КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

ПРИСТРІЙ ВИКОНАННЯ КОМП'ЮТЕРНОЇ ОПЕРАЦІЇ ДІЛЕННЯ ЦІЛИХ ЧИСЕЛ У ФОРМАТІ ЗАЛИШКІВ ДІЛЕНОГО ЗІ ЗНАКОЗМІННИМ НУЛЕМ

Дипломна робота бакалавра
студента 4 року навчання
Спеціальність: 123 «Комп'ютерна інженерія»
Георгія ТАТАРЧЕНКА

Науковий керівник
канд. техн. наук Олександр САМОЩЕНКО,
доцент кафедри комп'ютерної інженерії

Рецензент
канд. фіз.-мат. наук Іван КОЛОМІЄЦЬ,
асистент кафедри електрофізики

До захисту допускаю:
Завідувач кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри "____" _____2022 р., протокол № ____

Київ 2022

РЕФЕРАТ

Пояснювальна записка містить 61 с., 19 рис., 20 джерел посилання, 4 таблиці.

Ключові слова: ділення, ціле число, кодування, частковий залишок, залишок від ділення, переповнення, знак числа, модуль доповняльного коду, залишок за модулем, прямий код, зворотний код, доповняльний код, негативний і позитивний нуль доповняльного коду залишків діленого.

Об’єкт дослідження – комп’ютерна (арифметична) операція ділення. Предметом роботи є віртуальний пристрій, що виконує арифметичну операцію ділення цілих чисел у доповняльних кодах з позитивним і негативним нулем залишків ділення.

Методи розроблення: комп’ютерне моделювання, симуляція роботи пристрою, розробка пристрою на основі теоретичних положень. Інструменти розроблення: середовище Proteus, Intel Altera Quartus II, ModelSim, мова програмування FPGA – Verilog HDL.

Результати роботи: виконано загальний огляд теоретичних положень досліджуваного алгоритму, проведено порівняння описаного методу ділення цілих чисел із типовими, проведено синтез та розроблено віртуальний пристрій, що виконує операцію арифметичного ділення цілих чисел у доповняльних кодах із позитивним і негативним нулем залишків діленого двома способами виконання: у вигляді окремої схеми та схеми на базі FPGA.

Отриманий пристрій ділення може застосовуватися як окремий ALU або бути вбудованим у більш складний пристрій, де необхідне застосування арифметичної операції ділення цілих чисел.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ЕОМ – електронна обчислювальна машина

ALU – arithmetic logic unit (арифметико-логічний пристрій)

ДК – доповняльний код

ПЛІС – програмована логічна інтегральна схема

FPGA – field-programmable gate array

I/O інтерфейс – інтерфейс вводу/виводу

ВСТУП

В теперішній час більшість, або навіть усі обчислювальні машини є цифровими, тобто працюють за принципами перетворення цифрових даних. У цифровій техніці поширення здобуло двійкове кодування, а саме кодування, в основі якого покладена двійкова система числення (цифри «1» та «0»). Принцип роботи ЕОМ або ALU також ґрунтується на двійковій системі.

Але для виконання будь-яких обчислень у двійковій системі спочатку необхідно визначити принципи таких обчислень. Для вирішення даної задачі створена двійкова «арифметика», яка вказує на певні закономірності при виконанні найпростіших комп’ютерних арифметичних операцій для двійкових чисел, а саме додавання, віднімання, множення та ділення. В свою чергу, ділення цілих чисел із залишком — арифметична операція, що грає велику роль в арифметиці, теорії чисел, алгебрі та криптографії. Оскільки ділення є однією з елементарних операцій, схему, яка виконує дану операцію, можна використовувати як окремий ALU, або у складі більш складних алгоритмів, в залежності від очікуваного функціоналу.

В [1] запропоновано математичний опис виконання операції ділення цілих чисел у доповняльному коді. Доповняльні коди цілих чисел зі знаком в операції ділення описуються єдиною формулою для позитивних та негативних чисел і обчислюються як залишок за модулем, який дорівнює кількості двійкових наборів доповняльного коду числа. На базі запропонованої моделі доповняльних кодів цілих чисел аналітично обґрунтовано оригінальні способи фіксації переповнення розрядної сітки частки при діленні цілих чисел з використанням вихідного переносу та ознаки нульового значення старшого часткового залишку діленого. Показано, що нульові значення часткових залишків діленого при визначенні цифрових розрядів частки та обчисленні наступного часткового

залишку діленого слід розглядати як позитивний або негативний двійковий код в залежності від знаку діленого. Зважаючи на особливості кодування у доповняльних кодах отримані в циклі ділення цифри частки корегуються лише при від’ємному значенні частки, при чому єдиним способом – шляхом додавання одиниці до молодшого розряду отриманого інверсного коду частки.

Наведені алгоритми є перспективними для створення конкурентоспроможних арифметичних пристроїв. Але практична реалізація наведеного підходу вимагає виконання попередніх досліджень з метою оцінки ефективності виконання однієї з основних комп'ютерних арифметичних операцій.

Таким чином, **метою роботи** є реалізація теоретичних положень, що роз’яснені у статті [1], шляхом побудови пристрою, який виконує комп'ютерну дію за поданим алгоритмом. Для досягнення цієї мети поставлено такі задачі.

- дослідити комп’ютерну операцію ділення цілих чисел у доповняльному коді із негативним і позитивним нулем залишків діленого;

- навести приклади типових алгоритмів виконання операції ділення;

- провести порівняння методів, що розглянуті;

- побудувати схему, що виконує дію за досліджуваним алгоритмом та провести симуляцію її роботи.

РОЗДІЛ 1 ПРЕДСТАВЛЕННЯ ЦІЛИХ ЧИСЕЛ У КОМП'ЮТЕРНОМУ ФОРМАТІ

1.1 Загальні уявлення про представлення цілих чисел в комп'ютерному форматі

Для спрощення розуміння інформації, вхідні та вихідні дані повинні бути подані у формі, зручній для людини. Для людей звичною стала десяткова система числення, а для комп'ютерів – двійкова. Пояснити таку особливість можна тим, що технічна реалізація пристрою з двома стійкими станами (є електричний струм - немає струму) є набагато простішою, аніж із десятьма. Зазвичай один стан приймають за нуль, а другий – за одиницю.

Отже, оскільки для комп'ютера є зручнішою двійкова система, усі дані, що зберігаються в ньому, представляються у вигляді нулів та одиниць, а точніше – їх послідовностей. Якщо ці два стани (нуль та один) подати у вигляді алфавіту {0,1}, то такі послідовності можна назвати словами у цьому алфавіті. Тому обробка даних у комп'ютерній системі описується як перетворення комп'ютерних слів за правилами, які задає процесор (мікропроцесор). Елемент заданої послідовності з нулів та одиниць (букву слова) називають бітом – елементарною одиницею інформації.

Перетворення зовнішньої інформації у внутрішнє зображення називають кодуванням. Кодом називають як сам спосіб відображення, так і безліч слів (кодових комбінацій), що використовуються при кодуванні.

Для представлення чисел в ЕОМ зазвичай використовують бітові набори фіксованої довжини, що складаються з нулів та одиниць. Обробка послідовностей фіксованої довжини є технічно легшою, аніж наборів змінної. Порядковий номер (позиція) окремого біта у послідовності називається

розрядом. У ЕОМ розрядом називають також частину регістру (або комірку пам'яті), що зберігає один біт.

Задля спрощення виконання та урахування особливостей кожної з арифметичних операцій і визначення знаку результату застосовують спеціалізовані методи кодування чисел. Операція віднімання трактується як арифметичне додавання кодованих чисел, полегшується обробка наявного переповнення розрядної сітки. Як результат спрощується проектування та подальша робота пристроїв, що виконують комп'ютерні операції. Для кодування чисел зі знаком у ЕОМ застосовують прямий, зворотний та доповняльний коди.

У загальному випадку коди трактують як число без знаку, а діапазон чисел без знаку розбивається на дві складових. Одна з них містить позитивні числа, інша – негативні. Розбиття виконується таким чином, щоб належність до піддіапазону визначалася якомога простіше.

Найрозповсюдженішим і зручним методом є формування кодів у такий спосіб, щоб значення старшого біта вказувало на знак числа, тобто використання такого способу кодування дозволяє відобразити старший розряд як знаковий (біт знаку), а інші – як цифрові розряди коду.

1.2 Цілі числа без знаку

Кількість різноманітних бітових наборів довжини n дорівнює 2^n (кожний розряд може прийняти одиницю або нуль), саме тому щоб визначити, яке число представляє певний бітовий набір, вигідно різним наборам поставити в відповідність різні числа. Це дозволяє закодувати 2^n різних (від 0 до n-1) чисел.

Серед усіх теоретично можливих способів представлення чисел найбільш зручним є бітовий набір, відповідний числу, який представляється n-розрядним записом цього числа в двійковій системі. Таким чином, можна реалізувати

арифметичні операції над числами, використовуючи відомі шкільні алгоритми порозрядної обробки бітових наборів.

1.3 Цілі числа зі знаком

Для представлення цілих чисел у двійковому форматі використовуються наступні головні варіанти кодування:

- прямий код;
- зворотний код;
- доповняльний код.

Усі три способи використовують лівий (старший) розряд послідовності з n бітів для кодування знаку числа: знак “плюс” кодується нулем, а “мінус” — одиницею. Інші (n-1) розрядів (названі цифровими або мантисою) використовуються для представлення абсолютної величини (модуля) числа.

Додатні числа у прямому, зворотному та доповняльному кодах зображуються однаково – цифрова частина містить запис числа у двійковому форматі, а у знаковому розряді знаходиться закодований знак “+”, тобто цифра 0. Наприклад, кодування для n-розрядного додатного числа з n = 8 можна побачити на рис.1.1

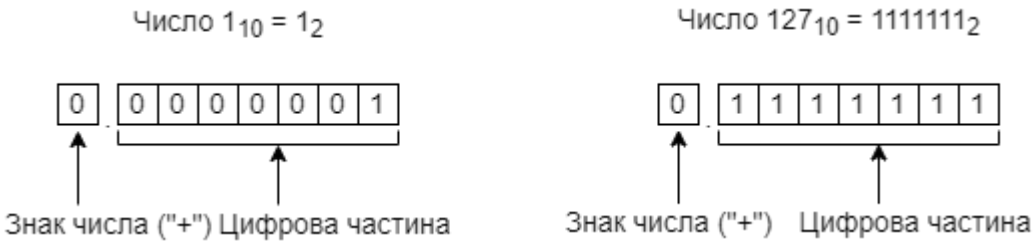


Рисунок 1.1 – Представлення додатних чисел у прямому, зворотному та доповняльному кодах

Діапазон чисел для кодування знаходиться в межах $[0, 2^{(n-1)} - 1]$.

Від’ємні числа у прямому, зворотному та додатковому кодах отримуються у різний спосіб.

Прямий код формується тим же чином, що і на рис.1.1, але тут вже знаковий розряд (знак “-”) кодується одиницею. Наприклад, число (-1_{10}) буде дорівнювати 1.0000001_2, а (-127_{10}) - 1.1111111_2.

Діапазон чисел для кодування знаходиться в межах $[-(2^{(n-1)} – 1), 0]$.

Формування зворотного коду відбувається у наступний спосіб: зворотний код визначають як інвертування всіх розрядів двійкового коду абсолютної величини числа, включаючи знаковий – одиниці замінюються нулями, а нулі – одиницями (рис.1.2).

Рисунок 1.2 – Представлення від'ємного числа у зворотному коді

Діапазон чисел для кодування знаходиться в межах $[-(2^{(n-1)} – 1), 0]$.

Доповняльний код, в свою чергу, можна назвати модифікацією зворотного, оскільки такий код визначається як утворення зворотного коду з наступним додаванням одиниці до його молодшого розряду. Наприклад (рис.1.3):

Рисунок 1.3 – Представлення від’ємного числа у доповняльному коді

Діапазон чисел для кодування знаходиться в межах $[-2^{(n-1)}, -1]$.

1.4 Діапазони значень цілих чисел

Зазвичай, цілі числа у комп'ютерній техніці характеризуються розрядностями (форматами), наведеними у табл. 1.1.

Таблиця 1.1 – Комп'ютерні формати цілих чисел

Формат числа в байтах	Діапазон			
	Запис з порядком		Звичайний запис чисел	
	Зі знаком	Без знаку	Зі знаком	Без знаку
1	$[-2^7;\ 2^7-1]$	$[0;\ 2^8-1]$	[-128; 127]	[0; 255]
2	$[-2^{15};\ 2^{15}-1]$	$[0;\ 2^{16}-1]$	[-32786; 32767]	[0; 65535]

Продовження таблиці 1.1

Формат числа в байтах	Діапазон			
	Запис з порядком		Звичайний запис чисел	
	Зі знаком	Без знаку	Зі знаком	Без знаку
4	$[-2^{31};\ 2^{31}-1]$	$[0;\ 2^{32}-1]$	[-2147483648; 2147483647]	[0; 4294967295]
8	$[-2^{63};\ 2^{63}-1]$	$[0;\ 2^{64}-1]$	[-9223372036854775887; 9223372036854775887]	[0; 18446744073709551615]

РОЗДІЛ 2 ТИПОВІ МЕТОДИ КОМП'ЮТЕРНОГО ДІЛЕННЯ ЦІЛИХ ЧИСЕЛ

2.1 Приклади типових методів комп'ютерного ділення цілих чисел

Операція ділення уявляє собою послідовність циклічних дій. Спочатку дільник віднімається від числа, що ділиться, а потім циклічно із його залишків. У результаті ділення цілих чисел отримують цілу частину частки і залишок від ділення, як додатковий результат, який не округляється:

$$Z = \frac{X}{Y}\ (\text{ціла частина}) + \frac{W(\text{залишок})}{Y}, \qquad (2.1)$$

де X – ділене;

Y – дільник;

Z – число результату;

W – залишок від ділення.

Основними двома способами ділення, згідно до [10] та [19] є:

- Ділення з відновленням залишку;
- Ділення без відновлення залишку.

2.2 Операція ділення цілих чисел з відновленням залишку

Використання методу ділення цілих чисел з відновленням залишку відбувається у прямому коді і передбачає циклічне виконання трьох послідовних дій:

1) Віднімання X – Y, а потім A_i- Y.

За знаком залишку A_i визначається цифра частки:

$$Z_i = \begin{cases} 1, \text{ якщо } A_i \geq 0; \\ 0, \text{ якщо } A_i < 0. \end{cases} \quad (2.2)$$

2) Відновлення від'ємного залишку, тобто $A_i + Y$, якщо $A_i < Y$;

3) Зсув $X(A_i)$ вліво на один розряд. В останньому циклі зсув не виконується.

Формати операндів можна побачити на рис. 2.1

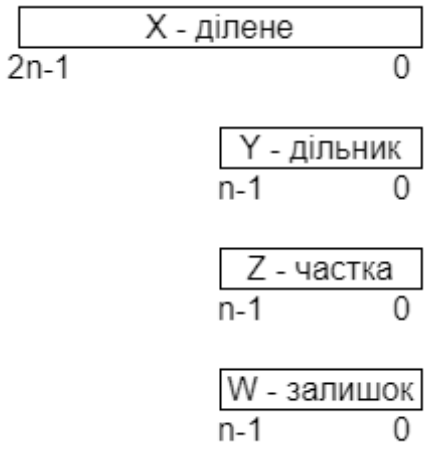


Рисунок 2.1 – формати (розрядність) операндів операції ділення

Залишок від ділення W розміщується у старших розрядах діленого.

Особливостями даного методу є:

- Частка рахується починаючи зі старших розрядів. При формуванні кожної наступної цифри Z_i частка Z зсувається вліво на один розряд, та доповнюється нулем або одиницею в залежності від результату віднімання.

- Перед початком ділення X та Y перевіряються на наявність переповнення, яке відбувається, коли старша половина розрядів X більша (або дорівнює) за дільник Y. Наприклад, $X = 13_{10} = 1101_2$, $Y = 2_{10} = 10_2$. Оскільки $X_{\text{старші}} = 11_2 > Y = 10_2$, відповідно, відбудеться переповнення. Справді, якщо перевірити дану особливість, можна побачити, що при діленні числа 13 на число 2 результатом буде число 6. Дане число не вміститься у розрядну сітку, що складається з двох біт, тому що $6_{10} = 110_2$. Оскільки частка формується зі старшого розряду,

старша одиниця може бути втрачена, а результат спотворений. Саме цьому перевірка на переповнення є необхідним кроком на початку виконання алгоритму.

- Знак частки при діленні у прямому коді формується за допомогою виконання операції додавання за модулем два знакових розрядів діленого та дільника:

$$S = \begin{cases} 1, \text{ якщо } S_x \oplus S_y = 1; \\ 0, \text{ якщо } S_x \oplus S_y = 0, \end{cases} \quad (2.3)$$

де S – знак результату Z;

S_x – знак діленого X;

S_y – знак дільника Y.

- Частка формується як результат ділення модулів початкових чисел;

- Кількість ітерацій залежить від вхідної розрядності дільника та діленого. Ділене має розрядність 2n, а дільник – n. Зважаючи на це, необхідною кількістю ітерацій є n+1. Наприклад, якщо маємо розрядність дільника чотири (діленого вісім) маємо виконати п’ять послідовних ітерацій, оскільки окрім чотирьох розрядів результату необхідно передбачити переповнення.

2.3 Операція ділення цілих чисел без відновлення залишку

Метод ділення без відновлення залишку має деякі відмінності і зводиться до таких кроків:

1) Залишок A_i завжди множиться на два.

До уваги береться знак залишку A_i, за яким визначається цифра частки:

$$Z_i = \begin{cases} 1, \text{ якщо } S\ (\mathrm{A_i}) = 0; \\ 0, \text{ якщо } S\ (\mathrm{A_i}) = 1, \end{cases} \quad (2.4)$$

де $S\ (\mathrm{A_i})$ – знак залишку $\mathrm{A_i}$;

2) Виконується одна з двох операцій в залежності від знаку $\mathrm{A_i}$:

а) віднімання ($2\mathrm{A_i} - Y$), якщо залишок додатній;

б) додавання ($-2\mathrm{A_i} + Y$), якщо залишок від'ємний.

Принцип формування частки (формування зі старших розрядів), кількість ітерацій алгоритму та знак результату визначаються аналогічно до методу із відновленням залишків.

Відмінною рисою даного методу від попереднього є те, що спосіб ділення з відновленням залишків має аритмічний алгоритм, оскільки проходить зі змінною кількістю кроків на кожній ітерації:

- три кроки при $2\mathrm{A_i} < Y$;
- два кроки при $2\mathrm{A_i} \geq Y$.

Це означає, що для ритмізації процесу необхідно привести алгоритм до виконання трьох кроків на кожній ітерації, в результаті чого час на виконання операції ділення збільшується. Проаналізувавши дану особливість, а також порівнявши обидва представлені алгоритми ділення, можна сказати, що ділення без відновлення залишків є спрощеною операцією ділення з відновленням залишків, оскільки має лише два кроки на виконання кожної ітерації, а значить має вищу швидкість, а також потребує менших апаратних затрат при проектуванні пристрою, який виконує дану дію. З огляду на такий результат дослідження, для подальшого порівняння буде використовуватися лише метод без відновлення залишків.

При виконанні операції ділення результат залишається однаковим, якщо замість подвоєння залишку зменшувати вдвічі дільник на кожній ітерації. Таким чином, отримуємо два алгоритми ділення чисел без відновлення залишку:

- Алгоритм із зсувом діленого (залишку);

- Алгоритм із зсувом дільника.

У демонстрації алгоритмів із зсувом залишку та без зсуву залишку у вищенаведених пунктах наведено алгоритм із зсувом діленого (залишку). Двома головними відмінностями алгоритму із зсувом дільника є:

1) Замість множення діленого на два (тобто зсуву вліво) використовується ділення дільника на два (зсув право). З цього випливає друга відмінність;

2) Отриманий залишок необхідно знімати вже не зі старших розрядів діленого, а з молодших (оскільки відбувалося постійне ділення замість множення).

Але, згідно до [16], майже в будь якій сучасній обчислювальній машині арифметичні операції виконуються у доповняльних кодах, оскільки комп’ютеру, наприклад, простіше виконати віднімання у вигляді додавання позитивного і негативного чисел у доповняльних кодах і отримати результат у тому ж доповняльному коді. Дану особливість можна пояснити на такому прикладі: маємо два числа, що представлені у прямому коді. Перше число негативне, інше - позитивне і ці числа необхідно додати. Але спочатку комп'ютер повинен визначити ці числа, тобто отримати дані про знаки та модулі чисел. Далі аналізується інформація про те, що одне з чисел негативне, і підсумкова операція визначається як віднімання. Потім комп'ютер порівнює модулі чисел, щоб з'ясувати, яке з чисел віднімати від іншого та отримується знак результату. У підсумку, виходить складний алгоритм. Набагато простіше додавати числа, якщо числа перетворені в доповняльний код. Результат також буде мати вигляд доповняльного коду, при чому, якщо число додатне – доповняльний код дорівнює прямому. Також таке кодування призводить до того, що від’ємних чисел за заданої довжини розрядної сітки можна представити на одне більше, ніж додатних. А оскільки операція додавання є базовою для усіх інших операцій,

можна зробити висновок, що інші арифметичні операції також будуть базуватися на числах у ДК.

Зважаючи на розглянуті особливості, основним методом ділення цілих чисел для досліджень обрано метод ділення цілих чисел в доповняльному коді із негативним та позитивним нулем залишків ділення.

РОЗДІЛ 3 МАТЕМАТИЧНИЙ ОПИС ОПЕРАЦІЇ ДІЛЕННЯ ЦІЛИХ ЧИСЕЛ В ДОПОВНЯЛЬНОМУ КОДІ ІЗ НЕГАТИВНИМ ТА ПОЗИТИВНИМ НУЛЕМ ЗАЛИШКІВ ДІЛЕНОГО

Згідно з [1], реалізація операції ділення цілих чисел в доповняльному коді при допустимих значеннях операндів передбачає контроль переповнення розрядної сітки частки. Спрощення контролю переповнення частки та зменшення апаратних затрат досягаються при використанні у схемі контролю стандартних інформаційних сигналів, які формуються при обчисленні старшого часткового залишку діленого – вихідного переносу та ознаки нульового значення старшого залишку діленого. Для формування коректного коду частки нульові значення часткових залишків діленого при діленні чисел з довільними знаками слід розглядати як додатній або від'ємний нуль у відповідності із знаком діленого. Отримані в результаті циклу ділення цифри частки коректуються лише при від'ємних значеннях частки, при чому однаково – додаванням одиниці до молодшого розряду інверсного коду частки.

3.1 Загальна методика ділення цілих чисел

При діленні модулів цілих чисел обчислюється значення частки, яке передбачає виконання рівняння:

$$a = bd + c, \tag{3.1}$$

де a, b, d, c – відповідно модулі діленого, дільника, частки та залишку від діленого.

Однозначність розкладання діленого на такі складові та цілочисельний характер ділення забезпечуються виконанням нерівності (3.2):

$$b \neq 0, c < b. \tag{3.2}$$

При діленні знакових цілих чисел вирішення задачі зводиться до рішення рівняння

$$A = BD + C. \tag{3.3}$$

де A, B, D, C – відповідно знакові цілі числа діленого, дільника, частки та залишку від діленого.

Якщо знаки ‹‹+›› та ‹‹-›› A, B, C, D позначити відповідно як NA, NB, ND, NC і закодувати таким чином, щоб:

$$NX = \begin{cases} 0 \text{ при } X \geq 0; \\ 1 \text{ при } X < 0, \end{cases} \tag{3.4}$$

де X = A, B, D і C, то знак частки, відповідно правилам алгебри, буде визначатися логічним виразом ND = NA $\oplus$ NB. Знак кінцевого залишку від ділення визначається як NC = NA.

В задачі (3.1) $a \geq b * d$. В силу цього отримаємо вираз:

$$l_a \geq l_b + l_d, \tag{3.5}$$

де l_a, l_b, l_d – відповідно довжина двійкового коду діленого, дільника та частки.

При використанні n-розрядних модулів дільника та частки:

$$l_b = l_d = n, \tag{3.6}$$

нерівність (3.5) задається у вигляді

$$l_a \geq 2n. \tag{3.7}$$

При однорідній пам’яті комп’ютера, що складається з комірок довжиною n + 1 (оскільки задача (3.3) потребує використання знакових чисел, до n-розрядного модуля числа додається біт знаку, тому формат знакових діленого та частки приймає такий вигляд), знакове ділене буде компонуватися у двох суміжних комірках даної довжини. Саме тому формат знакового діленого визначається як

$$l_A = 2n + 2, \tag{3.8}$$

де l_A – довжина знакового діленого.

Довжина l_C знакового коду залишку від ділення C використовує формат знакового дільника B:

$$l_C = l_B = n + 1. \tag{3.9}$$

В даному методі за основу подаються 4 можливі випадки:

1) A > 0, B > 0 → D > 0, C > 0;

2) A > 0, B < 0 → D < 0, C > 0;

3) A < 0, B > 0 → D < 0, C < 0;

4) A < 0, B < 0 → D > 0, C < 0.

3.2 Ділення додатних чисел у доповняльному коді

При діленні додатних чисел, максимальне значення модуля частки обмежене, і визначається нерівністю:

$$|\mathrm{D}|_{\max} < 2^{\mathrm{n}}. \tag{3.10}$$

В силу цього, при представленні зміщеного дільника B(n + 1, 1) $* 2^{\mathrm{n}}$ довжиною (2n + 1) в форматі діленого A(2n + 2, 1) довжиною (2n + 2) переповнення розрядної сітки описується системою:

$$\text{ОП}\left(\frac{+}{+}\right) = \begin{cases} 0, \text{ при } \mathrm{A}^{\mathrm{n+1}} < 0; \\ 1, \text{ при } \mathrm{A}^{\mathrm{n+1}} \geq 0, \end{cases} \tag{3.11}$$

де $\mathrm{A}^{\mathrm{n+1}} = 2 * \mathrm{A} - \mathrm{B} * 2^{\mathrm{n+1}} = 2 * \mathrm{A}_{\mathrm{n+1}}$;

$\text{ОП}\left(\frac{+}{+}\right)$ - ознака переповнення розрядної сітки частки при діленні додатних цілих чисел.

Відповідно, формування ознаки переповнення частки при діленні додатних чисел описується виразом:

$$\text{ОП}\left(\frac{+}{+}\right) = \mathrm{E}, \tag{3.12}$$

де E – вихідний перенос повної суми залишку діленого $\mathrm{A}_{\text{ДК}}^{\mathrm{n+1}}$.

За відсутності переповнення частки, виконавши ряд перетворень, числові розряди додатної частки у доповняльному коді при діленні додатних чисел визначаються як:

$$d_n\left(\frac{+}{+}\right) = \overline{\mathrm{NA}_{\text{ДК}}^{\mathrm{n}} \oplus \mathrm{NB}_{\text{дк}}}, \tag{3.13}$$

де $\mathrm{A}_{\text{ДК}}^{\mathrm{n}} = \begin{cases} \left(2 * \mathrm{A}_{\text{ДК}}^{\mathrm{n+1}} + \mathrm{B}_{\text{дк}} * 2^{\mathrm{n+1}}\right) \text{ при } \mathrm{NA}_{\text{ДК}}^{\mathrm{n+1}} \neq \mathrm{NB}_{\text{дк}}; \\ \left(2 * \mathrm{A}_{\text{ДК}}^{\mathrm{n+1}} + \mathrm{B}_{\text{ДК}}^{\text{ПР}} * 2^{\mathrm{n+1}}\right) \text{ при } \mathrm{NA}_{\text{ДК}}^{\mathrm{n+1}} = \mathrm{NB}_{\text{дк}}; \end{cases}$

$\mathrm{B}_{\text{ДК}}^{\text{ПР}} = (\overline{\mathrm{B}_{\text{дк}}(n+1,1)} + 1)$ – перетворений доповнювальний код дільника довжиною n+1.

Кінцеве значення залишку діленого від ділення формується по значенню першого часткового залишку діленого $A^1_{ДК}$:

$$C_{дк}(n+1,1) = \begin{cases} A^1_{ДК}(2n+2,\ n+2) \text{ при } NA^1_{ДК} = NA_{дк}; \\ \left(A^1_{ДК}(2n+2,\ n+2) + B_{дк}\ (n+1,1)\right) \\ \quad \text{при } NA^1_{ДК} \neq NA_{дк} : NA^1_{ДК} \neq NB_{дк}, \end{cases} \tag{3.14}$$

де $C_{дк}(n+1,1)$ – цілочисельний залишок діленого при діленні додатних чисел.

3.3 Ділення цілих чисел в доповняльному коді при додатному діленому і від'ємному дільнику

При діленні додатного діленого на від’ємний дільник, максимальне значення модуля частки обмежене, і визначається нерівністю:

$$|D|_{max} \leq\ 2^n. \tag{3.15}$$

В силу цього, переповнення розрядної сітки описується системою:

$$ОП\left(\frac{+}{-}\right) = \begin{cases} 0, \text{ при } A^{n+1} \leq 0; \\ 1, \text{ при } A^{n+1} > 0, \end{cases} \tag{3.16}$$

де $A^{n+1} = 2 * A + B * 2^{n+1} = 2 * A_{n+1}$;

$ОП\left(\frac{+}{-}\right)$ - ознака переповнення розрядної сітки частки при діленні додатного діленого на від’ємний дільник.

При обчисленні подвоєного залишку A^{n+1} в доповняльному коді операція задається виразом

$$A_{ДК}^{n+1} = \left(2 * A_{дк} + B_{дк} * 2^{n+1}\right), \tag{3.17}$$

де $A_{дк} = (+|A|)_{дк} = |A|;$

$B_{дк} * 2^{n+1} = 2^{n+2} - |B|.$

Відповідно, ознака переповнення розрядної сітки частки формується задовольняючи логічний вираз

$$ОП\left(\frac{+}{-}\right) = E \oplus ZA, \tag{3.18}$$

де $ZA = \begin{cases} 0, \text{ при } A_{ДК}^{n+1} \neq 0; \\ 1, \text{ при } A_{ДК}^{n+1}(2n+2, 1)=0; \end{cases}$

E – вихідний перенос повної суми залишку діленого $A_{ДК}^{n+1}$.

За відсутності переповнення частки, виконавши ряд перетворень, числові розряди від’ємної частки у доповняльному коді при діленні додатного діленого на від’ємний дільник визначаються як

$$\overline{d_n}\left(\frac{+}{-}\right) = \overline{NA_{ДК}^{n} \oplus NB_{дк}}, \tag{3.19}$$

де $A_{ДК}^{n} = \begin{cases} \left(2 * A_{ДК}^{n+1} + B_{дк} * 2^{n+1}\right) \text{ при } \overline{d_{n+1}} = 0 : NA_{ДК}^{n+1} \neq NB_{дк}; \\ \left(2 * A_{ДК}^{n+1} + B_{ДК}^{ПР} * 2^{n+1}\right) \text{ при } \overline{d_{n+1}} = 1 : NA_{ДК}^{n+1} = NB_{дк}; \end{cases}$

Кінцеве значення залишку діленого від ділення $C_{дк}(n+1, 1)$ формується за значенням часткового залишку діленого $A_{ДК}^{1}$, який при $NA_{ДК}^{1} \neq NA_{дк}$ коректується:

$$C_{дк}(n+1, 1) = \begin{cases} A_{ДК}^{1}(2n+2,\ n+2) \text{ при } NA_{ДК}^{1} = NA_{дк}; \\ \left(A_{ДК}^{1}(2n+2,\ n+2) + B_{ДК}^{ПР}(n+1, 1)\right) \\ \quad \text{при } NA_{ДК}^{1} \neq NA_{дк} : NA_{ДК}^{1} = NB_{дк}. \end{cases} \tag{3.20}$$

Доповняльний код частки формується шляхом корекції отриманого інверсного коду частки, а саме додаванням одиниці до молодшого розряду.

3.4 Ділення цілих чисел в доповняльному коді при від'ємному діленому та додатному дільнику

При діленні додатного діленого на від’ємний дільник, максимальне значення модуля частки обмежене, і визначається нерівністю:

$$|D|_{max} \leq 2^n. \tag{3.21}$$

Тоді ознака переповнення від’ємної частки при такому діленні задається як

$$ОП\left(\frac{-}{+}\right) = \begin{cases} 0 \text{ при } A^{n+1} > 0; \\ 0 \text{ при } A^{n+1} = 0; \\ 1 \text{ при } A^{n+1} < 0, \end{cases} \tag{3.22}$$

де $ОП\left(\frac{-}{+}\right)$ - ознака переповнення розрядної сітки частки при діленні додатного діленого на від’ємний дільник;

$A^{n+1} = 2*A + B * 2^{n+1}$, $A^{n+1} = 2 * A_{n+1}$.

При обчисленні подвоєного залишку A^{n+1} в доповнювальному коді операція задається виразом

$$A_{ДК}^{n+1} = \left(2 * A_{дк} + B_{дк} * 2^{n+1}\right), \tag{3.23}$$

де $A_{дк} = (2^{2n+2} + A) = 2^{2n+2} - |A|$;

$B_{дк} * 2^{n+1} = (2^{2n+2} + B * 2^{n+1}) = |B| * 2^{n+1}$.

Відповідно, формування ознаки переповнення частки при діленні додатного діленого на від'ємний дільник описується виразом:

$$\text{ОП}\left(\frac{-}{+}\right) = \overline{\text{E}}, \tag{3.24}$$

де E – вихідний перенос повної суми залишку діленого $\text{A}_{\text{ДК}}^{\text{n+1}}$.

За відсутності переповнення частки, виконавши ряд перетворень, числові розряди від'ємної частки у доповняльному коді при діленні від'ємного діленого на додатний дільник визначаються як

$$\overline{\text{d}_\text{n}}\ \left(\frac{-}{+}\right) = \overline{\text{N(n)} \oplus \text{NB}_{\text{дк}}}, \tag{3.25}$$

де $$\text{N(n)} = \begin{cases} \text{NA}_{\text{дк}} \text{ при } \text{ZA}^{\text{n}}(\text{H}) = 1; \\ \text{NA}_{\text{дк}}^{\text{n}} \text{ при } \text{ZA}^{\text{n}}(\text{H}) = 0; \end{cases}$$

$$\text{ZA}^{\text{n}}(\text{H}) = \begin{cases} 0 \text{ при } \text{A}_{\text{ДК}}^{\text{n}}(2\text{n}+2, \text{n}+2) \neq 0; \\ 1 \text{ при } \text{A}_{\text{ДК}}^{\text{n}}(2\text{n}+2, \text{n}+2) = 0; \end{cases}$$

$\text{ZA}^{\text{n}}(\text{H})$ – ознака нульового значення в обчислюваній частині часткового залишку діленого $\text{A}_{\text{ДК}}^{\text{n}}(2\text{n}+2, \text{n}+2)$;

N(n) – ознака знакового розряду від'ємного нуля залишку $\text{A}_{\text{ДК}}^{\text{n}}$ діленого або знак значущого часткового залишку $\text{A}_{\text{ДК}}^{\text{n}} \neq 0$ діленого при діленні від'ємного діленого на додатний дільник.

Кінцеве значення залишку діленого від ділення $\text{C}_{\text{дк}}(\text{n}+1, 1)$ формується за значенням часткового залишку діленого $\text{A}_{\text{ДК}}^{1}$, який при $\text{NA}_{\text{ДК}}^{1} \neq \text{NA}_{\text{дк}}$ коректується:

$$\text{C}_{\text{дк}}(\text{n}+1, 1) = \begin{cases} \text{A}_{\text{ДК}}^{1}(2\text{n}+2,\ \text{n}+2) \text{ при } \text{N}(1) = \text{NA}_{\text{дк}}; \\ \left(\text{A}_{\text{ДК}}^{1}(2\text{n}+2,\ \text{n}+2) + \text{B}_{\text{ДК}}^{\text{ПР}}\ (\text{n}+1, 1)\right) \\ \quad \text{при } \text{N}(1) \neq \text{NA}_{\text{дк}} : \text{N}(1) = \text{NB}_{\text{дк}}, \end{cases} \tag{3.26}$$

де $N(1) = \begin{cases} \mathrm{NA}_{\text{дк}} \text{ при } \mathrm{ZA}^1(\mathrm{H}) = 1; \\ \mathrm{NA}^1_{\text{дк}} \text{ при } \mathrm{ZA}^1(\mathrm{H}) = 0; \end{cases}$

$\mathrm{ZA}^1(\mathrm{H}) = \begin{cases} 0 \text{ при } \mathrm{A}^1_{\text{ДК}}(2n+2, n+2) \neq 0; \\ 1 \text{ при } \mathrm{A}^1_{\text{ДК}}(2n+2, n+2) = 0. \end{cases}$

Доповняльний код частки формується шляхом корекції отриманого інверсного коду частки, а саме додаванням одиниці до молодшого розряду.

3.5 Ділення цілих від'ємних чисел в доповняльному коді

При діленні від’ємних чисел, максимальне значення модуля частки обмежене, і визначається нерівністю:

$$|\mathrm{D}|_{\max} < 2^{\mathrm{n}}. \tag{3.27}$$

З огляду на це отримаємо:

$$\text{ОП}\left(\frac{-}{-}\right) = \begin{cases} 0, \text{ при } \mathrm{NA}^{\mathrm{n+1}}_{\text{ДК}} = 0 : \mathrm{ZA} = 0; \\ 1, \text{ при } \mathrm{NA}^{\mathrm{n+1}}_{\text{ДК}} = 1 : \mathrm{ZA} = 0; \\ 1, \text{ при } \mathrm{NA}^{\mathrm{n+1}}_{\text{ДК}} = 0 : \mathrm{ZA} = 1, \end{cases} \tag{3.28}$$

де $\mathrm{A}^{\mathrm{n+1}}_{\text{ДК}} = (2 * \mathrm{A}_{\text{дк}} + \mathrm{B}^{\text{ПР}}_{\text{ДК}} * 2^{\mathrm{n+1}});$

$\mathrm{A}^{\mathrm{n+1}}_{\text{ДК}} = 2^{2\mathrm{n}+2} - |\mathrm{A}|;$

$\mathrm{B}_{\text{дк}} * 2^{\mathrm{n+1}} = 2^{2\mathrm{n}+2} - |\mathrm{B}| * 2^{\mathrm{n+1}}.$

Відповідно, при діленні від’ємних чисел переповнення розрядної сітки частки описується логічним виразом:

$$\text{ОП}\left(\frac{-}{-}\right) = \overline{\mathrm{E} \oplus \mathrm{ZA}}, \tag{3.29}$$

де E – вихідний перенос повної суми залишку діленого $A_{ДК}^{n+1}$.

При відсутності переповнення частки та при представленні зміщеного коду $B * 2^{n-1}$ довжиною 2n в форматі діленого довжиною 2n + 2 отримаємо прямі значення цифрових розрядів частки:

$$\overline{d_n}\left(\frac{-}{-}\right) = \overline{N(n) \oplus NB_{дк}}, \tag{3.30}$$

де $$A_{ДК}^{n} = \begin{cases} \left(2 * A_{ДК}^{n+1} + B_{ДК}^{ПР} * 2^{n+1}\right) \text{ при } N(n) = NB_{дк}; \\ \left(2 * A_{ДК}^{n+1} + B_{дк} * 2^{n+1}\right) \text{ при } N(n) \neq NB_{дк}; \end{cases}$$

$$N(n) = \begin{cases} NA_{дк} \text{ при } ZA^{n}(H) = 1 : A_{ДК}^{n}(2n+2, n+2) = 0; \\ NA_{дк}^{n} \text{ при } ZA^{n}(H) = 0 : A_{ДК}^{n}(2n+2, n+2) \neq 0. \end{cases}$$

Кінцеве значення залишку діленого від ділення $C_{дк}(n+1, 1)$ формується за значенням часткового залишку діленого $A_{ДК}^{1}$, який при $N(1) \neq NA_{дк}$ коректується:

$$C_{дк}(n+1, 1) = \begin{cases} A_{ДК}^{1}(2n+2,\ n+2) \text{ при } N(1) = NA_{дк}; \\ \left(A_{ДК}^{1}(2n+2,\ n+2) + B_{дк}\ (n+1, 2)\right) \\ \quad \text{при } N(1) \neq NB_{дк} : N(1) = NA_{ДК}^{1}; \end{cases} \tag{3.31}$$

$$N(1) = \begin{cases} NA_{дк} \text{ при } ZA^{1}(H) = 1 : A_{ДК}^{1}(2n+2, n+2) = 0; \\ NA_{дк} \text{ при } ZA^{1}(H) = 0 : A_{ДК}^{1}(2n+2, n+2) \neq 0. \end{cases}$$

4 СХЕМОТЕХНІЧНА РЕАЛІЗАЦІЯ МЕТОДУ ДІЛЕННЯ У ДОПОВНЯЛЬНОМУ КОДІ З ПОЗИТИВНИМ І НЕГАТИВНИМ НУЛЕМ ЗАЛИШКІВ ДІЛЕННЯ

4.1 Побудова функціональної схеми пристрою, що виконує ділення у доповняльному коді

Для проведення досліджень вирішено побудувати пристрій з n = 4. Це означає, що ділене приймається за двійкове число довжиною 2n + 2 = 10, а дільник, частка та залишок від ділення – n + 1 = 5. Для практичної реалізації та подальшого синтезу керуючого і операційного автоматів необхідно вказати на алгоритм виконання операції ділення за опрацьованим методом (рис. 4.1).

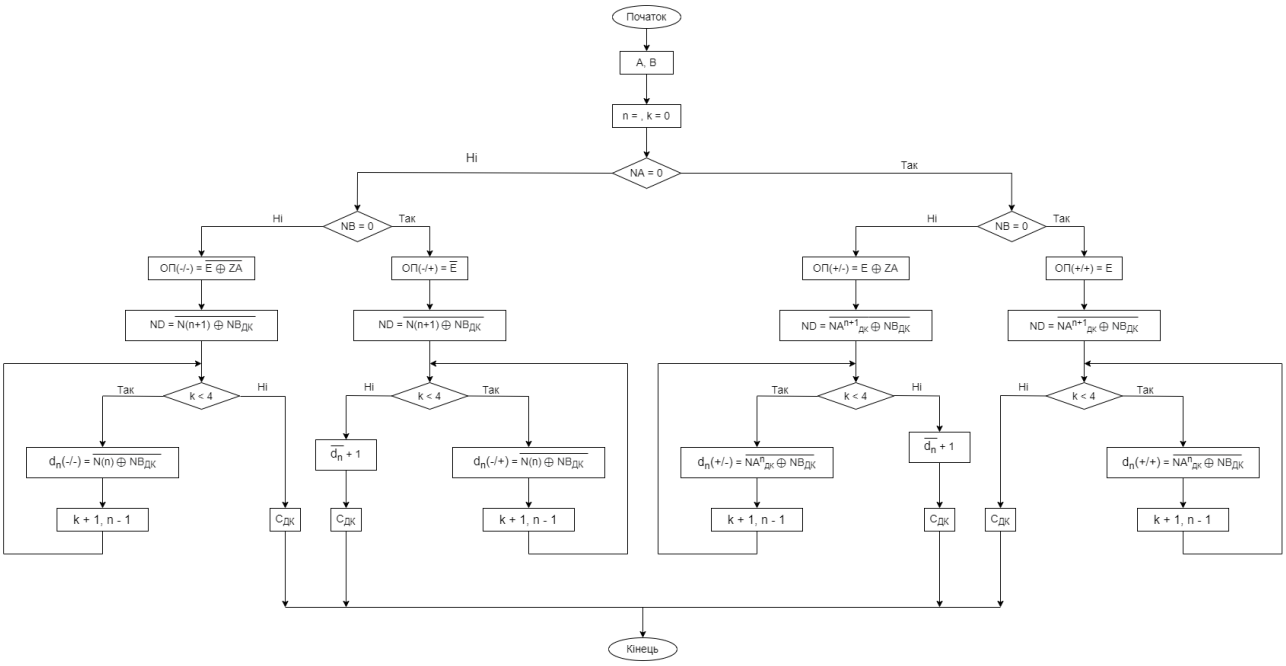


Рисунок 4.1 – Створений алгоритм для операції ділення у ДК

Зважаючи на те, що алгоритм зводиться до однотипного виконання одного з чотирьох варіантів в залежності від комбінації знаків, алгоритм, який буде

застосовано при синтезі керуючого автомату можна спростити до виконання одного шляху, а в операційному автоматі (схемі виконання арифметичної дії) в залежності від знаків виконувати варіації необхідних дій. Перетворений алгоритм зображено на рис. 4.2.

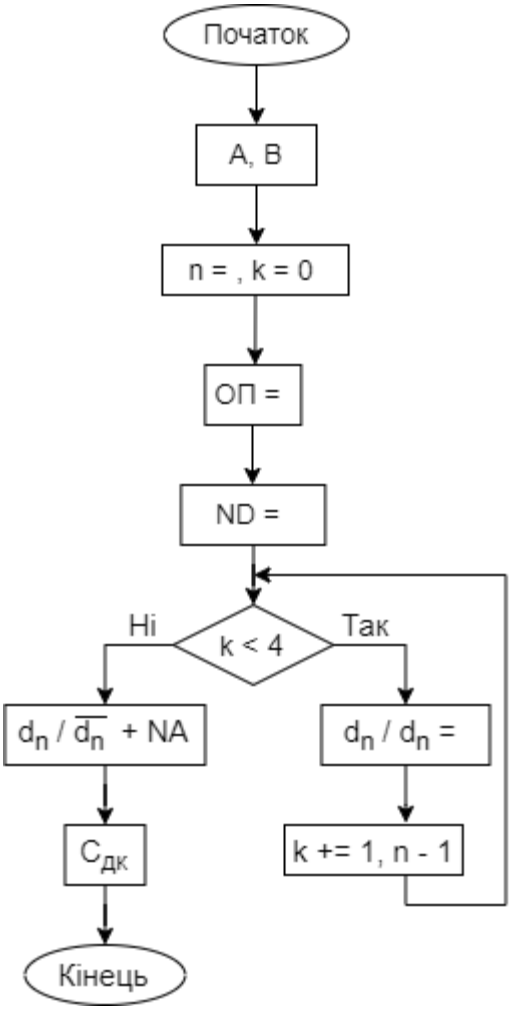


Рисунок 4.2 – Перетворений алгоритм ділення у ДК

Проведемо синтез керуючого автомата у вигляді автомата Мура. У таблиці 4.1 наведене кодування сигналів для такого автомата. Кожна пряма вершина алгоритму приймається за мікрооперацію, а умовна – за умовний перехід між станами. Мікрооперації, які можна виконати за один такт, об’єднано разом у

вихідних сигналах. У схемі буде використано регістри RGA, RGB, RGD, RGC, які будуть зберігати ділене, дільник, частку та кінцевий залишок відповідно.

Таблиця 4.1 – Кодування вершин

Код	Зміст	Примітки
Y1	RGA := 2A, RGB := B, k = 0, n =	Запис до регістрів RGA, RGB чисел 2*A, B відповідно, встановлення лічильника, визначення розрядності частки
Y2	ОП = , ND =	Визначення ознаки переповнення та знакового розряду частки
Y3	RG1 ←	Множення часткового залишку A на 2 (зсув вліво)
Y4	d_n / $\overline{d_n}$ = , n = n – 1, k = k + 1	Визначення цифрових розрядів частки, збільшення значення лічильника на 1, перехід до визначення наступного розряду частки
Y5	RGD = d_n / $\overline{d_n}$ + ND, RGC = $C_{дк}$	Корекція від'ємної частки (для випадку ND = 1 частка від'ємна, тому потребує корекції у вигляді додавання 1 до молодшого розряду, оскільки розряди формуються інверсно (у зворотному коді)) та визначення кінцевого залишку від ділення, запис результатів до відповідних регістрів
X1	k < 4	Умовна вершина: так – визначення цифрового розряду частки, ні – корекція частки та визначення кінцевого залишку

Сформовано відмічену граф-схему автомата у вигляді автомата Мура (рис. 4.3).

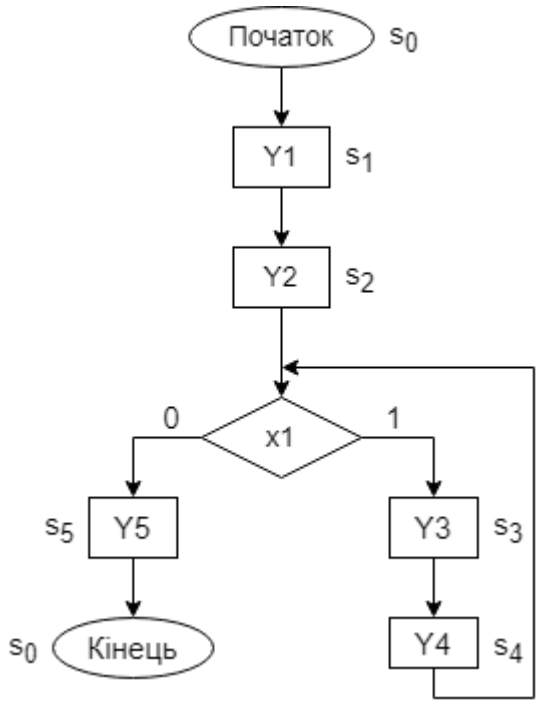


Рисунок 4.3 – Відмічена ГСА для керуючого автомата Мура

На рис. 4.4 зображено граф-схему переходів.

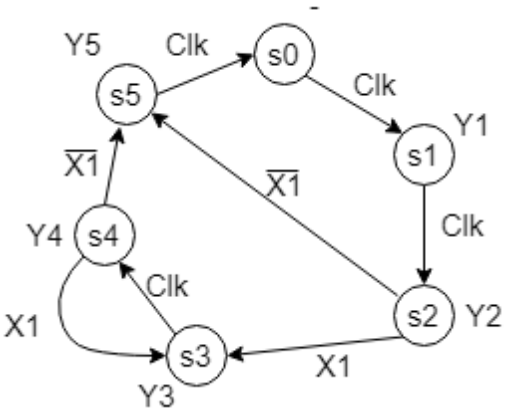


Рисунок 4.4 – ГС переходів автомата Мура

Проведемо кодування станів автомата. Застосуємо алгоритм кодування з використанням D-тригерів. Загальна кількість станів M = 6. Кількість елементів пам’яті m автомата визначається за формулою:

$$m = \log_2 M = \log_2 6 = 3. \quad (4.1)$$

Закодовані стани наведено у таблиці 4.2.

Таблиця 4.2 – Кодування станів керуючого автомата Мура

Стан автомата (s_k)	Кількість переходів (N_k)	Код К (s_k)
s_3	2	000
s_5	2	001
s_0	1	010
s_1	1	100
s_2	1	110
s_4	1	011

Використовуючи результати проведених дій, формується таблиця переходів-виходів автомата Мура (табл. 4.3).

Таблиця 4.3 – Структурна таблиця переходів-виходів керуючого автомата Мура

Початковий стан (s_k)	Код К (s_k)	Стан переходу (s_m)	Код К (s_m)	X (умова переходу)	Y (вихідний сигнал)	Функція збудження
s_5	001	s_0	010	1	–	D2
s_0	010	s_1	100	1	Y1	D1
s_1	100	s_2	110	1	Y2	D1D2
s_2	110	s_3	000	X1	Y3	–
s_4	011					
s_3	000	s_4	011	1	Y4	D2D3

Продовження таблиці 4.3

s_2	110	s_5	001	$\overline{X1}$	Y5	D3
s_4	011					

Згідно до таблиці 4.3 формується система рівнянь переходів (4.2) (має вигляд логічної суми доданків типу $s_k X$, де s_k – початковий стан, X – умова переходу) та система рівнянь виходів (4.3) (має вигляд $Y = s_m$, де s_m – кінцевий стан переходу, оскільки в автоматі Мура вихідний сигнал залежить лише від кінцевого стану):

$$\begin{cases} D1 = s_0 \vee s_1; \\ D2 = s_5 \vee s_1 \vee s_3; \\ D3 = s_3 \vee \overline{X1}(s_2 \vee s_4), \end{cases} \quad (4.2)$$

де D1, D2, D3 – активний сигнал (одиниця) відповідно на першому, другому і третьому тригері.

$$\begin{cases} Y1 = s_1; \\ Y2 = s_2; \\ Y3 = s_3; \\ Y4 = s_4; \\ Y5 = s_5; \end{cases} \quad (4.3)$$

За результатами проведеного синтезу будується логічна схема керуючого автомата. Після того, як керуючий автомат синтезовано та перевірено його роботу за допомогою симуляції, далі необхідно побудувати операційний автомат (тобто частину комбінаційної схеми, яка виконує саме операцію ділення) за початковим алгоритмом (рис. 4.1). Для цього створено функціональну схему операційної частини для подальшої її побудови (рис. 4.5).

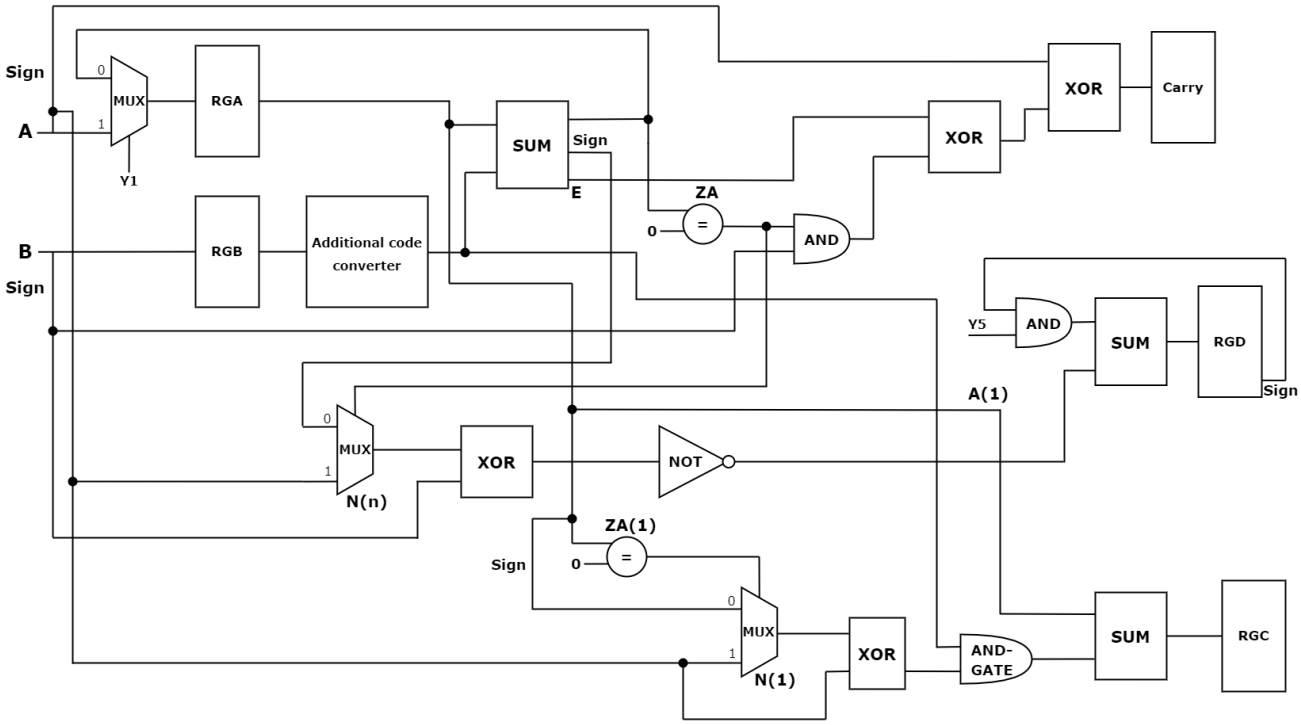
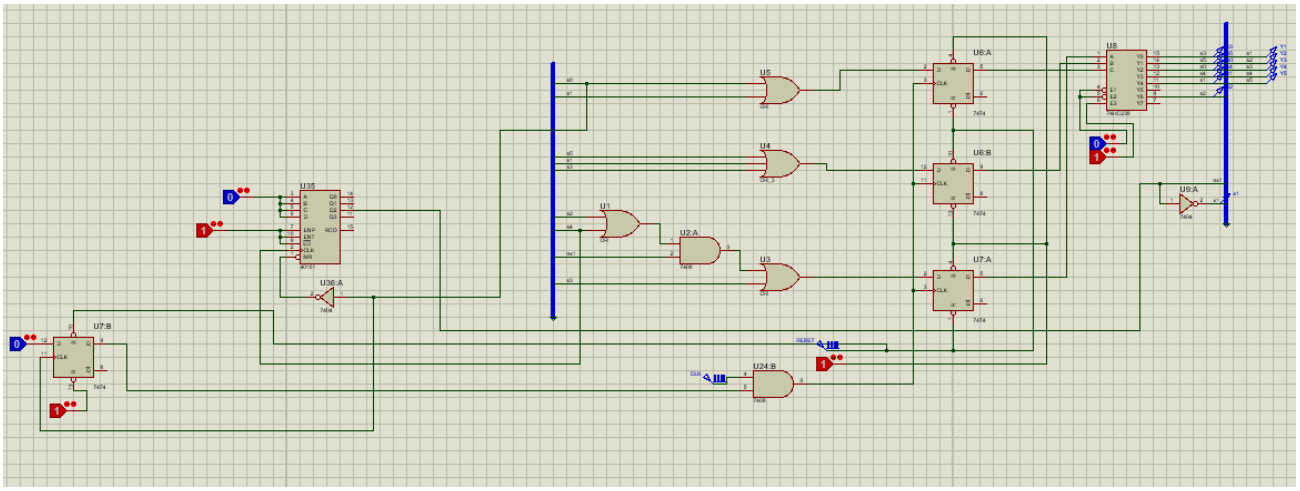


Рисунок 4.5 – Функціональна схема операційної частини пристрою ділення

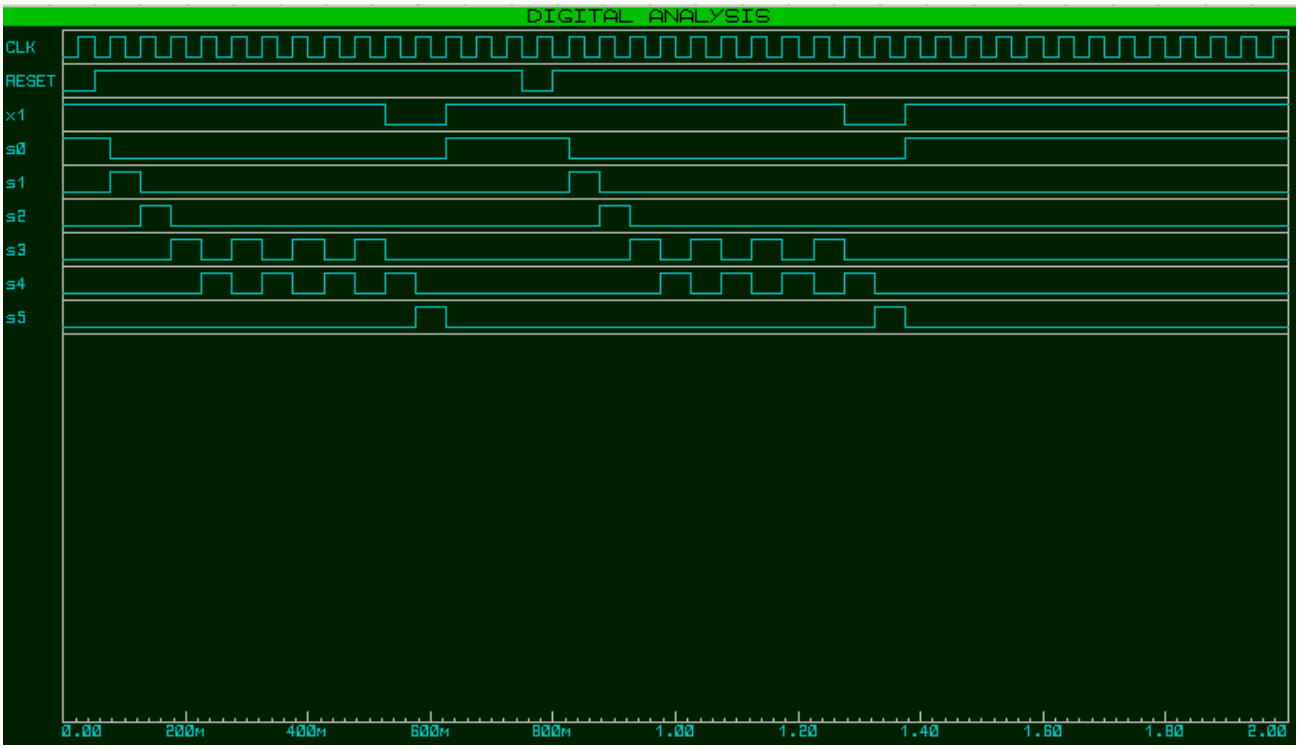
Задля отримання більш широкого спектру результатів дослідження вирішено реалізувати схему двома способами виконання: у вигляді окремої логічної схеми та схеми на базі ПЛІС.

4.2 Побудова пристрою ділення у доповняльному коді у вигляді окремої схеми

Для побудови та симуляції пристрою ділення у доповняльному коді у вигляді окремої схеми використано середовище Proteus. Спочатку, за результатами синтезу, використовуючи отримані рівняння 4.2 та 4.3 побудуємо схему керуючого автомата та перевіримо правильність його роботи. Схему та моделювання її роботи зображено на рис. 4.6 (а, б відповідно).



а)



б)

Рисунок 4.6 – Схема керуючого автомата Мура

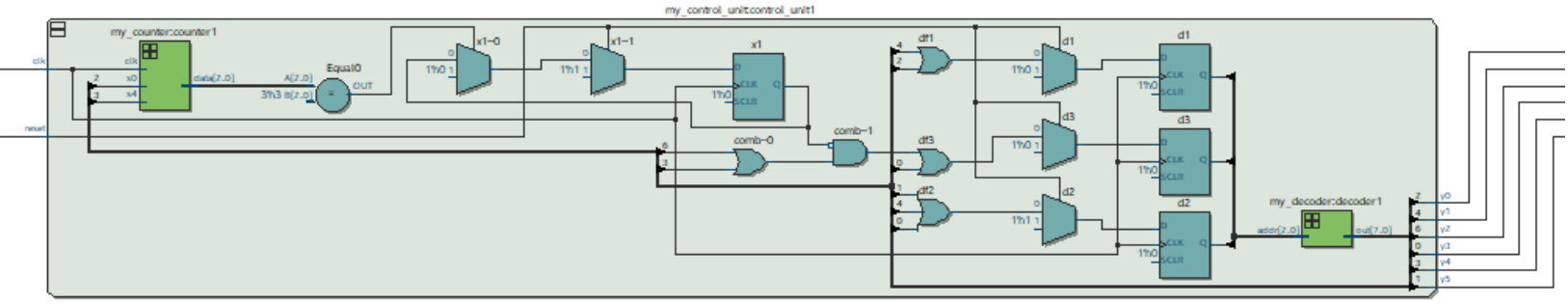
Далі будується схема операційної частини, яка в залежності від вихідних сигналів керуючого автомата та комбінації знаків діленого та дільника виконує

необхідні дії та формує результат. Реалізацію такої схеми за допомогою середовища Proteus наведено у додатку А.

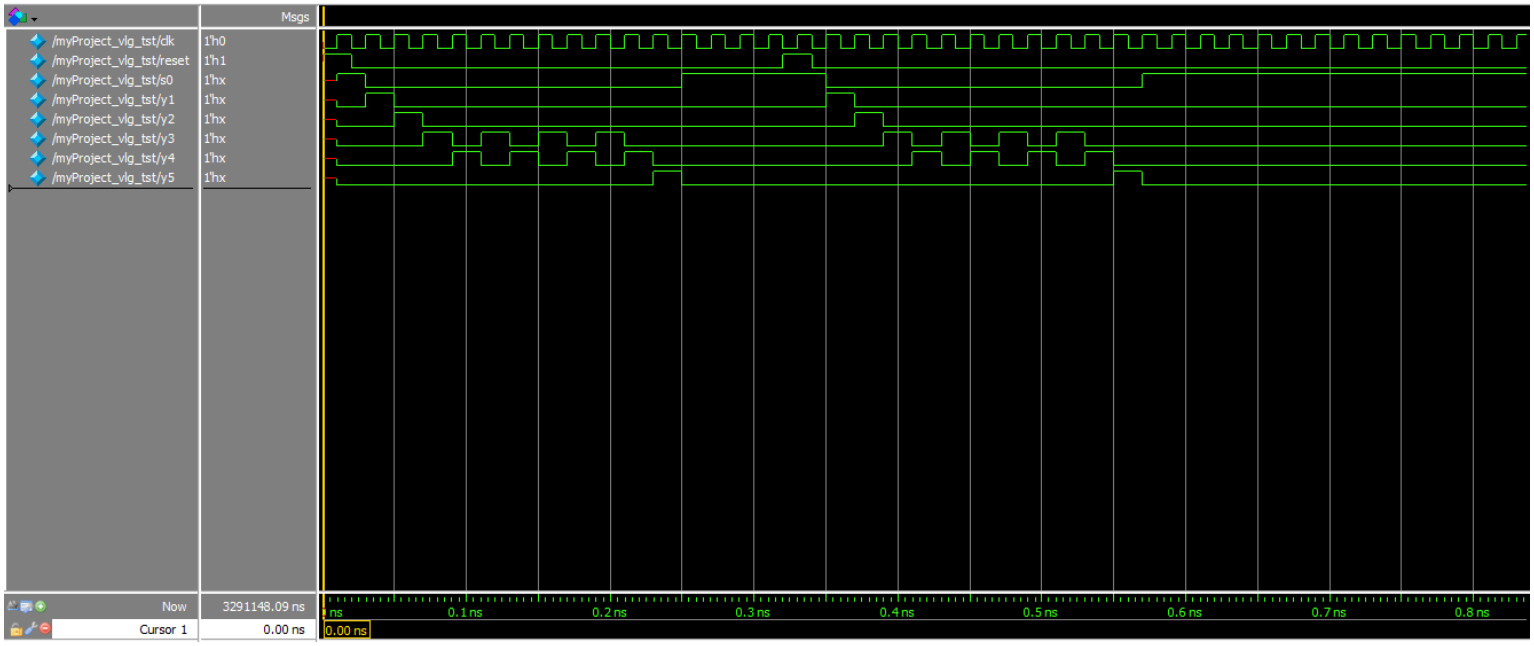
4.3 Побудова пристрою на базі ПЛІС

Подібно до алгоритму побудови у п. 4.2 будується і схема на базі ПЛІС.

Спочатку побудуємо керуючий автомат (рис. 4.7 (а)) та перевіримо правильність його роботи (рис. 4.7 (б)).



а)



б)

Рисунок 4.7 – Керуючий автомат у вигляді автомата Мура на базі ПЛІС

Далі будується операційна схема пристрою ділення. Логічну схему даної частини наведено на рис. Б.1.

Особливістю підходу роботи з ПЛІС є кардинальна відмінність у способі побудови пристрою. Логічну схему на базі FPGA і інших ПЛІС можна побудувати за допомогою коду мовою опису апаратури (в даному випадку Verilog HDL). Змінюючи цей код, можна змінювати конфігурацію схеми та модернізувати її не витрачаючи час на перебудову, як у випадку окремої схеми (якщо вона виконана у вигляді окремого кристалу тоді її взагалі неможливо змінити). Принцип написання коду для ПЛІС полягає у модульності, тобто після створення певного модуля (тобто пристрою) його можна використовувати як прототип стільки разів, скільки це необхідно. Загальна ієрархія будується таким чином, що основна побудова схеми відбувається в головному модулі – сутності вищого рівня (top-level entity). У цьому ж модулі використовуються (викликаються) інші описані модулі.

Таким чином, відокремлюючи елементи загальної схеми як окремі пристрої (модулі), отримуємо наглядний опис логічної складової побудованого пристрою (рис. 4.8). Пристрій має такі вхідні ніжки:

- operandA (ділене);
- operandB (дільник);
- clk (тактовий синхронізуючий сигнал);
- reset (сигнал скидання для повторного виконання алгоритму),

та вихідні:

- out_carry (сигнал переповнення, який означає що результат спотворено);
- out_quot_int (ціла частина від ділення);
- out_qout_rem (залишок від ділення).

Код, що описує логіку побудови пристрою ділення у ДК, та тестовий код для виконання операції наведено у додатку Б.

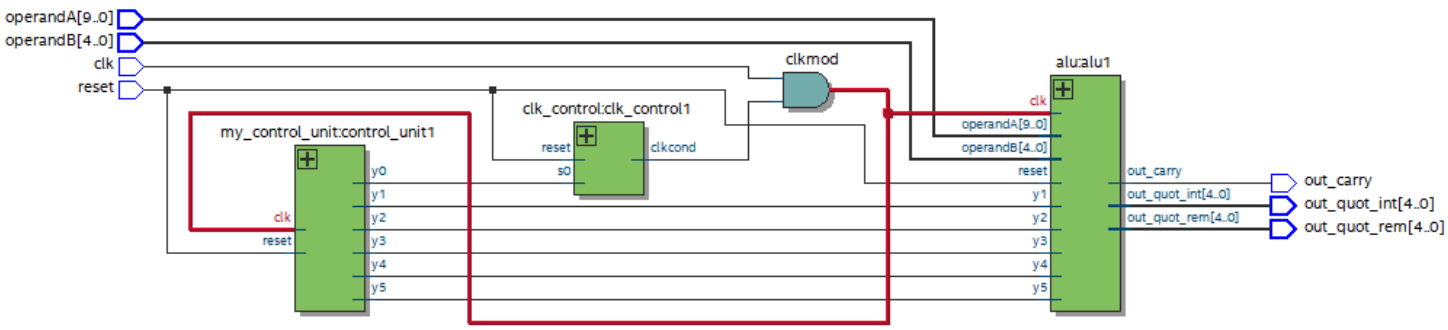


Рисунок 4.8 – Повна схема пристрою ділення у ДК на базі ПЛІС

У загальному випадку код підходить для будь-якої ПЛІС, але в даному – обрано найпростішу FPGA із сімейства Altera FPGA MAX10, а саме – MAX10 M02. Такий вибір зроблено через те, що дана схема є простою і не потребує великої кількості логічних елементів, входів/виходів або додаткових I/O інтерфейсів. Для моделювання схеми використовується фірмове програмне забезпечення Altera Quartus II від Intel, а для симуляції роботи – ModelSim.

4.4 Моделювання роботи схеми

Для того, щоб перевірити правильність роботи схеми, необхідно виконати ділення цілих чисел за класичними правилами, а потім порівняти отримані значення із результатами роботи схеми. Для прикладу візьмемо числа, під час ділення яких можна спостерігати певні особливості методу. Згідно з положеннями у Розділі 3 можливі чотири випадки:

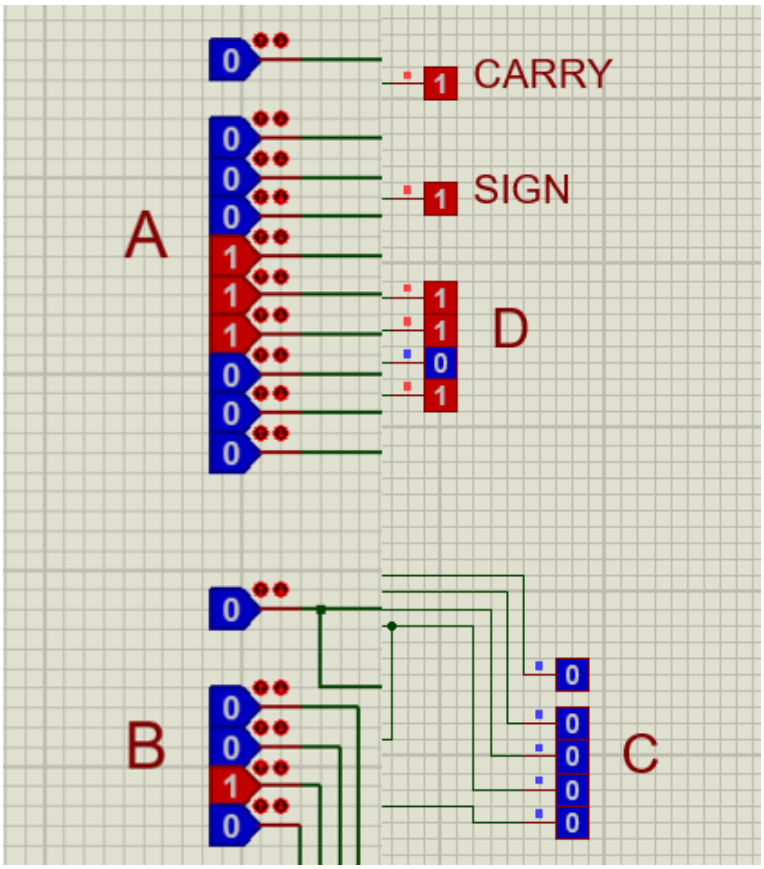
1. Ділення додатних чисел:

Поділимо число 56 на число 2. Така комбінація дозволить перевірити як працює сигналізація про переповнення, яка означає, що результат спотворено через невідповідність у розрядності чисел.

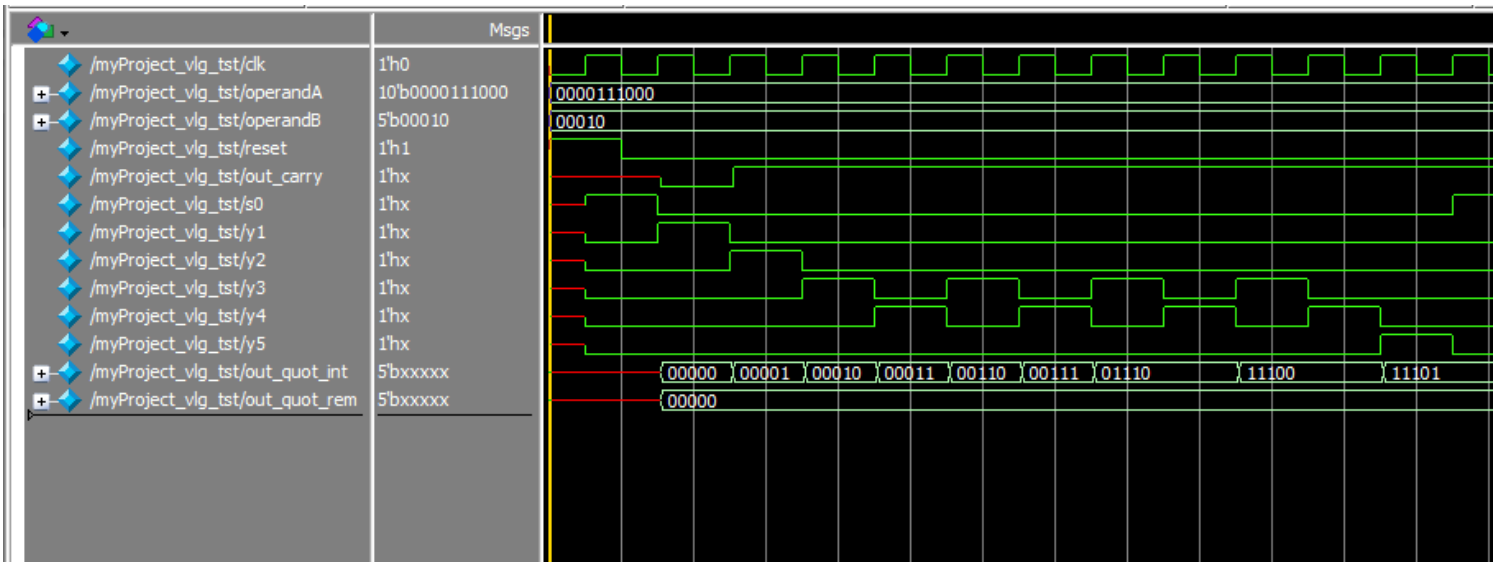
56 / 2 = 28 * 2 + 0. Дійсно, число 28 не вміщується у 4 розряди. Відповідне зображення у доповняльних кодах:

0.000111000 / 0.0010 = 0.11100 * 0.0010 + 0.0000.

На рис. 4.9 зображено результати симуляції при діленні даних чисел. Дійсно, можна побачити, що сигналізація переповнення працює вірно.



а)



б)

Рисунок 4.9 – Симуляція ділення двох додатних чисел з переповненням: а – ділення за допомогою окремої схеми; б – ділення за допомогою схеми на ПЛІС.

2. Ділення додатного діленого на від’ємний дільник:

Поділимо два числа, які не мають особливостей в процесі ділення. Наприклад, число 57 на число -6.

57 / (-6) = (-6) * (-9) + 3. Ціла частина – від’ємна, залишок – додатний. Виконаємо алгоритм покроково у доповняльних кодах:

1) 0.001110010 + 1.101000000 = 1.110110010, D = 0.0001;

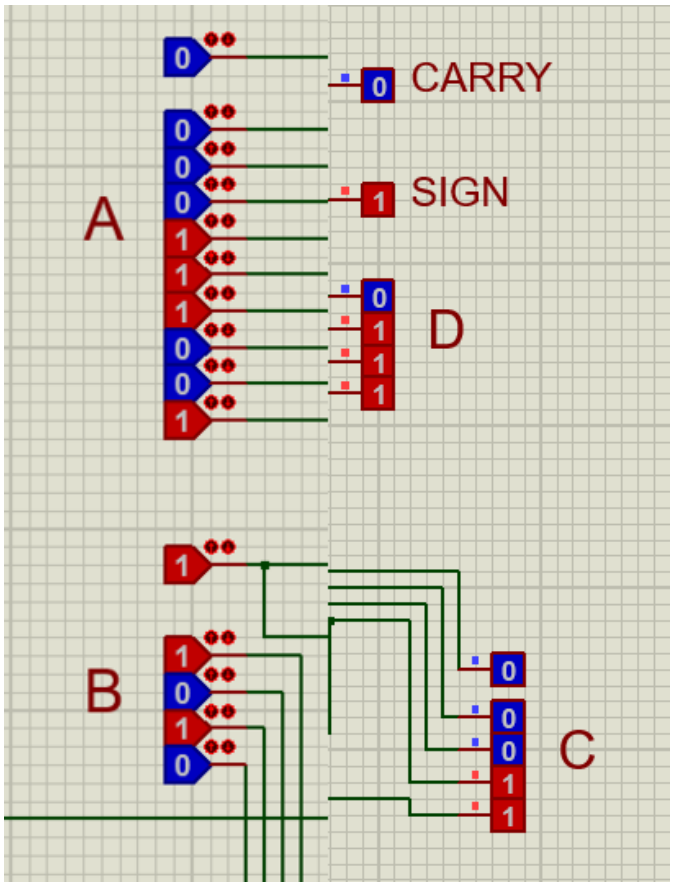
2) 1.101100100 + 0.011000000 = 0.000100100, D = 0.0010;

3) 0.001001000 + 1.101000000 = 1.110001000, D = 0.0101;

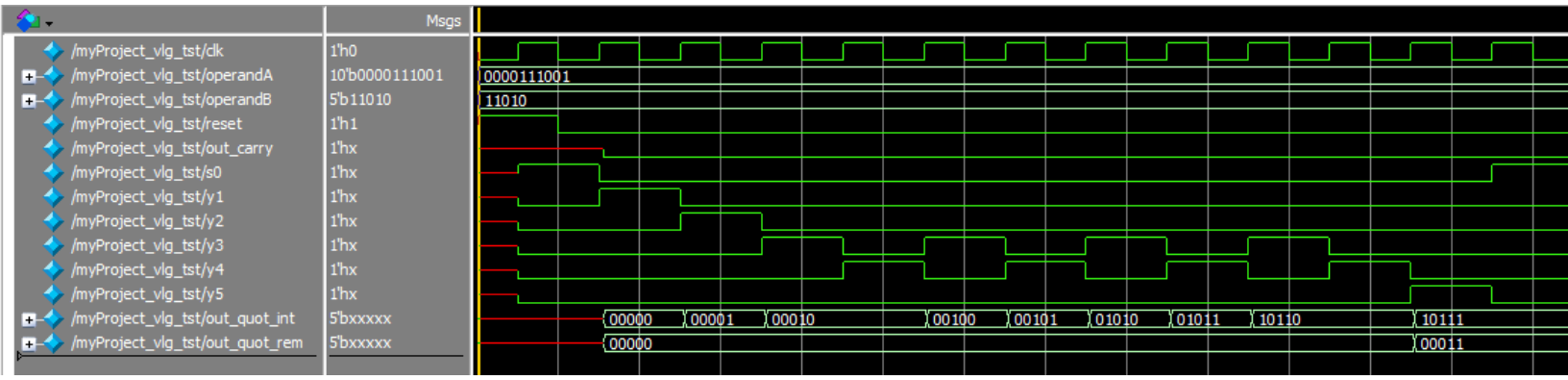
4) 1.100010000 + 0.011000000 = 1.111010000, D = 0.1011;

5) 1.110100000 + 0.011000000 = 0.001100000, D = 1.0110. Далі корекція:

NA(1) = NA, отже залишок дорівнює A(1) = 0.0011. D < 0, отже D = D + 1 = 1.0111. Відповідні результати зображено на рис. 4.10.



а)



б)

Рисунок 4.10 – Симуляція ділення додатного числа на від’ємне:

а – ділення за допомогою окремої схеми; б – ділення за допомогою схеми на ПЛІС.

3. Ділення від’ємного діленого на додатний дільник:

Поділимо число -20 на число 4. В результаті ділення отримаємо нульовий залишок від ділення. Це впливає як на формування молодшого розряду частки так і на корекцію залишку.

-20 / 4 = (-5) * 4 +- 0. Ціла частина – від’ємна, залишок – від’ємний нуль. Виконаємо алгоритм покроково у доповняльних кодах:

1) 1.111011000 + 0.010000000 = 0.001011000, D = 0.0001;

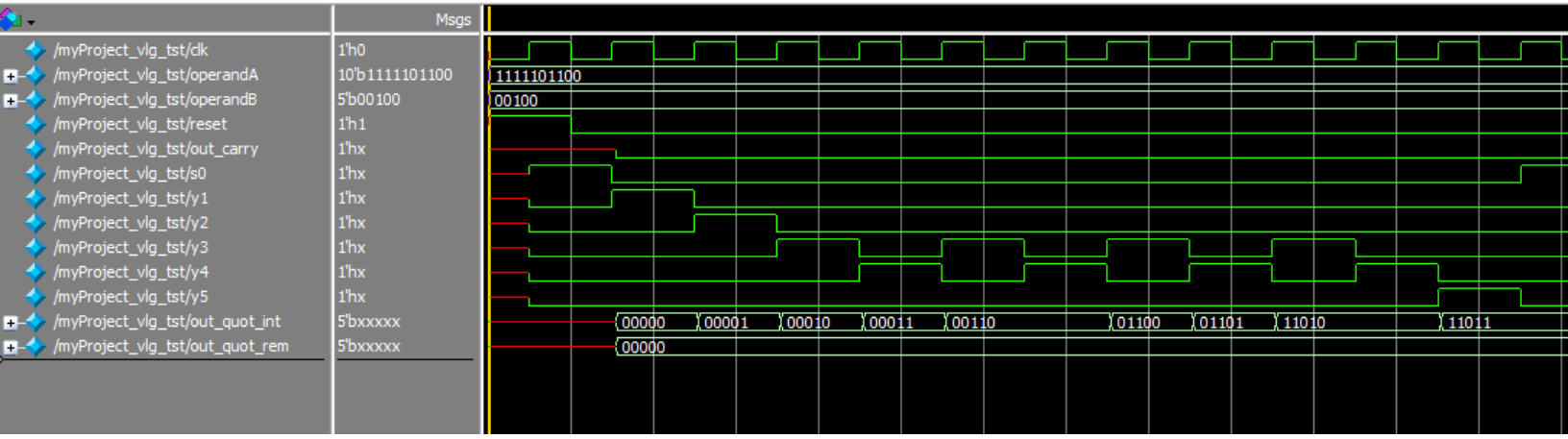
2) 0.010110000 + 1.110000000 = 0.000110000, D = 0.0011;

3) 0.001100000 + 1.110000000 = 1.111100000, D = 0.0110;

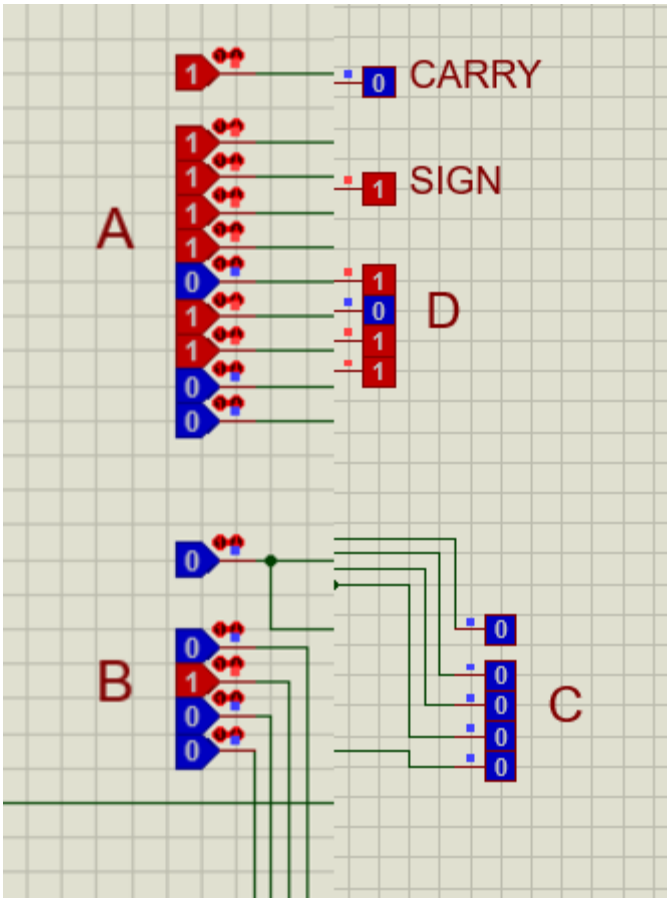
4) 1.111000000 + 0.010000000 = 0.001000000, D = 0.1101;

5) 0.010000000 + 1.110000000 = 0.000000000, D = 1.1010. Молодший розряд у даному випадку формується з використанням знаку діленого, оскільки особливістю даного методу є опис нульових залишків як позитивних і негативних нулів. Далі корекція:

N(1) = NA, отже C = A(1) = 0.0000. D < 0, отже D = D + 1 = 1.1011. Відповідні результати зображено на рис. 4.11.



а)



б)

Рисунок 4.11 – Симуляція ділення від’ємного числа на додатне:
а – ділення за допомогою схеми на ПЛІС; б – ділення за допомогою окремої схеми.

4. Ділення від’ємних цілих чисел:

Поділимо число -20 на число -5. В результаті ділення отримаємо нульовий залишок на третій ітерації. Це впливає формування цифрового розряду частки.

-20 / (-5) = (-5) * 4 +- 0. Ціла частина – додатна, залишок – від’ємний нуль. Виконаємо алгоритм покроково у доповняльних кодах:

1) 1.111011000 + 0.010100000 = 0.001111000, D = 0.0000;

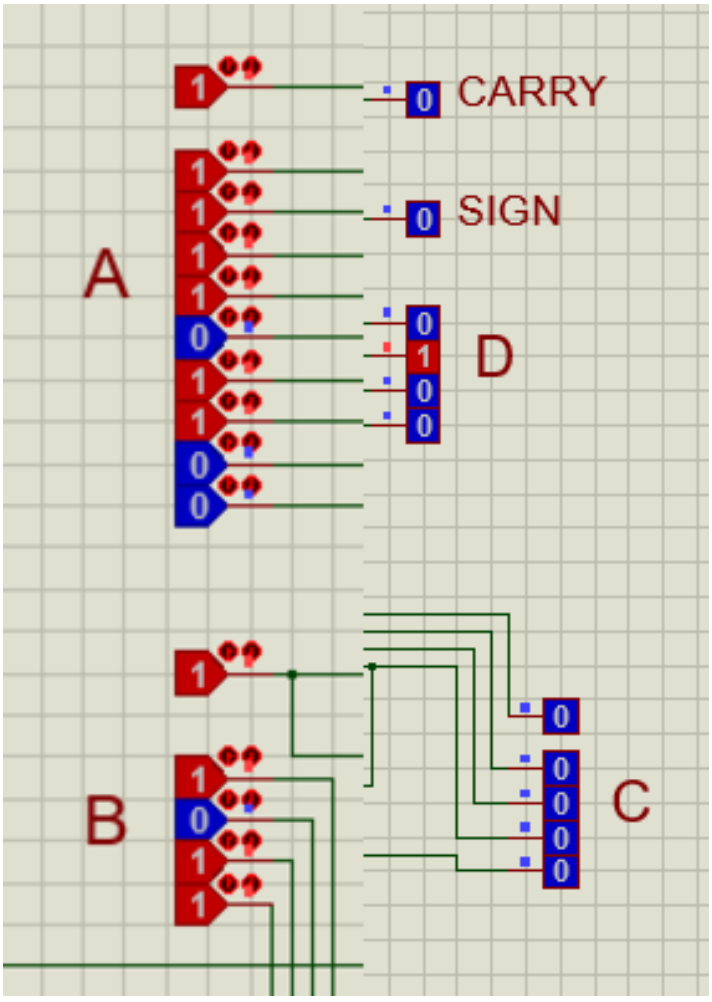
2) 0.011110000 + 1.101100000 = 0.001010000, D = 0.0000;

3) 0.010100000 + 1.101100000 = 0.000000000, D = 0.0001. Отримано нульовий залишок. Так, як ділене – від'ємне, даний залишок представляється як негативний нуль.

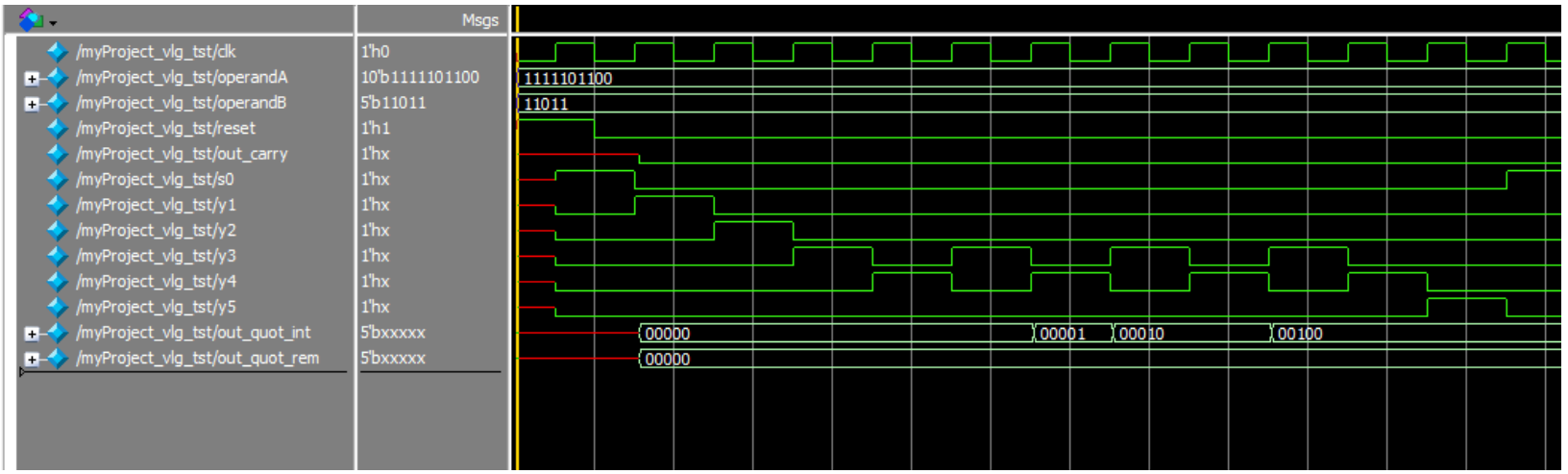
4) 0.000000000 + 0.010100000 = 0.010100000, D = 0.0010;

5) 0.101000000 + 1.101100000 = 0.010100000, D = 0.0100. Далі корекція:

N(1) не дорівнює NA, отже C = A(1) + B = 0.0101 + 1.1011 = 0.0000. D > 0, отже корекція не потрібна. Відповідні результати зображено на рис. 4.12.



а)



б)

Рисунок 4.12 – Симуляція ділення двох від’ємних чисел: а – ділення за допомогою окремої схеми; б – ділення за допомогою схеми на ПЛІС.

Порівняємо отримані результати із результатами роботи розробленої окремої схеми та схеми на базі ПЛІС. Зображені числа мають порядок бітів числа за спаданням згори до низу (зліва направо), тобто верхній (лівий) біт – старший (знаковий), нижній (правий) – молодший. Підсумкові результати роботи схеми збігаються з очікуваними. Вхідні числа та числа результату представлені у ДК. Отже, можна зробити висновки про правильність роботи схеми в обох випадках.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра проведено аналіз поточного стану щодо тематики роботи. У результаті дослідження сформовано аналітичний огляд та опис основних положень як загальноприйнятих методів комп’ютерного (арифметичного) ділення цілих чисел, так і конкретно досліджуваного – методу ділення цілих чисел в доповняльному коді із позитивним і негативним нулем залишків діленого. Сформовано мету та конкретний план роботи, для реалізації якого описано завдання, що необхідно виконати.

Проаналізовано стандартні методи ділення цілих чисел (у прямому коді) та метод, який взято за основу дослідження – метод ділення цілих чисел в доповняльному коді із позитивним і негативним нулем залишків діленого. Доведено, що метод ділення у ДК є більш практичним у порівнянні із діленням у прямому коді. За результатами аналізу алгоритм ділення у доповняльному коді із позитивним і негативним нулем залишків діленого можна поділити на чотири складові, а саме чотири різних випадки:

1) $A > 0, B > 0 \rightarrow D > 0, C > 0$;

2) $A > 0, B < 0 \rightarrow D < 0, C > 0$;

3) $A < 0, B > 0 \rightarrow D < 0, C < 0$;

4) $A < 0, B < 0 \rightarrow D > 0, C < 0$,

де A, B, D, C – відповідно знакові цілі числа діленого, дільника, частки та залишку від діленого. Кожен з наведених випадків має свої особливості щодо виконання операції. Для побудови та моделювання роботи пристрою, що виконує операцію ділення за даним алгоритмом синтезовано керуючий автомат у вигляді автомата Мура та операційний автомат, який виконує алгоритм в залежності від одного з чотирьох випадків. Практичну реалізацію пристрою наведено двома

способами реалізації: у вигляді окремої логічної схеми та схеми на базі ПЛІС. За результатами проведених досліджень можна зробити висновки щодо різних можливостей реалізації отриманої схеми:

- оскільки схема не має великої кількості логічних елементів та зв’язків, реалізація такого пристрою у вигляді окремої логічної схеми є дешевшим варіантом виконання, оскільки мікросхеми ПЛІС є порівняно дорогими для таких простих пристроїв (від 30-40$ і більше в залежності від маркетплейсу);

- в свою чергу варіативність роботи ПЛІС є великою перевагою, наприклад, для даної схеми можна за лічені хвилини змінити розрядність замінивши лише декілька рядків коду;

- також при використанні схеми на базі ПЛІС для модернізації не потрібно перебудовувати схему та докупляти нові елементи, достатньо зробити мінімальні зміни у коді і схема перебудується автоматично.

На тестових прикладах за допомогою симуляції роботи побудованого пристрою показано такі особливості досліджуваного методу:

- показано, що нульові значення залишків діленого при визначенні цифр часткового та обчисленні наступних часткових залишків діленого слід розглядати як позитивний нуль при позитивному діленому і як від'ємний нуль – при негативному діленому;

- результат ділення чисел у доповняльному коді коригується тільки за негативних значень частки;

- за відсутності переповнення знаковий та цифрові розряди доповняльного коду частки обчислюються однаковим способом.

Для неперервного багаторазового використання реалізовано зупинку роботи керуючого автомата при повторному надходженні стартового сигналу s_0 (який в свою чергу є також і кінцем алгоритму) і генератор імпульсів, який у схемі названий як Reset. Особливість полягає у тому, що при надходженні сигналу на

Reset алгоритм починає виконуватися з нуля починаючи з сигналу s_0 який до надходження імпульсу знаходиться у простої. Така конструкція дозволяє, наприклад, реалізувати пристрій ділення із спрацюванням по натиску на кнопку, або використовувати його у складі більш складних пристроїв, де необхідне багаторазове використання даної операції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Святный В.А., Лапко В.В., Самощенко А.В. Математическое описание операции деления целых чисел в дополнительном коде с отрицательным и положительным нулем остатков делимого: Наукові праці ДонНТУ №2 (23), 2016 (Серія “Інформатика, кібернетика та обчислювальна техніка”). УДК 004.315, ISSN 1996-1588. с. 22-29.
2. Бабаков Р. М. Операционное преобразование кодов состояний в микропрограммном автомате: монография. Винница: «ТВОРИ», 2019. 208 с.
3. Баркалов, А.А., Титаренко Л.А., Зеленева И.Я. Реализация совмещенного микропрограммного автомата в базисе FPGA: Наукові праці ДонНТУ. Красноармійськ: ДВНЗ «ДонНТУ», 2015. с. 84-88.
4. Гриценко А.А., Зеленева И.Я., Сероштан С.Ю., Татолов Е.Р. Методика исследования способов оптимизации управляющих автоматов в базисе ПЛИС FPGA: Радіоелектроніка, інформатика, управління. 2012 №2 (“Прогресивні інформаційні технології”). УДК 004.2, ISSN 1607-3274. с. 141-145.
5. Зацерковний В.І. Обчислювальна техніка: історія розвитку від найпростіших пристроїв для лічби до електромеханічних комп’ютерів: монографія. Ніжин: Аспект-Поліграф, 2012. 416 с.
6. Кононюк А.Е. Дискретно-непрерывная математика. Книга 11. Автоматы. Часть 2. Детерминированные автоматы. Киев: Освіта України, 2017. 578 с.
7. Матвієнко М.П. Архітектура комп'ютерів: навчальний посібник. Київ: ТОВ "Центр навчальної літератури", 2012. 264 с.
8. Матвієнко М.П. Комп'ютерна логіка: навчальний посібник. Київ: ТОВ "Центр навчальної літератури", 2012. 288 с.
9. Мірошник М. А. Технології та автоматизація проектування цифрових пристроїв складних комп’ютерних систем на ПЛІС: навч. посібник / Мірошник М. А., Клименко Л. А., Корольова Я. Ю – Харків: УкрДУЗТ, 2021. – 221 с.
10. Тарарака В. Д. Прикладна теорія цифрових автоматів: навчальний посібник. Житомир: ЖДТУ, 2019. 183 с.

11. Barkalov A.A., Titarenko L.A., Barkalov A.A. Structural decomposition as a tool for the optimization of an FPGA-based implementation of a mealy FSM. Cybern Syst Anal. 2012. Vol. 48, Pp. 313-322.
12. Digital Computer Arithmetic: Design and Implementation by Joseph J. F. Cavanaugh. McGraw-Hill, 1984. 468 pp.
13. Salcido A. Cellular Automata - Simplicity Behind Complexity. InTech, 2011. 580 pp.
14. Solov'ev V.V. Implementation of finite-state machines based on programmable logic ICs with the help of the merged model of Mealy and Moore machines. J. Commun. Technol. Electron. 2013. Vol. 58, Pp. 172-177.
15. Каневский Ю. С. Компьютерная арифметика URL: https://kanyevsky.kpi.ua/wp-content/uploads/2017/09/Arithmet_A5.pdf
16. Косолап А. І. Форми представлення чисел у пам'яті комп'ютера. Дніпропетровський національний університет. URL: https://www.sworld.com.ua/konfer24/503.htm
17. Подання цілих додатних чисел. Прямий код. Доповняльний код. URL: https://goo.su/9ghY
18. Фон-нейманівська архітектура обчислювальної машини. URL: https://dl.khadi.kharkov.ua/pluginfile.php/175764/mod_resource/content/1/%D0%9B%D0%BA1%20%D0%A4%D0%BE%D0%BD%20%D0%BC%D0%B0%D1%88_%D0%B0%D1%80%D0%B8%D1%84_%D1%83%D0%BA%D1%80%202021.pdf
19. Binary Multiplication & Division. URL: https://www.cs.utah.edu/~rajeev/cs3810/slides/3810-08.pdf
20. Intel MAX10 FPGA. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/br/br-max10-brochure.pdf

ДОДАТОК А

Практична реалізація алгоритму ділення у доповняльному коді у вигляді окремої логічної схеми

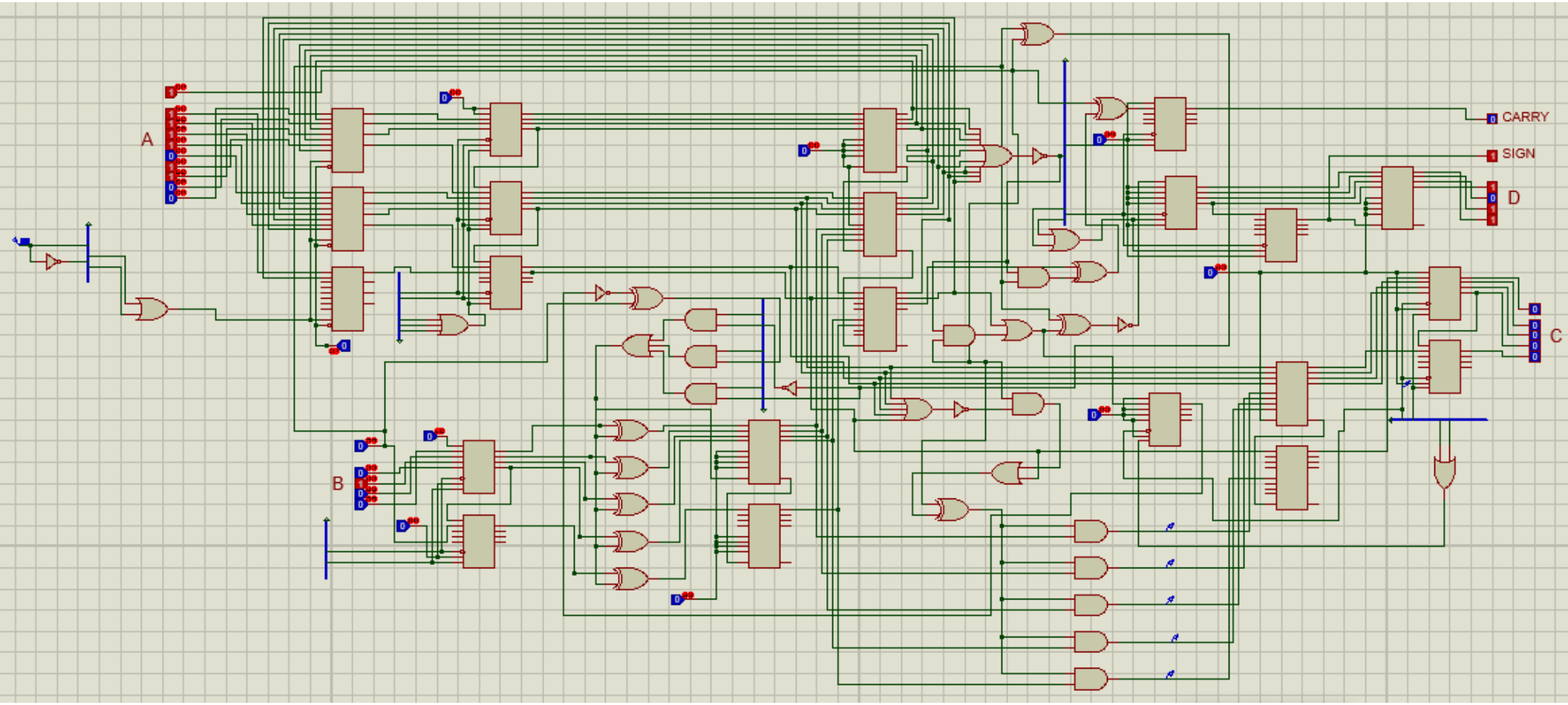


Рисунок А.1 – Операційна частина пристрою ділення

Продовження додатку А

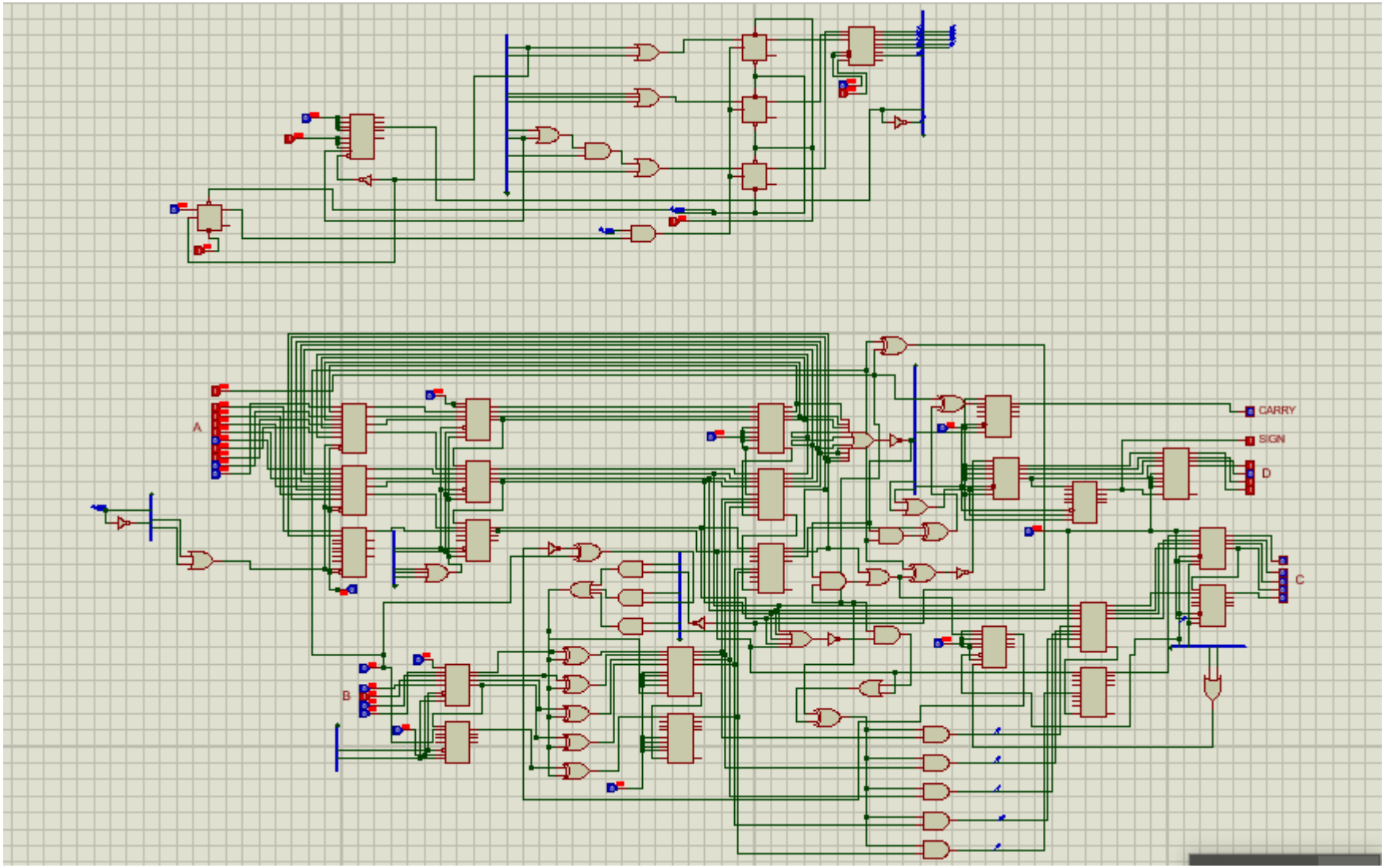


Рисунок А.2 – Загальна логічна схема пристрою ділення у ДК

ДОДАТОК Б

Побудова пристрою ділення на базі ПЛІС

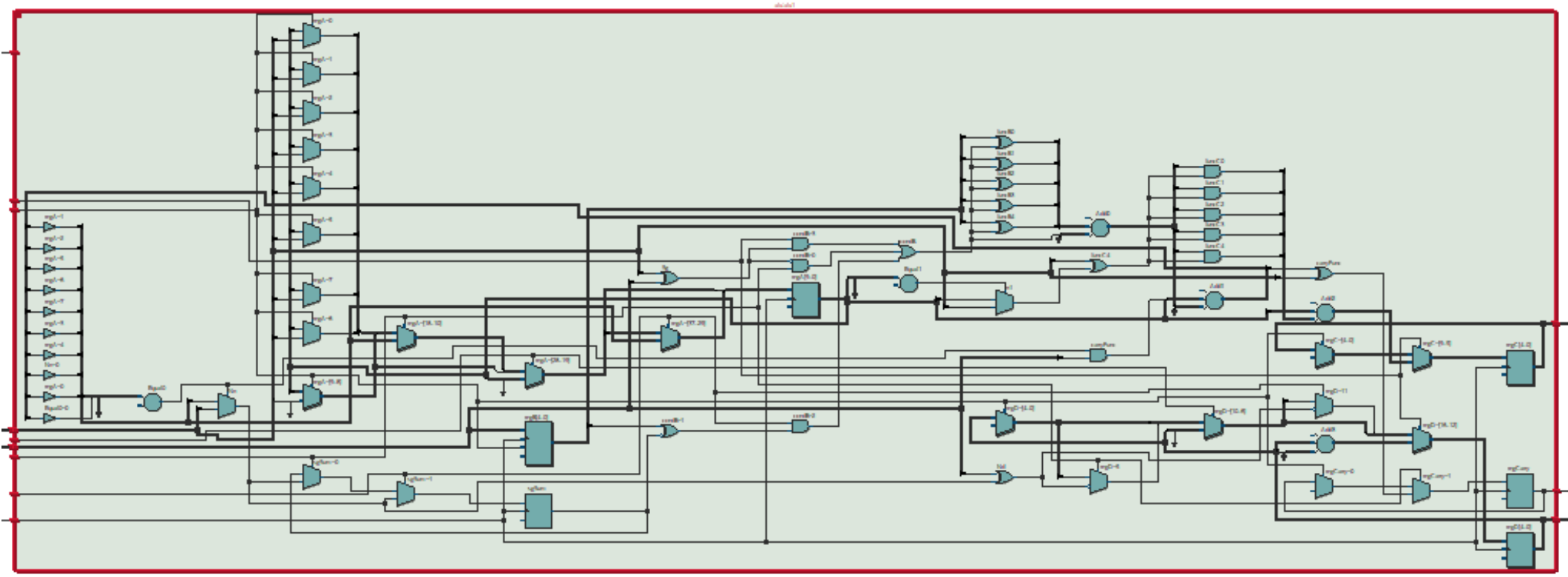


Рисунок Б.1 – Операційна частина пристрою ділення на базі ПЛІС.

Код для формування схеми пристрою ділення

```
`timescale 1 ps/ 1 ps

// ---------Модуль верхнього рівня, тобто основний модуль----------------

module myProject (
        input [9:0] operandA, input [4:0] operandB, input clk, reset,
        output [4:0] out_quot_int, out_quot_rem, output out_carry);


reg clkmod;
wire clkcond;
wire s0, y1, y2, y3, y4, y5;

clk_control clk_control1 (.reset(reset), .s0(s0), .clkcond(clkcond));
always @(clk) begin
    clkmod <= clk & clkcond;
end

my_control_unit control_unit1 (.clk(clkmod), .reset(reset), .y0(s0), .y1(y1),
.y2(y2), .y3(y3), .y4(y4), .y5(y5));
alu alu1 (.operandA(operandA), .operandB(operandB), .clk(clkmod), .reset(reset),
.y1(y1), .y2(y2), .y3(y3), .y4(y4), .y5(y5), .out_quot_int(out_quot_int),
.out_quot_rem(out_quot_rem), .out_carry(out_carry));

endmodule

module clk_control(
    input reset, s0,
    output wire clkcond);

reg clkexp = 1;
always @(posedge s0 or posedge reset) begin
    if (reset) begin
        clkexp <= 1;
    end else begin
        clkexp <= 0;
    end
end
assign clkcond = clkexp;

endmodule

// --------------Керуючий автомат-----------------------------

module my_control_unit(
    input clk, reset,
    output wire y0, y1, y2, y3, y4, y5
);

wire s0, s1, s2, s3, s4, s5;

reg d1, d2, d3, x1;
reg [2:0] counter;
wire [2:0] data;
```

```
wire df1 = s0 | s1;
wire df2 = s1 | s3 | s5;
wire df3 = s3 | (~x1 & (s2 | s4));

my_decoder decoder1 (.addr({d1, d2, d3}), .out({z, s2, z, s1, s4, s0, s5, s3}));

my_counter counter1 (.clk(clk), .s0(s0), .s4(s4), .data(data));

always @(posedge clk) begin
counter = data;
if (reset) begin
x1 <= 1;
d1 <= 0;
d2 <= 1;
d3 <= 0;
end
else begin
if (counter == 3'd3) begin
x1 = 0;
end
d1 <= df1;
d2 <= df2;
d3 <= df3;
end
end

assign y0 = s0;
assign y1 = s1;
assign y2 = s2;
assign y3 = s3;
assign y4 = s4;
assign y5 = s5;

endmodule


// Лічильник для керуючого автомата

module my_counter(
    input clk, s0, s4,
    output [2:0] data);

reg [2:0] counter;
assign data = counter;

always @(posedge clk) begin
    if (s0) begin
        counter = 0;
    end if (s4) begin
        counter = counter + 1;
    end
end
endmodule


// Декодер
```

```
module my_decoder(
  input wire [2:0]addr,
  output reg [7:0]out
);

always @*
begin
    case( addr )
    3'd0: out = 8'b00000001;
    3'd1: out = 8'b00000010;
    3'd2: out = 8'b00000100;
    3'd3: out = 8'b00001000;
    3'd4: out = 8'b00010000;
    3'd5: out = 8'b00100000;
    3'd6: out = 8'b01000000;
    3'd7: out = 8'b10000000;
    endcase
end
endmodule



// -------Операційна частина пристрою-----------------------

module alu(
    input [9:0] operandA, input [4:0] operandB,
    input clk, reset, input wire y1, y2, y3, y4, y5,
    output [4:0] out_quot_int, out_quot_rem, output out_carry/*, output [10:0]
outAw*/);

// Декларування провідників та змінних

wire clkw = clk;
reg clk1;

reg [9:0] regA;
reg [4:0] regB, regC, regD;
reg regCarry;
wire [10:0] sumAB;
wire Za, z1;
reg Nn, n1;
wire Nd, carryFunc, condB;
reg sgSum;
wire invB;
wire funcB0, funcB1, funcB2, funcB3, funcB4;
wire funcC0, funcC1, funcC2, funcC3, funcC4;
wire [9:0] funcBmul;
wire [4:0] funcB, funcC;

// Функції та дроти для з'єднання

wire Sg = operandA[9] ^ operandB[4];

assign invB = ~sgSum;
assign condB = (y2 & ~Sg) | (y4 & (regB[4] ^ invB)) | (y5 & Sg);
assign funcB0 = (condB ^ regB[0]);
assign funcB1 = (condB ^ regB[1]);
```

```
assign funcB2 = (condB ^ regB[2]);
assign funcB3 = (condB ^ regB[3]);
assign funcB4 = (condB ^ regB[4]);
assign funcB = {funcB4, funcB3, funcB2, funcB1, funcB0} + condB;
assign funcBmul  = {funcB, 5'b00000};

assign sumAB = regA[9:0] + funcBmul;

assign Za = sumAB[9:0] == 0;
always @* begin
    if (Za) begin
        Nn = operandA[9];
    end else begin
        Nn = sumAB [9];
    end
end

assign carryFunc = operandA[9] ^ (sumAB[10] ^ (operandB[4] & Za));
assign Nd = ~(Nn ^ operandB[4]);

assign z1 = regA[9:0] == 0;
always @* begin
    if (z1) begin
        n1 = operandA[9];
    end else begin
        n1 = regA [9];
    end
end

assign funcC0 = (funcB[0] & (n1 ^ operandA[9]));
assign funcC1 = (funcB[1] & (n1 ^ operandA[9]));
assign funcC2 = (funcB[2] & (n1 ^ operandA[9]));
assign funcC3 = (funcB[3] & (n1 ^ operandA[9]));
assign funcC4 = (funcB[4] & (n1 ^ operandA[9]));
assign funcC = regA[9:5] + {funcC4, funcC3, funcC2, funcC1, funcC0};

// Основні процеси синхронного виконання алгоритму

always @(clk) begin
    #1;
    clk1 = clkw;
end

always @(posedge clk1) begin
    if (y1) begin
    regA [0] <= 0;
    regA[9:1] <= operandA[8:0];
    regB <= operandB;
    regD <= 5'b00000;
    regC <= 5'b00000;
    regCarry <= 0;
end if (y2) begin
    regA[9:1] <= sumAB[9:1];
    sgSum <= Nn;
    regD[0] <= Nd;
    regCarry <= carryFunc;
end if (y3) begin
    regA <= regA << 1;
```

```
    regD <= regD << 1;
end if (y4) begin
    regA[9:1] <= sumAB[9:1];
    sgSum <= Nn;
    regD[0] <= Nd;
end if (y5) begin
    regC <= funcC;
    regD <= regD + regD[4];
end
end

// Присвоювання виходів

assign out_carry = regCarry;
assign out_quot_int = regD;
assign out_quot_rem = regC;

endmodule
```

Код для проведення симуляції

```
`timescale 1 ps/ 1 ps
module myProject_vlg_tst();

reg clk;
reg [9:0] operandA;
reg [4:0] operandB;
reg reset;


wire out_carry;
wire [4:0] out_quot_int, out_quot_rem;

// assign statements (if any)
myProject i1 (
// port map - connection between master ports and signals/registers
    .clk(clk),
    .operandA(operandA),
    .operandB(operandB),
    .reset(reset),
    .out_carry(out_carry),
    .out_quot_int(out_quot_int),
    .out_quot_rem(out_quot_rem)
);
initial begin
    clk = 0;
    forever begin
        #10 clk = ~clk;
    end
end
initial begin
    reset = 1;
    #20;
    reset = 0;
    #300;
```

```
    reset = 1;
    #20;
    reset = 0;
end

initial begin
    operandA = 10'b1111101100; // -20
    operandB = 5'b11011;       // -5
end

always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin

// --> end
end
endmodule
```