УДК 004.71

д.т.н., проф. **Мусієнко М.П.** (ЧорноморДУ, м. Миколаїв)
Крайник Я.М. (ЧорноморДУ, м. Миколаїв)

ПАРАЛЕЛЬНА РЕАЛІЗАЦІЯ АЛГОРИТМУ МІНІМАЛЬНОЇ СУМИ ДЛЯ LDPC-ДЕКОДЕРУ

Метою роботи є підвищення пропускної здатності LDPC-декодеру на основі паралельної організації обчислень відповідно до алгоритму мінімальної суми.

У роботі проводиться дослідження паралельної реалізації алгоритму м'якого декодування LDPC-кодів – алгоритму мінімальної суми. Представлена паралельна модель реалізації етапів декодування для частково паралельного LDPC-декодеру. Проведено дослідження параметрів, від яких залежить збільшення швидкодії, розроблена модель організації обчислень, а також виконана практична реалізація моделі для мікросхеми ПЛІС та проведено порівняння паралельної та послідовної моделей.

Ключові слова: LDPC-коди, алгоритм мінімальної суми, LDPC-декодер, паралельне виконання.

Вступ. Алгоритм мінімальної суми – алгоритм декодування для LDPC-кодів (Low Density Parity Check Codes – коди з низькою щільністю перевірки на парність), що

використовується для прямого виправлення помилок при передачі інформаційного повідомлення. Даний алгоритм відноситься до групи алгоритмів, що базуються на обміні повідомленнями (англ. message passing) між вузлами графа, побудованого на основі матриці перевірки парності. Він відноситься до алгоритмів, що забезпечують субоптимальне рішення. Операції, що використовуються в ході декодування, спрощені у порівнянні з алгоритмами, здатними забезпечити більшу точність декодування, наприклад, суми добутків. Тим не менш, програш у точності для даного алгоритму становить лише декілька десятих дБ відношення потужності сигналу та шуму, в той час, як складність апаратної реалізації є значно нижчою, через що він та його модифікації набули значного поширення при реалізації LDPC-декодерів.

Основними перевагами алгоритму мінімальної суми з точки зору апаратної реалізації є використання простих операцій порівняння та суми, а також великі можливості для реалізації паралельних обчислень. Алгоритм піддається розпаралелюванню як на рівні процесів (обчислення значень повідомлень, перевірка синдрому, виконання запису повідомлення і т.д.), так і на рівні простих операцій у складі процесів (виконання операцій зчитування/запису, обчислення значень повідомлень, виконання жорсткого декодування на основі обчислених елементів і т.д.). LDPC-декодери прийнято класифікувати за параметром паралельності виконання операцій в залежності від відображення вузлів графа Таннера на апаратну реалізацію на:

послідовні;
частково паралельні;
повністю паралельні.

Великого поширення набули частково паралельні декодери, що забезпечують часткове відображення апаратної реалізації на граф Таннера. Вони здатні забезпечувати збалансовані параметри швидкодії та використання апаратних ресурсів. Підвищення кількості операцій, що виконуються паралельно, для таких декодерів може значно підвищити пропускну здатність.

Огляд попередніх досліджень. У роботі [1] організація паралельних обчислень базується на реалізації конвеєру, в якому операції запису, обчислення та зчитування виконуються паралельно, проте для різних ітерацій. Недоліком даного підходу є необхідність залучення додаткових ресурсів пам'яті для організації паралельності операцій.

Автори роботи [2] досліджують архітектуру частково паралельного LDPC-декодеру, яка дозволяє розпаралелити виконання операцій на низькому рівні (дрібнозернистий паралелізм), проте розпаралелювання на більш високому рівні не розглядається.

Архітектура LDPC-декодеру, що представлена у [3], передбачає обчислення значень повідомлень вузлів перевірки на основі виконання паралельних порівнянь з використанням каскаду компараторів. Такий підхід за наявності всіх необхідних даних дозволяє проводити обчислення за мінімальний час, проте у випадку частково паралельного декодеру для отримання всіх даних необхідне очікування всіх значень, які можуть отримуватись послідовно, наприклад, у випадку використання нерегулярних LDPC-кодів. Це також має вплив на швидкодію декодеру. Тому можливий варіант, коли послідовна реалізація забезпечить кращу швидкодію, ніж паралельна.

Мета роботи. Метою даної роботи є підвищення пропускної здатності частково паралельного LDPC-декодеру за рахунок розробки моделі розпаралелювання виконання обчислень для алгоритму мінімальної суми.

Основна частина. Відповідно до алгоритму мінімальної суми послідовність дій, що виконується, є наступною (рис. 1).

Рис. 1. Загальна блок-схема для алгоритму мінімальної суми

Представлені стадії обчислень алгоритму мінімальної суми можуть бути реалізовані паралельно. Наприклад, обчислення синдрому та жорстке декодування, оскільки потребують однакових даних для виконання, тому їх реалізацію можна об'єднати.

Розглянемо детальніше процес виконання декодування для абстрактного декодера, що має паралельний доступ до пам'яті індексів значущих елементів та послідовний доступ до пам'яті повідомлення. Такий декодер може виконувати зчитування значень адрес елементів для декодування на поточному циклі ітерації як одну операцію. Проте, оскільки пам'ять повідомлення організована послідовно, то зчитування/запис елементів для даної пам'яті відбувається також послідовно. Це значно зменшує швидкодію декодеру. Крім того, зчитування/запис елементів на одному циклі для обчислення, перевірки синдрому та жорсткого декодування відбувається з використанням одних і тих самих адрес, що означає додаткове зменшення швидкодії за умови послідовної реалізації даних процесів.

Для підвищення швидкодії LDPC-декодеру розроблена модель організації роботи декодеру, яка передбачає паралельне виконання етапів обчислення повідомлень, обчислення синдрому та жорсткого декодування, а також обчислення значення повідомлень вузлів перевірки, реалізуючи таким чином принцип Single Instruction-Multiple Data (SIMD). На рис. 2 зображене представлення розробленої моделі.

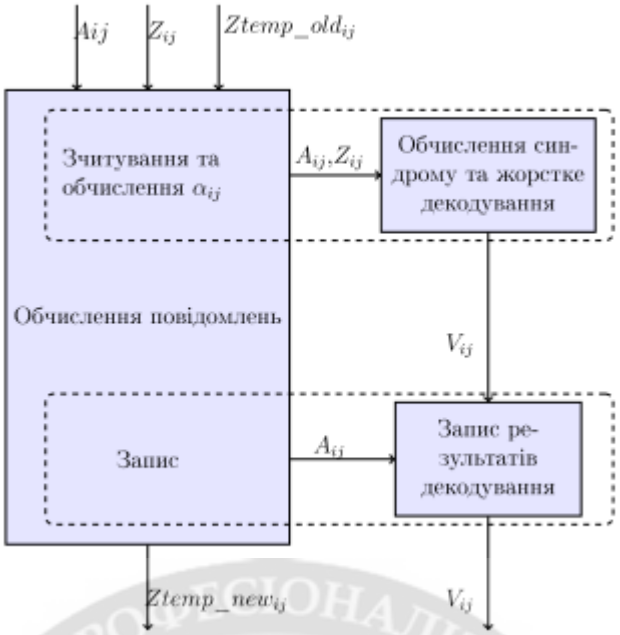

Рис. 2. Представлення моделі паралельного виконання операцій для LDPC-декодеру

Відповідно до рис. 2, до блоку виконання обчислень надходять дані про індекси значущих елементів A_{ij} (i – номер рядка, j – номер стовпця), за якими послідовно зчитуються значення результуючих Z_{ij} та проміжних $Ztemp\_old_{ij}$ обчислень. Паралельно зі зчитуванням значень Z_{ij} відбувається збір даних для перевірки синдрому для поточного рядка матриці перевірки парності, а також виконується жорстке декодування на основі знаку. Також на основі отриманих значень α_{ij} для попередньої ітерації та Z_{ij} проводиться обчислення значень вузлів перевірки для поточної ітерації. Після цього проводиться обчислення нових проміжних значень ітерації $Ztemp\_new_{ij}$. Далі слідує етап запису нових значень, паралельно з яким виконується також запис результатів декодування. При цьому, слід зазначити, що операції виконання перевірки синдрому та жорсткого декодування проводяться для даних попередньої ітерації обчислень, а не для поточної, оскільки остаточних даних про результати поточної ітерації немає. Таким чином, наприклад, на початку роботи декодеру проводиться перевірка синдрому та жорстке декодування для початкового вхідного повідомлення, паралельно з чим виконується обчислення нового повідомлення як результату першої ітерації. При виконанні другої ітерації обчислень проводиться перевірка синдрому та декодування результатів першої ітерації і т.д. Результати обчислень повідомлень для останньої ітерації не використовуються. При цьому для отримання результуючого повідомлення слід проводити додаткову ітерацію в порівнянні з послідовним підходом. Результатом додаткової ітерації є лише обчислення синдрому та жорстке декодування. Наприклад, якщо кількість ітерацій при послідовному підході становила 7, то для запропонованої моделі вона збільшується до 8.

Розглянемо детальніше, як відбувається обчислення значень перевірок для поточної ітерації. Відповідно до алгоритму мінімальної суми, значення перевірок для поточної ітерації визначаються наступним чином:

$$\alpha_{ij} = \prod_{j' \in V_i \setminus j} sign(\beta_{ij'}) \cdot \min(|\beta_{ij'}|), \qquad (1)$$

де V_i – множина вузлів значень для поточного вузла перевірки. У ході виконання зчитування до блоку обробки надходить інформація про значення α_{ij} і Z_{ij}, на основі яких обчислюються значення змінних β_{ij}. Подальшим етапом у обчисленні повідомлень є

обчислення значень перевірок відповідно до (1). Оскільки дані надходять послідовно і β_{ij} обчислюється послідовно, то обчислення α_{ij} можна також організувати послідовно. Для цього необхідно провести ініціалізацію початкового мінімального значення найбільшим можливим значенням, а початкового значення знаку як '+'. При надходженні наступного значення β_{ij} проводити порівняння з поточним мінімальним значенням та виконувати множення значень знаків. Таким чином, після подачі всіх значень отримаємо кінцеве значення α_{ij}.

Виконання обчислень відповідно до розробленої моделі потребує зчитування значень $A_{ij_{n\max}}$:

$$\begin{aligned} &i = 1...rows,\ j = 1...cols;\\ &n_{max} = \max_i(\text{count}_j(1)),\ j_{n\max} = 1...n_{max};\\ &A_{ij_{n\max}} = \text{read}(i). \end{aligned} \tag{3}$$

Далі виконується зчитування необхідних значень з інших блоків пам'яті за отриманими адресами та обробка даних:

$$\begin{aligned} &Z_j = \text{read}(A_{ij_{n\max}}),\ \ Z_{temp\_oldj} = \text{read}(A_{ij_{n\max}});\\ &\alpha_{ij}^{k-1} = \text{read}(A_{ij_{n\max}});\\ &V_j = \text{sign}(Z_j),\ \text{error} = \text{error} \oplus V_j;\\ &\beta_{ij} = Z_j - \alpha_{ij}^{k-1};\\ &\alpha_{ij}^{k} = \prod_{j' \in V_i \setminus j} \text{sign}(\beta_{ij'}) \cdot \min(|\beta_{ij'}|). \end{aligned} \tag{4}$$

Наступним етапом є перезапис тимчасових значень за ітераціями:

$$Z_{temp\_newj} = Z_{temp\_oldj} + \alpha_{ij}^{k}. \tag{5}$$

Останнім етапом, що може виконуватися паралельно є запис результаті до пам’яті:

$$\begin{cases} \text{write}(Z_{temp\_newj},\ A_{ij_{n\max}});\\ \text{write}(V_j,\ A_{ij_{n\max}}). \end{cases} \tag{6}$$

У (2) паралельне виконання операцій за наявності необхідних даних показано за допомогою фігурних дужок.

Проведемо порівняння пропускної здатності LDPC-декодеру для послідовної та паралельної моделей. У випадку послідовної моделі за обчисленням повідомлення слідує перевірка синдрому та жорстке декодування. У представленій паралельній моделі ці операції здійснюються одночасно. Для оцінки пропускної здатності декодера скористаємось наступним виразом [4]:

$$T = \frac{F \cdot L}{I \cdot N}, \tag{7}$$

де T – пропускна здатність декодеру, Мбіт/с; F – тактова частота роботи декодеру, МГц; L – довжина повідомлення, біт; I – кількість циклів декодування; N – кількість ітерацій декодування.

Введемо позначення для кількості операцій на проведення обчислення повідомлень для одного циклу ітерації m, кількості операцій на перевірку синдрому для одного циклу ітерації – s. Тоді з урахуванням того, що паралельна модель потребує проведення однієї додаткової ітерації, обчислення швидкодії для послідовної та паралельної моделей можна проводити наступним чином:

$$T_{sequential} = \frac{F \cdot L}{(m+s) \cdot I \cdot N}, \quad (8)$$

$$T_{parallel} = \frac{F \cdot L}{m \cdot I \cdot (N+1)}. \quad (9)$$

На основі (8) та (9) визначимо відносне збільшення пропускної здатності для паралельної моделі:

$$E = \frac{T_{parallel}}{T_{sequential}} = \frac{m+s}{m} \cdot \frac{N}{N+1}. \quad (10)$$

Визначимо значення границі для (10) за умови, що $N \to \infty$:

$$\lim_{N \to \infty} E = \frac{m+s}{m}. \quad (11)$$

Однак кількість ітерацій для практичного застосування часто обмежують певним значенням, тому множник, що містить значення кількості ітерацій, у такому випадку матиме значний вплив на показник E.

Для проведення практичного порівняння моделей реалізовано 2 LDPC-декодери. Реалізація виконана на мові схемотехнічного опису VHDL у середовищі ISE Design Suite WebPack 14.7 для мікросхеми System-on-Chip (SoC) сімейства Zynq-7020. Робоча тактова частота обох декодерів становила 250 МГц. Довжина вхідного повідомлення для декодування L становила 1000, кількість циклів для однієї ітерації I – 500, а параметри m і s дорівнювали 43 та 17 відповідно. Максимальна кількість ітерацій обмежена для послідовної моделі 10, а для паралельної – 11 ітераціями. Отримані результати швидкодії наведені на рис. 3.

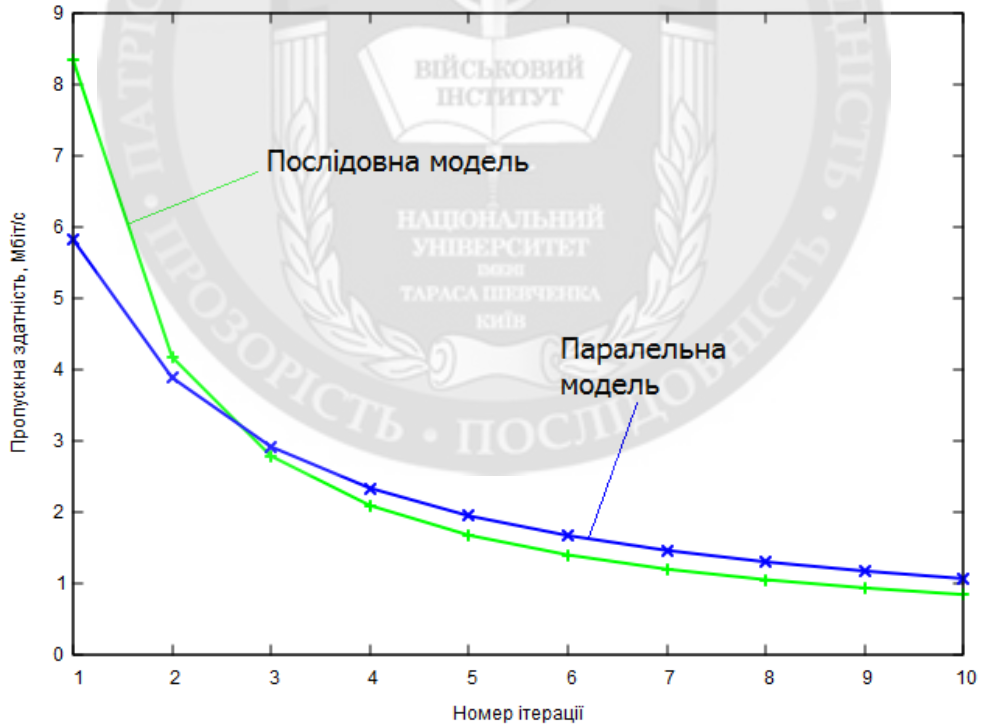

Рис. 3. Розрахункова пропускна здатність декодерів за ітераціями

Слід зазначити, що для порівняння перша ітерація для паралельного декодеру не використовувалась, оскільки для неї немає відповідного номеру для послідовного декодеру. Видача результату на першій операції для паралельного декодеру означатиме, що перевірка синдрому не виявила помилок парності та декодоване повідомлення доступне для зчитування. Значення пропускної здатності в такому випадку становить 11,63 Мбіт/с. Графік параметру E для розроблених декодерів наведено на рис. 4.

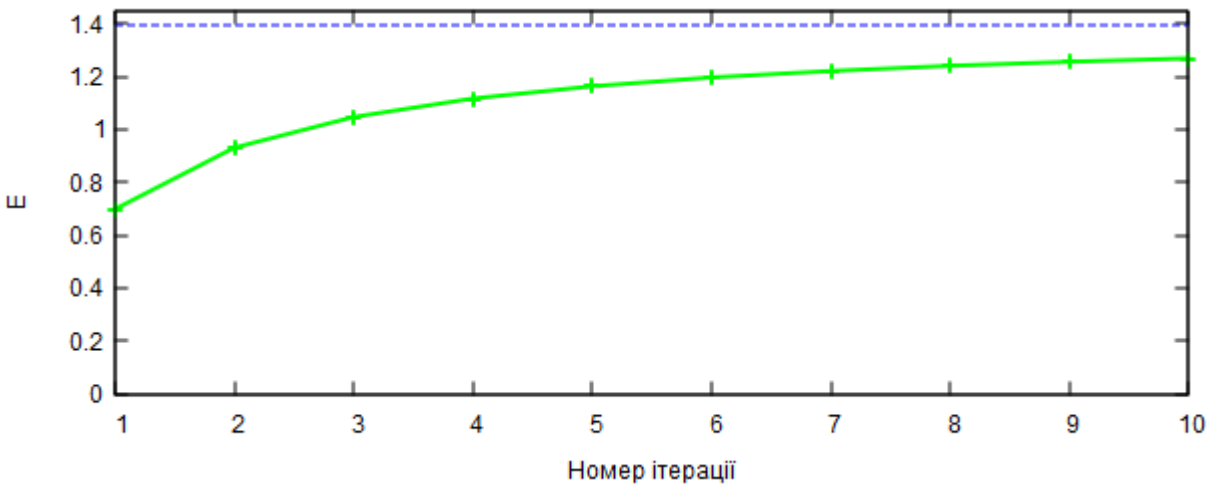

Рис. 4. Відносне збільшення пропускної здатності для паралельної реалізації декодеру

Відповідно до (11) граничне значення E становить 1,39. Таким чином, зі збільшенням кількості ітерацій, що може бути пов'язано, наприклад, з підвищенням рівня шуму в каналі, відносна пропускна здатність для запропонованої паралельної моделі буде збільшуватись.

Крім того, реалізована модель LDPC-декодеру з послідовним обчисленням значення α_{ij} при зчитуванні даних. Реалізація дозволила зменшити значення параметру m до 38, у порівнянні з 43 для паралельної організації обчислень. Циклограма розробленого модулю з прикладом роботи представлена на рис. 5.

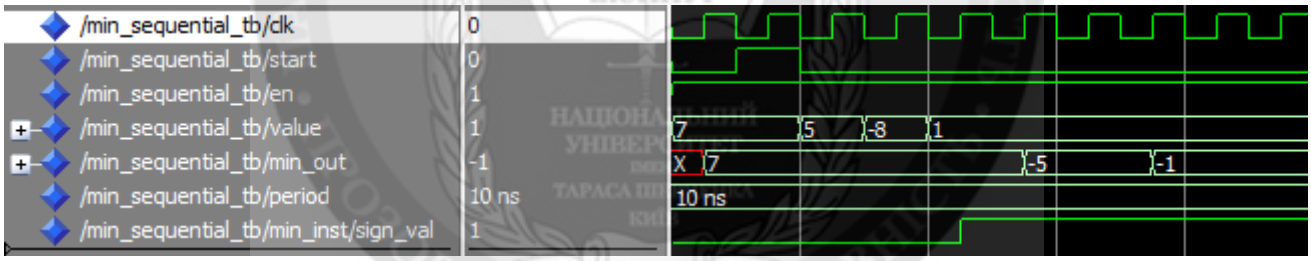

Рис. 5. Приклад роботи модуля послідовного обчислення α_{ij}

Для прикладу, представленого на рис. 5, тестування проводилося з 4-бітними даними.

Варто також зауважити, що реалізація представленої моделі для ПЛІС, потребуватиме виділення додаткового блоку пам'яті для збереження результатів ітерації $Memory_Z$. Це пов'язано з тим, що в ході запису оновлених проміжних значень необхідне виконання перезапису з пам'яті $Memory_{Ztemp}$ до пам'яті $Memory_Z$. Проте, до кінця обчислення для поточної ітерації вміст даної пам'яті не має змінюватися, оскільки це змінить результати перевірки синдрому. Саме тому, зчитування відбувається з одного блоку пам'яті, а запис вже в інший. Для кожної наступної ітерації функції даних блоків почергово змінюються. Для реалізованих декодерів різниця у використанні блоків пам'яті виявилася незначною: знадобився один додатковий блок пам'яті RAM18 на 18 кбіт. Порівняння використання основних ресурсів для обох реалізацій наведено в табл. 1.

Таблиця 1

Використання основних ресурсів для реалізацій декодерів

Назва ресурсу	Паралельна реалізація		Послідовна реалізація	
	Використано, од.	Використано, %	Використано, од.	Використано, %
Регістрів	1415	1	1529	1
Таблиць LUT	843	1	934	1
Блоків пам'яті RAM18	19	6	18	6

Висновки. Розроблена модель паралельного виконання операцій для алгоритму декодування мінімальної суми для LDPC-кодів з паралельним виконанням операцій обчислення значень повідомлень, перевірки синдрому і жорсткого декодування, дозволила підвищити швидкодію LDPC-декодеру. Проведена практична реалізація та порівняння пропускної здатності для паралельної та послідовної моделей, які показали, що відносне збільшення швидкодії підвищується відповідно до кількості ітерацій.

ЛІТЕРАТУРА:

1. Radosavljevic, P., Baynast, A., Karkooti, M., Cavallaro, J.R. MULTI-RATE HIGH-THROUGHPUT LDPC DECODER: TRADEOFF ANALYSIS BETWEEN DECODING THROUGHPUT AND AREA The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06).

2. Zhong, H., Xu, W., Xie, N., Zhang, T. Area-Efficient Min-Sum Decoder Design for High-Rate QC-LDPC Codes in Magnetic Recording, Magnetics, IEEE Transactions on Volume 43, Issue 12, Dec. 2007, 4117 - 4122 pp.

3. Mohsenin, T. Algorithms and Architectures for Efficient Low Density Parity Check (LDPC) Decoder Hardware / By Tinoosh Mohsenin // Submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering. – 2010.

4. Zhang, K., Huang, X., Wang, Z. High-Throughput Layered Decoder Implementation for Quasi-Cyclic LDPC Codes IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 27, NO. 6, AUGUST 2009.

Рецензент: д.т.н., проф. Кутковецький В.Я., професор кафедри інформаційних технологій та програмних систем Чорноморського державного університету імені Петра Могили

д.т.н., проф. Мусиенко М.П., Крайник Я.М.

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА МИНИМАЛЬНОЙ СУММЫ ДЛЯ LDPC-ДЕКОДЕРА

Целью данной работы было повышение пропускной способности LDPC-декодера на основе параллельной организации вычислений в соответствии с алгоритмом минимальной суммы.

В данной работе проводится исследование параллельной реализации алгоритма мягкого декодирование LDPC-кодов – алгоритма минимальной суммы. Представлена параллельная модель реализации этапов декодирования для частично параллельного LDPC-декодера. Проведено исследование параметров, от которых зависит увеличение быстродействия, разработана модель организации вычислений, а также выполнена практическая реализация модели для микросхемы ПЛИС и проведено сравнение параллельной и последовательной моделей.

Ключевые слова: LDPC-коды, алгоритм минимальной суммы, LDPC-декодер, параллельное выполнение.

prof. Musiyenko M.P., Krainyk Ya.M.

MINIMAL SUM ALGORITHM PARALLEL IMPLEMENTATION FOR LDPC-DECODER

The aim of the article was rising up throughput of LDPC-decoder. It is based on parallel computation organization according to minimal sum algorithm.

This work investigates minimal sum algorithm that is one of the soft decoding algorithms. Parallel model of decoding stages implementation is provided for partial parallel decoder case. Parameters that have influence on throughput are researched. Computation organization model is designed. The model is implemented with use of FPGA integrated circuit and comparison between parallel and sequential model is performed.

Keywords: LDPC-codes, minimal sum algorithm, LDPC-decoder, parallel execution.