виході відповідного елементу I (рис. 5) D5.1 або D5.2. Тоді на виході відповідно формується сигнал $nR = 0$. Це призводить до закривання вихідних транзисторів відповідного комутатора. На інші входи елементів I D5.1 та D5.2 подаються сигнали з виходів відповідно RB6 та RB7 процесора. Цими сигналами процесор має можливість також закривати вихідні транзистори.

Висновок. Запропонована система керування забезпечує керування вібраційними пристроями, що містять один або два віброзбудники та потребують регулювання як амплітуди, так і частоти коливань. Також при керування двома віброзбудниками забезпечене регулювання зсуву фаз керуючих сигналів.

1. Автоматическая загрузка технологических машин: Справочник/ И. С. Бляхеров, Г. М. Варьяш, А. А. Иванов и др.; Под общ. ред. И. А. Клусова. – М.: Машиностроение, 1990. – 400 с.
2. Мельничук І.М., Таянов С.А., Шенбор В.С., Беспалов Л.А. Мультичастотна система керування однотактним електромагнітним віброзбудником резонансними вібраційними пристроями // Наукові вісті Інституту менеджменту та економіки «Галицька академія». – 2006. – №2(10)..
3. Зелінський І.Д. Система керування електромагнітним віброзбудником // Автоматизація виробничих процесів у машинобудуванні та приладобудуванні, Український міжвідомчий науково-технічний збірник, Львів: Держ. ун-ту "Львівська політехніка". – 2003. – Вип. 37. – С.3–6.
4. http://www.irf.com/product-info/datasheets/data/ir2130.pdf

УДК 681.325

С.І. МЕЛЬНИЧУК

ПВНЗ "Галицька академія"

МЕТОДИ ТА АЛГОРИТМИ ДЕКОДУВАННЯ ДВІЙКОВИХ ЦИКЛІЧНИХ ПОСЛІДОВНОСТЕЙ ГАЛУА В АВТОМАТИЗОВАНИХ СИСТЕМАХ КОНТРОЛЮ СПОЖИВАННЯ ЕНЕРГОНОСІВ

Проведено систематизацію та аналіз методів декодування станів первинних дискретних інформаційних джерел, представлених у базисі Галуа.

A systematization and analysis methods will serve as decoding discrete information sources presented in basis Galois.

Вступ. Традиційно реалізація автоматизованих та автоматичних систем контролю витрати та споживання енергоносіїв ґрунтується на використанні первинних джерел інформації – переважно це лічильники, розташовані безпосередньо на замірних ділянках чи пунктах контролю.

Доцільно зазначити, що сучасна промисловість при реалізації перетворювачів інтегрального типу обмежується унітарним базисом. Фактично більшість лічильних механізмів реалізовано як імпульсні джерела інформації, в яких кількість одиничних відліків відповідає об'єму чи кількості контрольованого параметра [1,2]. Недоліками такого підходу є незадовільна точність перетворення, що ускладнює апаратно-програмні засоби їх оброблення. Крім того, виникає необхідність реалізації спеціалізованих каналів обміну даним, як б забезпечували надійний захист від імпульсних завад, особливо у випадку промислової реалізації чи значної віддаленості пунктів контролю автоматизованої системи.

Застосування інтегрально-імпульсної технології базису Галуа порівняно з іншими базисами (Радемахера, Крестенсона тощо) має ряд переваг, зокрема мінімальний розмір інформаційної посилки, підвищену надійність в умовах дії промислових завад тощо [3,4]. Практичне застосування циклічних послідовностей Галує є надійною основою для представлення (кодування) станів дискретних джерел інтегрального типу та ефективним засобом реалізації низових інформаційних каналів обміну даними автоматизованих систем контролю.

Аналіз досліджень та публікацій. Часто при дослідженні чи автоматичному контролі об'єктів доводиться опрацьовувати дані, що описують поведінку таких джерел інформації, як функцію від часу, що зумовлює необхідність опрацювання інтегральних та миттєвих значень відповідних параметрів [1].

Перспективним напрямком розвитку теорії та технології побудови первинних інформаційних джерел інтегрального типу є застосування інтегрально-імпульсної технології, що реалізується на двійкових циклічних М-послідовностях базису Галуа [2,3]. Основною перевагою застосування цього базису є формування біт-орієнтованого потоку інтегральних даних, які одночасно несуть інформацію про інтегральне та миттєве значення квантових процесів, що є визначальним фактором для низових джерел автоматизованих систем [3].

Фактично можливості інтегральної інформаційної технології при кодуванні дискретних станів об'єктів в автоматизованих та автоматичних системах контролю і перспективи розроблення ефективних методів зворотного перетворення обумовлюють необхідність досліджень у цьому напрямі, зокрема для розподілених автоматизованих систем обліку енергоносіїв.

Формування цілей роботи. Одним з важливих науково-технічних завдань є розроблення та систематизація процедур зворотного перетворення (декодування) станів дискретних інтегральних джерел, реалізованих на циклічних послідовностях Галуа у відповідне значення двійкового, десяткового чи іншого базису. Вирішення такого завдання дасть змогу чітко і обґрунтовано реалізувати вибір обчислювальних ресурсів, які необхідно забезпечити при реалізації відповідних компонентів первинних джерел даних автоматизованих систем контролю витрати енергоносіїв.

В загальному випадку методи декодування одномірних циклічних послідовностей, кодів Галуа, можна розділити на кілька категорій, які відрізняються підходом до організації збереження еквівалентних значень фрагментів коду та способом виявлення (локалізації) шуканого фрагмента послідовності.

Методи фіксованого пошуку (табличні). Табличні методи декодування, які ґрунтуються на використанні спеціалізованих таблиць з фрагментами кодової послідовності та їх відповідними значеннями.

Метод таблиці послідовних значень. Такий підхід ґрунтується на використанні впорядкованої таблиці кодів KeyCode та відповідних значень DecCode (табл. 1). Для оптимізації пошуку таблиці доповнюють індексом ключового поля, що традиційно реалізується на основі бінарних дерев пошуку. Для створення робочої таблиці вибирається необхідна довжина М-послідовності, задається базовий (ключовий) фрагмент та відповідний метод формування.

Використання індексів за полем KeyCode вибраної М-послідовності забезпечує зменшення часу виявлення (локалізації) конкретного кодового фрагмента. Приклад функції локалізації може бути описаний фрагментом програми на мові CA-Visual Object 7.xx:

```
INDEX ON KeyCode TAG KeyCode
SET INDEX TO TAG KeyCode
SEEK VarCode
```

IF !FOUND(); RETURN –1
ELSE; RETURN DecCode
ENDIF,

де KeyCode – поле відомих кодів; VarCode – заданий (шуканий) кодовий фрагмент; DecCode – значення коду в десятковій чи двійковій системі числення.

Таблиця 1

Кодові фрагменти та відповідні значення \в десятковому представленні

Фрагменти кодів послідовності (KeyCode)	Значення (DecCode)
…	…
011110101100100	1024
111101011001000	1025
111010110010011	1026
110101100100110	1027
…	…
011011100001010	11563
110111000010100	11564
…	…

Слід зазначити, що індексний файл потребує додаткового дискового простору, близько 40% від розміру вихідної таблиці даних (табл.2). При такій реалізації швидкість виявлення залежить від кількості можливих комбінацій (довжини) коду, тактової частоти процесора, об'єму оперативної пам'яті, швидкості обміну з пристроєм збереження (накопичення) даних.

Таблиця 2

Об’єми таблиць формату DBF для методу таблиці послідовних значень

Довжина ключа (біт)	Кількість кодів М-послідовності	Об’єм таблиці кодів / індексного файла (Кб)
4	15	0.23 / 0.07
10	1023	16 / 4.6
14	16383	250 / 7.25
17	131071	2000 / 580.5
20	1048575	21000 / 6000.2

Метод вектора послідовних значень. Особливістю реалізації такого методу є бітове представлення кодової послідовності Галуа BitCode та відповідних значень DecCode (табл.3). Фактично кодова послідовність зберігається у вигляді бітового вектора послідовних значень елементів послідовності.

Таблиця 3

Біти послідовності та відповідні значення в десятковому представленні

Біти кодів (BitCode)	…	1	1	0	1	…	1	1	…
Значення (DecCode)	…	125	126	127	128	…	11563	11564	…

Отже, кожному наступному біту послідовності Галуа відповідає чергове числове значення. Процедура локалізації послідовності здійснюється шляхом послідовного перебору, бітового порівнювання, послідовно розташованих елементів рядка таблиці BitCode з бітами заданого (прийнятого) фрагмента.

Таблиця 4

Об'єми таблиць формату DBF
для методу вектора послідовних значень

Довжина ключа (біт)	Кількість кодів М-послідовності	Об'єм таблиці кодів / індексного файла (Кб)
4	15	0.16 / 0.0
10	1023	6.24 / 0.0
14	16383	115 / 0.0
17	131071	1049 / 0.0
20	1048575	9438 / 0.0

Перевагою описаного методу є простота реалізації, проте такий підхід не дає змоги застосовувати сучасні технології обробки впорядкованих таблиць (баз даних). Порівнюючи з попереднім методом, слід зазначити, що тут при декодуванні ресурс пам'яті замінюється обчислювальним ресурсом.

Методи послідовного наближення (перебору). Найпростіші методи декодування, які ґрунтуються на послідовному переборі елементів (бітів) коду з подальшим порівнюванням отриманих фрагментів послідовності.

Метод одностороннього наближення. Суть методу полягає в проведенні інкрементного перебору можливих варіантів кодової послідовності, починаючи від базової (ключової), до виявлення збігу із заданим (прийнятим) фрагментом. Швидкість локалізації переданої в процедуру пошуку кодової послідовності залежить від кількості можливих комбінацій (довжини) коду, тактової частоти процесора та об'єму статичної пам'яті. Блок-схему алгоритму процедури локалізації необхідного фрагмента послідовності Галуа подано на рис.1а.

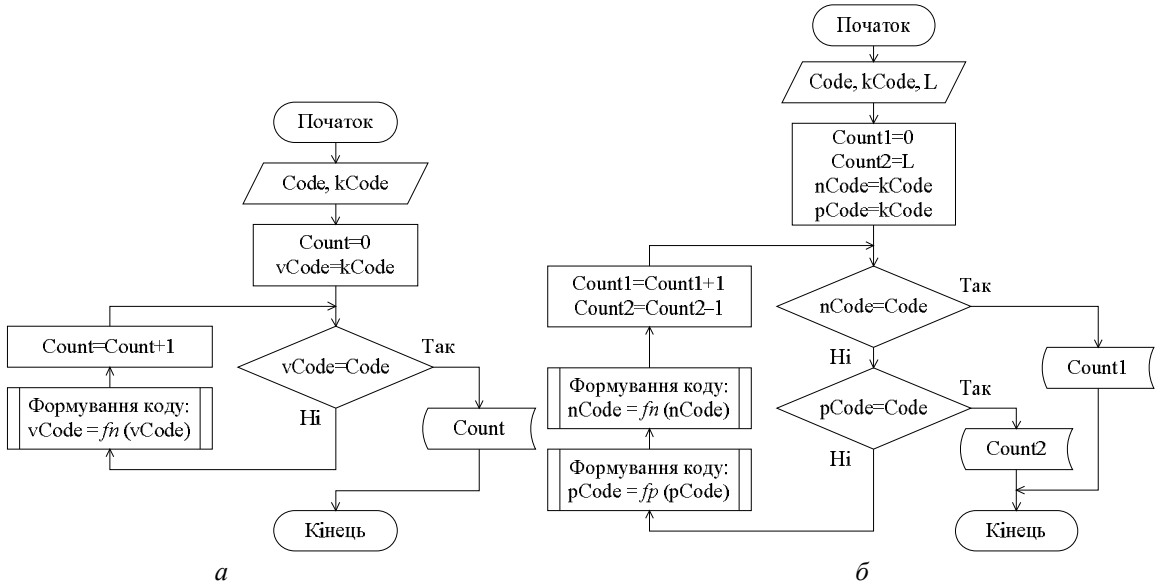


Рис. 1. Блок-схема алгоритму локалізації кодової послідовності методом:
а – одностороннього наближення; б – методом двостороннього наближення

У поданій блок-схемі змінна Code – значення фрагмента, що необхідно декодувати, змінна kCode – базовий (ключовий) фрагмент задіяної циклічної послідовності, змінна Count – лічильник зміщення від базового (нульового) фрагмента, що фактично представляє еквівалентне значення у двійковому базисі.

Функція *fn* – реалізує формування наступного елементу кодової послідовності на основі попереднього vCode і відповідного правила. Проміжна змінна vCode використовується як черговий фрагмент коду для завершення циклу локалізації.

Іншим способом реалізації описаного методу є проведення інкрементного перебору можливих варіантів кодової послідовності, починаючи від заданого і до виявлення збігу з базовим (ключовим) фрагментом. У такому випадку значення локалізованого фрагмента обчислюється як різниця (L – Count), де L – довжина M-послідовності.

Метод двостороннього наближення. Суть методу полягає в одночасному проведенні інкрементного та декрементного перебору можливих варіантів кодової послідовності, починаючи від базового (нульового) і до виявлення збігу із заданим (прийнятим) фрагментом M-послідовності. Блок-схему алгоритму процедури локалізації фрагмента циклічної послідовності Галуа подано на рис. 1, б.

У поданій блок-схемі змінна Code – значення фрагмента, що необхідно декодувати, змінна kCode – базовий (ключовий) фрагмент задіяної циклічної послідовності, змінна L – довжина M-послідовності. Змінні Count1 та Count1 – лічильники зміщення від базового (нульового) фрагмента у бікрону зростання і спадання значень у двійковому базисі відповідно.

Функції *fn* та *fp* – реалізують формування відповідно наступного і попереднього елементів кодової послідовності на основі змінних nCode та pCode і відповідного правила. Проміжні змінні nCode та pCode використовується як чергові фрагменти коду для завершення циклу локалізації.

Аналогічно до попереднього методу, проведення інкрементного та декрементного перебору можливих варіантів кодової послідовності можна реалізувати, починаючи від заданого і до виявлення збігу з базовим (ключовим) фрагментом.

Доцільно зазначити, що швидкість локалізації кодової послідовності залежить від кількості можливих комбінацій (довжини) коду, тактової частоти процесора, кількості процесорів та об'єму статичної пам'яті. Інтенсивний розвиток багатопроцесорних систем дає змогу реалізувати згадану процедуру декодування у вигляді двох паралельних процесів.

Метод фіксованої сітки. Реалізація методу ґрунтується на використанні масиву фіксованих точок, що можуть бути розташовані як з постійним, так і з довільним інтервалом на кодовій послідовності. В результаті утворюється сітка, яка покриває задіяну кодову послідовність. Фактично такий підхід комбінує табличні методи і методи послідовного перебору.

Отже, при локалізації проводять інкрементний перебір можливих варіантів кодової послідовності, починаючи із заданої, до виявлення збігу з однією із кодових комбінацій заданої сітки. Блок-схему алгоритму процедури локалізації необхідного фрагмента циклічної послідовності Галуа подано на рис.2.

У поданій блок-схемі змінна Code – значення фрагмента, що необхідно декодувати, масив mC[N][2] – таблиця з N –стовпців та 2 – рядків, в якій зазначено фрагменти кодової послідовності і відповідні значення двійкового базису. Змінна Count – лічильники зміщення до однієї з точок сітки mC[i][2]. Функція *fn* – реалізує формування наступного елементу кодової послідовності на основі попереднього vCode і відповідного правила. Проміжна змінна vCode використовується як черговий фрагмент коду для завершення циклу локалізації.

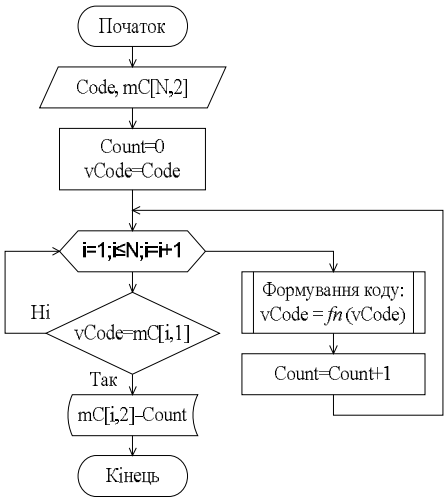


Рис. 2. Блок-схема алгоритму локалізації кодової послідовності методом фіксованої сітки

Доцільно зазначити, що реалізація описаного методу також можлива і з використанням методу двостороннього наближення.

Швидкість локалізації кодової послідовності залежить від кількості можливих комбінацій (довжини) коду, тактової частоти процесора, об'єму статичної та оперативної пам'яті.

Основною перевагою описаних методів декодування є простота реалізації, недоліком – неоптимальне використання обчислювальних ресурсів.

Методи наближення за ключем (ковзного кроку). Методи декодування з ковзним кроком переходу, який при формуванні кожного наступного елементу коду визначає необхідність проведення операцій порівнювання генерованого фрагмента з ключовим.

Адаптивного наближення. Суть методу полягає у проведенні перебору можливих фрагментів кодової послідовності від заданої m_{var} до ключової m_{key} інкрементним або декремнтним зсувом зі змінним кроком k. Величина кроку визначається кількістю незбігів елементів фрагментів m_{var} та m_{key} і довжиною породжуючого ключа h відповідної М-послідовності [5]:

$$k = \sum_{j=1}^{h} (m_{\text{var}})_j \oplus (m_{key})_j .$$

Тобто задана послідовність побітно порівнюється з базовою (ключовою), а результат такого порівняння визначає кількість проміжних генерувань до наступного кодового фрагмента. Якщо ж k=0, тобто фрагменти m_{var} та m_{key} ідентичні, то процедура декодування припиняється. Напрям зміщення – в кінець чи початок – визначає інкрементний чи декрементний правила наближення до базового коду. Також можлива реалізація двостороннього адаптивного наближення, тобто одночасно інкрементного та декрементного зміщення.

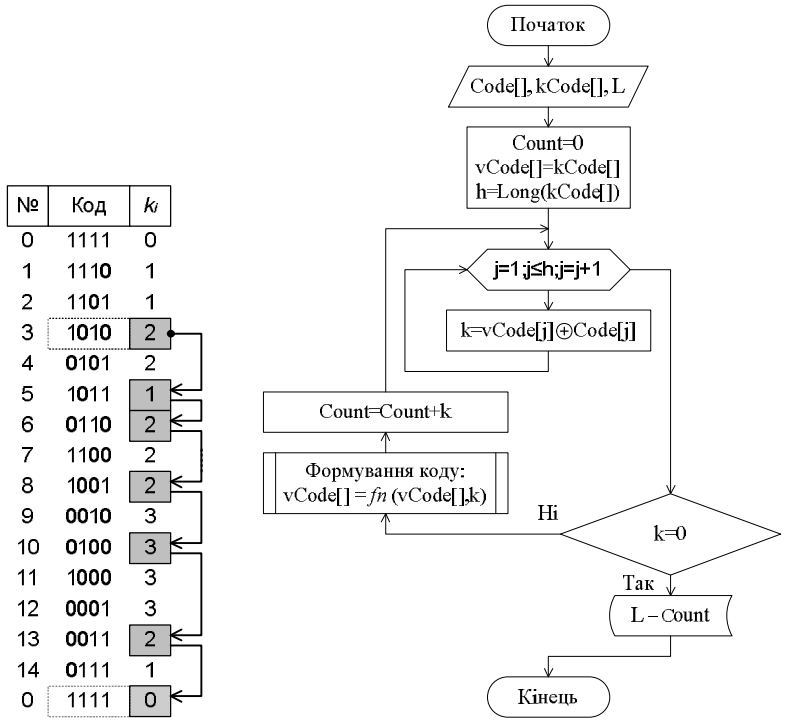
Декодування методом позиційного наближення на прикладі заданого фрагмента коду m_{var} = 1010 для послідовності Галуа з породжуючим ключем m_{key} = 1111, h = 4 біти подано на рис.3а.

Фактично запропонований метод ґрунтується на особливості генерування двійкових циклічних послідовностей Галуа, що не реалізовано у розглянутих вище методах. Блок-схему алгоритму процедури локалізації фрагмента циклічної послідовності Галуа подано на рис.3б.

У поданій блок-схемі змінна Code[] – фрагмент, що необхідно декодувати, масив kCode[] – базовий фрагмент послідовності. Змінна Count – лічильники зміщення, змінна L – довжина М-

послідовності. Функція *fn* реалізує послідовне формування k бітів (елементів) коду без проведення операцій порівнювання з базовою послідовністю.

Описаний метод дає змогу забезпечити більшу швидкість декодування, що зумовлено заміною операцій порівняння на бітові операції між заданим і ключовим фрагментами. Отже, у випадку програмної реалізації, при визначенні кожного наступного фрагмента послідовності необхідно реалізувати без порівнювання з базовою послідовністю. Крім того, описаний метод максимально ефективний у випадку монотонного породжуючого ключа, тобто коли усі його елементи (біти) однакові “0” чи “1”.



а *б*

Рис.3. Графічне представлення реалізації методу адаптивного наближення (а); блок-схема алгоритму локалізації кодової послідовності методом адаптивного наближення (б)

Фіксовано-позиційного наближення. Суть методу полягає у проведенні перебору можливих фрагментів кодової послідовності від заданої m_{var} до ключової m_{key} інкрементним або декремнтним зсувом зі змінним кроком k. Величина кроку визначається останнім бітом поточного $(m_{var})_h$ та ключового $(m_{const})_h$ фрагментів, а також довжиною породжуючого ключа h відповідної послідовності Галуа [6]:

$$k = \begin{cases} 1, & (m_{\text{var}})_h = (m_{key})_h; \\ L, & (m_{\text{var}})_h \neq (m_{key})_h. \end{cases}$$

Тобто на основі порівнювання останнього елемента (біта) заданого (прийнятого) кодового фрагмента послідовності $(m_{var})_h$ з відповідним елементом породжуючого фрагмента $(m_{key})_h$ вибирають крок зміщення k відносно поточного кодового фрагмента. Якщо k=1, то проводиться додаткове побітне порівнювання $(m_{var})_j$ та $(m_{key})_j$, у випадку їх еквівалентності декодування завершується, інакше на основі k генерується новий фрагмент циклічної послідовності Галуа, після чого порівнювання повторюється.

Описана процедура проводиться до повного збігу з породжуючим кодовим фрагментом mkey. Напрям зміщення – в кінець чи початок – визначає інкрементний чи декрементний правила наближення до базового коду.

Декодування методом фіксовано-позиційного наближення на прикладі заданого фрагмента коду m_{var} = 0110 для послідовності Галуа з породжуючим ключем m_{key} = 1111, h = 4 біти подано на рис. 4, а. Блок-схему алгоритму процедури локалізації необхідного фрагмента циклічної послідовності Галуа подано на рис.4б.

У поданій блок-схемі змінна Code[] – фрагмент, що необхідно декодувати, масив kCode[] – базовий фрагмент послідовності. Змінна Count – лічильники зміщення, змінна L – довжина М-послідовності. Функція *fn* реалізує послідовне формування k біт (елементів) коду без проведення операцій порівнювання з базовою послідовністю.

Отже, описаний метод дає змогу забезпечити більшу ефективність за рахунок мінімізації використання операцій порівняння. Тобто порівняння лише одного біту поточної та базової послідовностей зумовлює необхідність проведення додаткового повного порівнювання $(m_{var})_j$ та $(m_{key})_j$, внаслідок чого досягають збільшення швидкості декодування у випадку програмної реалізації. Найчастіше формування поточних бітів реалізовується на основі регістру зсуву, що дає змогу використати отриманий біт як ознаку, що керує блоком повного порівняння відповідних фрагментів заданої послідовності.

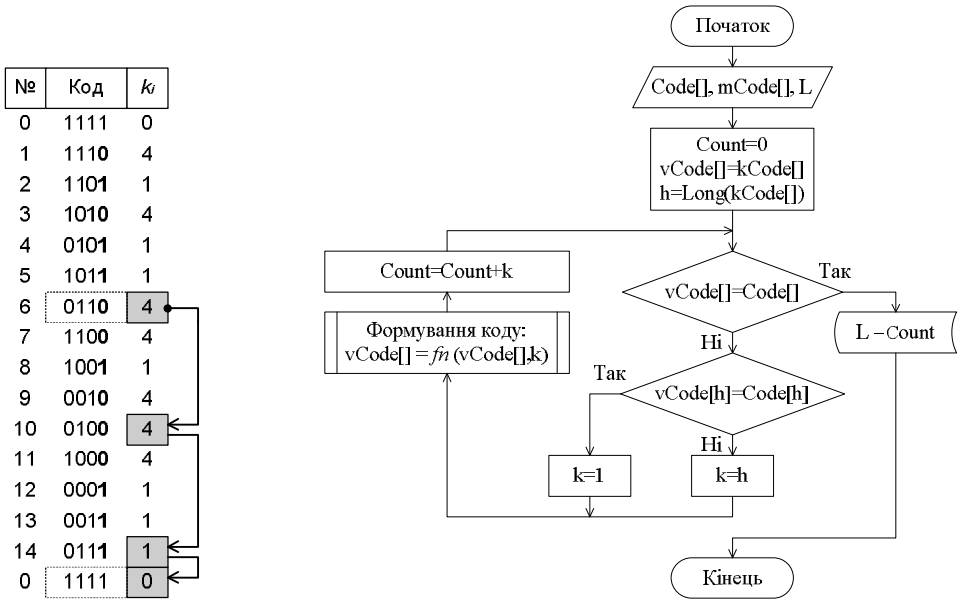


Рис. 4. Графічне представлення реалізації методу фіксовано-позиційного наближення (а); блок-схема алгоритму локалізації кодової послідовності методом фіксовано-позиційного наближення (б)

Змінно-позиційного наближення. Суть методу полягає у проведенні перебору можливих фрагментів кодової послідовності від заданої m_{var} до ключової m_{key} інкрементним або декремнтним зсувом зі змінним кроком k. Величина кроку визначається позицією біта $(m_{var})_j$, що не відповідає $(m_{key})_j$, починаючи від кінця фрагмента, а також довжиною породжуючого ключа h відповідної послідовності Галуа [7]:

$$k = j, (m_{\text{var}})_j \neq (m_{key})_j, j = \overline{h,0}$$

Отже, на основі послідовного порівнювання бітів елементів заданого (прийнятого) кодового фрагмента послідовності $(m_{var})_j$ з відповідним бітом породжуючого фрагмента $(m_{key})_j$ здійснюється обчислення кроку зміщення k відносно поточного кодового фрагмента. У випадку еквівалентності усіх бітів, якщо k=0, декодування завершується, інакше на основі k генерується новий фрагмент циклічної послідовності Галуа, після чого порівнювання повторюється.

Напрям зміщення – в кінець або початок – визначає інкрементне чи декрементне правила наближення до базового коду. Декодування методом змінно-позиційного наближення на прикладі заданого фрагмента коду m_{var} = 0101 для послідовності Галуа з породжуючим ключем m_{key} = 1111, h = 4 біти подано на рис.5а.

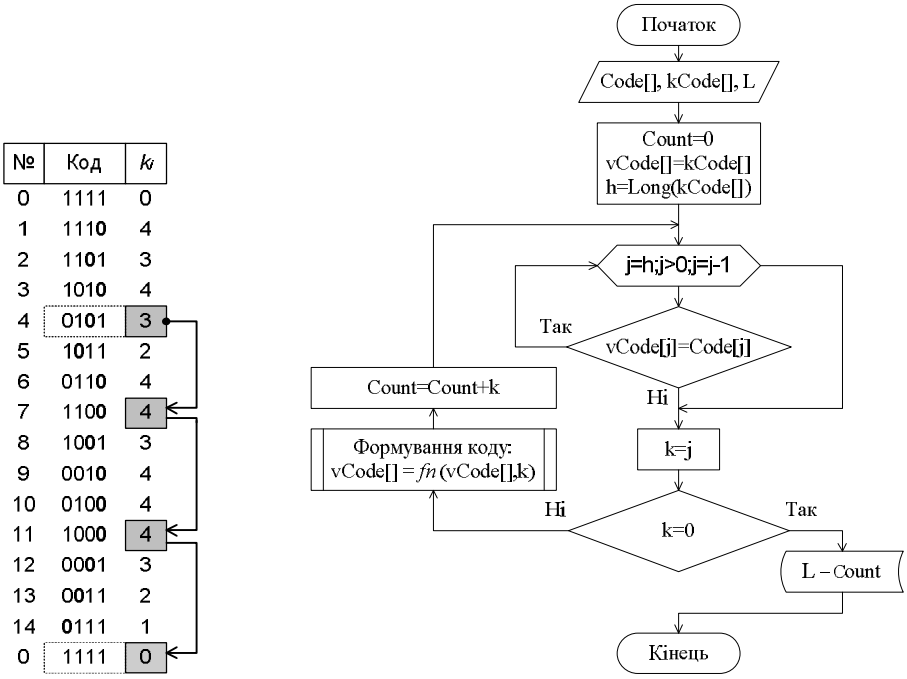
Розглянутий підхід дає змогу значно зменшити кількість операцій порівняння за рахунок ефективного використання елементів кодового фрагмента послідовності Галуа, що декодується.

Блок-схему алгоритму процедури локалізації необхідного фрагмента циклічної послідовності Галуа подано на рис.5б.

Як можна побачити, повна (побітна) перевірка поточного та базового (ключового) фрагментів здійснюється лише один раз – при досягненні ключа, в усіх інших випадках ця операція припиняється раніше, що фактично не реалізовано в жодному з вищезгаданих методів.

У поданій блок-схемі змінна Code[] – фрагмент, що необхідно декодувати, масив kCode[] – базовий (ключовий) фрагмент послідовності. Змінна Count – лічильники зміщення, змінна L – довжина М-послідовності. Функція *fn* реалізує послідовне формування k біт (елементів) коду без проведення операцій порівнювання з базовою послідовністю.

Отже, описаний метод забезпечує більшу ефективність за рахунок мінімізації використання операцій порівняння, внаслідок чого досягають більшою швидкості декодування у випадку програмної реалізації.



а *б*

Рис. 5. Графічне представлення реалізації методу змінно-позиціного наближення (а); блок-схема алгоритму локалізації кодової послідовності методом змінно-позиційного наближення (б)

Висновки. Порівняльні характеристики за кількістю необхідних операцій порівняння *R* при локалізації (декодуванні) відповідних фрагментів циклічної послідовності Галуа завдовжки L=15 бітів розглянутих методів подано на рис.6.

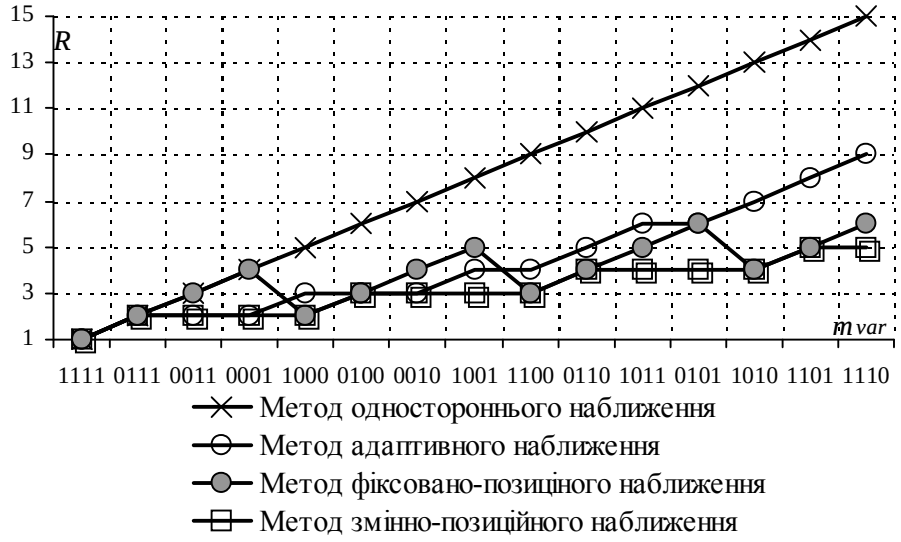


Рис. 6. Залежність кількості операцій порівняння R при локалізації фрагментів послідовності Галуа з породжуючим ключем (1111)

Отже, декодування двійкових циклічних М-послідовностей різних довжин методом адаптивного наближення порівняно з методом одностороннього наближення дає змогу зменшити кількість операцій порівняння на 6…67%. Методи фіксовано-позиціного наближення та змінно-позиційного наближення в аналогічних умовах забезпечують зменшення на 6…60% та 6…40% відповідно. Для апаратної реалізації методів цієї категорії описані вище процедури декодування дещо незручні, оскільки необхідно забезпечити циклічне порівняння кодових елементів. Проте сучасні апаратні засоби, зокрема програмовані логічні інтегральні схеми (ПЛІС), володіють псевдоциклічними операціями, що дає змогу реалізовувати алгоритми подібної структури на їх основі. Перевагою методів наближення за ключем є використання меншої порівняно з методами послідовного наближення кількості операцій порівняння, недоліком – обмеження щодо форми використовуваного ключа.

1. Мельничук С.І., Романів В.М. Метрологічна надійність засобів перетворення тахометричних лічильників газу // Тези доповідей IV наук.-техн. конференції "Приладобудування 2005": Стан і перспективи. – К.: НТУУ "Київський політехнічний інститут", 2005. – 268 с. 2. Николайчук Я.М. Теорія джерел інформації.2-ге вид., випр. і доп. – Тернопіль: ТзОВ "Терно-граф", 2010. – 536 с. 3. Ныколайчук Я.М., Шевчук Б.М. Методы цифровой обработки шумоподобных сигналов на основе кодовых ключей. Технические средства обработки для высокопроизводительных ЭВМ и систем. – К., 1988. 4. Петришин Л.Б. Теоретичні основи перетворення форми та цифрової обробки інформації. – К.: ІЗіМН МОУ, 1997. – 236 с. 5. Мельничук С.І. Методи динамічного декодування одновимірних М-послідовностей // Методи та прилади контролю якості. –1998. – №2. – С.66–68. 6. Мельничук С.І., Таянов С.А. Декодування одновимірних циклічних М-послідовностей методом адаптивного сходження // Автоматизація виробничих процесів у машинобудуванні та приладобудуванні. – 2001. – №36. – С.117–121. 7. Мельничук С.І. Декодування двійкових циклічних М-послідовностей методом збіжності за ключем // Методи та прилади контролю якості. – 2005. – №15. – С.40–42.