висувалось припущення, будемо називати твірним кон’юнктермом функції *f*. Коли наше припущення відносно гіпотетичного кон'юнктерму справдилося, тобто він існує у цій функції, то вважатимемо його істинним. Якщо ж наше припущення є хибним, кон'юнктерм відсутній у заданій функції, то вважатимемо гіпотетичний кон’юнктерм хибним. Гіпотетичне синтезування кон'юнктерма – це процедура

синтезування гіпотетичного кон'юнктерма булової функції та його перевірки (істинний він чи хибний). Для того, щоб синтезувати гіпотетичний кон'юнктерм певного рангу r на основі будь-якого твірного кон'юнктерма рангу (r + t), достатньо замінити символами поглинання t значимих бітів твірного. Наприклад, для твірного $(01--0)$ при $t = 2$ можна синтезувати такі кон'юнктерми: $(----0)$, $(-1---)$, $(0----)$. У разі застосування маскового зображення процедура гіпотетичного синтезування кон'юнктерма набуває вигляду побітової кон'юнкції між двійковим зображенням твірного і маскою похідної множини. Надалі використовуватимемо символом "&" для позначення операції побітової кон'юнкції двох двійкових чисел. До прикладу твірний кон'юнктерм $(01--0)$ у масковому зображенні набуває вигляду $(01000)_{11001}$. З використанням операції побітової кон'юнкції одержимо гіпотетичні кон'юнктерми у масковому зображенні:

$$(01000\&00001)_{00001} = (00000)_{00001},$$

$$(01000\&01000)_{01000} = (01000)_{01000},$$

$$(01000\&10000)_{10000} = (00000)_{10000}.$$

Умова істинності гіпотетичного кон'юнктерма. Для деякої булової функції розглядаємо маскову множину S_M рангу (r + t). Синтезуємо гіпотетичні кон'юнктерми рангу r. Якщо деякий гіпотетичний кон'юнктерм можна синтезувати на основі 2^t різних кон'юнктермів твірної маскової множини, то це істинний гіпотетичний кон'юнктерм, якщо ні — то він хибний.

Доведення. Нехай маємо деяку булову функцію від n змінних. На першому кроці приймемо $t = 1$. У цьому випадку в одній масковій множині рангу (r + 1) можна сформувати гіпотетичні кон'юнктерми рангу r $(a_{n-1}\dots a_{j+1} - a_{j-1}\dots a_1 a_0)$ тільки на основі кон'юнктермів виду $(a_{n-1}\dots a_{j+1} a_j a_{j-1}\dots a_1 a_0)$, де $a_j \in \{0,1\}$. А значить твірних кон'юнктермів є не більше двох, а саме: $(a_{n-1}\dots a_{j+1} 0 a_{j-1}\dots a_1 a_0)$ та $(a_{n-1}\dots a_{j+1} 1 a_{j-1}\dots a_1 a_0)$. Бо

тільки вони можуть склеїтись по змінній з індексом j і утворити заданий гіпотетичний кон'юнктерм рангу r:

$$\begin{gathered}\left(a_{n-1}\dots a_{j+1}0a_{j-1}\dots a_1a_0\right)\vee\left(a_{n-1}\dots a_{j+1}1a_{j-1}\dots a_1a_0\right)=\\=\left(a_{n-1}\dots a_{j+1}-a_{j-1}\dots a_1a_0\right)\end{gathered}\qquad(2.4)$$

У такому випадку наявність двох різних твірних є достатньою підставою, щоб вважати гіпотетичний кон'юнктерм істинним. Якщо неможливо синтезувати гіпотетичний кон'юнктерм $\left(a_{n-1}\dots a_{j+1}-a_{j-1}\dots a_1a_0\right)$ на основі різних твірних кон'юнктермів, тоді це означає, що у заданій буловій функції немає принаймні одного з двох розглянутих твірних: $\left(a_{n-1}\dots a_{j+1}0a_{j-1}\dots a_1a_0\right)$ чи $\left(a_{n-1}\dots a_{j+1}1a_{j-1}\dots a_1a_0\right)$. Відповідно гіпотетичний кон'юнктерм є хибним, а гіпотезу про його існування заперечено. Цим доведено, що умова істинності гіпотетичного кон'юнктерма справджується при $t=1$.

Якщо розглянута умова справедлива для деякого натурального t, у цьому випадку для заданої булової функції від n змінних гіпотетичний кон'юнктерм $\left(a_{n-1}\dots a_{j+t}0-\dots-a_{j-1}\dots a_1a_0\right)$ рангу r є істинним за умови, що існує 2^t різних твірних кон'юнктермів виду $\left(a_{n-1}\dots a_{j+t}0a_{j+t-2}\dots a_ja_{j-1}\dots a_1a_0\right)$, ранг кожного з них дорівнює (r+t), і всі вони містяться в одній масковій множині розглянутої функції. При цьому $a_i\in\{0,1\}$, для $i=j,j+1,\dots,j+t-1$.

Гіпотетичний кон'юнктерм $\left(a_{n-1}\dots a_{j+t}1-\dots-a_{j-1}\dots a_1a_0\right)$ рангу r, що відмінний від попереднього станом змінної з індексом (j+t-1), буде істинним, якщо в тій самій масковій множині для нього існує 2^t твірних кон'юнктермів $\left(a_{n-1}\dots a_{j+t}1a_{j+t}\dots a_ja_{j-1}\dots a_1a_0\right)$ рангу (r+t). Тоді ці кон'юнктерми рангу r можна склеїти по змінній з індексом (j+t-1) і одержати кон'юнктерм рангу (r-1):

$$\begin{gathered}\left(a_{n-1}\dots a_{j+t}0-\dots-a_{j-1}\dots a_1a_0\right)\vee\left(a_{n-1}\dots a_{j+t}1-\dots-a_{j-1}\dots a_1a_0\right)=\\=\left(a_{n-1}\dots a_{j+t}---\dots-a_{j-1}\dots a_1a_0\right).\end{gathered}\qquad(2.5)$$

Вихідна множина містить $2^t \cdot 2 = 2^{t+1}$ кон'юнктермів рангу (r+t), що є твірними по відношенню до одержаного.

Якщо був би відсутній у твірній масковій множині хоча б один з цих 2^{t+1} гіпотетичних кон'юнктермів, тоді хибним був би хоч один з гіпотетичних кон'юнктермів рангу r, а це заперечувало б існування кон'юнктерма $(a_{n-1}\dots a_{j+t} - -\dots - a_{j-1}\dots a_1 a_0)$.

У цьому випадку можна стверджувати, що умова істинності гіпотетичного кон'юнктерма справджується для t+1. Тоді, згідно з аксіомою про математичну індукцію, ця умова справджується для будь-якого натурального t. Доведення завершено.

Наочно умову істинності гіпотетичного кон'юнктерма можна пояснити так. Для формування гіпотетичного кон'юнктерма $(a_{n-1}\dots a_{j+t} - \dots - a_{j-1}\dots a_1 a_0)$ рангу r певної булової функції від n змінних одна маскова множина рангу (r+t) може містити тільки твірні кон'юнктерми виду $(a_{n-1}\dots a_{j+t} a_{j+t-1}\dots a_j a_{j-1}\dots a_1 a_0)$. Ці твірні відмінні по t змінних $a_j, a_{j+1}, \dots, a_{j+t-1}$, де $a_i \in \{0,1\}$ при $i = j, j+1, \dots, j+t-1$. Набір решти змінних у них однаковий. А значить у цій множині може перебувати твірних кон'юнктермів не більше ніж 2^t. А якщо можна синтезувати один і той самий гіпотетичний кон'юнктерм з 2^t відмінних між собою твірних кон'юнктермів, що належать одній масковій множині, то у розглянутій буловій функції є присутні усі кон'юнктерми відповідного виду. І їх можна склеїти по змінних $a_j, a_{j+1}, \dots, a_{j+t-1}$. При цьому утвориться кон'юнктерм $(a_{n-1}\dots a_{j+t} - \dots - a_{j-1}\dots a_1 a_0)$. Тож останній є істинним кон'юнктермом.

Ще до початку процедури формування гіпотетичних кон'юнктермів можна визначити чи буде порожньою утворена внаслідок гіпотетичного синтезування маскова множина (*похідна маскова множина*). Це дасть змогу зменшити часові затрати на гіпотетичне синтезування кон'юнктермів. Розглянемо умову заперечення похідних маскових множин.

Умова заперечення похідних маскових множин. Якщо у буловій функції від n змінних у деякій масковій множині S_M рангу $(r + t)$ кількість кон'юнктермів є меншою за 2^t, то всі її похідні маскові множини рангу r і нижче будуть порожні.

Доведення. Побітова кон'юнкція між маскою похідної множини та двійковим зображенням твірного в єдиний спосіб визначає гіпотетичний кон'юнктерм. Звідси, у кожній похідній множині на основі одного твірного кон'юнктерма є можливість сформувати один єдиний гіпотетичний кон'юнктерм. Якщо взяти деяку твірну маскову множину S_M рангу $(r+t)$, у якій є менше за 2^t твірних кон'юнктермів, то буде неможливо синтезувати більше за $(2^t - 1)$ гіпотетичних кон'юнктермів для похідної множини рангу r. А значить згідно з умовою істинності будуть хибними усі синтезовані гіпотетичні кон'юнктерми. Як наслідок її похідна множина буде порожня. Зрозуміло, що без твірних не синтезувати гіпотетичний кон'юнктерм, відповідно, похідні множини від порожньої теж будуть порожні. А значить, від S_M усі похідні множини рангу r і нижче будуть порожні. Доведення завершено.

Кількість одиниць у кон'юнктермі називатимемо його вагою. Розглянемо розбиття маскової множини S_M на підмножини за вагою кон'юнктерма

$$S_M = \{S_M^0, S_M^1, \ldots, S_M^n\}. \qquad (2.6)$$

Числом у верхньому індексі позначаємо вагу кон'юнктермів підмножини.

Умова розбиття твірної множини. Якщо у твірній масковій множині S_M відсутні кон'юнктерми із деякою вагою p, то множину S_M можна розбити на два незалежні набори підмножин $\{S_M^0, S_M^1, \ldots, S_M^{p-1}\}$ та $\{S_M^{p+1}, S_M^{p+2}, \ldots, S_M^n\}$. Причому між собою не склеюються кон'юнктерми похідних множин з різних наборів.

Доведення. Згідно із законом склеювання попарні склеювання кон'юнктермів однакового рангу r можливо здійснити між елементами підмножин з різними вагами, якщо ваги кон'юнктермів відрізняються на одиницю. Отже, за відсутності кон'юнктермів з певною вагою p твірна маскова

множина S_M розбивається на два набори підмножин $\{S_M^0, S_M^1, \ldots, S_M^{p-1}\}$ та $\{S_M^{p+1}, S_M^{p+2}, \ldots, S_M^n\}$ так, що неможливо склеїти кон'юнктермами з різних наборів. Також з умови істинності гіпотетичних кон'юнктермів випливає, що у похідній множині ваги p може існувати кон'юнктерм рангу (r - 1) тільки за умови наявності обох твірних. За означенням гіпотетичного кон'юнктерма ці твірні мають ваги p та (p + 1). З початкової умови відомо, що відсутні твірні з вагою p. Відповідно, відсутня підмножина з вагою p у похідній множині рангу (r - 1). Так само, необхідною умовою існування кон'юнктерма рангу (r - 1) у похідній множині ваги (p – 1) є наявність твірних з вагами (p – 1) та p. Як наслідок не може бути підмножини з вагою (p – 1) у похідній множині рангу (r - 1). Через те, що немає кон'юнктермів з вагами (p – 1) та p, похідні кон'юнктерми рангу (r - 1) від різних наборів $\{S_M^0, S_M^1, \ldots, S_M^{p-1}\}$ та $\{S_M^{p+1}, S_M^{p+2}, \ldots, S_M^n\}$ не можуть склеюватися між собою. У свою чергу похідні рангу (r - 1) є твірними для множин рангу (r - 2). Звідси можна зробити висновок, що не склеюються між собою похідні кон'юнктерми усіх нижчих рангів, що одержані від різних наборів $\{S_M^0, S_M^1, \ldots, S_M^{p-1}\}$ та $\{S_M^{p+1}, S_M^{p+2}, \ldots, S_M^n\}$. Доведення завершено.

Потрібно зауважити, якщо немає кон'юнктермів з деякою вагою у твірній множині, то з умови розбиття твірної множини випливає, що задача гіпотетичного синтезування кон'юнктермів розбивається на два окремі завдання. Як наслідок зменшуються вимоги до обчислювальних ресурсів комп'ютера на окремих етапах синтезу кон'юнктермів.

Процедура гіпотетичного синтезування кон'юнктермів. Розглянемо булову функцію від n змінних. У ній є твірна маскова множина S_M. Її маска значимих позицій кон'юнктерма:

$$M = m_{n-1}\ldots m_{j+t+1}1\ldots 1 m_{j-1}\ldots m_2 m_1 m_0, \qquad (2.7)$$

де m_i – це біти коду маски (i = 0, 1, 2, ... , j – 1 , j + t + 1, … , n - 1).

Задаємо t як різницю рангів твірної і похідної маскових множин.

Знайти похідну маскову множину

$$S_{m_{n-1}\ldots m_{j+t+1}0\ldots 0 m_{j-1}\ldots m_2 m_1 m_0}. \quad (2.8)$$

Етапи процедури гіпотетичного синтезування кон'юнктермів:

1) перевірити умову заперечення похідної маскової множини. (Якщо твірна маскова множина складається менше ніж з 2^t кон'юнктермів, то похідна маскова множина є порожня);

2) якщо похідна маскова множина не є порожня, розбити твірну маскову множину на підмножини за вагами кон'юнктермів: $S_M = \{S_M^0, S_M^1, \ldots, S_M^n\}$. Розташувати одержані підмножини за наростанням ваг;

3) якщо відсутні підмножини з деякою вагою, то розбити розглянуту задачу на незалежні завдання гіпотетичного синтезування кон'юнктермів;

4) якщо є підмножина з деякою вагою p, що немає підмножин з вагами $(p-1)$ та $(p+1)$, то виключити цю підмножину з подальшого розгляду;

5) починаємо з найменшої ваги. На основі кожної підмножини синтезуємо гіпотетичні кон'юнктерми, виконуючи побітову кон'юнкцію між двійковим зображенням твірного кон'юнктерма та маскою значимих позицій похідної множини $m_{n-1}\ldots m_{j+t+1}0\ldots 0 m_{j-1}\ldots m_2 m_1 m_0$. Якщо збігаються двійкові зображення гіпотетичного та твірного кон'юнктермів, то записуємо гіпотетичний кон'юнктерм у підмножину похідної маскової множини, що має вагу таку саму як вага твірного, тобто p. Вказуємо біля похідного кон'юнктерма кількість синтезованих копій („1"). Якщо двійкові зображення гіпотетичного та твірного кон'юнктермів не збігаються, то у похідних множинах з вагами від $(p-t)$ по $(p-1)$ шукаємо двійкове зображення, що збігається з гіпотетичним кон'юнктермом. Якщо знаходимо, то збільшуємо на один кількість копій похідного кон'юнктерма;

6) виписуємо з множини гіпотетичних кон'юнктермів такі, що мають 2^t копій. Вони є істинні.

Проілюструємо на прикладі процедуру гіпотетичного синтезування кон'юнктермів.

Приклад 2.2

Досконалою ТМФ задано твірну маскову множину булової функції від чотирьох змінних:

$$Y^1 = \{2,4,8,3,6,9,10,11,7,15\}^1 .$$

Знайти похідну маскову множину S_{0011}.

Розв'язання.

$Y^1 = S_{1111} = \{(0010), (0100), (1000), (0011), (0110), (1001), (1010), (1011), (0111), (1111)\}^1{}_{1111}$

Кон'юнктерми твірної маскової множини S_{1111} мають ранг 4. Кон'юнктерми шуканої похідної маскової множини S_{0011} мають ранг 2. Різниця їх рангів складає $t = 2$. У твірній масковій множині є десять кон'юнктермів. Це більше за 2^t. Таким чином, не виконується умова заперечення похідної маскової множини S_{0011}.

За вагами кон'юнктермів розіб'ємо твірну маскову множину S_{1111} на підмножини. Розташуємо ці підмножини за наростанням ваг:

$$S^1_{1111} = \{(0010), (0100), (1000)\}_{1111};$$

$$S^2_{1111} = \{(0011), (0110), (1001), (1010)\}_{1111};$$

$$S^3_{1111} = \{(1011), (0111)\}_{1111};$$

$$S^4_{1111} = \{(1111)\}_{1111}\ .$$

За маскою 0011 на основі твірних кон'юнктермів підмножини S^1_{1111} синтезуємо гіпотетичні кон'юнктерми:

$$(0010\&0011)_{0011} = (0010)_{0011}\ .$$

Двійкове зображення твірного кон'юнктерма збігається з двійковим зображенням гіпотетичного кон'юнктерма. Записуємо гіпотетичний кон'юнктерм у множину S^1_{0011}. Зазначимо над ним „1", як кількість копій:

$$S^1_{0011} = \left\{(\overset{1}{0010})\right\}_{0011}.$$

Для твірного кон'юнктерма $(0100)_{1111}$ одержимо похідний кон'юнктерм $(0000)_{0011}$. Їх двійкові зображення не збігаються, тому розглядаємо підмножини S_{0011}, що мають ваги від (1 - *t*) по 0. У них його копій немає. Отже, беремо для розгляду наступний твірний кон'юнктерм.

Унаслідок синтезу всіх гіпотетичних кон'юнктермів, що мають маску 0011 одержимо:

$$S^1_{0011} = \left\{(\overset{3}{0010})\right\}_{0011}\ ,\ \ S^2_{0011} = \left\{(\overset{4}{0011})\right\}_{0011}\ .$$

За умовою істинності гіпотетичних кон'юнктермів істинними є ті, кількість копій яких дорівнює 4. Вибираємо істинні з одержаної множини гіпотетичних кон'юнктермів. Одержимо шукану множину:

$$S_{0011} = \{(0011)\}_{0011} = \{(--11)\} = \{(3,7,11,15)\}.$$

Таким чином, у заданій функції присутній тільки один кон’юнктерм із маскою 0011: (3,7,11,15).

Із застосуванням процедури гіпотетичного синтезування кон’юнктермів відносно просто будується власне кон’юнктермове поле булової функції, яке для певної функції являє собою набір усіх маскових множин і є необхідне для пошуку скороченої диз’юнктивної нормальної форми булових функцій методом кон’юнктермового поля. Оскільки на кожному кроці синтезуємо тільки одну похідну маскову множину певного рангу r, то потрібно не більше 2^r комірок пам’яті для зберігання синтезованих кон’юнктермів. То є значно менше від того, що необхідно класичним методам, які резервують пам’ять під усі кон’юнктерми рангу r.

До переваг процедури гіпотетичного синтезування можна віднести такі:

1) простота реалізації засобами обчислювальної техніки;
2) відсутність тавтології;
3) кожен кон’юнктерм синтезується одним склеюванням;
4) можливість пошуку кон’юнктермів довільного рангу без здійснення проміжних склеювань;
5) відносно невисокі потреби в оперативній пам’яті.

Ці переваги дають змогу знизити потребу в обчислювальних ресурсах у задачах мінімізації булових функцій.

2.3 Удосконалення методу побітового сортування двійкових чисел

Вхідними даними для методу побітового вирощування простих кон’юнктермів є множина мінтермів, що не є мультимножиною. На першому кроці необхідно сортувати множину мінтермів за зростанням їх двійкових кодів.

Для цього обрано та удосконалено метод побітового сортування множини цілих чисел [99, 105] для випадку, коли задана множина не є мультимножиною. Запропоновано в процесі сортування за певним бітом обчислювати потужність отриманих підмножин. Це дає змогу виявити чи склеюється підмножина в один кон'юнктерм, у якого поглинуті всі біти, молодші від біта сортування.

При опрацьовуванні даних часто доводиться їх упорядковувати за деякою ознакою. Вже відомі різні алгоритми сортування загального призначення. Але ці методи не враховують специфіки прикладної задачі мінімізації булових функцій. Врахування такої специфіки може допомогти зменшити обчислювальні витрати на розв’язання задач.

У задачах мінімізації булових функцій доводиться оперувати скінченними множинами. Позначимо множину усіх натуральних чисел через N. За означенням скінченної множини, якщо існує взаємно однозначне відображення множини A на підмножину $N_m \subset N$: $N_m = \{x | x \in N, x \leq m\}$ (де m – деяке натуральне число) чи множина A є порожня, то A є скінченною множиною. Таким чином елементи множини A можна пронумерувати m натуральними числами. Ми будемо розглядати випадок, коли A не містить тавтології (тобто не є мультимножиною). Крім того упорядкованою вважатимемо єдину послідовність її елементів, а саме $A = \langle a_1, a_2, a_3, \ldots, a_{m-1}, a_m \rangle$. З іншого боку ця скінченна послідовність є взаємно однозначним відображенням (тобто функцією) f_A:

$$\{1,2,3,\ldots,(m-1),m\} \rightarrow \{a_1, a_2, a_3, \ldots, a_{m-1}, a_m\}, \tag{2.9}$$

$$f_A(i) = a_i \tag{2.10}$$

де i = 1, 2, 3, …, (m–1), m.

Позначимо через B множину m невід’ємних цілих чисел, що впорядковані за зростанням. Тож, множину B так само можна подати взаємно однозначним відображенням f_B:

$$\{1,2,3,\dots,(m-1),m\} \to \{b_1, b_2, b_3, \dots, b_{m-1}, b_m\}, \quad (2.11)$$

$$f_B(i) = b_i \quad (2.12)$$

де i = 1, 2, 3, …, (m-1), m.

Тож, f_A є взаємно однозначним відображенням $N_m \to A$, а f_B є взаємно однозначним відображенням $N_m \to B$. А значить також взаємно однозначним є відображення $B \to A$, що фактично є суперпозицією оберненого до f_B відображення та f_A, тобто $f_B^{-1} \circ f_A$. Звідси випливає, що задачу упорядкування даних можна розглядати як задачу сортування невід'ємних цілих чисел в лінійному списку і навпаки. З одного боку це має практичне значення у задачах мінімізації булових функцій, коли виникає потреба упорядковувати мінтерми [107]. З іншого боку, ми можемо використати математичний апарат булової алгебри для задачі сортування множини чисел, якщо будемо інтерпретувати числа як вершини булового простору, тобто мінтерми.

Нехай B список цілих невід'ємних чисел, що не містить тавтології. Нашим завданням є розставити його елементи у такому порядку

$$B = \langle k_0, k_1, \dots k_{m-1} \rangle, \quad (2.13)$$

щоб справджувалася умова

$$k_{i_g} < k_{i_{g+1}} \quad (2.14)$$

для довільного невід'ємного цілого числа g меншого від m ($0 \le g \le (m-1)$).

Частина методів сортування (обмінне, вставкою, вибором тощо) хоч і реалізовуються простими алгоритмами, але є повільні. Інші методи сортування (злиттям, розподільне, швидке, квадратичним вибором) мають вищу швидкодію, але потребують додаткової пам'яті.

У [99, 105] запропоновано модифікацію методу низхідного побітового сортування, що є однією з варіацій позиційного сортування. Розглядаємо B – лінійний список m невід'ємних n-бітних чисел, які необхідно упорядкувати за зростанням.

З позицій булової алгебри список B задає булову функцію від n змінних, де кожен елемент списку це мінтерм, а кожен біт мінтерма відображає пряме або інверсне входження відповідної змінної у цей мінтерм.

Розглянемо для такого списку побітове сортування зі склеюванням. Нехай найбільше ціле число, з тих що потрібно посортувати, має n біт у двійковому поданні. Вважатимемо числа мінтермами функції від n змінних. Сортування починаємо зі старшого біта.

Отже, здійснюємо сортування чисел за старшим бітом. Отримаємо дві підмножини. В одній усі числа з нулем у біті сортування, у другій — з одиницею. Під час сортування за наступним бітом опрацьовуємо ці підмножини незалежно одна від одної, можна навіть у паралельних процесах.

У процесі сортування виконуємо підрахунок елементів кожної з одержаних підмножин (обчислюємо потужність кожної підмножини). Якщо кількість кон'юнктермів у підмножині дорівнює 2^j, де j — номер біта, по якому здійснюється сортування, то у цій підмножині всі кон'юнктерми склеюються. В утвореному кон'юнктермі поглинуті всі біти, молодші від j. Замінюємо підмножину кон'юнктермом з поглинутими молодшими бітами. Утворені склеюванням кон'юнктерми містять підмножину вихідних чисел, що вже зайняла своє місце і не потребує подальшого сортування за молодшими бітами. Продовжуємо сортування іншої підмножини за молодшим бітом.

Після завершення сортування відновлюємо список мінтермів з кон'юнктермів, що мають поглинуті змінні. Для цього на їх місці достатньо послідовно згенерувати цілі числа від нуля до $(2^r - 1)$, де r — кількість поглинутих бітів. До кожного одержаного числа дописати старші біти, що не були поглинуті у замінюваному кон'юнктермі.

Проілюструємо процедуру склеювання під час сортування на прикладі чотирибітної маски літералів деякої початкової множини B чотирибітних чисел. На рисунку 2.4 зображено дерево побітового сортування. Літерал, що може набувати значення нуль “0” чи одиниця “1”, позначатимемо літерою l. Поглинуті біти позначатимемо рискою “–”. Тоді $l_3l_2l_1l_0$ — маска літералів початкової множини B (перший рядок на рисунку). Сортування по третьому біту дає підмножини B_0 і B_1 (другий рядок на рисунку). Множина B_0 складається з елементів множини B, що мають нуль у третьому біті. Відповідно, $0l_2l_1l_0$ — маска літералів множини B_0. Множина B_1 складається з решти елементів множини B, тобто з тих, що містять одиницю в третьому біті.

<table>
<tr><td colspan="16">$l_3l_2l_1l_0$</td></tr>
<tr><td colspan="8">$0l_2l_1l_0$
Якщо $|\boldsymbol{B}_0| = 2^3$, то існує (0 - - -)</td><td colspan="8">$1l_2l_1l_0$
Якщо $|\boldsymbol{B}_1| = 2^3$, то існує (1 - - -)</td></tr>
<tr><td colspan="4">$00l_1l_0$
(00 - -) при $|\boldsymbol{B}_{00}| = 2^2$</td><td colspan="4">$01l_1l_0$
(01 - -) при $|\boldsymbol{B}_{01}| = 2^2$</td><td colspan="4">$10l_1l_0$
(10 - -) при $|\boldsymbol{B}_{10}| = 2^2$</td><td colspan="4">$11l_1l_0$
(11 - -) при $|\boldsymbol{B}_{11}| = 2^2$</td></tr>
<tr><td colspan="2">$000l_0$
(000 -)</td><td colspan="2">$001l_0$
(001 -)</td><td colspan="2">$010l_0$
(010 -)</td><td colspan="2">$011l_0$
(011 -)</td><td colspan="2">$100l_0$
(100 -)</td><td colspan="2">$101l_0$
(101 -)</td><td colspan="2">$110l_0$
(110 -)</td><td colspan="2">$111l_0$
(111 -)</td></tr>
<tr><td>0000
0</td><td>0001
1</td><td>0010
2</td><td>0011
3</td><td>0100
4</td><td>0101
5</td><td>0110
6</td><td>0111
7</td><td>1000
8</td><td>1001
9</td><td>1010
10</td><td>1011
11</td><td>1100
12</td><td>1101
13</td><td>1110
14</td><td>1111
15</td></tr>
</table>

Рисунок 2.4 – Дерево побітового сортування зі склеюванням на прикладі чотирибітної маски літералів [99]

Відповідно, $1l_2l_1l_0$ — маска літералів множини B_1. Якщо потужність підмножини B_0 дорівнює 2^3, то вилучаємо її з подальшого розгляду. А замість неї записуємо кон’юнктерм (0 – – –). Аналогічно, B_1 замінюємо на кон’юнктерм (1 – – –), якщо її потужність дорівнює 2^3. Далі повторюємо процедуру для наступного біту.

Приклад 2.3 [99, 105]

Задано множину чисел:

{14,7,13,6,12,5,11,4,10,3,9,2,8,0,1}.

Необхідно впорядкувати числа за зростанням.

Розв’язання. Перетворимо десяткові коди у двійкові

$$\{1110,0111,1101,0110,1100,0101,1011,0100,1010,0011,1001,0010,1000,0000,0001\}$$

Виконуємо сортування за третім (старшим) бітом:

$$\{0111,0110,0101,0100,0011,0010,0000,0001\},$$
$$\{1110,1101,1100,1011,1010,1001,1000\}.$$

Підмножина молодших кодів (із нулем у третьому біті) налічує 2^3 елементів. Тому всі її мінтерми склеюються у кон’юнктерм $\{(0--)\}$, який записуємо замість цієї підмножини. І це вже буде остаточне місце у впорядкованій множині. У сортуванні за рештою бітів він участі не братиме.

$$\{(0--)\}, \{1110,1101,1100,1011,1010,1001,1000\}$$

Залишилася підмножина старших чисел. Сортуємо її за другим бітом:

$$\{(0--)\}, \{1011,1010,1001,1000\}, \{1110,1101,1100\}.$$

І так само отримали одне склеювання:

$$\{(0--)\}, \{(10-)\}, \{1110,1101,1100\}.$$

Сортуємо підмножину, що залишилася, за першим бітом:

$$\{(0--)\}, \{(10-)\}, \{1101,1100\}, \{1110\}.$$

Підмножина молодших чисел склеїлася, а підмножина старших чисел містить лише один елемент. Тож сортування завершено.

$$\{(0--\!-)\},\ \{(10--)\},\ \{(110-)\},\ \{1110\}$$

Із кон'юнктерма $\{(0--\!-)\}$ відновимо упорядковану множину мінтермів. Першим кроком згенеруємо послідовність чисел від 0 до (2^3-1) (бо поглинуто три біти):

$$\{0,1,2,3,4,5,6,7\}.$$

Що у двійковому поданні відповідає множині:

$$\{000,001,010,011,100,101,110,111\}.$$

До коду кожного елемента зліва допишемо старші біти вихідного кон'юнктерма:

$$\{0000,0001,0010,0011,0100,0101,0110,0111\}.$$

Аналогічно згенеруємо упорядкований список мінтермів для кожного кон'юнктерма, що має поглинуті біти. Одержимо:

$$\{0000,0001,0010,0011,0100,0101,0110,0111\},\ \{1000,1001,1010,1011\},$$
$$\{1100,1101\},\ \{1110\}.$$

Остаточно у десятковому поданні маємо:

$$\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}.$$

На рисунку 2.5 розв’язання прикладу 2.3 проілюстровано у десятковому поданні. У кожному рядку рисунку здійснюється сортування за одним бітом, починаючи зі старшого біта у верхньому рядку. Якщо множина не склеюється у кон’юнктерм, то відповідний кон’юнктерм на рисунку перекреслений. Таку множину сортують за наступним молодшим бітом у рядку нижче. У останньому рядку наведено відновлені з кон’юнктермів списки мінтермів у двійковому і десятковому поданні.

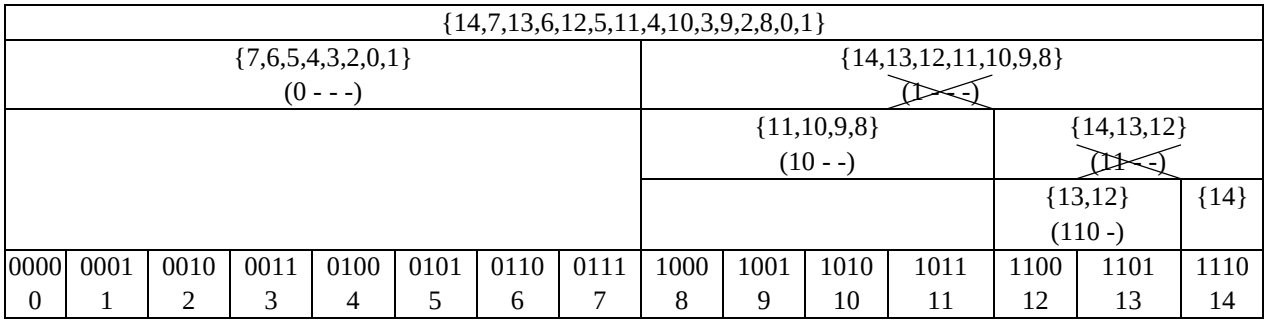


Рисунок 2.5 – Схема розв’язання прикладу 2.3 [99]

Для методу порозрядного вирощування простих кон'юнктермів булової функції необхідно упорядкувати за зростанням числових зображень вхідну множину мінтермів [107]. Застосувавши для цього запропоновану модифікацію побітового сортування зі склеюванням можна одержати деякі кон'юнктерми менших рангів без проміжних склеювань і тавтології.

2.4 Розвиток процедури спрощення множини символьних диз’юнктермів

У теоретико множинній модифікації мінімаксного методу покриття булових функцій [93] на кожному кроці покриття віднаходять один істотний простий кон’юнктерм, який записують у множину мінімального покриття. Після цього відбувається вилучення відповідного символьного диз’юнктерма та мінтермів, що містяться в істотному кон’юнктермі, з подальшого розгляду.

Наслідком цього є така зміна множини символьних диз'юнктермів, що може скластися ситуація, коли деякі з символьних диз'юнктермів містяться в інших. Тобто виконується умова

$$D_k \subseteq D_l \,, \tag{2.15}$$

а значить символьний диз'юнктерм D_l буде поглинутий символьним диз'юнктермом D_k. Відповідно, на кожному кроці покриття виникає потреба у процедурі спрощення множини символьних диз'юнктермів. Ця процедура здійснюється шляхом пошуку тих символьних диз'юнктермів, що містять інші. Такі диз'юнктерми необхідно вилучити з подальшого розгляду і в такий спосіб спростити розглянуту множину. Фактично процедура здійснюється за допомогою операції елементарного поглинання, яку необхідно виконати для кожної пари символьних диз'юнктермів. Виникає комбінаторна складність, яку можна зменшити у більшості випадків, якщо попередньо упорядкувати символьні диз'юнктерми $D_1, D_2, D_3, \ldots D_k$ за порядком наростання їх потужностей. Тоді кожен диз'юнктерм, починаючи з тих, що мають найменшу потужність, потрібно порівнювати тільки з диз'юнктермами, що мають меншу чи таку саму потужність, а не з усіма символьними диз'юнктермами. Крім того, скорочується шлях пошуку тих диз'юнктермів, що можуть бути вилучені з розгляду.

2.5 Покриття таблиці простих кон'юнктермів ланцюгової функції та фрагментів такої функції

У [98] запропоновано спосіб спрощення процедури покриття таблиці простих кон'юнктермів булової функції. Цей спосіб враховує взаємні перетини простих кон'юнктермів циклічної частини для побудови ланцюжків

кон’юнктермів. При цьому виділяється клас задач, які можна розв’язати без перебору, що характеризується комбінаторною складністю.

Задачу покриття таблиці простих кон’юнктермів булової функції можна спростити, якщо врахувати деякі закономірності булового простору. У першу чергу необхідно знайти істотні кон’юнктерми. Їх ознакою є одинична вага одного із стовпців таблиці простих кон’юнктермів, які покриває рядок розглянутого кон’юнктерма. Ізольовані кон’юнктерми необхідно перенести у результуючу множину мінімального покриття і редукувати таблицю простих кон’юнктермів, тобто вилучити відповідний рядок і покриті ним стовпці.

Вилучивши з таблиці усі ізольовані кон’юнктерми отримаємо циклічну частину. Кожен простий кон’юнктерм як множина мінтермів повністю міститься у деякій підмножині інших простих кон’юнктермів циклічної частини. Залежно від того яким кон’юнктермом покрити певний мінтерм, другий простий кон’юнктерм може бути надлишковим чи істотним. Від статусу другого може залежати статус ще декількох і так дальше. З’являється проблема перебору, що характеризується комбінаторною складністю.

Можна також спростити задачу покриття, використовуючи техніку поглинання стовпців таблиці простих кон’юнктермів [52].

У [98] запропоновано проаналізувати множину простих кон’юнктермів на предмет наявності незалежних підмножин та фрагментів ланцюгових функцій і у разі знаходження додатково спростити задачу покриття.

Розглянемо спосіб розбиття множини простих кон’юнктермів. Нехай циклічна частина описується множиною мінтермів M та множиною простих кон’юнктермів K, що покриває M. Вважаємо, що усі істотні кон’юнктерми вже вилучені, а також виконано усі можливі поглинання стовпців таблиці простих кон’юнктермів. З множини K потрібно виділити таку підмножину K_{min} , що має мінімальний кошт покриття множини мінтермів M.

Припустимо, що існує таке розбиття множини K на неперетні підмножини K_i

$$K = \bigcup_i K_i \,, \tag{2.16}$$

що серед мінтермів не існує такого, який міститься у простих кон'юнктермах різних підмножин. У цьому випадку існує таке розбиття множини мінтермів M на неперетні підмножини M_i

$$M = \bigcup_i M_i, \tag{2.17}$$

що кожна підмножина M_i буде покрита лише простими кон'юнктермами відповідної підмножини K_i . І навпаки, кожна підмножина K_i покриває тільки мінтерми підмножини M_i. Причому для будь-яких i та j ($i \neq j$) виконується умова

$$M_i \cap M_j = \emptyset. \tag{2.18}$$

У такому випадку задача покриття множини M розбивається на незалежні підзадачі покриття підмножин M_i, бо жоден кон'юнктерм одної підмножини не впливає на визначення кон'юнктерма іншої підмножини істотним чи надлишковим. Причому розмірність підзадач завжди буде меншою ніж розмірність усієї задачі.

Загальний розв'язок отримаємо шляхом об'єднання розв'язків окремих підзадач:

$$K_{\min} = \bigcup_i K_{i\,\min}. \tag{2.19}$$

Виявити розбиття множини K на підмножини, що не покривають спільних мінтермів, можна за допомогою матриці суміжностей, що використовується в

алгоритмі визначення зв’язності графа, застосувавши цей алгоритм до таблиці простих кон’юнктермів.

Розглянемо частковий випадок циклічної частини, коли кожен стовпець циклічної частини таблиці простих кон’юнктермів має вагу 2 і при цьому всі кон’юнктерми мають однаковий ранг. За таких умов кон’юнктерми циклічної частини утворюють ланцюг у двійковому просторі. При цьому рядки таблиці кон’юнктермів є ланками цього ланцюга, а стовпці таблиці є осями ланцюга. Наступним кроком потрібно упорядкувати множину кон’юнктермів за порядком слідування ланок у ланцюгу.

Алгоритм побудови ланцюга кон’юнктермів циклічної частини:

1) шукаємо рядок таблиці з вагою 1. За відсутності такого обираємо перший рядок. Присвоюємо цьому кон’юнктерму перший номер;

2) в обраному рядку починаючи зліва шукаємо одиницю, що ще не викреслена. Обираємо стовпець знайденої одиниці, а рядок при цьому викреслюємо (вилучаємо з подальшого розгляду);

3) в обраному стовпці шукаємо іншу одиницю, що ще не викреслена. Обираємо рядок знайденої одиниці. Цьому кон’юнктерму присвоюємо наступний номер, а стовпець при цьому викреслюємо (вилучаємо з подальшого розгляду);

4) якщо немає інших невикреслених рядків, то упорядкування завершено. Якщо є ще невикреслені рядки, то переходимо до пункту 2.

Покриття такої впорядкованої множини є тривіальним.

Алгоритм покриття ланцюга кон’юнктермів однакової ваги:

— якщо на першому кроці упорядкування вибрано рядок із вагою 2, то є два рівноцінні розв’язки. Перший розв’язок складається з кон’юнктермів, що мають непарні номери. У другому розв’язку всі кон’юнктерми з парними номерами;

— якщо на першому кроці упорядкування вибрано рядок із вагою 1 і при цьому циклічна частина містить непарну кількість кон’юнктермів, тоді розв’язок складається з кон’юнктермів, що мають парні номери;

— якщо на першому кроці упорядкування вибрано рядок із вагою 1 і при цьому циклічна частина містить парну кількість кон'юнктермів m, тоді існує $m/2$ розв'язків. Формування кожного розв'язку починаємо з того, що обираємо кон'юнктерм з деяким парним номером p $(1 < p \leq m)$. Наступним кроком заносимо до розв'язку всі кон'юнктерми із парними номерами k, що не більші за p $(1 < k \leq p)$. Потім заносимо до розв'язку всі кон'юнктерми із непарними номерами l, що більші за p $(p < l < m)$.

Розглянемо складніший випадок, коли кожен стовпець циклічної частини таблиці простих кон'юнктермів має вагу 2, але це кон'юнктерми різних рангів. За таких умов кон'юнктерми циклічної частини теж утворюють ланцюг у двійковому просторі. За описаним вище алгоритмом можна знайти набір тупикових диз'юнктивних нормальних форм, проте ці розв'язки можуть мати різний кошт реалізації. Мінімальним буде розв'язок з найменшою сумою рангів кон'юнктермів, що у нього входять.

Якщо ж у циклічній частині таблиці простих кон'юнктермів є стовпці з вагою більше два, то наведений вище алгоритм у такому стовпці прийде до точки розгалуження ланцюга. Необхідно прийняти рішення про розрив ланцюга. Для цього обираємо кон'юнктерм найменшої ваги з тих, що покривають розглянутий стовпець. Вважаємо обраний кон'юнктерм істотним. Таким чином формування одного ланцюга буде завершено, а покриття тих, що виходять з точки розгалуження може виконуватись у паралельних процесах. У цьому випадку не можна гарантувати точний розв'язок. Цей алгоритм є евристичний.

Процедура *ланцюгового покриття* циклічної частини таблиці простих кон'юнктермів:

— упорядкувати множину кон'юнктермів циклічної частини за порядком слідування ланок у ланцюгу;

— у точках розгалуження ланцюга прийняти рішення про розрив ланцюга;

— для кожного фрагмента ланцюга, що одержаний після розриву в точці розгалуження, знайти мінімальну диз'юнктивну нормальну форму.

Приклад 2.4

Необхідно виокремити множину мінімального покриття булової функції Y^1, що задана множиною всіх простих кон'юнктермів:

Y^1 = {(1,0), (8,0), (5,1), (10,8), (7,5), (14,10), (7,6), (14,6), (49,48), (122,58), (55,53), (53,49), (62,58), (55,54), (62,54)}[1].

Розв'язання. Складемо таблицю простих кон'юнктермів, у якій рядки відповідають простим кон'юнктермам, а стовпці — мінтермам.

Одразу вилучимо з таблиці кон'юнктерми (49,48) та (122,58), та віднесемо їх до множини істотних кон'юнктермів K_{ist}, оскільки вони є істотними. Залишається циклічна частина (рисунок 2.6).

Візьмемо перший рядок таблиці і перенесемо відповідний кон'юнктерм до множини K_1. Цей кон'юнктерм покриває мінтерми (0) та (1). У відповідних стовпцях знаходимо кон'юнктерми (8,0) та (5,1), що також покривають ці стовпці. Переносимо ці кон'юнктерми до множини K_1. Вони покривають ще й стовпці (8) та (5). Шукаємо у цих стовпцях кон'юнктерми, що їх покривають і також переносимо їх до множини K_1. Так продовжуємо формувати множину K_1, поки не обірветься «ланцюжок» чи буде вилучено всі прості кон'юнктерми. Якщо в таблиці залишились прості кон'юнктерми, то починаємо формувати другу множину K_2.

У розв'язуваній задачі отримаємо розбиття на дві підмножини простих кон'юнктермів циклічної частини, а саме:

$$K_1 = \{(1,0), (8,0), (5,1), (10,8), (7,5), (14,10), (7,6), (14,6)\},$$

$$K_2 = \{(55,53), (53,49), (62,58), (55,54), (62,54)\}.$$

Зауважимо, що жоден мінтерм не міститься у простих кон’юнктермах, які належать різним підмножинам. Значить можна здійснити покриття кожної з підмножин окремо.

	62	55	54	53	14	10	8	7	6	5	1	0
(62,58)	1											
(62,54)	1		1									
(55,54)		1	1									
(55,53)		1		1								
(53,49)				1								
(14,10)					1	1						
(10,8)						1	1					
(14,6)					1				1			
(7,6)								1	1			
(7,5)								1		1		
(5,1)										1	1	
(8,0)							1					1
(1,0)											1	1

Рисунок 2.6 – Циклічна частина таблиці простих кон’юнктермів

Візьмемо до розгляду підмножину K_1. Складемо для неї таблицю простих кон’юнктермів (рисунок 2.7). Зауважимо, що вага кожного стовпця дорівнює двом, усі кон’юнктерми мають ранг 2. Оскільки в таблиці немає рядків з вагою 1, обираємо кон’юнктерм, що відповідає першому рядку. Присвоюємо цьому кон’юнктерму перший номер (вказуємо у нижньому індексі): $(1,0)_1$.

Наступним у ланцюжку обираємо $(5,1)_2$, потім $(7,5)_3$ і так дальше, поки не опрацюємо всі кон’юнктерми. Одержимо упорядковану множину:

$$K_1 = \langle (1,0)_1, (5,1)_2, (7,5)_3, (7,6)_4, (14,6)_5, (14,10)_6, (10,8)_7, (8,0)_8 \rangle.$$

	14	10	8	7	6	5	1	0
(14,10)	1	1						
(10,8)		1	1					
(14,6)	1				1			
(7,6)				1	1			
(7,5)				1		1		
(5,1)						1	1	
(8,0)			1					1
(1,0)							1	1

Рисунок 2.7 – Таблиця простих кон’юнктермів підмножини K_1

Першому кон’юнктерму $(1,0)_1$ упорядкованої множини K_1 у таблиці простих кон'юнктермів (рисунок 2.7) відповідає рядок, що має вагу 2. Тому існує два рівноцінні розв’язки покриття. Перший містить усі прості кон’юнктерми, яким в упорядкованій множині присвоєно непарні номери:

$$K_{1,1} = \{(1,0)_1, (7,5)_3, (14,6)_5, (10,8)_7\},$$

Другий розв’язок містить усі прості кон’юнктерми, яким в упорядкованій множині присвоєно парні номери:

$$K_{1,2} = \{(5,1)_2, (7,6)_4, (14,10)_6, (8,0)_8\}.$$

Аналогічно, знайдемо покриття множини K_2. Складемо таблицю простих кон’юнктермів (рисунок 2.8).

Зауважимо, що ранг кожного простого кон’юнктерма дорівнює 2, як і вага кожного стовпця. Оскільки в таблиці є рядок, що має одиничну вагу, першим обираємо відповідний кон’юнктерм і присвоюємо йому перший номер: $(53,49)_1$. Тоді наступним буде $(55,53)_2$, потім $(55,54)_1$ і так дальше.

	62	55	54	53
(54,62)	1		1	
(62,58)	1			
(53,55)		1		1
(55,54)		1	1	
(49,53)				1

Рисунок 2.8 – Таблиця простих кон’юнктермів підмножини K_2

Одержимо упорядковану множину:

$$K_2 = \langle (53,49)_1, (55,53)_2, (55,54)_3, (62,54)_4, (62,58)_5 \rangle.$$

Кон’юнктерму із номером 1 відповідає рядок одиничної ваги. При цьому потужність множини K_2 є непарним числом. Тому розв’язок мінімального покриття K_2 складають усі мінтерми з парними номерами, а саме

$$K_{2,1} = \{(55,53)_2, (62,54)_4\}.$$

Мінімальним покриттям заданої функції Y є об’єднання розв’язків покриття підмножин K_1 та K_2 із множиною істотних кон’юнктермів. Оскільки для K_1 маємо два розв’язки, то і для Y одержимо два рівноцінні розв’язки, а саме

$$K_{1,1} \cup K_{2,1} \cup K_{ist} =$$
$$= \{(1,0)_1, (7,5)_3, (14,6)_5, (10,8)_7\}, \{(55,53)_2, (62,54)_4\}, \{(49,48), (122,58)\},$$

$$K_{1,2} \cup K_{2,1} \cup K_{ist} =$$
$$= \{(5,1)_2, (7,6)_4, (14,10)_6, (8,0)_8\}, \{(55,53)_2, (62,54)_4\}, \{(49,48), (122,58)\}$$

У запропонованому способі ланцюгового покриття враховано взаємне розташування кон’юнктермів у буловому просторі, що дає змогу знайти тупикову диз’юнктивну нормальну форму булової функції без перебору. У випадку ланцюгової функції чи її фрагментів знайдена тупикова диз’юнктивна нормальна форма є мінімальною.

2.6 Висновок до розділу 2

1) Розроблено числове подання кон’юнктермів «маскове зображення», що являє собою пару чисел (числове зображення маски літералів і числове зображення кон'юнктерма з нулями замість символів поглинання). Оскільки склеювання можливі лише між кон’юнктермами, що мають однаковий набір поглинутих бітів, розглядають множини кон’юнктермів із спільною маскою літералів. Тоді немає необхідності зберігати для кожного кон'юнктерма обидва числа його маскового зображення. Спільну маску літералів кон’юнктермів зберігають як одне число для усієї розглянутої множини — маску літералів множини кон'юнктермів. У такому випадку операції над кон’юнктермами можна виконувати у формі побітових операцій над двійковими числами. Це потребує менших обчислювальних затрат, ніж зберігання кон’юнктермів у символьному поданні та операції над символами.

2) Розроблено процедуру гіпотетичного синтезування кон’юнктермів, за допомогою якої можна без проміжних склеювань шукати кон'юнктерми будь-якого рангу заданої булової функції. Це дає змогу уникнути появи тавтології в процесі пошуку простих кон'юнктермів. Як наслідок зменшуються обчислювальні витрати в задачах мінімізації булових функцій.

3) Удосконалено метод побітового сортування множини цілих чисел, яка не містить тавтології, а саме додано процедуру виявлення склеювання підмножини у кон’юнктерм. Для сортування такої підмножини достатньо

згенерувати послідовність чисел від нуля до (2^n–1) (де n — кількість поглинутих бітів в одержаному кон’юнктермі) та дописати непоглинуті біти до кожного з цих чисел. Для методу порозрядного вирощування простих кон'юнктермів булової функції необхідно упорядкувати за зростанням числових зображень вхідну множину мінтермів [107]. Застосувавши для цього запропоновану модифікацію побітового сортування зі склеюванням можна одержати деякі кон'юнктерми менших рангів без проміжних склеювань і тавтології.

4) Набула подальшого розвитку процедура спрощення множини символьних диз’юнктермів для теоретико-множинної модифікації мінімаксного методу покриття булових функцій. Запропоновано впорядкувати множину символьних диз’юнктермів за наростанням їх потужностей, що дає змогу видозмінити алгоритм процедури спрощення так, що скорочується шлях пошуку тих символьних диз’юнктермів, які необхідно вилучити з розгляду. Як наслідок загальна кількість порівнянь зменшується.

5) Запропоновано процедуру ланцюгового покриття таблиці простих кон’юнктермів булової функції, у якій ланцюжки кон’юнктермів будують на основі інформації про взаємні перетини простих кон’юнктермів циклічної частини, що дає змогу знайти тупикову диз’юнктивну нормальну форму булової функції без перебору. Для ланцюгової функції чи її фрагментів знайдена тупикова диз’юнктивна нормальна форма є мінімальною.

3 РОЗРОБЛЕННЯ ТА УДОСКОНАЛЕННЯ АЛГОРИТМІВ ПРОЦЕДУР ТА МЕТОДІВ МІНІМІЗАЦІЇ БУЛОВИХ ФУНКЦІЙ

Для подальших досліджень щодо оптимізації обчислювальних затрат обрано метод розчеплення кон'юнктермів (наближений метод пошуку множини простих кон'юнктермів), метод порозрядного вирощування (точний метод пошуку множини простих кон'юнктермів), мінімаксний метод покриття (наближений метод покриття множини простих кон'юнктермів).

3.1 Дослідження та удосконалення алгоритму мінімізації булових функцій розчепленням кон’юнктермів

У методі розчеплення кон’юнктермів [52] необхідно сформувати матрицю розчеплення мінтермів і здійснити її покриття векторами. Далі процедури розчеплення і покриття виконують окремо для кожного одержаного вектора покриття. Процес завершується, коли будуть опрацьовані усі літерали чи залишаться вектори, що містять не більше одного кон’юнктерма. У додатку Б наведено блок-схеми, що відображають алгоритм методу розчеплення кон’юнктермів. У разі програмної реалізації цього методу використання псевдотрійкового подання кон’юнктермів призводить до підвищення обчислювальних витрат. Для зменшення цих витрат запропоновано у наведеному алгоритмі перейти до числового подання кон’юнктермів на основі маскового зображення, а також якомога раніше виявляти та усувати неістотні змінні.

3.1.1 Удосконалена процедура розчеплення кон’юнктермів

Розчеплення кон’юнктермів відбувається послідовно по одному біту на кожному рівні. Розроблену в [97, 106] процедуру гіпотетичного синтезування кон’юнктермів можна вважати узагальненням процедури розчеплення кон’юнктермів на деякому наборі бітів одночасно. Якщо кількість мінтермів заданої функції від n змінних близька до 2^n, тоді гіпотетичне синтезування кон’юнктермів дає змогу зменшити кількість проміжних склеювань. Але в іншому випадку в матрицях розчеплення з’являється більша кількість непокритих векторів. Тому відмовимось від уведення процедури гіпотетичного синтезування кон’юнктермів в алгоритм методу розчеплення кон’юнктермів.

У [104] запропоновано виконувати процедуру розчеплення у числовій формі із використанням маскового зображення кон’юнктерма. Блок-схему алгоритму процедури формування маскового зображення кон’юнктерма із псевдотрійкового зображення наведено на рисунку 3.1.

На вхід алгоритму (рисунок 3.1) подається псевдотрійкове зображення кон’юнктерма (блок NM1). Далі можна одночасно в паралельних процесах отримати числове зображення маски літералів заміною нулів на одиниці та символів поглинання на нулі (блок NM2) і отримати числове зображення кон’юнктерма заміною символів поглинання на нулі (блок NM3). На виході алгоритму (блок NM4) отримаємо маскове зображення кон’юнктерма як пару чисел.

Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(n)$ (де n — кількість змінних булової функції).

У разі опрацювання множини мінтермів алгоритм спрощується, оскільки числове подання маски літералів необхідно одержати лише один раз для усієї множини.

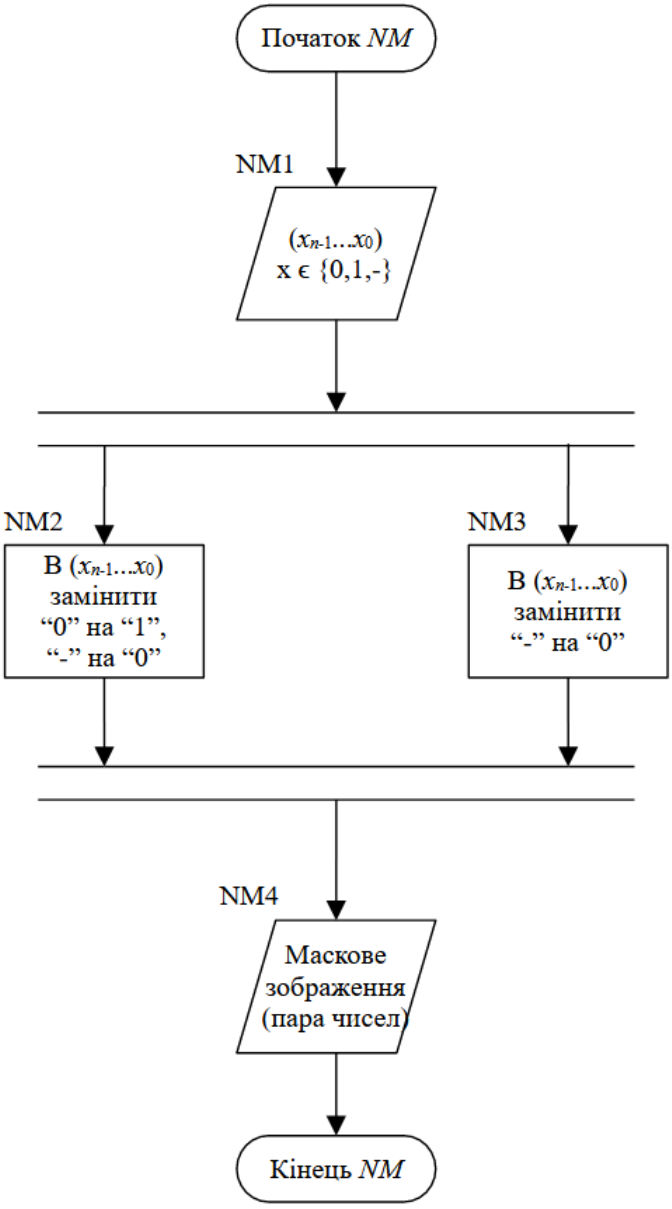


Рисунок 3.1 – Блок-схема алгоритму процедури формування маскового зображення кон’юнктерма із псевдотрійкового зображення

Оскільки в матриці розчеплення в усіх кон’юнктермів одного вектора однакова маска літералів, то кожен кон’юнктерм подається одним числом, а маска літералів вказується один раз для всього вектора. Це дає змогу надалі оперувати числами замість символьних зображень.

Проілюструємо на прикладі процедуру розчеплення в масковому зображенні.

$$Y^1 = \{(111),(110),(101),(000)\}^1 \overset{s}{\Rightarrow} \begin{bmatrix} l & l & - \\ l & - & l \\ - & l & l \end{bmatrix} =$$

$$= \begin{bmatrix} \underline{11-} & \underline{11-} & 10- & 00- \\ \underline{1-1} & 1-0 & \underline{1-1} & 0-0 \\ -11 & -10 & -01 & -00 \end{bmatrix} =$$

$$= \begin{bmatrix} \underline{110_{110}} & \underline{110_{110}} & 100_{110} & 000_{110} \\ \underline{101_{101}} & 100_{101} & \underline{101_{101}} & 000_{101} \\ 011_{011} & 010_{011} & 001_{011} & 000_{011} \end{bmatrix} =$$

$$= \begin{bmatrix} \{\underline{110} & \underline{110} & 100 & 000\}_{110} \\ \{\underline{101} & 100 & \underline{101} & 000\}_{101} \\ \{011 & 010 & 001 & 000\}_{011} \end{bmatrix}$$

Числа, що відповідають вагам бітів двійкової маски літералів, називатимемо вагами розчеплення.

Процедура розчеплення у масковому зображенні:

1) з твірної маски літералів складаємо матрицю-стовпець присутніх у ній ваг розчеплення;

2) виконуємо побітову інверсію кожного елемента матриці-стовпця ваг розчеплення;

3) для кожного вектора обчислюємо кожен елемент матриці розчеплення операцією побітової кон'юнкції між числовим зображенням твірного кон'юнктерма та інвертованою вагою розчеплення, що відповідає заданому вектору.

Для наведеного вище прикладу перші два кроки набувають вигляду:

$$111 \overset{\text{ваги розчеплення}}{\Rightarrow} \begin{bmatrix} 001 \\ 010 \\ 100 \end{bmatrix} \overset{\text{побітова інверсія}}{\Rightarrow} \begin{bmatrix} 110 \\ 101 \\ 011 \end{bmatrix}.$$

Розв’язування у масковому зображенні має вигляд:

$$Y^1 = \{(111),(110),(101),(000)\}^1_{111} \overset{S}{\Rightarrow}$$

$$\overset{S}{\Rightarrow} \begin{bmatrix} 110 \\ 101 \\ 011 \end{bmatrix} \& \{(111),(110),(101),(000)\}^1_{111} =$$

$$= \begin{bmatrix} \{\underline{110} & \underline{110} & 100 & 000\}_{110} \\ \{\underline{101} & 100 & \underline{101} & 000\}_{101} \\ \{011 & 010 & 001 & 000\}_{011} \end{bmatrix}$$

На рисунку 3.2 наведено блок-схему алгоритму процедури розчеплення кон’юнктерма по одному біту в масковому зображенні. На вході алгоритму (блок SNMa1) маємо номер біта розчеплення a та маскове зображення кон’юнктерма (пара чисел Num і Mask). У блоці SNMa2 виконується побітова інверсія числа, що є вагою заданого біта:

$$B = \sim(2^a). \tag{3.1}$$

Далі у паралельних процесах виконують блоки SNMa3 та SNMa4.

У блоці SNMa3 обраховується числове подання маски літералів розчепленого кон’юнктерма, а саме занулення біта a шляхом побітової кон’юнкції числової маски літералів вихідного (твірного) кон’юнктерма та обрахованого на попередньому кроці числа B:

$$Mask_a = Mask \,\&\, B, \tag{3.2}$$

де & — символ операції побітової кон’юнкції двох чисел.

У блоці SNMa4 обраховується числове зображення розчепленого кон’юнктерма, а саме занулення біта a в числовому зображенні твірного за

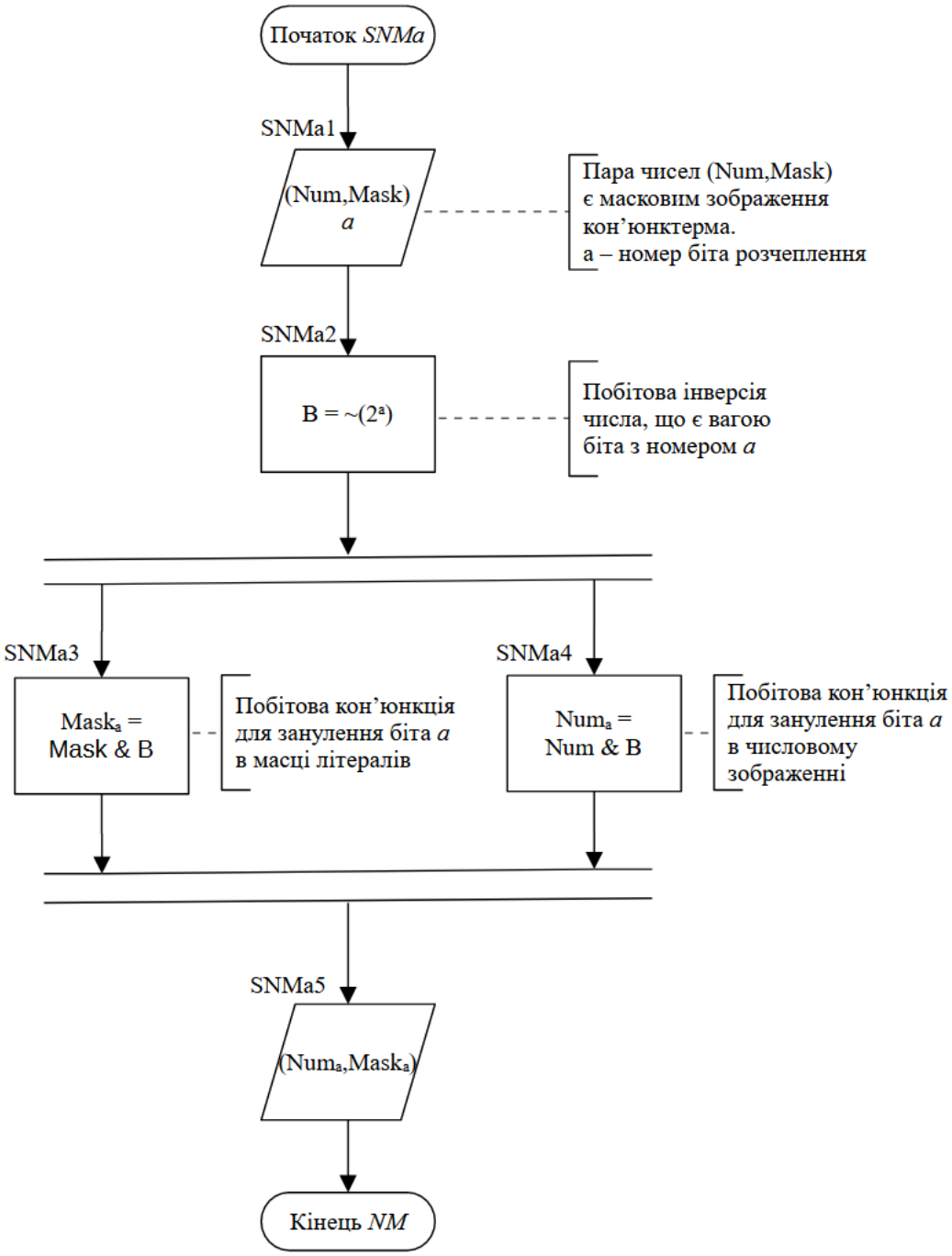


Рисунок 3.2 – Блок-схема алгоритму процедури розчеплення кон'юнктерма по одному біту в масковому зображенні

допомогою операції побітової кон'юнкції із числом B:

$$Num_a = Num \,\&\, B. \tag{3.3}$$

Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(1)$, тобто не залежить від кількості змінних булової функції.

Кожен процесор має апаратну реалізацію побітових операцій. Порівняно з маніпуляціями із символьними псевдотрійковими зображеннями побітові операції спричиняють значно менші обчислювальні витрати.

При здійсненні процедури розчеплення “вручну” людині зручніше скористатись вісімковою чи шістнадцятковою системами числення замість двійкової для запису маскового зображення. Тоді запис таблиці розчеплення кон’юнктермів буде компактнішим. Крім того простіше буде шукати кон’юнктерми-копії на векторах матриці. Оскільки кожній позиції вісімкового

8-4-2-1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0 0 0 0	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0 0 1	**1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 0 1 0	**2**	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
0 0 1 1	**3**	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0 1 0 0	**4**	0	0	0	0	4	4	4	4	0	0	0	0	4	4	4	4
0 1 0 1	**5**	0	1	0	1	4	5	4	5	0	1	0	1	4	5	4	5
0 1 1 0	**6**	0	0	2	2	4	4	6	6	0	0	2	2	4	4	6	6
0 1 1 1	**7**	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1 0 0 0	**8**	0	0	0	0	0	0	0	0	8	8	8	8	8	8	8	8
1 0 0 1	**9**	0	1	0	1	0	1	0	1	8	9	8	9	8	9	8	9
1 0 1 0	**A**	0	0	2	2	0	0	2	2	8	8	A	A	8	8	A	A
1 0 1 1	**B**	0	1	2	3	0	1	2	3	8	9	A	B	8	9	A	B
1 1 0 0	**C**	0	0	0	0	4	4	4	4	8	8	8	8	C	C	C	C
1 1 0 1	**D**	0	1	0	1	4	5	4	5	8	9	8	9	C	D	C	D
1 1 1 0	**E**	0	0	2	2	4	4	6	6	8	8	A	A	C	C	E	E
1 1 1 1	**F**	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Рисунок 3.3 – Карта побітової кон’юнкції пар кон’юнктермів функції від чотирьох змінних, що тотожна одиниці [104]

подання відповідає три біти двійкового подання, а кожній позиції шістнадцяткового подання — чотири біти, то побітові операції над числовими зображеннями кон’юнктермів можна виконувати окремо над кожною цифрою як у вісімковому, так і в шістнадцятковому поданні. Щоб полегшити формування матриці розчеплення, складемо карту розчеплення літералів функції від чотирьох змінних у поданні шістнадцятковими цифрами. Для цього спочатку складемо у поданні шістнадцятковими цифрами карту побітової кон’юнкції (рисунок 3.3) усіх можливих пар кон’юнктермів функції від чотирьох змінних, що тотожна одиниці.

Числа 7, B, D, E є шістнадцятковими зображеннями побітової інверсії ваг розчеплення маски літералів булової функції від чотирьох змінних. Тому з карти на рисунку 3.3 виокремимо стовпці 7, B, D, E (рисунок 3.4).

8-4-2-1		**7**	**B**	**D**	**E**
0 0 0 0	**0**	0	0	0	0
0 0 0 1	**1**	1	1	1	0
0 0 1 0	**2**	2	2	0	2
0 0 1 1	**3**	3	3	1	2
0 1 0 0	**4**	4	0	4	4
0 1 0 1	**5**	5	1	5	4
0 1 1 0	**6**	6	2	4	6
0 1 1 1	**7**	7	3	5	6
1 0 0 0	**8**	0	8	8	8
1 0 0 1	**9**	1	9	9	8
1 0 1 0	**A**	2	A	8	A
1 0 1 1	**B**	3	B	9	A
1 1 0 0	**C**	4	8	C	C
1 1 0 1	**D**	5	9	D	C
1 1 1 0	**E**	6	A	C	E
1 1 1 1	**F**	7	B	D	E

Рисунок 3.4 – Фрагмент карти побітової кон’юнкції інверсій ваг розчеплення

Замінимо у першому рядку карти рисунку 3.4 інверсні значення ваг розчеплення на прямі. Одержимо карту розчеплення літералів функції від чотирьох змінних у поданні шістнадцятковими цифрами (рисунок 3.5).

8-4-2-1		**8**	**4**	**2**	**1**
0 0 0 0	**0**	0	0	0	0
0 0 0 1	**1**	1	1	1	0
0 0 1 0	**2**	2	2	0	2
0 0 1 1	**3**	3	3	1	2
0 1 0 0	**4**	4	0	4	4
0 1 0 1	**5**	5	1	5	4
0 1 1 0	**6**	6	2	4	6
0 1 1 1	**7**	7	3	5	6
1 0 0 0	**8**	0	8	8	8
1 0 0 1	**9**	1	9	9	8
1 0 1 0	**A**	2	A	8	A
1 0 1 1	**B**	3	B	9	A
1 1 0 0	**C**	4	8	C	C
1 1 0 1	**D**	5	9	D	C
1 1 1 0	**E**	6	A	C	E
1 1 1 1	**F**	7	B	D	E

Рисунок 3.5 – Карта розчеплення літералів функції від чотирьох змінних у поданні шістнадцятковими цифрами

Аналогічно, складемо карту розчеплення літералів функції від трьох змінних у поданні вісімковими цифрами. Для цього з карти побітової кон’юнкції на рисунку 3.3 виокремимо стовпці 6, 5 та 3. Нас цікавлять тільки рядки, що відповідають вісімковим цифрам, тому відкинемо рядки, починаючи з цифри “8”. Замінимо у першому рядку інверсні значення ваг розчеплення на прямі. Одержимо карту розчеплення літералів функції від трьох змінних у поданні вісімковими цифрами (рисунок 3.6).

4-2-1		**4**	**2**	**1**
0 0 0	**0**	0	0	0
0 0 1	**1**	1	1	0
0 1 0	**2**	2	0	2
0 1 1	**3**	3	1	2
1 0 0	**4**	0	4	4
1 0 1	**5**	1	5	4
1 1 0	**6**	2	4	6
1 1 1	**7**	3	5	6

Рисунок 3.6 – Карта розчеплення літералів функції від трьох змінних у поданні вісімковими цифрами

Для здійснення процедури розчеплення “вручну”, необхідно у числовій карті розчеплення (рисунок 3.5 чи 3.6) вибрати рядок твірного кон’юнктерма і стовпець ваги вектора розчеплення. На їх перетині знайдемо розчеплений кон’юнктерм у числовому поданні. Наприклад, для розчеплення кон’юнктерма 101 (5) по нульовому біту обираємо рядок карти розчепленя, у першому стовпці якої вказано число “5”. Оскільки вага нульового біту дорівнює $2^0 = 1$, то обираємо стовпець таблиці, у заголовку якого вказано “1”. На перетині обраних рядка і стовпця комірка містить число 4 (100 у двійковому поданні). Порівняно з (5) (101 у двійковому поданні) обнулено біт з номером “0”.

Застосування запропонованої модифікації процедури розчеплення кон’юнктермів дає змогу зменшити обчислювальні витрати. Використання числового подання у вісімковій чи шістнадцятковій системі числення дає змогу розширити коло завдань, які можна розв’язати “вручну”, тобто без залучення обчислювальної техніки.

3.1.2 Усунення несуттєвих змінних

У [104] запропоновано зміни до алгоритму розчеплення кон’юнктермів, що полягають у зміні порядку формування матриці розчеплення з метою виявлення неістотних змінних за допомогою частково сформованої матриці. Це дає змогу зменшити обчислювальні витрати.

У методі розчеплення кон’юнктермів на етапі покриття матриці розчеплення можуть бути виявленні повністю покриті вектори матриці розчеплення. Такі вектори складаються лише з кон’юнктермів-копій. Це означає, що є несуттєвою змінна, яка відповідає повністю покритому вектору матриці розчеплення. У цьому випадку необхідно вилучити з подальшого розгляду відповідні вектори, а також неістотні змінні у кожному кон’юнктермі як твірному, так і розчепленому. Крім того необхідно вилучити з матриці по одному стовпцю з кожної пари копій, що знайдені на повністю покритому векторі матриці розчеплення. Продовжити процедуру покриття можна тільки після усунення усіх знайдених у такий спосіб несуттєвих змінних.

Можна скоротити обсяг обчислювальних витрат, якщо формувати матрицю розчеплення по рядках а не по стовпцях. Після формування кожного рядка одразу шукати пари кон’юнктермів-копій, щоб якомога швидше виявити повністю покритий вектор матриці розчеплення і усунути несуттєву змінну з розгляду. При цьому удвічі зменшиться кількість стовпців. Таким чином вдасться зменшити розмірність матриці розчеплення ще на етапі її формування і тільки тоді перейти до опрацювання наступної змінної, тобто формування наступного вектора матриці розчеплення.

На рисунку 3.7 наведено блок-схему алгоритму процедури формування вектора розчеплених кон’юнктермів у масковому зображенні (модуль SV). Розчеплення здійснюється по деякому біту *a*. Вхідними даними алгоритму є вектор маскових зображень твірних кон’юнктермів (блок SV1). У блоці SV2

виконується побітова інверсія числа, що є вагою біта *a*. У блоці SV3 відбувається формування числової маски літералів розчепленого вектора. Блок SV4

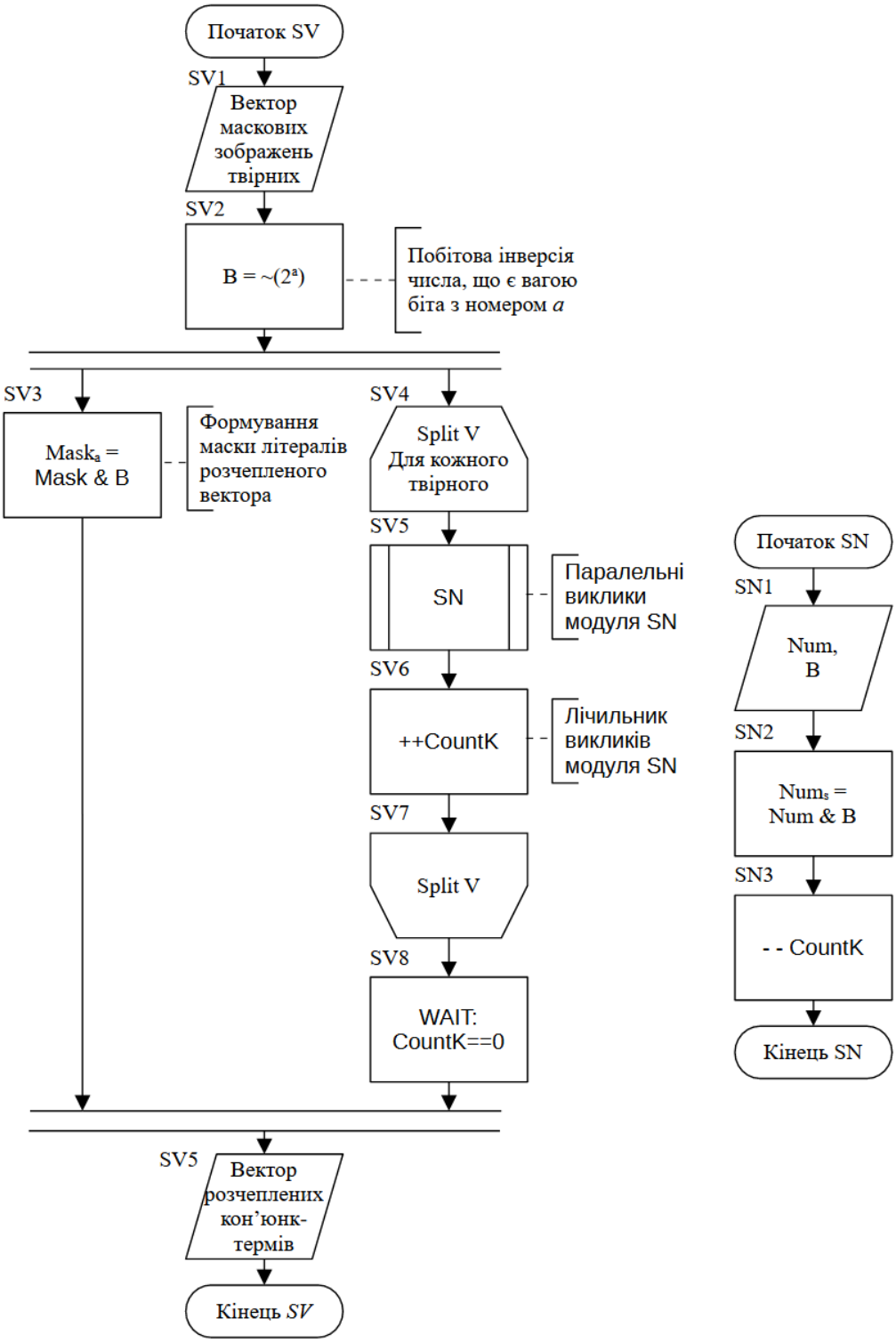


Рисунок 3.7 – Блок-схема алгоритму процедури формування вектора розчеплених кон'юнктермів у масковому зображенні

розпочинає цикл “Split V” опрацювання твірних кон’юнктермів. У кожній ітерації блок SV5 для чергового твірного кон’юнктерма викликає модуль SN обрахунку числового зображення відповідного розчепленого кон’юнктерма. Кожна копія модуля SN може працювати у незалежному від інших копій процесі. Крім того у кожній ітерації циклу “Split V” блок SV6 інкрементує лічильник викликів модуля SN (CountK). Це необхідно, щоб контролювати кількість незавершених процесів. Модуль SV7 завершує ітерацію і передає керування блоку SV4 (перевірка умови виконання наступної ітерації). Цикл завершується, коли опрацьовано всі твірні кон’юнктерми. Тоді керування переходить до блоку SV8, який тимчасово зупиняє виконання модуля SV до завершення всіх копій модуля SN. Модуль SN отримає від процесу, що його викликав, числове зображення твірного кон’юнктерма *Num* та число *B* (побітову інверсію ваги біта розчеплення). Блок SN2 обраховує числове зображення розчепленого кон’юнктерма як побітову кон’юнкцію *Num* та *B*. Блок SN3 декрементує лічильник викликаних копій модуля SN і завершує роботу поточної копії. Коли завершаться всі копії, у параметрі CountK встановиться нульове значення, модуль SV продовжить роботу, а саме видасть результат — вектор розчеплених кон’юнктермів, після цього завершить роботу.

Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(N)$ (де N — кількість твірних кон’юнктермів у векторі).

У разі подання кон’юнктермів масковим зображенням алгоритм усунення неістотної змінної формується на основі побітових операцій, а саме:

1) для повністю покритого вектора розраховуємо побітову кон’юнкцію числового зображення кожного твірного кон’юнктерма із вагою вектора розчеплення. Якщо результат дорівнює нулю, одразу вилучаємо твірний і відповідний стовпець із матриці;

2) для твірного вектора та для кожного вектора розчеплення обчислюємо нову маску літералів як побітову кон’юнкцію старої маски літералів і ваги розчеплення вектора несуттєвої змінної;

3) вилучаємо повністю покритий вектор з матриці розчеплення.

Проілюструємо усунення несуттєвої змінної на прикладі функції від чотирьох змінних, яка задана множиною кон’юнктермів у масковому зображенні за допомогою вісімкових чисел:

$$Y^1 = \{00,01,02,03,04,05,06,07,10,11,12,13,14,16\}_{17}^{1}$$

Послідовно формуємо вектори матриці розчеплення. Одразу після формування кожного вектора матриці розчеплення здійснюємо пошук кон’юнктермів-копій на ньому. Виявляємо, що другий вектор є повністю покритий. Тому зупиняємо процедуру розчеплення, щоб усунути несуттєву змінну і половину твірних кон’юнктермів.

$$\begin{matrix} \{00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 10 & 11 & 12 & 13 & 14 & 16\}_{17} \end{matrix}$$
$$\begin{bmatrix} \{\underline{00} & \underline{00} & \underline{02} & \underline{02} & \underline{04} & \underline{04} & \underline{06} & \underline{06} & \underline{10} & \underline{10} & \underline{12} & \underline{12} & 14 & 16\}_{16} \\ \{\underline{00} & \underline{01} & \underline{00} & \underline{01} & \underline{04} & \underline{05} & \underline{04} & \underline{05} & \underline{10} & \underline{11} & \underline{10} & \underline{11} & \underline{14} & \underline{14}\}_{15} \\ \{ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \}_{13} \\ \{ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \}_{07} \end{bmatrix}$$

Унаслідок скорочення отримаємо:

$$= \begin{matrix} \{00 & 01 & 04 & 05 & 10 & 11 & 14\}_{15} \end{matrix}$$
$$\begin{bmatrix} \{\underline{00} & \underline{00} & \underline{04} & \underline{04} & \underline{10} & \underline{10} & 14\}_{14} \\ \{ ? & ? & ? & ? & ? & ? & ? \}_{11} \\ \{ ? & ? & ? & ? & ? & ? & ? \}_{05} \end{bmatrix}.$$

Тепер можна продовжувати розчеплення за рештою позицій (змінних). Як бачимо для подальших операцій маємо вдвічі менший твірний вектор.

На рисунку 3.8 подано блок-схему алгоритму удосконаленої процедури формування матриці розчеплених кон’юнктермів у масковому зображенні. Вхідними даними алгоритму є вектор твірних кон’юнктермів у масковому зображенні із спільною маскою літералів (блок SNM1). Розчеплення починаємо з молодшого біта. Тому в блоці SNM2 ініціалізуємо нулем номер біта розчеплення a. Тоді вагу біта розчеплення b встановлюємо $2^0 = 1$. Блок SNM3 запускає цикл “Split NM” опрацювання бітів маски літералів Mask. Якщо маска

літералів подана непарним числом (блок умови SNM4), значить її молодший біт має значення “1”, тобто цей літерал присутній у масці. Тоді блок SNM5 викликає модуль SV формування вектора розчеплених кон’юнктермів. Блок SNM6 відшукує кон’юнктерми-копії на сформованому векторі.

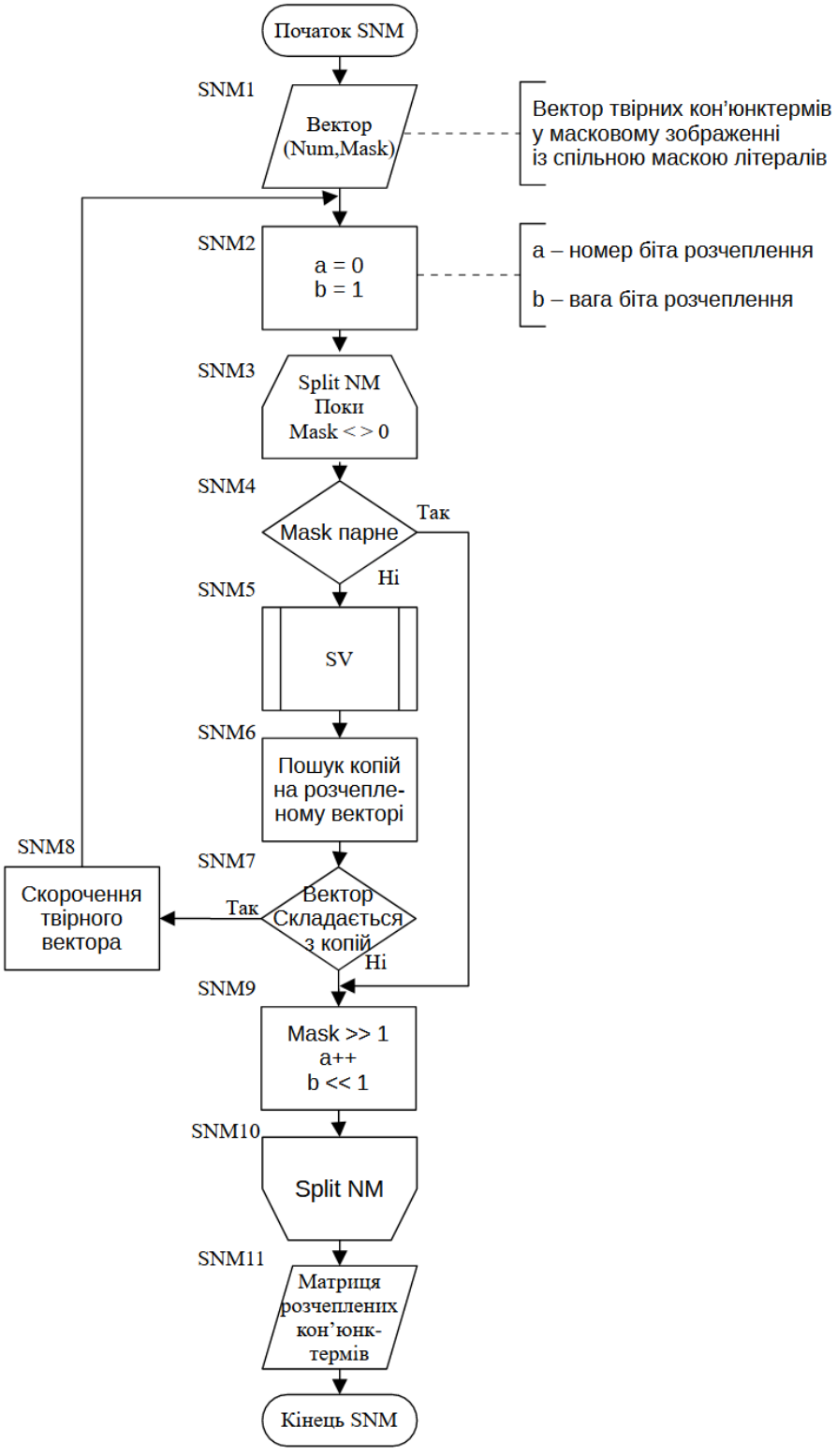


Рисунок 3.8 – Блок-схема алгоритму удосконаленої процедури формування матриці розчеплених кон’юнктермів у масковому зображенні

Якщо блок умови SNM7 виявляє, що вектор матриці розчеплення є повністю покритим, тобто складається лише з пар кон’юнктермів-копій, то розглянута змінна є несуттєвою. Тоді алгоритм виходить із циклу “Split NM”, блок SNM8 усуває несуттєву змінну та скорочує удвічі твірний вектор. Керування передається на блок SNM2, щоб наново розпочати цикл “Split NM”, але вже для удвічі коротшого твірного вектора матриці розчеплення.

Якщо блок умови SNM7 виявив, що одержаний у модулі SN вектор на є повністю покритим, то блок SNM9 здійснює підготовку наступної ітерації, а саме інкрементує номер біта розчеплення a, встановлює у змінній b вагу нового біта розчеплення шляхом зсуву вправо на один біт числа b, відсікає опрацьований біт маски літералів Mask шляхом зсуву вправо на один біт. Тепер у масці літералів новий біт розчеплення є молодшим бітом.

Також керування може бути передано блоку SNM9 від блока SNM4, якщо блок SNM4 виявляє, що маска літералів подана парним числом, тобто її молодший біт має значення “0”, а значить розглянутий літерал відсутній у масці. Оскільки процедура розчеплення кон’юнктермів здійснюється інваріантно щодо символів поглинання, то без виклику процедури формування вектору розчеплення здійснюється перехід до блоку SNM9 для підготовки наступної ітерації.

Блок SNM10 завершує ітерацію і передає керування на блок SNM3 для перевірки умови циклу. Цикл завершується, коли параметр Mask дорівнюватиме нулю, тобто будуть опрацьовані усі літерали маски. На виході алгоритму отримаємо матрицю розчеплених кон’юнктермів (блок SNM11).

Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(n \cdot N)$ (де n — кількість змінних у буловій функції, а N — кількість твірних кон’юнктермів у векторі).

3.1.3 Удосконалення алгоритму розчеплення кон’юнктермів

У [104] запропоновано удосконалення алгоритму розчеплення кон’юнктермів. За відсутності на певному векторі кон’юнктермів-копій не має сенсу в подальшому виконувати розчеплення по відповідному біту маски літералів. Цей біт маски літералів вважатимемо ізольованим, як і відповідний вектор матриці розчеплення. Так само, якщо є лише одна пара кон’юнктермів-копій на певному векторі матриці розчеплення, то на наступних рівнях розчеплення за відповідним бітом маски літералів не буде жодної пари кон’юнктермів-копій. Тож, за наявності не більше одної пари кон’юнктермів копій на певному векторі розчеплення відповідний біт маски літералів буде ізольований, а значить у подальшому розчепленні участі не братиме.

Застосування інформації про ізольовані біти маски літералів дасть змогу позбутися надлишкових обчислювальних витрат на розчеплення за набором ізольованих бітів. Маскою ізольованих бітів називатимемо двійкове число, у якому нулі стоять у тих бітах, що відповідають ізольованим бітам. У інших бітах маски ізольованих бітів містяться одиниці. Будемо записувати маску ізольованих бітів після маски літералів і знак коса риска «/» між масками. Щоб не виконувати розчеплення по ізольованих векторах матриці розчеплення, матриця-стовпець ваг розчеплення має бути обчислена як побітова кон’юнкція маски ізольованих бітів і маски літералів твірного вектора. За умови виявлення ізольованого вектора необхідно обчислити нову маску ізольованих бітів як побітову кон’юнкцію старої маски ізольованих бітів та інвертованої ваги розчеплення.

Розширимо поняття ізольованого кон’юнктерма. Нехай для деякого твірного кон’юнктерма K рангу r у матриці розчеплення $MP(r\text{-}1)$ у відповідному стовпці присутня лише одна копія K_i, де i — номер вектора матриці розчеплення. Іншими словами K склеївся по біту i з деяким іншим кон’юнктермом рангу r і таким чином утворив кон’юнктерм K_i рангу $(r-1)$, у якого біт i поглинутий. Розглянемо гіпотезу, що існує такий кон’юнктерм K_{ij} $(r-2)$-рангу з поглинутими

бітами i та j, що містить K_i . У такому випадку K_{ij} також містить кон'юнктерм K_j $(r-1)$-рангу, який можна одержати з K склеюванням з деяким іншим кон'юнктермом по біту j. Тоді у матриці розчеплення $MP(r-1)$ на векторі з номером j у стовпці розглянутого твірного кон'юнктерма K має міститись ще один кон'юнктерм-копія K_j. Але це протирічить початковій умові, яка стверджує, що у відповідному стовпці матриці є тільки одна копія. Значить гіпотеза хибна, а у стовпці твірного K_i матриці $MP(r-2)$ не буде жодного кон'юнктерма-копії. Тоді на наступному кроці покриття кон'юнктерм K_i буде ізольованим, буде віднесений у множину покриття і буде усунутий з розгляду. Тож немає сенсу виконувати розчеплення кон'юнктерма на наступному кроці, якщо на попередньому він був єдиною копією у своєму стовпці. Потрібно такий кон'юнктерм одразу віднести до ізольованих і усунути з подальшого розгляду, а два стовпці матриці, що містять цей кон'юнктерм-копію позначити покритими та вилучити з матриці. Таким чином будемо відносити кон'юнктерм до ізольованих за однією з двох ознак:

1) якщо розглянутий твірний кон'юнктерм у відповідному стовпці матриці розчеплення немає жодного кон'юнктерма копії;

2) якщо розглянутий розчеплений кон'юнктерм є єдиною копією в своєму стовпці матриці розчеплення.

У блок-схемі на рисунку 3.9 відображено запропоновані зміни щодо опрацювання ізольованих кон'юнктермів в алгоритмі процедури покриття матриці розчеплених кон'юнктермів.

Вхідними даними алгоритму є матриця розчеплених кон'юнктермів (блок CNM1). Кожен стовпець матриці відповідає певному твірному кон'юнктерму, а кожен рядок – непоглинутій змінній твірних кон'юнктермів. У циклі "Isolated NM", що обмежений блоками CNM2 – CNM7, для кожного стовпця матриці розчеплення визначається, чи є відповідний твірний кон'юнктерм ізольованим. Зокрема, блок CNM3 перевіряє, чи є копії у стовпці матриці. За їх відсутності твірний кон'юнктерм відносять до множини покриття, а стовпець вилучають з подальшого розгляду (блок CNM4). На цьому ітерація завершується. Якщо ж

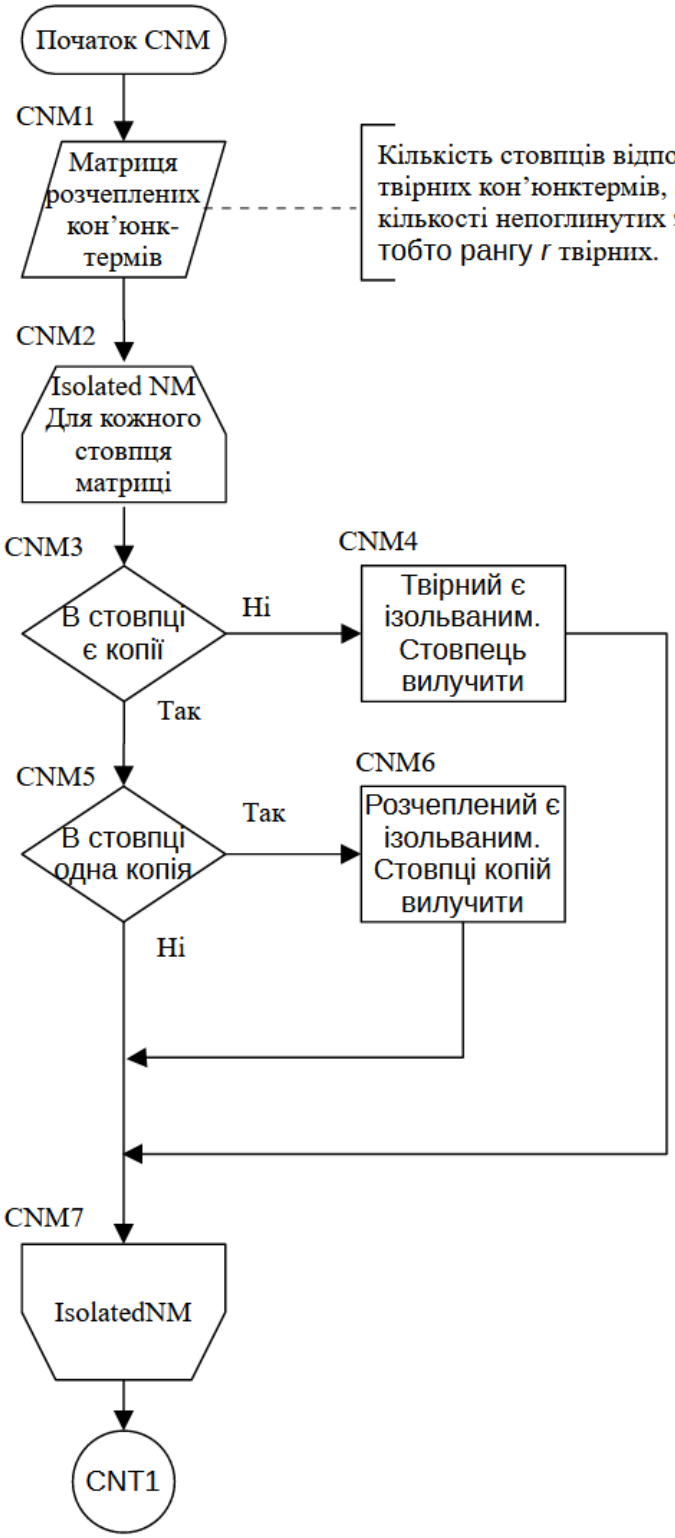


Рисунок 3.9 – Блок-схема алгоритму удосконаленої процедури покриття матриці розчеплення кон'юнктермів (початок)

копії у стовпці виявленні, то блок CNM5 перевіряє чи присутня тільки одна копія у стовпці. Якщо ні, ітерація завершується. Якщо так, то блок CNM6 відносить

розчеплений кон’юнктерм-копію до множини покриття, а відповідні два стовпці усуває з матриці. Ітерація завершується.

Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(n \cdot N)$ (де n — кількість змінних булової функції, а N — кількість твірних кон’юнктермів матриці розчеплення).

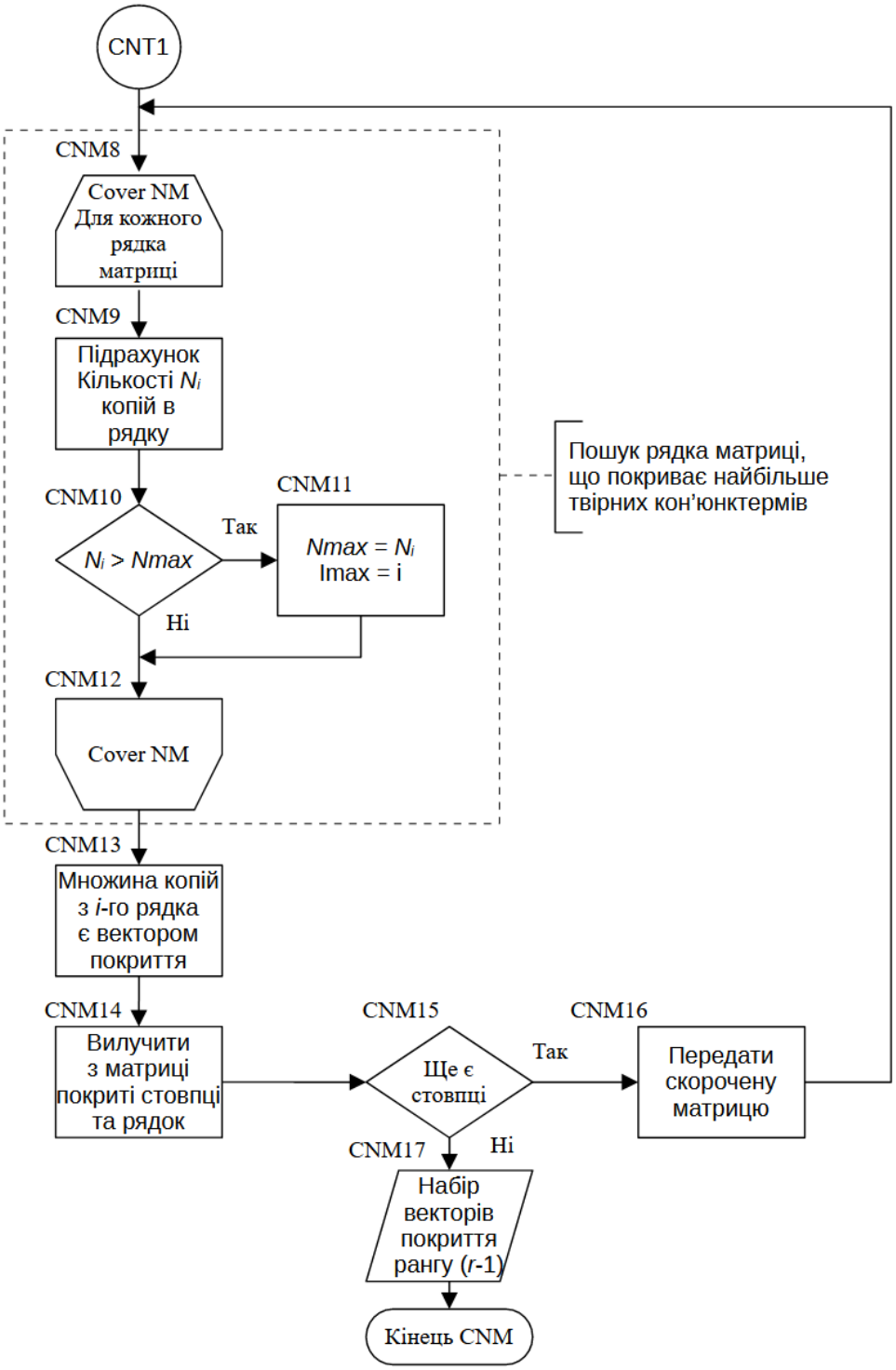


Рисунок 3.10 – Блок-схема алгоритму удосконаленої процедури покриття матриці розчеплення кон’юнктермів (продовження)

Алгоритм подальшого покриття вже скороченої матриці розчеплення не зазнав змін. Його блок-схему подано на рисунку 3.10. У циклі “Cover NM” (блоки CNM8 – CNM12) у кожній ітерації відбувається опрацювання одного рядка матриці розчеплення кон’юнктермів з метою знайти такий, що покриває найбільшу кількість стовпців. Множину копій зі знайденого рядка відносять до покриття (блок CNM13). Цей рядок і покриті ним стовпці усувають з матриці (блок CNM14). Якщо у матриці розчеплення ще залишились непокриті стовпці (блок CNM15), то скорочену матрицю передають знов на вхід циклу “Cover NM” блок CNM8. Інакше покриття завершено, одержано набір векторів рангу, що менший на одиницю від рангу твірних кон’юнктермів.

Після завершення покриття матриці розчеплення можна приступати до наступного кроку — розчеплення кон’юнктермів одержаних векторів покриття. Причому кожен вектор можна опрацьовувати у незалежному процесі, тобто виконувати паралельні обчислення.

Оскільки на кожному кроці покриття матриці розчеплення зменшується кількість рядків, сумарна кількість опрацьованих рядків не перевищує суми перших n членів арифметичної прогресії 1,2,3, … n, тобто $(1 + n) \cdot n / 2$. Тоді верхня оцінка асимптотичної складності алгоритму покриття матриці розчеплення дорівнює $O(n^2 \cdot N)$ (де n — кількість змінних булової функції, а N — кількість твірних кон’юнктермів матриці розчеплення).

На рисунку 3.11 наведено блок-схему удосконаленого алгоритму розчеплення кон'юнктермів. На вхід алгоритму подається вектор мінтермів (блок SCNM1). Другим блоком SCNM2 викликається модуль процедури розчеплення для вхідного вектора. У третьому блоці SCNM3 інкрементується лічильник векторів, що перебувають в опрацюванні. Четвертим блоком SCNM4 здійснюється очікування завершення опрацювання усіх векторів. П'ятий блок виводить множину простих кон'юнктермів. Після цього алгоритм завершується.

Модуль процедури розчеплення SCV у першому блоці SCV1 викликає модуль удосконаленої процедури побудови матриці розчеплення. Наступний блок SCV2 викликає модуль процедури покриття матриці розчеплення,

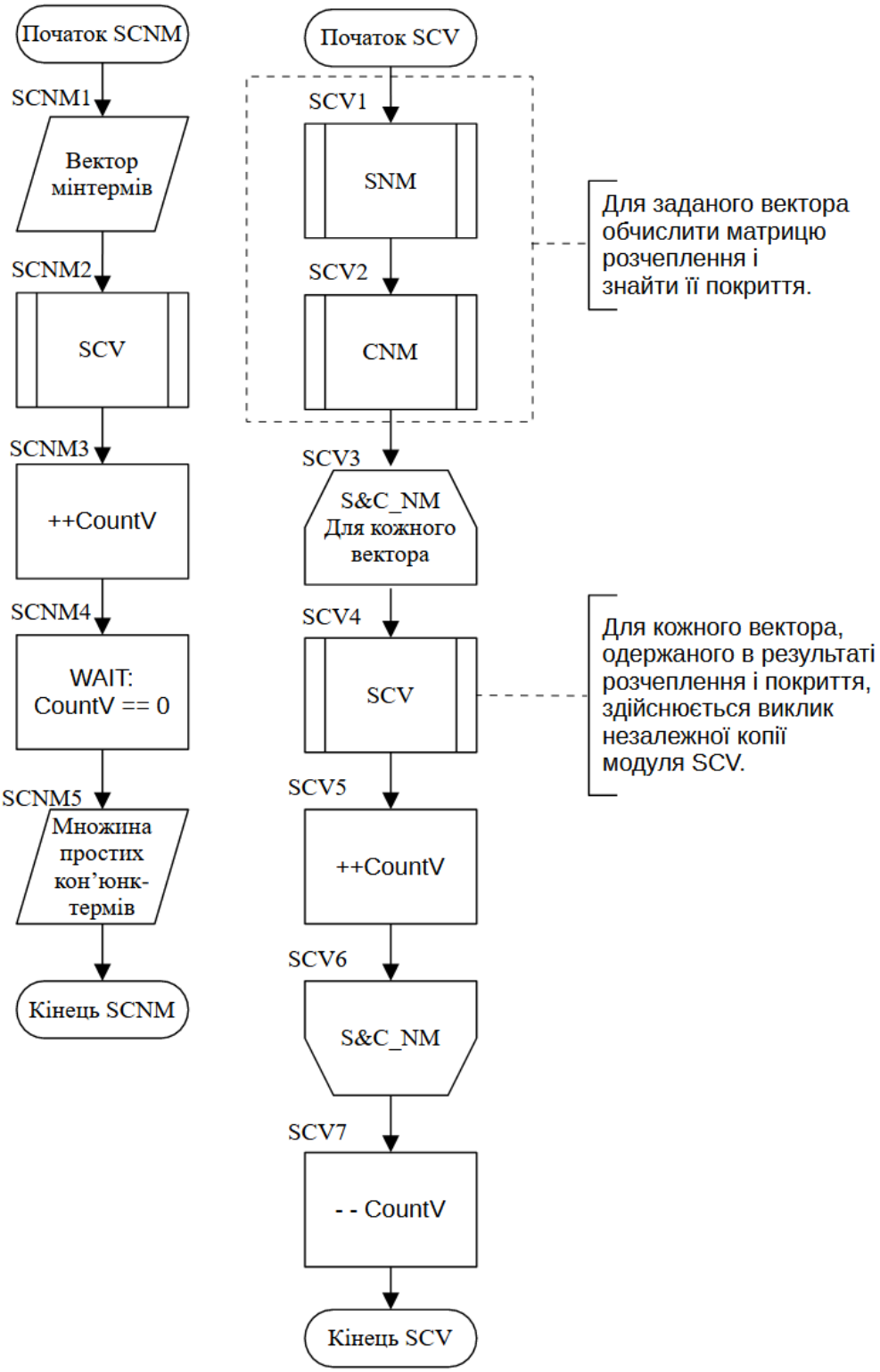


Рисунок 3.11 – Блок-схема алгоритму удосконаленого методу розчеплення кон'юнктермів

результатом роботи якого є множина векторів розчеплення. Блок SCV3 запускає цикл опрацювання одержаних векторів. На кожній ітерації береться один з

одержаних векторів. Для нього блоком SCV4 здійснюється виклик незалежної копії модуля SCV. Блок SCV5 інкрементує лічильник векторів, що перебувають на опрацюванні. Блок SCV6 завершує ітерацію циклу і переходить до блоку SCV3, тобто на початок наступної ітерації. Алгоритм виходить із циклу, коли опрацьовано усі вектори матриці розчеплення, що одержана з вхідного вектора.

Тоді блок SCV7 декрементує лічильник векторів, що перебувають на опрацюванні. Процедура завершується.

За рахунок рекурентного виклику процедури розчеплення для кожного новоутвореного вектора досягається швидке розпаралелення алгоритму. Тому час, необхідний на виконання алгоритму фактично визначається двома послідовними процедурами (побудови матриці розчеплення і її покриття), які викликають не більше ніж n разів (де n – кількість змінних). З цих двох процедур вищу верхню оцінку асимптотичної складності має алгоритм покриття матриці розчеплення $O(n^2 \cdot N)$. Тому для алгоритму розчеплення кон'юнктермів асимптотична верхня оцінка складності становить $O(n^3 \cdot N)$.

3.2 Дослідження та удосконалення алгоритму порозрядного вирощування

У [107] проаналізовано метод порозрядного вирощування простих кон'юнктермів, автор якого стверджує, що цей алгоритм не виконує надлишкові порівняні кон'юнктермів, відповідно, не допускає появу тавтології і у цьому плані не може бути покращений.

На основі певних закономірностей булового простору в [100, 107] запропоновано зміни та доповнення до методу порозрядного вирощування, які дають можливість за певних обставин знизити обсяг обчислень. Також розширено сферу застосування методу на недовизначені функції.

Розглянемо зміни та доповнення, що запропоновані до методу порозрядного вирощування. Псевдотрійкове зображення кон'юнктермів

ускладнює комп’ютерну реалізацію обчислень. Тому використовуватимемо числове подання у вигляді маскового зображення [97]. У методі порозрядного вирощування простих кон’юнктермів двійкове подання одного і того ж кон’юнктерма на різних кроках сортування містило різну кількість позицій, відповідно кон’юнктерм мав різне зображення. Використання подання у масковому зображенні забезпечує незмінне числове подання кожного кон’юнктерма на усіх етапах виконання алгоритму. Наприклад, кон’юнктерм (00101) на усіх етапах подається незмінним числовим зображенням, а не (0010), (001), (00), (0), як запропоновано у [107].

У процесі сортування по кожному біту отримаємо множини кон’юнктермів, характерною особливістю яких є спільна маска літералів для усіх кон’юнктермів в одній множині. Тому позначатимемо маску літералів єдиним числом для усієї множини (у нижньому індексі). Тоді кожен кон’юнктерм достатньо позначити одним числом — його числовим зображенням із маскового подання. Наприклад, для кон’юнктермів

$$\{(-1-00),(-1-01),(-1-10)\}$$

маскове зображення із спільною маскою літералів формуємо так

$$\{(01000),(01001),(01010)\}_{01011}\,.$$

У [100, 107] запропоновано на відміну від методу порозрядного вирощування простих кон’юнктермів формувати коди помітки однаково для усіх кон’юнктермів. Якщо n — це кількість змінних заданої булової функції, то код помітки кожного кон’юнктерма встановимо як двійкове число, що має n бітів. Одиниця у певному біті коду помітки означає, що кон’юнктерм брав участь у склеюванні по цьому біту, а нуль означає, що склеювання не було. Таким чином, код помітки мінтермів дорівнює нулю. Нагадаємо, що в методі порозрядного вирощування код помітки мінтерма, що не брав участь у склеюванні, взагалі не

існує. Визначаємо код помітки кон'юнктерма, що одержаний склеюванням, побітовою кон'юнкцією кодів помітки твірних кон'юнктермів. А у кодах помітки твірних кон'юнктермів встановлюємо одиницю у тому біті, по якому відбулося склеювання. Після завершення процедури сортування і склеювання по усіх бітах відкидаємо кон'юнктерми, що мають код помітки відмінний від нуля, оскільки вони містяться у кон'юнктермах меншого рангу. Таким чином, кон'юнктерм записуємо у множину простих кон'юнктермів, якщо його код помітки нульовий. Для числового зображення кон'юнктерма код помітки будемо записувати у верхньому індексі.

Одночасно із встановленням коду помітки кон'юнктерма встановлюємо код помітки множини кон'юнктермів як побітову кон'юнкцію його попереднього значення і зміненого унаслідок склеювання коду помітки кон'юнктерма. Якщо після завершення формування множини, її код помітки відмінний від нуля, це означає, що усі кон'юнктерми множини брали участь у склеюванні принаймні по одному спільному біту. Тоді усі ці кон'юнктерми містяться у кон'юнктермах менших рангів. А значить, цю множину потрібно усунути з розгляду.

Під час сортування запропоновано підраховувати кількість кон'юнктермів у новоутворених множинах. Якщо під час сортування по деякому біту з номером m одержано множину, що містить 2^{n-1-m} елементів, то така множина може бути замінена кон'юнктермом, у якого поглинуті усі біти старші за m.

Блок-схему удосконаленого алгоритму порозрядного вирощування наведено на рисунку 3.12. Вхідними даними є множина мінтермів, що упорядкована за зростанням числових кодів (блок DG1). Для забезпечення максимально можливого розпаралелювання обчислень опрацювання кожного вектора кон'юнктермів відбувається модулем SDG із обліком запущених копій модуля у змінній CountSDG. Блок DG2 запускає перший примірник цього модуля. Блок DG3 встановлює лічильник активних копій CountSDG в одиницю. Блок DG4 тимчасово зупиняє виконання головного потоку до завершення роботи усіх копій SDG, коли лічильник CountSDG буде скинутий у нуль. Після

обнулення лічильника здійснюється відновлення зображення кон’юнктермів, код помітки яких дорівнює нулю (блок DG5). Одержані кон’юнктерми є результуючою множиною простих кон’юнктермів заданої функції (блок DG6).

Модуль SDG першим блоком викликає підпрограму Sort, що на основі вхідної множини кон’юнктермів створює три інші множини. У першій містяться усі кон’юнктерми з нулем у молодшому біті, у другій – з одиницею, а у третій – кон’юнктерми із символом поглинання у тому ж біті. Крім того ця підпрограма забезпечує встановлення кодів помітки для кон’юнктермів і множин, а також відсікає множини з ненульовим кодом помітки. Для кожної з трьох одержаних множин у циклі SDG2 – SDG4 блок SDG3 рекурентно запускає незалежну копію модуля SDG, а блок SDG4 інкрементує лічильник запущених копій модуля. Перед завершенням модуля SDG блок SDG6 декрементує лічильник копій.

Починаючи з масиву мінтермів у розглянутому алгоритмі з кожного масиву кон’юнктермів отримують три масиви. Таким чином формують трійкове дерево, що має не більше n ярусів, де n – кількість змінних у мінтермі. На кожному ярусі кількість мінтермів зростає не більше ніж у півтора рази порівняно з попереднім ярусом. Тому максимальний обсяг пам’яті дорівнює елементу геометричної прогресії з номером n, знаменник якої 1,5, а перший елемент N, а саме: $O(N \cdot 1{,}5^n)$. Оскільки N не перевищує 2^n, то одержимо

$$O(2^n \cdot 1{,}5^n) = O(2^n \cdot (3/2)^n) = O(3^n).$$

Верхня оцінка асимптотичної складності розглянутого алгоритму розраховується як сума перших n членів зазначеної геометричної прогресії, а саме:

$$O(N \cdot (1{,}5^n\text{-}1)/(1{,}5\text{-}1)) = O(2N \cdot (1{,}5^n - 1)) = O(2 \cdot 2^n \cdot (1{,}5^n - 1))) = O(3^n - 2^n).$$

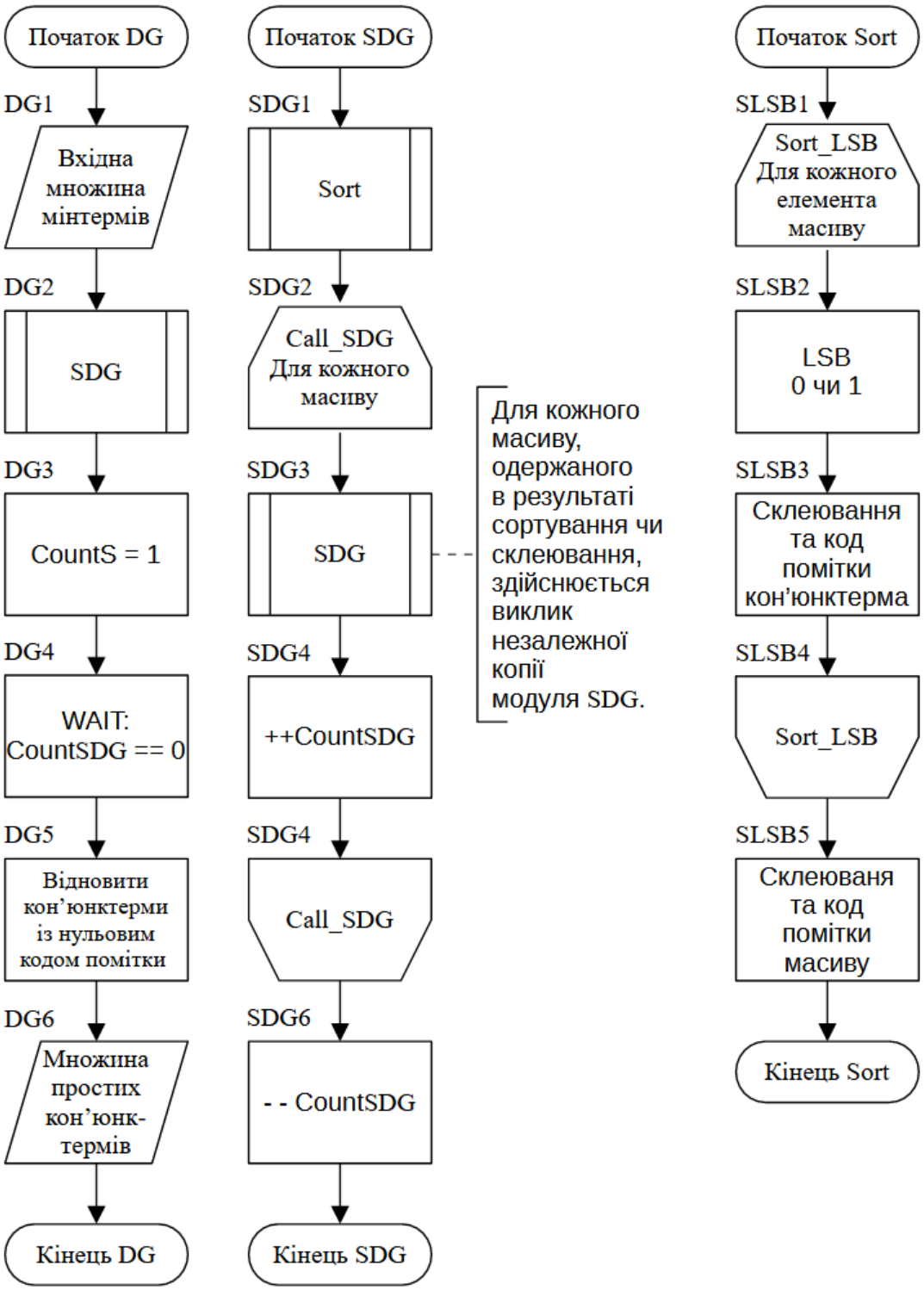


Рисунок 3.12 – Блок-схема удосконаленого алгоритму порозрядного вирощування

Вхідними даними для методу порозрядного вирощування є множина мінтермів, що упорядкована за зростанням чисел, якими ці мінтерми подають. Якщо вхідна множина не упорядкована попередньо, тоді до неї можна

застосувати метод низхідного побітового сортування зі склеюванням [99]. У цьому випадку можна виявити частину кон'юнктермів менших рангів у процесі сортування. Причому будуть знайдені усі кон'юнктерми, що мали би бути обчисленні на першому кроці методу порозрядного вирощування під час сортування за молодшим бітом.

Для мінімізації недовизначеної функції від n змінних формуємо код помітки мінтермів таким чином:

1) для мінтермів, на яких функція набуває одиничне значення, обнулюємо код помітки;

2) для мінтермів, на яких функція не визначена, встановлюємо у коді помітки усі одиниці, іншими словами присвоюємо коду помітки число 2^{n-1}.

Множину мінтермів, на якій задана функція встановлюється в одиницю об'єднуємо з множиною мінтермів, на яких функція не визначена. Упорядковуємо об'єднану множину за зростанням числових кодів мінтермів. Тепер можна подати цю множину на вхід алгоритму порозрядного вирощування.

3.3 Розроблення алгоритму пошуку простих кон'юнктермів побітовим розбиттям множини кон'юнктермів

Застосування багатьох відомих методів пошуку множини усіх простих кон'юнктермів призводить до появи тавтології. А це надлишкові обчислювальні витрати. У методі порозрядного вирощування [107] не виникає такої проблеми. Крім того у запропонованій у [100] модифікації у ряді випадків вдається зменшити кількість порівнянь. При цьому потрібно зауважити, що хоч вхідна множина мінтермів є відсортована, все одно при розгляді кожного біту доводиться пересортовувати всі наявні на цей момент кон'юнктерми. Це вимагає додаткової пам'яті та машинного часу. Аналіз алгоритму показав, що причиною тому є спосіб пошуку склеювань, що визначає порядок опрацювання бітів

починаючи з наймолодшого. Подальші розвідки щодо зменшення обчислювальних витрат призвели до розроблення алгоритму пошуку простих кон’юнктермів, що базується на кардинально відмінній процедурі пошуку склеювань, яка не потребує пересортування кон’юнктермів, а саме на процедурі розбиття масиву за старшим бітом.

Уперше запропоновано алгоритм пошуку простих кон'юнктермів булової функції побітовим розбиттям зі склеюванням множини кон’юнктермів [101]. Для подання кон’юнктермів використовується маскове зображення. Оскільки всі мінтерми мають спільну маску літералів, на вхід алгоритму подається масив чисел зі спільною маскою. Цей масив упорядковують за зростанням чисел. Якщо початкова множина чисел є мультимножиною, то після сортування масиву однакові числа опиняться у сусідніх комірках. Щоб виявити тотожні мінтерми достатньо порівняти сусідні елементи упорядкованого масиву і вилучити копії.

Нехай найбільше число у масиві має n бітів. Методом половинного січення розбиваємо масив на старший і молодший масиви. У молодшому масиві містяться усі числа менші ніж 2^{n-1}. Таким чином ми розбили множину кон’юнктермів за старшим бітом. В одній підмножині усі кон’юнктерми з нулем у старшому біті, а в другій — з одиницею.

Для кожного одержаного масиву з різниці індексів першого і останнього елементів дізнаємося чи містить масив 2^{n-1} елементів. Якщо так, то такий масив склеюється по усіх молодших бітах у один кон’юнктерм. У цьому випадку для кожного елемента другого масиву в першому знайдеться кон’юнктерм, з яким цей елемент другого масиву склеюється по біту розбиття. Тому в другому масиві в усіх кон'юнктермах поглинається старший біт.

Якщо ні старший, ні молодший масиви не склеюються, шукаємо множину кон’юнктермів, що мають символ поглинання у старшому біті. Для цього обнулюємо розглянутий біт у старшому масиві і порівнюємо найменші елементи цих масивів. Якщо вони однакові, то знайшли склеювання. Тоді беремо для порівняння наступні елементи розглянутих масивів. Якщо найменші елементи не

однакові, беремо наступний елемент у масиві того елемента, який при порівнянні виявився меншим.

У результаті одержимо три множини кон'юнктермів заданої булової функції. У одній усі кон'юнктерми з одиницею в старшому біті. У другій усі кон'юнктерми з нулем у старшому біті. У третій усі кон'юнктерми з поглинутим старшим бітом. Надалі кожну з одержаних множин розглядаємо окремо і в описаний спосіб розбиваємо за наступним бітом.

Для уникнення тавтології запропоновано формувати маску склеювань кон’юнктермів, яка є удосконаленням коду помітки, що використовується в методі порозрядного вирощування [107]. Для зменшення обчислювальних витрат уведено маску склеювань масиву кон’юнктермів, на основі якої можна вилучати з подальшого розгляду цілі масиви кон’юнктермів.

Розглянемо блок-схему алгоритму встановлення маски склеювань кон’юнктермів за деяким бітом m (рисунок 3.13). Вхідними даними є маски склеювань твірних кон’юнктермів і твірного масиву (блок МС1). Якщо унаслідок склеювання одержано новий кон’юнктерм із поглинутим бітом m, то його маска склеювань -МС є побітовою кон’юнкцією масок склеювань твірних (блок МС2):

$$-\text{МС} = 0\text{МС} \,\&\, 1\text{МС}, \tag{3.4}$$

де 0МС — маска склеювань твірного кон’юнктерма, що містить нуль у розглянутому біті,

1МС — маска склеювань твірного кон’юнктерма, що містить одиницю у розглянутому біті,

& — символ операції побітової кон’юнкції.

У масці склеювань кон’юнктерма, що містить нуль у біті m, встановлюють одиницю у цьому біті (блок МС3):

$$0\text{МС} = 0\text{МС} \mid 2^m, \tag{3.5}$$

де “ | ” — символ операції побітової диз’юнкції.

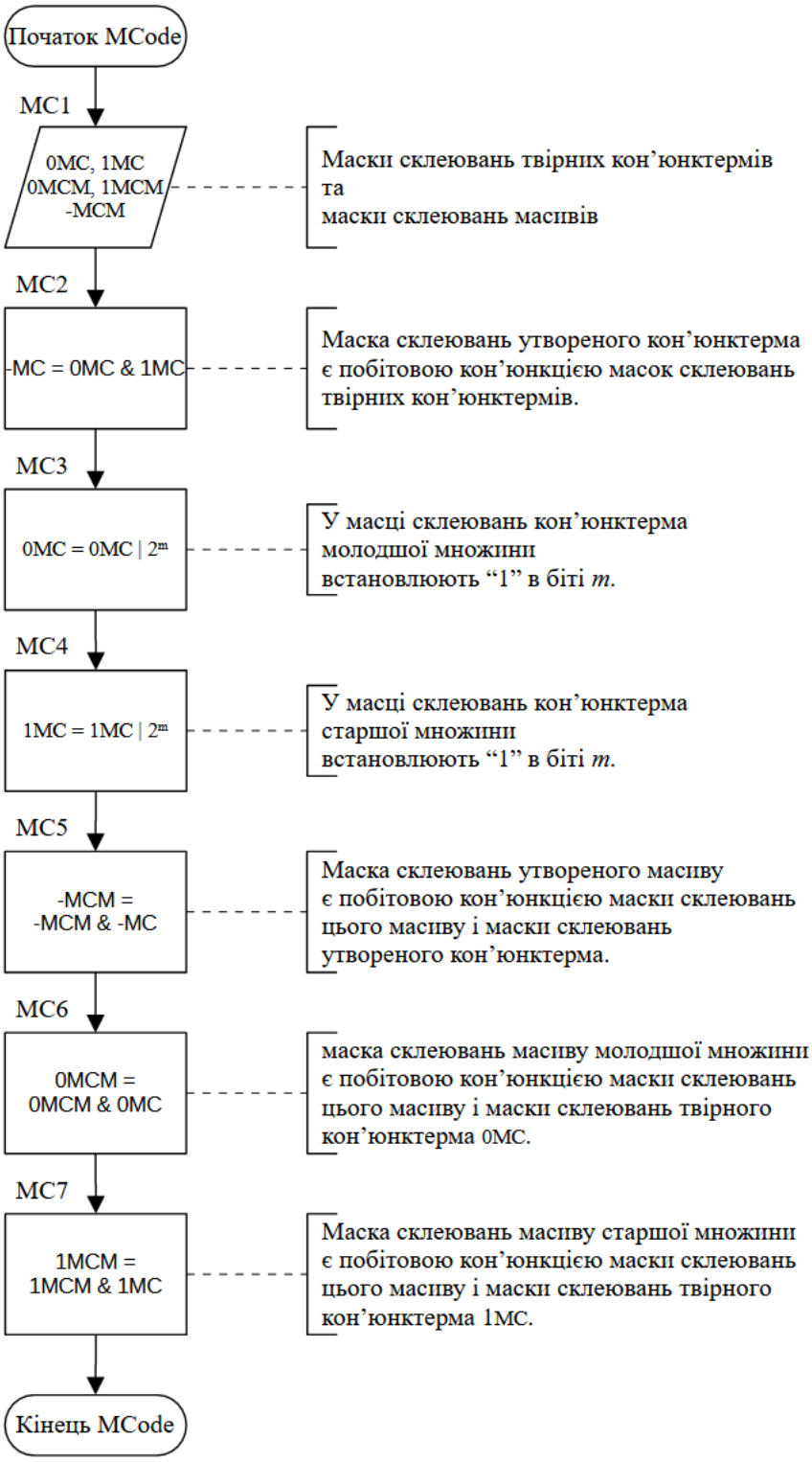


Рисунок 3.13 – Блок-схема алгоритму встановлення маски склеювань кон'юнктермів за бітом *m*

Аналогічно, блок MC4 встановлює одиницю у біті *m* маски склеювань кон'юнктерма, що містить одиницю у біті *m*.

$$1\text{MC} = 1\text{MC} \mid 2^m. \quad (3.6)$$

Маска склеювань -MCM масиву кон’юнктермів, що утворені склеюванням по біту m, одержує нове значення шляхом побітової кон’юнкції поточного значення із маскою склеювань одержаного кон’юнктерма (блок MC5):

$$- \text{MCM} = - \text{MCM} \,\&\, - \text{MC}. \quad (3.7)$$

Маска склеювань масиву, що одержаний унаслідок сортування і складається з елементів, які містять нуль у біті m, одержує нове значення шляхом побітової кон’юнкції поточного значення із маскою склеювань відповідного твірного кон’юнктерма (блок MC6):

$$0\text{MCM} = 0\text{MCM} \,\&\, 0\text{MC}. \quad (3.8)$$

Аналогічно, маска склеювань масиву, що одержаний унаслідок сортування і складається з елементів, які містять одиницю у біті m, одержує нове значення шляхом побітової кон’юнкції поточного значення із маскою склеювань відповідного твірного кон’юнктерма (блок MC7):

$$1\text{MCM} = 1\text{MCM} \,\&\, 1\text{MC}. \quad (3.9)$$

Верхня оцінка асимптотичної складності розглянутого алгоритму не залежить від кількості змінних $O(1)$.

На рисунку 3.14 наведено блок-схему алгоритму попередньої підготовки вхідних даних. У паралельних процесах опрацьовують множину мінтермів, на яких функція набуває значення одиниці, і множину мінтермів, на яких функція не визначена. Для перших встановлюється нульова маска склеювань MC, для других — $(2^n–1)$, тобто одиниці в усіх бітах. При цьому обчислюють потужність заданих множин. Потім множини об’єднують (блок MSI9) і сортують за

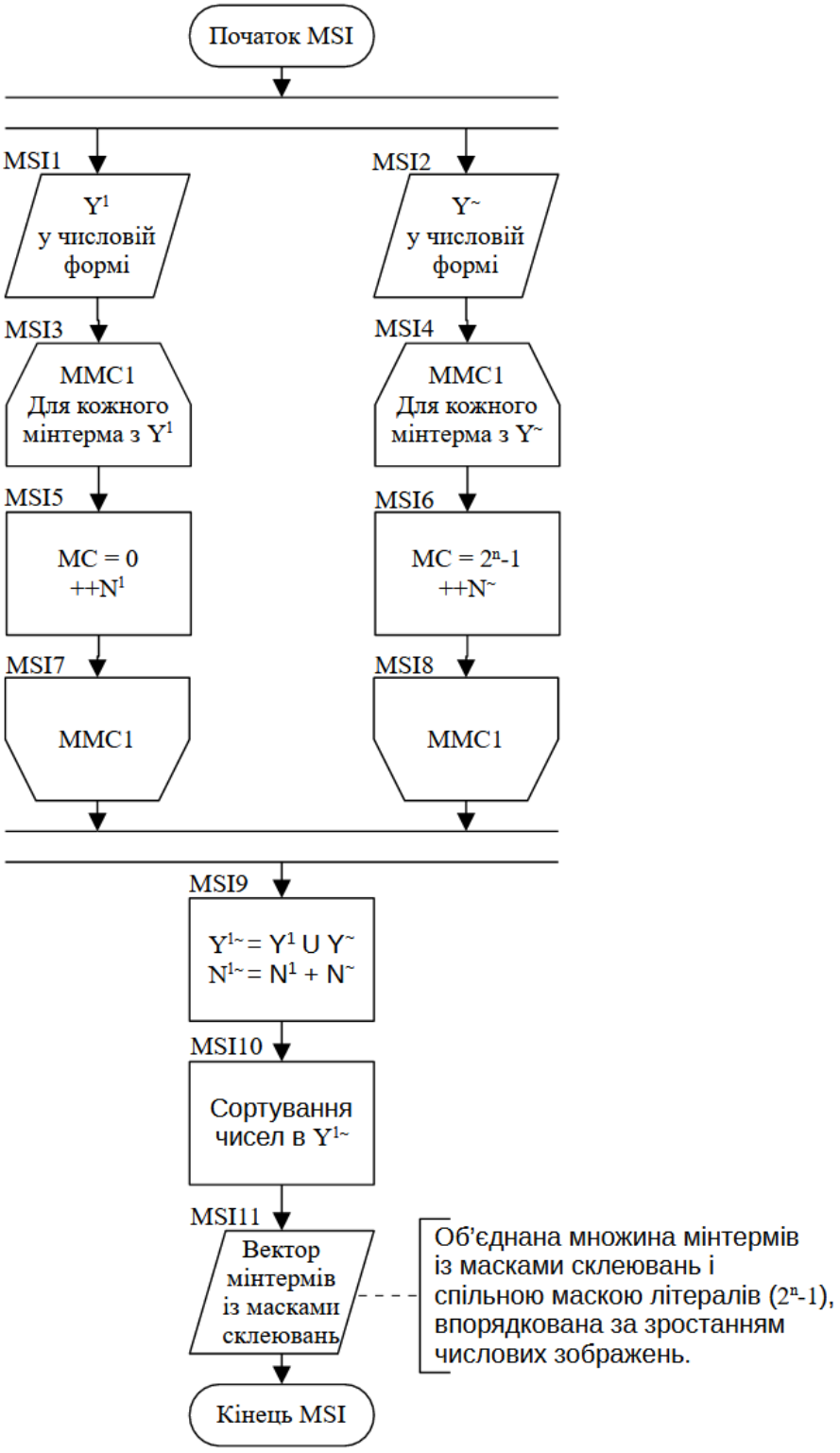


Рисунок 3.14 – Блок-схема алгоритму ініціалізації методу мінімізації побітовим розбиттям зі склеюванням

зростанням числових зображень кон'юнктермів (блок MSI10). На виході маємо впорядкований вектор мінтермів із спільною маскою літералів (2^n–1) і сформованими масками склеювань.

На рисунку 3.15 наведено блок-схему алгоритму попередньої перевірки масиву кон’юнктермів та виклик процедури розбиття масиву. Для вхідного масиву кон’юнктермів блок MSC2 оцінює кількість елементів. Якщо є тільки один елемент, то при виконанні умови MSC3 (є нульова маска склеювань)

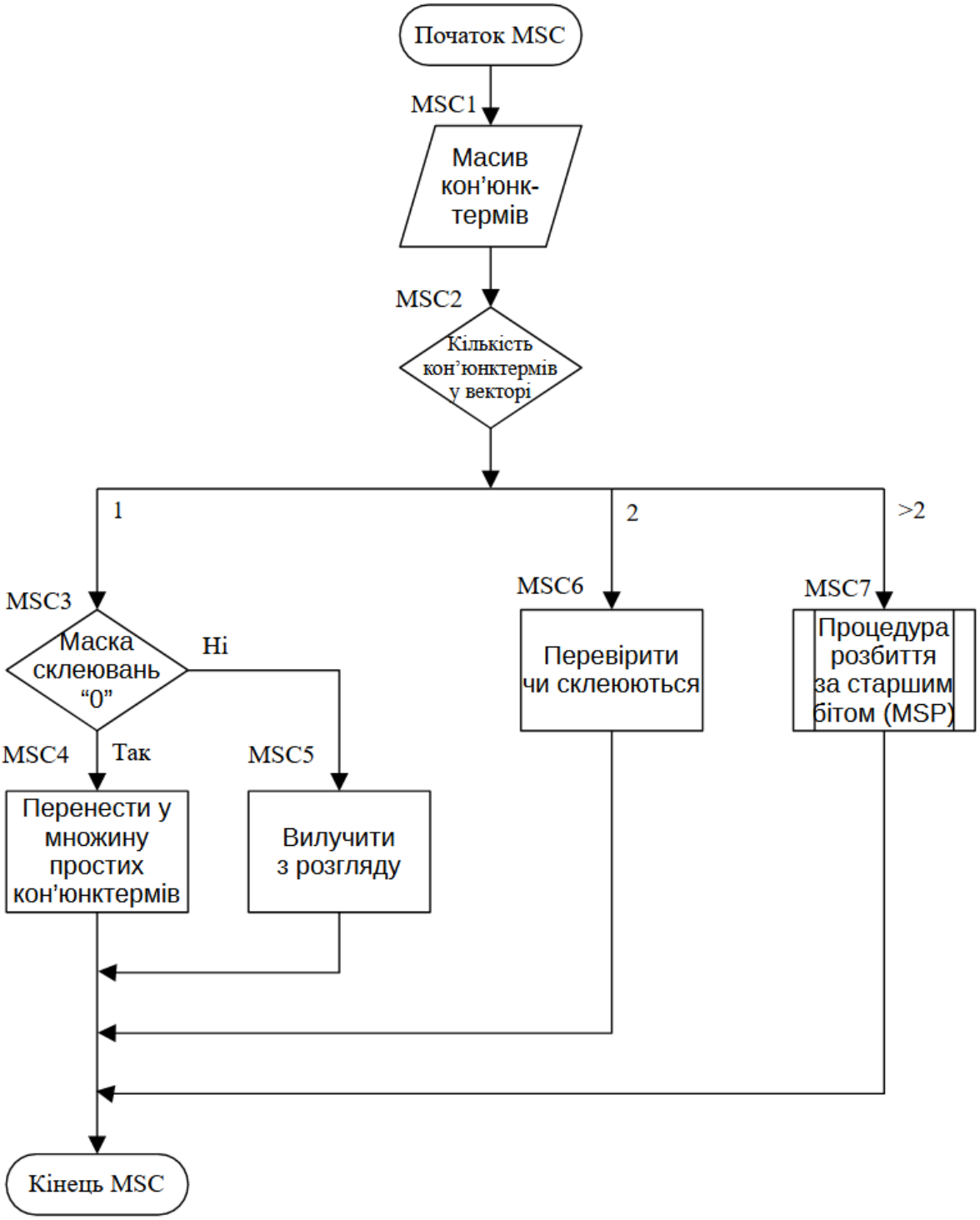


Рисунок 3.15 – Блок-схема алгоритму попередньої перевірки масиву кон’юнктермів та виклик процедури розбиття масиву

кон’юнктерм записують у множину простих кон’юнктермів. Якщо умова MSC3 не виконується, то кон’юнктерм вилучається з розгляду. Якщо блок MSC2 виявляє тільки два кон’юнктерма, то блок MSC6 просто перевіряє чи вони склеюються. Якщо блок MSC2 виявляє більше двох кон’юнктермів, тоді блок MSC7 викликає процедуру розбиття масиву за старшим бітом. Верхня оцінка асимптотичної складності розглянутого алгоритму не залежить ні від кількості змінних, ні від кількості кон’юнктермів і дорівнює $O(1)$.

На рисунку 3.16 наведено блок-схему алгоритму розбиття вектора за старшим бітом. Вхідними даними є вектор кон’юнктермів (масив). Методом половинного січення здійснюємо пошук межі між елементами, що містять нуль у біті сортування, і елементами, що містять одиницю в цьому біті. Для цього обираємо елемент у центрі масиву (блок MSP2). Перевіряємо чи цей елемент менший ніж число, що містить одиницю в біті сортування і нулі в усіх молодших бітах, тобто 2^{n-1}. Якщо ні, то обираємо молодшу частину масиву і повертаємось до MSP2. Інакше блок MSP5 перевіряє чи елемент з наступним порядковим номером менший за 2^{n-1}. Якщо так, то обираємо старшу частину масиву (блок MSP6) і повертаємось до MSP2. Інакше знайшли межу між старшим і молодшим масивами. Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(\log N)$.

Після того як знайдено множину кон’юнктермів із нулем у певному біті та множину кон’юнктермів із одиницею в цьому біті, потрібно знайти множину кон’юнктермів, у яких розглянутий біт поглинутий. Враховуючи той факт, що одержані перші дві множини відсортовані, блок-схема алгоритму формування третьої множини набуває вигляду, що зображено на рисунку 3.17.

Позначимо через c номер найбільшого елемента у молодшому масиві. Блок MSBG2 ініціалізує індекс молодшого масиву i нулем, а індекс старшого масиву j номером його молодшого елемента (c + 1). Блок MSBG3 розпочинає цикл Compare1 пошуку склеювань між елементами двох масивів по біту сортування m. Якщо різниця між елементами старшого і молодшого масивів з індексами j та i відповідно є більша за 2^m, то інкрементуємо i (блок MSBG5) та починаємо

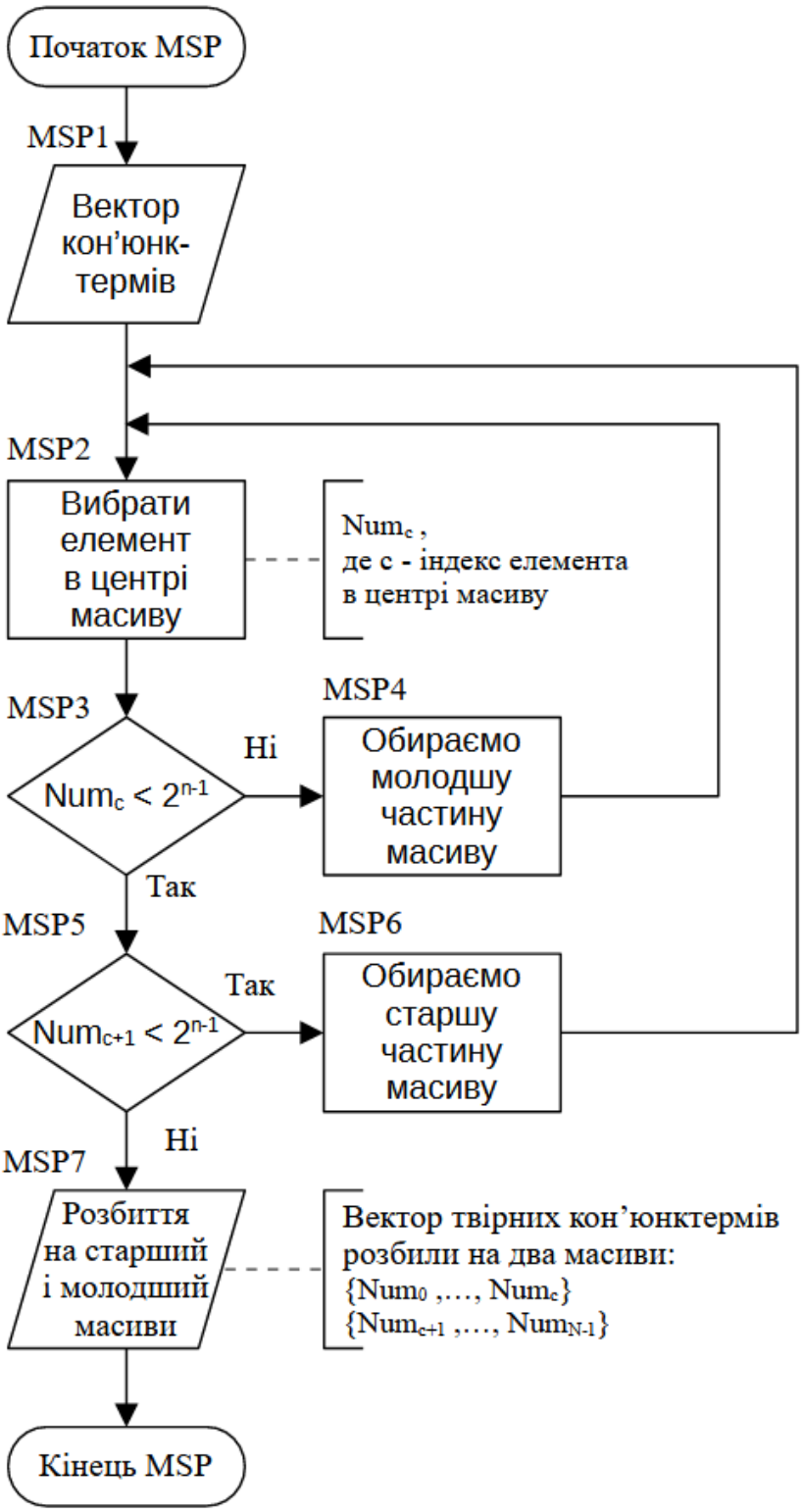


Рисунок 3.16 – Блок-схема алгоритму розбиття вектора за старшим бітом

наступну ітерацію циклу. Якщо різниця менша за 2^m, то інкрементуємо j (блок MSBG6) та переходимо до наступної ітерації циклу. Якщо різниця дорівнює 2^m, то елементи склеюються. Склеєний кон'юнктерм заносимо у вихідний вектор

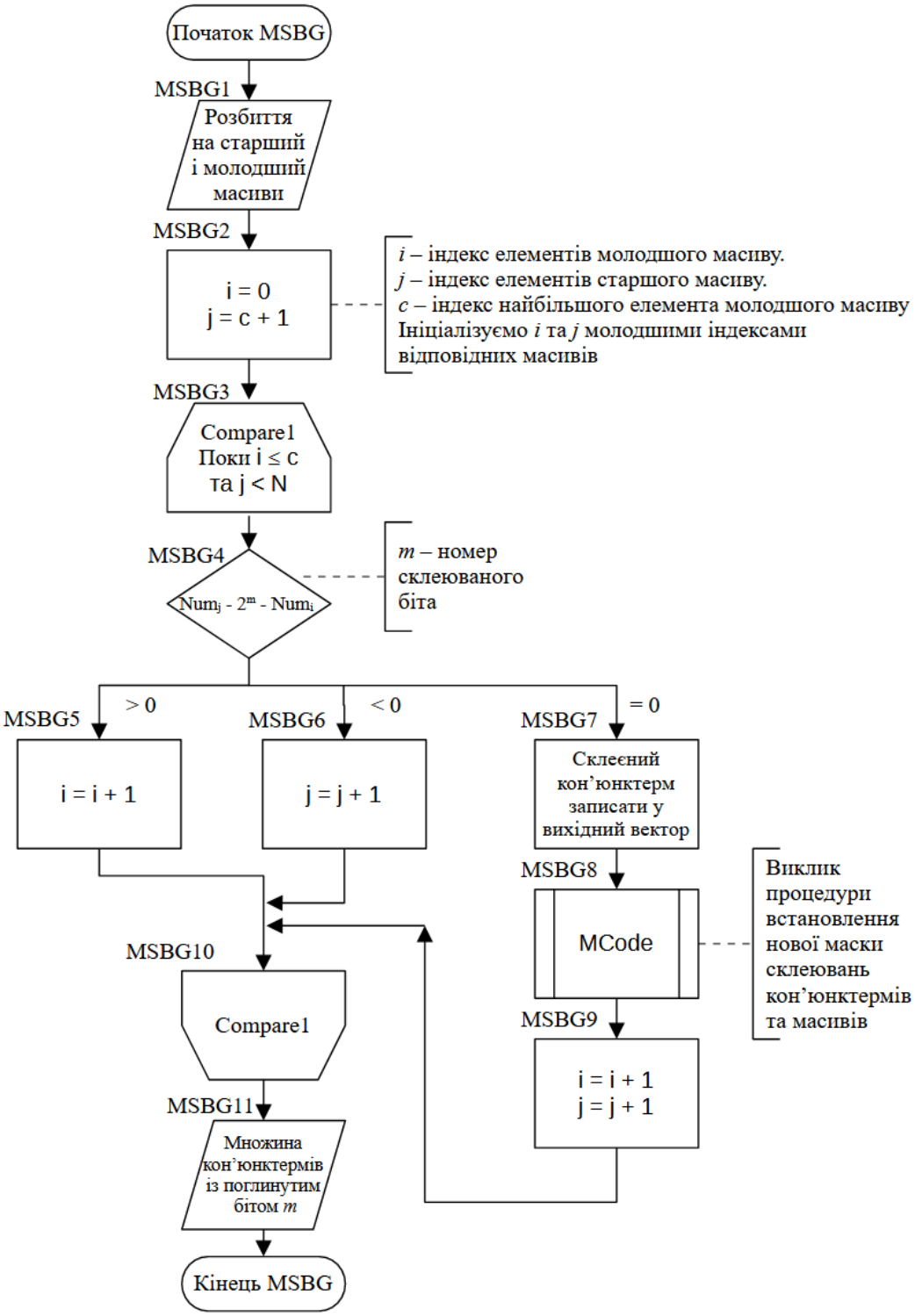


Рисунок 3.17 – Блок-схема алгоритму пошуку склеювань кон'юнктермів двох підмножин за старшим бітом

(блок MSBG7). Викликаємо модуль MCode встановлення масок склеювань (блок MSBG8). Інкрементуємо обидва індекси (блок MSBG9) і починаємо наступну ітерацію. Цикл Compare1 продовжується, поки *i* менше чи дорівнює *c*, а також *j*

менше від номера N найбільшого елемента старшого масиву. На виході (блок MSBG11) отримаємо множину кон'юнктермів, що одержана склеюванням елементів молодшого масиву з елементами старшого масиву множини, що зазнала розбиття. Верхня оцінка асимптотичної складності розглянутого алгоритму дорівнює $O(N)$.

Ключова особливість пропонованого підходу — можливість знайти шляхом розбиття масиву кон'юнктермів такий підмасив, що склеюється в один кон'юнктерм. На рисунку 3.18 наведено блок-схему алгоритму, що забезпечує таку можливість.

Блок MSG2 перевіряє чи номер найбільшого елемента молодшого масиву дорівнює 2^{n-1}, а значить цей масив склеюється в один кон'юнктерм. Аналогічно, блок MSG3 перевіряє чи склеюється старший масив. Якщо один з масивів склеївся, то у другому в усіх кон'юнктермах відповідний біт замінюється символом поглинання (блок MSG5 чи MSG9). Якщо жоден з двох масивів не склеївся, тоді блок MSG12 викликає модуль процедури склеювання за старшим бітом між елементами двох масивів.

На виході одержимо до трьох масивів. У першому всі кон'юнктерми з нулем у біті сортування. У другому всі кон'юнктерми з одиницею у біті сортування. У третьому всі кон'юнктерми з поглинутим бітом сортування.

Верхня оцінка асимптотичної складності розглянутого алгоритму визначається складністю процедури склеювання між елементами двох масивів і дорівнює $O(N)$.

Розглянемо блок-схему алгоритму пошуку простих кон'юнктермів побітовим розбиттям зі склеюванням (рисунок 3.19). Вхідними даними є множина мінтермів (блок MS1). Блок MS2 викликає модуль підготовки вхідних даних. Блок MS3 викликає першу копію модуля MSCG розбиття зі склеюванням для множини усіх мінтермів. Блок MS4 встановлює значення 1 для лічильника запущених копій модуля MSCG.

У модулі MSCG перший блок викликає модуль MSC, що здійснює попередню перевірку масиву кон'юнктермів та викликає процедуру розбиття масиву по

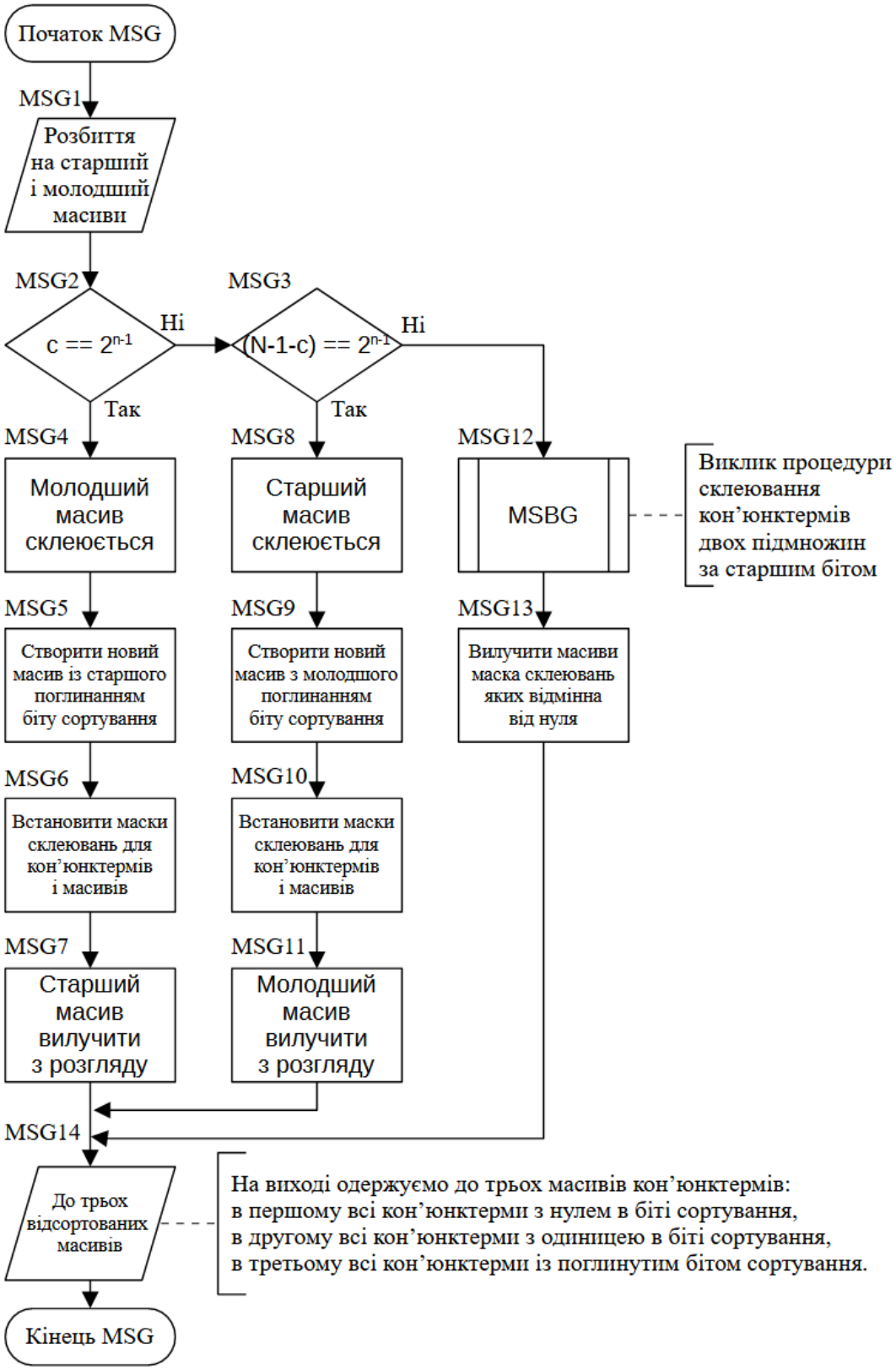


Рисунок 3.18 – Блок-схема алгоритму склеювання масиву

старшому біту. Блок MSCG2 викликає модуль MSG пошуку склеювань масиву та множини кон'юнктермів з поглинутим старшим бітом. Блок MSCG3 запускає

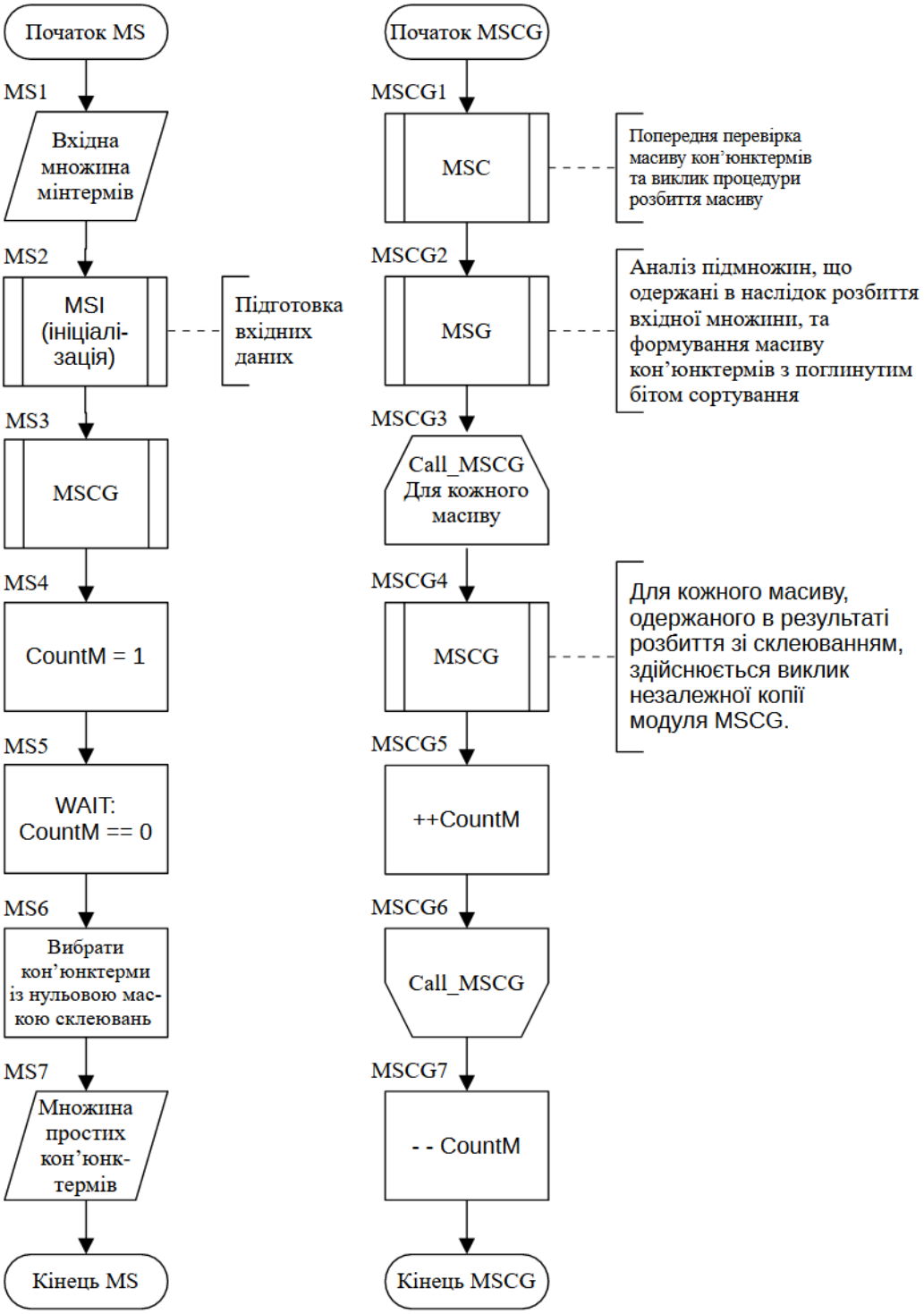


Рисунок 3.19 – Блок-схема алгоритму пошуку простих кон'юнктермів побітовим розбиттям зі склеюванням

цикл Call_MSCG, у кожній ітерації якого блоком MSCG4 рекурентно викликається незалежна копія модуля MSCG для одного з масивів, що одержані у блоці MSCG2. При цьому блок MSCG5 інкрементує лічильник CountM запущених копій модуля MSCG. Блок MSCG6 завершує ітерацію циклу

Call_MSCG і передає керування на блок MSCG3 для перевірки умови продовження циклу. Після опрацювання всіх масивів, що одержані у блоці MSCG2, цикл Call_MSCG завершується, блок MSCG7 декрементує лічильник CountM запущених копій модуля MSCG. Робота модуля завершується.

Блок MS5 очікує, поки усі запущені копії модуля MSCG завершать роботу. Блок MS6 вибирає кон'юнктерми із нульовою маскою склеювань і формує результуючу множину простих кон'юнктермів, яку виводить блок MS7.

Зауважимо, що у запропонованому алгоритмі не відбувається сортування елементів масивів. Здійснюється тільки пошук переходу певного біта з нуля в одиницю серед елементів з однаковим набором старших бітів. Усі кон'юнктерми залишаються на своїх місцях аж до вилучення з розгляду чи віднесення до остаточної множини простих кон'юнктермів. При розбитті кожного масиву може виникнути масив кон'юнктермів з поглинутим бітом розбиття. Кількість утворених елементів нового масиву не перевищує половини початкового масиву.

3.4 Розвиток алгоритму теоретико-множинної модифікації мінімаксного методу покриття булових функцій

Розглянемо алгоритм теоретико-множинної модифікації мінімаксного методу покриття множини простих кон'юнктермів булової функції, що розроблений у співавторстві із професором Рицарем Б. Є. [92]. Блок-схему цього алгоритму зображено на рисунку 3.20.

Вхідними даними алгоритму покриття є множина простих кон'юнктермів деякої булової функції *f* від *n* змінних. Блок MM1 забезпечує їх введення.

У наступному блоці MM2 відбувається формування символьних диз'юнктермів, кожен з яких являє собою множину простих кон'юнктермів, що містять певний мінтерм. Таким чином за наявності у функції *N* мінтермів отримаємо множину з *N* символьних диз'юнктермів.

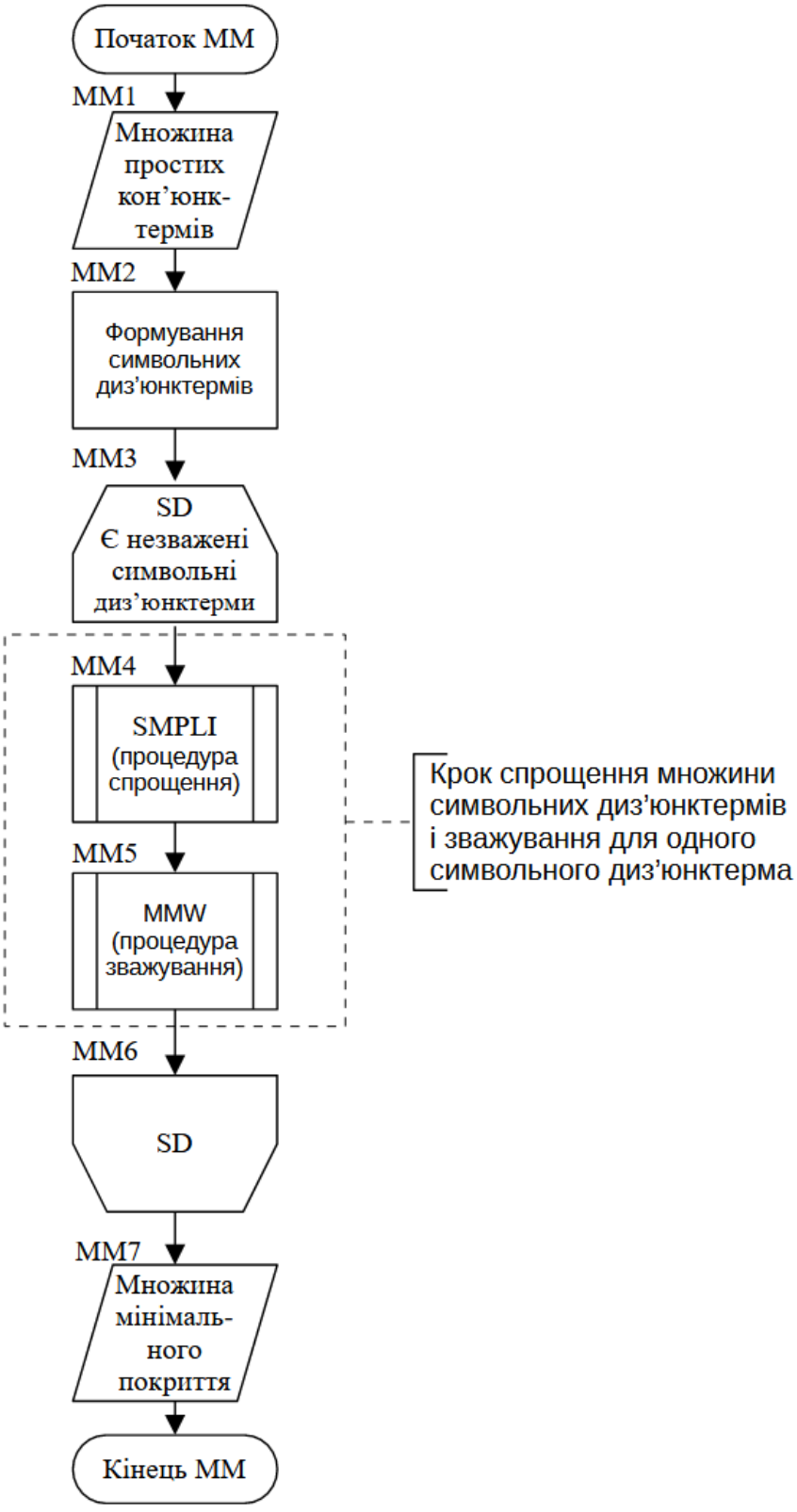


Рисунок 3.20 – Блок-схема алгоритму теоретико-множинної форми мінімаксного методу покриття

Одержана множина подається на вхід циклу SD (блок ММ3), у якому на кожній ітерації опрацьовується один символьний диз'юнктерм, а саме:

обирається істотний кон’юнктерм з числа тих, що містяться у розглянутому диз’юнктермі, шляхом послідовного виконання процедури спрощування множини символьних диз’юнктермів та процедури зважування простих кон’юнктермів символьного диз’юнктерма.

У множині символьних диз’юнктермів можуть бути такі, що містять у собі інші. Для економії обчислювальних ресурсів їх необхідно виключити з розгляду, оскільки вони будуть покриті при покритті тих диз’юнктермів, що містяться у них. Це забезпечує блок ММ4, який викликає процедуру спрощення множини символьних диз’юнктермів.

Після спрощення викликається процедура зважування, результатом виконання якої є виявлений істотний простий кон’юнктерм, що покриває розглянутий символьний диз’юнктерм. При цьому з множини символьних диз’юнктермів вилучається розглянутий диз’юнктерм, а також мінтерми, що містяться в обраному істотному кон’юнктермі. Таким чином розглянута множина символьних диз’юнктермів зазнає змін, через що у ній знову можуть з’явитись такі символьні диз’юнктерми, що містять інші. Тому на наступній ітерації циклу знов необхідно виконати процедуру спрощення множини символьних диз’юнктермів. Таким чином у кожній ітерації необхідно послідовно викликати процедури і спрощення, і зважування. Процедура зважування у цій роботі не зазнала змін. Її опис та блок-схема наведені у додатку В.

Блок ММ6 (кінець ітерації) передає управління на блок ММ3, щоб обрати наступний символьний диз’юнктерм. Цикл завершується, коли не залишилось незважених символьних диз’юнктермів. Тоді блок ММ7 виводить одержану множину істотних простих кон’юнктермів, тобто мінімальне покриття заданої булової функції. Алгоритм завершується.

Розглянемо детальніше процедуру спрощення (рисунок 3.21). Вхідними даними для цієї процедури є множина символьних диз’юнктермів (блок SMPL1).

Задана множина передається у цикл “SYMPLIFYING NxN” (блок SMPL2). На кожній ітерації обирається один з N символьних диз’юнктермів D_i, викликається вкладений цикл, після чого ітерація завершується блоком SMPL6,

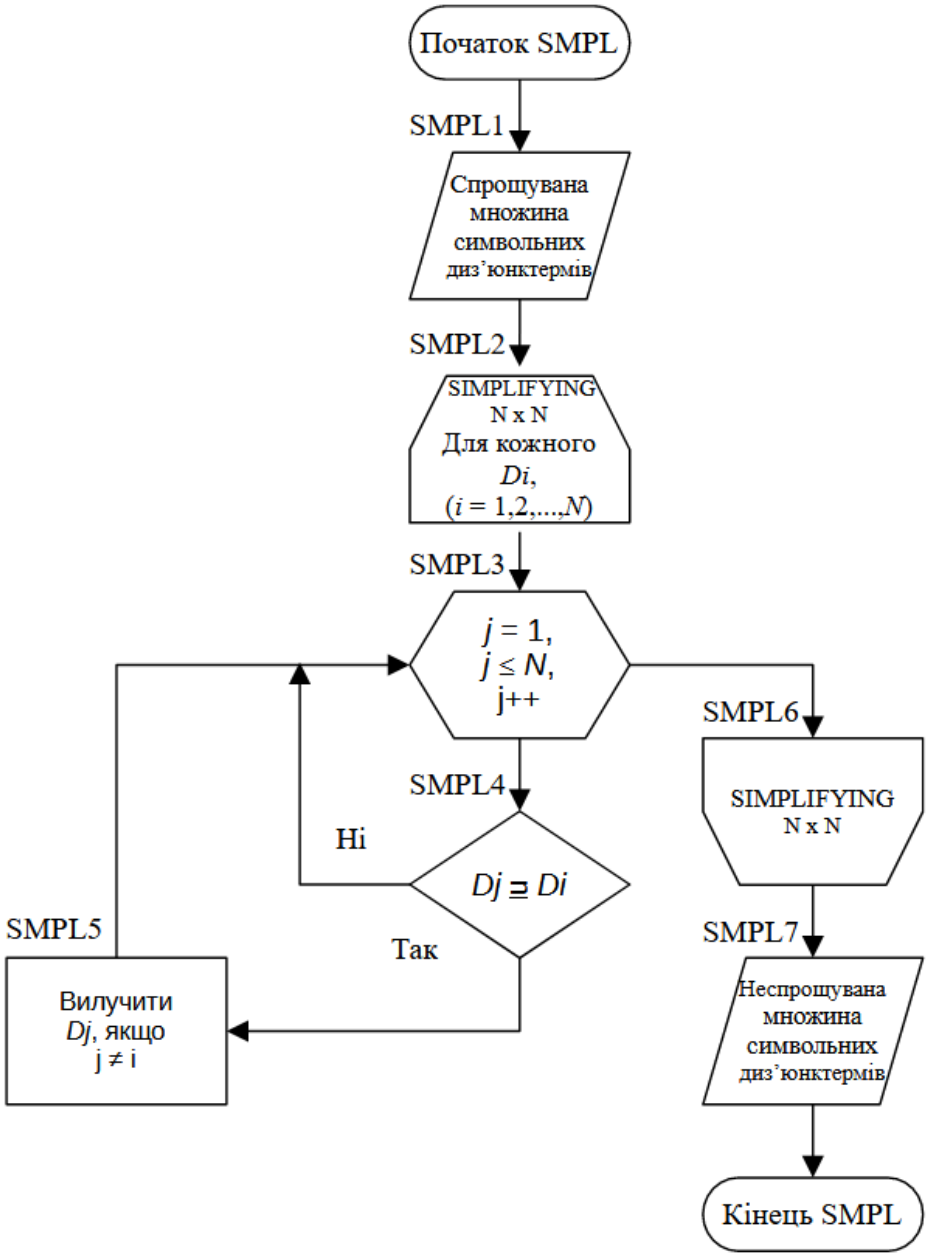


Рисунок 3.21 – Блок-схема алгоритму процедури спрощення множини символьних диз'юнктермів

який передає керування на блок SMPL2, щоб обрати наступний символьний диз'юнктерм для чергової ітерації.

Вкладений цикл SMPL3 з параметром *j* по черзі бере кожен елемент вхідної множини і у блоці SMPL4 перевіряє, чи міститься в обраному елементі D_j елемент зовнішнього циклу D_i. Якщо ні, то керування передається блоку SMPL3, бере наступний елемент внутрішнього циклу для виконання нової ітерації. Якщо ж D_j містить D_i, то відбувається перехід до блоку SMPL5, що вилучає елемент D_j з множини символьних диз'юнктермів, і переходить до наступної ітерації (блок

SMPL3). Внутрішній цикл завершується, коли виконано перевірку всіх елементів множини відносно D_i.

Цикл “SYMPLIFYING NxN” завершується, коли виконано перевірку відносно всіх символьних диз’юнктермів D_i. Тоді блок SMPL7 виводить результат процедури – неспрощувану множину символьних диз’юнктермів. Процедура завершується, керування передається в точку виклику процедури.

Очевидно, що для перевірки усіх можливих пар символьних диз’юнктермів за допомогою циклу “SYMPLIFYING NxN” необхідно здійснити N^2 порівнянь. Для зменшення кількості порівнянь введемо в алгоритм деякі зміни. Блок-схему алгоритму удосконаленої процедури спрощення множини символьних диз’юнктермів наведено на рисунку 3.22.

Одразу після уведення вхідних даних (блок SMPLI1) впорядкуємо символьні диз’юнктерми за зростанням їх потужності (кількості простих кон’юнктермів, що міститься в одному диз’юнктермі). На вхід циклу “SYMPLIFYING IMPROVED” передаємо вже впорядковану множину. Тепер у внутрішньому циклі SMPLI4 достатньо виконати (N - i) порівнянь, а не N, як в алгоритмі на рисунку 3.21. На кожній ітерації зовнішнього циклу зменшується на одиницю кількість ітерацій внутрішнього циклу. Таким чином загальна кількість порівнянь N_{SMPLI} є сумою перших N членів арифметичної прогресії 1, 2, 3, …, N:

$$N_{SMPLI} = \frac{1+N}{2} \cdot N = \frac{N^2}{2} + \frac{N}{2} \tag{3.10}$$

У разі великої потужності множини символьних диз’юнктермів, можна вважати:

$$\frac{N^2}{2} \gg \frac{N}{2} \tag{3.11}$$

Тому

$$N_{SMPLI} \approx \frac{N^2}{2} \tag{3.12}$$

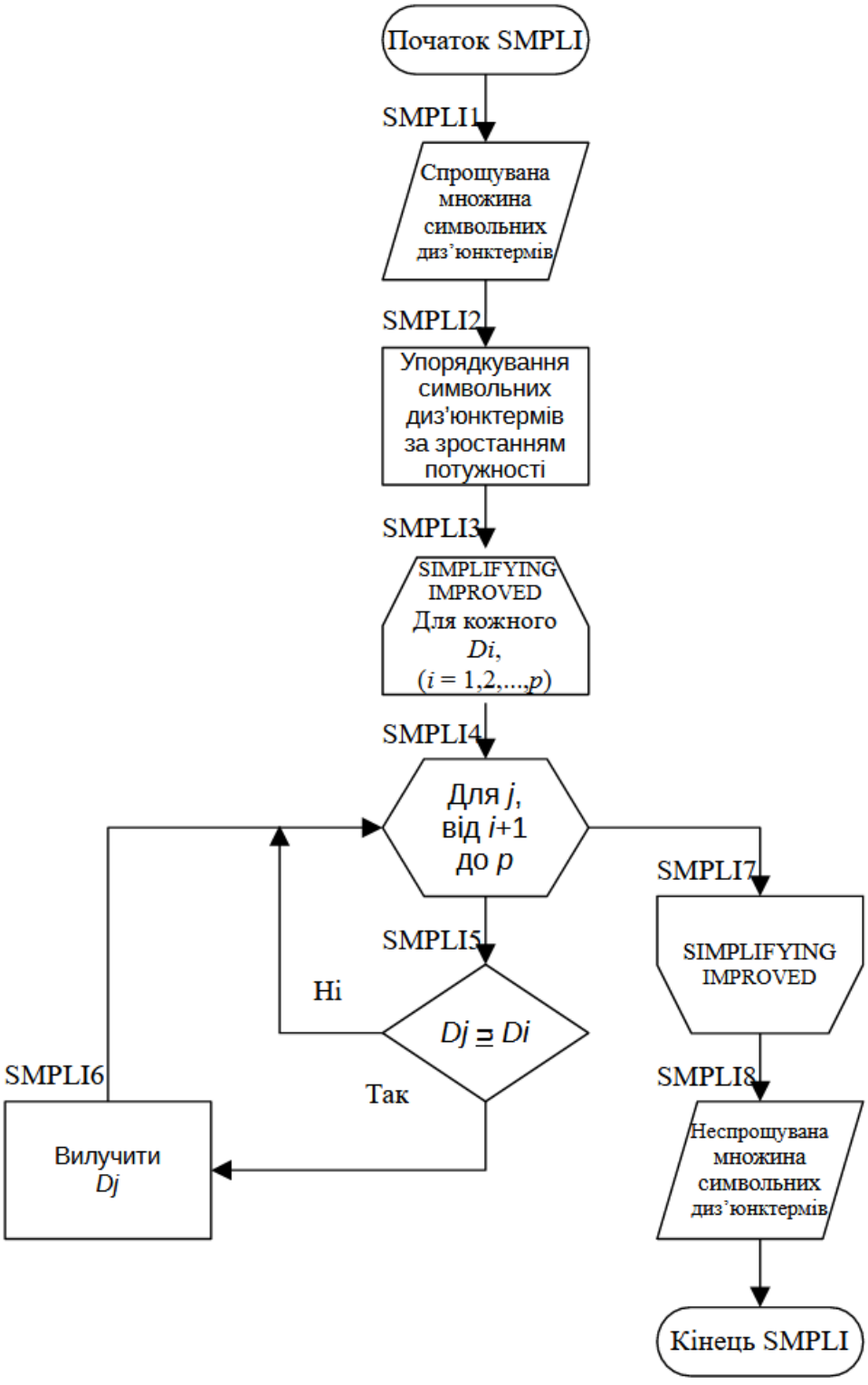


Рисунок 3.22 – Блок-схема алгоритму удосконаленої процедури спрощення множини символьних диз'юнктермів

Практично удвічі зменшується кількість операцій у процедурі спрощення, яка виконується на кожному кроці покриття.

3.5 Висновок до розділу 3

1) Удосконалено алгоритм методу розчеплення кон’юнктермів. Запропоновано виконувати процедуру розчеплення у числовій формі на основі маскового зображення кон’юнктермів. Запропоновано змінити порядок формування матриці розчеплення з метою виявлення неістотних змінних за допомогою частково сформованої матриці. Застосування запропонованої модифікації процедури розчеплення кон’юнктермів, дає змогу знизити обчислювальні витрати. Використовуючи числове подання у вісімковій чи шістнадцятковій системі числення, можна розширити коло завдань, що розв’язують “вручну”, тобто без залучення обчислювальної техніки.

2) Удосконалено алгоритм порозрядного вирощування простих кон’юнктермів. Використано розроблене маскове зображення замість псевдотрійкового, що дало змогу оперувати числовим поданням кон'юнктермів у підмножинах з однаковою маскою, уведено поняття коду помітки множини кон’юнктермів для усічення трійкового дерева вирощування простих кон’юнктермів, розроблено процедуру виявлення підмножин, елементи яких склеюються в один кон’юнктерм простим підрахунком кількості елементів одразу на етапі сортування по заданому біту, розширено область застосування методу на недовизначені функції. Запропоновані зміни дають змогу розширити коло розв’язуваних задач за рахунок зменшення обчислювальних витрат та розширення області застосування методу.

3) Уперше запропоновано алгоритм пошуку простих кон’юнктермів булових функцій побітовим розбиттям множини кон’юнктермів на основі розробленої модифікації побітового сортування зі склеюванням, що дає змогу виявляти прості кон’юнктерми низького рангу без проміжних склеювань. Цей алгоритм позбавлений тавтології.

4) Набув подальшого розвитку алгоритм мінімаксного методу покриття у теоретико-множинній формі. Для виявлення символьних диз’юнктермів, що

містять інший символьний диз’юнктерм і, відповідно, мають бути вилучені з розгляду, необхідно виконати перевірку для кожної пари символьних диз’юнктермів, тобто N^2 порівнянь, де N — кількість диз’юнктермів. Виконувана на кожному кроці покриття удосконалена процедура спрощення потребує кількість порівнянь, що дорівнює сумі перших N членів арифметичної прогресії 1, 2, 3, …, N, що дорівнює ($N^2/2 + N/2$). Оскільки можна вважати, що $N^2/2 >> N/2$, то кількість операцій у процедурі спрощення стала меншою майже удвічі.

5) Удосконалені та розроблені алгоритми можна використати для удосконалення та розроблення методів мінімізації булових функцій.

4 РЕАЛІЗАЦІЯ ТА ЗАСТОСУВАННЯ РОЗРОБЛЕНИХ АЛГОРИТМІВ МІНІМІЗАЦІЇ БУЛОВИХ ФУНКЦІЙ ДЛЯ ПРОЕКТУВАННЯ ЦИФРОВИХ КОМБІНАЦІЙНИХ ПРИСТРОЇВ

4.1 Удосконалений метод розчеплення кон’юнктермів

На основі запропонованого числового подання кон’юнктермів у вигляді маскового зображення [97, 106] та запропонованого способу виявлення та усунення неістотних змінних розроблено алгоритм удосконаленої процедури формування матриці розчеплення у масковому зображенні.

На основі запропонованого способу виявлення та вилучення ізольованих кон’юнктермів розроблено алгоритм удосконаленої процедури покриття матриці розчеплення.

Процедури формування та покриття матриці розчеплення виконують послідовно на кожному рівні розчеплення, викликають рекурентно для векторів результату покриття. Ці процедури є основою методу розчеплення кон’юнктермів. Імплементація розроблених алгоритмів удосконалених процедур в алгоритм цього методу дала змогу реалізувати **удосконалений метод розчеплення кон’юнктермів** [104], який сформулюємо так:

1) множину мінтермів функції від n змінних прийняти твірним вектором кон’юнктермів. Подати мінтерми у масковому зображенні. Масці неістотних змінних встановити значення $(2^n\text{-}1)$;

2) обчислити матрицю стовпець інверсій ваг розчеплення твірного вектора. Для цього:

2.1) у масці літералів обнулити біти неістотних змінних;

2.2) для кожного біта, у якому залишилась одиниця, записати число, що відповідає його вазі;

2.3) до одержаних чисел застосувати операцію побітової інверсії;

3) для кожного елемента матриці-стовпця інверсій ваг розчеплення обчислити вектор матриці розчеплення. Для цього:

3.1) обчислити числове зображення кожного розчепленого кон’юнктерма як побітову кон’юнкцію числового зображення твірного кон’юнктерма та інвертованої ваги розчеплення шуканого вектора;

3.2) обчислити числове зображення маски літералів розчепленого вектора як побітову кон’юнкцію числового зображення маски літералів твірного вектора та інвертованої ваги розчеплення;

3.3) виявити на розчепленому векторі всі кон’юнктерми копії;

3.4) у разі одержання повністю покритого вектора усунути неістотну змінну з розгляду. Для цього:

3.4.1) вилучити стовпці матриці розчеплення та відповідні твірні кон’юнктерми, якщо результат побітової кон’юнкції числового зображення твірного із вагою розчеплення повністю покритого вектора відмінний від нуля. (У разі мінімізації вручну для кожної пари кон’юнктермів-копій вилучити стовпець із більшим числовим зображенням твірного);

3.4.2) обчислити нове числове зображення маски літералів твірної множини як побітову кон’юнкцію поточного значення із інверсією ваги розчеплення повністю покритого вектора;

3.4.3) обчислити нове числове значення маски літералів векторів матриці розчеплення як побітову кон’юнкцію поточних значень із інверсією ваги розчеплення повністю покритого вектора;

3.4.4) обчислити нову маску неістотних змінних як побітову кон’юнкцію поточної маски із інверсією відповідної ваги розчеплення;

3.4.5) із матриці розчеплення вилучити повністю покритий вектор;

3.4.6) на уже обчислених векторах розчеплення виявити ізольовані кон’юнктерми та перенести їх до множини результату. Вилучити з матриці розчеплення відповідні стовпці;

3.4.7) якщо після скорочення матриці розчеплення якийсь із уже обчислених векторів матриці розчеплення став повністю покритим, то виконати усунення неістотної змінної (перейти до пункту 3.4);

3.4.8) перейти до обчислення наступного вектора зредукованої матриці (пункт 3.1).

3.5) на поточному векторі виявити ізольовані кон’юнктерми та перенести їх до множини результату. Вилучити з матриці розчеплення відповідні стовпці;

3.6) якщо після вилучення ізольованих кон’юнктермів якийсь з розчеплених векторів став повністю покритим, то виконати усунення неістотної змінної (перейти до пункту 3.4);

3.7) якщо залишилися неопрацьовані елементи матриці-стовпця інверсій ваг розчеплення, обчислити наступний вектор (перейти до пункту 3.1);

4) виявити вектори мінімального покриття матриці розчеплення;

5) для кожного вектора одержаного мінімального покриття матриці розчеплення запустити в незалежному процесі процедуру розчеплення (пункт 2), якщо кількість кон’юнктермів вектора більше двох, інакше перевірити пару кон’юнктермів на склеювання;

6) після завершення усіх незалежних процесів виконання процедури склеювання результуюча множина ізольованих кон’юнктермів є множиною простих кон’юнктермів заданої булової функції. Здійснити покриття цієї множини для вилучення надлишкових кон’юнктермів.

Щоб уможливити застосування удосконаленого методу розчеплення кон’юнктермів для алгоритмічного забезпечення роботи телевізійного сканувального оптичного мікроскопа під час дослідження динамічних мікрооб'єктів [94, 95] було розроблено цифровий метод формування розгортки телевізійного сканувального оптичного мікроскопа [102], у якому забезпечено

можливість формування траєкторії світної плями шляхом реалізації певних булових функцій на базі ПЛІС. Розглянемо приклад мінімізації булової функції, яка забезпечує спеціальну траєкторію розгортки для відслідковування зміни положення динамічного об’єкта певної форми за набором контрольних точок (приклад 4.1).

Приклад 4.1

Мінімізувати булову функцію від п’яти змінних, що задана множиною мінтермів у вигляді вісімкових чисел:

$$Y^1 = \{00,01,02,03,04,05,06,07,10,11,12,13,14,16,31,33,35,37\}^1$$

Розв’язання. Формуємо твірний вектор мінтермів.

$$Y^1 = \{00,01,02,03,04,05,06,07,10,11,12,13,14,16,31,33,35,37\}^1_{37/37} \overset{S}{\Rightarrow}$$

Повекторно обчислюємо матрицю розчеплення.

$$\begin{array}{c} \\ 01 \\ \overset{S}{\Rightarrow} 02 \\ 04 \\ 10 \\ 20 \end{array}\begin{bmatrix} \{00 & 01 & 02 & 03 & 04 & 05 & 06 & 07 & 10 & 11 & 12 & 13 & 14 & 16 & 31 & 33 & 35 & 37\}_{37/37} \\ \{\underline{00} & \underline{00} & \underline{02} & \underline{02} & \underline{04} & \underline{04} & \underline{06} & \underline{06} & \underline{10} & \underline{10} & \underline{12} & \underline{12} & 14 & 16 & 30 & 32 & 34 & 36\}_{36/37} \\ \{\underline{00} & \underline{01} & \underline{00} & \underline{01} & \underline{04} & \underline{05} & \underline{04} & \underline{05} & \underline{10} & \underline{11} & \underline{10} & \underline{11} & \underline{14} & \underline{14} & \underline{31} & \underline{31} & \underline{35} & \underline{35}\}_{35/37} \\ \{ & & & & & & & & & & & & & & & & & \} \\ \{ & & & & & & & & & & & & & & & & & \} \\ \{ & & & & & & & & & & & & & & & & & \} \end{bmatrix}$$

Одержано повністю покритий рядок матриці розчеплення, що свідчить про наявність несуттєвої змінної. Після її усунення і скорочення матриці розчеплення маємо:

$$Y^1 = \begin{array}{c} \\ 01 \\ 04 \\ 10 \\ 20 \end{array} \begin{array}{c} \begin{array}{ccccccccc} \{00 & 01 & 04 & 05 & 10 & 11 & 14 & 31 & 35\}_{35/37} \end{array} \\ \begin{bmatrix} \{\underline{00} & \underline{00} & \underline{04} & \underline{04} & \underline{10} & \underline{10} & 14 & 30 & 34\}_{34/37} \\ \{\underline{00} & \underline{01} & \underline{00} & \underline{01} & \underline{10} & 11 & \underline{10} & \underline{31} & \underline{31}\}_{31/37} \\ \{\underline{00} & \underline{01} & \underline{04} & 05 & \underline{00} & \underline{01} & \underline{04} & 21 & 25\}_{25/37} \\ \{00 & 01 & 04 & 05 & 10 & \underline{11} & 14 & \underline{11} & 15\}_{15/37} \end{bmatrix} \end{array} \overset{C}{\Rightarrow}$$

$$\overset{C}{\Rightarrow} \{00,04,10\}_{34/17}, \{00,01,10,31\}_{31/17} \overset{S}{\Rightarrow}$$

$$\overset{S}{\Rightarrow} \begin{array}{c} \\ 04 \\ 10 \end{array} \begin{array}{c} \begin{array}{ccc} \{00 & 04 & 10\}_{34/17} \end{array} \\ \begin{bmatrix} \{\underline{00} & \underline{00} & 10\}_{30/3} \\ \{\underline{00} & 04 & \underline{00}\}_{24/3} \end{bmatrix} \end{array}, \begin{array}{c} \\ 01 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccc} \{00 & 01 & 10 & 31\}_{31/17} \end{array} \\ \begin{bmatrix} \{\underline{00} & \underline{00} & 10 & 30\}_{30/6} \\ \{\underline{00} & 01 & \underline{00} & 21\}_{21/6} \end{bmatrix} \end{array} \overset{C}{\Rightarrow}$$

$$\overset{C}{\Rightarrow} \{0\}_{30}, \{0\}_{24}, \{0\}_{21}, \{31\}_{31} =$$

$$= (00---), (0-0--), (0---0), (11--1).$$

У заданій функції від п’яти змінних присутні 18 мінтермів. Без мінімізації її кошт реалізації становить 108 літералів (18 кон’юнктермів по 5 змінних). Унаслідок мінімізації удосконаленим методом розчеплення кон’юнктермів отримали форму заданої функції, що має кошт реалізації 13 літералів.

З використанням удосконаленого методу розчеплення кон’юнктермів у ТзОВ «Міта-техніка» було розв’язано практичне завдання проектування універсального сигнального перетворювача промислової автоматики на основі мережі первинних перетворювачів із оцифрованими вихідними сигналами. Ефект, отриманий від впровадження запропонованого удосконаленого методу на етапі синтезу цифрової комбінаційної схеми, полягає у зменшенні на 5% кількості комірок ПЛІС, які необхідні для реалізації потрібних у завданні булових функцій, що дало змогу використати вивільнені ресурси для реалізації

інших функцій системи, зокрема функції самоконтролю, що підтверджує відповідний акт, наведений у додатку А.

Удосконалений метод розчеплення кон’юнктермів було використано для синтезу перетворювача кодів при виконанні держбюджетної науково-дослідної роботи ДБ/ВЕРБАЛЬ (держреєстр. № 0104U002291) у Національному університеті „Львівська політехніка”, що підтверджено відповідним актом.

4.2 Розвиток теоретико-множинної модифікації мінімаксного методу покриття булових функцій

На основі удосконаленої процедури спрощення множини символьних диз’юнктермів та внесених змін до алгоритму ТМФ мінімаксного методу покриття сформульовано розвинуту теоретико-множинну модифікацію мінімаксного методу покриття булових функцій [92], а саме:

1) сформувати усі символьні диз’юнктерми;

2) впорядкувати символьні диз’юнктерми за зростанням їх потужності (кількості простих кон'юнктермів, що міститься у кожному диз'юнктермі);

3) виконати процедуру спрощення множини символьних диз'юнктермів (вилучити ті диз'юнктерми, що містять інші символьні диз’юнктерми);

4) обрати символьний диз'юнктерм найменшої потужності та виконати процедуру зважування. (У множині простих кон'юнктермів обраного диз'юнктерма знайти кон'юнктерм найменшої потужності, перенести його як істотний у множину мінімального покриття. Вилучити його мінтерми з подальшого розгляду інших простих кон'юнктермів. Вилучити з розгляду символьний диз'юнктерм.);

5) якщо залишились нерозглянуті символьні диз'юнктерми, перейти до пункту 3, інакше покриття завершено.

Розглянемо приклад обчислення покриття булової функції за допомогою запропонованої удосконаленої теоретико-множинної модифікації мінімаксного методу покриття.

Приклад 4.2

Визначити мінімальне покриття булової функції, що задана множиною простих кон'юнктермів:

Y^1 = {(30,22,14,6),(18,16),(30,26,22,18),(8,0),(30,26,14,10),(17,16),(6,4), (29,25,21,17),(16,0),(4,0),(29,21,13,5),(10,8),(5,4),(29,25,13,9),(9,8)}[1].

Розв'язання. Позначимо прості кон'юнктерми заданої булової функції латинськими літерами.

Таблиця кодування простих кон'юнктермів символами наведена на рисунку 4.1.

Створюємо символьні диз'юнктерми:

$0 \in (P,N,M)$, $4 \in (P,L,K)$, $5 \in (L,D)$, $6 \in (K,A)$, $8 \in (N,J,I)$, $9 \in (J,E)$,
$10 \in (I,B)$, $13 \in (E,D)$, $14 \in (B,A)$, $16 \in (M,H,G)$, $17 \in (H,F)$, $18 \in (G,C)$,
$21 \in (F,D)$, $22 \in (C,A)$, $25 \in (F,E)$, $26 \in (C,B)$, $29 \in (F,E,D)$, $30 \in (C,B,A)$.

Спрощуємо множину символьних диз'юнктермів за допомогою операції поглинання:

$$\{(L,D),(K,A),(J,E),(I,B),\underline{(E,D)},\underline{(B,A)},(H,F),(G,C),(F,D),(C,A),(F,E),$$
$$(C,B),(P,N,M),(P,L,K),(N,J,I),(M,H,G),\underline{(F,E,D)},\underline{(C,B,A)}\}=$$
$$=\{(L,D),(K,A),(J,E),(I,B),(E,D),(B,A),(H,F),(G,C),(F,D),(C,A),$$
$$(F,E),(C,B),(P,N,M),(P,L,K),(N,J,I),(M,H,G)\}$$

(30,22,14,6)	*A*
(30,26,14,10)	*B*
(30,26,22,18)	*C*
(29,21,13,5)	*D*
(29,25,13,9)	*E*
(29,25,21,17)	*F*
(18,16)	*G*
(17,16)	*H*
(10,8)	*I*
(9,8)	*J*
(6,4)	*K*
(5,4)	*L*
(16,0)	*M*
(8,0)	*N*
(4,0)	P

Рисунок 4.1 – Таблиця кодування простих кон’юнктермів символами

Тепер необхідно спростити множину символьних диз’юнктермів шляхом зважування простих кон’юнктермів, що належать одному диз’юнктерму. Здійснимо це у декілька етапів.

Етап 1. У спрощеному виразі візьмемо перший диз’юнктерм, потужність якого більша від одиниці (тобто він містить більше одного простого кон’юнктерма). Це елемент (*L*,*D*). Згідно з формулою (1.14) обчислюємо потужності для *L* і *D*:

$$P_L^1 = 2,$$

$$P_D^1 = 4.$$

Маємо

$$P_L^1 < P_D^1,$$

тому для покриття обираємо *D*:

$$(L, D) \Rightarrow (D).$$

Зауважимо, що (*D*) поглинає (*E*,*D*), (*F*,*D*), тому їх можна відкинути як поглинуті. Тепер множина символьних кон'юнктермів набуває вигляду:

$$\{(D), (K, A), (J, E), (I, B), (B, A), (H, F), (G, C), (C, A), (F, E),$$

$$(C, B), (P, N, M), (P, L, K), (N, J, I), (M, H, G)\};$$

Етап 2. Наступний візьмемо диз'юнктерм (*K*,*A*) одержаного виразу. Потужності його символьних кон'юнктермів *K* та *A* становлять $P_K^2 = 2$, $P_A^2 = 4$. Маємо $P_K^2 < P_A^2$, тому для покриття обираємо *A*: $(K, A) \Rightarrow (A)$. Множина символьних диз'юнктермів набуває вигляду:

$$\{(D), (A), (J, E), (I, B), (H, F), (G, C), (F, E), (C, B),$$

$$(P, N, M), (P, L, K), (N, J, I), (M, H, G)\};$$

Етап 3. Визначаємо потужність елементів наступного символьного диз'юнктерма (*J*,*E*). Зауважимо, що істотний символьний кон'юнктерм *D* містить мінтерми (13) і (29). Тому на цьому етапі покриття $P_{13}^3 = P_{29}^3 = 0$, а значить $P_J^3 = 2, P_E^3 = 2$.

Отримали однакові значення потужностей $P_J^3 = P_E^3$, тому відкладемо прийняття рішення про визначення істотного кон'юнктерма у диз'юнктермі (*J*,*E*).

Візьмемо до розгляду наступний символьний диз'юнктерм (*I*,*B*) та визначимо потужності його елементів: $P_I^3 = 2$, $P_B^3 = 2$, оскільки $P_{14}^3 = P_{30}^3 = 0$. Тож, знову отримали однакові потужності $P_I^3 = P_B^3$.

Наступний символьний диз'юнктерм – (*H*,*F*). Для нього $P_{21}^3 = P_{29}^3 = 0$, тому $P_H^3 = P_F^3 = 2$.

Для (*G*,*C*) $P_{22}^3 = P_{30}^3 = 0$, тому $P_G^3 = P_C^3 = 2$.

Для (*F*,*E*) $P_{13}^3 = P_{21}^3 = P_{29}^3 = 0$, тому $P_F^3 = P_E^3 = 2$.

Для (*C*,*B*) $P_{14}^3 = P_{22}^3 = P_{30}^3 = 0$, тому $P_C^3 = P_B^3 = 2$.

Отже, ми отримали циклічну частину, що складається з елементів, які мають однакові потужності:

$$P_J^3 = P_I^3 = P_H^3 = P_G^3 = P_F^3 = P_E^3 = P_C^3 = P_B^3.$$

Приймаємо істотним довільний символьний кон'юнктерм циклічної частини. Нехай це буде *J*. Тепер можна виконати поглинання, внаслідок якого одержимо:

$$\{(D), (A), (J), (I, B), (H, F), (G, C), (F, E),$$

$$(C, B), (P, N, M), (P, L, K), (M, H, G)\};$$

Етап 4. Беремо наступний символьного диз'юнктерма (*I*,*B*), для якого $P_8^4 = P_{14}^4 = P_{30}^4 = 0$, тому $P_I^4 = 1$, $P_B^4 = 2$. Звідси $P_I^4 < P_B^4$, $(I, B) \Rightarrow (B)$. Тепер можна здійснити поглинання. Одержимо:

$$\{(D), (B), (J), (B), (H, F), (G, C), (F, E), (P, N, M), (P, L, K), (M, H, G)\};$$

Етап 5. Наступний символьний диз'юнктерм (H,J). Для нього одержимо $P_H^5 = P_F^5 = 2$. Тому переходимо до наступного диз'юнктерма (I,M): $P_G^5 = 2$, $P_C^5 = 1$, $P_G^5 > P_C^5$, $(G, C) \Rightarrow (G)$. Одержимо

$$\{(D), (A), (J), (B), (G), (H, F), (F, E), (P, N, M), (P, L, K)\};$$

Етап 6. Для (H,F) маємо: $P_H^6 = 1$, $P_F^6 = 2$, $P_H^6 < P_F^6$, $(H, F) \Rightarrow (F)$. Звідси

$$\{(D), (A), (J), (B), (G), (F), (P, N, M), (P, L, K)\};$$

Етап 7. Для (P,N,M): $P_P^7 = 2$, $P_N^7 = P_M^7 = 1$, $P_P^7 > P_N^7 = P_M^7$, $(P, N, M) \Rightarrow (P)$. Звідси

$$\{(D), (A), (J), (B), (G), (F), (P)\}.$$

Таким чином отримали множину істотних кон'юнктермів, що є одним із декількох розв'язків мінімального покриття:

$Y^1 = \{P, J, G, F, D, B, A\}^1$ ={(4,0), (9,8), (18,16), (29,25,21,17), (29,21,13,5), (30,26,14,10), (30,22,14,6)}[1].

Якщо обирати у точці розгалуження інші прості кон'юнктерми, то можна одержати інші розв'язки покриття.

4.3 Удосконалений метод порозрядного вирощування

Запропоновано внести зміни та доповнення в алгоритм методу порозрядного вирощування, а саме використовувати числове подання кон'юнктермів у вигляді маскового зображення [97], формувати коди помітки

однаково для усіх кон’юнктермів [107], встановлювати код помітки множини кон’юнктермів, під час сортування визначати підмножини, що склеюються в один кон'юнктерм, розширити область застосування на недовизначені функції.

На основі змін та доповнень внесених в алгоритм методу порозрядного вирощування запропоновано удосконалений метод порозрядного вирощування простих кон'юнктермів [100]:

1) сформувати маскове зображення мінтермів булової функції від *n* змінних;

2) встановити нулі в усіх бітах коду помітки мінтермів, на яких функція набуває одиничне значення;

3) встановити одиниці в усіх бітах коду помітки мінтермів, на яких функція не визначена;

4) об’єднати дві сформовані множини в одну. Відсортувати мінтерми за зростання числових зображень;

5) встановити одиниці в усіх бітах числового зображення маски літералів об’єднаної множини;

6) встановити нулі в усіх бітах коду помітки об’єднаної множини;

7) із тих бітів, що ще не розглядалися, аналізувати молодший біт кон’юнктермів. Необхідно утворити три множини, а саме: у першій – кон’юнктерми з нулем у розглянутому біті, у другій – з одиницею, у третій – із поглинутим бітом. Для цього:

7.1) встановити лічильники кон’юнктермів множин у нуль; встановити коди помітки множин у нуль;

7.2) почати з кон’юнктерма з найменшим числовим зображенням;

7.3) якщо у розглянутому біті кон’юнктерма нуль, то перевірити чи не склеюється цей кон’юнктерм з наступним. Якщо склеїлись, встановити код помітки одержаного кон’юнктерма як побітову кон’юнкцію кодів помітки твірних, а у відповідному біті кодів помітки твірних встановити одиницю. Твірні кон’юнктерми та новоутворений занести у відповідні множини. Інкрементувати лічильники множин. Розрахувати нове значення

коду помітки кожної множини як побітову кон’юнкцію поточного значення і коду помітки відповідного кон’юнктерма. Твірні кон’юнктерми вилучити з твірної множини;

7.4) якщо у розглянутому біті кон’юнктерма одиниця, перемістити кон’юнктерм у відповідну множину, інкрементувати лічильник множини, розрахувати нове значення коду помітки множини як побітову кон’юнкцію поточного значення і коду помітки відповідного кон’юнктерма;

7.5) взяти для аналізу наступний кон’юнктерм твірної множини (перейти до пункту 7.3);

8) коли у твірній множині закінчились кон’юнктерми, аналізувати код помітки новоутворених множин. Якщо код помітки множини не дорівнює нулю, то множину вилучають з розгляду;

9) якщо у лічильнику множини встановилось число (2^{n-1-m}) (де m – номер розглянутого біта), то замість множини записують кон’юнктерм, у якого поглинуто усі біти, що старші за m. Цей кон'юнктерм перенести у множину простих кон’юнктермів.

10) у незалежних процесах проаналізувати новоутворені множини за наступним бітом. Якщо у новоутвореній множині тільки два кон’юнктерма, то перевірити цю пару на склеювання.

11) після завершення аналізу всіх утворених множин до множини простих кон’юнктермів перенести одержані кон’юнктерми, що мають нульовий код помітки.

Розглянемо приклад мінімізації булової функції удосконаленим методом порозрядного вирощування.

Приклад 4.3

Знайти множину простих кон’юнктермів булової функції від чотирьох змінних, що задана множиною мінтермів, на яких набуває значення «1», і множиною мінтермів, на яких не визначена.

$$\begin{cases} Y^1 = \{(11), (9), (8), (7), (6), (4), (2), (1)\}^1 \\ Y^{\sim} = \{(15), (3), (0)\}^{\sim} \end{cases}$$

На рисунку 4.2 зображено трійкове дерево вирощування простих кон'юнктермів заданої булової функції. У дужках позначено числові зображення кон'юнктермів, у верхньому індексі — коди помітки кон'юнктермів і множин, у нижньому індексі — числове зображення маски літералів.

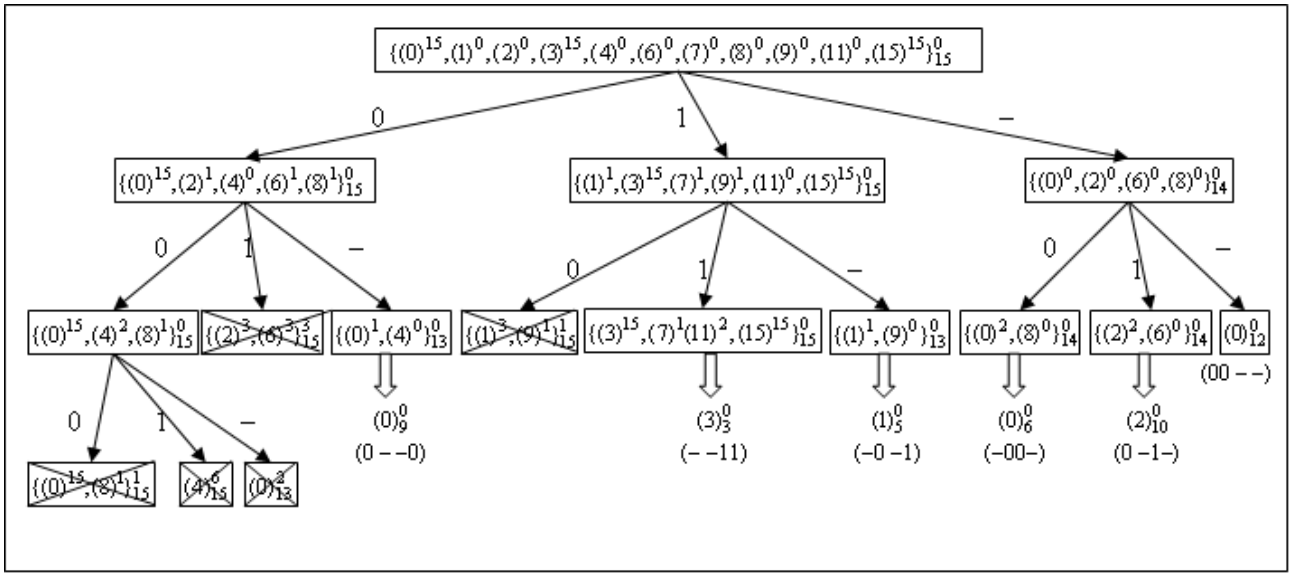


Рисунок 4.2 – Трійкове дерево визначення простих кон'юнктермів [100]

На верхньому ярусі трійкового дерева розташована об'єднана множина заданих мінтермів. Її сортування за молодшим бітом дає нам відповідні дві множини другого ярусу. Третя множина другого ярусу отримана шляхом послідовної перевірки мінтермів на склеювання. Множини, що вилучаються з розгляду через ненульовий код помітки, зображено перекресленими. На кінцях гілок дерева залишились одиночні кон'юнктерми. Ті з них, що мають нульовий код помітки, є простими. Для них відновлено псевдотрійкове зображення:

$$Y^1 = \{(00--), (0-1-), (-00-), (-0-1), (--11), (0--0)\}^1.$$

Запропонований удосконалений метод порозрядного вирощування порівняно з немодифікованим забезпечив одержання результату шляхом вирощування трійкового дерева, що має менше на п’ять гілок, відповідно, потребував менших обчислювальних витрат. Крім того удосконалений метод дає змогу мінімізувати недовизначені функції.

Удосконалений метод порозрядного вирощування було використано для синтезу перетворювача кодів при виконанні держбюджетної науково-дослідної роботи ДБ/ВЕРБАЛЬ (держреєстр. № 0104U002291) у Національному університеті „Львівська політехніка”, що підтверджено відповідним актом (додаток А).

4.4 Метод мінімізації булових функцій побітовим розбиттям множини кон’юнктермів та покриттям ланцюгами

Мінімізації булових функцій переважно здійснюється у два етапи. На першому етапі здійснюють пошук множини простих кон’юнктермів. У відомих методах на цьому етапі з’являється тавтологія. Враховуючи велику розмірність сучасних задач тавтологія призводить до значного зростання обчислювальних витрат.

Другим етапом шукають мінімальне покриття одержаної множини простих кон’юнктермів, що є задачею неполіномного типу, а значить такою що має комбінаторну складність. Для розв’язання неполіномних задач застосовують степеневі алгоритми [52, 92].

У [101] запропоновано метод пошуку простих кон’юнктермів побітовим розбиттям множини, у якому на відміну від відомих способів [52] можна одержати кон’юнктерми вищих рангів не здійснюючи проміжні склеювання. Крім того у запропонованому методі не виникає тавтологія.

У [98] запропоновано варіант спрощення задачі покриття циклічної частини таблиці простих кон'юнктермів за умови, що у ній присутні ланцюгові функції чи їх фрагменти. Як розвиток цієї ідеї запропоновано ідею покриття ланцюгами кон'юнктермів [103].

Об'єднання методу ланцюгового покриття та методу пошуку простих кон'юнктермів побітовим розбиттям множини дало метод мінімізації булових функцій, у якому не виникає тавтологія. Такий метод буде корисним для проектування цифрових комбінаційних пристроїв.

Запропонований алгоритм пошуку простих кон'юнктермів побітовим розбиттям зі склеюванням множини кон'юнктермів, реалізований у відповідному методі пошуку простих кон'юнктермів [101], та запропонований спосіб покриття таблиці простих кон'юнктермів ланцюгами [98] стали основою методу мінімізації булових функцій побітовим розбиттям [103]. Запропонований метод мінімізації є двоетапним.

Перший етап: пошук простих кон'юнктермів побітовим розбиттям множини заданих кон'юнктермів.

Нехай двома множинами мінтермів задано недовизначену булову функцію від n змінних. Функція набуває одиничне значення на наборах з першої множини і не визначена на наборах з другої множини. Номери змінних у кожному наборі збігаються з номерами бітів у числах, якими подано мінтерми. Одиницями закодовано прямі значення змінних, а нулями – інверсні. Для кожного кон'юнктерма першої множини встановлено нульову маску склеювань. Для кожного кон'юнктерма другої множини встановлено маску склеювань 2^{n-1}. Задані множини об'єднуємо в одну і впорядковуємо елементи за зростанням кодів. Присвоюємо масці склеювань упорядкованої множини число 2^{n-1}.

1) Аналізуємо множину:

- у разі ненульової маски склеювань множину вилучають з розгляду;

- якщо є тільки один елемент із нульовою маскою склеювань, то переносимо його до результуючої множини простих кон'юнктермів;

- якщо є два елементи, то перевіряємо чи вони не склеюються;

- якщо є більше двох елементів, то відправляємо їх у процедуру розбиття множини.

2) Процедура розбиття виконується по найстаршому біту m з тих, по яких розбиття ще не відбувалося. Для цього методом половинного січення шукаємо елемент менший від 2^m, щоб наступний після нього дорівнював 2^m або був більший. Тоді у множині молодших чисел будуть усі кон'юнктерми, що мають нуль у розглянутому біті, а в іншій – ті, що мають одиницю.

3) Якщо у кожній підмножині менше ніж 2^m кон'юнктермів, то переходимо до пункту 4. Інакше замінюємо підмножину, що має 2^m кон'юнктермів, одним кон'юнктермом нижчого рангу з поглинутими бітами, що молодші за m. При цьому в усіх кон'юнктермах іншої підмножини у біті m встановлюємо символ поглинання.

Встановлюємо маски склеювань утворених множин. Для цього з множини кон'юнктермів із поглинутим бітом m беремо молодший елемент. У другій множині починаємо пошук з молодшого елемента і знаходимо той, що поглинається обраним елементом першої множини, для якого встановлюємо нову маску склеювань як побітову кон'юнкцію поточного значення із маскою знайденого кон'юнктерма. Після цього у масці склеювань знайденого кон'юнктерма встановлюємо одиницю в біті m. Тоді для наступного елемента першої множини шукаємо у другій множині поглинутий елемент, починаючи з наступного елемента. Процедура закінчується, коли опрацьовано всі елементи першої множини. Тоді необхідно перейти до пункту 5.

4) Якщо у кожній підмножині менше ніж 2^m кон'юнктермів, то обчислюємо множину кон'юнктермів із поглинутим бітом m. Для цього візьмемо по молодшому елементу в кожній підмножині і віднімемо від більшого менший.

- Якщо різниця склала 2^m, то елементи склеюються. Встановлюємо маску склеювань утвореного кон'юнктерма як побітову кон'юнкцію масок склеювань твірних. У біт m масок склеювань твірних заносимо одиницю.

- Якщо різниця менша за 2^m, то беремо для порівняння наступний елемент множини старших чисел.

- Якщо різниця більша за 2^m, то беремо для порівняння наступний елемент множини молодших чисел.

Порівняння продовжуються поки в одній з підмножин не залишиться елементів.

5) Результатом процедури розбиття за бітом m є три нові множини:

- у першій кон'юнктерми з нулем у біті m;

- у другій кон'юнктерми з одиницею у біті m;

- у третій кон'юнктерми із поглинутим бітом m.

Маска склеювань кожної множини є побітовою кон'юнкцією масок склеювань кон'юнктермів цієї множини.

У разі склеювання множини у кон'юнктерм, його маска склеювань дорівнює масці склеювання відповідної множини.

Множини з ненульовою маскою склеювань вилучають з розгляду.

Аналіз кожної з отриманих множин відбувається у незалежному процесі по наступному молодшому біту (m–1).

6) Після завершення опрацювання усіх створених множин до множини простих кон'юнктермів записують кон'юнктерми з нульовою маскою склеювань.

Приклад 4.4

Знайти множину простих кон'юнктермів недовизначеної булової функції від чотирьох змінних:

$$\begin{cases} Y^1 = \{(11), (9), (8), (7), (6), (4), (2), (1)\}^1 \\ Y^{\sim} = \{(15), (3), (0)\}^{\sim} \end{cases}.$$

Розв'язання. Продублюємо числове подання псевдотрійковим для кращого розуміння. У псевдотрійковому поданні зверху зазначимо маски склеювань у двійковій формі.

Встановлюємо нульову маску склеювань 0 (0000) кон’юнктермам з Y^1.

Встановлюємо маску склеювань 15 (1111) кон’юнктермам з $Y^{\sim}$, бо вони неістотні.

Тепер потрібно об’єднати множини кон’юнктермів Y^1 та $Y^{\sim}$.

Упорядковуємо за зростанням елементи об’єднаної множини.

Встановлюємо нульову маску склеювань 0 (0000) об’єднаної множини.

$$\underset{1111}{\{(15)^{15}}\ \underset{0000}{(11)^0},\ \underset{0000}{(9)^0},\ \underset{0000}{(8)^0},\ \underset{0000}{(7)^0},\ \underset{0000}{(6)^0},\ \underset{0000}{(4)^0},\ \underset{1111}{(3)^{15}},\ \underset{0000}{(2)^0},\ \underset{0000}{(1)^0},\ \underset{1111}{(0)^{15}}\}^0$$
$$\{(1111),\ (1011),\ (1001),\ (1000),\ (0111),\ (0110),\ (0100),\ (0011),\ (0010),\ (0001),\ (0000)\}^{0000}$$

Розбиваємо множину за старшим бітом (третім бітом). У двійковому поданні видно, що межа проходить між кон’юнктермами (1000) та (0111). У числовому зображенні до підмножини молодших чисел відносять ті, що менші від $2^3 = 8$. Одержуємо дві підмножини:

$$\underset{1111}{\{(15)^{15}},\ \underset{0000}{(11)^0},\ \underset{0000}{(9)^0},\ \underset{0000}{(8)^0}\},\ \{\underset{0000}{(7)^0},\ \underset{0000}{(6)^0},\ \underset{0000}{(4)^0},\ \underset{1111}{(3)^{15}},\ \underset{0000}{(2)^0},\ \underset{0000}{(1)^0},\ \underset{1111}{(0)^{15}}\}^0$$
$$\{(1111),\ (1011),\ (1001),\ (1000)\},\ \{(0111),\ (0110),\ (0100),\ (0011),\ (0010),\ (0001),\ (0000)\}^{0000}$$

Тепер необхідно знайти множину кон’юнктермів, які склеїлися по третьому біту. Обираємо у кожній з підмножин наймолодший елемент. Маємо (8) і (0) або у двійковій формі (1000) та (0000) у більшому числі обнулюємо третій біт. Одержали однакові числа (0000) і (0000), тоді обрані кон’юнктерми склеюються у (–000). У числовій формі від більшого числа потрібно відняти $2^3 = 8$ і відняти менше число. Маємо 8 – 0 – 8 = 0, тоді існує кон’юнктерм (8,0).

Для кон’юнктерма (–000) обчислюємо маску склеювань як побітову кон’юнкцію масок склеювань твірних:

$$0000\&1111=0000,$$

де “&” — символ бінарної операції побітової кон’юнкції.

Для твірних кон’юнктермів змінюємо маски склеювань. Встановлюємо одиницю в третьому біті. Тепер маємо $\overset{0000}{(-000)}$, $\overset{1000}{(1000)}$, $\overset{1111}{(0000)}$ або у числовому поданні $(8,0)^0$, $(8)^8$, $(0)^{15}$.

Беремо наступні елементи у кожній множині: $\overset{0000}{(1001)}$ та $\overset{0000}{(0001)}$. Для них одержимо $\overset{0000}{(-001)}$, $\overset{1000}{(1001)}$, $\overset{1000}{(0001)}$.

Наступні елементи: $\overset{0000}{(1011)}$ та $\overset{0000}{(0010)}$. Або у числовому поданні $(11)^0$ та $(2)^0$. Перевіряємо: 11 – 2 – 8 = 1 > 0, тому беремо наступний елемент множини молодших чисел. Маємо: 11 – 3 – 8=0. Кон’юнктерми склеїлися: $(11,3)^0$, $(11)^8$, $(3)^{15}$. Порівняння продовжується поки в обох множинах є елементи. Унаслідок склеювання за третім бітом одержимо.

$$\{(15)^{15},\ (11)^8,\ (9)^8,\ (8)^8\}^8,\quad \{(7)^8,\ (6)^0,\ (4)^0,\ (3)^{15},\ (2)^0,\ (1)^8,\ (0)^{15}\}^0$$
$$\overset{1111}{\{(1111)},\ \overset{1000}{(1011)},\ \overset{1000}{(1001)},\ \overset{1000}{(1000)}\}^{1000},\quad \overset{1000}{\{(0111)},\ \overset{0000}{(0110)},\ \overset{0000}{(0100)},\ \overset{1111}{(0011)},\ \overset{0000}{(0010)},\ \overset{1000}{(0001)},\ \overset{1111}{(0000)}\}^{0000}$$

$$\{(15,7)^0,\ (11,3)^0,\ (9,1)^0,\ (8,0)^0\}^0$$
$$\overset{0000}{\{(-111)},\ \overset{0000}{(-011)},\ \overset{0000}{(-001)},\ \overset{0000}{(-000)}\}^{0000}$$

Отримали три множини. У першій кон’юнктерми мають нуль у третьому біті, у другій — одиницю, у третій — поглинуту змінну.

Розрахуємо маску склеювань кожної множини шляхом побітової кон’юнкції масок склеювання кон’юнктермів, що належать множині. Для множини кон’юнктермів з одиницею в третьому біті отримаємо у двійковому вигляді 1000. Вилучаємо з розгляду множину з ненульовою маскою. Дві інші множини мають нульові маски склеювань. Надалі можна опрацьовувати ці множини незалежно одна від одної. Наступною розглянемо

$$\{(7)^8, (6)^0, (4)^0, (3)^{15}, (2)^0, (1)^8, (0)^{15}\}^0.$$

Розіб’ємо за другим бітом:

$$\underset{1000}{\{(7)^8},\quad \underset{0000}{(6)^0},\quad \underset{0000}{(4)^0}\},\quad \underset{1111}{\{(3)^{15}},\quad \underset{0000}{\{(2)^0},\quad \underset{1000}{(1)^8},\quad \underset{1111}{(0)^{15}}\}$$
$$\{(0111),\quad (0110),\quad (0100)\},\quad \{(0011)\},\quad \{(0010),\quad (0001),\quad (0000)\}$$

У множині $\{(3)^{15}, (2)^4, (1)^8, (0)^{15}\}$ є 2^2 кон'юнктермів, тому вона склеюється у кон'юнктерм (3,2,1,0) ((00– –) — у псевдотрійковому зображенні). Тоді у кон'юнктермах другої множини другий біт поглинається. (У числовому поданні кожен кон'юнктерм необхідно доповнити числом, що на 2^2 менше від тих, що у ньому уже є.) Отримаємо другу множину:

$$\underset{1000}{\{(7,3)^8},\quad \underset{0000}{(6,2)^0},\quad \underset{0000}{(4,0)^0}\}$$
$$\{(0-11),\quad (0-10),\quad (0-00)\}.$$

Тепер необхідно обчислити маски склеювань кон'юнктермів обох множин. Беремо молодший елемент другої множини $\overset{0000}{(0-00)}$ ($(4,0)^0$). У першій множині знаходимо $\overset{1111}{(0000)}$ ($(0)^{15}$). Побітова кон'юнкція масок склеювань становить: 1111&0000 = 0000. Маємо нову маску склеювань кон'юнктерма старшої множини. А в маску склеювань відповідного кон'юнктерма молодшої множини записуємо одиницю у другому біті. Після опрацювання множин одержимо:

$$\underset{1000}{\{(7,3)^8},\quad \underset{0000}{(6,2)^0},\quad \underset{0000}{(4,0)^0\}^0},\quad \underset{1111}{\{(3)^{15}},\quad \underset{0100}{\{(2)^4},\quad \underset{1000}{(1)^8},\quad \underset{1111}{(0)^{15}\}^0}$$
$$\{(0-11),\quad (0-10),\quad (0-00)\}^{0000},\quad \{(0011),\quad (0010),\quad (0001),\quad (0000)\}^{0000}$$

Встановлюємо маску склеювань кон'юнктерма (3,2,1,0) такою ж як маска склеювань відповідної множини, тобто 0. Тому цей кон'юнктерм є простим, а множину, яку він замінив, потрібно вилучити з розгляду.

Другу множину $\{(7,3)^8, (6,2)^0, (4,0)^0\}^0$ розіб'ємо за першим бітом. До молодшої підмножини відносимо кон'юнктерми, у яких наймолодше число менше 2^1.

$$\underset{1000}{\{(7,3)^8},\quad \underset{0000}{(6,2)^0\}},\quad \underset{0000}{\{(4,0)^0\}}$$
$$\{(0-11),\quad (0-10)\},\quad \{(0-00)\}$$

У другій множині 2^1 кон'юнктермів. Отже, існує кон'юнктермом (0–1–) (або у числовому поданні (7,6,3,2)). Значить кон'юнктерм молодшої множини має поглинутий перший біт. Після обчислення масок склеювань маємо два простих кон'юнктерми:

$$(7,6,3,2)^0\ ,\quad (6,4,2,0)^0$$
$$\underset{0000}{(0-1-)},\quad \underset{0000}{(0--0)}$$

Наступною розглядаємо множину, що була отримана склеюванням за третім бітом $\{(15,7)^0, (11,3)^0, (9,1)^0, (8,0)^0\}^0$. Розбиваємо за другим бітом

$$\{(15,7)^0,\quad (11,3)^0,\quad (9,1)^0\},\quad \{(8,0)^0\}$$
$$\underset{0000}{\{(-111)},\quad \underset{0000}{(-011)},\quad \underset{0000}{(-001)}\},\quad \underset{0000}{\{(-000)\}}$$

Для перевірки склеювань з числових зображень необхідно брати найменші числа. Одержимо склеювання:

$$\{(15,11,7,3)^0\}^0\quad \{(15,7)^4\}^4,\quad \{(11,3)^4,\quad (9,1)^0,\quad (8,0)^0\}^0$$
$$\underset{0000}{\{(--11)\}^{0000}}\quad \underset{0100}{\{(-111)\}^{0100}},\quad \underset{0100}{\{(-011)},\quad \underset{0000}{(-001)},\quad \underset{0000}{(-000)\}^{0000}}$$

Друга множина має ненульову маску склеювань, тому буде вилучена з розгляду. У третій множині є тільки один елемент $(--11)$ ((15,11,7,3) — у числовому поданні). Його маска склеювань дорівнює нулю, значить це простий кон'юнктерм.

Розіб'ємо за першим бітом множину $\{(11,3)^4, (9,1)^0, (8,0)^0\}^0$:

$$\{(11,3)^4 \quad (9,1)^0\}, \quad \{(8,0)^0\}$$

$$\overset{0100}{\{(-011)} \quad \overset{0000}{(-001)}\}, \quad \overset{0000}{\{(-000)\}}$$

Підмножина молодших елементів містить 2^1 кон'юнктермів, тому маємо (– 00 –) ((9,8,1,0) — у числовому зображенні). У кон'юнктермі іншої множини поглинається перший біт. У них нульові маски склеювань, тому кон'юнктерми прості:

$$(11,9,3,1)^0, \quad (9,8,1,0)^0$$

$$\overset{0000}{(-0-1)}, \quad \overset{0000}{(-00-)}.$$

У псевдотрійковому поданні множина простих кон'юнктермів заданої функції:

$$\{(-0-1), (-00-), (--11), (0--0), (0-1-), (00--)\}^1,$$

у числовому поданні:

$$\{(11,9,3,1), (9,8,1,0), (15,11,7,3), (6,4,2,0), (7,6,3,2), (3,2,1,0)\}^1.$$

На відміну від методу порозрядного вирощування [107] у методі побітового розбиття не потрібно на кожному кроці сортувати кожну множину. Потрібно тільки здійснити пошук межі за заданим бітом між кон'юнктермами, що мають нуль у цьому біті, і кон'юнктермами, що мають одиницю у цьому біті.

Для реалізації комп'ютерних обчислень потрібно використовувати маскове зображення кон'юнктермів [97]. У разі, якщо вхідна множина мінтермів не відсортована за зростанням, потрібно застосувати метод побітового

сортування зі склеюванням [100]. Це дасть змогу одержати деякі кон’юнктерми нижчих рангів ще під час впорядкування множини мінтермів.

У розробленому методі мінімізації побітовим розбиттям зі склеюванням має місце швидке розбиття задачі на незалежні підзадачі меншої розмірності. Ці підзадачі можна виконувати одночасно у паралельних процесах.

Виконано оцінку алгоритмічної складності розробленого методу. Та надано рекомендації щодо області застосування. Для подальших міркувань вважатимемо, що розглядаємо множину з N кон’юнктермів булової функції від n змінних. Верхня оцінка складності алгоритму одержання множини кон’юнктермів із поглинутим бітом розбиття за умови існування кон’юнктерма з поглинутими молодшими бітами дорівнює $O(\log N)$.

У іншому випадку, тобто якщо функція не містить кон’юнктерма з поглинутими бітами молодшими за біт розбиття, оскільки елементи розташовані у порядку наростання, для одержання множини кон’юнктермів із поглинутим бітом розбиття потрібно не більше N попарних порівнянь кон'юнктермів, а верхня оцінка складності алгоритму в цьому випадку дорівнює $O(N)$.

У загальному випадку розбиття масиву кон'юнктермів по кожній змінній призводить до появи масиву кон'юнктермів, у яких ця змінна поглинута. Причому кількість елементів нового масиву не перевищує половини кількості твірного масиву, тобто сумарна кількість елементів усіх масивів зростає не більше ніж в півтора рази. Впорядкована множина кон'юнктермів не потребує пересортування і, відповідно, додаткових обсягів пам'яті. Тому максимальна кількість кон’юнктермів розраховується як елемент із номером $(n+1)$ геометричної прогресії, знаменник якої (1,5), а перший елемент дорівнює N :

$$N\,(1{,}5)^n. \qquad (4.1)$$

Для верхньої оцінки можна прийняти $N = 2^n$. Тоді потрібно врахувати, що у випадку склеювання масиву в кон’юнктерм, вивільняється пам’ять, у якій зберігався твірний масив. При цьому загальний обсяг зайнятої пам'яті не зростає.

Тому максимальний обсяг пам’яті дорівнює елементу геометричної прогресії з номером n, знаменником прогресії (1,5), перший елемент якої ($2^n - 2^{n-1} = 2^{n-1}$):

$$2^{n-1} \cdot (1,5)^{n-1} = 2^{n-1} \cdot (3/2)^{n-1} = 3^{n-1} \tag{4.2}$$

Отже, верхня оцінка обсягу необхідної пам'яті дорівнює $O(3^{n-1})$.

Склеювання по нульовому біту будуть виявлені при опрацюванні біту (n-1). Тому максимальна кількість опрацювань кон'юнктермів дорівнює сумі перших (n-1) елементів зазначеної вище геометричної прогресії:

$$2^{n-1} \cdot \frac{(1,5)^{n-1} - 1}{(1,5) - 1} = 2^n \cdot ((1,5)^{n-1} - 1) = 2 \cdot 3^{n-1} - 2^n = \\ = 2 \cdot (3^{n-1} - 2^{n-1}). \tag{4.3}$$

Максимальна кількість порівнянь між елементами масивів молодших і старших чисел дорівнює сумарній кількості елементів у них. Але така кількість порівнянь може відбутися тільки за умови, що елементи не склеїлись. Тоді не виникає третій масив, що містить склеєні елементи, і кількість операцій на наступному кроці не збільшується. Щоб отримати масив склеєних елементів максимально можливої потужності мають склеїтись усі пари елементів твірних масивів. При цьому кількість операцій порівняння елементів буде у два рази меншою від максимально можливої. Тому максимальна кількість операцій удвічі менша від поданої вище, а саме:

$$3^{n-1} - 2^{n-1}. \tag{4.4}$$

Верхня оцінка складності запропонованого методу дорівнює $O(3^{n-1} - 2^{n-1})$.

Розглянемо приклад, що має найвищу комбінаторну складність для більшості методів мінімізації. Нехай задана булова функція тотожна одиниці. У цьому випадку верхня оцінка складності для методу мінімізації побітовим

розбиттям дорівнює $O(\log N)$. До порівняння, оцінка складності для методу Квайна-Мак-Класкі становитиме $O(9^n / n^2)$. І чим менша різниця $(2^n – N)$ у заданій буловій функції, тим більша ймовірність появи у процесі розбиття таких масивів кон'юнктермів, для яких комбінаційна складність методу мінімізації побітовим розбиттям зі склеюванням становитиме $O(\log N)$. Отже, окреслюється область застосування, у якій запропонований метод дає найбільшу перевагу над іншими.

Другий етап мінімізації: ланцюгове покриття множини простих кон'юнктермів

У [98] розглянуто випадок циклічної частини таблиці простих кон'юнктермів одного рангу, коли кожен стовпець може бути покритий на вибір одним з двох простих кон'юнктермів, а кожен простий кон'юнктерм покриває два стовпці. Тоді прості кон'юнктерми у буловому просторі утворюють своєрідний ланцюг, у якому для кожних трьох послідовних ланок середня покривається двома сусідніми. Запропоновано методику визначення послідовності простих кон'юнктермів у такому ланцюгу. Кон'юнктерми пронумеровують у порядку розташування. Подальше визначення мінімального покриття не потребує перебору. Для замкнутого ланцюга є два розв'язки. В одному всі прості кон'юнктерми з парними номерами, у другому – з непарними. Для розімкнутого ланцюга розв'язок теж тривіальний, але залежить від того чи парна кількість ланок у ньому, чи непарна. Покриття ланцюга з непарною кількістю ланок складають прості кон'юнктерми з парними номерами. Якщо кількість ланок m парне число, то існує $m/2$ розв'язків, що мають однаковий кошт реалізації. Для формування одного з розв'язків необхідно обрати простий кон'юнктерм із деяким парним номером k в ланцюгу, тоді доповнити розв'язок кон'юнктермами, що мають парні номери менші від k, потім дописати кон'юнктерми, що мають непарні номери більші від k.

Розглянемо складніший випадок, коли кон'юнктерми в ланцюгу є різних рангів. Тоді кожен розв'язок має своє значення коефіцієнта складності. У найгіршому випадку для визначення мінімального розв'язку необхідно

обчислити коефіцієнт складності для кожного з $m/2$ локальних розв'язків. Ще складнішим є випадок, коли з'являється точка розгалуження ланцюгів.

Кожен терм вносить у коефіцієнт складності число, що дорівнює сумі літералів плюс одиниця. Назвемо це число *вагою терму*. Якщо вага терму більша за суму ваг термів, що його покривають, то такий терм є неістотний. Можна вилучити цей терм з розгляду. Ланцюг розривається. Задача покриття спрощується. Тепер можна перевірити у незалежному потоці чи буде істотним кон'юнктерм, що має перетин із вилученим кон'юнктермом.

Сформулюємо метод покриття ланцюгами.

Потрібно починати формувати ланцюг з простого кон'юнктерма, що покриває тільки один мінтерм у циклічній частині. При досягнення точки розгалуження необхідно зупинити формування цього ланцюга і розпочати формування іншого ланцюга, що починається з простого кон'юнктерма, який покриває тільки один мінтерм циклічної частини. Якщо таких немає, то сформувати новий ланцюг з точки розгалуження. Після побудови усіх ланцюгів розглядаємо для точки розгалуження два варіанти: у першому вважаємо простий кон'юнктерм, який є точкою розгалуження, істотним, у другому — неістотним. У будь-якому з цих випадків відбувається розрив ланцюга. Задача розбивається на незалежні потоки. Тоді обираємо розв'язок із меншим коефіцієнтом складності. У загальному випадку такий підхід є евристичним, хоча за деяких обставин розрив ланцюга може призвести до спрощення задачі, що забезпечить точний розв'язок.

У запропонованому методі мінімізації побітовим розбиттям не виникає тавтологія, частина простих кон'юнктермів виявляється без проміжних склеювань. Для спрощення циклічної частини таблиці простих кон'юнктермів запропоновано ланцюгове покриття. Додатково для спрощення задачі можна обчислити коефіцієнт складності у точці розгалуження ланцюгів і на його основі віднести певний кон'юнктерм до множини покриття, що призведе до розриву ланцюга кон'юнктермів і зниженню обчислювальної складності процесу. Але у цьому випадку метод стає евристичним.

За допомогою методу мінімізації булових функцій побітовим розбиттям множини кон'юнктермів було розв'язано практичне завдання. Для цього у співавторстві було розроблено радіохвильовий сенсор [96], у якому забезпечено можливість цифрового керування за допомогою комбінаційної схеми на базі ПЛІС.

У Львівському центрі ІКД НАН та ДКА України за допомогою методу мінімізації булових функцій побітовим розбиттям множини кон'юнктермів було спроектовано цифрову комбінаційну схему на базі ПЛІС для керування мережею радіохвильових сенсорів. Впровадження цього методу на етапі синтезу цифрової комбінаційної схеми дало змогу:

- зменшити інформаційну ємність програмованої логікової інтегральної схеми (ПЛІС);
- розмістити додатковий вузол у тій же ПЛІС на вивільненій площі.

У відділі №111 ЛЦ ІКД НАН та ДКА України було виготовлено прототип спроектованої цифрової комбінаційної схеми і виконано його експериментальне дослідження, яке підтвердило стовідсоткову точність реалізації заданих булових функцій, що підтверджує відповідний акт (додаток А).

Розглянемо на прикладі мінімізацію однієї з реалізованих функцій.

Приклад 4.5

Мінімізувати булову функцію від семи змінних.

Y^1 = {(86), (87), (102), (103), (22), (38), (39), (0), (1), (5), (7), (8), (6), (10), (14), (48), (49), (53), (54), (22), (55), (58), (122), (62)}[1]

Розв'язання.

На першому етапі знайдемо множину простих кон'юнктермів.

Елементи множини упорядкуємо за наростанням. Маскам склеювань мінтермів і масці склеювань множини присвоюємо нуль.

$\{(122)^0, (103)^0, (102)^0, (87)^0, (86)^0, (62)^0, (58)^0, (55)^0, (54)^0, (53)^0, (49)^0, (48)^0, (39)^0, (38)^0, (23)^0, (22)^0, (14)^0, (10)^0, (8)^0, (7)^0, (6)^0, (5)^0, (1)^0, (0)^0\}^{1\,0}$

Вихідну множину розбиваємо за старшим бітом. У множину молодших чисел попадають елементи, що є менші від $2^6 = 64$, інші — у множину старших чисел.

$\{(62)^0, (58)^0, (55)^0, (54)^0, (53)^0, (49)^0, (48)^0, (39)^0, (38)^0, (23)^0, (22)^0, (14)^0, (10)^0, (8)^0, (7)^0, (6)^0, (5)^0, (1)^0, (0)^0\}^{1\,0}$

$\{(122)^0, (103)^0, (102)^0, (87)^0, (86)^0\}^{1\,0}$

У кожній підмножині менше ніж 2^6 кон’юнктермів, тому шукаємо склеювання за шостим бітом. Для цього від кожного елемента старшої множини віднімаємо $2^6 = 64$:

$\{(58)^0, (39)^0, (38)^0, (23)^0, (22)^0\}^{1\,0}$

Щоб отримати кон’юнктерми з поглинутим шостим бітом, шукаємо числа, що є в обох множинах:

$\{(122,58)^0, (103,39)^0, (102,38)^0, (87,23)^0, (86,22)^0\}^{1\,0}$

Обчислюємо маску склеювань одержаних кон’юнктермів як побітову кон’юнкцію масок склеювань твірних. Після цього у масках склеювань твірних встановлюємо одиницю в шостому біті.

$\{(62)^0, (58)^{64}, (55)^0, (54)^0, (53)^0, (49)^0, (48)^0, (39)^{64}, (38)^{64}, (23)^{64}, (22)^{64}, (14)^0, (10)^0, (8)^0, (7)^0, (6)^0, (5)^0, (1)^0, (0)^0\}^{1\,0}$

$\{(122)^{64}, (103)^{64}, (102)^{64}, (87)^{64}, (86)^{64}\}^{1\,64}$

Множину старших чисел вилучаємо з розгляду, бо її маска склеювань відмінна від нуля.

Одержані множини розглядаємо у незалежних процесах.

Одержану множину розбиваємо за п’ятим бітом.

$\{(87,23)^{0}, (86,22)^{0}\}^{1\,0}$

$\{(122,58)^{0}, (103,39)^{0}, (102,38)^{0}\}^{1\,0}$

Склеювання за п’ятим бітом неможливі. Кон’юнктерми множини молодших чисел склеюються по нульовому біту. Тому вилучаємо розглянуту множину. Натомість одержуємо $(87,22)^{0}$. Маска склеювань дорівнює нулю, тому це простий кон’юнктерм.

Множину старших чисел розбиваємо за четвертим бітом.

$\{(103,39)^{0}, (102,38)^{0}\}^{1\,0}$

$\{(122,58)^{0}\}^{1\,0}$

За четвертим бітом склеювання неможливі. Кон’юнктерми підмножини молодших чисел склеюються по нульовому біту. Множина вилучається з розгляду. Одержуємо $(103,38)^{0}$ — простий кон’юнктерм (нульова маска склеювань).

У підмножин старших чисел один простий кон’юнктерм (122,58) (нульова маска склеювань).

Розбиваємо за п’ятим бітом неопрацьовану множину, що одержана під час розбиття за шостим бітом.

$\{(23)^{64}, (22)^{64}, (14)^{0}, (10)^{0}, (8)^{0}, (7)^{0}, (6)^{0}, (5)^{0}, (1)^{0}, (0)^{0}\}^{1\,0}$

$\{(62)^{0}, (58)^{64}, (55)^{0}, (54)^{0}, (53)^{0}, (49)^{0}, (48)^{0}, (39)^{64}, (38)^{64}\}^{1\,0}$

Для пошуку склеювань за п’ятим бітом обнулимо його в елементах підмножини старших чисел.

$\{(30)^{0}, (26)^{64}, (23)^{0}, (22)^{0}, (21)^{0}, (17)^{0}, (16)^{0}, (7)^{64}, (6)^{64}\}^{1\,0}$

Шукаємо числа, що містяться в обох множинах:

$\{(30)^{0}, (26)^{58}, (55,23)^{0}, (54,22)^{0}, (21)^{0}, (17)^{0}, (16)^{0}, (39,7)^{0}, (38,6)^{0}\}^{1\,0}$

Обчислюємо маски склеювань

$\{(23)^{96}, (22)^{96}, (14)^{0}, (10)^{0}, (8)^{0}, (7)^{32}, (6)^{32}, (5)^{0}, (1)^{0}, (0)^{0}\}^{1\,0}$

$\{(62)^{0}, (58)^{64}, (55)^{32}, (54)^{32}, (53)^{0}, (49)^{0}, (48)^{0}, (39)^{96}, (38)^{96}\}^{1\,0}$

Здійснюємо розбиття за четвертим бітом

$\{(30)^{0}, (26)^{64}, (55,23)^{0}, (54,22)^{0}, (21)^{0}, (17)^{0}, (16)^{0}, (39,7)^{0}, (38,6)^{0}\}^{1\,0}$

Маємо

$\{(39,7)^{0}, (38,6)^{0}\}^{1\,0}$

$\{(30)^{0}, (26)^{64}, (55,23)^{0}, (54,22)^{0}, (21)^{0}, (17)^{0}, (16)^{0}\}^{1\,0}$

Для пошуку склеювань за четвертим бітом обнулимо його.

$\{(14)^0, (10)^{58}, (39,7)^0, (38,6)^0, (5)^0, (1)^0, (0)^0\}^{1\,0}$

Шукаємо числа, що містяться в обох множинах:

$\{(55,7)^0, (54,6)^0\}^{1\,0}$

— розглянуті кон'юнктерми склеяться за нульовим бітом у кон'юнктерм $(55,6)^0$ із нульовою маскою склеювань. Він є простий, а множину вилучають з розгляду.

$\{(39,7)^{16}, (38,6)^{16}\}^{1\,16}$

— вилучаємо одержану множину, бо маска склеювань ненульова.

$\{(30)^0, (26)^{64}, (55,23)^{16}, (54,22)^{16}, (21)^0, (17)^0, (16)^0\}^{1\,0}$

Після розбиття усіх одержаних множин одержуємо множину простих кон'юнктермів:

Y^1={(122,58), (62,58), (62,54), (55,53), (53,49), (49,48), (103,38), (87,22), (14,10), (10,8), (14,6), (55,6), (7,5), (5,1), (8,0), (1,0)}^{1}

На другому етапі шукаємо мінімальне покриття знайденої скороченої теоретико-множинної форми.

Формуємо таблицю простих кон'юнктермів, у якої рядки — це прості кон'юнктерми, а стовпці — це мінтерми.

Істотними є кон'юнктерми (122,58), (49,48), (103,38), (87,22). У них один з поглинутих мінтермів не покривається більше жодним простим кон'юнктермом. Після скорочення таблиці отримаємо циклічну частину (рисунок 4.3).

	62	55	54	53	14	10	8	7	6	5	1	0
(62,58)	1											
(62,54)	1		1									
(55,53)		1		1								
(53,49)				1								
(14,10)					1	1						
(10,8)						1	1					
(55,6)		1	1					1	1			
(14,6)					1				1			
(7,5)								1		1		
(5,1)										1	1	
(8,0)							1					1
(1,0)											1	1

Рисунок 4.3 – Циклічна частина таблиці простих кон’юнктермів

Простий кон’юнктерм (53,49) покриває лише один мінтерм циклічної частини, тому обираємо його першим. Формуємо від нього ланцюг до точки розгалуження. Одержуємо:

$$K_1=\langle(53,49)_1,(55,53)_2,(55,6)_3\rangle.$$

Аналогічно формуємо ланцюг від кон’юнктерма (62,58). Маємо:

$$K_2=\langle(62,58)_1,(62,54)_2,(55,6)_3\rangle.$$

Оскільки не залишилося простих кон’юнктермів, що покривають лише один мінтерм, формуємо ланцюг від точки розгалуження:

$$K_3=\langle(7,5)_1,(5,1)_2,(1,0)_3,(8,0)_4,(10,8)_5,(14,10)_6,(14,6)_7,(55,6)_8\ \rangle.$$

Простий кон’юнктерм (55,6) із вагою 5 є точкою розгалуження. Зробимо припущення, що він є істотний.

У цьому випадку для покриття першого ланцюга маємо два розв’язки з однаковим коефіцієнтом складності (вага кожного дорівнює 7):

- (53,49);
- (55,53).

Аналогічно для покриття другого ланцюга є два розв’язки з однаковим коефіцієнтом складності (вага кожного дорівнює 7):

- (62,58);
- (62,54).

У третьому ланцюзі є сім кон’юнктермів однакового рангу (непарна кількість). Тому його покриттям є множина простих кон’юнктермів, що мають парні номери в ланцюгу:

$$\{(5,1)_2, (8,0)_4, (14,10)_6\}^1.$$

Вага цього покриття становить

$$7 + 7 + 7 = 21.$$

За умови, що простий кон’юнктерм (6,55) є істотний, коефіцієнт складності покриття циклічної частини становить:

$$21 + 7 + 7 + 5 = 40.$$

Розглянемо інший варіант. Вважатимемо простий кон'юнктерм (55,6) неістотним і вилучимо з розгляду. Тоді є істотним кон'юнктерм (55,53) з першого ланцюга. Є неістотним другий кон'юнктерм (53,49). Вага дорівнює 7. Так само є істотним кон'юнктерм (62,54) з другого ланцюга. Є неістотним другий кон'юнктерм (62,58). Вага дорівнює 7. Також будуть істотними кон'юнктерми (7,5) та (14,6). Вага кожного з них дорівнює 7. Тепер третій ланцюг має вигляд:

$$K_3=\langle(5,1)_1,(1,0)_2,(8,0)_3,(10,8)_4,(14,10)_5\rangle.$$

У ланцюгу непарна кількість (п'ять) кон'юнктермів однакового рангу. Множина простих кон'юнктермів з парними номерами є покриттям:

$$\{(10,8)_4, (1,0)_2\}^1.$$

Вага покриття становить

$$7 + 7 = 14.$$

За умови, що простий кон'юнктерм (55,6) є неістотний, коефіцієнт складності покриття циклічної частини становить

$$14 + 7 + 7 + 7 + 7 = 42.$$

Тому простий кон'юнктерм (55,6) є істотним, а покриття циклічної частини запишемо так:

$$\left\{\begin{Bmatrix}(62,58)\\(62,54)\end{Bmatrix}, \begin{Bmatrix}(55,53)\\(53,49)\end{Bmatrix}, (14,10), (55,6), (5,1), (8,0)\right\}^1$$

Повний розв’язок покриття (з урахуванням істотних кон’юнктермів, що одержані на першому кроці спрощення таблиці простих кон’юнктермів) становить:

$$\{\begin{Bmatrix}(62,58)\\(62,54)\end{Bmatrix}, \begin{Bmatrix}(55,53)\\(53,49)\end{Bmatrix}, (14,10), (55,6), (5,1), (8,0), (49,48),$$
$$(122,58), (103,38), (87,22)\}^1.$$

Розроблений метод мінімізації булових функцій побітовим розбиттям множини кон’юнктермів та метод побітового сортування цілих чисел зі склеюванням було використано у задачі кластеризації даних при виконанні держбюджетної науково-дослідної роботи ДБ\Шум (держреєстр. № 0108U000326) у Національному університеті „Львівська політехніка”, що підтверджено відповідним актом (додаток А).

4.5 Синтез макромоделей перетворювача кодів

Розроблені методи мінімізації було використано у науково-практичній задачі при виконанні держбюджетної науково-дослідної роботи ДБ/ВЕРБАЛЬ (держреєстр. № 0104U002291) у Національному університеті „Львівська політехніка”, що підтверджує відповідний акт. Необхідно було синтезувати перетворювач кодів, що описується системою рівнянь:

$$\begin{cases} Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{0,1,5,6,7,8,10,14,112,114,116\}^1 \\ Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{48, 49, 53, 54, 55, 58, 62, 80, 122\}^1 \\ Y_3^1(x_3, x_2, x_1, x_0) = \{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}^1 \end{cases}$$

4.5.1 Синтез перетворювача кодів за допомогою удосконаленого методу розчеплення кон'юнктермів.

Мінімізуємо функцію Y_1^1

$$Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{0,1,5,6,7,8,10,14,112,114,116\}^1$$

$$Y_1^1 = [127] \overset{S}{\Rightarrow} \begin{bmatrix} [127-1] \\ [127-2] \\ [127-4] \\ [127-8] \\ [127-16] \\ [127-32] \\ [127-64] \end{bmatrix} =$$

$$= \begin{bmatrix} \{\underline{0} & \underline{0} & 4 & \underline{6} & \underline{6} & 8 & 10 & 14 & 112 & 114 & 116\}_{126} \\ \{0 & 1 & \underline{5} & 4 & \underline{5} & \underline{8} & \underline{8} & 12 & \underline{112} & \underline{112} & 116\}_{125} \\ \{0 & \underline{1} & \underline{1} & 2 & 3 & 8 & \underline{10} & \underline{10} & \underline{112} & 114 & \underline{112}\}_{123} \\ \{\underline{0} & 1 & 5 & \underline{6} & 7 & \underline{0} & 2 & \underline{6} & 112 & 114 & 116\}_{119} \\ \{0 & 1 & 5 & 6 & 7 & 8 & 10 & 14 & 96 & 98 & 100\}_{111} \\ \{0 & 1 & 5 & 6 & 7 & 8 & 10 & 14 & 80 & 82 & 84\}_{95} \\ \{0 & 1 & 5 & 6 & 7 & 8 & 10 & 14 & 48 & 50 & 52\}_{63} \end{bmatrix} \overset{C}{\Rightarrow}$$

$$\overset{C}{\Rightarrow} \{[125], [123], \begin{Bmatrix} [126] \\ [119] \end{Bmatrix}\} = \{\{5,8,112\}_{125}, \{1,10,112\}_{123}, \begin{Bmatrix} \{0,6\}_{126} \\ \{0,6\}_{119} \end{Bmatrix}\}^1$$

Маємо два розв'язки (Y_{1a}^1 та $Y_{1б}^1$):

$$Y_{1a}^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{\{5,8,112\}_{125}, \{1,10,112\}_{123}, \{0,6\}_{126}\}^1$$

$$Y_{1б}^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{\{5,8,112\}_{125}, \{1,10,112\}_{123}, \{0,6\}_{119}\}^1$$

На рисунку 4.4 подано таблицю простих кон'юнктермів розв'язку Y_{1a}^{1}.

	0	1	5	7	6	14	10	8	112	114	116
$(0,1)_{126}$	1	1									
$(1,5)_{123}$		1	1								
$(8,10)_{125}$							1	1			
$(5,7)_{125}$			1	1							
$(10,14)_{123}$						1	1				
$(6,7)_{126}$				1	1						
$(112,114)_{125}$									1	1	
$(112,116)_{123}$									1		1

Рисунок 4.4 – Таблиця простих кон'юнктермів розв'язку Y_{1a}^{1}

$$Y_{1a}^{1}(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, 0_{126}, 8_{125}, 10_{123}, 6_{126}, \begin{Bmatrix} 1_{123} \\ 5_{125} \end{Bmatrix}\}^1$$

На рисунку 4.5 подано таблицю простих кон'юнктермів розв'язку $Y_{1б}^{1}$.

	0	1	5	7	6	14	10	8	112	114	116
$(0,8)_{119}$	1							1			
$(1,5)_{123}$		1	1								
$(8,10)_{125}$							1	1			
$(5,7)_{125}$			1	1							
$(10,14)_{123}$						1	1				
$(6,14)_{119}$					1	1					
$(112,114)_{125}$									1	1	
$(112,116)_{123}$									1		1

Рисунок 4.5 – Таблиця простих кон'юнктермів розв'язку $Y_{1б}^{1}$

$$Y_{1б}^{1}(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, 0_{119}, 1_{123}, 5_{125}, 6_{119}, \begin{Bmatrix} 8_{125} \\ 10_{123} \end{Bmatrix}\}^1$$

Остаточно повна відповідь:

$$Y_1^1(x_6,x_5,x_4,x_3,x_2,x_1,x_0)=\{112_{125},112_{123},\begin{Bmatrix}0_{119},1_{123},5_{125},6_{119},\begin{Bmatrix}8_{125}\\10_{123}\end{Bmatrix}\\0_{126},8_{125},10_{123},6_{126},\begin{Bmatrix}1_{123}\\5_{125}\end{Bmatrix}\end{Bmatrix}\}^1.$$

Мінімізуємо функцію Y_2^1

$$Y_2^1(x_6,x_5,x_4,x_3,x_2,x_1,x_0)=\{48,49,53,54,55,58,62,80,122\}^1$$

$$Y_2^1=[127]\overset{S}{\Rightarrow}\begin{bmatrix}[127-1]\\[127-2]\\[127-4]\\[127-8]\\[127-16]\\[127-32]\\[127-64]\end{bmatrix}=$$

$$=\begin{bmatrix}\{\underline{48}&\underline{48}&52&\underline{54}&\underline{54}&58&52&80&122\}_{126}\\\{48&49&\underline{53}&52&\underline{53}&56&60&80&120\}_{125}\\\{48&\underline{49}&\underline{49}&50&51&\underline{58}&\underline{58}&80&122\}_{123}\\\{48&49&53&\underline{54}&55&50&\underline{54}&80&114\}_{119}\\\{32&33&37&38&39&42&46&64&106\}_{111}\\\{16&17&21&22&23&26&30&80&90\}_{95}\\\{48&49&53&54&55&\underline{58}&62&16&\underline{58}\}_{63}\end{bmatrix}\overset{C}{\Rightarrow}$$

$$\overset{C}{\Rightarrow}\{80_{127},[63],[126],[123]\}=\{80_{127},58_{63},\{48{,}54\}_{126},\{49{,}58\}_{123}\}^1$$

На рисунку 4.6 подано таблицю простих кон'юнктермів функції Y_2.

	48	49	53	55	54	62
$(48,49)_{126}$	1	1				
$(49,53)_{123}$		1	1			
$(58,62)_{123}$						1
$(54,55)_{126}$				1	1	

Рисунок 4.6 – Таблиця простих кон'юнктермів функції Y_2

$$Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{80_{127}, 58_{63}, 48_{126}, 49_{123}, 58_{123}, 54_{126}\}^1$$

Мінімізуємо функцію Y_3^1

$$Y_3^1(x_3, x_2, x_1, x_0) = \{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}^1$$

$$Y_3^1 = [15] \overset{S}{\Rightarrow} \begin{bmatrix} 15-1 \\ 15-2 \\ 15-4 \\ 15-8 \end{bmatrix} =$$

$$= \begin{bmatrix} \underline{0} & \underline{0} & \underline{2} & \underline{2} & \underline{4} & \underline{4} & \underline{6} & \underline{6} & \underline{8} & \underline{8} & \underline{10} & \underline{10} & \underline{12} & \underline{12} & 14 \\ \underline{0} & \underline{1} & \underline{0} & \underline{1} & \underline{4} & \underline{5} & \underline{4} & \underline{5} & \underline{8} & \underline{9} & \underline{8} & \underline{9} & \underline{12} & 13 & \underline{12} \\ \underline{0} & \underline{1} & \underline{2} & \underline{3} & \underline{0} & \underline{1} & \underline{2} & \underline{3} & \underline{8} & \underline{9} & \underline{10} & 11 & \underline{8} & \underline{9} & \underline{10} \\ \underline{0} & \underline{1} & \underline{2} & \underline{3} & \underline{4} & \underline{5} & \underline{6} & 7 & \underline{0} & \underline{1} & \underline{2} & \underline{3} & \underline{4} & \underline{5} & \underline{6} \end{bmatrix} \overset{C}{\Rightarrow}$$

$$\overset{C}{\Rightarrow} \{[14], [13]\}^1 = \{\{0,2,4,6,8,10,12\}_{14}, \{0,1,4,5,8,9,12\}_{13}\}^1 =$$

$$\overset{S}{\Rightarrow} \{\begin{bmatrix} 14-2 \\ 14-4 \\ 14-8 \end{bmatrix}, \begin{bmatrix} 13-1 \\ 13-4 \\ 13-8 \end{bmatrix}\}^1 =$$

$$= \{\begin{bmatrix} \{\underline{0} & \underline{0} & \underline{4} & \underline{4} & \underline{8} & \underline{8} & 12\}_{12} \\ \{\underline{0} & \underline{2} & \underline{0} & \underline{2} & \underline{8} & 10 & \underline{8}\}_{10} \\ \{\underline{0} & \underline{2} & \underline{4} & 6 & \underline{0} & \underline{2} & \underline{4}\}_6 \end{bmatrix}, \begin{bmatrix} \{\underline{0} & \underline{0} & \underline{4} & \underline{4} & \underline{8} & \underline{8} & 12\}_{12} \\ \{\underline{0} & \underline{1} & \underline{0} & \underline{1} & \underline{8} & 9 & \underline{8}\}_9 \\ \{\underline{0} & \underline{1} & \underline{4} & 5 & \underline{0} & \underline{1} & \underline{4}\}_5 \end{bmatrix}\}^1 \overset{C}{\Rightarrow}$$

$$\overset{C}{\Rightarrow} \{[12], [10], [9]\}^1 = \{\{0,4,8\}_{12}, \{0,2,8\}_{10}, \{0,1,8\}_9\}^1 =$$

$$\overset{S}{\Rightarrow} \{\begin{bmatrix} 12-4 \\ 12-8 \end{bmatrix}, \begin{bmatrix} 10-2 \\ 10-8 \end{bmatrix}, \begin{bmatrix} 9-1 \\ 9-8 \end{bmatrix}\} = \{\begin{bmatrix} \{\underline{0} & \underline{0} & 8\}_8 \\ \{\underline{0} & 4 & \underline{0}\}_4 \end{bmatrix}, \begin{bmatrix} \{\underline{0} & \underline{0} & 8\}_8 \\ \{\underline{0} & 2 & \underline{0}\}_2 \end{bmatrix}, \begin{bmatrix} \{\underline{0} & \underline{0} & 8\}_8 \\ \{\underline{0} & 1 & \underline{0}\}_1 \end{bmatrix}\} =$$

$$\overset{C}{\Rightarrow} \{[8], [4], [2], [1]\}^1 = \{0_8, 0_4, 0_2, 0_1\}^1$$

На рисунку 4.7 подано таблицю простих кон'юнктермів функції Y_3.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$(0)_8$	1	1	1	1	1	1	1	1							
$(0)_4$	1	1	1	1					1	1	1	1			
$(0)_2$	1	1			1	1			1	1			1	1	
$(0)_1$	1		1		1		1		1		1		1		1

Рисунок 4.7 – Таблиця простих кон'юнктермів функції Y_3

$$Y_3^1(x_3, x_2, x_1, x_0) = \{0_8, 0_4, 0_2, 0_1\}^1$$

Остаточно одержимо систему функцій

$$\begin{cases} Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, \begin{cases} 0_{119}, 1_{123}, 5_{125}, 6_{119}, \begin{Bmatrix} 8_{125} \\ 10_{123} \end{Bmatrix} \\ 0_{126}, 8_{125}, 10_{123}, 6_{126}, \begin{Bmatrix} 1_{123} \\ 5_{125} \end{Bmatrix} \end{cases} \}^1 \\ Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{80_{127}, 58_{63}, 48_{126}, 49_{123}, 58_{123}, 54_{126}\}^1 \\ Y_3^1(x_3, x_2, x_1, x_0) = \{0_8, 0_4, 0_2, 0_1\}^1 \end{cases}$$

Для одного з розв'язків функція Y_1 в аналітичній формі має вигляд:

$$Y_1 = x_6\, x_5\, x_4\, \overline{x_3}\, \overline{x_2}\, \overline{x_0} \vee x_6\, x_5\, x_4\, \overline{x_3}\, \overline{x_2}\, \overline{x_0} \vee \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, \overline{x_2}\, \overline{x_1}\, \overline{x_0} \vee$$

$$\vee\ \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, \overline{x_3}\, \overline{x_1}\, x_0 \vee\ \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, \overline{x_3}\, x_2\, x_0 \vee\ \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, x_2\, x_1\, \overline{x_0} \vee\ \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, x_3\, \overline{x_2}\, \overline{x_0}.$$

Функція Y_2 та Y_3 в аналітичній формі набувають вигляду:

$$Y_2 = x_6\, \overline{x_5}\, x_4\, \overline{x_3}\, \overline{x_2}\, \overline{x_1}\, \overline{x_0} \vee\ x_5\, x_4\, x_3\, \overline{x_2}\, x_1\, \overline{x_0} \vee\ \overline{x_6}\, x_5\, x_4\, \overline{x_3}\, \overline{x_2}\, \overline{x_1} \vee$$
$$\vee\ \overline{x_6}\, x_5\, x_4\, \overline{x_3}\, \overline{x_1}\, x_0 \vee\ \overline{x_6}\, x_5\, x_4\, x_3\, x_1\, \overline{x_0} \vee\ \overline{x_6}\, x_5\, x_4\, \overline{x_3}\, x_2\, x_1.$$

$$Y_3 = x_3 \vee x_2 \vee x_1 \vee x_0$$

Схему синтезованого перетворювача кодів подано у додатку В (рисунок В.1).

4.5.2 Синтез перетворювача кодів за допомогою методу побітового розбиття зі склеюванням

Розглянемо функцію від чотирьох змінних:

$$Y_3^1(x_3, x_2, x_1, x_0)=\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}^1$$

Маска літералів для мінтермів функції від чотирьох змінних дорівнює 15.

Розбиваємо множину мінтермів за старшим бітом, тобто на ті, що менші від 2^3, і ті, що не менші. Отримаємо

$$\{0,1,2,3,4,5,6,7\}^1,\{8,9,10,11,12,13,14\}^1.$$

У підмножині менших елементів є 2^3 елементів, тому вони склеюються у кон'юнктерм (0 – – –). Тоді у старшій підмножині в усіх елементах потрібно замінити одиницю в старшому біті на символ поглинання, для цього у числовому зображення і у масці літералів цей біт обнулюють (віднімаємо 8 від кожного числа).

$$\{0,1,2,3,4,5,6\}_7^1$$

Розбиваємо за наступним (другим) бітом множину, що залишилась.

$$\{0,1,2,3\}_7^1,\ \{4,5,6\}_7^1$$

У молодшій підмножині є 2^2 елементів. Значить вони склеюються у кон'юнктерм $\{0,1,2,3\}_4 = (-\ 0\ -\ -)$. Тоді у старшій підмножині в усіх елементах потрібно замінити одиницю у другому біті на символ поглинання, для цього у числовому зображення і у масці літералів цей біт обнулюють (віднімаємо 4 від кожного числа).

$$\{0,1,2\}_3^1$$

Розбиваємо за наступним (першим) бітом множину, що залишилась.

$$\{0,1\}_3^1,\{2\}_3^1$$

У молодшій підмножині є 2^1 елементів. Значить вони склеюються у кон'юнктерм $\{0,1\}_2 = (-\ -\ 0\ -)$. Тоді у старшій підмножині в усіх елементах потрібно замінити одиницю у першому біті на символ поглинання, для цього у числовому зображення і у масці літералів цей біт обнулюють (віднімаємо 2 від кожного числа).

$$\{0\}_1^1 = (-\ -\ -\ 0)$$

Для розглянутої функції не було потреби обчислювати маски склеювань, оскільки кожен кон'юнктерм склеювався лише раз і одразу вилучався з розгляду.

Отримали множину простих кон'юнктермів функції Y_3:

$$Y_3 = \{(0\ -\ -\ -),(-\ 0\ -\ -),(-\ -\ 0\ -),(-\ -\ -\ 0)\}^1.$$

Аналогічно одержано множини усіх простих кон'юнктермів для функцій Y_1 та Y_2:

$Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0)$={$112_{125}$, 112_{123}, 0_{126}, 1_{123}, 5_{125}, 6_{119}, 6_{126}, 0_{119}, 8_{125}, 10_{123}}^{1}

$Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0)$={$80_{127}$, 58_{63}, 48_{126}, 49_{123}, 58_{123}, 53_{125}, 54_{119}, 54_{126}}^{1}

Виконаємо покриття таблиці простих кон'юнктермів функції Y_1^1 (рисунок 4.8).

	0	1	5	7	6	14	10	8	112	114	116
$(0,1)_{126}$	1	1									
$(0,8)_{119}$	1							1			
$(1,5)_{123}$		1	1								
$(8,10)_{125}$							1	1			
$(5,7)_{125}$			1	1							
$(10,14)_{123}$						1	1				
$(6,14)_{119}$					1	1					
$(6,7)_{126}$				1	1						
$(112,114)_{125}$									1	1	
$(112,116)_{123}$									1		1

Рисунок 4.8 – Таблиця простих кон'юнктермів $Y_1{}^1$

Після вилучення істотних кон'юнктермів 112_{125} та 112_{123} розглянемо задачу покриття для множини, що залишилась — K_1, усі прості кон'юнктерми якої є рангу 2, а вага кожного стовпця таблиці кон'юнктермів дорівнює 2. Упорядкуємо K_1 так, щоб кон'юнктерми утворили ланцюг. У таблиці кон'юнктермів не залишилось рядків одиничної ваги, тому вибираємо перший рядок і присвоюємо відповідному кон'юнктерму номер 1 (номер позначатимемо надрядковим індексом): $\overset{1}{0}_{126}$

$$K_1 = \left\langle \overset{1}{0}_{126}, \overset{2}{1}_{123}, \overset{3}{5}_{125}, \overset{4}{6}_{126}, \overset{5}{6}_{119}, \overset{6}{10}_{123}, \overset{7}{8}_{125}, \overset{8}{0}_{119} \right\rangle.$$

Оскільки під час упорядкування множини K_1 першим вибрано рядок із вагою 2, то існує два розв'язки; у перший входять усі кон'юнктерми з непарними номерами

$$K_{1,1} = \left\{ \overset{1}{0}_{126}, \overset{3}{5}_{125}, \overset{5}{6}_{119}, \overset{7}{8}_{125} \right\},$$

$$Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, 0_{126}, 5_{125}, 6_{119}, 8_{125}\}^1.$$

У другий розв'язок входять усі кон'юнктерми з парними номерами:

$$K_{1,2} = \left\{ \overset{2}{1}_{123}, \overset{4}{6}_{126}, \overset{6}{10}_{123}, \overset{8}{0}_{119} \right\}.$$

$$Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, 1_{123}, 6_{126}, 0_{119}, 10_{123}\}^1.$$

Повний розв'язок

$$Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, \begin{Bmatrix} 0_{126}, 5_{125}, 6_{119}, 8_{125} \\ 1_{123}, 6_{126}, 0_{119}, 10_{123} \end{Bmatrix}\}^1.$$

Виконаємо покриття функції Y_2^1 (рисунок 4.9).

	48	49	53	54	55	58	62	80	122
80_{127}								1	
$(58,122)_{63}$						1			1
$(48,49)_{126}$	1	1							
$(49,53)_{123}$		1	1						
$(58,62)_{123}$						1	1		
$(54,55)_{126}$				1	1				
$(54,62)_{119}$				1			1		
$(53,55)_{125}$			1		1				

Рисунок 4.9 – Таблиця простих кон'юнктермів Y_2

Після вилучення істотних кон'юнктермів 80_{127}, 58_{63} та 48_{126} розглянемо задачу покриття для множини що залишилась — K_2, всі прості кон'юнктерми якої є рангу 2, а вага кожного стовпця таблиці кон'юнктермів дорівнює 2. Упорядкуємо K_2 так, щоб кон'юнктерми утворили ланцюг. Знаходимо рядок із вагою 1 і присвоюємо відповідному кон'юнктерму номер 1: $\overset{1}{49}_{123}$.

Одержимо

$$K_2 = \left\langle \overset{1}{49}_{123}, \overset{2}{53}_{125}, \overset{3}{54}_{126}, \overset{4}{54}_{119}, \overset{5}{58}_{123} \right\rangle.$$

Оскільки під час упорядкування множини K_2 першим вибрано рядок із вагою 1 і кількість кон'юнктермів є непарною, то розв'язком є множина кон'юнктермів із парними номерами:

$$K_2 = \left\{ \overset{2}{53}_{125}, \overset{4}{54}_{119} \right\}.$$

$$Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{80_{127}, 58_{63}, 48_{126}, 53_{125}, 54_{119}\}^1$$

Остаточно одержимо систему функцій, що описують заданий перетворювач кодів:

$$\begin{cases} Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{112_{125}, 112_{123}, \begin{Bmatrix} 0_{126}, 5_{125}, 6_{119}, 8_{125} \\ 1_{123}, 6_{126}, 0_{119}, 10_{123} \end{Bmatrix}\}^1 \\ Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0) = \{80_{127}, 58_{63}, 48_{126}, 53_{125}, 54_{119}\}^1 \\ Y_3^1(x_3, x_2, x_1, x_0) = \{0_8, 0_4, 0_2, 0_1\}^1 \end{cases}$$

Для одного з розв'язків функція Y_1 у аналітичній формі має вигляд

$$Y_1 = x_6\, x_5\, x_4\, \overline{x_3}\, \overline{x_2}\, \overline{x_0} \vee x_6\, x_5\, x_4\, \overline{x_3}\, \overline{x_2}\, \overline{x_0} \vee \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, \overline{x_3}\, \overline{x_2}\, \overline{x_1} \vee$$
$$\vee\, \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, \overline{x_3}\, x_2\, x_0 \vee \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, x_2\, x_1\, \overline{x_0} \vee \overline{x_6}\, \overline{x_5}\, \overline{x_4}\, x_3\, \overline{x_2}\, \overline{x_0}.$$

Функція Y_2 у аналітичній формі має вигляд

$$Y_2 = x_6\,\overline{x_5}\,x_4\,\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} \vee x_5\,x_4\,x_3\,\overline{x_2}\,x_1\,\overline{x_0} \vee \overline{x_6}\,x_5\,x_4\,\overline{x_3}\,\overline{x_2}\,\overline{x_1} \vee$$
$$\vee\,\overline{x_6}\,x_5\,x_4\,\overline{x_3}\,x_2\,x_0 \vee \overline{x_6}\,x_5\,x_4\,x_2\,x_1\,\overline{x_0}.$$

Функція Y_3 у аналітичній формі має вигляд

$$Y_3 = x_3 \vee x_2 \vee x_1 \vee x_0$$

Схема синтезованого перетворювача кодів наведена у додатку В (рисунок В.2).

4.5.3 Синтез перетворювача кодів за допомогою удосконаленого методу порозрядного вирощування

Мінімізуємо функцію від чотирьох змінних:

$$Y_3^1(x_3, x_2, x_1, x_0)=\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}^1$$

1) пошук склеювань та сортування кон’юнктермів по нульовому біту

$\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}_{15}$		
$\{0,2,4,6,8,10,12,14\}_{15}=0_1$	~~$\{1,3,5,7,9,11,13\}_{15}$~~	$\{0,2,4,6,8,10,12\}_{14}$

Рисунок 4.10

Кількість парних мінтермів складає 2^{4-1}, значить вони склеюються по усіх бітах окрім нульового.

Усі непарні мінтерми брали участь у склеюванні по нульовому біту, тому виключаємо їх із подальшого розгляду.

2) пошук склеювань та сортування кон'юнктермів по першому біту

$\{0,2,4,6,8,10,12\}_{14}$		
$\{0,2,4,6\}_{14}=0_2$	~~$\{1,3,5\}_{14}$~~	$\{0,4,8\}_{12}$

Рисунок 4.11

Кількість кон'юнктермів із маскою "14", що містять "0" у першому біті, складає 2^{4-2}, значить вони склеюються по усіх бітах окрім першого.

Усі кон'юнктерми із маскою "14", що містять "1" у першому біті, брали участь у склеюванні по першому біту, тому виключаємо їх із подальшого розгляду.

3) пошук склеювань та сортування кон'юнктермів по другому біту

$\{0,4,8\}_{12}$		
$\{0, 8\}_{12}=0_4$	~~$\{4\}_{12}$~~	0_8

Рисунок 4.13

$$Y_3^1(x_3, x_2, x_1, x_0) = \{0_8, 0_4, 0_2, 0_1\}^1.$$

Аналогічно одержано такі ж множини усіх простих кон'юнктермів для функцій Y_1 та Y_2, як і методом побітового розбиття:

$Y_1^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0)$={$112_{125}$, 112_{123}, 0_{126}, 1_{123}, 5_{125}, 6_{119}, 6_{126}, 0_{119}, 8_{125}, 10_{123}}^{1}

$Y_2^1(x_6, x_5, x_4, x_3, x_2, x_1, x_0)$={$80_{127}$, 58_{63}, 48_{126}, 49_{123}, 58_{123}, 53_{125}, 54_{119}, 54_{126}}^{1}

Тому подальше покриття одержаних множин простих кон’юнктермів, дасть той самий результат, що і для методу побітового розбиття.

На відміну від удосконаленого метода розчеплення кон’юнктермів удосконалений метод порозрядного вирощування не обтяжений пошуком копій для склеювання кон’юнктермів, оскільки у цьому методі можуть склеїтись тільки ті кон’юнктерми, що є сусідніми у лінійному списку. Крім того запропонована модифікація дає змогу одразу визначити чи склеюється одержана сортуванням підмножина в один кон’юнктерм і, таким чином, пропустити опрацювання решти бітів, якщо такий кон’юнктерм знайдено.

Застосування удосконаленого методу порозрядного вирощування до розглянутої задачі дало той самий результат, що й метод побітового розбиття, але потребувало більшого обсягу обчислювальних ресурсів, оскільки на кожному етапі відбувалося пересортування елементів множин.

Одержані результати наочно демонструють переваги застосування методу побітового розбиття для пошуку множини усіх простих кон’юнктермів із подальшим застосуванням покриття ланцюгами порівняно із удосконаленим методом розчеплення кон’юнктермів та удосконаленим методом порозрядного вирощування. Апаратна реалізація розглянутого перетворювача кодів синтезованого за допомогою методу побітового розбиття матиме на два шестивходових кон’юнктора менше ніж той самий перетворювач кодів, синтезований за допомогою методу розчеплення кон’юнктермів.

4.6 Висновок до розділу 4

1) На основі алгоритму удосконаленої процедури формування матриці розчеплення у масковому зображенні та алгоритму удосконаленої процедури покриття матриці розчеплення удосконалено метод розчеплення кон’юнктермів. З використанням удосконаленого методу розчеплення кон’юнктермів розв’язано

практичне завдання проектування універсального сигнального перетворювача промислової автоматики на основі мережі первинних перетворювачів із оцифрованими вихідними сигналами. Ефект, отриманий від впровадження запропонованого удосконаленого методу на етапі синтезу цифрової комбінаційної схеми, полягає у зменшенні на 5% кількості комірок ПЛІС, які необхідні для реалізації заданих булових функцій.

2) На основі алгоритму удосконаленої процедури спрощення множини символьних диз'юнктермів набула подальшого розвитку теоретико-множинна модифікація мінімаксного методу покриття булових функцій.

3) На основі алгоритму, який в процесі сортування кон'юнктермів дає змогу виявити такі підмножин, що склеюються, та на основі модифікації коду помітки удосконалено метод порозрядного вирощування простих кон'юнктермів булових функцій.

4) На основі алгоритму пошуку простих кон'юнктермів побітовим розбиттям множини кон'юнктермів зі склеюванням та розробленої процедури ланцюгового покриття запропоновано метод мінімізації булових функцій побітовим розбиттям.

За допомогою методу мінімізації булових функцій побітовим розбиттям множини кон'юнктермів було розв'язано практичне завдання, а саме спроектовано цифрову комбінаційну схему на базі ПЛІС для керування мережею радіохвильових сенсорів. Впровадження цього методу на етапі синтезу цифрової комбінаційної схеми дало змогу:

- зменшити інформаційну ємність програмованої логікової інтегральної схеми (ПЛІС);
- розмістити додатковий вузол у тій же ПЛІС на вивільненій площі.

Було виготовлено прототип спроектованої цифрової комбінаційної схеми і виконано його експериментальне дослідження, яке підтвердило стовідсоткову точність реалізації заданих булових функцій.

5) Виконано синтез перетворювачів кодів за допомогою удосконалених та розробленого методів, серед яких кращий результат показав метод мінімізації побітовим розбиттям.

ВИСНОВКИ

У дисертації розв’язано актуальне науково-практичне завдання, яке полягає в удосконаленні та розробленні алгоритмів та методів мінімізації булових функцій для проектування цифрових комбінаційних схем радіотехнічних пристроїв та засобів телекомунікацій.

1 На основі аналізу літературних джерел для подальших досліджень обрано методи мінімізації булових функцій перспективні щодо оптимізації комп’ютерної реалізації. Виявлено їх недоліки та окреслено напрями пошуку удосконалень.

2 Запропоновано числове подання кон'юнктермів довільного рангу у вигляді маскового зображення, що дає змогу використовувати в алгоритмах мінімізації булових функцій двійкові операції замість операцій над символами. Такий підхід спрощує комп'ютерну реалізацію методів мінімізації.

3 Удосконалено метод побітового сортування множини цілих чисел, що позбавлена тавтології, шляхом підрахунку елементів підмножини в процесі сортування по заданому біту з номером n, що дає змогу виявити склеювання одержаної підмножини у кон'юнктерм і замінити процедуру сортування по решті бітів простим перерахунком від 0 до 2^n–1. Крім того, оскільки певні методи мінімізації потребують попереднього сортування множини вихідних кон'юнктермів, запропонована модифікація дає змогу одержати деякі імпліканти вже на етапі сортування.

4 Уперше запропоновано метод мінімізації булових функцій побітовим розбиттям множини кон’юнктермів на основі розробленої модифікації побітового сортування зі склеюванням, що на відміну від існуючих дає змогу виявляти прості кон’юнктерми низького рангу без проміжних склеювань простим підрахунком кількості кон’юнктермів. Цей метод позбавлений тавтології.

5 Запропоновано процедуру ланцюгового покриття таблиці простих кон’юнктермів, яка враховує взаємне розташування кон’юнктермів у двійковому просторі, що дає змогу спростити циклічну частину таблиці простих кон’юнктермів, що у свою чергу призводить до зменшення витрат машинних ресурсів. Для зниження обчислювальної складності задачі можна розрахувати коефіцієнт складності в околі розгалуження ланцюгів і прийняти рішення про розрив ланцюга в цьому місці. Процедура ланцюгового покриття множини простих кон’юнктермів дає змогу розбивати задачу покриття на декілька обчислювальних потоків. Цей підхід є евристичний.

6 Удосконалено метод розчеплення кон’юнктермів. Запропонована модифікація процедури розчеплення кон’юнктермів на основі маскового зображення дає змогу зменшити витрати обчислювальних ресурсів, а використання додатково шістнадцяткової системи числення дає змогу розширити коло задач, які можна розв’язати без застосування комп’ютера.

7 Набув подальшого розвитку метод мінімаксного покриття у теоретико-множинній формі, а саме запропоновано упорядковувати символьні диз’юнктерми за наростанням їх потужностей, тобто починаючи з мінімальної. Це пришвидшує процедуру спрощення сформованої множини за рахунок скорочення шляху пошуку спрощуваних диз’юнктермів.

8 Удосконалено метод порозрядного вирощування простих кон’юнктермів. Для цього використано розроблене маскове зображення замість псевдотрійкового, що дало змогу оперувати числовим поданням кон'юнктермів у підмножинах з однаковою маскою, уведено поняття коду помітки множини кон’юнктермів для усічення трійкового дерева вирощування простих кон’юнктермів, розроблено процедуру виявлення підмножин, елементи яких склеюються в один кон’юнктерм простим підрахунком кількості елементів одразу на етапі сортування по заданому біту, розширено область застосування методу на недовизначені функції. Запропоновані зміни дають змогу розширити коло розв’язуваних задач за рахунок зменшення обчислювальних витрат та розширення області застосування методу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. *B. Madoš, Z. Bilanová, E. Chovancová and N. Ádám*, "Field Programmable Gate Array Hardware Accelerator of Prime Implicants Generation for Single-Output Boolean Functions Minimization," 2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA), Starý Smokovec, Slovakia, 2019, pp. 493-498, doi: 10.1109/ICETA48886.2019.9040020.
2. *H. -G. Vu, D. -D. Tran, N. -D. Bui, T. -B. Le and H. -D. Nguyen*, "Mapping Boolean Functions onto Lookup-Tables on FPGAs," 2022 RIVF International Conference on Computing and Communication Technologies (RIVF), Ho Chi Minh City, Vietnam, 2022, pp. 508-512, doi: 10.1109/RIVF55975.2022.10013797.
3. *D. -D. Tran and H. -G. Vu*, "Boolean-Function-based IP Lookup on FPGAs," 2023 International Conference on Advanced Technologies for Communications (ATC), Da Nang, Vietnam, 2023, pp. 393-398, doi: 10.1109/ATC58710.2023.10318917.
4. *R. K. Yousif, I. A. Hashim and B. H. Abd*, "FPGA Implementation of Polynomial Curve Fitting Approximation for Sine and Cosine Generator," 2022 5th International Conference on Engineering Technology and its Applications (IICETA), Al-Najaf, Iraq, 2022, pp. 361-366, doi: 10.1109/IICETA54559.2022.9888742.
5. *J. Echavarria, S. Wildermann and J. Teich*, "AConFPGA: A Multiple-Output Boolean Function Approximation DSE Technique Targeting FPGAs," 2018 International Conference on Field-Programmable Technology (FPT), Naha, Japan, 2018, pp. 326-329, doi: 10.1109/FPT.2018.00065.
6. *M. X. Lyons and K. Gaj*, "Sampling from Discrete Distributions in Combinational Hardware with Application to Post-Quantum Cryptography," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 610-613, doi: 10.23919/DATE48585.2020.9116434.

7. *C. Yu*, "FlowTune: Practical Multi-armed Bandits in Boolean Optimization," 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2020, pp. 1-9.
8. *A. Klimowicz*, "Balanced Power, Speed and Area Transformation Procedure for Finite State Machines," 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 2022, pp. 2313-2318, doi: 10.1109/SMC53654.2022.9945212.
9. *A. N. Borodzhieva, I. I. Stoev, I. D. Tsvetkova, S. L. Zaharieva and V. A. Mutkov*, "FPGA Design of Boolean Functions Using a Cascade of Decoders and Logic Gates," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2020, pp. 1596-1600, doi: 10.23919/MIPRO48935.2020.9245448.
10. *F. Liu, M. Yan, Y. Mao and J. Wang*, "Application of Semi-tensor Product-based Bi-decomposition to FPGA Mapping," 2022 41st Chinese Control Conference (CCC), Hefei, China, 2022, pp. 5945-5949, doi: 10.23919/CCC55666.2022.9901693.
11. *A. Borodzhieva, I. Stoev and V. Mutkov*, "FPGA Implementation of Boolean Functions Using Decoders and Logic Gates," 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), Cluj-Napoca, Romania, 2019, pp. 164-167, doi: 10.1109/SIITME47687.2019.8990690.
12. *X. Shi, M. Yang, Z. Li, K. Zhu and L. Wang*, "Exploration of FPGA PLB Architecture Base on LUT and Microgates," 2023 International Symposium of Electronics Design Automation (ISEDA), Nanjing, China, 2023, pp. 184-189, doi: 10.1109/ISEDA59274.2023.10218468.
13. *A. N. Borodzhieva, I. I. Stoev and V. A. Mutkov*, "FPGA Implementation of Boolean Functions Using Multiplexers," 2019 IEEE XXVIII International Scientific Conference Electronics (ET), Sozopol, Bulgaria, 2019, pp. 1-4, doi: 10.1109/ET.2019.8878504.
14. *A. N. Borodzhieva*, "An Approach for Designing Boolean Functions with Decoders," 2020 International Symposium on Fundamentals of Electrical

Engineering (ISFEE), Bucharest, Romania, 2020, pp. 1-5, doi: 10.1109/ISFEE51261.2020.9756186.

15. *D. Havrilov, A. Volovyk, L. Koval, M. Vasylkivskyi, A. Semenov and N. Havrilova*, "Design of Digital Data Selectors on FPGA in a Laboratory Environment," 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), Kharkiv, Ukraine, 2021, pp. 495-500, doi: 10.1109/PICST54195.2021.9772137.
16. *M. X. Lyons and K. Gaj*, "Sampling from Discrete Distributions in Combinational Hardware with Application to Post-Quantum Cryptography," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 610-613, doi: 10.23919/DATE48585.2020.9116434.
17. *A. Boutros and V. Betz*, "FPGA Architecture: Principles and Progression," in IEEE Circuits and Systems Magazine, vol. 21, no. 2, pp. 4-29, Secondquarter 2021, doi: 10.1109/MCAS.2021.3071607.
18. S. Ammu and A. S. Remya Ajai, "VLSI implementation of Boolean algebra based cryptographic algorithm," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016, pp. 2102-2105, doi: 10.1109/ICEEOT.2016.7755059.
19. *J. Anwer and M. Platzner*, "Boolean Difference Based Reliability Evaluation of Fault-Tolerant Circuit Structures on FPGAs," 2016 Euromicro Conference on Digital System Design (DSD), Limassol, Cyprus, 2016, pp. 1-8, doi: 10.1109/DSD.2016.35.
20. *S. A. Chin, J. Luu, S. Huda and J. H. Anderson*, "Hybrid LUT/Multiplexer FPGA Logic Architectures," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 4, pp. 1280-1292, April 2016, doi: 10.1109/TVLSI.2015.2451658.
21. *H. -G. Vu, N. -D. Bui, A. -T. Nguyen and ThanhBangLe*, "Performance Evaluation of Quine-McCluskey Method on Multi-core CPU," 2021 8th NAFOSTED Conference on Information and Computer Science (NICS), Hanoi, Vietnam, 2021, pp. 60-64, doi: 10.1109/NICS54270.2021.9701506.

22. *Garrido, M.* Simplifying Karnaugh Maps by Making Groups of Non-power-of-two Elements. Circuits Syst Signal Process 41, 5895–5902 (2022). https://doi.org/10.1007/s00034-022-02040-4
23. *M. Soeken* et al., "Practical exact synthesis," 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2018, pp. 309-314, doi: 10.23919/DATE.2018.8342027.
24. *H. Riener, W. Haaswijk, A. Mishchenko, G. De Micheli and M. Soeken*, "On-the-fly and DAG-aware: Rewriting Boolean Networks with Exact Synthesis," 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 2019, pp. 1649-1654, doi: 10.23919/DATE.2019.8715185.
25. *C. D. Murray and R. R. Williams*, "On the (non) NP-hardness of computing circuit complexity", Proc. 30th Conf. Comput. Complexity, pp. 365-380, 2015.
26. *A. Bernasconi, V. Ciriani, G. Trucco and T. Villa*, "Boolean Minimization of Projected Sums of Products via Boolean Relations," in IEEE Transactions on Computers, vol. 68, no. 9, pp. 1269-1282, 1 Sept. 2019, doi: 10.1109/TC.2019.2906201.
27. *A. Bernasconi, S. Cimato, V. Ciriani and M. C. Molteni*, "Multiplicative Complexity of XOR Based Regular Functions," in IEEE Transactions on Computers, vol. 71, no. 11, pp. 2927-2939, 1 Nov. 2022, doi: 10.1109/TC.2022.3141249.
28. *L. Amarú* et al., "Improvements to Boolean resynthesis," 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2018, pp. 755-760, doi: 10.23919/DATE.2018.8342108.
29. *A. Kagliwal and S. Balachandran*, "Set-Cover Heuristics for Two-Level Logic Minimization," 2012 25th International Conference on VLSI Design, Hyderabad, India, 2012, pp. 197-202, doi: 10.1109/VLSID.2012.70.
30. *M. Nazemi, H. Kanakia and M. Pedram*, "Heuristics for Million-scale Two-level Logic Minimization," 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Munich, Germany, 2021, pp. 1-7, doi: 10.1109/ICCAD51958.2021.9643572.

31. *P. Balasubramanian, A. Bernasconi, V. Ciriani and T. Villa*, "A Boolean Heuristic for Disjoint SOP Synthesis," 2021 24th Euromicro Conference on Digital System Design (DSD), Palermo, Italy, 2021, pp. 62-68, doi: 10.1109/DSD53832.2021.00019.
32. *B. A. de Abreu* et al., "Fast Logic Optimization Using Decision Trees," 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401664.
33. *J. Qiu, X. Gu, D. Ji, F. Li, P. He and B. Wang*, "Logic functions minimization algorithm based on recognition of essential prime implicants," 2010 Sixth International Conference on Natural Computation, Yantai, China, 2010, pp. 367-371, doi: 10.1109/ICNC.2010.5583819.
34. *M. Soeken, L. G. Amarù, P. -E. Gaillardon and G. De Micheli*, "Exact Synthesis of Majority-Inverter Graphs and Its Applications," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 11, pp. 1842-1855, Nov. 2017, doi: 10.1109/TCAD.2017.2664059.
35. *A. Chattopadhyay, L. Amarú, M. Soeken, P. -E. Gaillardon and G. De Micheli*, "Notes on Majority Boolean Algebra," 2016 IEEE 46th International Symposium on Multiple-Valued Logic (ISMVL), Sapporo, Japan, 2016, pp. 50-55, doi: 10.1109/ISMVL.2016.21.
36. *E. C. Ferraz and A. C. R. D. Silva*, "Linear Optimization Models for Maj-3 and Maj-5 Exact Synthesis," in IEEE Access, vol. 10, pp. 130518-130531, 2022, doi: 10.1109/ACCESS.2022.3229205.
37. Z. Chu, C. Shang, T. Zhang, Y. Xia, L. Wang and W. Liu, "Efficient Design of Majority-Logic-Based Approximate Arithmetic Circuits," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 12, pp. 1827-1839, Dec. 2022, doi: 10.1109/TVLSI.2022.3210252.
38. *X. Ge and S. Kimura*, "Topology-Based Exact Synthesis for Majority Inverter Graph," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 2022, pp. 3255-3259, doi: 10.1109/ISCAS48785.2022.9937527.

39. *C. -C. Ko, C. -C. Lin, Y. -C. Chen and C. -Y. Wang*, "Majority Logic Circuit Minimization Using Node Addition and Removal," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 3, pp. 642-655, March 2022, doi: 10.1109/TCAD.2021.3060648.
40. *E. C. Ferraz, J. V. O. Júnior, I. Grout and A. C. R. da Silva*, "Synthesis and Optimization of Majority Expressions through a Mathematical Model," 2020 33rd Symposium on Integrated Circuits and Systems Design (SBCCI), Campinas, Brazil, 2020, pp. 1-6, doi: 10.1109/SBCCI50935.2020.9189906.
41. *Z. Chu, W. Haaswijk, M. Soeken, Y. Xia, L. Wang and G. De Micheli*, "Exact Synthesis of Boolean Functions in Majority-of-Five Forms," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702141.
42. *A. Kabulov, A. Baizhumanov, I. Saymanov and M. Berdimurodov*, "Algorithms for Minimizing Disjunctions of Complex Conjunctions Based on First-Order Neighborhood Information for Solving Systems of Boolean Equations," 2022 International Conference of Science and Information Technology in Smart Administration (ICSINTESA), Denpasar, Bali, Indonesia, 2022, pp. 100-104, doi: 10.1109/ICSINTESA56431.2022.10041529.
43. *A. Kabulov, I. Normatov and A. Ashurov*, "Computational methods of minimization of multiple functions", Journal of Physics: Conference Series, vol. 1260, pp. 102007, 2019. DOI 10.1088/1742-6596/1260/10/102007
44. *T. Sasao*, "Easily Reconstructable Logic Functions," 2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL), Matsue, Japan, 2023, pp. 12-17, doi: 10.1109/ISMVL57333.2023.00014.
45. *Sasao, T.* (2024). Easily Reconstructable Functions. In: Classification Functions for Machine Learning and Data Mining. Synthesis Lectures on Digital Circuits & Systems. Springer, Cham. https://doi.org/10.1007/978-3-031-35347-5_9
46. *A. F. da Silva and A. A. Santos*, "Symbolic Manipulation for Optimization of Boolean Functions for Control of Pneumatic and Electropneumatic Circuits," 2021

16th Iberian Conference on Information Systems and Technologies (CISTI), Chaves, Portugal, 2021, pp. 1-4, doi: 10.23919/CISTI52073.2021.9476491.

47. *M. Berdimurodov and A. Baizhumanov*, "Algorithms for minimizing functions of the algebra of logic in the class of disjunctive normal forms and estimating their complexity," 2022 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 2022, pp. 1-5, doi: 10.1109/ICISCT55600.2022.10147007.
48. *W. Luo, H. Want, H. Zhong, O. Wei, B. Fang and X. Song*, "An Efficient Two-phase Method for Prime Compilation of Non-clausal Boolean Formulae," 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Munich, Germany, 2021, pp. 1-9, doi: 10.1109/ICCAD51958.2021.9643520.
49. *J. Echavarria, S. Wildermann and J. Teich*, "Design Space Exploration of Multi-output Logic Function Approximations," 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 2018, pp. 1-8, doi: 10.1145/3240765.3240795.
50. *J. Echavarria, S. Wildermann and J. Teich*, "Approximate Logic Synthesis of Very Large Boolean Networks," 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2021, pp. 1552-1557, doi: 10.23919/DATE51398.2021.9473952.
51. *Mishchenko A., Sasao T.* Large-Scale SOP Minimization Using Decomposition and Functional Properties // Proc. DAC'2003, Anaheim. – June.2003. – P.1-6. PDF.
52. *Рицар Б.Є.* Мінімізація булових функцій методом розчеплення кон'юнктермів // Управляющие системы и машины. – 1998. – № 5. – С. 14-22.
53. *Рицар Б.Є.* Редукція несуттєвих змінних системи булових функцій // Вісник НУ „Львівська політехніка". „Радіоелектроніка і телекомунікації". – 2001. – №428. – С. 184-189.
54. *B. Rytsar*, "Set-theoretical decomposition on the basis of symmetric functions", 2018 14th International Conference on Advanced Trends in Radioelecrtronics,

Telecommunications and Computer Engineering (TCSET), Lviv, UKraine, 2018, pp. 868-872.

55. *B. Rytsar and A. Shvaj*, "Experimental research of functional decomposition algorithm using the method of q-partition conjuncterms," 2008 International Conference on "Modern Problems of Radio Engineering, Telecommunications and Computer Science" (TCSET), Lviv, UKraine, 2008, pp. 144-144.
56. *Соломко М. Т.* Мінімізація частково визначених булевих функцій методом образних перетворень / Двадцять п'ятий Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування». - Запоріжжя-Кропивницький, 14-16 червня 2023 року. – ст. 183 – 190.
57. *Mykhaylo Solomko, Mykola Antoniuk, Ihor Voitovych, Yuliia Ulianovska, Nataliia Pavlova, Viacheslav Biletskyi*. Implementation of the method of figurative transformations to minimizing partially defined boolean functions // Eastern - European Journal of Enterprise Technologies. - Vol. 1, No 4(121). 2023. - Mathematics and cybernetics – applied aspects. pp. 6 - 25.
58. *Volodymyr Riznyk, Ivan Demydov, Mykola Medykovskyy, Vasyl Tesluyk, Mykhaylo Solomko*. Models of Intelligent Systems as the Toroidal Combinatorial Configurations / 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). - Lviv, Ukraine, 22-25 September, 2021.
59. *Mykhaylo Solomko, Petro Tadeyev, Liudmyla Zubyk, Stepaniia Babych, Yuliia Mala, Oksana Voitovych*. Implementation of the method of figurative transformations to minimizing symmetric boolean functions // Eastern - European Journal of Enterprise Technologies. - Vol. 4, No 4(112). 2021. - Mathematics and cybernetics – applied aspects. pp. 23 - 39.
60. *Соломко М. Т.* Мінімізація булевих функцій у поліномному та змішаному базисах методом образних перетворень / Двадцять третій Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування». - Запоріжжя-Кропивницький, 13-15 травня 2021 року. – ст. 153 – 159.

61. *Solomko M., Batyshkina Iu., Khomiuk N., Ivashchuk Ya., Shevtsova N*. Developing the minimization of a polynomial normal form of Boolean functions by the method of figurative transformations // Eastern - European Journal of Enterprise Technologies. - Vol. 2, No 4(109). 2021. - Mathematics and cybernetics – applied aspects. pp. 22 - 37.
62. *Solomko M*. Developing an algorithm to minimize boolean functions for the visual-matrix form of the analytical method // Eastern - European Journal of Enterprise Technologies. - Vol. 1, No 4(109). 2021. - Mathematics and cybernetics – applied aspects. pp. 6 - 21.
63. *Volodymyr Riznyk, Mykhaylo Solomko, Yuriy Tsymbal, Oleg Riznyk, Daniel Skrybailo-Leskiv and Roman Sydorenko*. Combinatorial Optimization of CAD Systems / 2021 IEEE 16-th International Conference on the Experience of Designing and Application of CAD Systems (CADSM). - Lviv, Ukraine, 22-26 February, 2021.
64. *Solomko M., Batyshkina Iu., Voitovych I., Zubyk L., Babych S., Muzychuk K*. Devising a method of figurative transformation for minimizing boolean functions in the implicative basis // Eastern - European Journal of Enterprise Technologies. - Vol. 6, No 4(108). 2020. - Mathematics and cybernetics – applied aspects. pp. 32 - 47.
65. *Попович В. І., Соломко М. Т*. Мінімізація повного суматора бінарних кодів за допомогою фіктивної змінної / IX Наукова конференція «НАУКОВІ ПІДСУМКИ 2020 РОКУ». Збірка наукових праць. – Харків, ІХ.: Технологічний Центр, 2020. – 70 с.
66. *Solomko M., Zubyk L., Zubyk Y., Ivanytska A*. Modified Algorithm for Transformation of Boolean Functions / VII INTERNATIONAL CONFERENCE "Information Technology and Interactions (Satellite)" 04 December, 2020. Conference Proceedings. Taras Shevchenko National University of Kyiv: Stylos, 2020. 388 p.
67. *Solomko M., Khomiuk N., Ivashchuk Y., Nazaruk V., Reinska V., Zubik L., Popova A*. Implementayion of the method of transformation for image minimizing

the Sheffer functions // Eastern - European Journal of Enterprise Technologies. - Vol. 5, No 4(107). 2020. - Mathematics and cybernetics – applied aspects. pp. 19 - 34.

68. *Різник В. В., Соломко М. Т., Тадеєв П. О, Назарук В. Д., Зубик Л. В., Волошин В. С*. Алгоритм мінімізації булевих функцій методом оптимального комбінування послідовності образних перетворень // Східно-Європейський журнал передових технологій.- Vol. 3, No 4(105). 2020. - Математика та кібернетика - прикладні аспекти. ст. 43 - 60.
69. *Соломко М. Т., Замрій Б. А*. Оптимальне чергування логічних операцій при мінімізація булевих функцій / Двадцять другий Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування». - Запоріжжя-Кропивницький, 15-16 травня 2020 року. – ст. 161 – 169.
70. *Solomko M., Tadeyev P, Nazaruk V, Khariv N*. Optimal performance of 16-bit Acyclic adders of binary codes // Eastern-European Journal of Enterprise Technologies.- Vol. 3, No 4(99). 2019. - pp 21 - 36.
71. *Solomko M., Tadeyev P, Zubyk Y, Hladka O*. Reduction and is optimal of speed of acyclic adders of binary codes // Eastern-European Journal of Enterprise Technologies.- Vol. 1, No 4(97). 2019. - pp 40 - 53.
72. *Solomko M. T*. Boolean functions minimization by the method of figurative transformations / Technology transfer: fundamental principles and innovative technical solutions: materials of the scientific conference. Thalin, Estonia, EU, November 2018. P. 62–68.
73. *Дубич Л. Ю., Соломко М. Т*. Мінімізація булевих функцій методом образних перетворень / VII Міжнародна науково-практична конференція Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах 8-10 листопада 2018 року м. Чернівці, Україна
74. *Riznyk V., Solomko M*. Minimize of conjustable normal forms of boolean functions combinatorial method // Technology audit and production reserves.- Vol 5/2 (43), 2018. - pp 42 - 55.

75. *Riznyk V., Solomko M.* Research of 5-bit boolean functions minimization protocols by combinatorial method // Technology audit and production reserves.- Vol 4/2 (42), 2018. - pp 41 - 52.
76. *Solomko M.* Optimization of the acyclic adders of binary codes // Technology audit and production reserves.- Vol 3/2 (41), 2018. - pp 55 - 65.
77. *Соломко М. Т.* Мінімізація 5-розрядних булевих функцій комбінаторним методом / Двадцятий Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування». - Кропивницький, 13-14 квітня 2018 року. – ст. 120 – 127.
78. *Riznyk V., Solomko M.* Application of super-sticking algebraic operation of variables for boolean functions minimization by combinatopial method // Technology audit and production reserves.- Vol 6/2 (38), 2017. - Kharkov, Ukraine, pp 60 - 76.
79. *Різник В. В., Соломко М. Т.* Комбінаторний метод мінімізації булевих функцій // Науковий часопис національного університету «Львівська політехніка» Комп'ютерні системи та мережі, Львів. Вид-во НУ «Львівська політехніка», 2017, Випуск 881. – 135 - 152.
80. *Riznyk V., Solomko M.* Minimization of boolean functions by combinatorial method // Technology audit and production reserves.- Vol 4/2 (36), 2017. – Kharkov, Ukraine, pp 49 - 64.
81. *Соломко М. Т.* Комбінаторний метод мінімізації булевих функцій / ІІІ Всеукраїнська науково-практичної конференція "Комп'ютерне моделювання та програмне забезпечення інформаційних систем і технологій" (КМПЗ-2017) ст. 28 - 30.
82. *Соломко М. Т.* Рекурсивні бінарні коди / Х Всеукраїнська науково-практична конференція «Комп'ютерні технології: наука і освіта» 30-31 березня 2017 р., м. Київ ст. 72 - 75.
83. *Ольшанський П. В., Соломко М. Т.* Побудова комплексного алгоритму захисту програмного забезпечення від несанкціонованого використання та декодування / Х Всеукраїнська науково-практична конференція

«Комп’ютерні технології: наука і освіта» 30-31 березня 2017 р., м. Київ ст. 56 - 59.

84. *Solomko M. T., Olshanskyi P. V.* The Parallel Acyclic Adder / CADSM'2017, February 21 – 25, 2017, Lviv-Slavske, Ukraine, pp 125 - 129.
85. *Соломко М. Т., Круліковський Б. Б.* Оптимізація перенесення при додаванні двійкових чисел у теоретико-числовому базисі Радемахера // Науковий часопис національного університету «Львівська політехніка» Комп’ютерні системи та мережі, Львів. Вид-во НУ «Львівська політехніка», 2016, Випуск 857. – 88 - 102.
86. *Solomko M., Zubyk L., Olshansky P., Nazaruk V.* Summation of binary codes without carry // Eastern-European Journal of Enterprise Technologies.- Vol 4, No 4(82) (2016). - Kharkov, Ukraine, pp 28 - 41.
87. *Solomko M. T., Krulykovskyi B. B.* Study of carry optimization while adding binary numbers in the rademacher number-theoretic basis // Eastern-European Journal of Enterprise Technologies.- Vol 3, No 4(81) (2016). - Kharkov, Ukraine, pp 56 - 63.
88. *Solomko M. T.* The Tabular Transfer Mode in the Rademacher NTB Adders / TCSET'2016, February 23 – 26, 2016, Lviv-Slavske, Ukraine, pp 483 - 487.
89. *Соломко М. Т., Круліковський Б. Б., Николайчук Я. М.* Паралельний суматор без перенесення на логічних елементах XAND // Вісник Національного університету "Львівська політехніка" Комп’ютерні системи та мережі, Львів. Вид-во НУ «Львівська політехніка», 2015, Випуск 830. – 145 - 159.
90. *Рицар Б.Є., Мінзюк В.В.* Один спосіб зображення кон’юнктермів // Вісник ДУ „Львівська політехніка”. „Радіоелектроніка і телекомунікації”. – 1999. – №367. – С. 138-142.
91. *Rytsar B., Minzyuk V., Sizonov Y.* Minimization Method of Boolean Functions System // Proc. Conf. TCSET’2000, February 14-19, 2000. – Lviv-Slavsko: Lvivska polytechnika, 2000. – P. 86-87.

92. *Рицар Б.Є., Мінзюк В.В.* Теоретико-множинна модифікація мінімаксного методу покриття булових функцій // Управляющие системы и машины. – 2005. – № 5. – С. 43-47.

93. *Rytsar B., Minzyuk V.* The Set-theoretical Modification of Boolean Functions Minimax Covering Method // Proc. Conf. TCSET'2004, February 24-28, 2004. – Lviv- Slavsko: Lvivska polytechnika, 2004. – P.46-48.

94. *Шклярський В.І., Матієшин Ю.М., Баланюк Ю.В., Мінзюк В.В.* Алгоритмічне забезпечення роботи телевізійного сканувального оптичного мікроскопа під час дослідження динамічних мікрооб'єктів // Вісник НУ „Львівська політехніка". „Радіоелектроніка і телекомунікації". – 2017. – №885. – С. 15-21.

95. *Matiieshyn, Y., Minziuk, V., Mankovskyy, S.* Algorithmic support of the television scanning optical microscope in the study of microobjects // Proc. Conf. UkrMiCo'2019, September 9-13, 2019. – 2019. – P. 368-373.

96. Пат. 149637 Україна. МПК H04B1/00 H04B1/04. Радіохвильовий сенсор / *В. І. Оборжицький, В. Г. Сторож, Ю. М. Матієшин, В. Г. Протасевич, В. В. Мінзюк*; заявник та власник патенту Національний університет "Львівська політехніка". – № u202103668 ; заявл. 25.06.2021 ; опубл. 24.11.2021, Бюл. № 47.

97. *Мінзюк В.В.* Спосіб синтезування кон'юнктермів булових функцій // Вісник НУ „Львівська політехніка". „Радіоелектроніка і телекомунікації". – 2004. – №508. – С. 562-262.

98. *Мінзюк В.В.* Спосіб спрощення задачі покриття булових функцій // Вісник НУ „Львівська політехніка". „Радіоелектроніка і телекомунікації". – 2005. – №534. – С. 24-28.

99. *Мінзюк В.В.* Спосіб сортування цілих чисел для задач мінімізації бульових функцій // Вісник НУ „Львівська політехніка". „Радіоелектроніка і телекомунікації". – 2011. – №705. – С. 135-137.

100. *Мінзюк В.В.* Модифікація методу порозрядного вирощування простих кон'юнктивних термів булових функцій // Моделювання та інформаційні

технології. Зб. наук. пр. ІПМЕ НАН України. – К.: 2012. – №65. – С. 129-134.

101. *Мінзюк В.В.* Метод пошуку простих кон'юнктивних термів булових функцій побітовим розбиттям множини // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – К.: 2013. – №66. – С. 95-103.
102. *Мінзюк В.* Метод формування розгортки телевізійного сканувального оптичного мікроскопа // Інфокомунікаційні технології та електронна інженерія. – Львів: 2022. – Вип. 2, №2. – С. 88-95.
103. *Мінзюк В.* Метод мінімізації булевих функцій для проєктування цифрових комбінаційних схем // Інфокомунікаційні технології та електронна інженерія. – Львів: 2023. – Вип. 3, №1. – С. 146-152.
104. *Minzyuk V.* Modification of conjuncterms splitting method of boolean functions minimization // Proc. Conf. TCSET'2006, February 28 – March 4, 2006. – Lviv-Slavsko: Lvivska polytechnika, 2006. – P.81-82.
105. *Minzyuk V.* Integers sorting method for boolean functions minimization // Proc. Conf. TCSET'2012, February 28-24, 2012. – Lviv- Slavsko: Lvivska polytechnika, 2012. – P.61.
106. *Minzyuk V.* Technique of Boolean Functions Conjuncterm Synthesis // Proc. Conf. CADSM'2005, February 23-26, 2005. – Lviv- Polyana: Lvivska polytechnika, 2005. – P.207.
107. *Мінзюк В.В.* Модифікація методу порозрядного вирощування простих кон'юнктивних термів булових функцій // Материалы междунар. молодеж. науч.-технич. конф. студ., аспирантов и ученых. РТ-2006, 17-21 апреля 2006. – Севастополь : Изд-во СевНТУ, 2006. – С. 309.
108. *V. R. Pratt,* "The effect of basis on size of Boolean expressions," 16th Annual Symposium on Foundations of Computer Science (sfcs 1975), USA, 1975, pp. 119-121, doi: 10.1109/SFCS.1975.29.
109. Hodes L, Specker E Length of Formulas and Elimination of Quantifiers I. Contributions to Math. Logic, (K Schuette, ed) pp 175–188, Amsterdam: North Holland 1968 (appears in 24)

110. *Bonnet, R., Si-Kaddour, H*. Comparison of Boolean algebras. Order 4, 273–283 (1987). https://doi.org/10.1007/BF00337890

111. *Кметь А.Б.* Новий підхід до мінімізації логіковий функцій. Метод максимальних сегментів // Управляющие системы и машины. – 2001. – №2. – С. 10-21.

112. *A. Kabulov and M. Berdimurodov*, "Parametric Algorithm for Searching the Minimum Lower Unity of Monotone Boolean Functions in the Process Synthesis of Control Automates," 2021 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 2021, pp. 1-4, doi: 10.1109/ICISCT52966.2021.9670370.

ДОДАТОК А

Акти використання та впровадження результатів дисертаційних досліджень

"ЗАТВЕРДЖУЮ"
Проректор
з наукової роботи
Національного університету
"Львівська політехніка"
Іван ДЕМИДОВ
"01" травня 2023 року

АКТ

про впровадження результатів кандидатської дисертаційної роботи Мінзюка Вадима Володимировича "Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв"

Комісія у складі голови – начальника науково-дослідної частини Національного університету "Львівська політехніка" ст. досл. Небесного Р. В. та членів: заступника начальника планово-фінансового відділу (керівника планово-фінансової групи науково-дослідної частини) Чулой Т. М., завідувача кафедри РЕПС доц. Оборжицького В.І. склали цей Акт про те, що результати кандидатської дисертації Мінзюка В.В. "Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв" використано в таких роботах виконаних в Національному університеті "Львівська політехніка" з безпосередньою участю автора:

НДР ДБ/ВЕРБАЛЬ "Розробка макромоделей відмовостійких радіоелектронних засобів" (№ держреєстрації 0104U002291) – розроблено числове зображення кон'юнктермів булових функцій, удосконалено метод мінімізації булових функцій розчепленням кон'юнктермів, удосконалено метод побітового вирощування простих кон'юнктермів булових функцій, виконано порівняльний синтез макромоделей перетворювача кодів за допомогою цих методів.

НДР ДБ\Шум "Алгоритми екстракції даних методами ієрархічної кластеризації для важко вирішуваних комбінаторних задач великої розмірності" (№ держреєстрації 0108U000326) – розроблено модифікацію методу побітового сортування цілих чисел для мінімізації логікових функцій, метод мінімізації булових функцій побітовим сортуванням зі склеюванням, застосовано розроблені методи мінімізації до задачі кластеризації даних.

Голова комісії
Начальник НДЧ
д.т.н., ст.досл. Роман НЕБЕСНИЙ

Члени комісії

Заст. нач. ПФВ НУЛП
(керівник ПФГ НДЧ) Тетяна ЧУЛОЙ

Зав. каф. РЕПС
д.т.н., доц. Валерій ОБОРЖИЦЬКИЙ

ЛЬВІВСЬКИЙ ЦЕНТР ІНСТИТУТУ КОСМІЧНИХ ДОСЛІДЖЕНЬ
НАЦІОНАЛЬНОЇ АКАДЕМІЇ НАУК УКРАЇНИ
та ДЕРЖАВНОГО КОСМІЧНОГО АГЕНТСТВА УКРАЇНИ
(ЛЦ ІКД НАН та ДКА України)
79060, м. Львів-60, вул. Наукова, 5-а

"ЗАТВЕРДЖУЮ"
Директор
ЛЦ ІКД НАН та ДКА України
Адольф ЛУКЕНЮК
"11" травня 2023 р.

АКТ

про використання результатів кандидатської дисертаційної роботи
Мінзюка Вадима Володимировича

"Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв"

Даний акт складений про те, що у ЛЦ ІКД НАН та ДКА України для проектування цифрової комбінаційної схеми керування мережею радіохвильових сенсорів використані результати дисертаційної роботи Мінзюка В. В. "Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв", представленої на здобуття наукового ступеня кандидата технічних наук, а саме, метод мінімізації булових функцій побітовим розбиттям множини кон'юнктермів, впровадження якого на етапі синтезу цифрової комбінаційної схеми дало змогу:

- зменшити інформаційну ємність програмованої логікової інтегральної схеми (ПЛІС);
- розмістити додатковий вузол в тій же ПЛІС на вивільненій площі.

У відділі №111 ЛЦ ІКД НАН та ДКА України було виготовлено прототип спроектованої цифрової комбінаційної схеми і виконано його експериментальне дослідження, яке підтвердило стовідсоткову точність реалізації заданих логікових функцій.

Заступник директора Віра ПРОНЕНКО

"ЗАТВЕРДЖУЮ"
Директор
ТзОВ «Міта-Техніка»

_____ Ростислав НАКОНЕЧНИЙ

"16" травня 2023 р.

АКТ

про використання результатів кандидатської дисертаційної роботи
Мінзюка Вадима Володимировича

"Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв"

Даний акт складений про те, що розроблений в дисертаційній роботі Мінзюка Вадима Володимировича удосконалений метод розчеплення кон'юнктермів для мінімізації булових функцій використано для проектування універсального сигнального перетворювача промислової автоматики на основі мережі первинних перетворювачів із оцифрованими вихідними сигналами.

Ефект, отриманий від впровадження запропонованого удосконаленого методу на етапі синтезу цифрової комбінаційної схеми, полягає у зменшенні на 5% кількості комірок ПЛІС, які необхідні для реалізації заданих булевих функцій, що дало змогу використати вивільнені ресурси для реалізації інших функцій, зокрема функції самоконтролю.

Даний акт не є підставою для фінансових розрахунків.

Головний конструктор комп'ютерних систем _____ Сергій ЦАП

"ЗАТВЕРДЖУЮ"
Проректор
з науково-педагогічної роботи
Національного університету
"Львівська політехніка"
Олег ДАВИДЧАК
"03" травня 2023 року

АКТ

про впровадження у навчальний процес
Національного університету "Львівська політехніка"
результатів кандидатської дисертаційної роботи
Мінзюка Вадима Володимировича
"Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв"

Даний акт складено про те, що у Національному університеті "Львівська політехніка" на кафедрі "Радіоелектронні пристрої та системи" в лекційних курсах та лабораторних заняттях для студентів спеціальності 172 "Телекомунікації та радіотехніка" знайшли застосування наступні результати дисертаційної роботи Мінзюка В. В. " Розроблення та дослідження оптимальних алгоритмів мінімізації булових функцій у довільному логіковому базисі для проектування цифрових комбінаційних пристроїв" у курсах "Цифрова схемотехніка" та "Програмно-апаратні засоби в телекомунікаціях та радіотехніці" впроваджено вивчення таких методів для мінімізіції логікових функцій:

- удосконалений метод мінімізації розчепленням кон'юнктермів;
- удосконалена теоретико-множинна модифікація мінімаксного методу покриття булових функцій;
- удосконалений метод побітового вирощування простих кон'юнктермів;
- метод мінімізації побітовим сортуванням зі склеюванням.

Ефект від використання названих результатів полягає в ознайомленні майбутніх спеціалістів із сучасними методами проектування цифрових комбінаційних пристроїв та сприяють підвищенню рівня підготовки спеціалістів.

Директор інституту телекомунікацій,
радіоелектроніки та електронної техніки
д.т.н., проф. Богдан СТРИХАЛЮК

Голова НМК спеціальності 172
«Телекомунікації та радіотехніка»
д.т.н., доц. Леонід ОЗІРКОВСЬКИЙ

Зав. каф. РЕПС
д.т.н., доц. Валерій ОБОРЖИЦЬКИЙ

ДОДАТОК Б

Блок-схеми алгоритму розчеплення кон’юнктермів

Рисунок Б.1 – Блок-схема алгоритму процедури формування матриці розчеплених кон’юнктермів

Алгоритм процедури формування матриці розчеплення (рисунок Б.1) передбачає застосування процедури формування стовпця розчеплених кон’юнктермів до кожного кон’юнктерма, що належить твірному вектору.

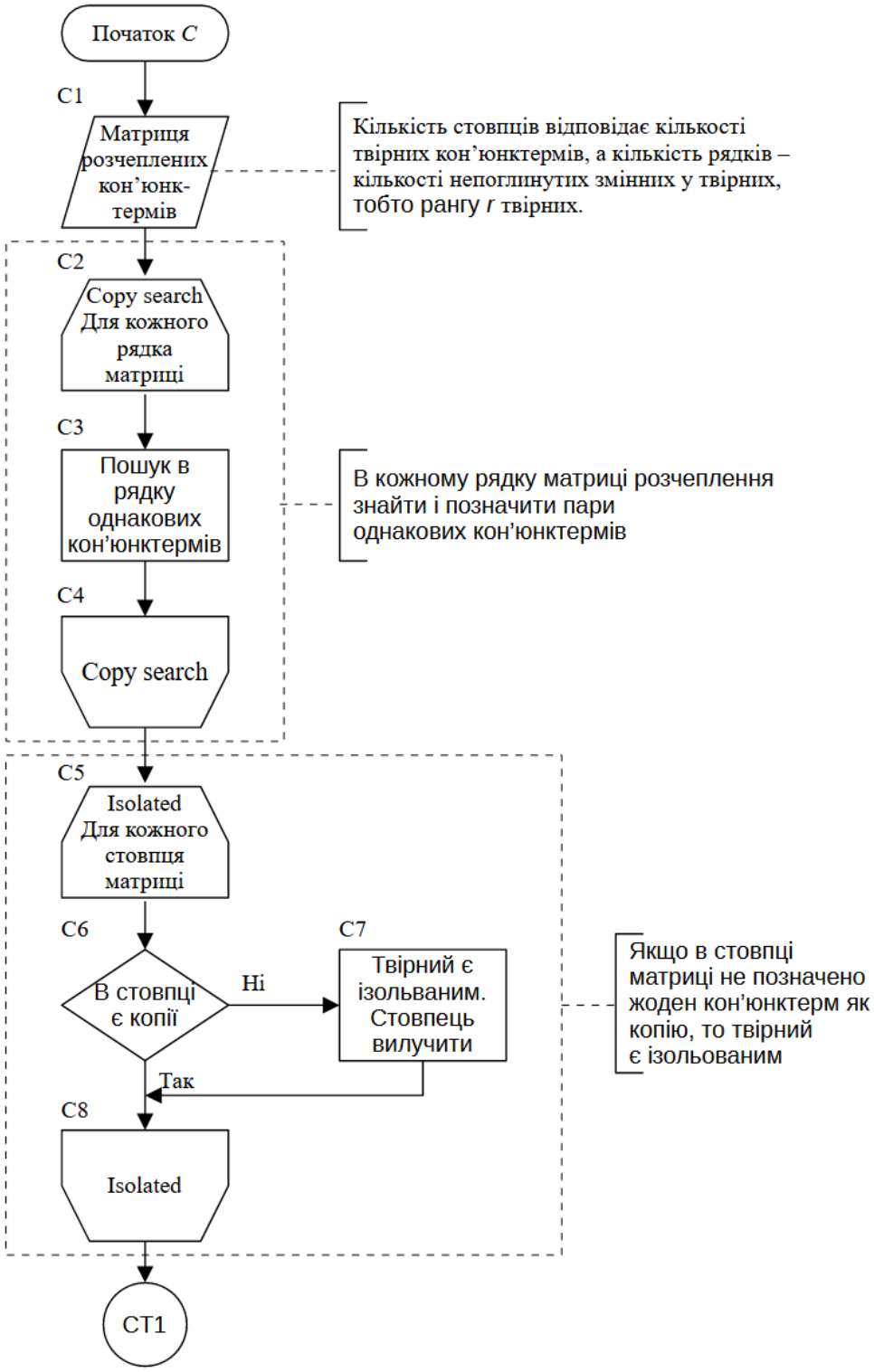


Рисунок Б.2 – Блок-схема алгоритму процедури покриття матриці розчеплення кон'юнктермів (початок)

Алгоритм покриття матриці розчеплення (рисунки Б.2 та Б.3) передбачає вибір вектора, що містить найбільшу кількість кон'юнктермів-копій.

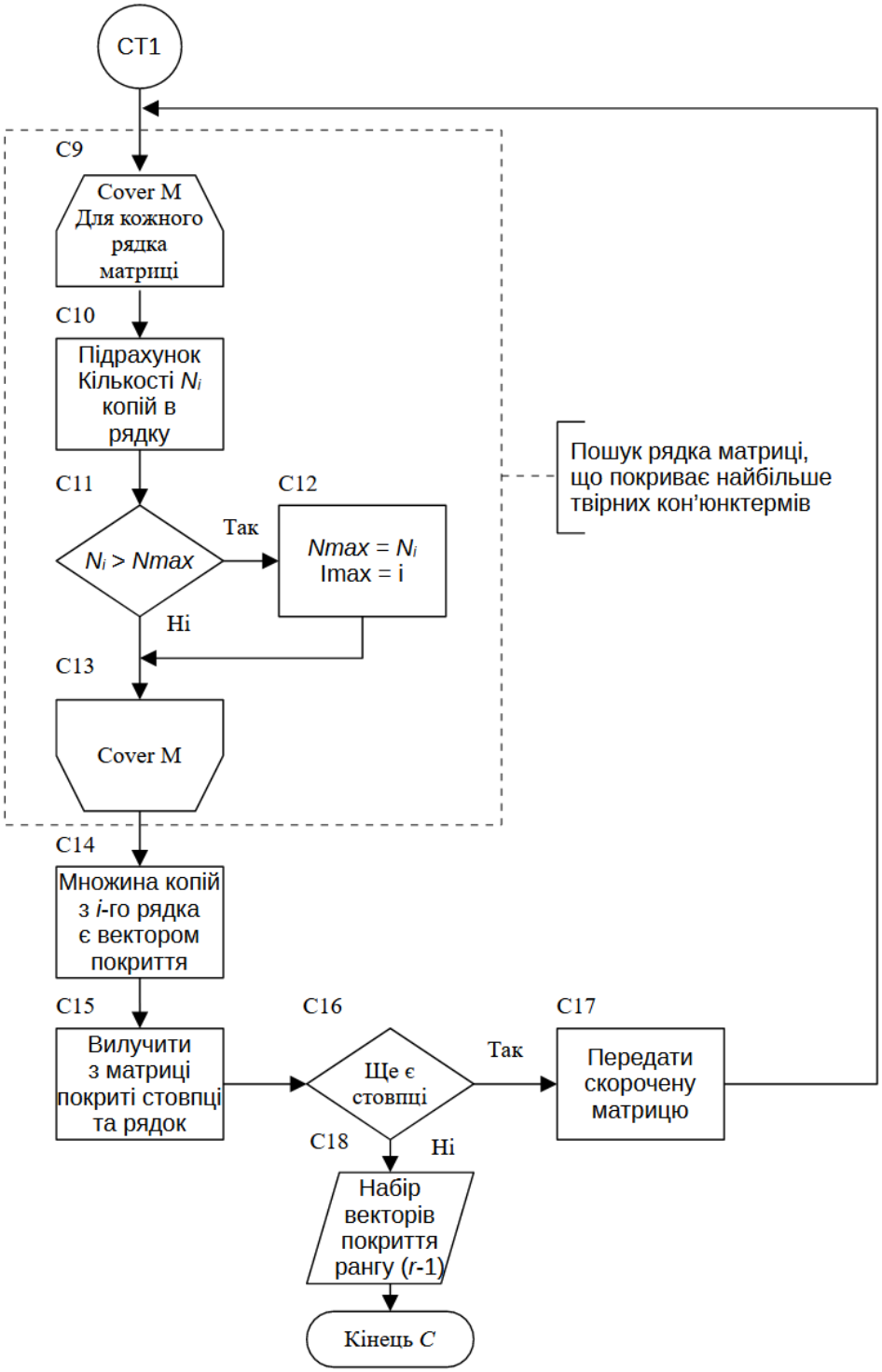


Рисунок Б.3 – Блок-схема алгоритму процедури покриття матриці розчеплення кон'юнктермів (продовження)

Алгоритм методу розчеплення кон'юнктермів полягає у рекурентному виклику послідовності процедур формування матриці розчеплення та її покриття для усіх одержаних внаслідок розчеплення векторів. Вглибину кількість рекурентних викликів не перевищує кількості змінних заданої булової функції.

ДОДАТОК В

Схеми перетворювача кодів

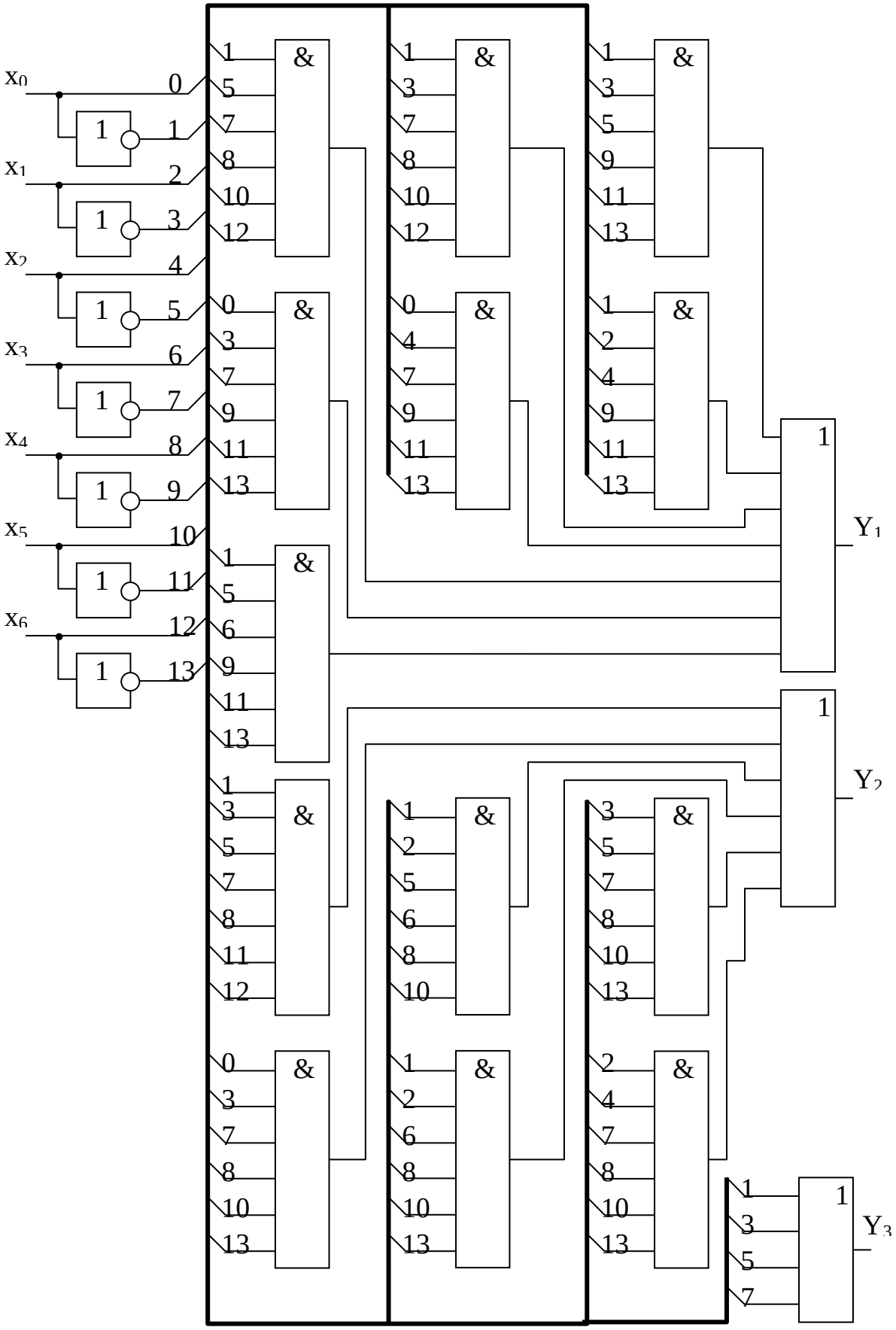


Рисунок В.1 – Схема перетворювача кодів

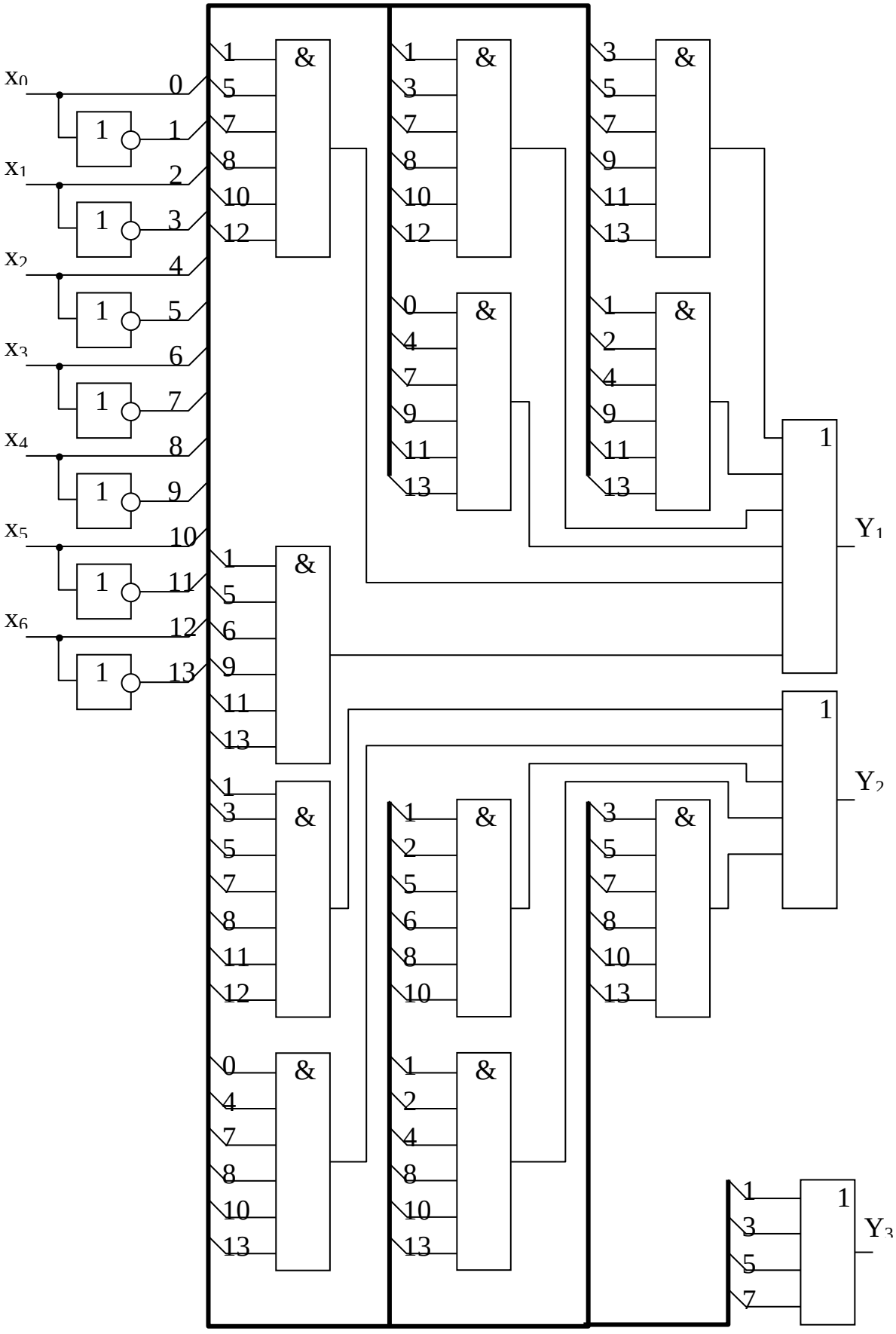


Рисунок В.2 – Схема перетворювача кодів