УДК 004.312.26:519.725
Крайник Я.М., Перов В.О.
Чорноморський національний університет імені Петра Могили

ОРГАНІЗАЦІЯ СКІНЧЕНОГО АВТОМАТУ УПРАВЛІННЯ ПРОЦЕСОМ ДЕКОДУВАННЯ ДЛЯ ДЕКОДЕРУ TURBO-PRODUCT-КОДІВ НА БАЗІ FPGA

Крайник Я.М., Перов В.О. Організація скінченого автомату управління процесом декодування для декодеру Turbo-Product-кодів на базі FPGA. У даній статті запропонований метод організації скінченого автомату для управління процесом декодування Turbo-Product-кодів на базі FPGA. Результуючий скінчений автомат використовує діапазони значень лічильника для реалізації під-станів скінченого автомату. Запропонований метод організації автомату для управління декодером дозволяє підвищити пропускну здатність декодеру, оскільки виконує обробку вхідних даних відповідно до довжини кодів за необхідну кількість тактів.
Ключові слова: скінчений автомат, Turbo-Product-коди, реконфігуровний декодер, діапазони, періоди активності.

Крайнык Я.М., Перов В.О. Организация конечного автомата управления процессом декодирования для декодера Turbo-Product-кодов на базе FPGA. В данной статье предложен метод организации конечного автомата для управления процессом декодирования Turbo-Product-кодов на базе FPGA. Результирующий конечный автомат использует диапазоны значений счетчика для реализации под-состояний конечного автомата. Предложенный метод организации автомата для управления декодером позволяет повысить пропускную способность декодера, поскольку выполняет обработку входных данных в соответствии с длиной кода за необходимое количество тактов.
Ключевые слова: конечный автомат, Turbo-Product-коды, реконфигурируемый декодер, диапазоны, периоды активности.

Krainyk Y.M., Perov V.O. Organization of Finite-State Machine for Decoding Process Control in Turbo-Product-Code Decoder based on FPGA. In this article method for organization of finite-state machine for decoding process control in Turbo-Product-code decoder based on FPGA has been proposed. The finite-state machine utilizes ranges of counter values to implement sub-states. The devised finite-state machine organization method allows improving in decoder throughput, as it performs processing of input data according to the code length and uses necessary number of clock pulses.
Keywords: Finite-State Machine, Turbo-Product-codes, reconfigurable decoder, ranges, activation periods.

Завадостійкі коди є ключовим компонентом систем високошвидкісної передачі інформації. Саме від властивостей таких кодів залежить, наскільки надійною буде передача даних та яка пропускна здатність можлива для реалізації певного інтерфейсу. Серед корегуючих кодів наявні коди, які забезпечують різні характеристики щодо корегуючої здатності. Їх вибір для використання у системі залежить від заданих критеріїв, що оцінюють середовище передачі сигналу та вплив зовнішніх факторів на даний процес. Коди з низькою щільністю перевірки на парність, коди Хеммінга, турбо-коди (англ. Turbo-Product Codes - TPC) демонструють різну пропускну здатність і відрізняються за складністю реалізації та вимогами щодо апаратних ресурсів. Тому кінцевий вибір для їх використання у системах передачі інформації у більшості випадків є компромісом між характеристиками коду та доступними ресурсами.

Програмовні логічні інтегральні схеми (ПЛІС, англ. Field-Programmable Gate Array - FPGA) є основним засобом для відлагодження роботи інтерфейсу передачі даних. Подальшим кроком для створення готових пристроїв є перенесення протестованого рішення на спеціалізовані мікросхеми. З іншого боку, використання ПЛІС у кінцевому пристрої може мати свої переваги, які полягають у можливості зміни налаштувань, удосконаленні початкової розробки, і, навіть, повній її переробці. З точки зору використання ПЛІС у системах передачі інформації, то реалізований з їх допомогою функціонал повинен забезпечувати певний резерв щодо пропускної здатності, відповідно, мати ресурси щодо розширення засобів для передачі даних. Тому при проектуванні рішення для системи високошвидкісної передачі інформації, яка базується на мікросхемах ПЛІС, необхідно передбачити використання мікросхеми, яка здатна забезпечити нові вимоги протягом кількох циклів розробки системи.

Декодер завадостійких кодів є ключовим компонентом у системі передачі інформації, оскільки саме на нього припадає найбільша кількість операцій по обробці даних. Реалізація апаратного декодування Turbo-Product-кодів (TP-кодів) [1] передбачає наявність кількох стадій декодування: запис вхідних даних, декодування рядків, декодування стовпців, видача результати та ін. проміжні стадії. Керування виконанням окремих стадій доцільно організувати з використанням технік побудов скінченого автомату. Керуючий автомат для декодеру, який може працювати з кодами довільної довжини без необхідності додаткового перепрограмування

(реконфігуровний декодер) повинен забезпечувати обробку вхідних даних на всіх стадіях декодування і при цьому виконувати кожну окрему стадію за мінімально необхідну кількість тактів. Існуючі рішення [2], в основному, орієнтовані на обробку одного коду або кодів, які мінімально відрізняються між собою. Тому проблема побудови скінченого автомата для управління процесом декодування ТР-кодів є актуальною та важливою для задачі реалізації реконфігуровного декодеру ТР-кодів на базі програмовних логічних інтегральних схем (ПЛІС) [3, 4].

Проблемі декодування ТР-кодів присвячена велика кількість наукових досліджень, серед яких [5-8]. Зазвичай, у цих роботах акцент робиться на алгоритмах декодування. При цьому, складність обчислень, які необхідно виконувати, не оцінюється відносно переносу на конкретну платформу для реалізації. Тим не менш, це є ключовим питанням при реалізації алгоритмів для FPGA, оскільки дані мікросхеми мають особливості, які не можуть бути не враховані (обмежений обсяг внутрішньої пам'яті, відсутність або невелика кількість обчислювачів, що дозволяють проводити операції з плаваючою точкою та ін.).

Ключовим параметром, який характеризує роботу декодеру завадостійких кодів, є пропускна здатність. Пропускна здатність реконфігуровного декодеру в залежності від того, як орагізоване управління процесом декодування може значно відрізнятись для різних кодів. У відомих дослідженнях обробці кодів з різною структурою присвячено недостатньо уваги, тому це питання потребує подальших досліджень і є актуальним.

Виклад основного матеріалу. Для реалізації керуючих блоків у складі FPGA для систем різного типу найбільш поширеним варіантом є реалізація на основі скінченого автомату (англ. Finite State Machine - FSM). Саме з використанням даного підходу має бути розроблений блок керування блоками декодування всередині декодеру. Проте, у наукових джерелах [5-8], наводиться опис архітектури декодеру без уточнень з приводу того, яким чином здійснюється керування. Даний момент є важливим, оскільки декодер повинен забезпечувати достатній рівень паралельності процесу декодування для отримання необхідної пропускної здатності. Тому невирішеною є проблема організації скінченого автомату для керування процесом декодування ТР-кодів.

Скінчений автомат передбачає керування декодером [3, 4], який має архітектуру, яка в загальному вигляді представлена на рис. 1. Декодер містить блоки пам'яті (вхідний, вихідний та проміжний), пристрої зсуву для забезпечення коректної обробки рядків та колонок, безпосередньо самі блоки декодування.

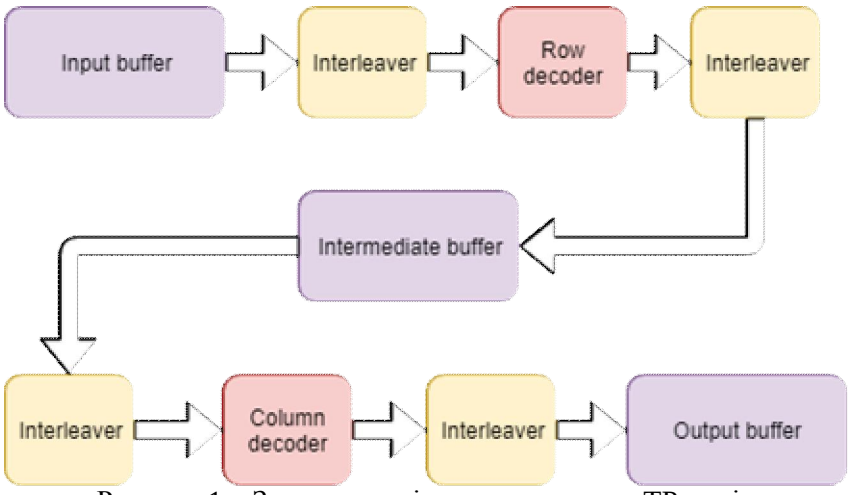

Рисунок 1 – Загальна архітектура декодеру ТР-кодів

Для демонстраційних цілей деякі зв'язки між блоками у декодері були опущені. Наприклад, на другій та наступних ітераціях зчитування відбувається не з вхідного буферу, а з додаткової проміжної пам'яті.

Основними вимогами до розроблюваного автомату є простота та гнучкість налаштувань. Тобто, він повинен містити якомога менше станів, але, при цьому, забезпечувати можливість контролю для усіх стадій декодування. Розроблений автомат представлено на рис. 2.

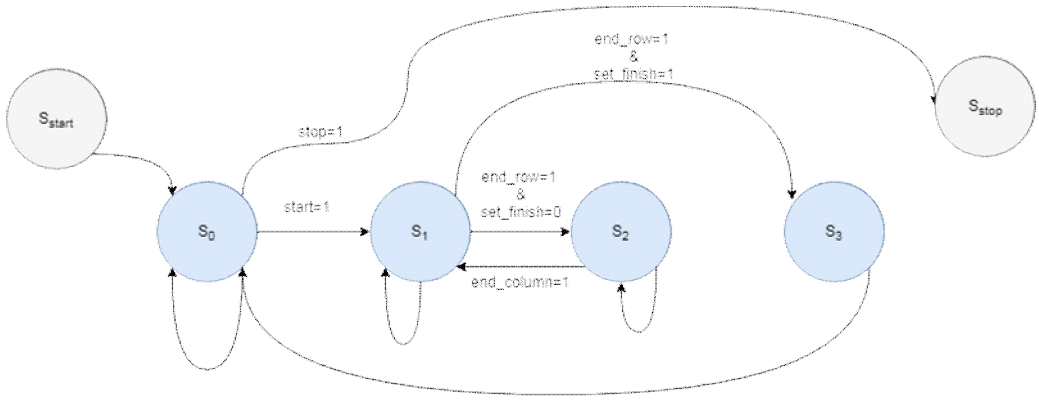

Рисунок 2 – Граф станів скінченого автомату для декодеру ТР-кодів

Особливістю автомату є те, що вихід з процесу декодування і перехід до видачі результату відбувається зі стану S_1. Це пов'язано з тим, що запис у вихідну пам'ять необхідно виконувати лише у випадку, якщо значення змінної set_finish дорівнює "1". На момент виконання другої пів-ітерації це не може бути відомим. Тому використовується підхід, коли для перезапису використовується ще одна додаткова перша пів-ітерація.

Автомат складається з 4 основних станів, а також двох службових (запуску та виходу). Під час перебування пристрою у основних станах забезпечується наступна функціональність:

– S_0 – очікування надходження сигналу запуску декодування у той момент, коли дані для декодування готові; також вихід з процесу декодування;

– S_1 – декодування рядків у матриці вхідних даних;

– S_2 – декодування стовпців у матриці вхідних даних;

– S_3 – завершення процесу декодування, видача сигналу про готовність оброблених даних до видачі.

Обробка вхідних даних за ітераціями декодування забезпечується за рахунок наявності взаємних переходів між станами S_1 та S_2. Обробка кількох пакетів забезпечується наявністю переходу між станами S_1 та S_3. Оскільки декодер є реконфігуровним, то передбачена також можливість виходу (S_{stop}) з робочого циклу та зупинка роботи декодеру для встановлення нових параметрів роботи декодеру. У даний момент часу декодер не може приймати вхідні дані або виконувати будь-яку обробку даних в основному циклі, оскільки це призведе до отримання неправильного результату.

У автоматі керування декодером для виділення під-станів використовується значення лічильника, який працює постійно під час роботи декодеру. Подібний підхід дозволяє, по-перше, забезпечити максимальну гнучкість налаштувань по управлінню керуючими імпульсами, по-друге, залучити для цього мінімальну кількість додаткових ресурсів – використовується лише регістр для значення лічильника. Кожному під-стану відповідає діапазон значень лічильника. Розбиття на діапазони відбувається для таких виділених стадій декодування (послідовність застосовується і для декодування рядків і стовпчиків):

- зчитування даних з пам'яті;
- передача даних на регістр зсуву;
- безпосередньо декодування;
- передача даних для зворотного зсуву;
- запис даних.

Усі зазначені стадії пов'язані таким чином, що вихід попередньої стадії є входом для наступної. Для ввімкнення відповідного блоку на нього подається сигнал активації (сигнал enable). Коли сигналу активації немає, то блок не є активним і не виконує жодних операцій. Відповідно, для коректної роботи необхідно узгодити готовність даних на виходах різних блоків та активацію наступних. Декілька блоків можуть бути активними одночасно.

Розподіл значень лічильника на діапазони відбувається відповідно до наступних правил. Введемо наступні позначення для представлення процесу формування діапазонів:

- T – загальна кількість діапазонів;

- $t = \overline{1,T}$;
- $\min_t$ – ліва границя діапазону;
- $\max_t$ – права границя діапазону;
- $[\min_t, \max_t]$ – представлення діапазону t.

Ліва границя першого діапазону має значення 0, а права границя останнього діапазону $t = T$ має значення $\max\_count$. Права границя попереднього діапазону та ліва границя наступного пов'язані між собою наступним співвідношенням:

$$\min_t = \max_{t-1} + 1. \quad (1)$$

Відповідно, діапазони не мають перетинатися між собою. Наступні параметри впливають на визначення значень діапазонів:

1. Довжина найдовшого коду n_c у групу кодів (у випадку рядків це буде довжина коду стовпців, а для стовпців – довжина коду рядків, оскільки саме таку кількість векторів необхідно обробити протягом однієї пів-ітерації).
2. Затримка для блоку обробки між отриманням даних та видачею результату (l_{block}).

Далі розглянемо формування діапазонів значень лічильника на основі періодів активностей. Діапазони визначаються таким чином, що стан усіх сигналів активації блоків є стабільним і не змінюється. Зміна стану одного з таких сигналів означає початок нового діапазону.

У загальному випадку кожному блоку для обробки та видачі результатів необхідна наступна кількість тактів

$$p_{block} = n_c + l_{block}. \quad (2)$$

Коли відомою є загальна кількість тактів, що витрачає блок на повне опрацювання і видачу результату, стає можливим узгодження періодів активностей блоків. Дані періоди є ключовими у формуванні границь діапазонів лічильника. Періоди активності блоків можуть належати різним суміжним діапазонам.

Розглянемо детальніше періоди активностей блоків та їх взаємозв'язок між собою. На початку має виконуватись стадія зчитування даних з вхідного буферу. Активація блоку наступної стадії відбувається у момент, коли з'являються перші результати обробки на виході попереднього блоку. Відповідно ліва границя другого діапазону буде обчислена як

$$\min_2 = l_{block1}. \quad (3)$$

Після цього моменту перший блок залишається активним ще протягом n_c тактів для того, щоб повністю передати результат у наступний блок. Також з цього можна зробити висновок про те, що права границя першого діапазону буде за значенням відповідати затримці, яка є на першому блоці:

$$\max_1 = l_{block1} - 1. \quad (4)$$

З наведених вище виразів можна визначити лише три границі для перших двох діапазонів (повністю вказати перший діапазон, а також ліву границю другого) для архітектури декодеру, яка передбачає послідовне з'єднання блоків між собою. Для визначення наступних границь необхідно також враховувати параметр n_c. Оскільки він дозволяє визначити, через яку кількість тактів має бути деактивований сигнал для попереднього блоку. До того, моменту, коли кількість поданих тактових імпульсів не перевищує значення p_{block1}, границі діапазонів можна визначити лише на основі затримок блоків обробки:

$$\min_{t+1} = \sum_t l_{blockt}, \quad (5)$$

$$\max_t = \sum_t l_{blockt} - 1. \quad (6)$$

Загальна кількість тактів, яка знадобиться для виконання однієї пів-ітерації визначиться як

$$\max\_count = \sum_t l_{blockt} + n_c \quad (7)$$

Після того, як кількість поданих тактів перевищила p_{block1}, як мінімум один блок має бути деактивований, що означає необхідність нового діапазону лічильника.

Розглянемо формування діапазонів для під-станів автомату на прикладі декодеру, виконання пів-ітерації у якому забезпечується п'ятьма блоками, і представленому на рис. 3.

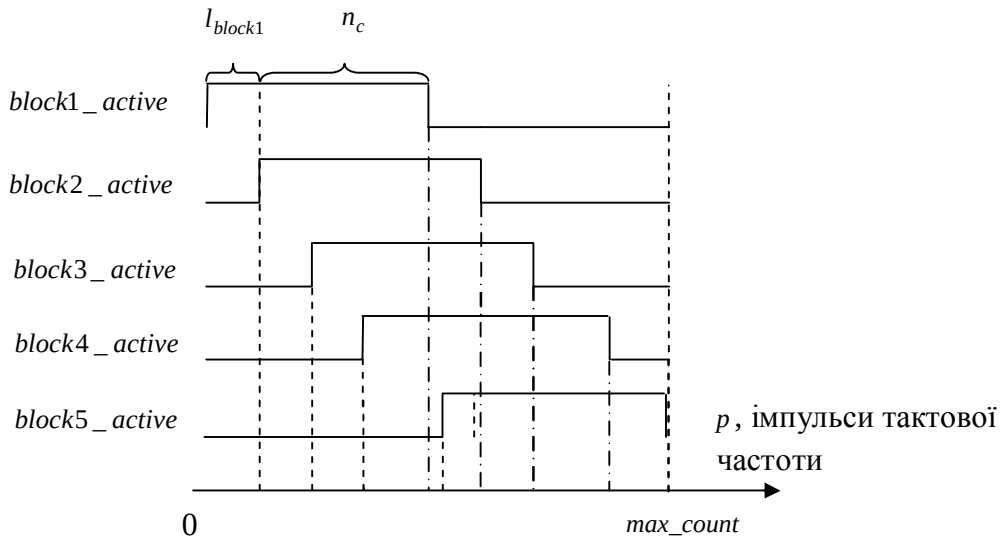

Рисунок 3 – Приклад діаграми сигналів активації блоків декодування

З даної діаграми значень сигналів підтверджується коректність виразу (7) для випадку, коли блоки з'єднані послідовно. Звертає на себе увагу той факт, що затримка останнього блоку у такому випадку не спричиняє формування нового діапазону. Крім того, на діаграмі чітко представлено розбиття на діапазони на основі періодів активностей блоків, що дозволяє визначити під-стани для запропонованого скінченого автомату. Робота реконфігуровного декодеру при обробці різних кодів забезпечується таким чином, що виконується обчислення діапазонів для всіх груп кодів, для яких планується обробка даних.

На основі вхідних сигналів, які дозволяють обрати код, якому відповідає вхідна матриця даних, визначається максимальне значення, до якого необхідно вести рахунок – *max_count* . Зміна значення лічильника та відповідність діапазонам для представлення під-станів показана на рис. 4.

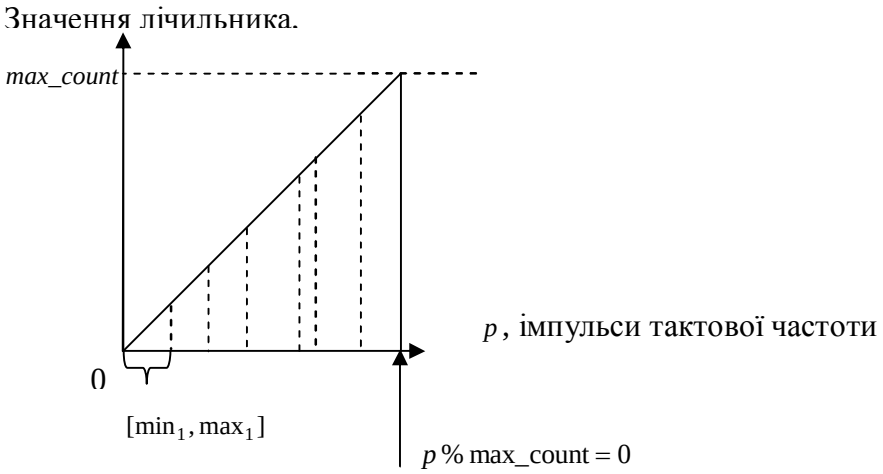

Рисунок 4 – Формування діапазонів значень лічильника

З кожним тактовим імпульсом значення лічильника збільшується на одиницю до того моменту, поки не досягне максимального значення *max_count* . Після цього значення обнулюється і рахунок починається спочатку.

На попередньому рисунку показано, як виглядає розподіл діапазонів для однієї пів-ітерації. На рис. 5 показано, як виглядає процес зміни лічильника протягом усього процесу декодування з акцентом на першу та останню пів-ітерації.

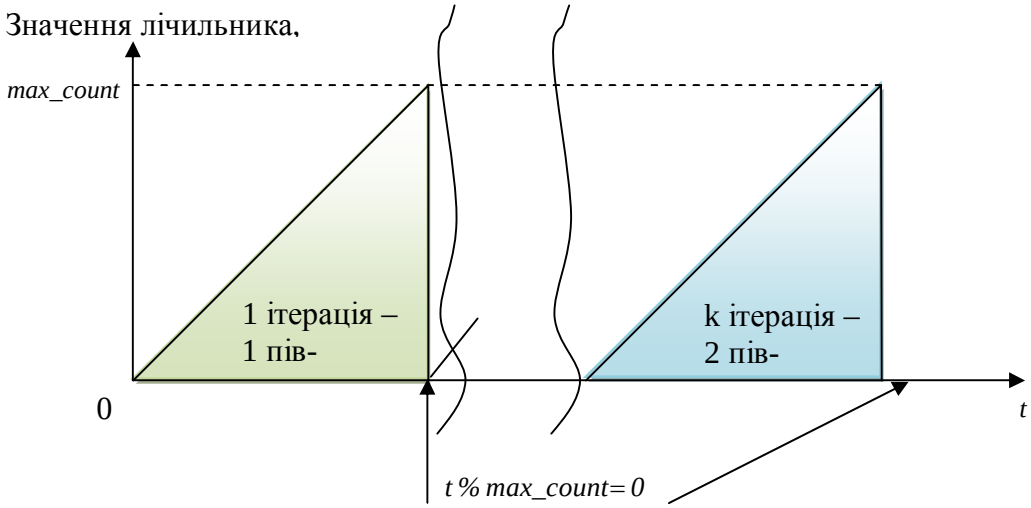

Рисунок 5 – Графік зміни значень лічильника протягом усього процесу декодування (для кодів, що відносяться до однієї групи)

На рисунку показаний випадок, коли обидва коди, що утворюють код-добуток, відносяться до однієї групи кодів. У випадку, коли для рядків та стовпців, групи кодів є різними, лічба ведеться попеременно до різних максимальних значень. Відповідно, різні значення діапазонів будуть активними під час декодування рядків та стовпців. Цю обставину варто враховувати під час розрахунку оцінки пропускної здатності декодеру.

У випадку порівняння декодеру, який для обробки даних використовує повний цикл (і, відповідно, максимальну кількість тактових імпульсів) та декодеру, керованого запропонованим скінченим автоматом, обробка кодів з максимальною довжиною слова відбувалась би однаково, тому виграшу у пропускній здатності немає. Проте, для кодів-добутків, які базуються на кодах з довжиною меншою за максимальну стає можливим отримати виграш у пропускній здатності за рахунок того, що на обробку буде витрачатись менша кількість тактових імпульсів, аніж для декодеру з менш гнучким керуючим блоком, який використовує однаковий цикл обробки для усіх кодів.

Розроблений у даній роботі скінчений автомат, який використовує лічильник і на основі діапазонів його значень, організовує під-стани для подачі керуючих імпульсів на блоки у реконфігуровному декодері дозволяє організувати обробку даних за мінімально необхідну для обраного коду кількість тактів. Такий автомат дозволяє підвищити пропускну здатність реконфігуровного декодеру при роботі з кодами, які не відносяться до групи кодів з найбільшою довжиною кодового слова. Виграш у швидкодії залежить від максимальної довжини коду у групі, а також від довжини коду групи, для якого проводиться обробка. За рахунок того, що не обробка проводиться не для всієї матриці, а лише для необхідної частини, яка заповнена вхідними даними, вдається підвищити пропускну здатність. Відносний виграш у такому випадку можна обчислити відповідно до (7) як

$$Q = \frac{\sum_t l_{blockt} + n_{c\,actual}}{\sum_t l_{blockt} + n_{c\,top}}, \qquad (8)$$

де $n_{c\,top}$ - довжина коду, який представляє найдовший код, який може оброблюватись декодером;

$n_{c\,actual}$ - довжина коду, який заданий поточною конфігурацією для обробки.

Даний показник (Q) вказує те, наскільки вдалося зменшити кількість тактів у порівнянні зі сценарієм, коли обробка проводилась з використанням ресурсів, призначених для найдовшого коду.

Висновки. Запропонований у роботі скінчений автомат дозволяє організувати роботу реконфігуровного декодеру ТР-кодів таким чином, щоб виконувати обробку повідомлень на

основі різних кодів з використанням найменшої необхідної кількості тактів. Він містить чотири основні стани, за допомогою яких здійснюється керування основними блоками обробки. Особливістю автомату у даній роботі є те, що він використовує лічильник для того, щоб реалізувати керування основними імпульсами запуску блоків обробки. Значення, які може набувати лічильник мають бути розділені на діапазони, відповідно до яких і виконуються необхідні операції. Розбиття на діапазони виконується на основі налаштувань модулів обробки даних та періодів активностей їх роботи (моменти часу, коли вони оброблюють вхідні дані). За рахунок того, що для декодування використовується мінімально необхідна кількість тактових імпульсів для конкретного коду, вдається підвищити пропускну здатність декодеру. Розроблений скінчений автомат був реалізований у складі керуючого модуля декодеру ТР-кодів на базі мікросхеми ПЛІС виробництва фірми Altera, яка була надана для проведення досліджень та тестувань фірмою «Віаком» (м. Київ). Пропускна здатність реалізованого декодеру залежить від конкретного типу коду, який подається на вхід, а також налаштувань декодеру (зокрема, щодо кількості ітерацій). Для кодів, які відносяться до групи з довжиною 64 ((64,57), (63,56), (58,51), (46,39)) пропускна здатність становила понад 200 Мбіт/с при 10 повноцінних ітераціях (відповідно, при кращій якості сигналу та меншому рівні шуму даний показник збільшується).

1. Berrou C. Near Shannon limit error-correcting coding and decoding : Turbo-codes / C. Berrou, A. Glavieux, P. Thitimajshima – IEEE ICC'93, 1993. – pp. 1064–1071.
2. Goubier T. Fine Grain Parallel Decoding of Turbo Product Codes: Algorithm and Architecture / T. Goubier, C. Dezan, B. Pottier, C. Jego – 2008 5th International Symposium on Turbo Codes and Related Topics, 2008. DOI: 10.1109/TURBOCODING.2008.4658678.
3. Krainyk Y. Low-complexity high-speed soft-hard decoding for turbo-product codes / Y. Krainyk, V. Perov, M. Musiyenko – 2017 IEEE 37th International Conference on Electronics and Nanotechnology (ELNANO), 2017. – Kyiv, 2017. - pp. 471-474.
4. Krainyk Y. Hardware-Oriented Turbo-Product Codes Decoder Architecture / Krainyk Y., Perov V., Musiyenko M., Davydenko Ye. // Conference Proceedings of IEEE IDAACS-2017. - Bucharest, 2017. -pp. 151-154.
5. Zhou L. Flexible and high-efficiency turbo product code decoder design / Zhou L., Liu H., Zhang B. // IEICE Electronics Express, Vol. 9, No. 12. – 2012. – pp. 1044-1050. DOI: 10.1587/elex.9.1044
6. Megha S. Iterative Decoding Algorithm for Turbo Product Codes / Megha M.S. / International Journal of Innovative Research and Advenced Engineering (IJIRAE), Vol. 1, Issue 2. – April 2014. – pp.151-154.
7. Han J. High Speed Max-Log-MAP Turbo SISO Decoder Implementation Using Branch Metric Normalization / Han J., Erdogan E., Arslan T. / Proceedings of the IEEE Computer Society Annual Symposium on VLSI. – 2005. DOI: 10.1109/ISVLSI.2005.37
8. Mathana J. FPGA Implementation of High Speed Architecture for Max Log Map Turbo SISO Decoder / Mathana J., Rangarajan Dr. / International Journal of Recent Trends in Engineerign, Vol. 2, No. 6. – 2009. – pp. 142-146.