УДК 004.31: 004.627

В. А. Лужецький, Л. А. Савицька, В. А. Каплун

СПЕЦІАЛІЗОВАНИЙ ПРОЦЕСОР ДЛЯ УЩІЛЬНЕННЯ ДАНИХ

Вінницький національний технічний університет, Вінниця

Анотація. Одним із ефективних підходів щодо ущільнення даних є підхід, що базується на використтанні оптимізуючих властивостей чисел Фібоначчі. Суть підходу полягає в тому, що в процесі ущільнення блок цифрових даних розглядається як надвелике ціле додатне число, що подається у вигляді лінійної форми Фібоначчі. Реалізація методів ущільнення даних на основі лінійної форми Фібоначчі програмними засобами вимагає великих витрат часу, які пов'язані з обчисленнями над числами великої розрядності (до 8000 двійкових розрядів). Для деяких застосувань такі витрати часу є неприпустимими, тому існує потреба створення спеціалізованого процесора, який забезпечить пришвидшення процесу ущільнення даних. Розроблення математичних і структурних моделей спеціалізованого процесора та його компонентів здійснено з використанням функціонально-структурного підходу щодо проектування цифрових пристроїв. Виходячи з узагальненої моделі процесу адаптивного ущільнення даних на основі лінійної форми Фібоначчі визначено основні функції, які має реалізовувати спеціалізований процесор. Цей процесор входить до складу комп'ютерної системи і певним чином підключений до центрального процесора комп'ютера. Оскільки файли, що підлягають ущільненню, та ущільнені файли зберігаються в пам'яті комп'ютера, то передбачається, що центральний процесор буде виконувати зчитування і запис файлу, формування послідовностей P і P* та реалізовувати функцію оптимізації на рівні послідовностей. На спеціалізований процесор покладаються обчислення над числами великої розрядності. Для реалізації сукупності всіх функцій запропоновано побудувати не один операційний автомат, а здійснити його декомпозицію на автомати, кожен з яких реалізує відповідну функцію. Розглянуто математичні моделі та структури таких модулів спеціалізованого процесора: моделювання джерела даних, кодування, декодування, оптимізації на рівні блоків, формування структури послідовності **P***. Апаратна реалізація обчислень над числами великої розрядності і можливість реалізації основних функціональних перетворень окремими модулями у конвеєрному режимі забезпечує пришвидшення процесу ущільнення даних порівняно з програмною реалізацією.

Ключові слова: ущільнення даних, лінійна форма Фібоначчі, спеціалізований процесор, операційний автомат.

Abstract. One of the effective approaches to data compression is the approach based on the use of optimizing properties of Fibonacci numbers. The essence of the approach is that in the process of compaction the block of digital data is considered as a large positive integer, given in the form of a linear Fibonacci form. The implementation of data compression methods based on the linear form of Fibonacci software requires a lot of time, which is associated with calculations over large numbers (up to 8000 binary digits). For some applications, such time is unacceptable, so there is a need to create a specialized processor that will speed up the process of data compression. The development of mathematical and structural models of a specialized processor and its components is carried out using a functional-structural approach to the design of digital devices. Based on the generalized model of the process of adaptive data compression based on the linear Fibonacci form, the main functions to be implemented by a specialized processor are identified. This processor is part of a computer system and is in some way connected to the computer's CPU. Because the files to be compressed and the compressed files are stored in computer memory, the CPU is expected to read and write the file, generate P and P* sequences, and implement a sequence-level optimization function. The specialized processor is responsible for calculations over large numbers. To implement a set of all functions, it is proposed to build more than one operating machine, and to decompose it into machines, each of which implements the corresponding function. Mathematical models and structures of such modules of the specialized processor are considered: modeling of a data source, coding, decoding, optimization at the level of blocks, formation of structure of sequence P*. Hardware implementation of calculations over large numbers and the ability to implement basic functional transformations of individual modules in the pipeline mode provides acceleration of the data compression process compared to software implementation.

Key words: data compression, linear Fibonacci form, specialized processor, operating machine.
DOI: https://doi.org/10.31649/1999-9941-2022-54-2-15-25.

Вступ

Ущільнення інформації скорочує обсяг пам'яті, що необхідна для її зберігання, і кількість часу, який потрібен для її передавання каналами з фіксованою пропускною здатністю. За останнє десятиліття продуктивність процесорів зросла експоненційно порівняно зі швидкістю доступу до пристроїв пам'яті, що є ґрунтовною причиною застосовувати ущільнення інформації для збільшення загальної продуктивності системи. Задача проектування комп'ютерних систем та їх компонентів часто формулюється як задача створення засобів високої продуктивності з урахуванням заданих обмежень на апаратні витрати. Основний шлях її вирішення полягає у використанні спеціалізованих процесорів, які апаратно реалізують елементи математичних обчислень.

Актуальність

Одним із ефективних підходів щодо ущільнення даних є підхід, що базується на використтанні оптимізуючих властивостей чисел Фібоначчі [1]. Суть підходу полягає в тому, що в процесі ущільнення блок цифрових даних будь-якої довжини розглядається як надвелике ціле додатне число, що подається у вигляді лінійної форми Фібоначчі [2,3,4]. Реалізація методів ущільнення даних на основі лінійної форми Фібоначчі програмними засобами вимагає великих витрат часу, які пов'язані з обчисленнями над числами великої розрядності (до 8000 двійкових розрядів). Для деяких застосувань такі витрати часу є неприпустимими, тому існує потреба створення спеціалізованого процесора, який забезпечить пришвидшення процесу ущільнення даних.

Мета досліджень

Метою статті є пришвидшення процесу ущільнення даних, що базується на використанні лінійної форми Фібоначчі, шляхом розробки спеціалізованого процесору.

Постановка задач

Спеціалізований процесор для ущільнення даних (DC-процесор) − це цифровий операційний пристрій, який входить до складу комп'ютерної системи і певним чином підключений до центрального процесора комп'ютера. Відомо, що будь-який операційний пристрій за виконуваними функціями і структурою складається з двох частин: операційного і керуючого автоматів. Доволі часто через складність обчислювальних процедур побудова єдиного цілісного операційного автомату викликає певні труднощі. Тому на практиці використовують його декомпозицію на більш прості операційні автомати, що реалізують потрібний набір функцій. Враховуючи сказане, в даній статті розв'язуються такі задачі:

- розробка узагальненої структури спеціалізованого процесора для ущільнення даних;
- аналіз виконуваних функції та розробка структури модулів DC-процесора.

Узагальнена модель процесу адаптивного ущільнення

У роботі [5] автори запропонували таку узагальнену модель процесу адаптивного ущільнення даних на основі лінійної форми Фібоначчі:

$$\mathbf{C}_A = \left\{\mathbf{P}, \mathbf{A}, \mathbf{M}, \mathbf{C}, \mathbf{D}, \mathbf{P}_M, \mathbf{P}_C, \mathbf{P}^*, S, f\right\},$$

де $\mathbf{P}$ – вихідна послідовність символів алфавіту $\mathbf{A} = \{0,1\}$;

$\mathbf{M} = \{M_i\}$ − множина правил моделювання джерела даних;

$\mathbf{C} = \{C_j\}$ − множина правил кодування даних;

$\mathbf{D} = \{D_j\}$ − множина правил декодування даних;

$\mathbf{P}_M = \{\mathbf{P}_{M_i}\}$ − множина послідовностей, що є результатом моделювання;

$\mathbf{P}_C = \{\mathbf{P}_{C_{ij}}\}$ − множина послідовностей, що є результатом кодування;

$\mathbf{P}^*$ − послідовність ущільнених даних;

S – правило формування структури послідовності;

f – функція оптимізації.

Процес відновлення даних описується такою узагальненою моделлю:

$$\mathbf{DC} = \left\{\mathbf{A}, \mathbf{P}^*, \mathbf{P}, \mathbf{D}, S\right\},$$

де $\mathbf{A} = \{0,1\}$ − алфавіт;

$\mathbf{P}^*$ − послідовність ущільнених даних;

$\mathbf{P}$ − послідовність відновлених (вихідних) даних;

$\mathbf{D}$ − множина правил декодування даних;

S − правило формування структури послідовності $\mathbf{P}^*$.

Саме ці моделі покладено в основу створення спеціалізованого процесора для ущільнення даних.

Узагальнена структура спеціалізованого процесора для ущільнення даних

У складі комп'ютерної системи спеціалізований і центральний процесори обмінюється потоками даних, виконуючи функції, перелік яких наведено в табл. 1.

Таблиця 1 − Розподіл виконуваних функцій між процесорами

Процес	Виконувані функції	
	Центральний процесор	Спеціалізований процесор
ущільнення	1. Зчитування файлу та формування послідовності **P**. 2. Формування послідовності $\mathbf{P}^*$ та запис файлу.	1. Моделювання за правилами. 2. Кодування за правилами. 3. Оптимізація на рівні блоків.
відновлення	1. Зчитування файлу та формування послідовності $\mathbf{P}^*$. 2. Формування послідовності **P** та запис файлу.	1. Декодування за правилами.

Оскільки файли, що підлягають ущільненню, та ущільнені файли зберігаються в пам'яті комп'ютера, то передбачається, що центральний процесор буде виконувати зчитування і запис файлу,

формування послідовностей **P** і **P*** та реалізовувати функцію оптимізації на рівні послідовностей. На DC-процесор покладаються обчислення над числами великої розрядності.

Схему оброблення потоку даних у DC-процесорі наведено на рис. 1.

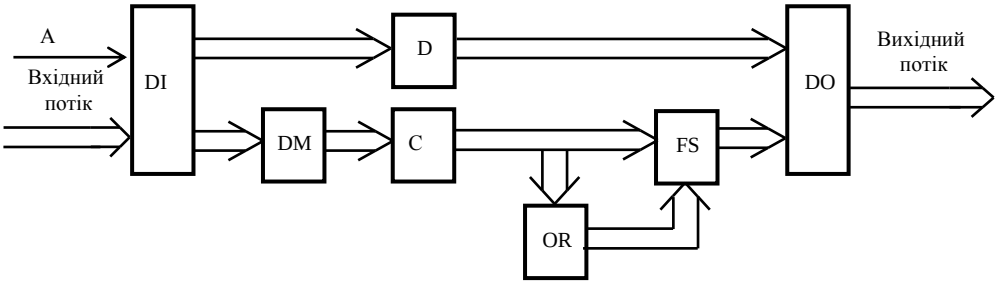

Рисунок 1 − Функціональна модель DC-процесора

Появлення заявки A на вхідній шині DC-процесора приводить до ініціювання процесу «ВХІДНИЙ ДИСПЕТЧЕР» (DI), який визначає тип заявки (ущільнення чи відновлення) і спрямовує потік даних (**P** чи **P***) до потрібного процесу. В разі заявки на ущільнення потік даних спрямовується до процесу «МОДЕЛЮВАННЯ ДЖЕРЕЛА ДАНИХ» (DM), після якого дані надходять до процесу «КОДУВАННЯ» (C). За результатами кодування ініціюється процес «ОПТИМІЗАЦІЯ РЕЗУЛЬТАТІВ» (OR), який, у свою чергу, ініціює процес «ФОРМУВАННЯ СТРУКТУРИ» (FS). У разі заявки на відновлення потік даних спрямовується до процесу «ДЕКОДУВАННЯ» (D). Завершення процесу OR або D приводить до дії процес «ВИХІДНИЙ ДИСПЕТЧЕР» (DO), який збирає готові результати і організовує з них потік даних до центрального процесора.

Для реалізації сукупності всіх процесів пропонується побудувати не один операційний автомат, а здійснити його декомпозицію на автомати, кожен з яких реалізує відповідний процес. Виходячи з цього, DC-процесор має структуру, до складу якої входять такі модулі (рис. 2):

- модуль системних операцій (МСО), що здійснює зв'язок з центральним процесором і реалізує процеси DI і DO;
- модуль моделювання джерела даних (ММДД), що реалізує процес DM;
- модуль кодування (МКод), що реалізує процес C;
- модуль декодування (МД), що реалізує процес D;
- модуль оптимізації (МО), що реалізує процес OR;
- модуль формування структури послідовності **P*** (МФСП), що реалізує процес FS;
- модуль керування (МКер), що координує роботу решти модулів.

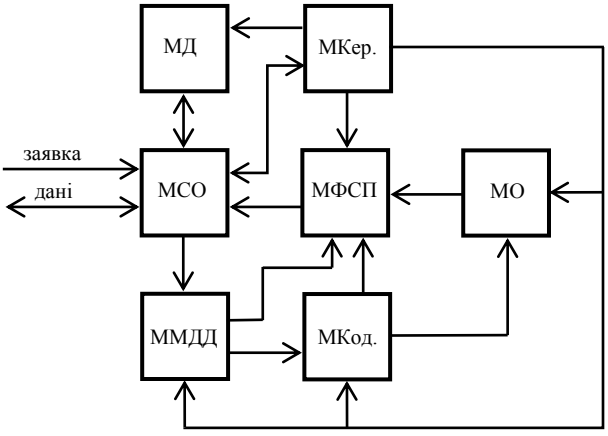

Рисунок 2 − Структура DC-процесора

Кожен з модулів МСО, ММДД, МКод, МФСП, МД і МО далі розглядається як окремий операційний автомат.

Відомо, що операційний автомат описується такими множинами [6].

1. Множиною вхідних слів **D**=$\{d_1,...,d_n\}$, що надходять до автомату як операнди.
2. Множиною вихідних слів **R**=$\{r_1,...,r_h\}$, що є результатами виконання операцій.
3. Множиною внутрішніх слів **S**=$\{s_1,...,s_n\}$, що використовуються для представлення інформації в процесі виконання операцій.

4. Множиною мікрооперацій **Y**={y_m}, m=1,...,M, що реалізують перетворення **S**=f_m(S) над словами інформації, де f_m − функція перетворення даних.

5. Множиною логічних умов **X**={x_l}, l=1,...,L, де $x_l = f_l$(**S**) і f_l − булева функція.

Таким чином, функція операційного автомата задана, якщо визначено множини **D, R, S, Y, X**. Час не є аргументом функції операційного автомату. Функція встановлює список дій, виконуваних автоматом, і характеризує засоби, що використовуються для обчислень, але не сам обчислювальний процес. Множині **D** відповідають входи операційного автомата. Множині **R** ставляться у відповідність виходи даних. Множині внутрішніх слів **S** ставляться у відповідність запам'ятовувальні пристрої, регістри та лічильники.

Порядок виконання дій у часі визначається у формі функцій керуючого автомата. Функція керуючого автомата описується схемою алгоритму, що встановлює порядок перевірки логічних умов $x_1,...,x_L$ і порядок виконання мікрооперацій $y_1,...,y_M$.

Дотримуючись цих теоретичних положень, розглянемо функції та структури модулів DC-процесора.

Структура модуля моделювання джерела даних

Запропоновані в [7] моделі джерела даних характеризуються такими параметрами:

- довжина блоку l (значення задається від $l_{\min}$ до $l_{\max}$);
- основа системи числення B (задається значення 256 або визначається за змістом блоку даних);
- значення числових еквівалентів байтів (незмінені чи зменшені на $S_{\min}$);
- порядок використання елементів послідовності $s_0, s_1, s_2, ..., s_{l-1}$, де s_j − числовий еквівалент ASCII коду j-го байту (j=0,1,2,…,l-1).

Для реалізації цих моделей потрібно таке:

1. Визначати $S_{\max} = \max(s_0, s_1, s_2, ..., s_{l-1})$ і здійснювати обчислення за формулами:

$$N_{\text{м}} = \sum_{j=0}^{l-1} s_j B^j \text{ і } N_{\text{с}} = \sum_{j=0}^{l-1} s_{(l-j-1)} B^{(l-j-1)}.$$

Число B для кожного блоку даних змінюється від $(S_{\max}+1)$ до 256.

2. Визначати $S_{\min} = \min(s_0, s_1, s_2, ..., s_{l-1})$ і здійснювати обчислення за формулами:

$$N_{\text{м}} = \sum_{j=0}^{l-1} d_j B^j \text{ і } N_{\text{с}} = \sum_{j=0}^{l-1} d_{(l-j-1)} B^{(l-j-1)}, \text{ де } d_j = s_j - S_{\min}.$$

Число B для кожного блоку даних змінюється від $(S_{\max} - S_{\min} + 1)$ до 256.

При цьому використовуються такі мікрооперації:

- порівняння «більше» (cph);
- порівняння «менше» (cpl);
- додавання (add);
- віднімання (sub);
- інкремент (inc);
- декремент (dec);
- множення (mul);
- запис у регістр (mov);
- зчитування з пам'яті (ld);
- запис у пам'ять (st).

Нехай $\mathbf{P}_i$ − блок даних, що підлягають ущільненню та N - число, що є результатом моделювання джерела даних. Тоді маємо такі множини, що описують модуль ММДД як операційний автомат.

1. Множина вхідних слів **D**={ $\mathbf{P}_i$, l}.
2. Множина вихідних слів **R**={ N, S_{min}, S_{max}, B}.
3. Множина внутрішніх слів **S**={ $\mathbf{P}_i$, l, S_{min}, S_{max}, N, B}.
4. Множина мікрооперацій **Y**={ cph, cpl, add, sub, mul, inc, dec, mov, ld, st}.
5. Множина логічних умов **X**={x_0, x_1, x_2, x_3}, де: x_0 – умова, що $s_i > s_j$; x_1 – умова, що $s_i < s_j$; x_2 – умова, що Ліч1=0, x_3 – умова, що Ліч2=257; x_4 – умова, що Ліч1=Рг1.

Зробивши функціонально-структурне відображення, в якому кожному математичному елементу формалізованого опису операційного автомату ставиться у відповідність елемент структури, отримаємо структурну схему модуля ММДД, яку наведено на рис. 3.

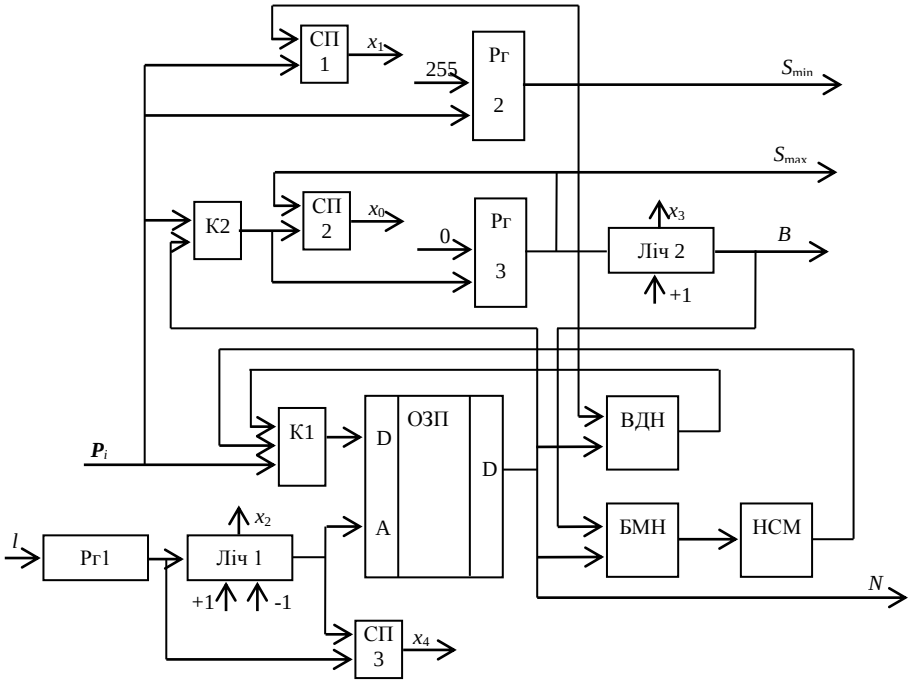

Рисунок 3 – Структурна схема модуля моделювання джерела даних

Тут пам'ять ОЗП обсягом l байтів використовується спочатку для зберігання блоку даних $\mathbf{P}_i$, а потім для накопичення коду числа N. У регістрі Рг1 зберігається код довжини блоку даних. Лічильник Ліч1 забезпечує послідовний запис і зчитування байтів даних у порядку від молодшого байту до старшого та навпаки. Схема порівняння СП1 і регістр Рг2 використовуються для визначення $S_{\min} = \min(s_0, s_1, s_2, ..., s_{l-1})$, а СП2 і Рг3 – для визначення $S_{\max} = \max(s_0, s_1, s_2, ..., s_{l-1})$. За допомогою віднімача ВДН обчислюються числові еквіваленти байтів $d_j = s_j - S_{\min}$, j=0,1,…, l-1. Блок множення БМН і нагромаджувальний суматор НСМ використовуються для обчислень за наведеними вище формулами. Лічильник Ліч2 використовується для формування значень B у діапазоні від $(S_{\max}+1)$ до 256.

Структура модуля кодування

Для кодування великих чисел використовуються два правила $C_{\text{лфф}}$ і $C_{\text{слфф}}$. Перше правило передбачає перетворення числа N у лінійну форму Фібоначчі (ЛФФ), а друге − числа N_1, що є частиною числа N. Тому основною функцією модуля кодування є формування лінійної форми Фібоначчі.

Реалізація цієї функції передбачає виконання таких дій.

1. Обчислення елементів убутної послідовності чисел Фібоначчі за формулою:

$$\varphi_1(j-2) = \varphi_1(j) - \varphi_1(j-1),\ j=12l,\ (12l-1),\ldots,1.$$

2. Визначення символів a_l коду Фібоначчі числа $N = (a_n a_{n-1} \ldots a_2 a_1)$.

3. Обчислення за формулою:

$$q = \sum_{j=n}^{1} a_j \varphi_1(j-1).$$

4. Обчислення за формулою:

$$w_i = w_{i-2} - w_{i-1},\quad i = 2, 3, \ldots,\ w_0 = N,\ w_1 = q \ \text{ поки } \ w_i > 0.$$

Якщо обчислення виконані j разів, то результатом перетворення числа N у лінійну форму Фібо-

наччі є цілі додатні числа: j, $q_2 = w_{j+1}$ і $q_1 = w_j$.

5. Формування кодів Q_j, Q_1, Q_2, l_{Q_1} і l_{Q_2}.

Враховуючи все це, маємо такі множини, що описують модуль МКод як операційний автомат.

1. Множина вхідних слів **D**={ N, l}.

2. Множина вихідних слів **R**={ Q_j, l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, Q_1, Q_2, Q_{N2}}.

3. Множина внутрішніх слів **S**={ N, N_1, N_2, l, $\varphi_1(12 \cdot l)$, $\varphi_1(12 \cdot l - 1)$, q, j, q_1, q_2}.

4. Множина мікрооперацій **Y**={ cph, add, sub, inc, dec, mov, ld, st}.

5. Множина логічних умов **X**={x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8}, де: x_0 – умова, що $n_1 > n_2$; x_1 – умова, що Ліч1=0, x_2 – умова, що Ліч2=0, x_3 – умова, що Ліч3=0, x_4 – умова, що Ліч4=0, x_5 – умова, що Ліч5=0, x_6 – умова, що ОЗП6[Ліч5]=0, x_7 – умова, що ОЗП7[Ліч5]=0, x_8 – умова, що ОЗП5[Ліч4]=0.

Виконавши функціонально-структурне відображення, отримаємо структурну схему модуля кодування, яку наведено на рис. 4.

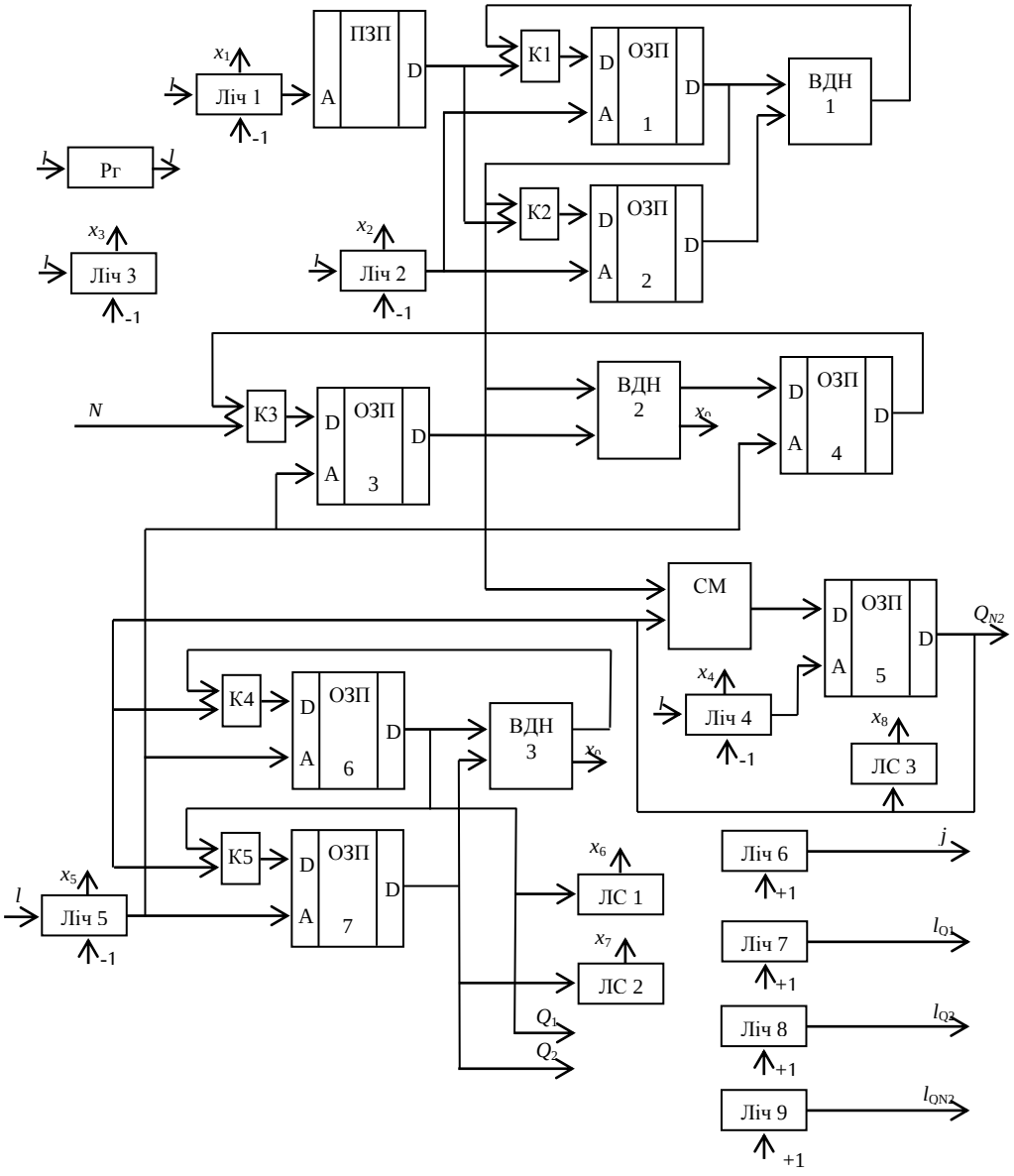

Рисунок 4 − Структурна схема модуля кодування

У постійному запам'ятовувальному пристрої (ПЗП) зберігаються два числа Фібоначчі, які використовуються як початкові для формування убутної послідовності чисел Фібоначчі. Оскільки ці числа мають велику розрядність, то вони зберігаються у певній кількості комірок ПЗП. Для вибору потрібної адреси

комірки використовується лічильник Ліч1. Перед початком формування послідовності чисел Фібоначчі початкові числа переписуються в оперативні запам'ятовувальні пристрої ОЗП1 і ОЗП2. Формування кодів адрес для них забезпечує лічильник Ліч2. Віднімач ВДН1 використовується для обчислення послідовності чисел Фібоначчі, а ВДН2 спільно з ОЗП3 і ОЗП4 − для порівняння числа, що перетворюється, з числами Фібоначчі. Лічильник Ліч3 формує коди адрес для ОЗП3 і ОЗП4. Зміну напрямків надходження даних до ОЗП1, ОЗП2, ОЗП3, ОЗП6 і ОЗП7 забезпечують комутатори К1, К2, К3, К4 і К5, відповідно.

Обчислення q забезпечується суматором СМ і ОЗП5, крім цього вони забезпечують обчислення числа N_2 при формуванні скороченої ЛФФ(*). Складові j, q_1 і q_2 лінійної форми Фібоначчі визначаються з використанням ОЗП6, ОЗП7, ВДН3 і Ліч6. Коди l_{Q_1}, l_{Q_2} і l_{QN2} одержуються за допомогою Ліч7, Ліч8 і Ліч9. Логічні схеми ЛС1, ЛС2 і ЛС3 формують логічну "1", коли на їхні входи надходить код, що складається з усіх нулів. Це забезпечує відкидання старших незначущих байтів коду числа.

Структура модуля декодування

Основою правил декодування є зворотне перетворення Фібоначчі, яке передбачає виконання j разів обчислень за рекурентною формулою $w_{i+1} = w_i + w_{i-1}, \quad i = 1, 2, .., j$.

У разі правила декодування $D_{\text{лфф}}$ число N є остаточним результатом декодування. Для правила декодування $D_{\text{сллф}}$ результатом обчислень є число N_1, до якого потрібно додати число N_2, щоб отримати остаточний результат. Число N_2 − це числовий еквівалент складової коду ущільненого блоку даних.

Враховуючи сказане, маємо такі множини, що описують модуль МД як операційний автомат.

1. Множина вхідних слів **D**={Q_j, l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, Q_1, Q_2, Q_{N2}, l}.

2. Множина вихідних слів **R**={N}.

3. Множина внутрішніх слів **S**={Q_j, l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, Q_1, Q_2, Q_{N2}, N, l }.

4. Множина мікрооперацій **Y**={add, dec, mov, ld, st}.

5. Множина логічних умов **X**={x_1, x_2, x_3, x_4}, де: x_1 – умова, що Ліч1=0, x_2 – умова, що Ліч2=0, x_3 – умова, що Ліч3=0, x_4 – умова, що Ліч4=Рг.

Виконавши функціонально-структурне відображення, отримаємо структурну схему модуля декодування, яку наведено на рис. 5.

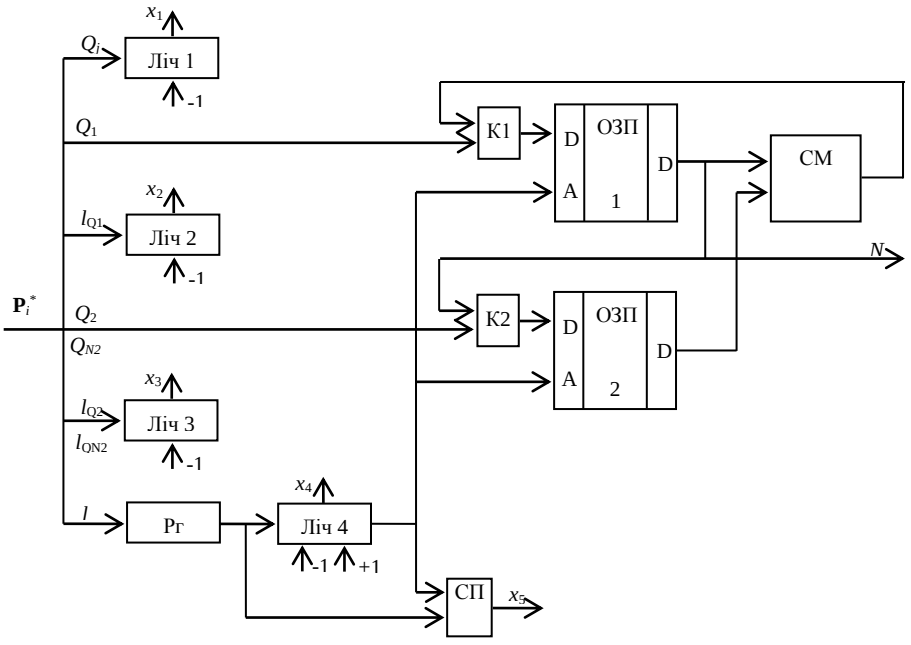

Рисунок 5 − Структурна схема модуля декодування

Тут лічильники Ліч1, Ліч2 і Ліч3 зберігають початкові та поточні значення кодів Q_j, l_{Q_1} і l_{Q_2} ($l_{Q_{N2}}$), відповідно. Лічильники Ліч2 і Ліч3 забезпечують запис кодів Q_1 і $Q_2(Q_{N2})$ в оперативні запам'ятовувальні пристрої ОЗП1 і ОЗП2. Кількість обчислень за рекурентною формулою (2.13) визначається змістом Ліч1. Лічильник Ліч4 формує коди адрес для ОЗП1 і ОЗП2 в процесі обчислень і при виведенні кодів результату декодування. Зміну напрямків надходження даних до ОЗП1 і ОЗП2 забезпечують комутатори К1 і К2.

Структура модуля оптимізації

Для вибору найкращого результату ущільнення використовується така функція оптимізації на рівні блоків даних:

$$f_{\text{бл}}^{\text{нп}} = \min\left\{\left(l_{\text{ф}}^{(1)}, l_{\text{ф}}^{(2)}, l_{\text{сф}}^{(1)}, l_{\text{сф}}^{(2)}\right)^{(i)}, l\right\}.$$

Ця функція передбачає спочатку вибір серед лінійних форм Фібоначчі такої форми, що утворює перетворений блок найменшої довжини $l_{\text{перет.}}$, і формування ознак:

$$c_1 = 0 \text{ і } c_2 = 0, \text{ якщо } l_{\text{перет.}} = l_{\text{ф}}^{(1)} = \min\left(l_{\text{ф}}^{(1)}, l_{\text{ф}}^{(2)}, l_{\text{сф}}^{(1)}, l_{\text{сф}}^{(2)}\right);$$

$$c_1 = 1 \text{ і } c_2 = 0, \text{ якщо } l_{\text{перет.}} = l_{\text{ф}}^{(2)} = \min\left(l_{\text{ф}}^{(1)}, l_{\text{ф}}^{(2)}, l_{\text{сф}}^{(1)}, l_{\text{сф}}^{(2)}\right);$$

$$c_1 = 0 \text{ і } c_2 = 1, \text{ якщо } l_{\text{перет.}} = l_{\text{сф}}^{(1)} = \min\left(l_{\text{ф}}^{(1)}, l_{\text{ф}}^{(2)}, l_{\text{сф}}^{(1)}, l_{\text{сф}}^{(2)}\right);$$

$$c_1 = 1 \text{ і } c_2 = 1, \text{ якщо } l_{\text{перет.}} = l_{\text{сф}}^{(2)} = \min\left(l_{\text{ф}}^{(1)}, l_{\text{ф}}^{(2)}, l_{\text{сф}}^{(1)}, l_{\text{сф}}^{(2)}\right).$$

Потім порівнюються довжина лінійної форми Фібоначчі $l_{\text{перет.}}$ і довжина початкового блоку даних l. Якщо $l_{\text{перет.}} < l$, то формується ознака p=1, а якщо $l_{\text{перет.}} \geq l$, то формується ознака p=0.

Виходячи з цього, маємо такі множини, що описують модуль МО як операційний автомат.

1. Множина вхідних слів **D**={ l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, Q_1, Q_2, Q_{N2}, l}.

2. Множина вихідних слів **R**={ l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, Q_1, Q_2, Q_{N2}, c_1, c_2, p }.

3. Множина внутрішніх слів **S**={ l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, Q_1, Q_2, Q_{N2}, c_1, c_2, p, l, $l_{\text{перет.}}$}.

4. Множина мікрооперацій **Y**={add, mov, cph}.

5. Множина логічних умов **X**={x_1, x_2, x_3, x_4, x_5, x_6}, де: x_1 – умова, що $l > l_{\text{ф}}^{(\ )}$, x_2 – умова, що $l_{\text{ф}}^{(2)} > l_{\text{ф}}^{(1)}$, x_3 – умова, що $l_{\text{сф}}^{(\ )} > l_{\text{ф}}^{(\ )}$, x_4 – умова, що $l_{\text{сф}}^{(2)} > l_{\text{сф}}^{(1)}$, x_5 – умова, що $l > l_{\text{сф}}^{(\ )}$, x_6 – умова, що B>256.

Виконавши функціонально-структурне відображення, отримаємо структурну схему модуля оптимізації, яку наведено на рис. 6.

Тут регістри Рг1, Рг2 , Рг3, Рг4 і Рг5 зберігають початкові значення кодів l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, l і B, відповідно. Регістри Рг6 − Рг11 призначені для зберігання поточних значень кодів l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$ та їх сум. Зміну напрямків надходження кодів до Рг13 і Рг14, в які записуються остаточні результати, забезпечують комутатори К1 і К2. Для порівняння значень кодів l_{Q_1}, l_{Q_2}, $l_{Q_{N2}}$, l і B за алгоритмом ущільнення використовуються схеми порівняння СП1 − СП6. Для зберігання ознак, що формуються за результатами порівняння, використовуються тригери Т1 − Т3.

Структура модуля формування структури послідовності

Результат ущільнення складається з певного набору елементів, який має таку структуру:

$$S = \left\{l \,\|\, l_k \,\|\, l_\pi \,\|\, \pi \,\|\, \text{Бл}^*1 \,\|\, \text{Бл}^*2 \,\|\, ... \,\|\, \text{Бл}^*k\right\},$$

де $\text{Бл}^*i = \left\{B_{\text{o}} \,\|\, S_{\min} \,\|\, Q_j \,\|\, l_{Q_1} \,\|\, l_{Q_2} \,\|\, Q_1 \,\|\, Q_2\right\}$.

Тут складники мають такі довжини: l, l_k, l_π, Q_j, l_{Q_1}, l_{Q_2} – 2 байти; B_o, $S_{\min}$ – 1 байт; Q_1 – l_{Q_1} байтів; $Q_2 - l_{Q_2}$ байтів; $\pi - k/8$ байтів.

Згідно з розподілом виконуваних функцій між процесорами (див. табл. 1), формування послідовності **P***, що має структуру S, та запис файлу реалізуються центральним процесором, а формування блоку перетворених даних здійснюється спеціалізованим процесором. З метою зменшення часу обміну даними між процесорами пропонується формувати послідовність бітових ознак π вигляді сукупності байтів у спеціалізованому процесорі.

Виходячи з цього, маємо такі множини, що описують модуль МФСП як операційний автомат.

1. Множина вхідних слів **D**={ l, l_k, m, Q_j, l_{Q_1}, l_{Q_2}, l_{QN2}, Q_N, Q_1, Q_2, Q_{N2}, B_o, $S_{\min}$, c_1, c_2, p}.

2. Множина вихідних слів **R**={ l, l_k, l_π, π, C, m, Бл*i }.

3. Множина внутрішніх слів **S**={l, l_k, m, l_{Q_1}, l_{Q_2}, l_{QN2}, π, C, Бл*i }.

4. Множина мікрооперацій **Y**={inc, dec, mov, ld, st }.

5. Множина логічних умов **X**={x_1, x_2, x_3, x_4, x_5}, де: x_1, x_2, x_3, x_4, x_5 – умова, що Ліч1=0, Ліч2=0, Ліч3=0, Ліч4=0, Ліч5=0, відповідно.

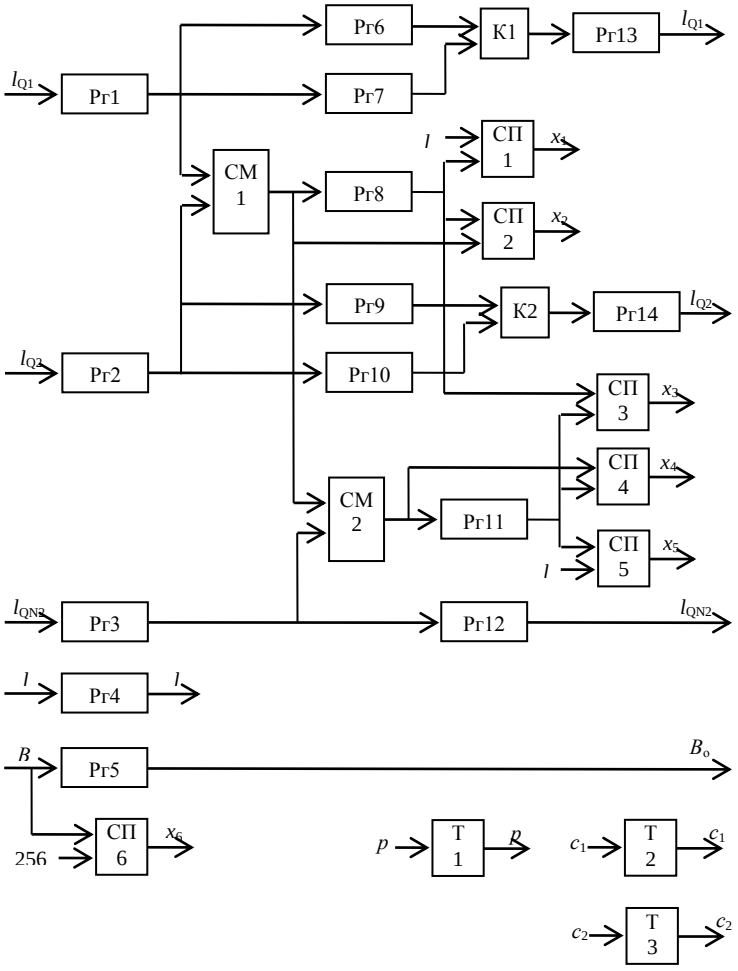

Рисунок 6 − Структурна схема модуля оптимізації

Виходячи з цього, маємо такі множини, що описують модуль МФСП як операційний автомат.

1. Множина вхідних слів **D**={ l, l_k, m, Q_j, l_{Q_1}, l_{Q_2}, l_{QN2}, Q_N, Q_1, Q_2, Q_{N2}, B_o, $S_{\min}$, c_1, c_2, p}.

2. Множина вихідних слів **R**={ l, l_k, l_π, π, C, m, Бл*i }.

3. Множина внутрішніх слів **S**={l, l_k, m, l_{Q_1}, l_{Q_2}, l_{QN2}, π, C, Бл*i }.

4. Множина мікрооперацій **Y**={inc, dec, mov, ld, st }.

5. Множина логічних умов **X**={x_1, x_2, x_3, x_4, x_5}, де: x_1, x_2, x_3, x_4, x_5 – умова, що Ліч1=0, Ліч2=0, Ліч3=0, Ліч4=0, Ліч5=0, відповідно.

Виконавши функціонально-структурне відображення, отримаємо структурну схему модуля формування структури послідовності, яку наведено на рис. 7.

Тут регістри Рг1 − Рг8 і Рг11 − Рг15 зберігають значення кодів *l*, l_k, *m*, Q_j, l_{Q_1}, l_{Q_2}, l_{QN2}, Q_N, Q_1, Q_2, Q_{N2}, B_0, $S_{\min}$. У регістрі Рг9 з бітових ознак *p* формується байт ознак шляхом зсуву коду. Лічильник Ліч1 підраховує кількість виконаних зсувів. Регістр Рг10 і лічильник Ліч2 забезпечують формування байтів ознак *C*. Байти ознак π і *C* накопичуються в пам'яті ОЗП1 і ОЗП2, відповідно. Комутатори К1, К2, К3, лічильник Ліч5 і пам'ять ОЗП3 забезпечують формування структури блоку Бл$^*i$.

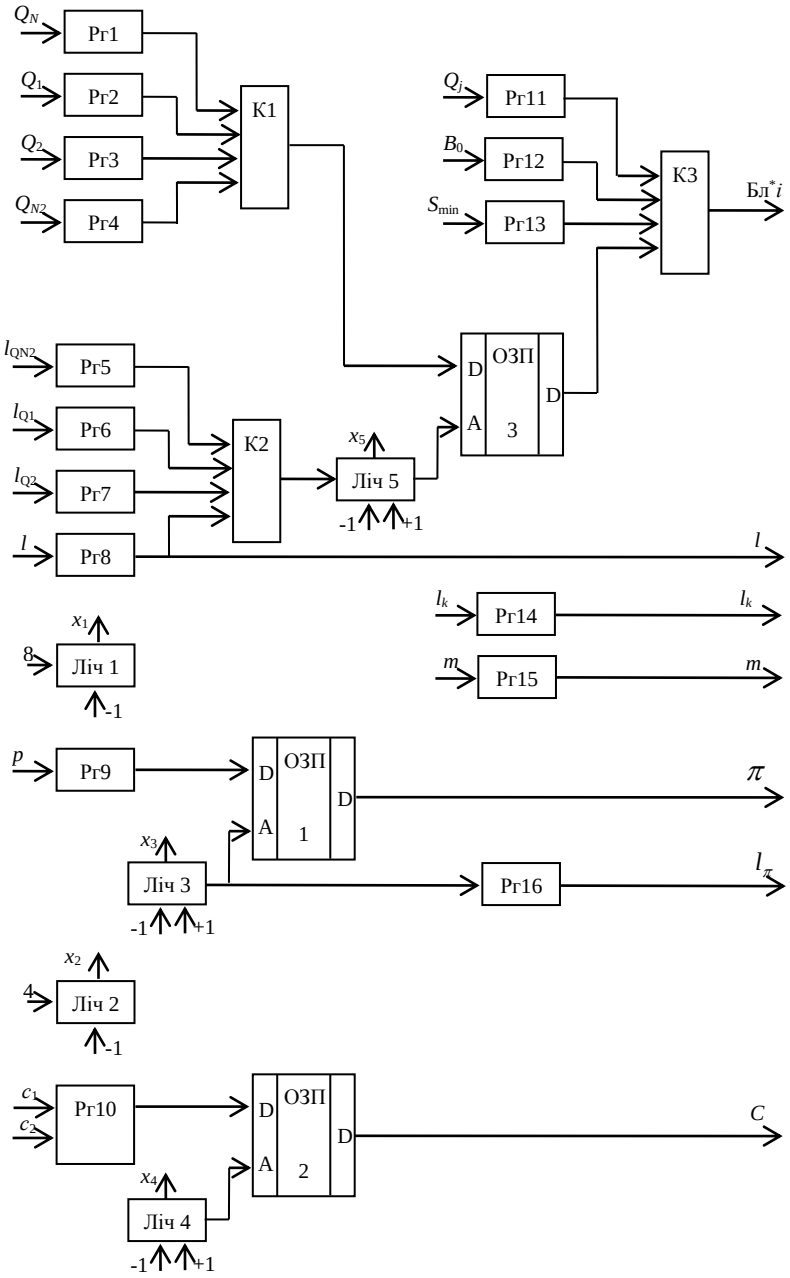

Рисунок 7 − Структурна схема модуля МФСП

Висновки

Реалізація методів ущільнення даних на основі лінійної форми Фібоначчі програмними засобами вимагає великих витрат часу, які є неприпустимими для деяких застосувань, тому для скорочення витрат часу розроблено структури спеціалізованого процесора та його компонентів. Оскільки значна частина часових витрат пов'язана з обчисленнями над числами великої розрядності (до 8000 двійкових розрядів), тому саме ці обчислення реалізуються апаратно DC-процесором, який певним чином підключений до

центрального процесора комп'ютера. При цьому центральний процесор забезпечує зчитування і запис файлу, формування послідовностей **P** і $\mathbf{P}^*$ та реалізацію функцій оптимізації на рівні послідовностей.

Використаний функціонально-структурний підхід щодо проектування цифрових пристроїв забезпечив розробку математичних і структурних моделей спеціалізованого процесора та його компонентів. Побудова окремих модулів для реалізації основних функціональних перетворень забезпечує одночасне виконання цих перетворень у конвеєрному режимі. Апаратна реалізація обчислень над числами великої розрядності і можливість реалізації основних функціональних перетворень у конвеєрному режимі забезпечує пришвидшення процесу ущільнення даних порівняно з програмною реалізацією.

Список літератури

[1] В. А. Лужецький, Л. А. Савицька, "Розробка та дослідження методів адаптивного ущільнення даних на основі лінійної форми фібоначчі," *Східно-європейський журнал передових технологій*, №1/9 (73), с. 16-22, 2015.

[2] А. В. Анисимов, Я. П. Рындин и С. Е. Редько, "Обратное преобразование Фибоначчи," *Кибернетика*, № 3, с. 9-11, 1982.

[3] А. В. Анисимов, "Линейные формы Фибоначчи и параллельные алгоритмы большой размерности," *Кибернетика и системный анализ,* № 3, с. 106–115, 1995.

[4] В. А. Лужецький і Мохаммад Аль-Майта, "Спосіб зображення цілих чисел великого діапазону," *Вимірювальна та обчислювальна техніка в технологічних процесах*, № 1, с. 156–162, 1998.

[5] В. А. Лужецький, Л. А. Савицька, Шахзада Ашрафул Хок, "Узагальнена модель адаптивного ущільнення даних," *Інформаційні технології та комп'ютерна інженерія*, № 1(14), с. 56-63, 2009.

[6] В. М. Глушков, Ю. В. Капитонова, А. Т. Мищенко, *Логическое проектирование дискретных устройств*. К.: Наукова думка, 1987.

[7] В. А. Лужецький, Л. А. Савицька, "Моделі і методи адаптивного ущільнення даних на основі лінійної форми Фібоначчі," *Збірник наукових праць Харківського університету Повітряних сил*, № 1 (42), с. 53-57, 2015.

Стаття надійшла: 25.04.2022.

References

[1] V. A. Luzhetskyi, L. A. Savytska, "Rozrobka ta doslidzhennya metodiv adaptyvnoho ushchilnennya danykh na osnovi liniynoyi formy Fibonachchi," Skhidno-yevropeyskyi zhurnal peredovykh tekhnolohiy, №1/9 (73), s. 16-22, 2015 [in Ukrainian].

[2] A. V. Anisimov, Ya. P. Ryndin, S. Ye. Redko, "Obratnoye preobrazovaniye Fibonachchi," Kibernetika, № 3, c. 9-11, 1982 [in Russian].

[3] A. V. Anisimov, "Lineynyye formy Fibonachchi i parallelnyye algoritmy bolshoy razmernosti," Kibernetika i sistemnyy analiz, № 3, c. 106–115, 1995 [in Russian].

[4] V. A. Luzhetskyi, Mokhammad Al-Mayta, "Sposib zobrazhennya tsilykh chysel velykoho diapazonu," Vymiryuvalna ta obchyslyuvalna tekhnika v tekhnolohichnykh protsesakh, № 1, c. 156–162, 1998 [in Ukrainian].

[5] V. A. Luzhetskyi, L. A. Savytska, Shakhzada Ashraful Khok, "Uzahalnena model adaptyvnoho ushchilnennya danykh," Informatsiyni tekhnolohiyi ta kompyuterna inzheneriya, № 1(14), c. 56-63, 2009 [in Ukrainian].

[6] V. M. Glushkov, Yu. V. Kapitonova, A. T. Mishchenko, *Logicheskoye proyektirovaniye diskretnykh ustroystv*. K.: Naukova dumka. 1987 [in Russian].

[7] V. A. Luzhetskyi, L. A. Savytska, "Modeli ta metody adaptyvnoho ushchilnennya danykh na osnovi liniynoyi formy Fibonachchi," *Zbirnyk naukovykh prats Kharkivskoho universytetu Povitryanykh syl*, № 1 (42), c. 53-57, 2015 [in Ukrainian].

Відомості про авторів

Лужецький Володимир Андрійович – доктор технічних наук, професор, завідувач кафедри захисту інформації.

Савицька Людмила Анатоліївна – кандидат технічних наук, доцент кафедри обчислювальної техніки.

Каплун Валентина Аполінаріївна – старший викладач кафедри захисту інформації.

V. A. Luzhetskyi, L. A. Savitska, V. A. Kaplun

SPECIALIZED DATA COMPRESSION PROCESSOR

Vinnytsia National Technical University, Vinnytsia