pressure sensor IC // IEEE J. Solid-State Circuits. – Dec. 1986. – Vol. SC-21. – P. 1045–1056. 3. Зенкевич О. Метод конечных элементов в технике. – М.: Мир, 1975. – 541 с. 4. Rusinski E. Metoda elementow skonczonych. System Cosmos. – M. – Warszawa: WKL, 1994. – 392 s. 5. Algor: Mechanical event simulation tutorial. Algor.inc, Pittsburgh, 2000.

А.О. Мельник, Р.В. Бачинський
Національний університет "Львівська політехніка"
кафедра електронних обчислювальних машин

ПРОЦЕСОРИ СТИСКУ ДАНИХ З ПЛАВАЮЧИМ ВІКНОМ БУФЕРА ПОПЕРЕДНЬО ЗБЕРЕЖЕНИХ ДАНИХ

Описано структуру процесора стиску та відновлення стиснених даних з плаваючим вікном буфера попередньо збережених даних.

Compression and decompression processor structure with floating window buffer is described.

Вступ. Проникнення інформаційних технологій у різні сфери побуту та промисловості спричинило лавиноподібне зростання обсягів інформації, яку необхідно обробляти, передавати та зберігати. Крім цього, постійне зростання кількості користувачів глобальної мережі Internet призводить до необхідності економного та ефективного використання каналів передачі даних для забезпечення всіх абонентів якісним сервісом.

Для ефективного використання ресурсів систем передачі та збереження даних використовуються системи стиску даних. Серед них слід виділити універсальні алгоритми стиску даних без втрат, які забезпечують відновлення стиснених даних без спотворень. Прикладом таких алгоритмів є алгоритми арифметичного кодування та різні варіанти кодування Хаффмана, а також алгоритми стиску даних Лемпеля–Зіва [1]. Тут розглянуто основні вимоги до систем стиску даних, структури процесорів стиску даних з плаваючим вікном буфера попередньо збережених даних та процесора відновлення стиснених даних.

Алгоритму стиску, основаного на використанні буфера попередньо збережених даних з плаваючим вікном. Алгоритм Лемпеля–Зіва оснований на пошуку однакових послідовностей байт у вхідній інформації і заміні цих послідовностей коротшими кодами, з яких можна під час відновлення отримати вхідні дані. Існує велика кількість реалізацій алгоритму Лемпеля–Зіва, які відрізняються методами пошуку однакових послідовностей та методами формування кодів для їх заміни. Для реалізації цього алгоритму необхідно зберігати частину попередньо прийнятих даних, в яких процесор стиску буде шукати збігання з плинною порцією даних і заміняти їх на коди, які залежать від розташування даних, які збігаються в цьому буфері, а процесор відновлення буде використовувати цей буфер для заміни кодів на відповідну їм послідовність. Візьмемо для прикладу варіант алгоритму Лемпеля–Зіва, запропонований для реалізації в RFC-2118 [2] та в RFC-1974 [3]. Цей варіант використовує восьмикілобайтний (RFC-2118) та двокілобайтний (RFC-1974) буфер і коди змінної довжини. Послідовності, які збігаються кодуються парою зміщення/довжина, де зміщення – це зміщення від початку буфера до початку збігання, а довжина – це довжина збігання в байтах. Для того, щоб розрізняти лексеми у вихідному потоці, вони кодуються спеціальним чином.

Основні вимоги до систем стиску даних та їхні характеристики. Оскільки основна задача систем стиску – зменшення об'ємів даних, яку необхідно передавати та зберігати, до більшості з них ставляться однотипні вимоги, хоча різні предметні області їх використання можуть ставити свої специфічні вимоги. Серед найважливіших вимог до систем стиску даних можна виділити такі:

- Забезпечення високого ступеня стиску для різних типів даних. Цей параметр залежить від вибраного алгоритму стиску.
- Мінімізація втрати ефективності при виникненні несприятливих умов для вибраного алгоритму стиску. Деякі алгоритми стиску можуть збільшувати об'єм даних при обробці даних, які не містять надлишковості. Можливість мінімізації втрати ефективності залежить як від алгоритму стиску, так і від архітектури системи, яка його реалізує. Наприклад, в системі можна передбачити використання аналізатора вхідного потоку даних, який у разі втрати ефективності від використання алгоритму стиску може перейти на використання іншого алгоритму стиску або відімкнути систему стиску взагалі і пропускати вхідний потік на вихід без змін.
- Забезпечення високої швидкості стиску даних. Виконання цієї вимоги залежить і від алгоритму стиску і від вибраних засобів його реалізації. Так, деякі алгоритми вимагають попереднього аналізу даних, які підлягають стиску (двопрохідні алгоритми). Швидкість стиску залежить також від архітектури процесора, засобів його реалізації (НВІС, ПЛІС, універсальний процесор), тактової частоти роботи процесора.
- Забезпечення можливості одночасної обробки декількох незалежних потоків даних (багатоканальність) з швидким перемиканням між каналами (переключення контексту). Ця вимога особливо актуальна для систем мережі, де багато вхідних каналів передачі даних мультиплексуються в один або декілька вихідних.
- Забезпечення невеликих потреб в апаратних ресурсах. Ця вимога актуальна для інтегрованих систем, де на одному кристалі розміщені різні підсистеми (в тому числі підсистема стиску даних).
- Забезпечення зручного інтерфейсу для під'єднання системи стиску до інших підсистем.
- Структура системи стиску повинна забезпечувати можливість швидкої перебудови для побудови на її базі аналогічних систем з іншими експлуатаційними характеристиками та архітектурними особливостями (забезпечення можливості реконфігурування системи).
- Забезпечення багатофункціональності. Система стиску повинна бути спроектована таким чином, щоб її можна було використовувати в різних областях можливого застосування без зміни її архітектури.

Основними елементами системи стиску даних є процесор стиску та процесор відновлення стиснених даних. У наступних розділах будуть розглянуті структури процесора стиску та процесора відновлення стиснених даних для алгоритмів стиску даних з плаваючим вікном буфера попередньо збережених даних.

Структура процесора стиску даних з плаваючим вікном буфера попередньо збережених даних. Розглянемо структуру процесора стиску даних, яка базується на використанні пам'яті з послідовним доступом. Альтернативою використанню пам'яті з послідовним доступом може бути регістрова пам'ять, коли буфер реалізується як один великий регістр зсуву, в якому порівняння вхідних даних відбувається одночасно у всіх комірках. Проте така система вимагає проектування нового кристала "з нуля", складних схем дешифрування кожної комірки регістра і через великі апаратні витрати її неможливо реалізувати на сучасних ПЛІС.

Процесор стиску даних складається з таких блоків: вхідного буфера; блока пам'яті з послідовним доступом; блока керування; операційного блока; буферного пристрою накопичення лексем; вихідного контролера.

Вхідний буфер накопичує вхідний потік даних. Блок керування аналізує вхідний потік даних, шукає надлишковості у цьому потоці, генерує лексеми (літерали та пари зміщення/довжина) та керує іншими пристроями процесора стиску даних. Блок керування є найскладнішим в системі і від продуктивності його роботи в парі з блоком пам'яті залежить продуктивність системи в цілому. Операційний блок модифікує літерали, пари зміщення/довжина і формує з них лексеми згідно з вимогами алгоритму. Операційний блок керується блоком керування. Буферний пристрій накопичення лексем накопичує і зберігає лексеми у випадку, коли вихідний контролер не встигає видавати весь потік даних на вихід або коли зовнішній пристрій – споживач стиснених даних, не

встигає її забирати. Вихідний контролер формує з послідовного набору лексем вихідний потік. Вихідний потік може бути послідовним або паралельним, при цьому, вихідний контролер з послідовності лексем різної довжини формує заданий потік даних.

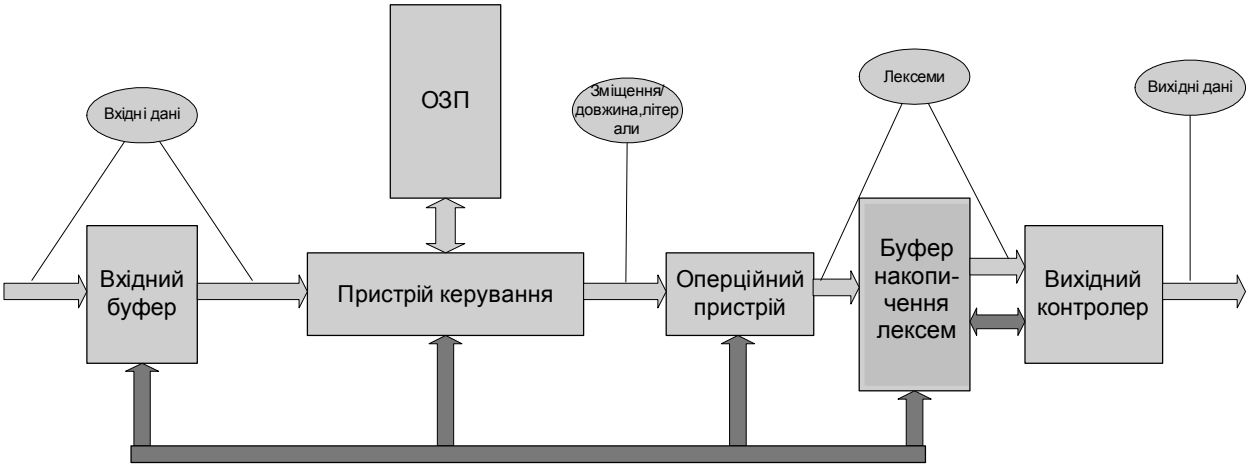

Рис. 1. Структура процесора стиску даних на пам'яті з послідовним доступом

Порівняно з процесором стиску, який використовує асоціативну пам'ять як буфер, ця структура характеризується значними затратами часу на пошук однакових послідовностей даних через послідовну природу пам'яті, що вимагає повного перегляду буфера для кожного порівняння. При цьому ускладнюється пристрій керування, який керує вибіркою з пам'яті та операціями порівняння, а також виникає необхідність у використанні додаткових структур даних для збереження проміжних результатів порівняння, що вимагає використання додаткової пам'яті і збільшує апаратні затрати на реалізацію системи. Але використання пам'яті з послідовним доступом забезпечує простоту і уніфікованість її складових компонент, що дозволяє у разі проектування кристалу використовувати вже створені бібліотечні компоненти, що здешевлює процес реалізації системи на кристалі і скорочує терміни його виконання. Крім цього, така структура може бути реалізована на сучасних ПЛІС. Ця структура допускає багатоканальність та переключення контексту, хоча й вимагає для цього значну кількість пам'яті і ускладнює керування.

Структура процесора відновлення стиснених даних з плаваючим вікном буфера попередньо збережених даних. Процесор відновлення стиснених даних складається з таких основних блоків: вхідного контролера; операційного блока; буферного пристрою для зберігання лексем; головного контролера; циклічного буфера, який складається з блока двохпортової пам'яті та керуючої логіки.

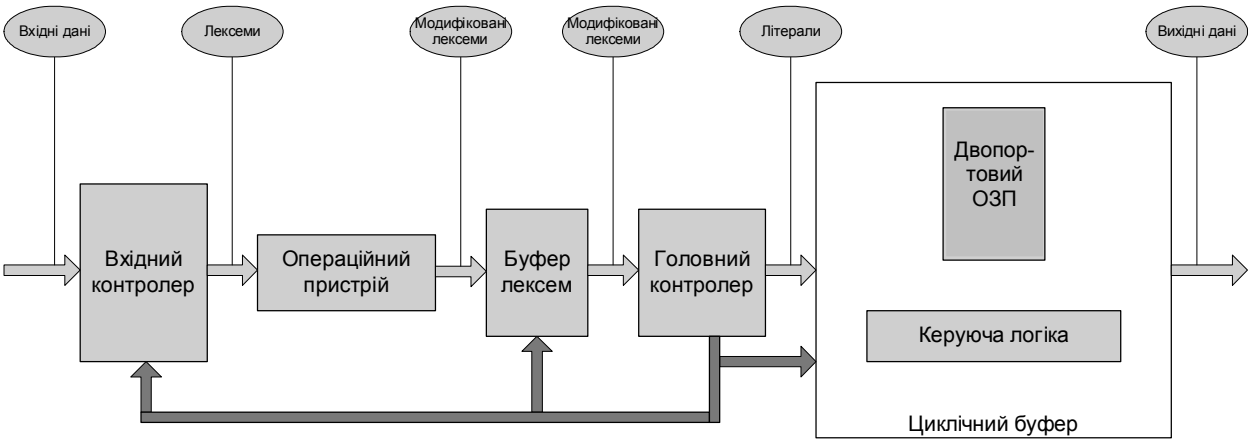

Рис. 2. Структура системи відновлення стиснених даних на двопортовій пам'яті з послідовним доступом

Вхідний контролер приймає і аналізує вхідний потік даних та розділяє його на лексеми. Лексеми в алгоритмах даного класу – це незалежні одиниці стиснених даних, які зазвичай можуть бути двох типів: літерали та лексична пара: зміщення/довжина. Літерал – це модифікована нестиснена одиниця даних. Пара зміщення/довжина – це ознака стисненої порції даних, яка являє собою зміщення від початку буфера та довжину повторюваного блока даних. Операційний блок відновлює лексеми від модифікації, яка могла мати місце при стисненні даних (визначається алгоритмом). Буферний пристрій зберігання лексем накопичує лексеми в тому випадку, коли кількість вихідних даних значно перевищує кількість вхідних даних, що може спостерігатися під час оброблення стиснених даних з високим коефіцієнтом стиску. Головний контролер керує роботою цілого пристрою. Він вибирає лексеми з накопичувального буфера і відправляє їх на циклічний буфер, де залежно від типу лексеми виконуються ті чи інші дії. Циклічний буфер зберігає певну кількість попередньо відновлених даних, яка використовується під час відновлення і складається з блока двопортової пам'яті і керуючої логіки. Керуюча логіка реалізує зациклювання буфера, приймає і обробляє лексеми.

Ця структура є гнучкою і її можна легко пристосувати для різних варіантів алгоритму відновлення. Зазвичай зміни торкаються тільки вхідного контролера та операційного блока. Крім цього, ця структура допускає багатоканальний режим роботи та переключення контексту. Для цього необхідно мати буферний пристрій для зберігання лексем та блок двопортової пам'яті окремо для кожного каналу.

Висновки. Запропонована структура системи стиску відповідає більшості вимог до систем стиску даних і може бути реалізована на сучасних ПЛІС, які містять блоки двопортової пам'яті. Однією з найбільших переваг цієї системи є можливість реалізувати її на сучасних ПЛІС, що спрощує тестування і скорочує терміни розроблення та введення системи в експлуатацію.

1. Lempel A. and Ziv J. A Universal Algorithm for Sequential Data Compression // IEEE Transactions On Information Theory, Vol. IT-23, №. 3, May 1977. 2. [RFC 2118] Microsoft Point-To-Point Compression (MPPC) Protocol. 3. [RFC 1974] PPP Stac LZS Compression Protocol.

УДК 681.3

А.О. Мельник, О.М. Сокіл
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ПРОЕКТУВАННЯ ОПТИМІЗОВАНИХ СТРУКТУР КОМП'ЮТЕРНИХ ПРИСТРОЇВ ПІДБОРОМ НЕОБХІДНИХ ЕЛЕМЕНТІВ БІБЛІОТЕКИ З ВИКОРИСТАННЯМ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Запропоновано новий підхід до проектування оптимізованих структур комп'ютерних пристроїв підбором необхідних елементів бібліотеки з використанням генетичних алгоритмів.

The new solution for an optimized structures of computer devices development, using genetic algorithms was proposed.

Вступ. Проектування комп'ютерних пристроїв є досить складним процесом. По-перше, існує багато підходів до побудови структури комп'ютерного пристрою та необхідність вибору оптимальної з множини можливих структур. По-друге, є багато варіантів відображення функціональних