МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА ЕЛЕКТРОНІКИ І КОМП'ЮТЕРНОЇ ТЕХНІКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи бакалавра на тему:

«Пристрій перетворення двійкових чисел в фібоначчієві числа»

Завідувач кафедри Опанасюк А.С.
Керівник Борисенко О.А.
Студент гр. ЕС-61 Литвиненко А.М.

Суми 2020 р.

РЕФЕРАТ

Пояснювальна записка: 36 аркушів; 10 рисунків; 5 джерел.

Пояснювальна записка складається зі вступу, 2 розділів, висновку і списку літератури.

Перший розділ включає синтез логічних функцій, постановку задачі та її розв’язання.

В другому розділі проводиться синтез перетворювача кодів: створення шифрувальної таблиці, реалізація логічних функцій, реалізація перетворювча кодів, мінімізація функцій та реалізація перетворювача кодів на 7-сегментному індикаторі.

ЗМІСТ

Зм.	Лист	№ докум.	Підпис	Дата	ЕлІТ 6.171.00.10.537 ПЗ		
Розроб.	Литвиненко А. М.				Пристрій перетворення двійкових чисел в фібоначчієві числа	Літ.	Аркуш	Аркушів
Перевір.	Борисенко О.А.						3	36
Т. Контр.						СумДУ, гр. ЕС-61		
Н. Контр.								
Затверд.	Опанасюк А.С.							

ВСТУП

Серед задач, які вирішуються в системах обробки інформації, досить часто зустрічаються задачі надійного збереження і передачі інформації, серед яких особливо важливими є задачі завадостійкого кодування. Для рішення даних задач використовуються різні завадостійкі коди від найпростіших, призначених для перевірки кодових слів на парність чи непарність, до досить складних циклічних кодів з виправленням багаторазових помилок.

В основі різних завадостійких кодів лежить використання надлишкової інформації, що розміщується в закодованих словах, яка, як правило, додається до них штучно. Однак має місце й інакший вид надлишкового кодування, при якому кодові слова використовують вже наявну в них стандартну надлишковість. Її наявність вимагає значно меншу кількість апаратних затрат для реалізації даних пристроїв кодування, так як вони частково вже були реалізовані в пристроях, які генерують кодові слова для інших основних завдань. Алгоритми і декодуючі пристрої також при цьому реалізуються значно простіше.

Наприклад, такі комбінаторні об'єкти як перестановки, які використовуються для вирішення задач комбінаторної оптимізації, містять стандартну надлишковість. Цю надлишковість досить ефективно можна використовувати для виявлення та виправлення помилок, як при передачі інформації, так і при її обробці. Таке досить стандартне поєднання функціональних можливостей оброблюваних кодів з завадостійкістю наділяє їх високою ефективністю, так як не вимагає додаткового штучного кодування. При цьому знижуються необхідні апаратні витрати і значно збільшується швидкість передачі та обробки інформації.

Важливим також є і те, що ефект захисту від помилок при використанні станартної надмірності може значним чином посилюватися звичайним додаванням штучної надлишковості. В такому випадку вдається досягти підвищених показників захисту оброблюваної і передаваної інформації від помилок при відносно невеликому ускладненні кодуючої апаратури. Як правило, таким ефектом який добре підсилює стійкість, володіють коди на парність чи непарність.

Значною перевагою кодів з стандартною надлишковістю є також і те, що вони мають можливість здійснювати наскрізний контроль. Він реалізується в захисті даних, як при їх перетворенні безпосередньо пристроями систем обробки інформації, так і при їх подальшій передачі по каналах зв'язку, що значно підвищує ефективність систем обробки інформації. Завадостійке кодування тільки зі штучною надлишковістю не призначене для наскрізного контролю, і саме тому воно вирішує завдання контролю помилок або в каналах зв'язку, або в системах обробки інформації.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Постановка задачі

Стандартну надлишковість можна виявити практично в будь-яких кодах які оброблюються в системах обробки інформації. Однак не у всіх з них її можна легко використовувати для завадостійкого кодування, саме це власне і створює обмеження для застосування стандартної надлишковості на практиці. Однак є коди, в яких стандартна надлишковість виявляється досить просто, і тому вона може використовуватися на практиці. Такими кодами з стандартною надлишковістю являються числа завадостійких систем числення, до них відносяться факторіальні, біноміальні і фібоначчієві.

Факторіальні коди цінні саме тим, що вони дозволяють на своїй основі будувати перестановки, які, володіють стандартною надлишковістю, можуть самостійно використовуватися для завадостійкого кодування. Однак самі ці коди, як і перестановки, є за своїм походженням багатозначними кодами, а рівень їх завадостійкості нажаль недостатньо високий, для ефективного завадостійкого кодування навіть з додатковим штучним кодуванням.

Біноміальні коди можуть бути двійковими, з вигляду не відрізнятись від звичайних двійкових кодів, однак вони мають заборонені кодові комбінації, які надають їм досить таки непогану завадостійкість, що дозволяє виявляти різноспрямовані двійкові помилки типу 0 в 1 і 1 в 0. Важливим також є і те, що їх кодові слова є числами, а значить, вони можуть використовуватися не тільки при передачі інформації, а також для її обробки, і відповідно завдяки цьому є можливість здійснювати наскрізний контроль як для обробки так і для передачі інформації. Та через свою структурою - це досить складні коди, і їхню стійкість бажано використовувати при вирішенні функціональних задач, як наприклад, при генеруванні сполучень, композицій і так далі.

Фібоначчієві коди за своїм походженням досить близькі до звичайних двійкових кодів і саме тому вони найбільш прості за своєю структурою з кодів, які можна отримати за допомогою завадостійких систем числення. Але в цей же час вони мають здатність виявляти помилки, як при передачі, так і при обробці інформації, а це значить, що вони здатні здійснювати наскрізний контроль в системах обробки інформації. Однак ці помилки є односторонніми, типу 0 в 1. Тому коди Фібоначчі найбільш ефективні в асиметричних каналах обробки і передачі інформації, що на зазвичай зустрічається досить часто. З огляду на те, що симетричних каналів в системах обробки інформації практично немає, то можна вважати, що цей недолік не принциповий і легко компенсується простотою алгоритмів виявлення і виправлення помилок. Крім того, при необхідності дану асиметричність досить легко можна компенсувати введенням простих завадостійких кодів зі штучною надлишковістю, в результаті отримавши кумулятивний ефект підсилення завадостійкості.

Однак у всіх наведених вище завадостійких кодів з звичайною надлишковістю, в тому числі і фібоначчієвих, наявний загальний недолік: вони потребують для зв'язку з двійковою технікою виконувати перетворення з фібоначчієвих кодів в двійкові коди і назад з двійкових в фібоначчієві коди. Воно вирішується стандартними методами перетворення кодів, які в базовому вигляді досить складні для ефективної реалізації. Виникає завдання максимально спростити таке перетворення, стандартизуючи її для будь-якої розрядності перетворюваних повідомлень.

1.2 Теоретичний аспект

Коди Фібоначчі породжуються Фібоначчієвою системою числення і складаються з фібоначчієвих чисел, які представляють собою послідовність чисел Фібоначчі 1 1 2 3 5 8 .... Fn. Кожне з них характеризуються рівністю: $F_n = F_{n-1} + F_{n-2}$ .. яка означає, що кожне наступне число ряду Фібоначчі дорівнює сумі двох його попередніх чисел.

Кількісне значення фібоначчієвих чисел задається нумераційною функцією, вагами якої є числа Фібоначчі:

$N = a_n F_n + a_{n-1} F_{n-1} + \cdots + a_i F_i + \cdots + a_1 F_1$, де $a_i \in \{0,1\}$ - двійкова цифра і-го розряду; F_i - вага і-го розряду, рівна і-му числу Фібоначчі. В скороченому вигляді рівняння можна записати як $a_n a_{n-1} \dots a_i \dots a_1$. Особливістю є те, що представлена нумераційна функція не має нульового розряду. Це пов'язано з тим, що цифра в даному розряді завжди рівна нулю, хоча ваговий коефіцієнт дорівнює одиниці. Тому даний розряд ніяк не впливає на кінцевий результат підсумовування і не пишеться.

Зазначене подання фібоначчієвих чисел є їх мінімальною або нормальною формою. Є й інші варіанти представлення фібоначчієвих чисел. У даній роботі розглядається тільки мінімальна форма подання. Наприклад, числа 25, 54 і 33 в формі фібоначчієвих чисел в мінімальній формі представляються так, як це показано в табл. 1:

Таблиця 1- Приклади чисел Фібоначчі

Номер розряду n	8	7	6	5	4	3	2	1
Вага розряду і F	34	21	13	8	5	3	2	1
N = 25	0	1	0	0	0	1	0	1
N = 54	1	0	1	0	1	0	1	0
N = 33	0	1	0	1	0	1	0	1

Діапазон фібоначчієвих чисел $P = F_n + F_{n-1}$. Для двох розрядів діапазон фібоначчієвих чисел рівний 1+1=2, для 3-2+3=5, 4-3+5=8. Відповідно і кількість двійкових розрядів фібоначчієвих чисел може мати різну довжину від 1 розряду для двох фібоначчієвих чисел - 0 і 1 і до n.

Особливістю фібоначчієвих чисел, яка відрізняє їх від звичайних двійкових чисел, є їх завадостійкість. Вона проявляється в тому, що в фібоначчієвих числах заборонена наявність одиниць які стоять поряд. Це призводить до того, що порушення цієї умови є ознакою помилки. Цю помилку легко виявити за наявністю поруч розташованих двох і більше одиниць. Тому і виникла задача перетворення звичайних двійкових чисел в фібоначчієві, для того щоб виконувати в них різні завдання завадостійкої обробки і передачі інформації, і потім завдання зворотного перетворення фібоначчієвих чисел в звичайний двійковий код. До завдань обробки інформації в першу чергу відносять арифметичні операції, і серед них операцію рахунку. Також не виключаються і логічні операції, наприклад, порівняння та інші. Фібоначчієві числа досить таки ефективно дозволяють все це виконувати.

1.3 Розв’язання задачі

Перетворення двійкового числа в фібоначчієву форму є можливим при обчисленні числової функції. Однак при досить великих значеннях n це обчислення стає скрутним. Це відбувається тому що кількість операцій підсумовування чисел Фібоначчі в двійковому коді лінійно збільшується від величини n, що позначається на швидкодії перетворення, значно зменшуючи його. Також при вирішенні оберненої задачі з перетворення фібоначчієвих чисел в двійкову форму швидкість перетворення з ростом їх довжини n дещо знижується. В даному випадку необхідне ще додаткове попереднє перетворення двійкових вагових степенних коефіцієнтів розрядів, що стоять при одиницях, в фібоначчієві числа, і лише після цього варто проводити їх складання за правилами фібоначчієвої системи числення. Є й інші варіанти перетворення одних систем числення в інші, але всі вони, як і описуваний, або вимагають досить значної кількості пам'яті для своєї реалізації, або значно підвищують час перетворення.

З метою виявлення компромісу між обсягом часу перетворення і необхідної ємністю пам'яті в даній роботі пропонується метод перетворення двійкових чисел будь-якої довжини в фібоначчієві числа і назад. Він використовує в своїй основі поділ перетворюваних чисел на сегменти. Такий поділ сприяє отриманню базових блоків перетворення і його стандартизації та універсальності. У свою чергу це сприяє збільшенню експлуатаційних характеристик безпосередньо перетворювачів і підвищує їх надійність.

Відповідно до запропонованого методу при перетворенні двійкових чисел в фібоначчієві спочатку вони поділяються на сегменти, що містять в собі три розряди - тріади, починаючи відповідно з нульового розряду. Кожна з тріад, як відомо, в двійковому коді представляє одну з вісімкових цифр. Тим самим вихідне двійкове число штучно перетворюється в двійково-вісімкове число, яке складається з двійкових тріад, з таким самим кількісним еквівалентом, як і вихідне двійкове число.

Після цього кожній тріаді ставиться у відповідність фібоначчієвий сегмент довжиною чотири біта, як це показано в табл. 2.

Таблиця 2 - Фібоначчієві числа для ряду 1 2 3 5

Номер розряду	Вісімкова цифра	4	3	2	1
		5	3	2	1
0	000	0	0	0	0
1	001	0	0	0	1
2	010	0	0	1	0
3	011	0	1	0	0
4	100	0	1	0	1
5	101	1	0	0	0
6	110	1	0	0	1
7	111	1	0	1	0

Виходить він за спеціальним алгоритмом, який встановлює цю саму відповідність. В результаті після перетворення буде отримана певна послідовність фібоначчієвих сегментів, що містить їх стільки, скільки було тріад в двійковій послідовності. Це буде вже Фібоначчі-вісімкове число, в якому кожен фібоначчієвий сегмент відповідає тріаді, що представляє одну з восьми цифр. В результаті цього буде отримано однозначну відповідність між вихідним двійково-вісімковим числом і Фібоначчі-вісімковим.

Надалі після завадостійкої обробки і передачі Фібоначчі-вісімкової інформації в системах обробки інформації вирішується завдання зворотного переходу від Фібоначчі-вісімкових чисел до двійково-вісімкових чисел і вже потім до двійкових. Вона полягає в тому, що кожному фібоначчієвому сегменту з чотирьох розрядів за певним спеціальним алгоритмом ставиться у відповідність двійково-вісімкова цифра, яка представлена у вигляді двійкової тріади, так як це показано в табл. 2. В результаті цього буде отримана певна послідовність двійково-вісімкових цифр, яка потім подається як звичайне двійкове число.

Особливістю такого перетворення чисел є те, що як в прямому, так і в зворотному випадку перетворення в якості проміжної використовується саме двійково-вісімкова система числення. У першому випадку вона використовується як вихідна система, у другому як результуюча. Як відомо, вона надає можливість легко переходити до неї від двійкової системи і назад. Це є особливістю яка закладена в самій структурі звичайних двійкових чисел, ваги яких йдуть як степені двійки. Наприклад, розглянемо перетворення двійкового числа 011100000010 в фібоначчієве число. Воно здійснюється наступним чином. Спочатку дане двійкове число розбивається на тріади: 011 100 000 010. Вони представляють цифри вісімкового числа 3402. Потім тріадам двійкового числа, які представляють вісімкові цифри, ставляться у відповідність фібоначчієві цифри з чотирьох розрядів. Після цього вони виписуються у вигляді послідовності фібоначчієвих сегментів, взятих з табл. 2: 0100 0101 0000 0010, за допомогою яких потім після перетворення відбувається завадостійка передача і обробка інформації в системах обробки інформації. Отримані результати потім, при необхідності, можуть бути перетворенні з фібоначчієвого числа в його звичайний двійковий вигляд.

В такому випадку за допомогою табл. 2 відбувається реалізація відповідності між Фібоначчі-вісімковими сегментами і двійково-вісімковими цифрами. Після цього вони виписуються у вигляді певної послідовності тріад і перетворюються в двійкову послідовність, яка є результатом даного перетворення.

Зазначена відповідність між двійково-вісімковими числами і звичайними двійковими числами випливає з загального досить добре відомого правила, за яким числа будь-якої позиційної системи числення із статичними вагами 4, 8 і так далі переводяться в двійкову систему попередніми групуванням їх двійкових цифр по 2, 3 і так далі розрядів. Так, наприклад, двійкове число 011100000010 перетворюється в двійково-вісімкове число простим групуванням двійкових цифр в двійкові тріади: 011 100 000 010 і після цього їх перетворенням в вісімкові цифри. Також можливий варіант зворотного перетворення, коли вісімкове число може бути представлене двійково-вісімковим числом:

3402 = 011 100 000 010.

При цьому воно дорівнюватиме десятковому числу

$$3402 = 3 \times 8^3 + 4 \times 8^2 + 0 \times 8^1 + 2 \times 8^0 = 179.$$

Отже, і двійкове число, отримане з двійково-вісімкового числа 011 100 000 010, також має бути рівним цьому десятковому числу:

$$111100000010 = 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^1 = 1024 + 512 + 256 + 2 = 1794.$$

Таким чином, підводячи підсумок вищесказаному, запропонований спосіб перетворення двійкових чисел в фібоначчієві і назад полягає в наступному. При перетворенні двійкового числа в двійково-вісімкове число воно поділяється на тріади, починаючи з нульового розряду і до старшого розряду. Якщо загальне число розрядів не кратне трьом, то зліва від старшого розряду проставляються нулі так, щоб загальне число розрядів в двійковому числі стало кратним трьом. Після цього кожен двійковий сегмент - тріада перетворюється у відповідний йому сегмент фібоначчієвого числа. Далі з таких сегментів утворюється результуюче фібоначчієве число, і на цьому процедура перетворення двійкового числа в сегментне Фібоначчі-вісімкове число завершується.

Назад, так само як і в вище описаному методі, кожному Фібоначчі-вісімковому сегменту ставиться у відповідність двійково-вісімкова цифра і формується в кінцевому підсумку двійкове число, яке рівне за кількісним значенням вихідному сегментному Фібоначчі-вісімковому числу.

Встановлення відповідності між вісімково-фібоначчієвими і двійково-вісімковими числами і назад для кожного сегмента і тріади проходить за однаковою процедурою. Це спрощує процедуру перетворення фібоначчієвих чисел будь-якої довжини в двійкові числа і назад, стандартизуючи її. В результаті це призводить до однорідності алгоритмів перетворення і реалізуючої її апаратури. При цьому збільшується її надійність, і зменшуються експлуатаційні витрати. Завадостійкість сегментних фібоначчієвих кодів також підвищується, так як в разі одиночної помилки можна перепитати інформацію тільки цього сегмента або включити замість нього дублюючий сегмент.

Недолік даного методу, це збільшення сумарної розрядності Фібоначчі-вісімкових сегментних чисел в порівнянні зі звичайними фібоначчієвими числами з ростом довжини перетворюваних кодів, можна компенсувати збільшенням швидкодії їх перетворення, так як перетворення окремих сегментів в двійкові комбінації можна проводити паралельно.

2 ЛОГІЧНИЙ СИНТЕЗ ПЕРЕТВОРЮВАЧА КОДІВ

2.1 Шифрувальна таблиця

Для того щоб побудувати перетворювач кодів використовується табл. 3. Вона складається в першій колонці при рахунку зліва направо із номерів восьми цифр, які пронумеровані від 0 до 7, їм в другій колонці відповідно надані трьох-розрядні двійково-десяткові набори значень змінних $X_1X_2X_3$. В третій колонці даним наборам відповідають 4-розрядні двійково-шістнадцятирічні набори значень чотирьох логічних функцій $f_1f_2f_3f_4$. Ці набори є двійково-шістнадцятковими тому, що кодують чотирма двійковими розрядами цифри шістнадцяткової системи числення. Вони пронумеровані десятковими числами в четвертій колонці справа табл. 3.

Значення логічних функцій f_1f_2 f_3 f_4 отримані випадково, тому в табл. 3 вони виконують шифрування вихідної інформації, захищаючи її від несанкціонованого доступу. Тож табл. 3 відповідно буде в подальшому називатися шифрувальною, а 4-разрядні двійково-шістнадцяткові комбінації в третій колонці при рахунку зліва шифрами 4-розрядних двійково-десяткових наборів.

Таблиця 3 – Шифрувальна таблиця

№10	**$X_1X_2X_3$**	**f_1 f_2 f_3 f_4**	**№16**
0	000	0000	0
1	001	0001	1
2	010	0010	2
3	011	0100	4
4	100	0101	5

Продовження таблиці 3

№10	**$X_1X_2X_3$**	**$f_1\,f_2\,f_3\,f_4$**	**№16**
5	101	1000	8
6	110	1001	9
7	111	1010	10

2.2 Синтез логічних функцій

Відповідно до даної шифрувальної табл. 3 отримаємо перш за все досконалу диз'юнктивну нормальну форму для кожної з чотирьох логічних функцій $f_1f_2f_3f_4$. Реалізація кожної з них відбувається наступним чином. Виписується для кожної одиниці вибраної з табл. 3 функції $f_1f_2f_3f_4$ кон'юнкції для трьох змінних $X_1X_2X_3$. Таких кон'юнкцій для функції f_1 буде три, для f_2- дві, для f_3 - дві, для f_4 - три. Там де в наборах стоять нулі над відповідними змінними кон'юнкцій потрібно ставити інверсії. Після цього отримані таким чином кон'юнкції з інверсіями об'єднуються за допомогою знаків диз'юнкції. Відповідно до наведеного алгоритму в табл. 3 подані в досконалій диз'юнктивній нормальній формі логічні функції будуть мати наступний вигляд:

$$f_1 = X_1\bar{X}_2X_3 + X_1X_2\bar{X}_3 + X_1X_2X_3;$$
$$f_2 = \bar{X}_1X_2X_3 + X_1\bar{X}_2\bar{X}_3;$$
$$f_3 = \bar{X}_1X_2\bar{X}_3 + X_1X_2X_3;$$
$$f_4 = \bar{X}_1\bar{X}_2X_3 + X_1\bar{X}_2\bar{X}_3 + X_1X_2\bar{X}_3.$$

2.3 Схемна реалізація логічних функцій в досконалій диз'юнктивній нормальній формі

Реалізація логічних функцій в досконалій диз'юнктивній нормальній формі відбувається на базі логічних схем *І* та *АБО*. Схема *І* дорівнює одиниці тільки в тому випадку, коли на всіх її входах будуть подані одиниці, а схема *АБО,* коли хоча б на один її вхід буде подана одиниця. Саме тому, щоб отримати на виході схеми *І* одиницю необхідно, щоб на її входах там де змінна дорівнює нулю стояв інвертор, який перетворить нуль в одиницю. Тож в результаті, якщо, наприклад, перетворюється комбінація 0000, то на всіх входах відповідної конюнкції обов'язково повинні стояти інверсії, які позначаються на її входах колами.

Виходи усіх схем *І* кожної функції обєднуються за допомогою схем *АБО.* На їх виходах будуть отримані двійково-шістнадцяткові значеня, які відповідають конкретним значенням двійково-десяткових цифр. Тобто дані логічні схеми в купі реалізують перетворювач кодів із вихідних двійково-десяткових цифр в двійково-шістнадцяткові (рис. 1).

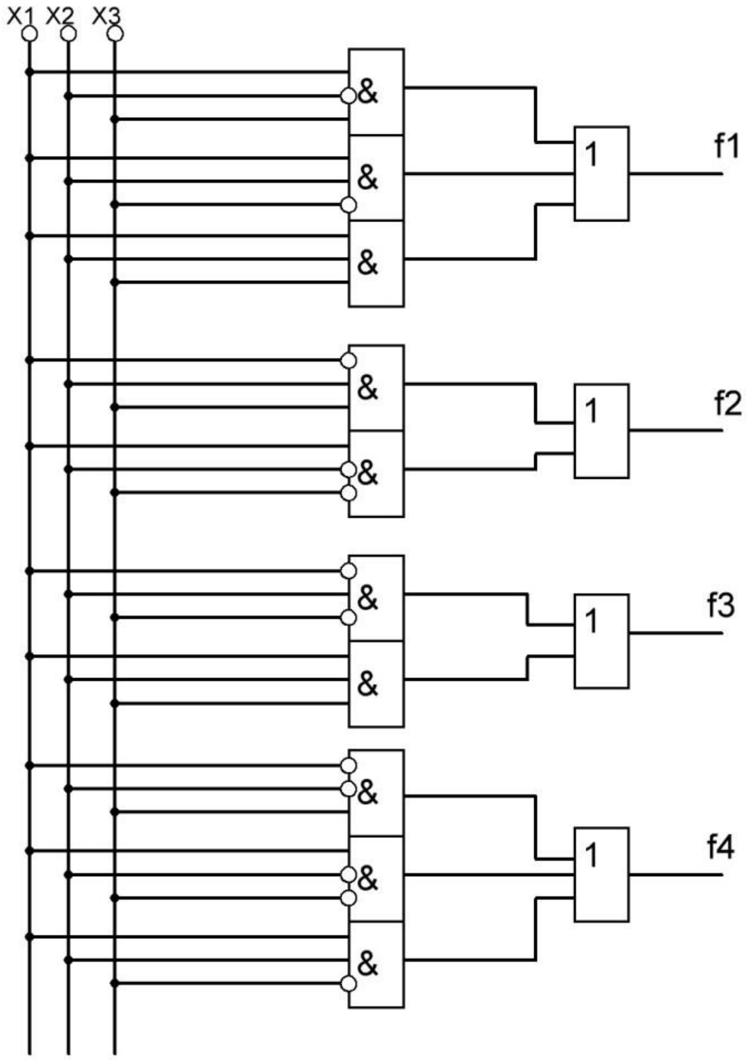

Рисунок 1 – Функціональна схема перетворювача кодів

Апаратні витрати на реалізацію даної схеми підраховуються шляхом підсумування кількості входів, які необхідно витратити на реалізацію інверторів, схем *І* та *АБО*. Тож для цього в даних логічних функціях насамперед підраховується кількість схем *І* та інверторів. Їхня кількість відповідно буде дорівнювати 10 та 12. Кожна схема *І* має чотири входи, а інвертор один. Тож кількість входів потрібних для реалізації схем *І* та інверторів буде дорвівнювати 10×3+12×1=30+12=42. До них треба додати десять входів, які необхідні для реалізації схем *АБО*. В результаті загальна кільність входів, а відповідно й кількість апаратних витрат у перетворювача кодів, що розглядається, 42+10=52.

2.4 Реалізація перетворювача кодів за допомогою програмно-логічної матриці

Отримані вище схеми логічних функцій f_1 f_2 f_3 f_4 можна реалізувати в вигляді програмної логічної матриці показаної на рис. 2. В даній матриці поданим трьом горизонтальним лініям зверху відповідають змінні $X_1X_2X_3$, а лініям, що йдуть під кожною з них їх інверсії. Цих ліній, відповідно, буде три. Кола на верхній лінії до яких під'єднується нижня лінія означають, що вона реалізує інверсії змінних, пов'язані з верхньою лінією. Вертикальні лінії, що перетинають горизонтальні лінії, кількість яких в випадку, що розглядається, дорівнює шести, зображають конюнкції &1 &2, …, &10. Щоб їх реалізувати, на місці перетину горизонтальної лінії з вертикальною при наявності відповідної змінної з інверсією чи без в відповідній конюнкції ставиться крапка. Якщо вона ставиться на верхній лінії, то контактує змінна без інверсії. В іншому випадку – змінна з інверсією. Цих точок буде рівно стільки скільки є змінних в конюнкції, тобто три. В результаті на кожній вертикальній лінії реалізується одна зі схем *І*, яка входить в логічну функцію. Їх буде рівно стільки, скільки кон'юнкцій входить в ту чи іншу логічну функцію, тобто або три, або два.

Нижче в програмно-логічній матриці під всіма шістьома горизонтальними лініями, які реалізують схеми *І,* розміщені чотири горизонтальні лінії, які реалізують схеми *АБО*.

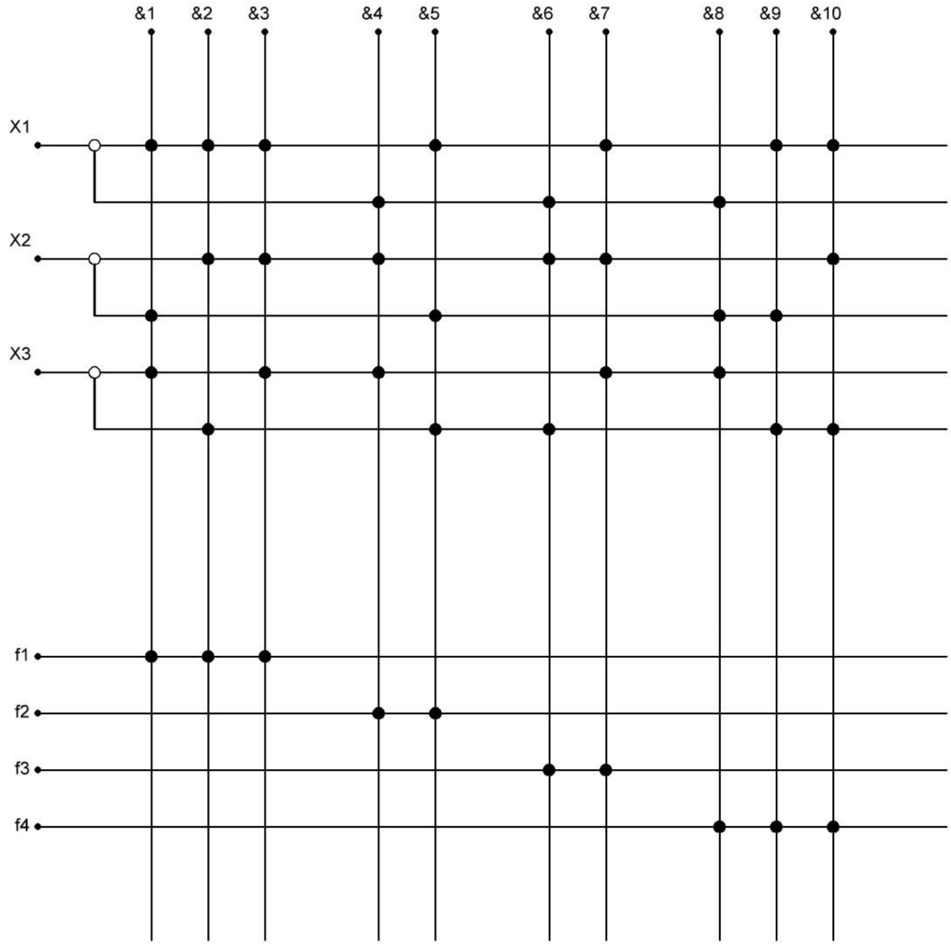

Рисунок 2 – програмно-логічна матриця логічних функцій

Їх перетинають ті ж самі вертикальні лінії, які перетинали горизонтальні лінії всіх трьох змінних та їх інверсій, однак, на них тепер показані контакти виходів схем І із входами схем АБО для кожної логічної функції в вигляді крапок. А це означає, що кожна горизонтальна нижня лінія зображає собою схему АБО.

Далі необхідно підрахувати кількість входів які йдуть на реалізацію програмно-логічної матриці в цілому. Для цього перш за все підраховується кількість крапок на верхніх горизонтальних лініях які без інверсії, а вже потім на нижніх інверсних лініях, що під ними. Остання кількість крапок помножується на два, тому що окрім входів схем І, також треба врахувати один вхід інвертора який розміщується перед кожним із них. Всього крапок на верхніх горизонтальних лініях дорівнює 18+12х2=18+24=42. Додавши кількість крапок на нижніх горизонтальних лініях f_1 f_2 f_3 f_4, кількість яких рівна десяти, отримаємо загальну кількість крапок, тобто 42+10=52. Їхня кількість співпадає з раніше підрахованою кількістю входів в схемі 1 перетворювача кодів на базі логічних схем *І* та *АБО*, а саме 52.

2.5 Мінімізація логічних функцій

Одержані в досконалій диз’юнктивній нормальній формі логічні функції необхідно мінімізувати. Для цього будуються діаграми Вейча, кожна з яких має вісім клітинок, так як це зображено на рис. 3 - 6. Перш за все в клітинки, які знаходяться на перехрещенні трьох змінних з інверсіями чи без, що відповідають тій або іншій кон’юнкції, ставляться одиниці. Після цього шукають клітинки, в яких стоять поряд чотири одиниці, дві одиниці та об’єднують їх. Причому одна й та ж клітинка цілком може знаходиться в різних об’єднаннях. В першому випадку, коли об’єднуються чотири одиниць, шукають одну змінну, яка їх покриває, коли дві одиниці то дві змінні. Якщо клітинка з одиницею не може об’єднатися з іншими клітинками з одиницями, то вона покривається трьома змінними, тобто мінімізація в цьому випадку для відповідної кон’юнкції відсутня.

Так, для функції f_1 діаграма Вейча буде наступною.

	X_2		$\overline{X}_2$	
$\overline{X}_1$				
X_1	1	1	1	
	$\overline{X}_3$	X_3		$\overline{X}_3$

Рисунок 3 – Таблиця Вейча

В результаті мінімізації за допомогою діаграми Вейча логічна функція f_1 буде мати більш стислий запис, чим раніше:

$$f_1 = X_1X_2 + X_1X_3.$$

Для функції f_2 діаграма Вейча буде наступною.

	X_2		$\bar{X}_2$	
$\bar{X}_1$		1		
X_1				1
	$\bar{X}_3$	X_3		$\bar{X}_3$

Рисунок 4 – Таблиця Вейча

Так як немає можливості об’єднати клітинки, в яких є поряд хоча б дві одиниці, запис логічної функції f_2 не скорочується і не змінюється:

$$f_2 = \bar{X}_1X_2X_3 + X_1\bar{X}_2\bar{X}_3 \,.$$

Для функції f_3 діаграма Вейча має наступний вигляд:

	X_2		$\bar{X}_2$	
$\bar{X}_1$	1			
X_1		1		
	$\bar{X}_3$	X_3		$\bar{X}_3$

Рисунок 5 – Таблиця Вейча

Також немає можливості об’єднати клітинки, в яких є поряд хоча б дві одиниці, тому запис логічної функції f_3 не скорочується і не змінюється:

$$f_3 = \bar{X}_1 X_2 \bar{X}_3 + X_1 X_2 X_3.$$

Діаграму Вейча для функції f_4 сформуємо таким чином:

	X_2		$\bar{X}_2$	
$\bar{X}_1$			1	
X_1	1			1
	$\bar{X}_3$	X_3		$\bar{X}_3$

Рисунок 6 – Таблиця Вейча

Після мінімізації за допомогою діаграми Вейча логічна функція f_4 буде записана наступним чином:

$$f_4 = \bar{X}_1\bar{X}_2X_3 + X_1\bar{X}_3.$$

2.6 Схемна реалізація мінімізованих логічних функцій

Мінімізовані логічні функції f_1 f_2 f_3 f_4 реалізуються за допомогою схем аналогічно з реалізацією не мінімізованих функцій, яка була розглянута вище. Схемна реалізація логічних функцій в мінімальних диз'юнктивних нормальних формах наведена на рис. 7.

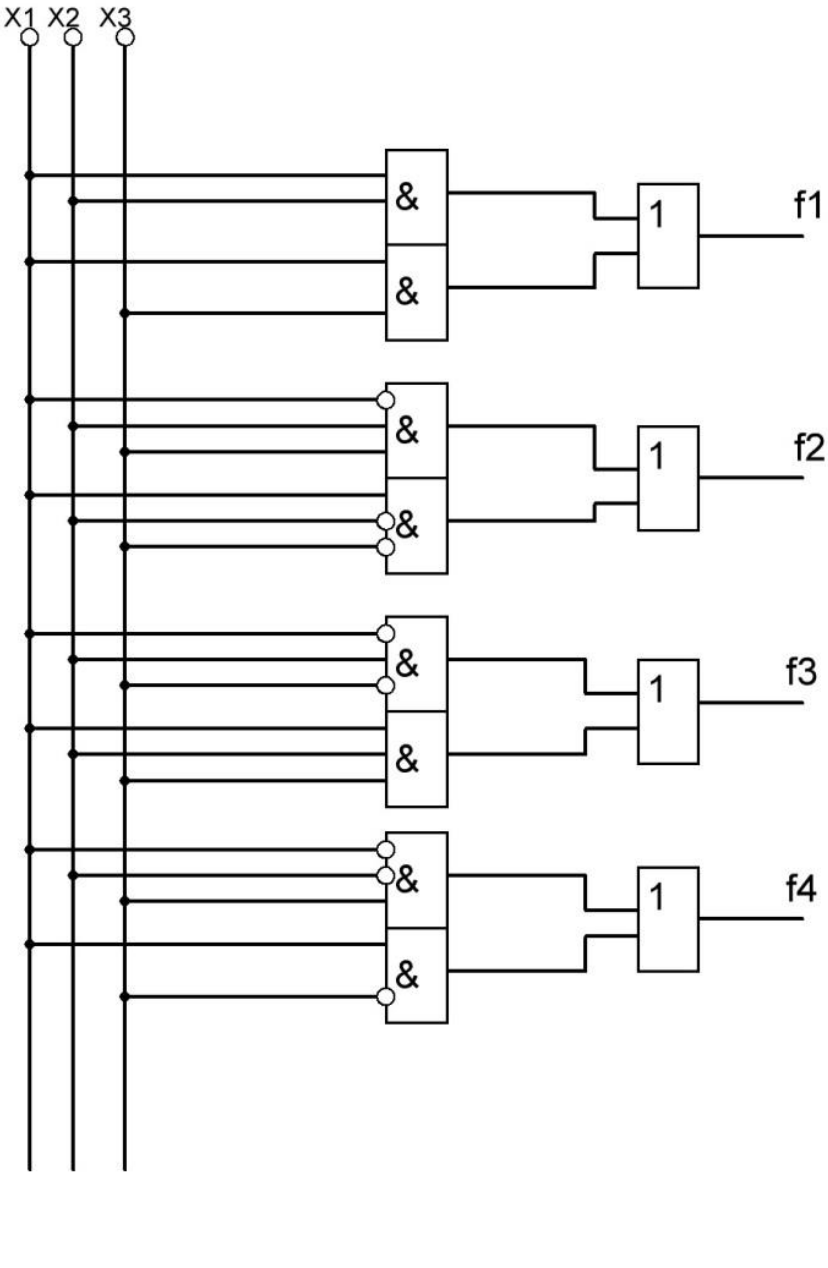

Рисунок 7 – Функціональна схема перетворювача кодів з мінімальними ДНФ

					ЕлІТ 6.171.00.10.537 ПЗ	Арк.
						27
Зм..	Лист	№ докум.	Підпис	Дата		

Розрахуємо апаратні витрати необхідні для реалізації цієї схеми. Схема перетворювача кодів мстить вісім інверторів з одним входом в кожному та вісім схем *І*, п’ять з яких мають лише по три входа, а три - по два входа. Тож кількість входів необхідна для реалізації схем *І* та инверторів буде дорвівнювати 3×2+5×3+8×1=6+15+8=29. Кількість входів для реалізації схем *АБО* дорвівнює восьми. Тому після того як була проведена мінімізація загальна кількість входів дорівнює 29+8=37. Таким чином, проведена мінімізація дозволила зменшити апаратні витрати на реалізацію перетворювача кодів з 52 до 37, тобто в 1,41 раза.

2.7 Програмно-логічна матриця мінімальних диз'юнктивних нормальних форм

Сформовані вище схеми мінімізованих логічних функцій $f_1\,f_2\,f_3\,f_4$ можна легко реалізувати в вигляді програмно-логічної матриці. Вони формуються за аналогією з програмно-логічними матрицями для досконалих диз'юнктивних нормальних форм на основі отриманих вище мінімальних логічних функцій.

Програмно-логічна матриця для мінімальних диз'юнктивних нормальних форм наведена на рис. 8.

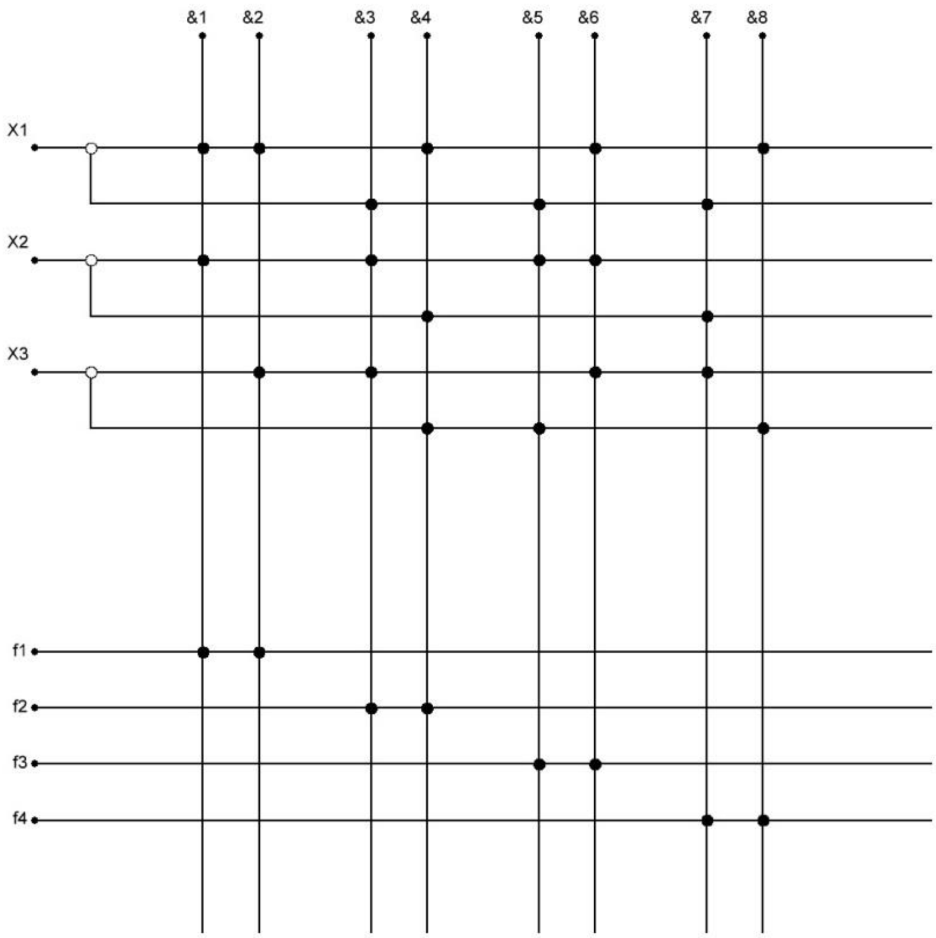

Рисунок 8 – ПЛМ для мінімальних ДНФ

Підрахуємо кількість входів в даній матриці відповідно до розглянутого вище правила, коли кількість крапок в нижній горизонтальній лінії помножається на два, а в відповідній верхній на один, та додається кількість точок на нижніх горизонтальних лініях для функцій f_1 f_2 f_3 f_4. Відповідно, отримуємо наступний результат: 13+8·2+8=37, який співпадає з вже раніше розрахованою кількістю входів в схемі перетворювача кодів з мінімальними диз'юнктивними нормальними формами на базі логічних схем *І* та *АБО*, а саме дорівнює 37.

Звернемо також увагу і на те, що кожна логічна функція $f_1 f_2 f_3 f_4$ на рис. 1 – 8 створює неповний лінійний дешифратор, а кожна схема АБО, яка об'єднує виходи цього самого дешифратора, неповний шифратор з одним виходом. Сумісно виходи цих самих шифраторів створюють виходи перетворювача кодів, який синтезується. В ньому кожній двійково-десятковій комбінації на вході ставиться в відповідність двійково-шістнадцяткова комбінація на виході. Аналогічно можна будувати перетворювач кодів на будь-яке число входів і виходів.

2.8 Перетворювач кодів для 7-сегментного індикатора

Блок перетворювача кодів Фібоначчі складається з дешифратора і зв'язаного з ним шифратора. На входи цього дешифратора надходить 4-розрядний двійковий сигнал, який являє собою шифр однієї з восьми десяткових цифр від нуля до семи. Відповідно на одному з його виходів в цей час з'явиться сигнал одиниці, а на всіх інших нуля. Сигнал одиниці передається на відповідний вхід шифратора і перетворюється ним в 7-розрядну двійкову комбінацію, яка показує сегменти індикатора, які треба підсвітити для відображення десяткової цифри, двійково-десятковий код якої був поданий на вхід цифрових систем передачі і відображення інформації.

Так як відображатися може абсолютно будь-яка цифра з десяти, то треба перетворити кожну двійково-десяткову цифру в її зображення на 7-сегментному індикаторі. Відповідно для цього потрібно перетворити 4-розрядні двійкові зображення десяткових цифри в двійкові 7-сегментні зображення. А для цього потрібно кожній двійково-десятковій комбінації поставити в відповідність 7-сегментну комбінацію, як це подано в табл. 4. Тож ця комбінація буде отримана на виході шифратора, на входи якого були заведені відповідні виходи дешифратора, кількість яких дорівнює восьми. Входів повного шифратора для семи розрядів на його виході, очевидно, буде 128, але використовуватися з них буде лише десять. Їх номера в десятковому вигляді показані в табл. 4 в третій колонці. А це значить, що табл. 4 має всю необхідну інформацію для синтезу перетворювача кодів на основі дешифратора-шифратора.

Таблиця 4 - Відображення сегментів на індикаторі

Номер сегмента / Цифра	1234567	Номер входу шифратора
0	1111110	126
1	0011000	24
2	0110111	55
3	0111101	61
4	1011001	89
5	1101101	109
6	1101111	111
7	0111000	56

Тепер необхідно на основі отриманої блок-схеми перетворювача кодів одержати його функціональна схему. Для цього будуються сім схем *АБО*, виходи яких відповідають сегментам індикатора, що підсвічуються, а потім на них підключаються відповідні зв’язки з виходів шифратора. Ці зв’язки йдуть таким чином, що під’єднують входи, кількість яких для кожної схеми *АБО* не перевищує десяти, тільки якщо на них повинна подаватися одиниця. А це значить, що для відображення нуля під’єднуються до відповідного виходу дешифратора по одному входу всіх схем *АБО* за виключенням останньої схеми.

Також необхідно враховувати таблицю шифрування, відповідно до якої, наприклад шифр цифри 5 буде 1000. Значить, для відображення даної цифри на індикаторі треба буде знімати на шифратор сигнал з восьмого виходу дешифратора. Подібним чином треба зв’язати і інші виходи дешифратора. Тим самим ці зв’язки зможуть реалізовувати функцію декодування шифру системи. А їх знання розкриває шифр.

На рис. 9 показано з’єднання восьми виходів дешифратора з вісьмома входами шифратора у відповідній блок-схемі перетворювача кодів Фібоначчі. Дана блок – схема налічує вісім зв’язків у відповідності до кількості десяткових цифр, але виходи в неї беруться з шістнадцяти номерів, які можуть не відповідати цифрі, що відображається. Ці номери зв’язуються із входами шифратора, які відповідають десятковим значенням 7-розрядних двійкових номерів, які підсвічують сегменти індикатора. Максимально можливий вхідний номер шифратора відповідно буде 127 = 1111111, а кількість всіх його входів буде дорівнювати восьми. Тобто це буде неповний дешифратор на вісім двійкових входів і сім таких же виходів.

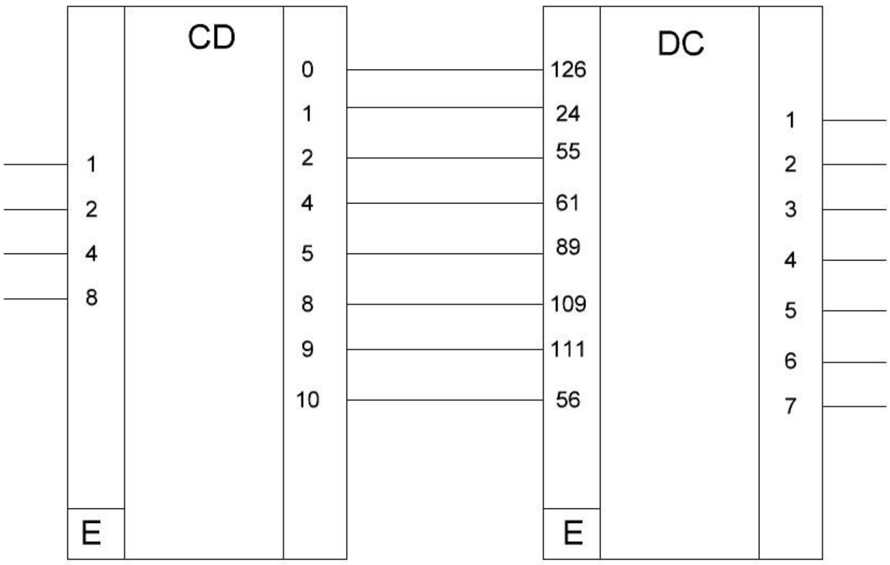

Рисунок 9 – Блок-схема перетворювач кодів Фібоначчі

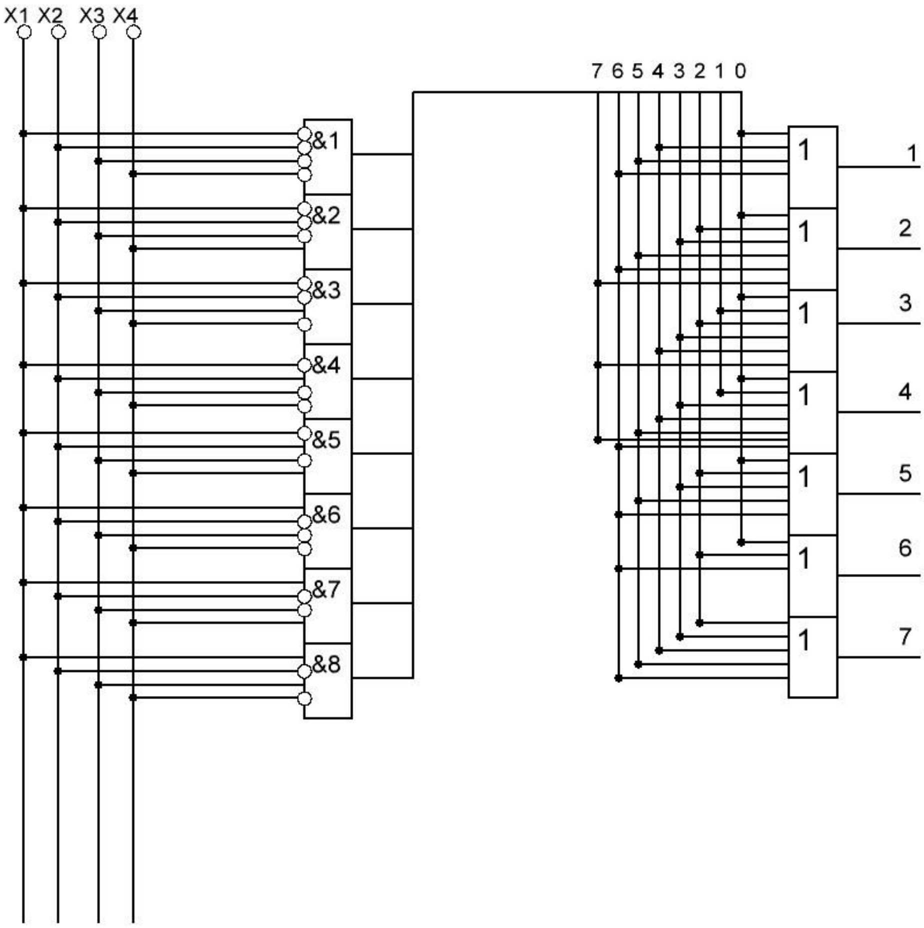

Рисунок 10 – Схема перетворювача кодів Фібоначчі

ВИСНОВОК

Підводячи підсумки по даній роботі, можна відзначити, що в ній пропонується принципово новий підхід до побудови обчислювальної техніки, який поєднує переваги як Фібоначчієвої системи числення, так і двійково-вісімкової. Його головною перевагою є сегментна обробка інформації, яка дозволяє з окремих універсальних блоків збирати будь-які обчислювальні пристрої з завадостійкою обробкою і передачею інформації. Такий підхід усуває принциповий недолік двійкових систем числення - відсутність завадостійкості. Його легко долають завадостійкі системи числення і зокрема фібоначчієва.

Використання ж Фібоначчі-вісімкової системи числення спільно з двійково-вісімковою знімає досить багато питань побудови завадостійкої і швидкодіючої комп'ютерної техніки, в тому числі, якщо говорити про перспективу, і універсальних комп'ютерів. Ця техніка, хоча і є в своїй основі фібоначчієвою, проте досить легко поєднується з двійковими системами, тим самим дозволяючи будувати гібридні двійково-фібоначчієві комплекси з наскрізним контролем обробки і передачі інформації. До цього часу таку техніку не вдавалося розробити через складнощі її побудови. Однак і тут, перш ніж говорити про реальні системи, потрібно пройти тривалий шлях теоретичних досліджень і практичних випробувань.

ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Кравчук С. Основи комп'ютерної техніки / С. Кравчук, В.Шонін Київ, 2015. – 156с.
2. Таненбаум Е. Архитектура компьютера / Е. Таненбаум, Т. Остін – СПб.: Издательский дом Питер, 2016. - 816с.
3. Борисенко О. А. Цифрова схемотехніка посібник / О. А. Борисенко. – Суми : Вид-во СумДУ, 2016. – 203 с.
4. О. Д. Азаров Комп'ютерна схемотехніка / О. Д. Азаров – Вінниця : ВНТУ, 2018. – 230 с.
5. Коваленко А.Є. Комп'ютерна схемотехніка і архітектура комп'ютерів. / А.Є.Коваленко.- К.: НТУУ «КПІ», 2016.-472 с.