УДК 004.71

Крайник Я. М.,
аспірант кафедри інформаційних технологій та програмних систем,
Чорноморський державний університет імені Петра Могили, м. Миколаїв, Україна

ПОБУДОВА ЧАСТКОВО ПАРАЛЕЛЬНОГО LDPC-ДЕКОДЕРУ З ВИКОРИСТАННЯМ ОСОБЛИВОСТЕЙ ДВОПОРТОВОЇ БЛОКОВОЇ ПАМ'ЯТІ ПЛІС

У статті досліджено проблему побудови частково паралельного LDPC-декодеру для матриці перевірки парності з довільним характером розташування значущих елементів. Представлено розроблений алгоритм для попередньої обробки матриці на основі перестановки рядків, що забезпечує уникнення колізій. Запропонована організація пам'яті збереження індексів та її взаємодія з іншими компонентами декодеру, зокрема, блоками декодування. Використання особливостей блокової пам'яті ПЛІС дозволило забезпечити коректну паралельну роботу двох таких блоків. На основі проведених досліджень реалізовано LDPC-декодер, пропускну здатність якого збільшено в два рази.

Ключові слова: *LDPC-декодер; ПЛІС; блокова пам'ять; декодування.*

Постановка проблеми

Матриця перевірки парності LDPC-декодеру (англ. *Low Density Parity Check, LDPC*), частіше за все, представляє собою розріджену матрицю зі значущими елементами, що мають значення 1. Кількість значущих елементів у кожному рядку є значно меншою за загальну довжину рядка. Кожен рядок представляє рівняння перевірки парності для складових отриманого повідомлення, на основі якого можна визначити наявність/відсутність помилок у вхідному повідомленні. У випадку, якщо результати перевірки вказують на виникнення помилки, то виконується пряме виправлення помилок (англ. *Forward Error Correction, FEC*) без передачі додаткових даних у зворотному напрямку для повторної передачі, уточнення тощо.

Для виправлення помилок використовуються алгоритми на основі обміну повідомленнями (англ. *Message Passing*). Найбільшу ефективність з точки зору основних параметрів (роботи при різних значеннях відношення потужності сигналу до шуму та точності отриманих даних) показують алгоритми, що працюють з м'якими рішеннями. Вони можуть працювати з повідомленнями, що початково мають велику кількість помилок та зменшувати їх кількість у ході ітерацій корекції на основі всіх отриманих даних.

Характер розташування значущих елементів значно впливає на складність реалізації декодеру. З цієї точки зору, особливої уваги заслуговують нерегулярні матриці, що при організації обчислень у декодері можуть спричиняти колізії доступу до пам'яті. Можливість виникнення таких колізій при паралельній реалізації призводить до її заміни на послідовну, внаслідок чого швидкодія частково паралельного декодеру суттєво зменшується. Уникнення колізій доступу в такому випадку стає основною проблемою для забезпечення коректності обробки даних у процесі корекції та декодування.

Аналіз останніх досліджень та публікацій

Структура LDPC-матриці впливає на реалізацію декодеру. Необхідність урахування розташування значущих елементів може значно ускладнити організацію декодеру. Тому важливим ресурсом для спрощення структури матриці є використання елементарних перетворень, основними з яких є перестановка рядків та стовпців матриці. У роботі [2] серед варіантів попередньої обробки для нерегулярних матриць з випадковою структурою пропонується використовувати перестановку рядків, проте алгоритму виконання перестановки не наведено. Можливість виконання таких операцій значною мірою залежить від алгоритму, що реалізується в декодері. Алгоритм мінімальної суми [1], що відноситься до алгоритмів з м'яким рішенням, передбачає отримання остаточного результату на основі обчислення суми повідомлень від вузлів перевірки (рядки матриці) вузлами значень (стовпці матриці) у випадку представлення матриці у вигляді графа Таннера. Враховуючи властивість комутативності операції додавання, для алгоритму мінімальної суми перетворення на основі перестановки рядків практично не впливають складність організації обчислень та отримання кінцевого результату відносно матриці без проведення перетворень. Відповідно, подібна попередня обробка матриці не має впливати і на складність декодеру.

Частково паралельним LDPC-декодерам присвячено велику кількість робіт, серед яких [3; 4], через те, що вони здатні забезпечити високу швидкодію при відносно простій реалізації. Однак більшість досліджень орієнтовані на конкретний тип матриць та не здатні працювати з іншими типами матриць.

Програмовані логічні інтегральні схеми (ПЛІС) використовуються для апаратної побудови LDPC-декодеру [5–7]. Врахування особливостей реалізації, які надають такі мікросхеми, також дозволяє організувати

обчислення для збільшення пропускної здатності декодеру. Проте можливості використання двопортової блокової пам'яті [8] при реалізації декодеру довільної LDPC-матриці не описані. Тому важливим ресурсом є використання характеристик апаратної реалізації елементів декодеру, серед яких основою виступає блокова пам'ять ПЛІС. Саме блокова пам'ять дозволяє зберігати кінцеві та проміжні результати декодування. На даний момент розвитку технологій блокова пам'ять ПЛІС дозволяє реалізувати два порти запису/зчитування. Тому важливим при розробці методу побудови декодеру є урахування даної обставини.

Метою роботи є підвищення швидкодії частково паралельного LDPC-декодеру за рахунок організації паралельних обчислень двох блоків декодування та зменшення використання ресурсів блокової пам'яті ПЛІС.

Основна частина

Матриця перевірки парності H LDPC-декодеру має M рядків та N стовпців. Максимальну кількість значущих елементів у окремому рядку матриці позначимо як n_{max}. Для представлення H при реалізації декодеру з метою збереження пам'яті доцільно зберігати лише значення індексів значущих елементів, а не всю матрицю. Визначення для такого представлення елементів матриці є наступним:

$$ind_{ij} = ord_i(1_j); i = 1...M; j = 1...n_{max}, \qquad (1)$$

де ord – функція, що повертає порядковий номер елементу в рядку.

Частково паралельний LDPC-декодер, робота якого організована на основі послідовної поелементної обробки повідомлень відповідно до індексів, що наявні в поточному рядку, забезпечує уникнення колізій доступу до пам'яті. Така реалізація передбачає наявність лише одного функціонального блоку, який відповідає за операції зчитування-обчислення-запису. Відповідно, використовується лише один порт пам'яті, до якого підключений обчислювальний блок. Це означає, що можливість реалізації другого порту не задіяна, а, значить, не використовуються ресурси для забезпечення максимальної пропускної здатності.

Наявність двох повноцінних портів запису/зчитування в блоковій пам'яті ПЛІС означає, що можлива робота двох обчислювальних блоків, що можуть працювати паралельно. Проте просте підключення другого блоку до пам'яті без попереднього аналізу розташування елементів та можливих перетворень може призвести до виконання некоректних обчислень через наявність колізій, у ході яких виконується обчислення для одного елементу.

Уникнення такого роду колізій можна забезпечити на основі попередньої обробки H. У випадку, якщо робота обчислювальних блоків організована таким чином, що кожен із них одночасно виконує обробку одного рядка, то для уникнення колізій потрібно забезпечити, щоб виконувалась наступна умова:

$$\forall ind_{i1j}, \forall ind_{i2j}, \neg\exists ind_{i1j} = ind_{i2j}, j = 1...n_{max},$$

(2)

де $i1$, $i2$ – номери рядків, що обробляються першим та другим обчислювальними блоками. Відсутність повторів індексів у двох рядках означатиме відсутність колізій доступу до пам'яті та при виконанні обчислень, що дозволяє проводити обробку одразу двох рядків, а, значить, підвищити швидкодію в два рази.

Для забезпечення виконання умови (2) при попередній обробці H пропонується виконувати наступні дії. Вважатимемо, що H не містить двох рядків, що мають два однакові значення індексів ind_{ij}. У протилежному випадку, це означає наявність у матриці циклів з довжиною 4, що говорить про недостатню придатність такої матриці для декодування через утворення поглинаючих множин. Також позначимо як m_{max} максимальну кількість значущих елементів, що наявні в одному стовпці матриці. Для попередньої обробки матриці використаємо лише один тип елементарних перетворень – перестановку рядків.

Реалізацію перестановки рядків для уникнення колізій будемо проводити за допомогою заповнення результуючої матриці, яка початково є порожньою, рядками вихідної матриці, що еквівалентно проведенню серії операцій перестановок рядків.

Для використання пропонується використовувати наступний алгоритм. Перший рядок вихідної матриці додається в першу позицію результуючої матриці. Перевіряється можливість додавання наступного рядка в наступну вільну позицію. Якщо немає можливості додати поточний рядок, то переходять до наступного. Подібна перевірка проводиться до моменту, поки не буде додано новий рядок. Індекс доданого рядка запам'ятовується та більше не використовується у ході додавання. Умовою завершення цього процесу є заповнення результуючою матриці рядками вихідною. При цьому, в найгіршому випадку можлива ситуація, коли m_{max}–1 рядків не вдається додати таким чином, через те, що вони всі містять індекс, що знаходиться в рядку, що був доданий перед цим. У такому випадку поточний рядок додається до таблиці, але для нього одразу шукається перестановка, яка дозволить зберегти умову неповторюваності індексів для сусідніх рядків, з якими виконується перестановка. Як тільки така перестановка знайдена, так само виконується перевірка можливості додавання наступних рядків, поки не будуть додані всі рядки. Псевдокод для отримання матриці представлено на рис. 1.

На початковому етапі заповнення матриці рядками контролюється умова неповторюваності лише відносно індексів попереднього рядка для рядка, який є поточним. Проте, як тільки вільні рядки завершуються, та необхідно використовувати перестановки в самій результуючій матриці, перевірка умов неповторюваності проводиться для обох суміжних рядків, з яким проводиться перестановка. У випадку, якщо за вказаним алгоритмом не вдається розмістити усі рядки, то їх обробку можна організувати лише для одного блоку декодування.

Варто також зазначити, що зміна послідовності появи індексів у рядку для уникнення колізій не може використовуватися, оскільки слід враховувати увесь цикл обробки – зчитування-обчислення-запис. Якщо при зчитуванні колізій не виникає, то виконання обчислень з подальшим записом даних означатиме втрату результату обчислення, що завершиться та буде записано першим. Отже, не буде врахована інформація, отримана від усіх вузлів.

```
function PERMUTEROWS(inputMatrix)
    newMatrix(1) ← inputMatrix(1)
    permut_ind ← 2
    start_ind ← 2
    while permut_ind < length(inputMatrix) do
        prev_row ← newMatrix(permut_ind − 1)
        cur_ind ← start_ind
        check ← true
        while check do
            temp_row ← inputMatrix(cur_ind)
            check ← hasSameElements(prev_row, temp_row)
            if check then
                newMatrix(permut_ind) ← temp_row
                permut_ind ← permut_ind + 1
                addToUsed(temp_row)
                start_ind ←= findNextStartInd()
                break
            else
                check ← true
                cur_ind ← cur_ind + 1
                if cur_ind == length(inputMatrix) then
                    newMatrix(permut_ind) ← temp_row
                    for i ← 1 to permut_ind − 1 do
                        permuteRows(inputMatrix, i, permut_ind)
                        check1 ← hasSameElements(i, i + 1)
                        if i ≠ 1 then
                            check2 ← hasSameElements(i, i − 1)
                        else
                            check2 ← true
                        end if
                        check3 ← hasSameElements(prev_row, prev_row − 1)
                        if check1 & check2 & check3 then
                            break
                        else
                            rollbackPermutation()
                        end if
                    end for
                end if
            end if
        end while
    end while
    return newMatrix
end function
```

Рис. 1. Псевдокод алгоритму перестановки рядків

Отримана матриця індексів при реалізації декодеру має бути описана як двопортова блокова пам'ять. Кожному стовпцю матриці в такому випадку відповідатиме окремий модуль пам'яті для збереження індексів. Два блоки декодування використовують окремі порти пам'ять для зчитування. При цьому робота блоків декодування має бути організована таким чином, щоб один блок відповідав за обробку парних рядків, а інший – непарних. Обидва блоки працюють паралельно, тому швидкість обробки усієї матриці збільшується в 2 рази. Доступ до елементів у пам'яті результуючого повідомлення також виконується з використання різних портів блокової пам'яті. Оскільки одночасно відбувається звернення за різними адресами, то вдається уникнути колізій.

Важливим моментом є також синхронізація паралельної роботи двох блоків декодування. Для коректної роботи пропонується наступний механізм синхронізації. За наявності двох модулів можлива ситуація при перевірці синдрому, що кількість помилок для одного блоку виявилась рівною нулю, в той час як при перевірці синдрому іншим блоком помилки були виявлені. В такому випадку слід забезпечити продовження роботи першого блоку, оскільки при подальшому декодуванні та корекції, може виявитися, що помилкові біти перемістились до рядків, що обробляються першим блоком. Тому синхронна робота та її закінчення для обох модулів є обов'язковими. Таким чином, умова завершення роботи декодеру є наступною:

$$finish = (iter_{current} = iter_{max}) \vee (error_1 = 0 \wedge error_2 = 0) \quad (3)$$

де $iter_{current}$ – номер поточної ітерації;

$iter_{max}$ – максимальна встановлена кількість ітерацій;

$error_1$, $error_2$ – кількість помилок перевірки парності, виявлених двома блоками декодування на поточній ітерації. Умова (3) є найбільш загальною, проте можуть використовуватися також інші показники з урахуванням додаткової специфіки реалізації.

Як зазначалось, при реалізації декодеру, стовпці H будуть представлені як двопортова блокова пам'ять. Такий підхід дозволяє отримувати доступ до індексів у рядках паралельно за 1 такт. Проте це означатиме необхідність великої кількості портів блокової пам'яті. Даний фактор негативно впливає на максимально можливу робочу тактову частоту, оскільки порт блокової пам'яті є складним у реалізації.

Специфіка частково паралельного декодеру, що досліджується, полягає в тому, що обробка даних на основі отриманих індексів відбувається поелементно.

Дана обставина дозволяє зменшити кількість окремих блоків пам'яті та об'єднати їх в один, організувавши послідовне зчитування індексів. При цьому швидкість роботи декодеру не зменшиться, оскільки обробка виконується поелементно. При використанні такої організації пам'яті індексів є можливість збільшити робочу тактову частоту пристрою.

Заміна окремих блоків пам'яті на один передбачає, що необхідно створити блокову пам'ять для наступної кількості елементів:

$$mem_{length} = 2^{\lfloor log_2 n_{max} * M \rfloor}. \quad (4)$$

Заповнення пам'яті індексів відбуватиметься таким чином:

$$ind_k = ind_{i=k/n_{max}, j=k \bmod n_{max}}, k = 1...mem_{length}. \quad (5)$$

У такому випадку два блоки декодування підключаються до двох портів пам'яті та у ході обробки зчитують послідовно n_{max} значень. Крім того, оскільки обробка даних також потребуватиме час на виконання, то випереджаюче зчитування під час обробки дозволить отримати усі індекси паралельно при наступному запиті.

Для функціонування декодеру важливою є розрядність при представленні даних м'яких рішень. Даний параметр також впливає на використання пам'яті при побудові декодеру. Вибір розрядності для представленні даних у ході обробки має базуватися на тому, при якій кількості помилок у вхідному повідомленні декодер зможе забезпечувати їх декодування, а також на обмеження відносно пам'яті.

На основі розробленого підходу реалізовано LDPC-декодер для матриць, що можуть містить до 16000 стовпців та до 8000 рядків із максимальною кількістю значущих елементів у рядку – 20. Для реалізації використано середовище Quartus II Web Edition. Розрахункова пропускна здатність декодеру збільшилась з 23,6 Мбіт/с до 47,2 Мбіт/с. Для збереження індексів реалізовано блок у складі декодеру з використанням 249 блоків пам'яті типу M9K. Це дозволило у 20 разів зменшити кількість портів блокової пам'яті. Проведено симуляцію декодування пристроєм після отримання даних з каналу з шумами (AWGN-канал). При симуляції використано розроблене програмне забезпечення на мові програмування Java (рис. 2).

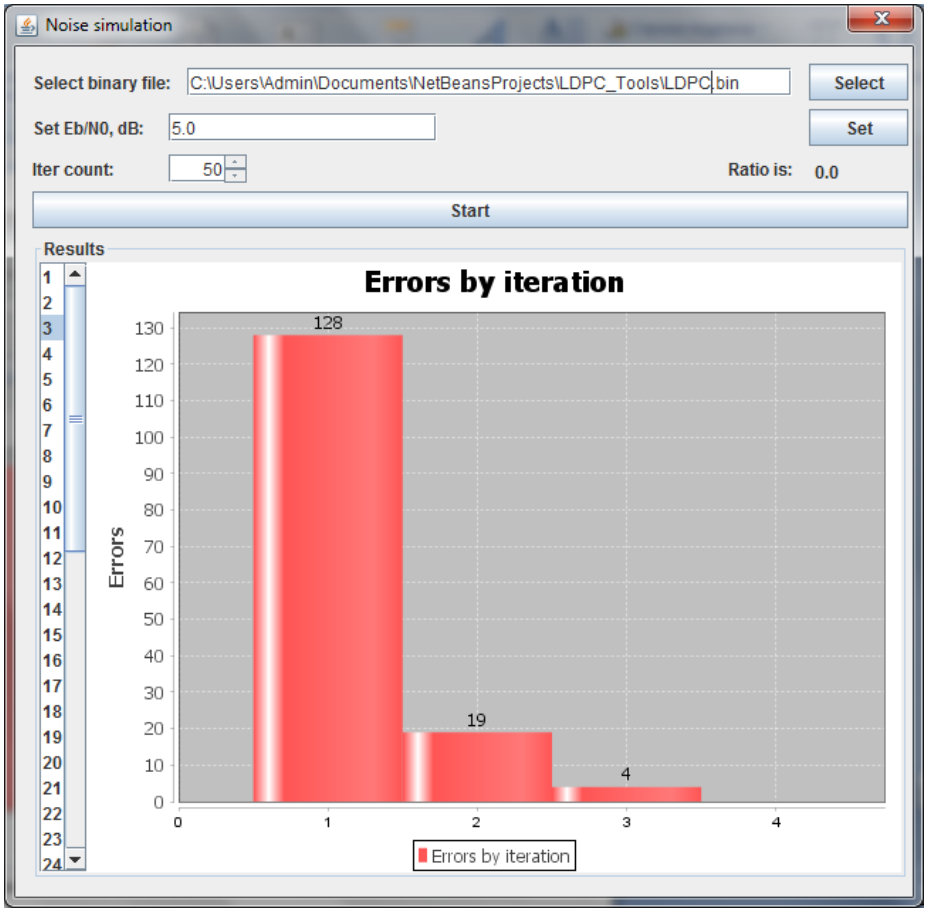

Рис. 2. Вікно програми проведення симуляції декодування

Тестування проводилось для декодеру з розрядністю від 4 до 8 бітів, у діапазоні відношення сигнал/шум 10-1,5 дБ. Досліджуваним параметром обрано відношення кількості рівнянь парності, що показали помилку після останньої ітерації декодування до загальної кількості рівнянь парності. При симуляції використовувався стандартний пакет коректних даних, на який накладали шум, а результат подавали для декодування декодеру. Для отримання результатів ця операція повторювалась протягом 100 ітерацій. Результати проведених досліджень представлено на рис. 3.

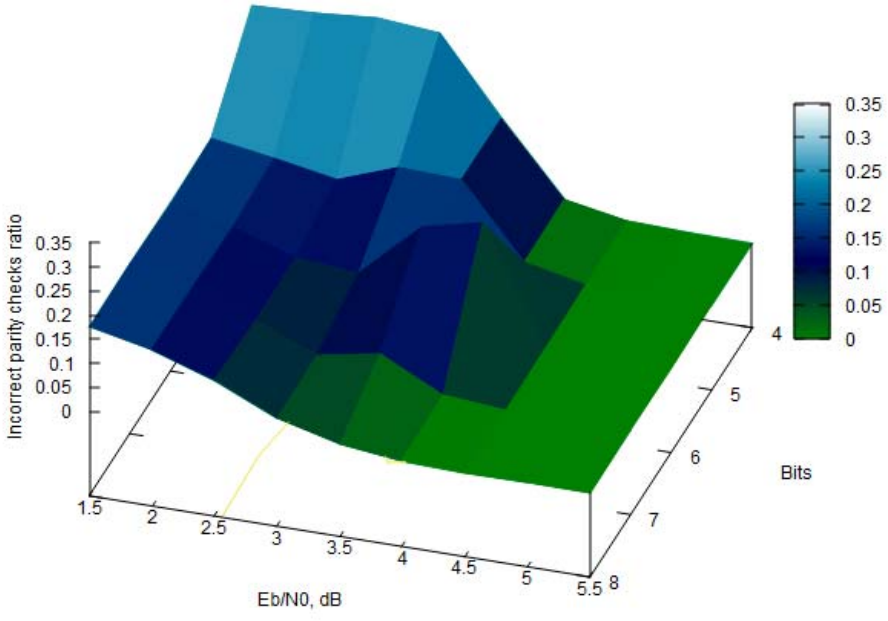

Рис. 3. Результати симуляції роботи декодеру

Отримані результати показали, що при низькому значенні співвідношення сигнал/шум, декодер з меншою розрядністю показує значно гірші результати. Проте за умови, що цей параметр не знижуватиметься нижче 5 дБ, розрядність у 4 біти є достатньою.

Висновки

Попередня обробка матриці перевірки парності на основі запропонованого алгоритму забезпечує уникнення колізій під час виконання циклу зчитування-обробка-запис при роботі декодеру. Запропонований у статті метод організації обчислень дозволяє використовувати одразу два блоки декодування, що працюють паралельно. Це дозволило збільшити пропускну здатність декодеру у 2 рази. Об'єднання пам'яті представлення індексів значущих елементів у один блок зменшило використання портів блокової пам'яті. Симуляція роботи декодеру дозволила обрати необхідну розрядність для представлення даних при відомому співвідношенні сигнал/шум.

ЛІТЕРАТУРА

1. Fossorier, M.P.C. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation / Fossorier, M. P. C., Mihaljević, M., Imai, H. // IEEE Transactions on Communications. – Volume 47, Issue 5. – 1999. – 673-680 pp.
2. Guilloud, F. Generic Architecture for LDPC Codes Decoding / By Frederic Guilloud // PhD Thesis, ENST Paris. – July, 2004. – 200 p.
3. Shirani-Mehr, H. A reduced routing network architecture for partial parallel LDPC decoders / Shirani-Mehr, H., Mohsenin, T., Baas, B. // 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR). – 2011. – 2192–2196 pp.
4. Zhang, Z. An Efficient 10GBASE-T Ethernet LDPC Decoder Design With Low Error Floors / Zhang, Z., Anantharam, V., Wainwright, M. J., Nikolic, B. // IEEE Journal of Solid-State Circuit. – Volume 45, Issue 4. – 2010. – 843–855 pp.
5. Chen, Y.-H. FPGA implementation and verification of LDPC minimum sum algorithm decoder with weight (3, 6) regular parity check matrix / Chen, Y.-H., Chu, C.-L., He, J.-S. // 2013 IEEE 11th International Conference on Electronic Measurement & Instruments (ICEMI). – Volume 2. – 2013. – 682–686 pp.
6. Boncalo, O. Cost-efficient FPGA layered LDPC decoder with serial AP-LLR processing / Boncalo, O., Amaricai, A., Hera, A., Savin, V. // 2014 24th International Conference on Field Programmable Logic and Applications (FPL). – 2014. – 1–6 pp.
7. Darabiha, A. A bit-serial approximate min-sum LDPC decoder and FPGA implementation / Darabiha, A., Carusone, A. C., Kschischang, F. R. // 2006 IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006. Proceedings. – 2006.
8. Internal Memory (RAM and ROM) User Guide [Електронний ресурс]. – Режим доступу : URL : http://www.altera.com/literature/ug/ug_ram_rom.pdf. – Загол. з екрану.

Крайнык Я. М., *Черноморский государственный университет имени Петра Могилы, г. Николаев, Украина*

ПОСТРОЕНИЕ ЧАСТИЧНО ПАРАЛЛЕЛЬНОГО LDPC-ДЕКОДЕРА С ИСПОЛЬЗОВАНИЕМ ОСОБЕННОСТЕЙ ДВУХПОРТОВОЙ БЛОЧНОЙ ПАМЯТИ ПЛИС

В статье проведено исследование проблемы построения частично параллельного LDPC-декодера для матрицы проверки парности с произвольным размещением значащих элементов. Представлен разработанный алгоритм для предварительной обработки матрицы на основе перестановки строк, который обеспечивает избегание коллизий. Предложена организация памяти сохранения индексов и ее взаимодействие с другими

компонентами декодера, в частности, с блоками декодирования. Использование особенностей блоковой памяти ПЛИС позволило обеспечить корректную параллельную работу двух таких блоков. На основе проведенных исследований реализовано LDPC-декодер, пропускная способность которого увеличена в два раза.

Ключевые слова: *LDPC-декодер; ПЛИС; блочная память; декодирование.*

Krainyk. Y., *Petro Mohyla Black Sea State University, Mykolaiv, Ukraine*

BUILDING PARTIALLY PARALLEL LDPC-DECODER WITH USAGE OF FPGA DUAL-PORT BLOCK MEMORY PECULIARITIES

Low density parity check (LDPC) codes provides significant results in forward error correction. LDPC-codes can be represented in form of sparse matrix. Structure of the matrix makes great influence on decoder implementation. Matrices with random significant element positioning are complex for decoder design. Decoder processing for this type of matrices is hardly to parallelize. Thus, the resulting throughput is lower than in case of special type of matrices that is suitable for parallel processing. However, they allow to get better correction capabilities and are closer to Shannon limit. The main problem during decoder design is avoiding access and processing collision in matrix processing. Partially parallel decoders are widely investigated and considered in large amount of scientific works. They provide balance between main characteristics of the decoder. Great throughput during appropriate complexity implementations makes them a clear choice for decoder design. Thus, this type of decoder is selected for implementation in the article.

The aim of the article is increasing throughput of partially parallel LDPC-decoder and to make consumption of FPGA block memory resources lower.

Avoiding collision in access and processing can be satisfied with parity check matrix preprocessing. Proposed algorithm of preprocessing leverages only row permutation operation. This means that symbol permutation in codeword is not required. Algorithm guaranties that there are no two sequential rows with one simultaneous index in the result matrix.

This condition makes possible to have two processing units. Limitation of two units is based on FPGA block memory peculiarity. Block memory can have only two ports for reading/writing even in state-of-the-art FPGA. These decoding units can work concurrently. Matrix processing in this case is performed by rows. The first decoding unit processes only odd rows and the second one process only even. Each block is connected to different port of memory. Preprocessed index memory provides different indices values for different decoding units. Access to result memory is performed with use of different ports. This fulfill avoiding collision requirements on every of read/process/write stages of processing.

Two concurrent decoding blocks double the throughput. However, block memory resource consumption affects on maximum work frequency of decoder. Ports of block memory in FPGA are very complex resources and require great amount of combinational logic for implementation. In case of partially parallel decoder that processes only single element in a row access to the index memory can be provided sequentially without lose in throughput. This can be organized as read-on-remand scheme or with lookup reading. As decoding unit logic requires only single element reading from the result memory only single index value is necessary on reading stage. This also means that index memory can be organized as single dual-port unit. Thus, the number of memory ports is decreased significantly.

According to the proposed approach LDPC-decoder was implemented. Selected environment for implementation is Quartus II Web Edition 13.1. Calculated throughput of implemented decoder is 47,2 Mbit/s. This is twice larger in compare with decoder with single processing unit. Index memory was organized as single block memory unit with dual-port. This makes it possible to decrease number of block memory ports in 20 times.

Parity check matrix preprocessing guaranties avoiding collision of access during all of read-process-write stage. Proposed algorithm for matrix preprocessing performs appropriate matrix construction with use of row permutation operation. Investigated method of computation organization allows having two processing blocks that work concurrently. This organization double throughput of the decoder. Union of index memory block into single unit reduces number of memory ports significantly. Therefore, overall complexity of decoder is reduced and it is possible to get larger maximum work frequency of the decoder.

Key words: *LDPC-decoder; FPGA; block memory; decoding.*

© Крайник Я. М., 2014 *Дата надходження статті до редколегії 16.12.2014 р.*