УДК: 004.056.3

Є.Я. Ваврук, В.В. Грос
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

МЕТОДИ ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ ВУЗЛІВ ВИМІРЮВАННЯ ШВИДКОСТІ ОБ'ЄКТА

Проведено порівняльний аналіз методів забезпечення відмовостійкості вузлів вимірювання швидкості об'єкта на основі ШПФ. Наведені кількісні показники додаткових затрат на їх реалізацію.

Methods to ensure the fault-tolerance of nodes measure the speed of the object based on FFT comparative analysis are performed. Quantitative indicators of overheads on their implementation are introduced.

Вступ. Задачу вимірювання швидкості об'єкта необхідно розв'язувати у багатьох галузях техніки та наукових досліджень. У більшості випадків вимірювання потрібно проводити в режимі реального часу із забезпеченням гарантоздатної роботи як відповідних систем, так і їх складових частин. Проблема полягає в тому, що такі системи характеризуються складністю алгоритмів, високою швидкодією та великою кількістю каналів передавання інформації. Причому помилки роботи вузлів можуть виявлятися на всіх структурних рівнях: система, процесор, окремий вузол, канал передавання. Хоча результати досліджень у цьому напрямку відображено у багатьох джерелах, наприклад [1, 2], забезпечення відмовостійкості залишається актуальною задачею.

Огляд літературних джерел. Для вимірювання швидкості об'єкта застосовують аерометричний, тепловий, термодинамічний, механічний, локаційний, доплерівський, інерційний та геодезичний методи [3]. Одним із найпоширеніших є метод вимірювання доплерівського зсуву частоти відбитого від рухомої цілі сигналу. Він, крім того, забезпечує достовірність вимірювання в умовах дії різних завад.

Для визначення значення доплерівського зсуву частот прийнятий та опорний сигнали піддають частотному аналізу: знаходять пікові значення частот (виявляють цілі), зсув спектра прийнятого сигналу відносно опорного [4, 5]. Тому необхідно зіставити сигнал, заданий у часовій області, із його еквівалентним поданням у частотній області.

Для визначення спектра сигналу використовують дискретне перетворення Фур'є (ДПФ), яке визначають за формулою:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \text{ , k=0,…,N-1,}$$

де N – число відліків сигналу, $x(n)$ – дискретний сигнал у вигляді часової послідовності $(0 \le n \le N)$, $X(k)$ – спектральна складова сигналу $(0 \le k \le N)$, $W_N = e^{-j(2\pi/N)}$.

Для зменшення часової складності обчислення ДПФ використовують швидкі алгоритми – швидке перетворення Фур'є (ШПФ). Граф 8-точкового ШПФ з прорідженням за часом та граф його базової операції (метелика) наведено на рис. 1, а та рис. 1, б відповідно.

Метелик ШПФ виконує такі обчислення:

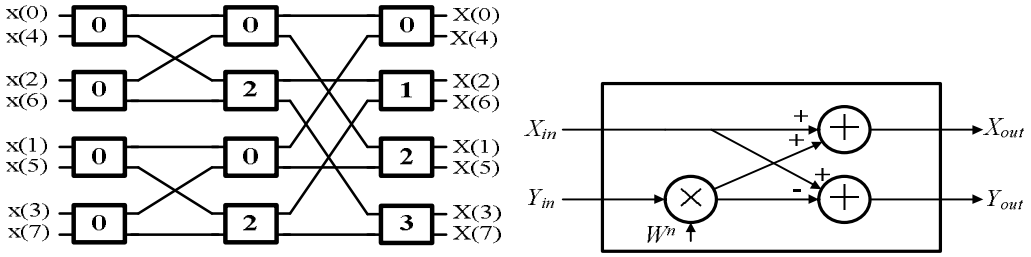
$$X_{out} = X_{in} + Y_{in} W^n; \qquad (1)$$
$$Y_{out} = X_{in} - Y_{in} W^n;$$

У комплексній формі:

$$X_{in} = X_R + jX_I;$$
$$Y_{in} = Y_R + jY_I; \quad (2)$$
$$W^n = W_R + jW_I.$$

Об'єднавши рівняння (1) і (2), отримаємо:

$$X_{out} = (X_R + W_R Y_R - W_I Y_I) + j(X_I + W_I Y_R + W_R Y_I)$$
$$Y_{out} = (X_R - W_R Y_R + W_I Y_I) + j(X_I - W_I Y_R - W_R Y_I) \quad (3)$$



а б

Рис. 1. Граф 8-точкового ШПФ з проріджениям за часом (а) та граф базової операції (метелика) ШПФ з проріджениям за часом (б)

На рис. 2 наведено граф алгоритму 8-точкового ШПФ з досконалим тасуванням, де стрілками показано напрям заведення даних для виконання перетворення.

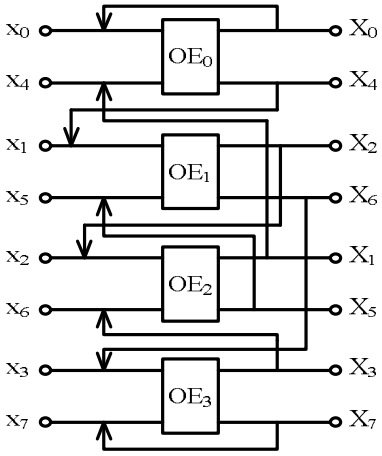


Рис. 2. Граф алгоритму 8-точкового ШПФ з досконалим тасуванням

Згідно з алгоритмом ШПФ з досконалим тасуванням вхідні дані перетворюються шляхом повторення операцій множення та додавання log_2N разів на $N/2$ обчислювальних елементах ОЕ (метеликах).

Відмовостійкість роботи вузлів вимірювання швидкості забезпечується на кількох рівнях [6–11]: бітовому, модульному, процесорному.

Постановка задачі. Метою досліджень є вибір оптимальних методів забезпечення відмовостійкості вузлів вимірювання швидкості об'єкта на базі ШПФ та визначення затрат на їх реалізацію.

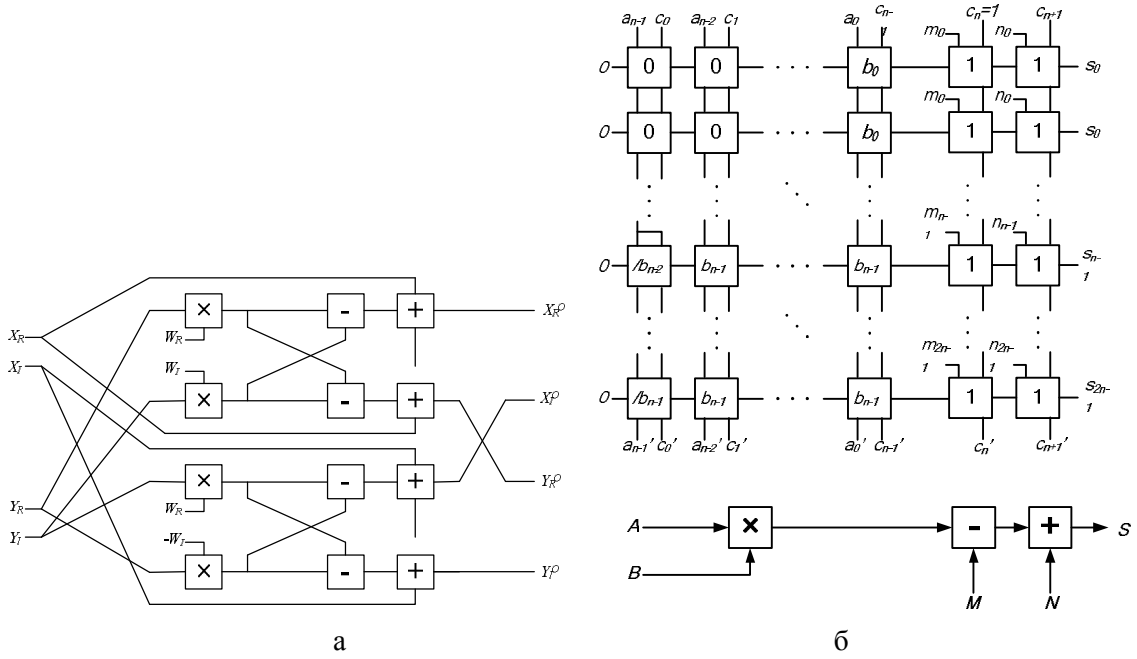
Методи забезпечення відмовостійкості ШПФ. Виявлення відмов, що виникають впродовж обчислення ДПФ, і подальша реорганізація процесу опрацювання вимагають додаткових ресурсів:

алгоритмів обходу помилки, додаткових метеликів, надлишкового часу виконання та інших. Який саме тип ресурсів необхідний, визначають за методом, що використовується для виявлення/корекції.

Забезпечення відмовостійкості на "бітовому" рівні:

Розглянемо випадок, коли відмова локалізується на "бітовому рівні" одиничного елемента (метелика) [6–9], тобто на рівні порозрядної обробки даних.

Обчислення метелика ШПФ згідно з (3) може бути реалізоване на 4-х ідентичних MSA (multiply-substract-add – множення–віднімання–додавання) модулях, як показано на рис. 3, а. На рис. 3, б наведено схему (матрицю) реалізації ітеративної процедури порозрядної обробки даних в MSA модулі, де n – розрядність операнда; $a_{n-1}…a_0$, $b_{n-1}…b_0$, $m_{2n-1}…m_0$, $n_{n-1}…n_0$ – послідовності бітів вхідних операндів A, B, M та N, відповідно; $c_{n-1}…c_0$ – біти перенесення; $s_{n-1}…s_0$ – біти результату. На перших n елементах матриці логічних елементів (МЛЕ) виконується перемноження вхідних операндів, а саме – виконання багатомісної операції додавання часткових добутків. Кожний з n перших стовпців МЛЕ обчислює часткові добутки, розряди яких, з однаковими вагами, додаються по рядках з врахуванням перенесень з молодших розрядів. У результаті одержуємо $2n$ розрядів добутку, $n+1$-й та $n+2$-й стовпці МЛЕ виконують відповідно операцію віднімання та додавання одержаного на попередніх стовпцях МЛЕ результату з вхідними операндами M та N, відповідно.



а б

Рис. 3. Структурна схема метелика ШПФ на MSA модулях (а), структурна схема MSA модуля у вигляді матриці логічних елементів (б)

Відмовостійкість метелика ШПФ може забезпечуватися внесенням в структуру МЛЕ надлишковості як на рівні стовпця, так і на рівні рядка.

Забезпечення відмовостійкості MSA модуля з використанням резервного стовпця. У межах кожного стовпця МЛЕ всі елементи виконують однакову функцію. Структурну схему MSA модуля з резервним стовпцем наведено на рис. 4. У цій схемі резервні елементи розміщуються між елементами множення та віднімання і тому вони повинні бути модифіковані, щоб бути спроможними виконувати функції сусідніх елементів. Отже, у структурній схемі MSA модуля містяться такі елементи: комірка перемноження (MC), модифікована комірка множення-віднімання (MS), модифікована комірка віднімання-додавання (SA), комірка додавання (AC). Необхідні функції модифікованих елементів резерного стовпця після реконфігурації визначаються сигналом керування. Детальніше про структурні схеми модифікованих комірок та можливості їх реорганізації наведено у [9, 10].

Реконфігурація: якщо елемент *k-го* стовпця відмовив, *k-й* стовпець заміняється *(k+1)-м*, який, своєю чергу, *(k+2)-м* і т.д. до *n-го*. Комірки стовпців *(n+1)* та *(n+2)* залишаються без змін. Якщо ж відмовив елемент *(n+1)*-го або *(n+2)*-го стовпця, процедура заміни проводиться аналогічно, але у протилежному напрямку.

Забезпечення відмовостійкості MSA модуля з використанням резервного рядка. Структурну схема MSA модуля з резервним рядком наведено на рис. 5а. Така схема не вимагає модифікації складових компонентів MSA модуля. Якщо *a-й* рядок відмовив, він заміняється *(a-1)-м*, який, своєю чергою, *(a-2)-м* і т.д. У результаті перший рядок заміняється резервним рядком. Схему реконфігурації модифікованого MSA модуля з надлишковим рядком наведено на рис. 5б.

Після реконфігурації MSA модуля всі зв'язки з елементами рядка (стовпця), який відмовив, повинні бути перенапрямлені.

Але перед заміною рядка (стовпця) потрібно знайти елемент, який вийшов з ладу. Виявлення та локалізацію несправностей забезпечують виконанням набору з 18 тестових комбінацій вхідних даних [9] незалежно від розмірності ШПФ на кожному MSA модулі у тестовому режимі ШПФ процесора. Після виявлення несправності тестування продовжується для локалізації рядка (стовпця) з елементом, що відмовив. Це забезпечується послідовною заміною кожного рядка (стовпця), починаючи з першого, резервним і виконанням набору тестових комбінацій після кожного кроку модифікації MSA модуля. Процес продовжується до виявлення рядка (стовпця) з несправним елементом. У найгіршому випадку таку процедуру необхідно повторити *2b* разів (*b* – розрядність вхідних даних) для MSA з надлишковим рядком і *b* разів для MSA з надлишковим стовпцем.

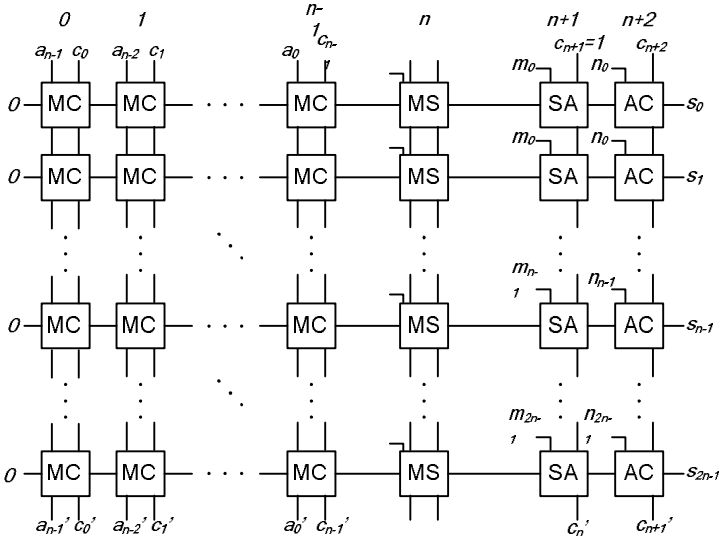
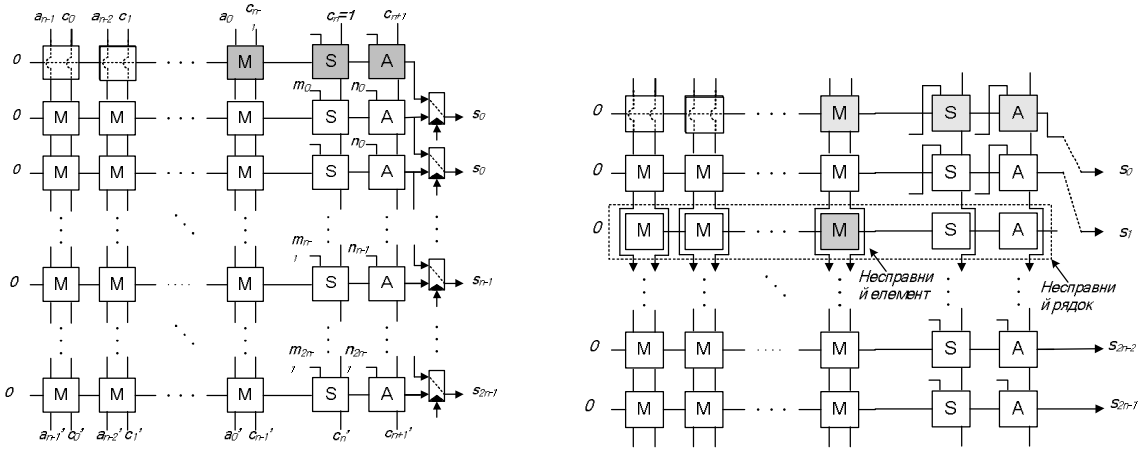


Рис. 4. Структурна схема MSA модуля з надлишковим стовпцем



а б

Рис. 5. Структурна схема MSA модуля з надлишковим рядком (а), схема реконфігурації MSA модуля з надлишковим рядком (б)

Забезпечення відмовостійкості метелика ШПФ на модульному рівні [9, 10]

Розглянемо 4 MSA модулі, наведені на рис. 6а, розділені на дві групи по два модулі, що обчислюють реальну та уявну частини результату. Для забезпечення відмовостійкості до структурної схеми MSA модуля вводять додатковий резервний модуль (MSA_0), який розміщується поряд з іншими MSA модулями (див. рис. 6а).

Якщо виявлено несправний модуль, активується процес реконфігурації, і такий модуль заміняють резервним. Припустимо, що модуль MSA_i відмовив, тоді, згідно з алгоритмом реконфігурації, він заміняється $MSA_{i-1м}$, який заміняється $MSA_{i-2м}$ і так до 1-го резервного модуля. На рис. 6б наведено випадок, коли відмовив модуль MSA_2.

Несправності виявляються при виконанні тестового режиму роботи ШПФ, аналогічно методу забезпечення відмовостійкості на бітовому рівні. Але у цьому випадку необхідно виконати у найгіршому випадку 4 операції заміни модулів і виконання тестових комбінацій.

Модульна надлишковість з розподіленим часом виконання

Традиційна модульна надлишковість може використовуватись для виявлення і корекції будь-якого роду помилок, але вона характеризується високою складністю (рівнем надлишковості). Подібним чином, традиційні рішення з використанням часової надлишковості характеризуються значним збільшенням затримки і зменшенням пропускної здатності.

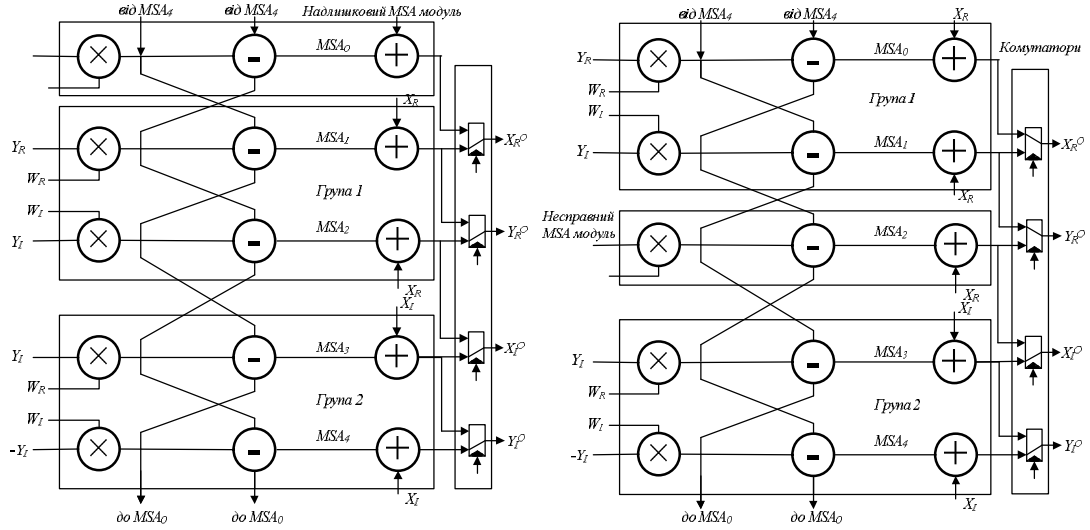


Рис. 6. Відмовостійка структурна схема метелика ШПФ (а); реконфігурація MSA модуля (б)

Основна концепція часового резервування – виконання обчислень кілька разів з метою виявлення помилок і, можливо, для їх усунення [11]. Повторне обчислення на одних і тих самих функціональних модулях і значеннях вхідних даних може виявити лише нестійкі відмови. Постійні відмови можна виявити шляхом зміни функціональних модулів або операндів.

Модульна надлишковість з розподіленим часом виконання [11] передбачає, що арифметичні операції, такі як додавання і множення, діляться на частини, розкладаючи операнди на групи сусідніх розрядів. Кожна частина операндів обробляється відповідною частковою операцією.

Для виявлення помилки достатньо порівняти два результати. Але для забезпечення відмовостійкості необхідно корегувати результати. Для цього необхідно виконати три ітерації номінальної операції на різних арифметичних модулях та сформувати три копії перетворених даних *X(k)* з подальшою верифікацією їх достовірності за допомогою мажоритарної схеми голосування. Корекція проміжних результатів за цим методом не потрібна, оскільки за цим підходом можна виправляти можливі помилки опрацюванням лише остаточних результатів перетворення.

За цим методом забезпечення відмовостійкості кожний *B*-бітний вхідний операнд *x(n)* ШПФ процесора ділиться на три частини по *éB/3ù* біт кожна. Розрядність повертаючих множників $W_N=e^{-}$

$^{j(2\pi/N)}$ залишається без змін. Над кожною частиною вхідних операндів виконуються паралельно три незалежні ітерації алгоритму ШПФ; перенесення передаються на останній етап опрацювання – об'єднання часткових результатів. У результаті формуються три копії перетворених даних ***X(k)***, до яких застосовується мажоритарна схема голосування.

Розглянемо структурну схему ШПФ процесора з одночасним виправлення помилок, яку наведено на рис. 7. Основними структурними одиницями цієї схеми є вхідні блоки розподілу, каскад модифікованих функціональних блоків (МФБ) та вихідні блоки об'єднання та порівняння.

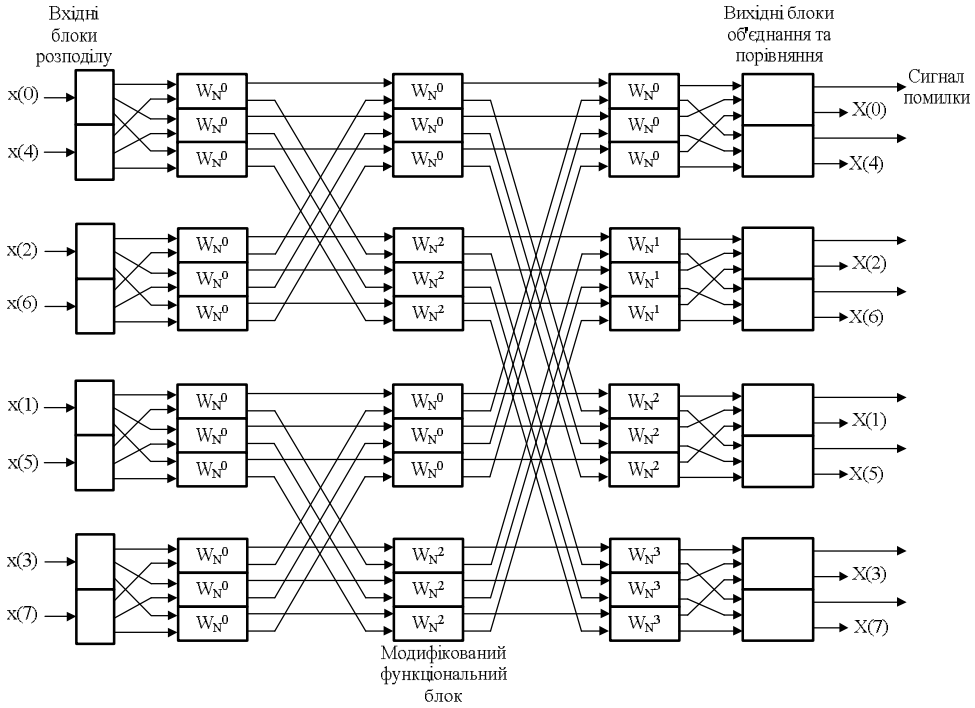


Рис. 7. Модульна архітектура ШПФ процесора з одночасним виявленням помилки на основі модульної надлишковості з розподіленим часом виконання

Вхідні блоки розподілу (див. рис. 8) для кожного входу відмовостійкого ШПФ процесора використовують три мультиплексори 3 в 1 для вибору необхідної частини операнда. В такому випадку відмова одного з мультиплексорів ніяк не вплине на роботу МФБ, приєднаних до інших мультиплексорів.

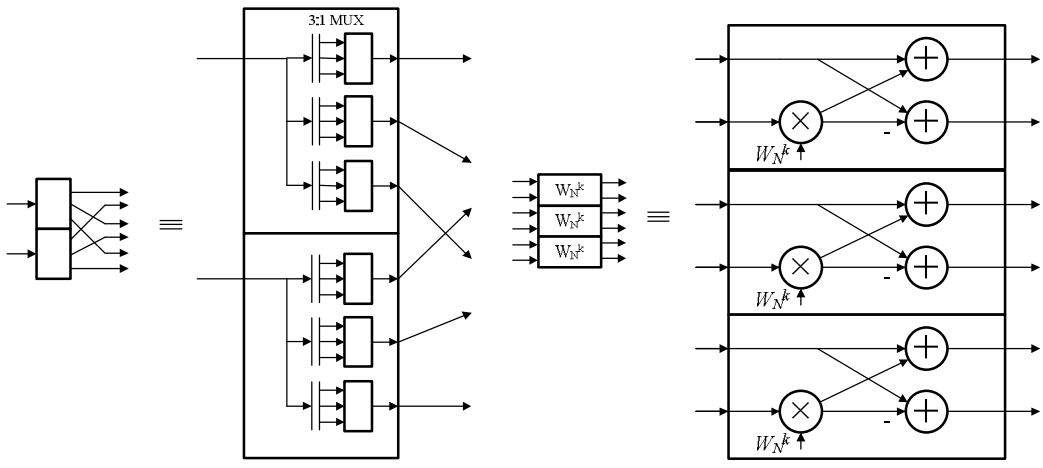


Рис. 8. Вхідні блоки розподілу (а) та модифіковані функціональні блоки (б)

Впродовж першої ітерації операції виконуються на МФБ згідно з алгоритмом ШПФ над молодшими бітами вхідних операндів та повертаючим множником повного розміру. На останньому етапі перетворення ШПФ процесора отримуємо молодші розряди результату, які зберігаються в регістрі. Наступні дві ітерації виконуються аналогічно першій. Далі у вихідних блоках об'єднання та порівняння (див. рис. 9) для завершення обчислення результату *X(k)* необхідно просумувати часткові результати, отримані після розподіленого в часі виконання трьох ітерацій. Для цього з відповідних регістрів зчитуються часткові результати, зсуваються ліворуч на необхідну кількість розрядів і додаються. Після цього три копії остаточного результату подаються на схему голосування.

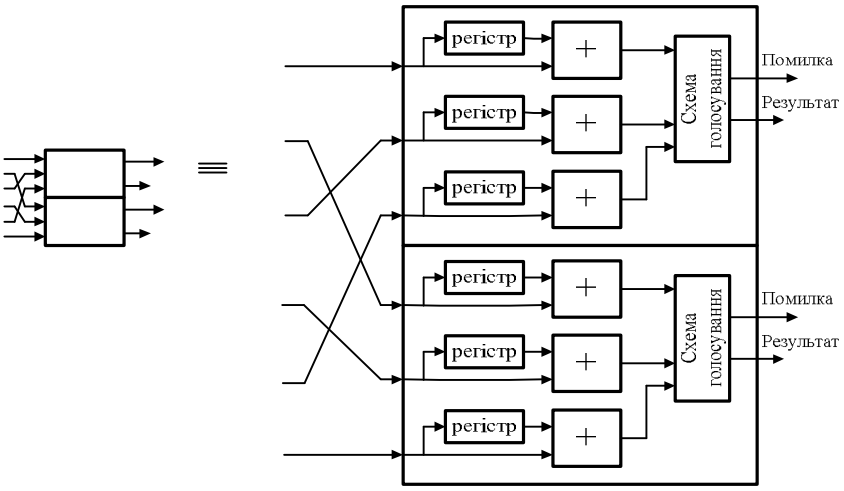


Рис. 9. Вихідні блоки об'єднання і порівняння

Для підвищення продуктивності розглянутого методу забезпечення відмовостійкості можна опрацьовувати дані у конвеєрному режимі. Але для цього необхідно використати додаткові конвеєрні регістри на кожному вході метеликів ШПФ і у вихідному блоці об'єднання та порівняння.

Цей метод забезпечення відмовостійкості є компромісним стосовно підвищення складності схеми і зменшення продуктивності [11].

Відмовостійкий ШПФ процесор на базі моделі лінійного клітинного автомату

Лінійний клітинний автомат (ЛКА) можна використовувати як модель для побудови відмовостійкого ШПФ процесора з досконалим тасуванням [12]. Функція переходу клітинного автомата забезпечує можливість швидкого реконфігурування.

Лінійний клітинний автомат, що має n елементів, описується так:

$$M = (Z_n, Q, W),$$

де $Z_n = \{0,1,\ldots n-1\}$, $Q=GF(q)$, $q=p^t$, p – просте число, t – натуральне число, W: $S_s \rightarrow S_s$, $S_s = \{(s_0,s_1,\ldots,s_{n-1})\} \mid s_i \in GF(q)$, $0 \le i \le n$.

Z_n визначає позицію елемента, Q – множина станів елемента (вершини), W – вагова функція, що визначає зміну стану клітинного автомату, вектор $S=(s_0,s_1,\ldots,s_{n-1})$ визначає стани елементів, S_s – множину векторів станів. Клітинний автомат з ваговою функцією, що має такі властивості, називається лінійним клітинним автоматом, у якому S – поточний стан автомату, S' – майбутній стан автомату.

$$s'_z = \sum_{i+j=z} w_i \cdot s_j, \; 0 \le z \le n-1,$$

де $S=(s_0,s_1,\ldots,s_{n-1})$, $S'=(s'_0,s'_1,\ldots,s'_{n-1})$, $W=(w_0,w_1,\ldots,w_{n-1})$, $w_i \in GF(q)$, $0 \le i \le n-1$.

ЛКА повинен мати незмінне значення коду Хеммінга і постійну кодову відстань для множини векторів станів. Розглянемо приклад ЛКА з постійним значенням Хемінга і постійною кодовою відстанню, що описується так:

$$M = (Z_7, GF(2), W);$$

$$Z_7 = \{0,1,...,6\};$$

$$W = x^4 + x^3 + x^2 + 1.$$

У табл. 1 наведено множину векторів станів *S(i)*, де $0 \le i \le 6$, $s_t(0 \le i \le 6)$ стан для елемента *t*.

Таблиця 1

	s_0	s_1	s_2	s_3	s_4	s_5	s_6
$S^{(0)}$:	1	0	1	1	1	0	0
$S^{(1)}$:	1	1	0	0	1	0	1
$S^{(2)}$:	0	1	0	1	1	1	0
$S^{(3)}$:	1	1	1	0	0	1	0
$S^{(4)}$:	0	0	1	0	1	1	1
$S^{(5)}$:	0	1	1	1	0	0	1
$S^{(6)}$:	1	0	0	1	0	1	1

Відмовостійкий ШПФ процесор (FP) опишемо так:

$$FP = (P, QP, D, I, R)$$

де $P=\{p_i \mid 0 \le i \le n-1\}$, Q_p – множина станів для метелика ШПФ, D: Sps→ Sps, Sps = $\{(Sp_0, Sp_1,\ldots,Sp_{n-1}) \mid Sp_i \in Q_p, 0 \le i \le n-1\}$, $I \subset Q_p$, R:IDO → IDI.

P – множина метеликів, D – функція переходів станів для процесора, Sps – множина векторів станів для метеликів, Sp – вектор станів процесора, I – множина станів для недіючих (резервних) метеликів, R – відображення, що визначає сполучення між виходами і входами метеликів.

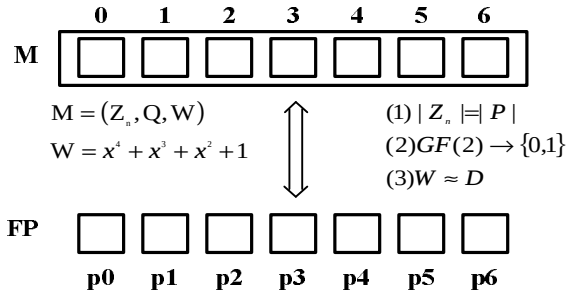
Відмовостійкий ШПФ процесор може бути реалізований на основі моделі ЛКА при забезпеченні таких умов:

1. Кількість метеликів дорівнює кількості елементів ЛКА.
2. Існує бієктивне відображення з множини Q_p в множину Q.
3. Функція переходів станів D ізоморфна до вагової функції W.

На рис. 10 наведено структурні схеми ЛКА(M) і процесора ШПФ(FP), що задовольняють вищенаведені умови.

Відмовостійкий ШПФ процесор у складі метелика зі схемою виявлення несправності, схеми комутації та схеми керування задовольняє вищенаведені умови (див. рис. 11).

Схема комутації забезпечує сполучення входів та виходів відповідних метеликів згідно з алгоритмом досконалого тасування. Схема керування продукує перехідні стани, визначає сполучення виходів та входів відповідних метеликів.



$FP = (P, \{0,1\}, D, I, R)$

$P = \{p0, p1, p2, p3, p4, p5, p6\}$

$D : (1011100) \to (1001011)$

...

$(1100101) \to (1011100)$

Рис. 10. Структурні схеми відмовостійкого ШПФ процесора і ЛКА

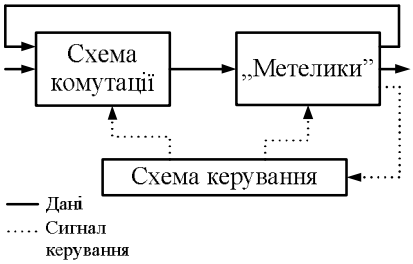


Рис. 11. Блок-схема відмовостійкого ШПФ процесора

Схема виявлення несправності. Кожен метелик опрацьовує дані двічі. Перший раз згідно з рівністю (1). Потім ті самі дані кодуються і знову опрацьовуються. Результати роботи суматора та помножувача, отримані на другому циклі опрацювання, декодуються і порівнюються з результатами першого циклу. Ознака несправності формується за різних значень результатів. У [12] запропоновано простий алгоритм кодування та декодування, що не вимагає значних додаткових затрат на реалізацію. Для кодування у цьому алгоритмі пропонується використати регістр зсуву для декодування: регістр зсуву, вентиль І, суматор та схему віднімання. Ці додаткові компоненти необхідні для кожного метелика ШПФ.

Якщо несправність локалізується в метелику, її можна виявити повторним опрацюванням, як описано вище. Відмову в множині ліній сполучень можна розглядати як відмову на входах або виходах метелика.

На рис. 12 наведено метелик із схемою виявлення несправності.

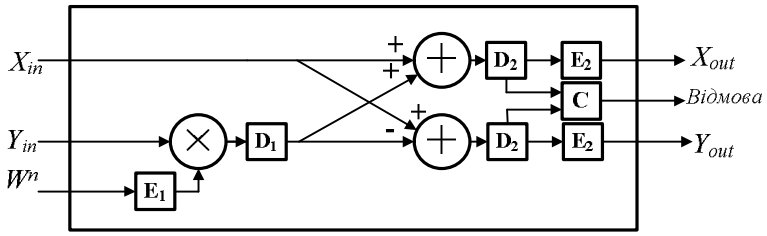
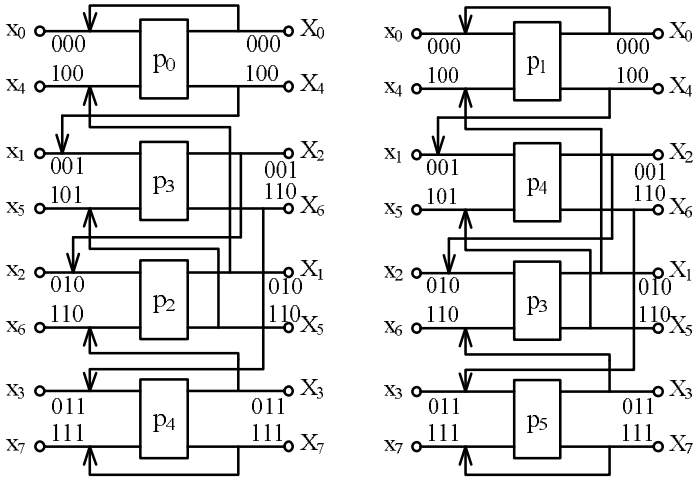


Рис. 12. Структурна схема метелика ШПФ
із схемою виявлення несправності: C – компаратор; D_1, D_2 – декодери; E_1 E_2 – кодери

На рис. 12 “E_1” і “E_2” кодують, відповідно, повертаючий множник та входи метелика , “D_1” та “D_2” декодують відповідно виходи помножувача та суматора.



а б

Рис 13. Структурна схема відмовостійкого процесора ШПФ на моделі ЛКА (а);
реконфігурація відмовостійкого процесора ШПФ на моделі ЛКА (б)

Схема реконфігурації. Якщо виявлено несправний метелик (один або декілька), здійснюють процес реконфігурації. Для цього визначають придатний для заміни вектор станів метеликів, який містить справні елементи, що залишилися, і один чи декілька (залежно від кількості несправних елементів) метеликів із множини резервних елементів *I*; перекомутовуються зв'язки виходів і входів метеликів; стани метеликів, які відмовили, встановлюються в 0.

Приклад процесу реконфігурації відмовостійкого ШПФ процесора наведено на рис. 13, а і 13, б. У цьому прикладі припускається, що елементи p_0, p_2 – несправні.

Розглянутий метод забезпечення відмовостійкості ШПФ дозволяє виявляти і реконфігурувати більше однієї помилки (максимальне число можливих відмов дорівнює кількості резервних елементів – n-$N/2$).

Визначення додаткових структурних та часових затрат методів забезпечення відмовостійкості ШПФ. Оскільки для систем опрацювання сигналів найважливішими є часові та структурні додаткові затрати, доцільно порівнювати вищенаведені структурні схеми та методи саме за цими параметрами. Результати аналізу цих методів наведено у табл. 2, де b – розрядність вхідних даних, n – кількість метеликів, N – кількість відліків сигналу.

Проаналізуємо методи забезпечення відмовостійкості на бітовому рівні. Вони забезпечується двома режимами роботи ШПФ процесора, а саме: тестовим (для виявлення несправності та за необхідності реконфігурації елементів) та робочим (для опрацювання вхідних даних). З табл. 2 видно, що на етапі тестування (у разі виявлення відмови) метод забезпечення відмовостійкості із використанням надлишкових стовпців вимагає удвічі менших додаткових часових затрат порівняно з методом із використанням резервних рядків, але він має складнішу структуру, оскільки необхідно модифікувати елементи окремих його стовпців. Вибір одного із цих двох шляхів забезпечення відмовостійкості у цьому випадку залежатиме від необхідних часових обмежень на тестування та складності реалізації модифікованих арифметичних блоків на конкретній елементній базі.

Також для забезпечення відмовостійкості цього методу необхідна схема керування режимами роботи ШПФ процесора: робочим і тестовим.

Так, у випадку використання 16-розрядних вхідних даних додаткові апаратні затрати для реалізації методу забезпечення відмовостійкості на бітовому рівні з використанням резервного стовпця становитимуть приблизно 16%, а з резервним рядком – 4%. Але для тестування метеликів ШПФ необхідно виконати на MSA з резервним стовпцем від 18-ти до 18*16 тестових комбінацій, а для MSA з резервним рядком – 18–36*16 тестових комбінацій. У загальному випадку систематичні додаткові часові затрати розглянутих методів залежатимуть від частоти виконання процедури тестування.

Компромісним рішенням може бути використання методу забезпечення відмовостійкості метелика ШПФ на модульному рівні, який вимагає лише 25% додаткових апаратних витрат на реалізацію і виконання від 18 до 4*16 тестових комбінацій на виявлення і реконфігурацію.

Отже, перевагами проаналізованих вище методів є незначні додаткові апаратні затрати на реалізацію, а недоліком – висока часова складність виявлення, локалізації та реконфігурації несправності. Тому цей метод є непридатним для використання в системах реального часу. Проте його можна використовувати для реконфігурації несправних модулів ШПФ (метеликів), виявлених іншим методом виявлення/локалізації на вищому рівні забезпечення відмовостійкості.

Проаналізуємо складність схеми модульної надлишковості з розподіленим часом виконання порівняно зі складністю невідмовостійкого ШФП процесора залежно від розрядності вхідних даних і розмірності перетворення. Вона зростає в середньому на 10–20% для схеми виявлення помилки і 50–75% для схеми корекції результатів.

Часова затримка опрацювання даних залежно від розрядності вхідних даних і розмірності перетворення для виявлення несправності зростає в середньому на 70% для схеми без конвеєра і на 100% для схеми з конвеєрним опрацюванням; для корекції результатів – на 160% і 200% для послідовного та конвеєрного опрацювання відповідно.

Відмовостійкий ШПФ процесор на базі моделі ЛКА для виявлення несправності вимагає використання у кожному метелику регістра зсуву для кодування; регістра зсуву, логічного вентиля I, суматора і схеми віднімання для декодування та додаткового циклу опрацювання вхідних даних з подальшим порівнянням результатів обох циклів.

Для забезпечення можливості реконфігурації у відмовостійкому ШПФ процесорі необхідно використати схему комутації, схему керування та схеми метеликів з можливістю виявлення помилки.

Апаратна складність схеми комутації за цим методом забезпечення відмовостійкості ШПФ становить $O(N \log_2 N)$ у разі використання багатоярусної мережі комутації.

З табл. 2 видно, що найефективнішими з погляду необхідних додаткових структурних затрат є методи забезпечення відмовостійкості метелика ШПФ на бітовому та модульному рівнях. Проте вони непридатні для використання у системах реального часу, оскільки мають високі систематичні часові затримки. Компромісною відносно часових та структурних додаткових затрат є структурна схема ШПФ процесора з модульною надлишковістю з розподіленим часом виконання, яка також має порівняно високу пропускну здатність. Перспективним є використання ідеї побудови ШПФ процесора, що ґрунтується на моделі ЛКА, оскільки він забезпечує можливість швидкої реконфігурації системи опрацювання, має невисокі додаткові часові затрати на виявлення-локалізацію несправності та здатний забезпечувати необхідний рівень відмовостійкості при виникненні більше ніж однієї помилки, який регулюється кількістю резервних елементів (метеликів).

Напрямком для подальших досліджень є розроблення та забезпечення відмовостійкості схем комутації і керування та організація системи на конкретній елементній базі.

Висновки. У статті розглянуто існуючі методи вимірювання швидкості об’єкта та вибрано метод вимірювання доплерівського зсуву частоти відбитого від рухомої цілі сигналу, у якому необхідно аналізувати частотний спектр сигналу, для чого використовують ШПФ. Проведено порівняльний аналіз методів забезпечення відмовостійкості ШПФ. Критерієм порівняння обрано необхідні для реалізації методів додаткові структурні та часові затрати. Найефективнішою з погляду необхідних додаткових структурних затрат є схема забезпечення відмовостійкості метелика ШПФ на бітовому рівні, а за додатковими часовими затратами – ШПФ процесор на базі моделі лінійного клітинного автомату.

Таблиця 2

Структурні та часові додаткові затрати методів забезпечення відмовостійкості

Схема забезпечення відмовостійкості		Структурні додаткові затрати	Часові додаткові затрати	
			Систематичні	При виникненні відмови
На основі моделі лінійного клітинного автомату	Виявлення	Для кожного метелика: зсувний регістр для кодування; зсувний регістр, логічний вентиль I, суматор і схема віднімання для декодування	Додатковий цикл виконання базового алгоритму (100%)	–
	Реконфігурація	Схема комутації ($O(N \log_2 N)$), схема керування, n-N/2 метеликів	Неістотні	Неістотні

<table>
<tr><td colspan="3" rowspan="2">Метелика ШПФ на бітовому рівні</td><td>Реконф. по рядку</td><td colspan="2">$\frac{56b^2+460b+4}{312b^2+976b+4}$ транзисторів (для b=16 ≈ 16%)</td><td colspan="2" rowspan="2">$\frac{18}{\text{Період тестування}}\cdot 100\%$</td><td>до 36 b тестових комбінацій</td></tr>
<tr><td>Реконф. по стовпцю</td><td colspan="2">$\frac{8b^2+28b+398}{256b^2+516b}$ транзисторів (для b=16 ≈ 4%)</td><td>до 18 b тестових комбінацій</td></tr>
<tr><td colspan="3">Метелика ШПФ на модульному рівні</td><td colspan="2">Один додатковий MSA модуль на кожний метелик (25%) (N/2 MSA)</td><td colspan="2">$\frac{18}{\text{Період тестування}}\cdot 100\%$</td><td>до 72 тестових комбінацій</td></tr>
<tr><td rowspan="4">Модульна надлишковість з розподіленим часом виконання</td><td rowspan="2">Виявлення</td><td>без конвеєра</td><td rowspan="2">10-20 %</td><td>N/2 $\log_2$N МФБ розрядністю b/2 біт, 2N (регістрів зсуву, суматорів, мультиплексорів)</td><td>70%</td><td rowspan="4">Послідовне виконання часткових операцій, упорядкування результатів і синхронізація виконання опрацювання, виконання порівняння/ голосування</td><td rowspan="4">Нехтуються</td></tr>
<tr><td>з конвеєром</td><td>N/2 $\log_2$N МФБ розрядністю b/2 біт, 2N(регістрів зсуву, суматорів, мультиплексорів) 2N $\log_2$N конвеєрних регістрів</td><td>100%</td></tr>
<tr><td rowspan="2">Корекція</td><td>без конвеєра</td><td rowspan="2">50-70 %</td><td>N $\log_2$N МФБ розрядністю b/3 біт, 3N (регістрів зсуву, суматорів, мультиплексорів)</td><td>160%</td></tr>
<tr><td>з конвеєром</td><td>N $\log_2$N МФБ розрядністю b/3 біт, 3N (регістрів зсуву, суматорів, мультиплексорів) 3N($\log_2$N+1) конвеєрних регістрів</td><td>200%</td></tr>
</table>

1. Харченко В.С. Гарантоспособность и гарантоспособные системы: элементы методологии / Харченко В.С. //Радіоелектронні і компютерні системи. – 2006. – №5. – С. 7–19 2. Ваврук Є.Я. Організація відмовостійкості в системах опрацювання сигналів // Вісник Нац. ун-ту "Львівська політехніка" "Комп'ютерні науки та інформаційні технології". – 2006. – №565. – С.36–43. 3. Козлов П., Осипова М. Измеритель начальной скорости объекта // Електроніка та системи управління. – 2010. – №2(24). – С.12–17. 4. Amit Aggarwal, Erlend Hansen RADAR: Velocity Analysis // ECE 301 Projects Fall – 2003. – P.121–125. 5. Бакулев П.А. Радиолокационные системы. – М.: Радиотехника, 2004. – 320 с. 6. Jin-Fu Li, Cheng-Wen Wu Testable and fault tolerant design for FFT networks // International Symposium Defect and Fault Tolerance in VLSI Systems, 1999. - P.201-209. 7.Shyue-Kung Lu, Cheng-Wen Wu, Sy-Yen Kuo On fault-tolerant FFT butterfly network design // IEEE International Symposium 'Connecting the World', 1996. –P.69-72. 8. Shyue-Kung Lu, Jen-Sheng Shih, Cheng-Wen Wu A testable/fault-tolerant FFT processor design // Proceedings of the Ninth Asian Test Symposium, 2000. - P.429–433. 9. Jin-Fu Li, Cheng-Wen Wu Testable and fault tolerant design for FFT networks // International Symposium Defect and Fault Tolerance in VLSI Systems, 1999. – P.201–209. 10.Shyu-Kung Lu, Chien-Hung Yeh Easily testable and fault-tolerant design of FFT butterfly networks // Proceedings of the 11th Asian Test Symposium, 2002. – P.230–235. 11. Piuri V., Swartzlander E.E. Time-shared modular redundancy for fault-tolerant FFT processors // International Symposium Defect and Fault Tolerance in VLSI Systems, 1999. – P.265–273. 12. Tsunoyama M., Naito S. A fault-tolerant FFT processor // Twenty-First International Symposium Fault-Tolerant Computing, 1991. – P.128–135. 13 Koren I.,Krishna C.M. Fault-tolerant systems //Morgan Kaufmann Publishers. San Francisco-USA, 2007. – 399 p.