

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Комп'ютерні системи та мережі

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему Розробка підсистеми тренувальної лабораторії Renesas Electronics

Виконав: студент 4 курсу, групи КНТ-519
Спеціальності 123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Комп'ютерна інженерія»

КУХТІН К. І.

(прізвище та ініціали)

Керівник ЗЕЛЕНЬОВА І. Я.

(прізвище та ініціали)

Рецензент КАЧАН О. І.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри КУДЕРМЕТОВ Р.К.
 « » 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КУХТІНА Костянтина Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка підсистеми тренувальної лабораторії Renesas Electronics

керівник проєкту (роботи) доцент, ЗЕЛЕНЬОВА Ірина Яківна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від «24» березня 2023 року № 71

2. Строк подання студентом проєкту (роботи) 10 травня 2023 року

3. Вихідні дані до проєкту (роботи) Go Configure Software Hub

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз технічного завдання, дослідження наявних у вільному доступі віртуальних лабораторій різних ІТ компаній;

2) Розробка алгоритму та структури цифрового пристрою;

3) Моделювання та тестування розробленого цифрового пристрою.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Алгоритм 1 – Змістовий граф алгоритму мікрообчислювача

Алгоритм 2 – Розмічена граф-схема алгоритму мікрообчислювача

Плакат 1 – Приклад тестування пристроїв

Плакат 2 – Операційний і керуючий автомати. Схеми електричні функціональні

Плакат 3 – Інструменти для налаштування тестування

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ЗЕЛЕНЬОВА І. Я., доцент		
нормоконтроль	ЩЕРБАК Н. В., ст. викл.		

7. Дата видачі завдання 04.03.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз технічного завдання та наявних у вільному доступі віртуальних лабораторій різних ІТ компаній	03.04.2023 р.	
2	Розробка алгоритму пристрою	08.05.2023 р.	
3	Розробка структурних елементів пристрою	10.05.2023 р.	
4	Моделювання структурних елементів пристрою	12.05.2023 р.	
5	Тестування структурних елементів пристрою	13.05.2023 р.	
6	Оформлення графічного матеріалу	15.05.2023 р.	
7	Оформлення пояснювальної записки	05.06.2023 р.	

Студент (ка)


(підпис)

Костянтин КУХТІН
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Ірина ЗЕЛЕНЬОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 68 с; 46 рис; 31 табл; 10 джерел.

RENESAS ELECTRONICS, ВІРТУАЛЬНА ЛАБОРАТОРІЯ, КЕРУЮЧИЙ АВТОМАТ, МІКРООБЧИСЛЮВАЧ, ОПЕРАЦІЙНИЙ АВТОМАТ, ТЕСТУВАННЯ.

Об'єкт дослідження – віртуальна тренувальна лабораторія.

Предмет дослідження – віртуальні тренувальні лабораторії різних ІТ компаній.

Мета роботи – створення віртуальної тренувальної лабораторії на базі програмного забезпечення компанії Renesas Electronics.

Матеріали, методи та технічні засоби: програмний пакет Go Configure Software Hub, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено основу для подальшої розробки віртуальної тренувальної лабораторії на базі програмного забезпечення компанії Renesas Electronics.

Висновки. Розроблено мікрообчислювач, процес проектування якого буде слугувати прикладом роботи у програмному пакеті Go Configure Software Hub.

Галузь використання – освіта.

ЗМІСТ

Вступ	6
1 Аналіз предметної області	7
1.1 Аналіз загальнодоступних віртуальних лабораторій.....	7
1.2 Короткий огляд засобів проєктування компанії Renesas	8
1.3 Теоретичне підґрунтя для синтезу цифрових пристроїв	10
1.4 Постановка задачі проєктування.....	12
2 Основні етапи проєктування цифрового пристрою	13
2.1 Розробка операційного автомата	14
2.2 Розробка керуючого автомата	16
2.3 Розробка компаратора	22
3 Моделювання і тестування цифрового пристрою у пакеті Go Configure Software Hub.....	26
3.1 Моделювання операційного автомата.....	26
3.2 Моделювання керуючого автомата	39
3.3 Моделювання компараторів	44
3.4 Тестування змодельованих пристроїв	49
Висновки.....	62
Перелік джерел посилання	63

ВСТУП

У сучасному світі, завдяки швидкому технічному прогресу та розвитку комунікацій, абсолютна більшість людей має доступ до мережі інтернет. Завдяки цьому людина може отримувати доступ майже до будь-якої інформації з майже будь-якої точки нашої планети. Те саме стосується і навчальних матеріалів. Тож завдяки мережі Інтернет стало реальним здобуття освіти у дистанційному форматі.

Одним із видів такого дистанційного навчання стали онлайн курси, спрямовані на вивчення широкого спектра різних освітніх програм. Такі курси здебільшого складаються з онлайн лекцій та домашніх завдань які має виконувати здобувач освіти. І якщо для вивчення, наприклад, якої-небудь іноземної мови такого може бути достатньо, то деякі спеціальності потребують великої кількості саме практики. І тому для цих спеціальностей краще підійде інший вид дистанційного навчання, а саме, навчання на базі віртуальних тренувальних лабораторій.

Віртуальні тренувальні лабораторії використовують для ознайомлення із приладами або програмним забезпеченням, які доступні у цій лабораторії, не тільки майбутніх співробітників компанії, а й клієнтів. Ці лабораторії здебільшого мають вигляд вебресурсів, через які й відбувається взаємодія користувача з обраним приладом або програмним забезпеченням. І оскільки це вебресурс це означає, що отримати доступ до нього може майже будь-хто, не зважаючи на місце знаходження та обладнання користувача, достатньо мати лише доступ до мережі Інтернет. Основна перевага віртуальної лабораторії полягає у тому, що здебільшого навчання відбувається безпосередньо на практиці. До того ж за приклади беруть задачі максимально наближені до тих, які можуть виникнути під час роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз загальнодоступних віртуальних лабораторій

1.1.1 The Grid of Online Laboratory Devices Ilmenau

The Grid of Online Laboratory Devices Ilmenau, являє собою віртуальну лабораторію, створену корпорацією Technische Universität Ilmenau. В цій лабораторії користувач може спроектувати та протестувати власні алгоритми керування для віртуальних та реальних фізичних пристроїв. Алгоритми керування можуть бути реалізовані у вигляді: Булевих функцій кінцевих автоматів, реального обладнання або програм для мікроконтролерів що доступні у лабораторії [1]. Спочатку користувач тестує свій алгоритм у симуляції пристрою для виявлення більшості помилок, і звідси у користувача в будь-який момент часу є можливість внести зміни у разі виникнення. Після завершення тестування у симуляції настає етап тестування вже з використанням реального пристрою, для того, щоб виявити помилки які не відтворюються у симуляції.

1.1.2 Virtual FPGA Lab

Virtual FPGA Lab – це повністю віртуальна лабораторія, створена на основі вебзастосунку Makerchip, призначена для навчання програмуванню FPGA плат. У цій лабораторії реалізовано симуляції як самих FPGA, так і їх базову периферію, такі як діоди, семисегментні дисплеї й т.д. Це служить для кращого розуміння результатів роботи введеної програми. Також серед особливостей зазначається наявність візуального дебагінгу. Він спрощує пошук помилок шляхом візуалізації операцій які відбуваються у ході програми [2].

1.1.3 FPGA Laboratory

FPGA Laboratory – це віртуальна лабораторія розміщена на вебресурсі LabsLand і надає користувачам можливість взаємодіяти із реальними FPGA платами, розміщеними у чотирьох різних університетах. Кожна FPGA обладнана десятьма діодами, шістьма семисегментними дисплеями, десятьма перемикачами та чотирма кнопками. Спочатку користувач на мові опису

апаратури, VHDL або Verilog, пише програму, після чого може відіслати цей код на вільну FPGA і коли код завантажиться, відкривається екран, де демонструється сама FPGA, зображення якої транслюється у прямому ефірі, та інтерактивні об'єкти через які відбувається взаємодія із перемикачами та кнопками розташованими на платі. Завдяки цьому неважливо який у користувача пристрій та чи встановлено на ньому спеціалізоване ПЗ. До того ж тестування роботи програми проводиться одразу на реальному пристрої [3].

1.2 Короткий огляд засобів проєктування компанії Renesas

Renesas Electronics Corporation пропонує надійні, вбудовані інноваційні розробки технологій із повними напівпровідниковими рішеннями, які включають мільярди підключених цифрових пристроїв, призначених для покращення побуту та роботи людей. Глобальний лідер у сфері виготовлення мікроконтролерів, аналогових, силових та SoC виробів, Renesas забезпечує комплексні рішення для широкого асортименту індустріальної, автомобільної та домашньої електроніки, автоматизацію офісу та застосування інформаційно-комунікаційної технології [4].

Основним засобом проєктування цієї компанії є пакет Go Configure Software Hub. У цьому пакеті реалізовано графічний спосіб розробки проєктів, завдяки чому немає потреби у використанні яких-небудь мов програмування та компіляторів. До того ж разом з пакетом встановлюється бібліотека мікросхем з якими можна працювати, серед них є сімейство інтегральних схем зі змішаними сигналами GreenPAK [5] та сімейство пристроїв ForgeFPGA [6].

Оскільки перед початком роботи необхідно обрати мікросхему, з якою будемо працювати, до кожної мікросхеми, в окремому вікні, надається невеликий дайджест. У ньому зазначено короткий опис мікросхеми, до якого пакету вона відноситься, список підтримуваних платформ розробки, і,

найголовніше, кількість та вид пристроїв або логічних елементів які можуть встановлюватись у ній. Для отримання більш докладної інформації про конкретну мікросхему, у верхній частині того ж вікна, розташовано посилання на її документацію. Приклад вікна наведено на рис. 1.1.

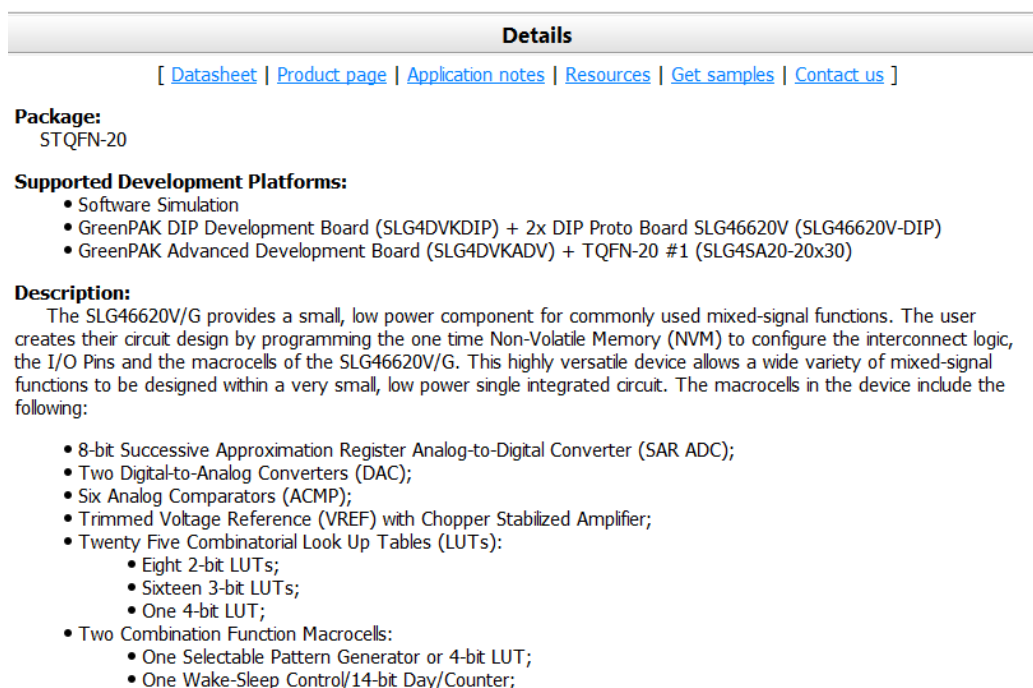


Рисунок 1.1 – Приклад вікна дайджесту

Варто зазначити також непогану гнучкість у налаштуванні як самого списку елементів мікросхеми, так і налаштування безпосередньо самих її елементів. Елементи мікросхеми поділені на групи, всередині яких у налаштуваннях елементів можна змінювати вид елемента, що продемонстровано на рис. 1.2. Завдяки цьому розширюється спектр задач, для вирішення яких можна використовувати одну й ту саму мікросхему.

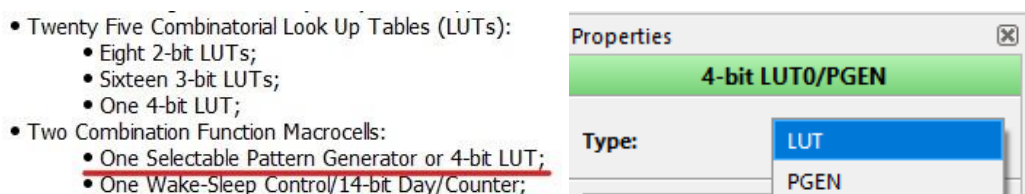
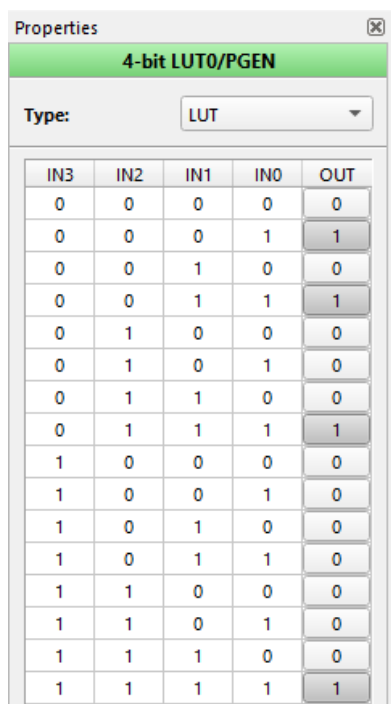


Рисунок 1.2 – Приклад вибору виду елемента мікросхеми

Крім того, як вже було згадано вище, доступна можливість налаштування окремо самих елементів, включаючи логічні елементи, що може бути дуже зручним при розробці будь-якого пристрою. Кожному логічному елементу можна задавати будь-яку довільну таблицю істинності, що можна побачити на рис. 1.3. Це також вирішує проблему неможливості налаштування кількості входів у логічних елементів.



IN3	IN2	IN1	IN0	OUT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Рисунок 1.3 – Приклад налаштування таблиці істинності логічного елемента

1.3 Теоретичне підґрунтя для синтезу цифрових пристроїв

1.3.1 Операційний автомат

Операційний автомат служить для збереження слів інформації, виконання набору мікрооперацій і обчислення значень логічних умов, тобто він є структурою, організованою для виконання дій над інформацією. Мікрооперації, виконувані операційним автоматом, задаються множиною керуючих сигналів з кожним з яких ототожнюється визначена мікрооперація. Мікрооперації - це

основні обчислювальні дії, які виконуються автоматом над даними. Кожна мікрооперація відповідає конкретній функції, яку автомат може виконувати.

Керуючі сигнали в операційному автоматі використовуються для визначення, які мікрооперації повинні бути виконані над вхідними даними. Кожен керуючий сигнал відповідає певній мікрооперації, яку автомат повинен виконати. Поєднання цих керуючих сигналів визначає послідовність мікрооперацій, які виконуються над даними.

1.3.2 Керуючий автомат

Керуючий автомат генерує запропоновану мікропрограмою послідовність керуючих сигналів, відповідно до значень логічних умов. Тобто, керуючий автомат задає порядок виконання дій в операційному автоматі згідно з закладеним алгоритмом виконання операцій та вхідними значеннями логічних умов на кожному етапі виконання алгоритму. Цей алгоритм задається у вигляді граф-схеми. Вона складається з вузлів, які представляють окремі операції, та ліній, які показують переходи між цими операціями в залежності від умов. Кожен вузол виконує певну операцію або групу операцій.

1.3.3 Компаратор

Цифрові компаратори, або відомі ще як схеми порівняння, призначені для порівняння між собою двох двійкових величин. Результатом порівняння може бути виключно або true(1), коли умову порівняння було задовільнено, або false(0) в інших випадках.

Схеми порівняння – це комбінаційні схеми, які призначені для реалізації логічних умов в мікропрограмах, а також використовуються у всіх випадках, коли необхідно виконати порівняння двох чисел.

Компаратори можуть використовуватись для генерації вхідних сигналів умови для керуючих автоматів або «прапорців» для команд передачі керування. Також компаратори являються невіддільною частиною пристроїв контролю та діагностики.

1.4 Постановка задачі проєктування

Необхідно спроектувати віртуальну лабораторію у якій користувач зможе навчитися взаємодіяти з пакетом Go Configure Software Hub [7]. Виходячи з того що, скоріше за все, лабораторією будуть користуватись люди які вже знайомі з проєктуванням цифрових пристроїв, та того що обраний програмний пакет буде досить складним для розуміння новачка у цій сфері, навчання користувача проєктуванню з нуля не є доцільним для даної віртуальної лабораторії. Отже, необхідно провести проєктування цифрового пристрою, на прикладі реалізації якого будуть демонструватися можливості програмного пакета Go Configure Software Hub і як саме у ньому необхідно працювати.

Виходячи з цього, тепер, необхідно визначитись з тим, який саме цифровий пристрій треба розробити враховуючи те що він при проєктуванні повинен:

- бути достатньо простим у реалізації, щоб не перевантажувати користувача зайвими розрахунками;
- повноцінно та зрозуміло демонструвати можливості програмного пакета;
- бути наближеним до реальних задач з якими може надалі зустрітись користувач;
- містити мінімально необхідну елементну базу для того, щоб вміщуватись в сучасних мікросхемах представлених за замовчування у програмному пакеті.

У якості цього цифрового пристрою було вирішено розробити мікрообчислювач, у ході проєктування якого можливо достатньою мірою продемонструвати функціонал, доступний для використання в обраному програмному пакеті.

2 ОСНОВНІ ЕТАПИ ПРОЄКТУВАННЯ ЦИФРОВОГО ПРИСТРОЮ

Мікрообчислювач повинен складатись з керуючого автомату, операційного автомата та компараторів, зчитувати одне чотирибітове число та обробляти його згідно з описаним нижче алгоритмом.

Порядок обробки числа повинен бути наступним:

- провести порівняння вхідного числа з константним значенням, за для того, щоб визначити чи є вхідне число більшим ніж дванадцять;
- провести порівняння вхідного числа з константним значенням, за для того, щоб визначити чи є вхідне число меншим ніж чотири;
- провести диз'юнкцію вхідного числа з поточним вмістом регістру та занести результат обчислення до регістру;
- у випадку якщо вхідне число більше ніж дванадцять занести до регістру інверсію вхідного числа;
- у випадку якщо вхідне число менше ніж дванадцять виконати циклічний зсув вмісту регістра на два біти праворуч;
- у випадку якщо вхідне число менше ніж дванадцять і менше ніж чотири провести диз'юнкцію вхідного числа з поточним вмістом регістру та занести результат обчислення до регістру після чого занести до регістру інверсію вхідного числа.

Операційний та керуючий автомати неможливо сконструювати без елементів пам'яті, тому спочатку треба визначитись з їх типом.

Оскільки, згідно з завданням, пристрій повинен бути простим у реалізації найкращим варіантом буде обрання синхронного D-тригера. Синхронний D-тригер, як і усі тригери, запам'ятовує лише один біт даних у момент спрацьовування синхросигналу, але, на відміну від інших тригерів, в нього лише один вхід даних що значно спрощує процес проєктування.

2.1 Розробка операційного автомата

У якості операційного автомата використаємо багатофункціональний регістр. Він зможе виконувати одразу як задачу зберігання даних, так і задачу їх обробки. Крім того багатофункціональний регістр буде достатньо простим для проектування, проте повною мірою буде задовольняти усі функціональні потреби.

2.1.1 Визначення мікрооперацій та побудова таблиці мікрооперацій

Виходячи з аналізу алгоритму мікрообчислювача можемо виділити у ньому три різні мікрооперації:

- диз'юнкція значення вхідного числа із вмістом самого регістру;
- інверсія значення вхідного числа;
- циклічний зсув вмісту регістру на два біти праворуч.

Згідно з задачею ці мікрооперації виконуються над одним чотирибітовим числом яке позначимо A , і відповідно кожен біт цього числа позначатиметься a_i . Оскільки число A складається з чотирьох бітів інформації для його зберігання необхідно і достатньо використати чотири D-тригери, які разом будуть виконувати функцію регістру. Поточне значення цього регістру позначимо Rg , а кожен його біт окремо - Q_i .

Для простоти розуміння позначимо визначені раніше мікрооперації як y_i , та запишемо їх у вигляді:

- $y_1(Rg \vee A)$ – мікрооперація 1(MO1);
- $y_2(\text{not}(A))$ – мікрооперація 2(MO2);
- $y_3(RC_2(Rg))$ – мікрооперація 3(MO3).

Наступним кроком створимо таблицю мікрооперацій, яка у подальшому спростить для нас процес формування системи рівнянь, призначеної для синтезу операційного автомата. Для цього проаналізуємо визначені раніше мікрооперації та розкладемо кожен з них, після чого запишемо результат у табл. 2.1.

Таблиця 2.1 – Таблиця мікрооперацій

Мікрооперація	D ₃	D ₂	D ₁	D ₀
y ₁ (Rg ∨ A)	Q ₃ ∨ a ₃	Q ₂ ∨ a ₂	Q ₁ ∨ a ₁	Q ₀ ∨ a ₀
y ₂ (not(A))	not(a ₃)	not(a ₂)	not(a ₁)	not(a ₀)
y ₃ (RC ₂ (Rg))	Q ₁	Q ₀	Q ₃	Q ₂

2.1.2 Складання рівнянь для подальшого синтезу багатфункціонального регістру

На цьому етапі необхідно скласти рівняння за допомогою яких у подальшому буде проводитись синтез багатфункціонального регістру. Цю задачу повною мірою можна задовольнити склавши функції збудження кожного D-тригера.

Система функцій збудження D-тригерів буде мати наступний вигляд:

$$\begin{cases} D_0 = ((Q_0 \vee a_0) * y_1) \vee (\text{not}(a_0) * y_2) \vee (Q_2 * y_3) \\ D_1 = ((Q_1 \vee a_1) * y_1) \vee (\text{not}(a_1) * y_2) \vee (Q_3 * y_3) \\ D_2 = ((Q_2 \vee a_2) * y_1) \vee (\text{not}(a_2) * y_2) \vee (Q_0 * y_3) \\ D_3 = ((Q_3 \vee a_3) * y_1) \vee (\text{not}(a_3) * y_2) \vee (Q_1 * y_3) \end{cases} \quad (2.1)$$

Також необхідно додати рівняння для обмеження сигналу синхронізації. Це необхідно для того, щоб тригери не спрацьовували тоді коли жоден з сигналів мікрооперації не є активним.

Виходячи з цього, рівняння матиме наступний вигляд:

$$C = CLK * (y_1 \vee y_2 \vee y_3), \quad (2.2)$$

2.2 Розробка керуючого автомата

Задачу керуючого автомату виконуватиме автомат Мілі. Він достатньо простий у розробці, вимагає кодування відносно невеликої кількості станів, проте дозволяє повною мірою виконувати задачі покладені на нього.

2.2.1 Формування граф-схеми алгоритму згідно з поставленою задачею

Першим етапом розробки автомата буде формування функціональної граф-схеми його алгоритму. Завдяки цій схемі можна буде візуально представити послідовність кроків алгоритму та зрозуміти логіку його роботи. Формування граф-схеми алгоритму дозволяє систематизувати й структурувати ідеї та дії, які потрібно реалізувати.

Граф-схема буде мати вигляд діаграми, що складається з різних блоків поєднаних лініями, які показують послідовність виконання операцій. Ці блоки можна розділити на два типи, операторні вершини та логічні вершини.

Операторні вершини в граф-схемі представляють мікрооперації, які потрібно виконати. Ці блоки можуть вміщувати у собі будь-які операції, починаючи з логічних операторів, таких як наприклад оператор АБО, закінчуючи складними арифметичними операціями, усе залежить від того що було задано у алгоритмі.

Логічні вершини в граф-схемі відображають умови переходу з однієї гілки алгоритму на іншу. За своєю концепцією вони схожі на оператор if-else з мов програмування високого рівня, тобто якщо задана умова виконується то алгоритм піде одним шляхом, а якщо умова не виконується то цей шлях буде іншим.

Отже, проаналізувавши поставлену задачу сформуємо функціональну граф-схему алгоритму, або як ще її називають змістовий граф, для цього запишемо в операторних вершинах відповідні мікрооперації, визначені раніше у пункті 2.2.1, а у логічних вершинах умови переходів зазначені в описі алгоритму. Результат зображено на рис. 2.1.

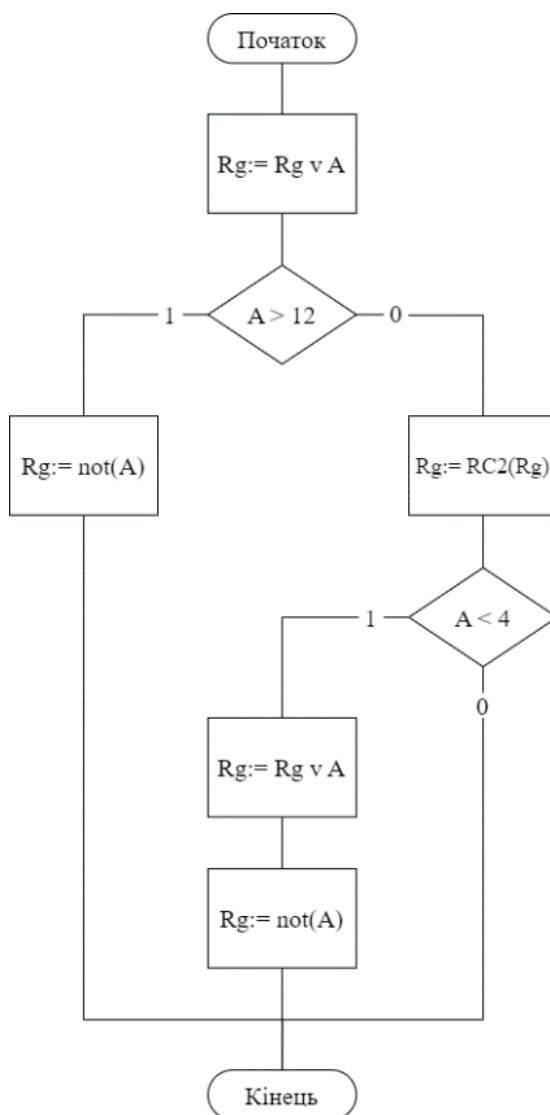


Рисунок 2.1 – функціональна граф-схема алгоритму(змістовний граф)

2.2.2 Розмічення граф-схеми алгоритму керуючого автомату

Далі, на отриманій у попередньому етапі граф-схемі, виконаємо розмітку станів у яких може перебувати автомат. Початок і кінець алгоритму завжди позначаються станом a_0 , а кожна операторна вершина станом a_i . Таким чином, виконавши розмітку граф-схеми алгоритму, яку зображено на рис. 2.2, отримаємо чотири різні стани.

Для поліпшення подальших розрахунків та спрощення вигляду граф-схеми замінимо мікрооперації на їх позначки у вигляді y_i як це було зроблено при розробці операційного автомата, і також позначимо умови переходів як x_i .

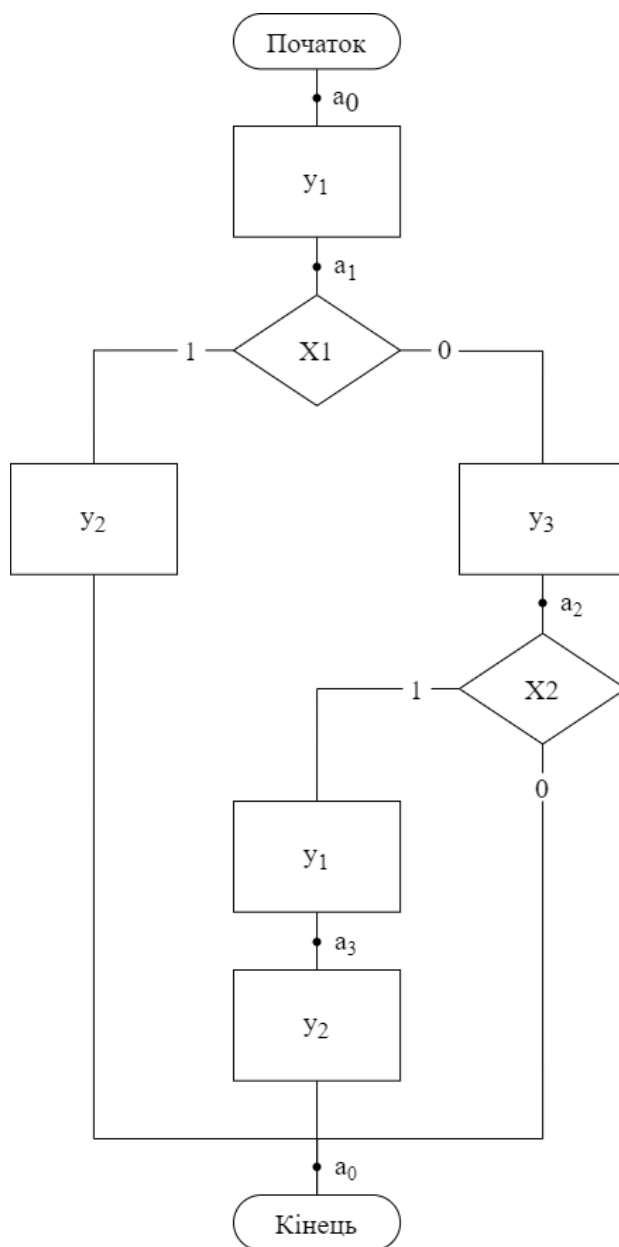


Рисунок 2.2 – Граф-схема алгоритму з розміченими станами автомату Мілі

2.2.3 Побудова графа переходів керуючого автомата

Згідно з розміченою у попередньому етапі граф-схемі алгоритму(рис. 2.2) побудуємо граф переходів керуючого автомата.

Кожній вершині графу ставимо у відповідність стан a_i автомату. Ці вершини поєднуються між собою лініями зі стрілкою яка вказує напрямок переходу, додатково на цій лінії вказується умова переходу та функція яка виконується при переході, якщо вони є.

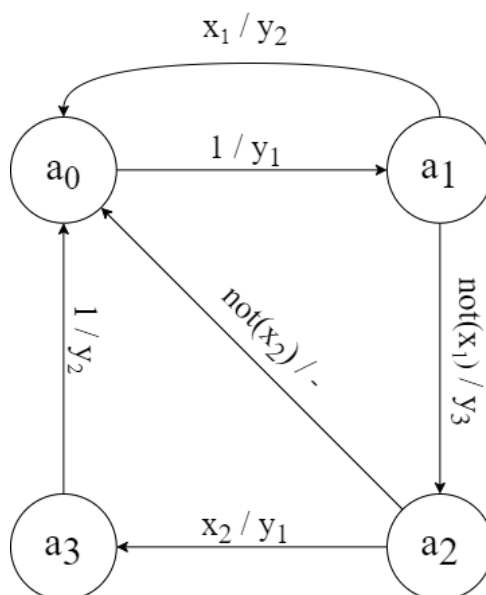


Рисунок 2.3 – Граф переходів керуючого автомата

2.2.4 Визначення кількості елементів пам'яті та кодування станів керуючого автомату

Для того, щоб провести кодування станів керуючого автомату кожному його стану необхідно поставити у відповідність певну кодову комбінацію. Кількість розрядів такої комбінації визначається спираючись на кількість станів, які необхідно закодувати, та обрану систему числення.

Найзручнішим для використання у таких умовах буде двійковий код, розрядність числа якого можна вирахувати за наступною формулою:

$$n = \langle \log_2 S \rangle, \quad (2.3)$$

де n – кількість розрядів числа;

S - число станів автомата;

$\langle \rangle$ - округлення в більшу сторону.

Виходячи з того, що число станів автомата дорівнює чотирьом, визначаємо необхідну кількість розрядів:

$$n = \langle \log_2 4 \rangle = 2, \quad (2.4)$$

Оскільки, спираючись на розрахунки вище, для кодування станів необхідно число з двох розрядів то і кількість елементів пам'яті також дорівнює двом.

Тепер, визначившись з необхідною кількістю розрядів числа, можемо записати коди усіх станів автомата у табл. 2.2.

Таблиця 2.2 – Кодові комбінації станів автомата

Стан	Код	
	Q_1	Q_2
a_0	0	0
a_1	1	1
a_2	1	0
a_3	1	1

2.2.5 Побудова таблиці переходів та вихідних функцій керуючого автомата

На цьому етапі необхідно сформувати функції переходу, збудження і виходів пристрою, спираючись на розрахунки проведені у попередніх етапах. Результати формування цих функцій занесено до табл. 2.3.

Таблиця 2.3 – Таблиця переходів автомата

Q_t	Код		Q_{t+1}	Код		Умова переходу	D_1	D_0	Функції
	Q_1	Q_0		Q_1	Q_0				
a_0	0	0	a_1	0	1	1	0	1	y_1
a_1	0	1	a_0	0	0	x_1	0	0	y_2
a_1	0	1	a_2	1	0	$\text{not}(x_1)$	1	0	y_3
a_2	1	0	a_0	0	0	$\text{not}(x_2)$	0	0	-
a_2	1	0	a_3	1	1	x_2	1	1	y_1
a_3	1	1	a_0	0	0	1	0	0	y_2

2.2.6 Формування та мінімізація функцій збудження елементів пам'яті й функцій виходів

Сформуємо, за допомогою таблиці переходів, систему логічних рівнянь яка стане основою для подальшого синтезу керуючого автомата.

$$\begin{cases} D_0 = \text{not}(Q_1) * \text{not}(Q_0) \vee Q_1 * \text{not}(Q_0) * x_2 \\ D_1 = \text{not}(Q_1) * Q_0 * \text{not}(x_1) \vee Q_1 * \text{not}(Q_0) * x_2 \\ y_1 = \text{not}(Q_1) * \text{not}(Q_0) \vee Q_1 * \text{not}(Q_0) * x_2 \\ y_2 = \text{not}(Q_1) * Q_0 * x_1 \vee Q_1 * Q_0 \\ y_3 = \text{not}(Q_1) * Q_0 * \text{not}(x_1) \end{cases}, \quad (2.5)$$

2.2.7 Визначення термів

З системи логічних рівнянь визначимо повний перелік термів для керуючого автомата. Це спростить процес синтезу та дозволить економніше використати ресурси мікросхеми завдяки виключенню вірогідності повторення одних і тих самих термів у різних рівняннях системи.

$$\begin{aligned} T_0 &= \text{not}(Q_1) * \text{not}(Q_0); \\ T_1 &= Q_1 * \text{not}(Q_0) * x_2; \\ T_2 &= \text{not}(Q_1) * Q_0 * \text{not}(x_1); \\ T_3 &= \text{not}(Q_1) * Q_0 * x_1; \\ T_4 &= Q_1 * Q_0. \end{aligned} \quad (2.6)$$

Таким чином система рівнянь 2.5 буде мати наступний вигляд:

$$\begin{cases} D_0 = T_0 \vee T_1 \\ D_1 = T_2 \vee T_1 \\ y_1 = T_0 \vee T_1, \\ y_2 = T_3 \vee T_4 \\ y_3 = T_2 \end{cases} \quad (2.7)$$

2.3 Розробка компараторів

Відповідно до поставленої задачі потрібно розробити два компаратори із різними умовами, які будуть порівнювати вхідне число із константним значенням.

Компаратори будуть генерувати сигнали умови для керуючого автомата, відповідно можемо для зручності позначити їх так само як вони названі у керуючому автоматі, тобто x_1 та x_2 . Виходячи з цього умови цих компараторів можемо записати таким чином:

- для виконання умови x_1 необхідно реалізувати нерівність $A > 12_{10}$ (1100_2);
- для виконання умови x_2 необхідно реалізувати нерівність $A < 4_{10}$ (0100_2).

2.3.1 Визначення базових функцій

Для початку необхідно визначити базові функції i -тих розрядів чисел для обох умов. Найпростішим та найзручнішим варіантом для цього буде скласти до кожної з них ТІ. Ці ТІ будуть відображені у табл. 2.4 та табл. 2.5.

Таблиця 2.4 – Таблиця істинності для умови x_1

a_i	b_i	m_i
0	0	0
0	1	0
1	0	1
1	1	0

Проаналізувавши ТІ зображену у табл. 2.4 можемо дійти висновку, що базова функція для цієї умови матиме наступний вигляд: $m_i = a_i * \text{not}(b_i)$.

Таблиця 2.5 – Таблиця істинності для умови x_2

a_i	b_i	l_i
0	0	0
0	1	1
1	0	0
1	1	0

Проаналізувавши ТІ зображену у табл. 2.5 можемо зробити висновок, що базова функція для цієї умови матиме наступний вигляд: $l_i = b_i * \text{not}(a_i)$.

2.3.2 Формування повних функцій компараторів

На цьому етапі необхідно сформуванати повні функції за допомогою яких у подальшому буде відбуватись синтез компараторів. У загальному вигляді формула для формування повної функції записується таким чином:

$$Y = y_i \vee (R_i * y_{i-1}) \vee (R_i * R_{i-1} * y_{i-2}) \vee \dots \vee (R_i * \dots * R_1 * y_0), \quad (2.8)$$

де y_i – базова функція;

R_i – функція рівнозначності, яка визначається за формулою $R_i = a_i * b_i \vee \text{not}(a_i) * \text{not}(b_i)$.

Для умови x_1 повна функція буде мати наступний вигляд:

$$M = m_3 \vee R_3 * m_2 \vee R_3 * R_2 * m_1 \vee R_3 * R_2 * R_1 * m_0, \quad (2.9)$$

Для умови x_2 повна функція буде мати наступний вигляд:

$$L = l_3 \vee R_3 * l_2 \vee R_3 * R_2 * l_1 \vee R_3 * R_2 * R_1 * l_0, \quad (2.10)$$

2.3.3 Підстановка константних значень у попередньо визначені формули

Спочатку підставимо константні значення у формули базових функцій та функцій рівнозначності. Завдяки цьому можна скоротити та спростити запис повної функції після підстановки у неї значень u_i та R_i .

Підстановка константних значень до базових функцій умови x_1 :

$$\begin{aligned} m_3 &= a_3 * \text{not}(b_3) = a_3 * 0 = 0; \\ m_2 &= a_2 * \text{not}(b_2) = a_2 * 0 = 0; \\ m_1 &= a_1 * \text{not}(b_1) = a_1 * 1 = a_1; \\ m_0 &= a_0 * \text{not}(b_0) = a_0 * 1 = a_0. \end{aligned} \tag{2.11}$$

Підстановка константних значень до функцій рівнозначності умови x_1 :

$$\begin{aligned} R_3 &= a_3 * b_3 \vee \text{not}(a_3) * \text{not}(b_3) = a_3 * 1 \vee \text{not}(a_3) * 0 = a_3; \\ R_2 &= a_2 * b_2 \vee \text{not}(a_2) * \text{not}(b_2) = a_2 * 1 \vee \text{not}(a_2) * 0 = a_2; \\ R_1 &= a_1 * b_1 \vee \text{not}(a_1) * \text{not}(b_1) = a_1 * 0 \vee \text{not}(a_1) * 1 = \text{not}(a_1); \\ R_0 &= a_0 * b_0 \vee \text{not}(a_0) * \text{not}(b_0) = a_0 * 0 \vee \text{not}(a_0) * 1 = \text{not}(a_0). \end{aligned} \tag{2.12}$$

Підстановка константних значень до базових функцій умови x_2 :

$$\begin{aligned} l_3 &= a_3 * \text{not}(b_3) = a_3 * 1 = a_3; \\ l_2 &= a_2 * \text{not}(b_2) = a_2 * 0 = 0; \\ l_1 &= a_1 * \text{not}(b_1) = a_1 * 1 = a_1; \\ l_0 &= a_0 * \text{not}(b_0) = a_0 * 1 = a_0. \end{aligned} \tag{2.13}$$

Підстановка константних значень до функцій рівнозначності умови x_2 :

$$\begin{aligned} R_3 &= a_3 * b_3 \vee \text{not}(a_3) * \text{not}(b_3) = a_3 * 0 \vee \text{not}(a_3) * 1 = \text{not}(a_3); \\ R_2 &= a_2 * b_2 \vee \text{not}(a_2) * \text{not}(b_2) = a_2 * 1 \vee \text{not}(a_2) * 0 = a_2; \\ R_1 &= a_1 * b_1 \vee \text{not}(a_1) * \text{not}(b_1) = a_1 * 0 \vee \text{not}(a_1) * 1 = \text{not}(a_1); \\ R_0 &= a_0 * b_0 \vee \text{not}(a_0) * \text{not}(b_0) = a_0 * 0 \vee \text{not}(a_0) * 1 = \text{not}(a_0). \end{aligned} \tag{2.14}$$

Останнім кроком буде підстановка отриманих значень у попередньо визначені повні функції.

Повна функція для умови x_1 матиме наступний вигляд:

$$M = 0 \vee a_3 * 0 \vee a_3 * a_2 * a_1 \vee a_3 * a_2 * \text{not}(a_1) * a_0 = a_3 * a_2 * a_1 \vee a_3 * a_2 * \text{not}(a_1) * a_0, \quad (2.15)$$

Повна функція для умови x_2 матиме наступний вигляд:

$$L = a_3 \vee \text{not}(a_3) * 0 \vee \text{not}(a_3) * a_2 * a_1 \vee \text{not}(a_3) * a_2 * \text{not}(a_1) * a_0 = \text{not}(a_3) * \text{not}(a_2), \quad (2.16)$$

У другому розділі було проведено аналіз поставленої задачі. Згідно з цим аналізом було розроблено усі структурні складові цифрового пристрою, а саме, операційний автомат, керуючий автомат та два компаратори. Було проведено усі необхідні розрахунки для подальшого синтезу моделей кожного зі структурних елементів цифрового пристрою.

3 МОДЕЛЮВАННЯ І ТЕСТУВАННЯ ЦИФРОВОГО ПРИСТРОЮ У ПАКЕТІ GO CONFIGURE SOFTWARE HUB

3.1 Моделювання операційного автомата

3.1.1 Визначення елементної бази та підходящої мікросхеми

Перед початком синтезу пристрою необхідно визначитись з тим яка буде кількість логічних та запам'ятовуючих елементів, що стане критерієм для вибору мікросхеми. Для цього скористаємось формулами 2.1 та 2.2 та запишемо результат у табл. 3.1.

Таблиця 3.1 – Логічні елементи для реалізації операційного автомата

Логічний елемент	Кількість входів логічних елементів			Всього
	1 вхід	2 входи	3 входи	
Не	4	–	–	26
І	–	13	–	
Або	–	4	5	

Виходячи з цього, нам необхідна мікросхема яка буде містити 26 логічних елементів та чотири D-тригери. Однак, завдяки тому, що у пакеті Go Configure Software Hub ми маємо можливість налаштовувати таблицю істинності усіх логічних елементів ми можемо зменшити необхідну кількість логічних елементів. Наприклад ми можемо в одному логічному елементі на три входи сумістити «І» на два входи та слідуючий за ним «Або» на два входи. Докладніше про це буде у наступному пункті.

Отже, проаналізуємо ще раз формулу 2.1 за для того, щоб виявити операції які можна сумістити та враховуючи ці дані запишемо оновлений список логічних елементів у табл. 3.2. Однак треба враховувати те, що у більшості доступних мікросхем кількість LUT на чотири входи не більше двох.

Таблиця 3.2 – Кількість LUT для реалізації операційного автомата

Логічний елемент	Кількість входів логічних елементів			Всього
	2 входи	3 входи	4 входи	
LUT	7	7	2	16

Отже, підсумовуючи усе вище зазначене, переглянувши бібліотеку мікросхем пакета Go Configure Software Hub оберемо SLG46620V. Основна інформація яка нас цікавить з дайджесту обраної мікросхеми наведена на рис. 3.1.

- Twenty Five Combinatorial Look Up Tables (LUTs):
 - Eight 2-bit LUTs;
 - Sixteen 3-bit LUTs;
 - One 4-bit LUT;
- Two Combination Function Macrocells:
 - One Selectable Pattern Generator or 4-bit LUT;
 - One Wake-Sleep Control/14-bit Day/Counter;
- Three Digital Comparators/Pulse Width Modulators (DCMPs /PWMs) w/ Selectable Deadband;
- Nine Counter / Delay Generators (CNT/DLY):
 - Two 14-bit Delay/Counter;
 - One 14-bit Delay/Counter/Finite State Machine;
 - Five 8-bit Delay/Counter;
 - One 8-bit Delay/Counter/Finite State Machine;
- Twelve D Flip-flops/Latches;

Рисунок 3.1 – Дайджест мікросхеми SLG46620V

3.1.2 Налаштування LUT (логічних елементів)

Використовуваний програмний пакет, як вже раніше згадувалось, дозволяє налаштовувати логічні елементи чим ми скористаємось за для економії ресурсів мікросхеми та демонстрації можливостей пакету.

З самого початку, коли ми розміщуємо логічний елемент у робочому просторі, він позначається як Look up table, або скорочено LUT, його вигляд та налаштування, встановлені за замовченням, можемо побачити на рис. 3.2. Він може виконувати функції як стандартних логічних елементів, так і слідувати будь-якій іншій, створеній користувачем, таблиці істинності.

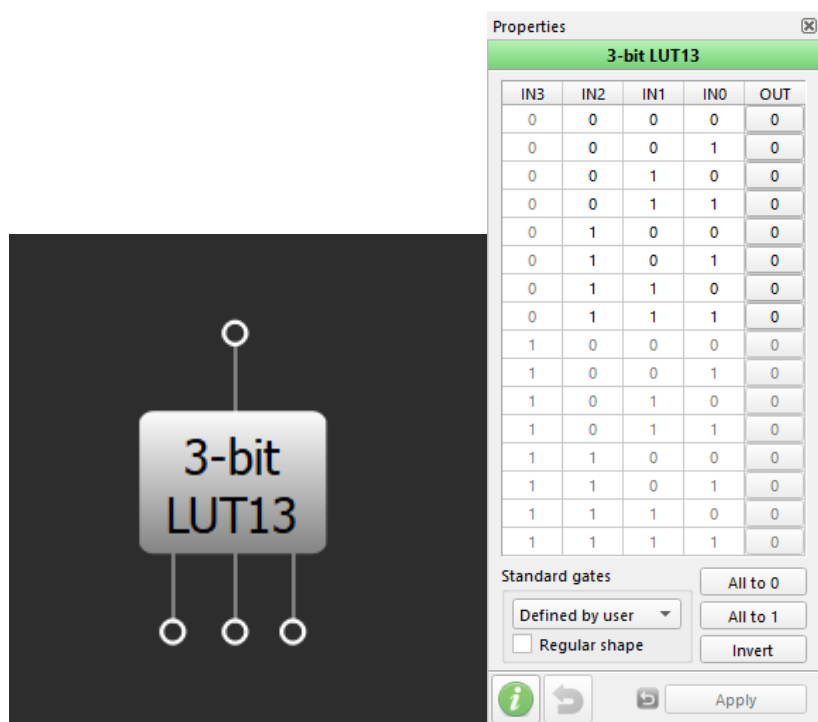


Рисунок 3.2 – LUT та вікно його налаштування

Для того, щоб переключити LUT у режим роботи стандартного логічного елемента буде достатньо розкрити меню Standard gates та обрати потрібний елемент. Це меню продемонстровано на рис. 3.3.

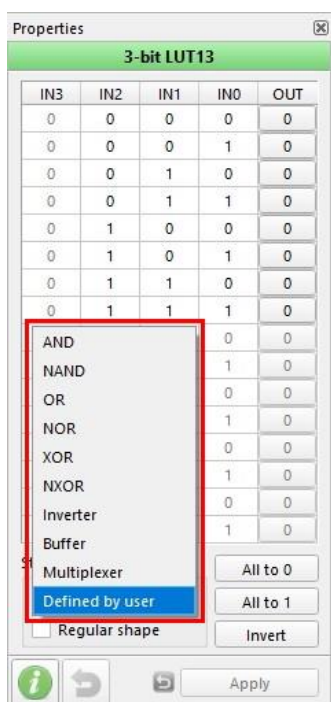


Рисунок 3.3 – Меню Standard gates з вибором логічного елемента

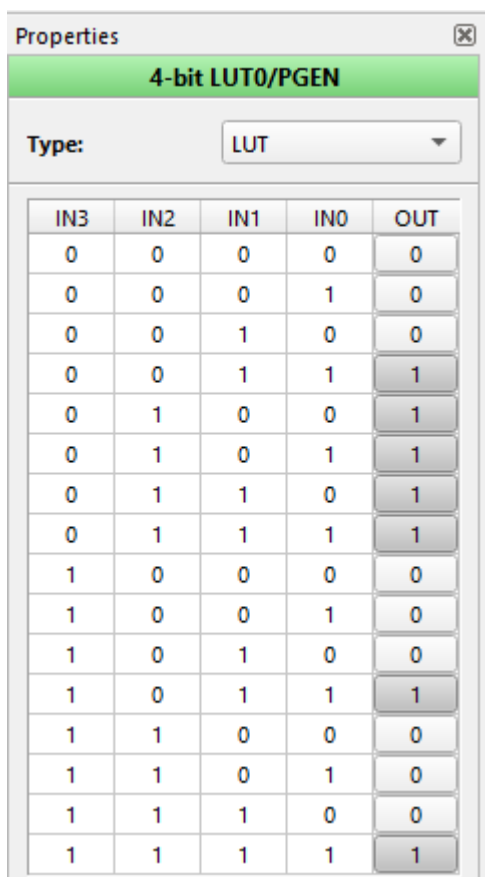
Якщо ж необхідно налаштувати LUT на роботу за користувацькою таблицею істинності скористаємось таблицею значень яку можемо бачити у вікні налаштувань елемента на рис. 3.2. Оскільки елемент який ми розглядаємо має три входи то і таблиця, доступна для налаштування обмежена 3 колонками входів IN3-IN0 та колонкою з вихідним значенням OUT. У колонці OUT вказується вихідне значення для кожної з комбінацій IN3-IN0.

Перейдімо до прикладу який надалі буде використано при збірці операційного автомата. Візьмемо з формули 2.1 фрагмент з функції збудження D_0 , а саме $\text{not}(a_0) * y_2 \vee Q_2 * y_3$, та реалізуємо цей фрагмент на одному LUT на чотири входи. Таблицю істинності для цього елемента запишемо у табл. 3.3.

Таблиця 3.3 – Таблиця істинності для LUT на чотири входи

a_0 (IN3)	y_2 (IN2)	Q_2 (IN1)	y_3 (IN0)	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Для утворення цієї таблиці необхідно розділити фрагмент функції D_0 на $\text{not}(a_0) \cdot y_2$ та $Q_2 \cdot y_3$ й розглянути їх окремо. У частині $\text{not}(a_0) \cdot y_2$ ми отримуємо «1» лише коли комбінація цих значень буде «01», а у частині $Q_2 \cdot y_3$ – лише коли «11». Відповідно до цього можемо записати фрагмент як «01» АБО «11». Тож, знаючи це, нам достатньо проставити у колонці OUT значення «1» у тих випадках коли входи для $\text{not}(a_0)$ та y_2 будуть утворювати комбінацію «01», а входи Q_2 та y_3 утворюватимуть «11». Результат налаштування LUT для фрагмента функції D_0 продемонстровано на рис. 3.4. Через те, що у мікросхемі лише два LUT на чотири входи, другий, аналогічним способом, налаштуємо для відповідного фрагменту функції D_3 , а такі ж фрагменти функцій D_2 та D_1 реалізуємо звичайним способом.



IN3	IN2	IN1	IN0	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Рисунок 3.4 – Результат налаштування LUT на чотири входи

Далі налаштуємо LUT на 3 входи для іншого фрагменту функції D_i з формули 2.1, а саме $(Q_i \vee a_i) \cdot y_1$. Таблицю істинності для цього елемента запишемо у табл. 3.4.

Таблиця 3.4 – Таблиця істинності для LUT на три входи

Q_i (IN2)	a_i (IN1)	y_1 (IN0)	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Останнім налаштуємо LUT на два входи для фрагментів функцій D_2 та D_1 з формули 2.1, а саме $\text{not}(a_i) * y_2$. Тут все дуже просто оскільки нам лише необхідно у таблиці істинності врахувати інверсію на вході з сигналом a_i . Тобто такий LUT буде формувати на виході «1» лише при вхідній комбінації «01». Таблицю істинності цього LUT відображено у табл. 3.5.

Таблиця 3.5 – Таблиця істинності для LUT на два входи

a_i (IN1)	y_2 (IN0)	OUT
0	0	0
0	1	1
1	0	0
1	1	0

3.1.3 Збірка схеми операційного автомата

Для початку збірки схеми заходимо у програмний пакет Go Configure Software Hub, відкриваємо вікно Develop та шукаємо у з'явившомуся списку

обрану у пункті 3.1.1 мікросхему SLG46620V. Коли знаходимо її, відкриваємо подвійним кліком та чекаємо доки завантажиться вікно роботи з мікросхемою. Це вікно зображено на рис. 3.5.

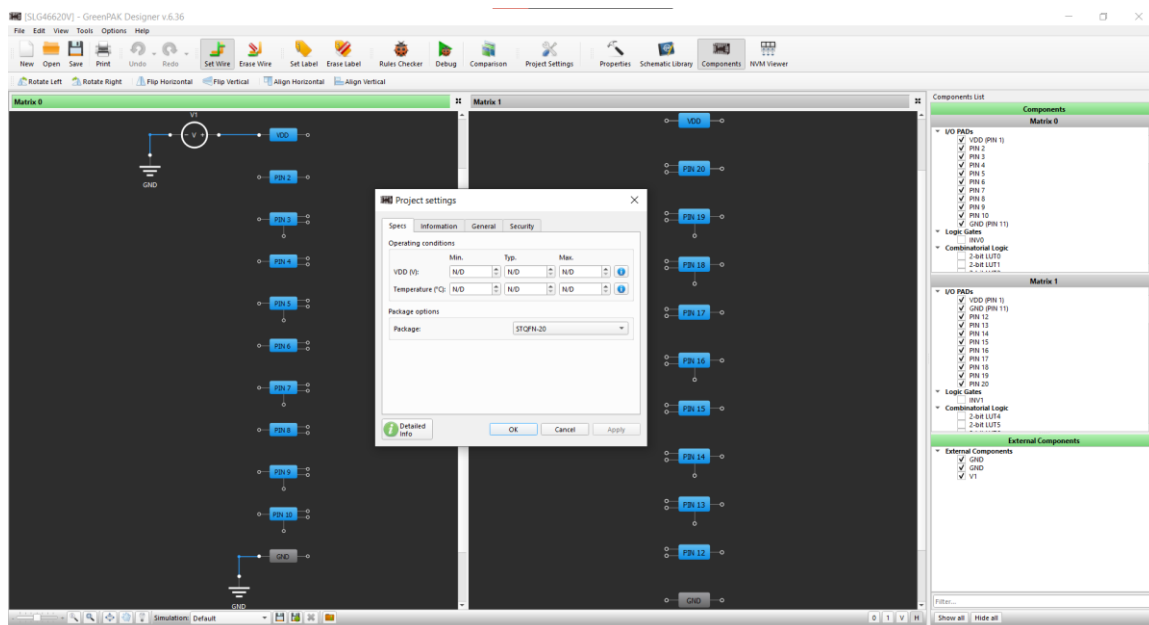


Рисунок 3.5 – Вікно для роботи з мікросхемою SLG46620V у програмному пакеті Go Configure Software Hub

На рис. 3.5 можемо побачити вікно Project settings у якому необхідно вказати параметри вхідної напруги та температури для нашої мікросхеми. Вказуємо мінімальні значення усіх параметрів(достатньо клацнути на стрілочку вгору біля значення кожного з параметрів) та підтверджуємо. За потреби надалі ці параметри можна буде змінити у будь-який момент, відкривши це вікно з панелі швидкого доступу.

Також з рис. 3.5 видно що робочий простір мікросхеми поділений на дві частини, це особливість обраної мікросхеми, бо вона складається з двох матриць. У самому ж робочому просторі поки що можемо побачити тільки зовнішні піни мікросхеми.

Самі ж логічні елементи, за допомогою яких буде відбуватись збірка схеми операційного автомата, знаходяться у вікні Components List яке знаходиться праворуч від робочого простору, це вікно зображено на рис. 3.6. Якщо це вікно

закрите його можна відкрити з панелі швидкого доступу. Components List також поділено, але вже на три частини, елементи кожної з матриць та зовнішні елементи, які, як зрозуміло з назви, знаходяться поза межами мікросхеми та можуть використовуватись, наприклад для генерації вхідних або зчитування вихідних сигналів мікросхеми.

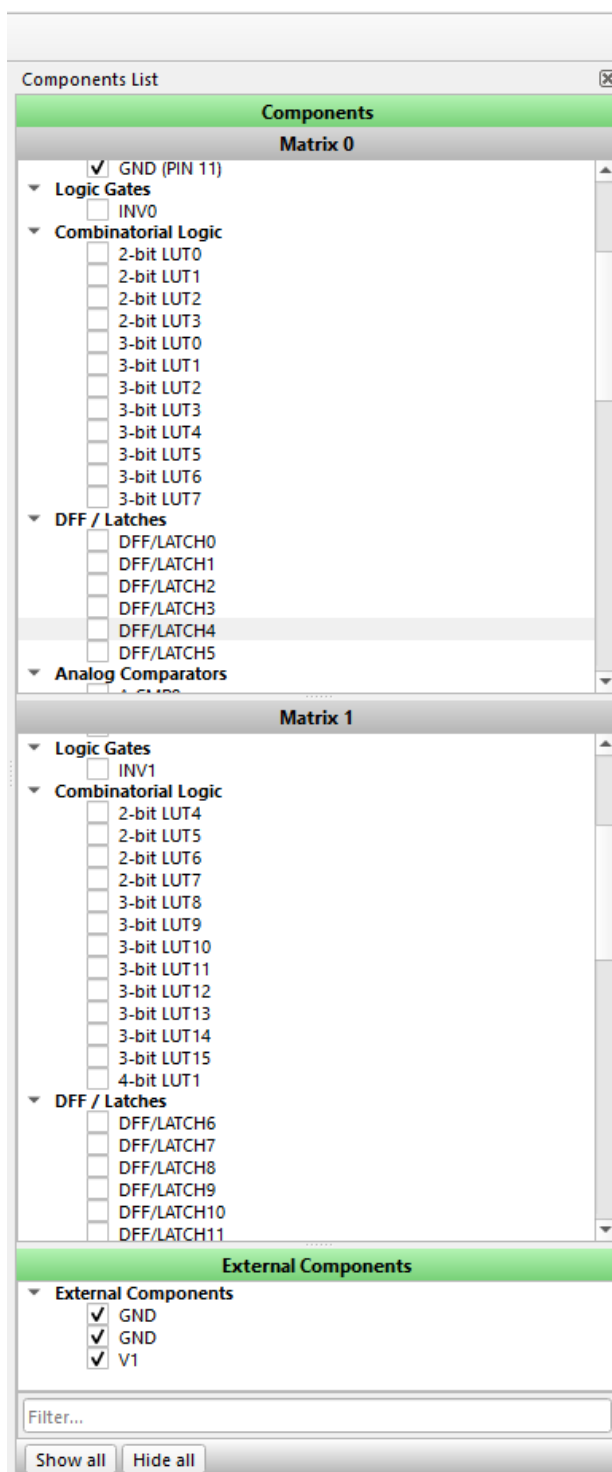


Рисунок 3.6 – Вікно Components List

Для того, щоб розмістити будь-який з доступних елементів мікросхеми у робочому просторі достатньо поставити галочку навпроти потрібного елементу, після чого він з'явиться у робочому просторі, готовий для подальшого налаштування та підключення. Однак, треба враховувати те, що у кожній з матриць обраної нами мікросхеми окремий набір елементів, тобто якщо у вікні Components List обрати, наприклад, LUT на два входи з розділу нульової матриці то цей LUT з'явиться саме у її робочому просторі.

Варто також зазначити те, що оскільки мікросхема складається з двох матриць ми не можемо просто протягнути дріт між елементами які розташовані на різних матрицях. Для того, щоб передавати сигнали між матрицями передбачені порти, які можна знайти у списку елементів, що показано на рис. 3.7. При розміщенні порту їх розміщується одразу два, один на вхід з тієї матриці на якій він був обраний, інший на вихід до другої матриці, приклад цього продемонстровано на рис. 3.8.

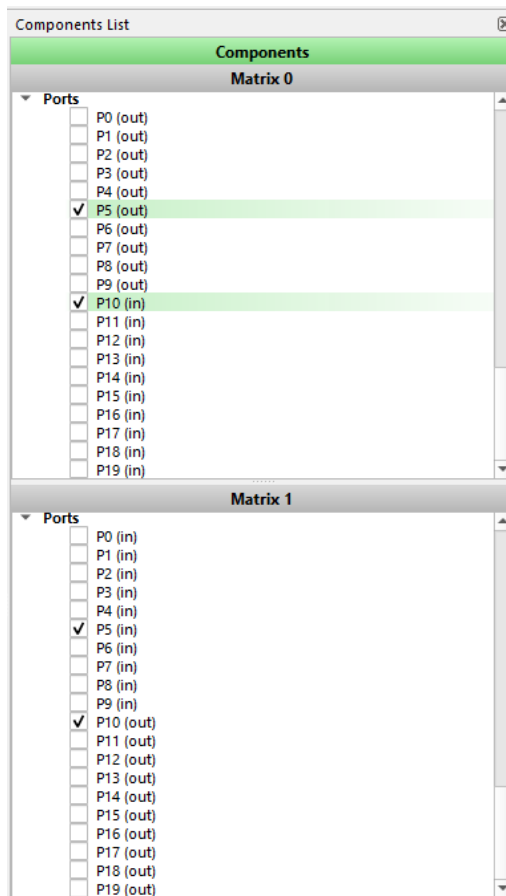


Рисунок 3.7 – Порти у списку елементів мікросхеми

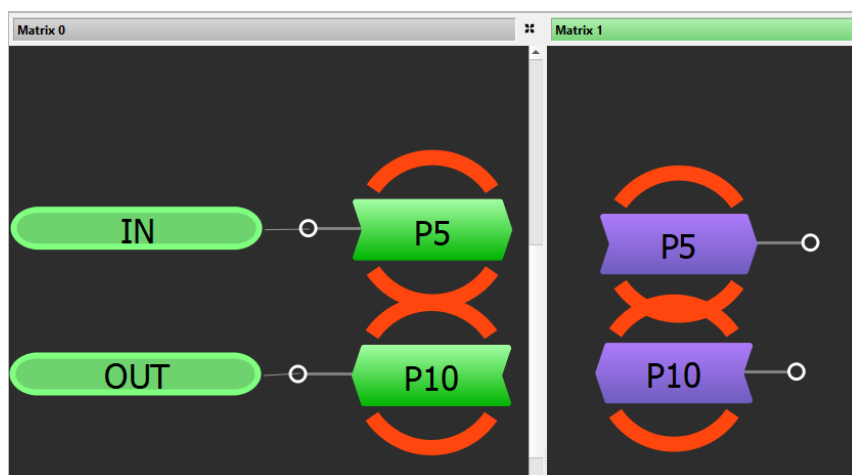


Рисунок 3.8 – Порти зв'язку між матрицями

Можна ще додати про можливість створювати мітки, які ми можемо прив'язувати до будь-яких елементів мікросхеми. За допомогою цих міток можна значно зменшити вірогідність заплутування у дротах, особливо це корисно як раз у двоматричних мікросхемах, оскільки зручніше підписати який сигнал йде через певний порт, ніж кожного разу дивитись на те звідки цей порт приймає сигнали. Також ми маємо можливість змінювати колір тексту мітки, що також може допомогти у позначенні різних типів сигналів розділяючи їх, наприклад, на вхідні, проміжкові та вихідні. Щоб створити мітку достатньо клацнути по обраному елементу ПКМ та обрати пункт Set Label. Після цього відкриється вікно де можемо записати текст мітки та налаштувати її колір, це продемонстровано на рис. 3.9, а результуючий вигляд мітки на рис. 3.10.

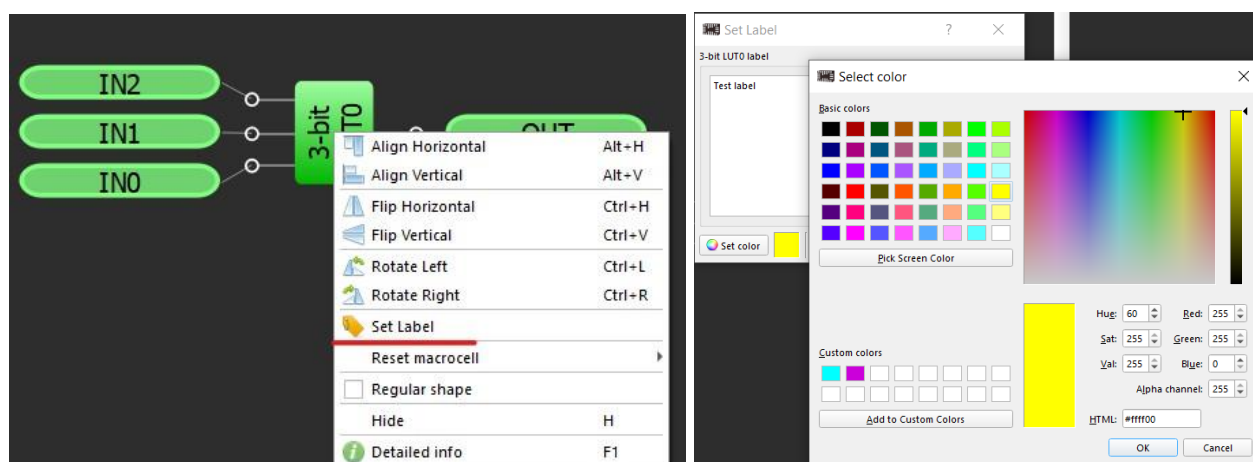


Рисунок 3.9 – Створення та налаштування мітки

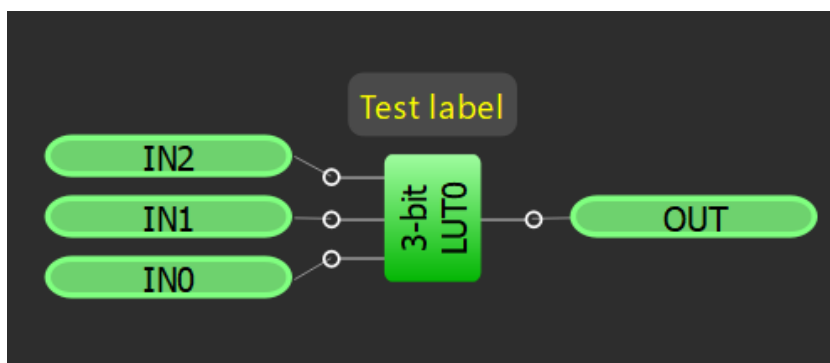


Рисунок 3.10 – Створена мітка

Тепер, враховуючи усе вище зазначене, переходимо безпосередньо до збірки схеми операційного автомата.

Для цього спочатку за допомогою міток позначимо вхідні та вихідні сигнали схеми. Через те що у кожній матриці мікросхеми SLG46620V лише по три D-тригери з вбудованим сигналом Reset було вирішено розділити регістр операційного автомата навпіл, тобто у кожній матриці буде розміщено по два D-тригери. Враховуючи це зручніше буде вводити та виводити старші біти числа у нульовій матриці, а молодші у першій. Таким чином розмітка пінів матиме вигляд, який зображено на рис. 3.11.

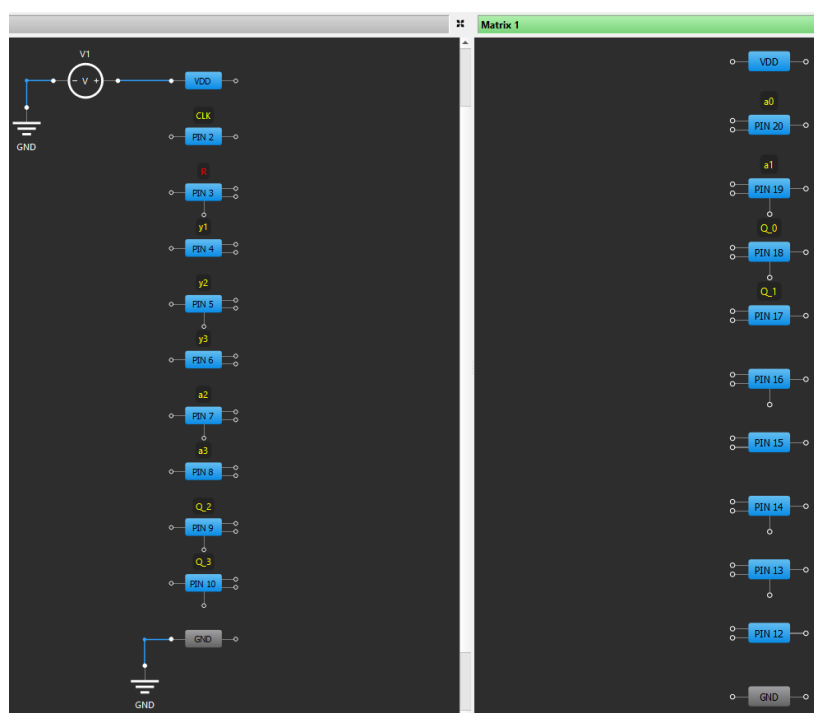


Рисунок 3.11 – Розмітка пінів мікросхеми

Наступним кроком розмістимо на матрицях D-тригери, у списку компонентів вони позначені як DFF, тобто D Flip-Flop. Після чого одразу можемо підключити до них сигнал R та утворити, за допомогою двох LUT на два та три входи, синхросигнал C за формулою 2.2. Сигнали R та C до матриці один передамо через порти та підпишемо їх. Підписуємо кожен з тригерів, щоб знати надалі те за який розряд числа відповідає кожен з тригерів. Результат можна побачити на рис. 3.12.

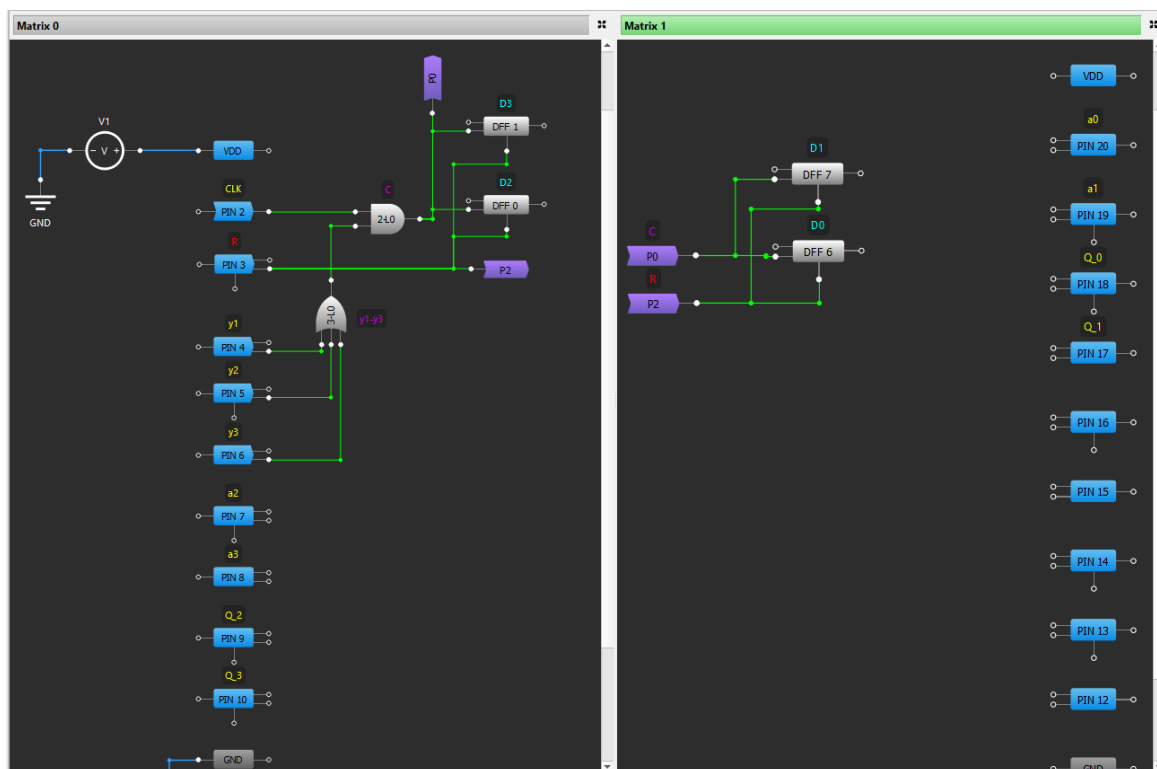


Рисунок 3.12 – Розміщення та підключення тригерів

Далі необхідно розмістити у робочому просторі усі необхідні нам логічні елементи згідно з формулою 2.1, з'єднати їх та налаштувати необхідні елементи спираючись на визначені у попередньому пункті таблиці істинності. Підключати сигнали до входів налаштованих елементів обов'язково у такому самому порядку як і у таблиці істинності. Варто також встановлювати мітки до кожного елементу, щоб у ході налаштування розуміти за яку операцію відповідає той чи інший елемент.

Результат збірки операційного автомата продемонстровано на рис. 3.13.

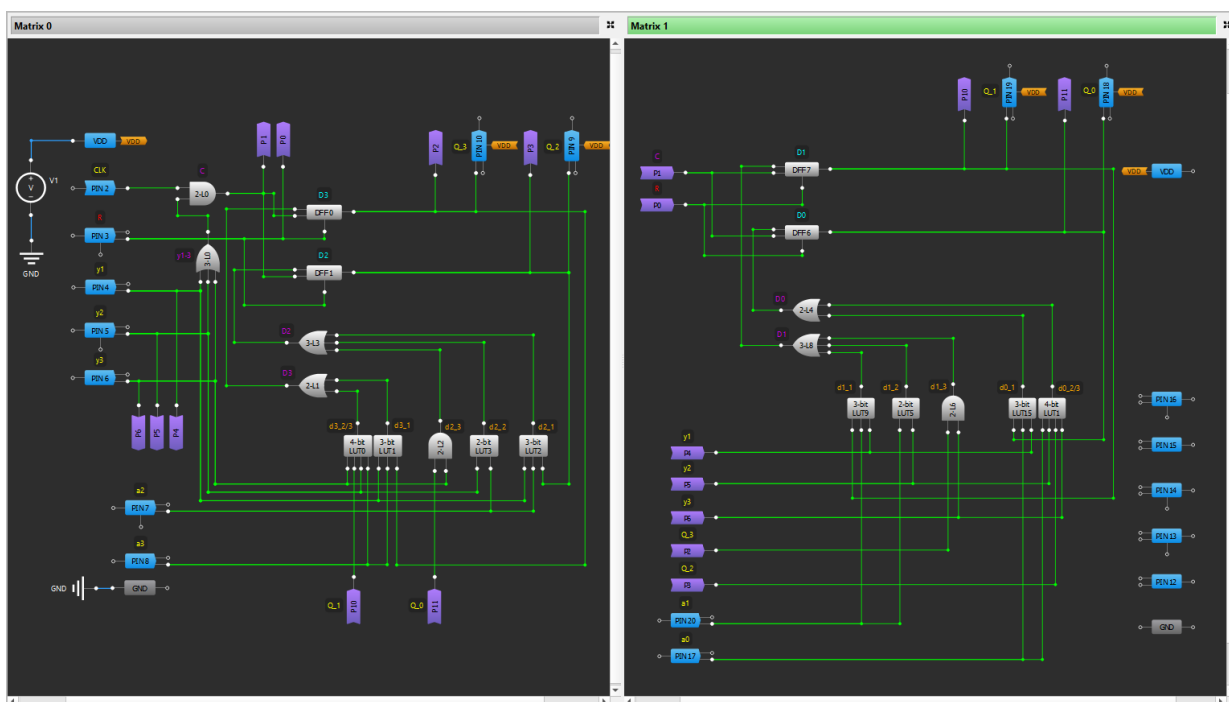


Рисунок 3.13 – Зібрана схема операційного автомата

3.1.4 Підготовка схеми до тестування

Останнім кроком буде невелика підготовка схеми до подальшого тестування. Для цього необхідно підключити до усіх входів схеми джерела напруги. Їх ми можемо знайти у вікні Schematic Library що показано на рис. 3.14. Це вікно можна відкрити з панелі швидкого доступу. Результуючий вигляд схеми продемонстровано на рис. 3.15.

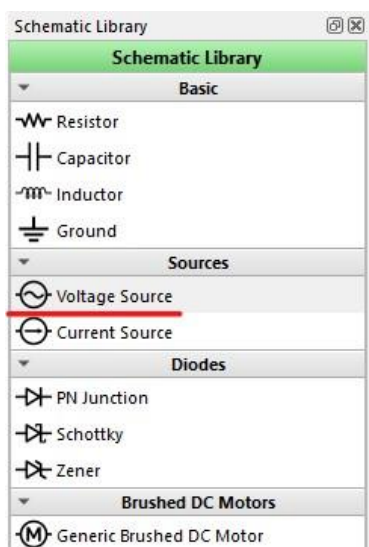


Рисунок 3.14 – Джерело напруги у вікні Schematic Library

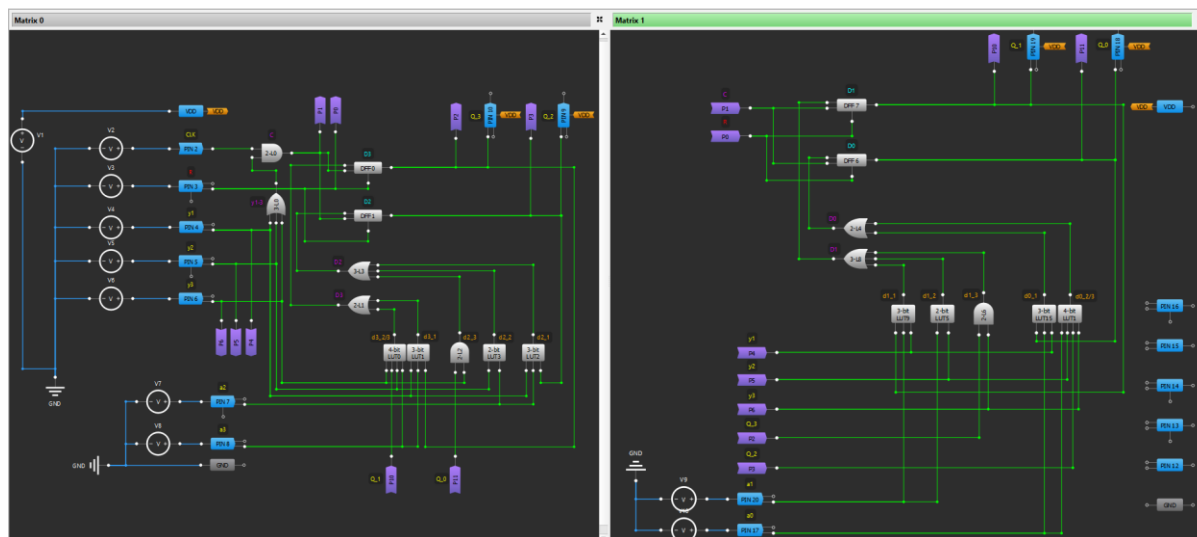


Рисунок 3.15 – Схема операційного автомата готова до тестування

3.2 Моделювання керуючого автомата

3.2.1 Визначення елементної бази та підходящої мікросхеми

Так само як і на початку синтезу операційного автомата спершу необхідно визначитись з кількістю логічних та запам'ятовуючих елементів, і надалі спираючись на ці дані обрати підходящу мікросхему. Для цього проаналізуємо формулу 2.5 та запишемо результат у табл. 3.6.

Таблиця 3.6 – Кількість LUT для реалізації керуючого автомата

Логічний елемент	Кількість входів логічних елементів		Всього
	2 входи	3 входи	
LUT	6	3	9

Отже, спираючись на результати попереднього аналізу, переглянувши бібліотеку мікросхем пакета Go Configure Software Hub було обрано мікросхему

SLG46536V. Основна інформація яка нас цікавить з дайджесту обраної мікросхеми наведена на рис. 3.16.

- Twenty-four Combination Function Macrocells:
 - Three Selectable DFF/Latch or 2-bit LUTs;
 - One Selectable Continuous DFF/Latch or 3-bit LUT;
 - Twelve Selectable DFF/Latch or 3-bit LUTs;
 - One Selectable Pipe Delay or 3-bit LUT;
 - One Selectable Programmable Pattern Generator or 2-bit LUT;
 - Five 8-bit delays/counters or 3-bit LUTs;
 - Two 16-bit delays/counters or 4-bit LUTs;
- Combinatorial Logic:
 - One 4-bit LUT;

Рисунок 3.16 – Дайджест мікросхеми SLG46536V

Хоча кількість LUT більша ніж нам потрібно ми цим скористаємось при синтезі компараторів. А саме, розмістимо їх разом із керуючим автоматом в одній мікросхемі.

3.2.2 Налаштування LUT

Проаналізувавши формулу 2.5 можемо дійти висновку що тут немає операцій які ми можемо сумістити, окрім врахування інверсій входів безпосередньо у LUT. Для цього необхідно і достатньо навпроти потрібної комбінації входних значень IN2-IN0, у колонці OUT, поставити «1».

У якості прикладу приведемо налаштування для T_1 з формули 2.6. Для T_1 нам необхідно реалізувати $Q_1 * \text{not}(Q_0) * x_2$, що можна записати як комбінацію сигналів «101». Тож нам буде достатньо поставити у колонці OUT значення «1» лише навпроти комбінації «101», це продемонстровано на рис. 3.17.

3-bit LUT1/DFF/LATCH4					
Type:	LUT				
IN3	IN2	IN1	IN0	OUT	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	1	0	

Рисунок 3.17 – Налаштування LUT для реалізації T_1

Також подивившись на табл. 3.6 та рис. 3.16 можемо побачити що нам не вистачає двох LUT на два входи, тому ми можемо їх компенсувати за рахунок LUT на три входи налаштувавши їх. Це налаштування відбувається майже так само, як й усі інші, нам достатньо поставити значення «1» у колонці OUT навпроти потрібної комбінації, відмінність буде лише в тому, що в нас один з контактів не буде підключатись. Проте це може не знадобитись якщо необхідно реалізувати операцію АБО що пов'язано з принципом роботи цієї операції.

3.2.3 Збірка схеми керуючого автомата

Збірка керуючого автомата буде проводитись способом, аналогічним до збірки операційного автомата. Для збірки скористаємось спочатку формулою 2.6 для збірки окремих термів, а потім формулою 2.7 для об'єднання цих термів у необхідні нам сигнали.

Додатково відмітимо ще одну з можливостей програмного пакета Go Configure Software Hub, а саме можливість підключати сигнали за допомогою лейблів замість дротів. Фактично нічого не змінюється, оскільки це змінює лише візуальне відображення підключень, проте це спрощує розуміння схеми за рахунок зменшення кількості дротів на екрані. Для того, щоб створити такий лейбл необхідно підключити перший дріт між контактами, після чого клацнути по цьому дроту правою кнопкою миші та обрати пункт «Convert to labeled connection» як це показано на рис. 3.18.

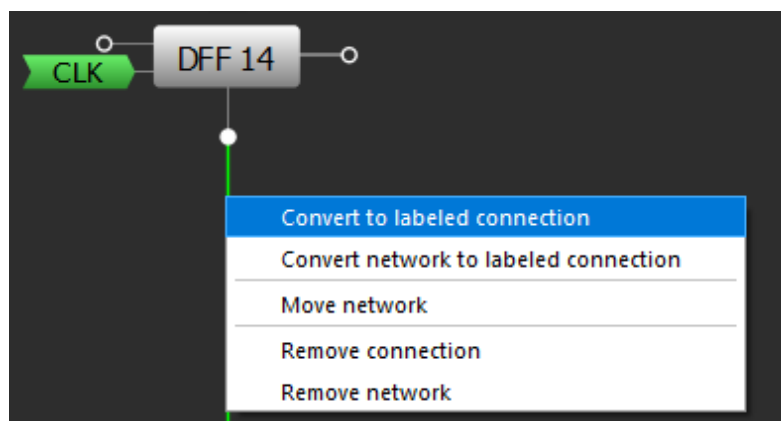


Рисунок 3.18 – Створення лейблового підключення

Щоб змінити підпис створеного лейбла достатньо двічі клацнути по ньому лівою кнопкою миші після чого відкриється вікно у якому записується назва цього лейблу.

Тепер перейдімо безпосередньо до збірки схеми керуючого автомату. Спочатку розставляємо необхідні мітки на пінні мікросхеми, розміщуємо у робочому просторі D-тригери та підключаємо сигнали скидання і синхронізації. Результат зображено на рис. 3.19.



Рисунок 3.19 – Розмітка пінів, розміщення та підключення тригерів

Наступним кроком реалізуємо усі необхідні терми згідно за формулою 2.6. Для цього розмістимо у робочому просторі п'ять LUT на три входи, розставимо до кожного з них мітки з номером терму за який вони відповідають, та підключимо до них виходи тригерів. Щоб було простіше налаштовувати ті LUT до яких повинно входити лише два операнди виходи тригерів підключимо до входів IN1 та IN0, в той час, як усі інші підключимо до IN2 та IN1, це дозволить менше плутатись у майбутньому. У результаті проведених маніпуляцій схема матиме вигляд який продемонстровано на рис. 3.20.

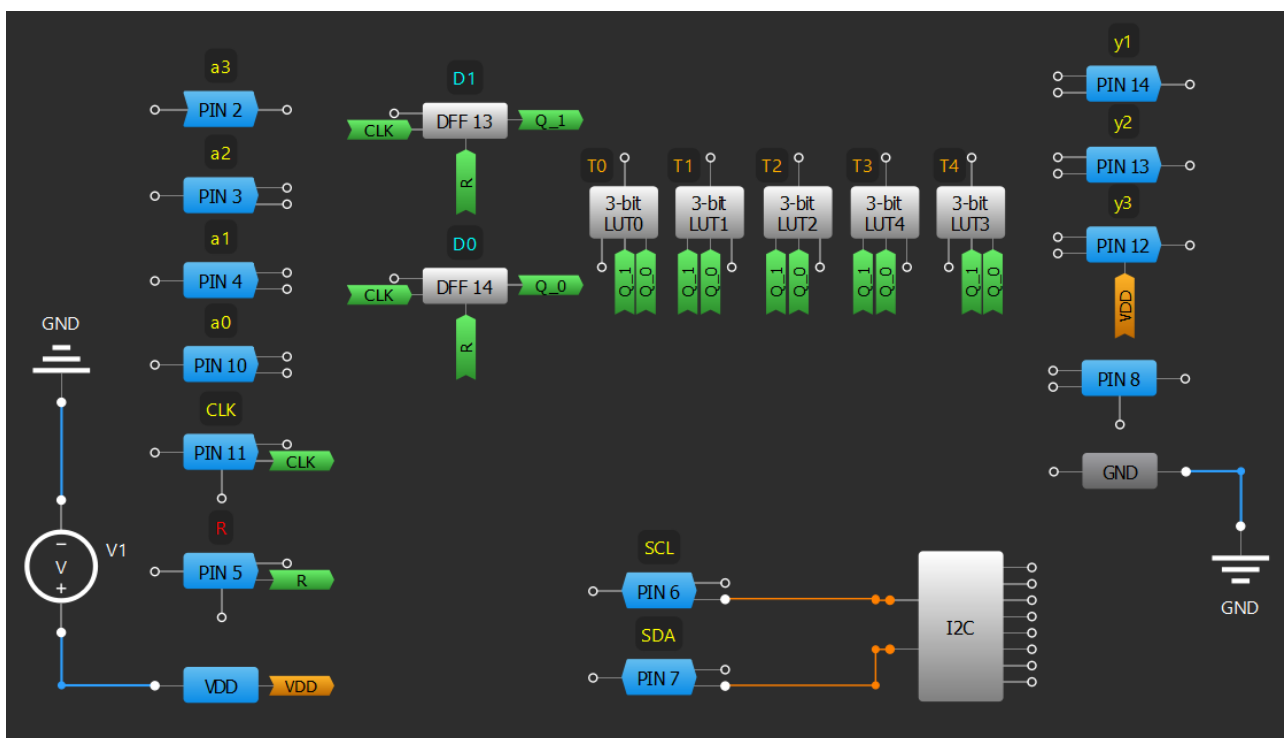


Рисунок 3.20 – Результат реалізації термів

Зараз у всіх LUT буде залишатись один вільний вхід, але пізніше до частини з них будуть підключені сигнали умов згідно з формулою 2.6.

Останнім кроком буде утворення, за допомогою створених у попередньому етапі термів, сигналів збудження тригерів та вихідних сигналів згідно з формулою 2.7. Таким чином отримаємо схему яку зображено на рис. 3.21.

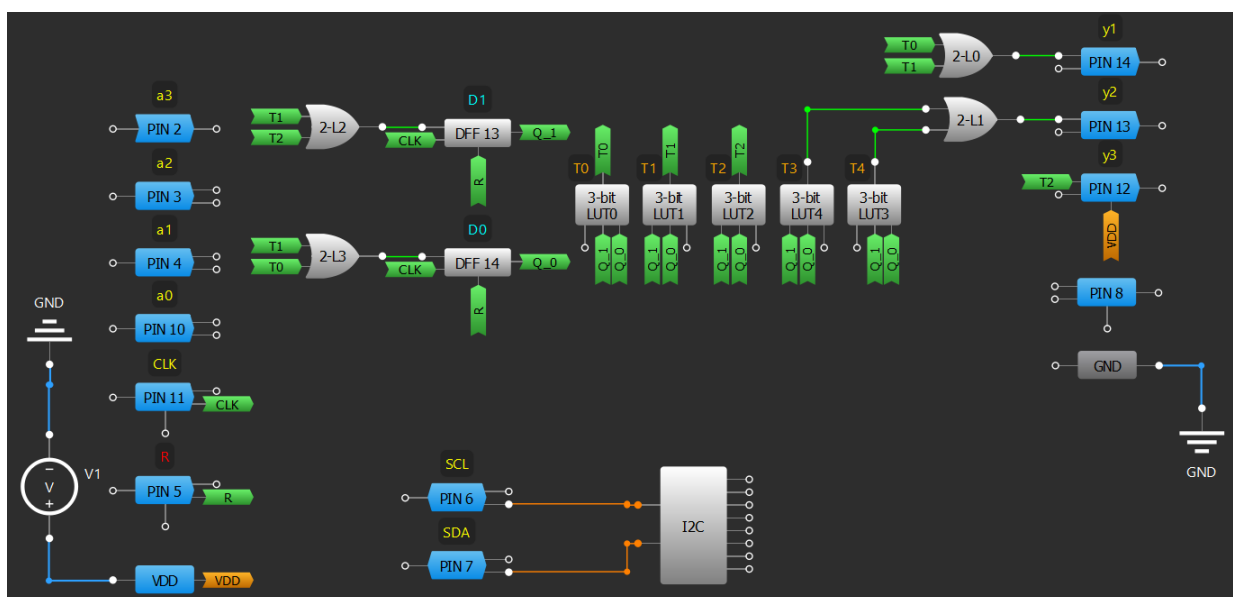


Рисунок 3.21 – Схема керуючого автомату

3.2.4 Підготовка схеми до тестування

У цьому випадку підготовка до тестування схеми керуючого автомата проводиться так само як і для операційного автомата. Тобто нам лише потрібно до усіх вхідних пінів мікросхеми підключити зовнішні джерела напруги. Результат продемонстровано на рис. 3.22.

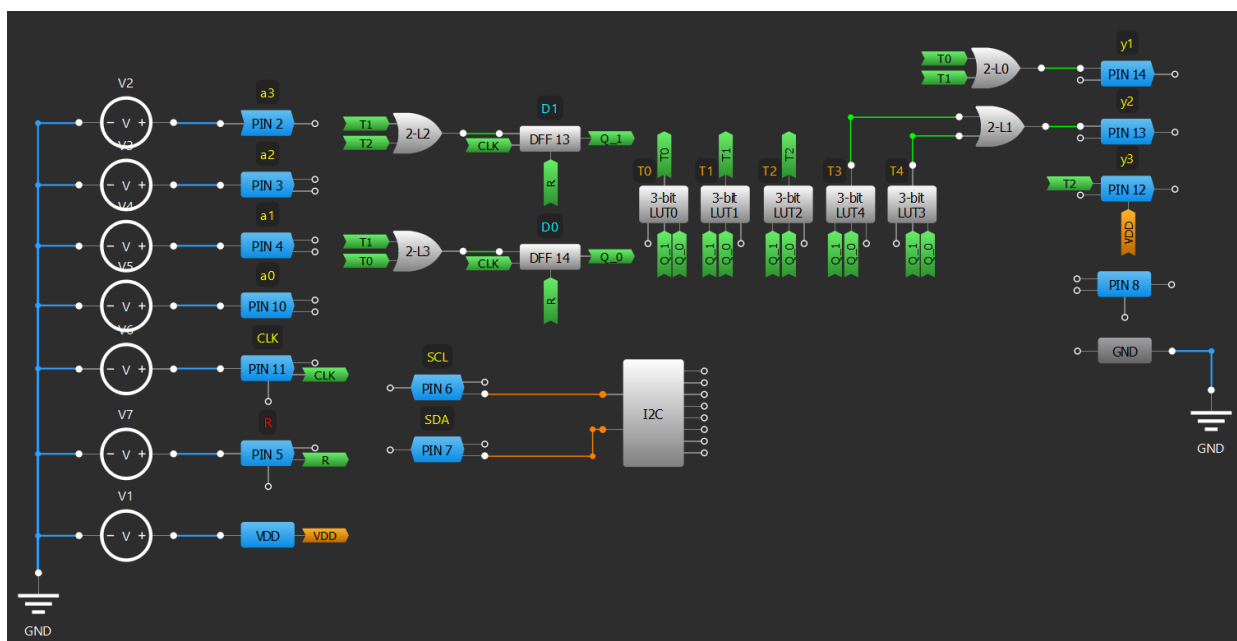


Рисунок 3.22 – Схема керуючого автомата підготовлена до тестування

3.3 Моделювання компараторів

Проаналізувавши формули 2.15 та 2.16 можемо зробити висновок, що для побудови компараторів буде достатньо двох LUT на чотири входи та одного на два входи. Отже, усього для двох компараторів нам необхідно лише три LUT, тому виділення для них окремої мікросхеми виглядатиме як марнотратство. Тож, згадуючи те, що у мікросхемі, в якій розміщений керуючий автомат, залишилось достатньо вільних LUT, і компаратори порівнюють лише вхідне число з певною константою можемо дійти висновку що доцільним буде реалізувати компаратори у тій самій мікросхемі, у якій вже було зібрано керуючий автомат.

3.3.1 Налаштування LUT та збірка компараторів

Спочатку проведемо налаштування для компаратора за формулою 2.15, для цього візьмемо два LUT на чотири входи. Перший буде реалізовувати другу половину формули, тобто $a_3 * a_2 * \text{not}(a_1) * a_0$. Таблиця істинності цього елемента відображена у табл. 3.7.

Таблиця 3.7 – Таблиця істинності LUT для реалізації другої половини формули 2.15

a_3 (IN3)	a_2 (IN2)	a_1 (IN1)	a_0 (IN0)	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Далі налаштуємо LUT для реалізації першої половини формули, тобто $a_3 * a_2 * a_1$. Враховуючи те, що у цій частині формули лише три сигнали, четвертий вхід LUT використаємо для того, щоб реалізувати операцію АБО між двома половинами формули під'єднавши до цього четвертого входу другого LUT вихідний сигнал першого LUT. Таблицю істинності другого LUT відображено у табл. 3.8.

Таблиця 3.8 – Таблиця істинності LUT для остаточного формування необхідного сигналу за формулою 2.15

a_3 (IN3)	a_2 (IN2)	a_1 (IN1)	a_0 (IN0)	OUT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Наступним налаштуємо LUT для реалізації формули 2.16. Для цього нам було б достатньо LUT на два входи, проте оскільки у використуванні для керуючого автомата мікросхеми вони закінчились налаштуємо LUT на три входи. Тож, як й в усіх попередніх налаштуваннях, нам необхідно і достатньо у колонці OUT проставити значення «1» навпроти усіх комбінацій які нам підходять згідно з формулою 2.16. Таблицю істинності цього елементу відображено у табл. 3.9.

Таблиця 3.8 – Таблиця істинності LUT формування необхідного сигналу за формулою 2.16

a_1 (IN2)	a_0 (IN1)	IN0	OUT
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Тепер залишається розмістити усі необхідні LUT у робочому просторі мікросхеми, налаштувати їх згідно з табл. 3.7-3.9 та підключити до необхідних сигналів. Результат збірки компараторів продемонстровано на рис. 3.23.

Згадаємо також про те, що через відсутність під час виконання попереднього розділу компараторів ми не підключали сигнали умов до термів керуючого автомата. Тепер коли компаратори зібрані ми можемо підключити їх виходи до необхідних термів спираючись на формули 2.15. Результат зображено на рис. 3.24.

Рисунок 3.24 – Результируючий вигляд схеми керуючого автомата

3.4 Тестування змодельованих пристроїв

Тестування є одним з основних етапів проєктування будь-чого. Тестування особливо важливе при розробці мікросхем, де дуже просто заплутатись серед безлічі різних сигналів. Саме на цьому етапі виявляється більша частина помилок пов'язаних здебільшого або з неправильним підключенням сигналів до входу елементу, або з неправильними розрахунками.

У пакеті Go Configure Software Hub для тестування пристроїв ми можемо проводити симуляцію їх роботи. Для цього нам спочатку необхідно підключити до пінів мікросхеми джерела напруги, що ми й робили на етапі підготовки автоматів до тестування.

У якості прикладу виконаємо симуляцію роботи деякого простого пристрою. Для цього у будь-якій мікросхемі зберемо просту схему з формулою $y = (A \vee B) * C$. Цю схему, вже з підключеними зовнішніми джерелами напруги, зображено на рис. 3.25.

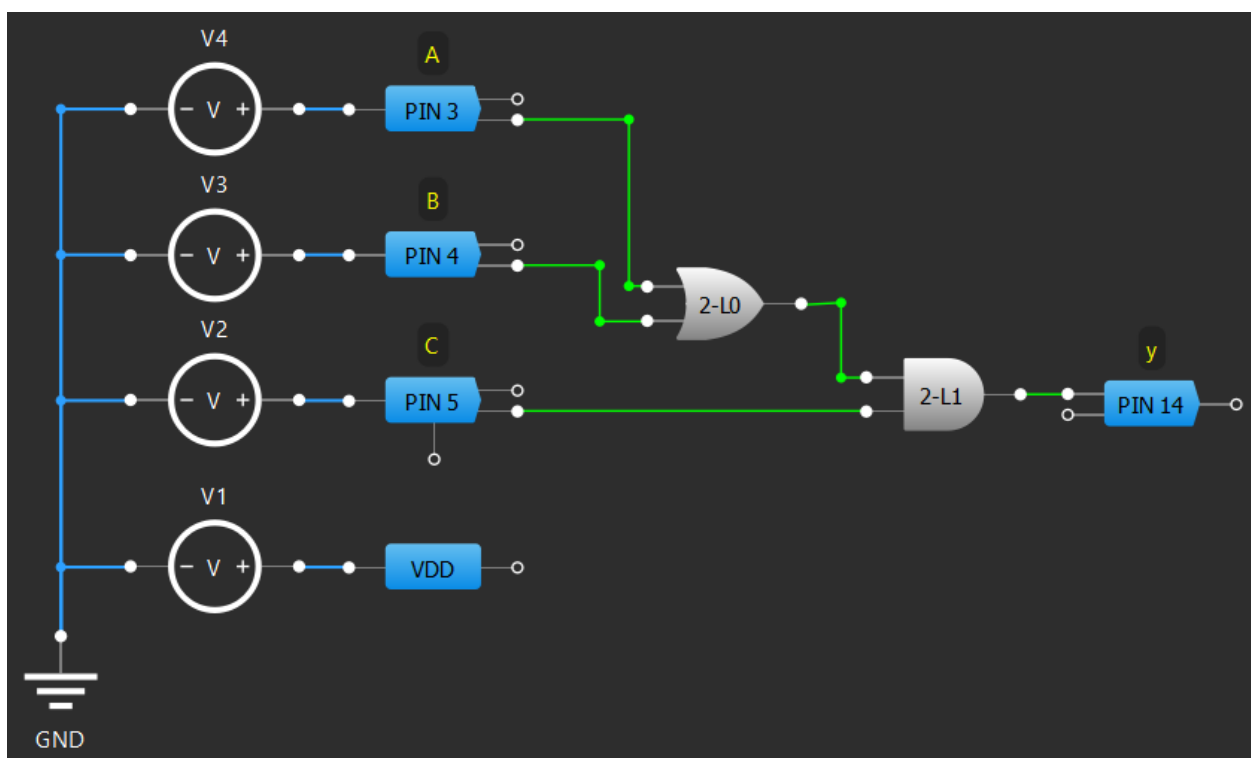


Рисунок 3.25 – Схема прикладу для тестування

Далі знаходимо на панелі швидкого доступу іконку з написом Debag та натискаємо на неї. Однак для проведення симуляції у мікросхеми повинні бути задані налаштування проєкту, у разі якщо вони не задані, програма сповістить про це та відкриє вікно для введення цих налаштувань. У результаті відкриється невелике вікно де ми можемо обрати спосіб за допомогою якого будемо проводити симуляцію роботи пристрою. Для обраної зараз мікросхеми SLG46536V доступні способи проведення симуляцій зображено на рис. 3.26.

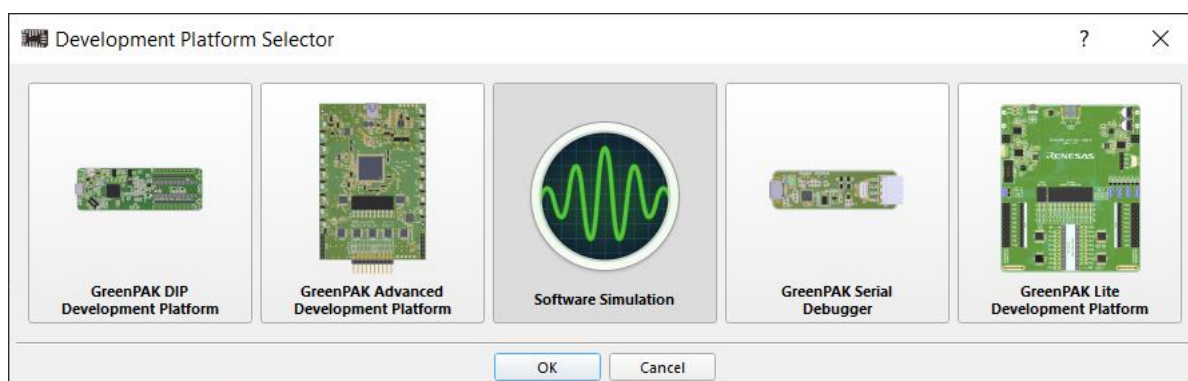


Рисунок 3.26 – Вікно вибору способу тестування

Обираємо Software Simulation, після чого у правій частині вікна програми відкриється вікно відображене на рис. 3.27.

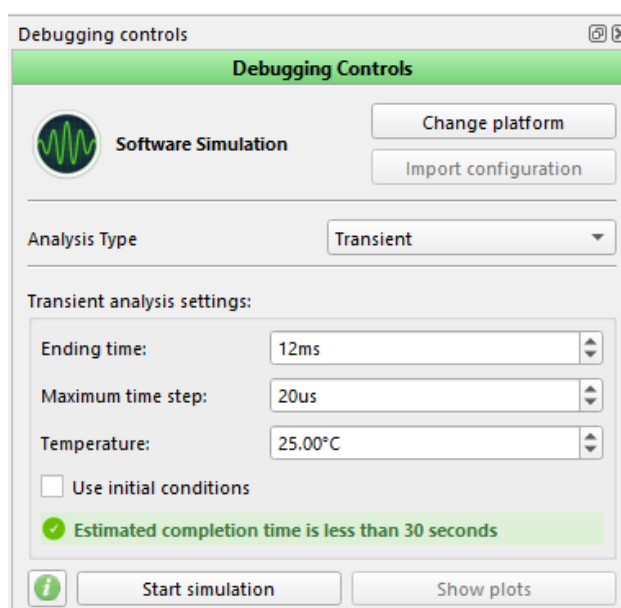


Рисунок 3.27 – Вікно налаштування симуляції

Наступним кроком необхідно налаштувати кожне джерело напруги. Для цього подвійним кліком по будь-якому з них відкриємо вікно налаштування яке продемонстровано на рис. 3.30.

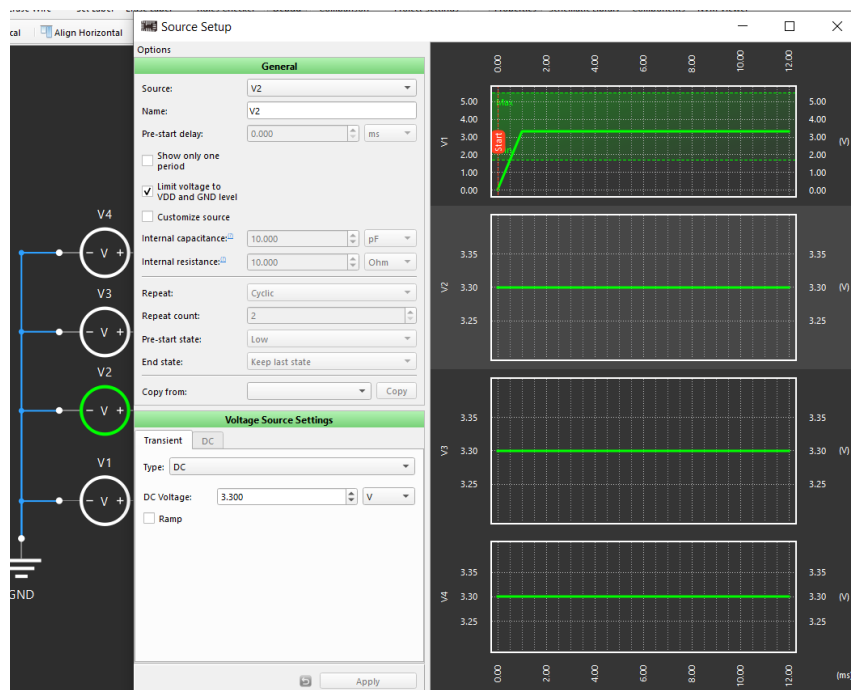


Рисунок 3.30 – Вікно налаштування джерел напруги

На рис. 3.30 ми можемо побачити список джерел напруги які розміщені у робочому просторі мікросхеми та деякі налаштування які заблоковані через виставлений за замовчуванням тип сигналу джерела. Від цього типу залежить те, як у ході симуляції буде змінюватись сигнал який генерує налаштовуване джерело напруги. Список доступних типів сигналу продемонстровано на рис. 3.31.

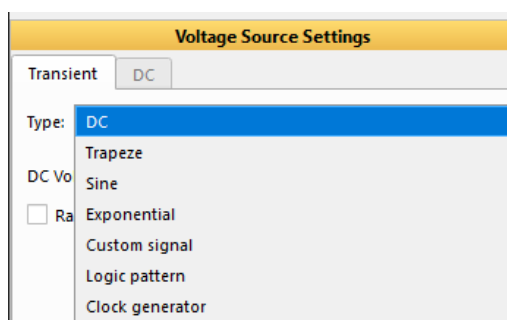


Рисунок 3.31 – Список типів сигналу джерела напруги

Для проведення симуляцій роботи пристроїв заснованих на Булевій алгебрі найкращім варіантом буде використання типу Logic pattern для звичайних сигналів та Clock generator для синхросигналу. Оскільки у прикладі який ми розглядаємо немає сигналів синхронізації усі три джерела напруги перемикаємо на тип Logic pattern.

Тепер залишається лише налаштувати патерни зміни сигналів які генерують джерела напруги. Для цього скористаємось меню Voltage Source Settings яке після зміни типу генерованого сигналу змінилось. Це меню відображено на рис. 3.32.

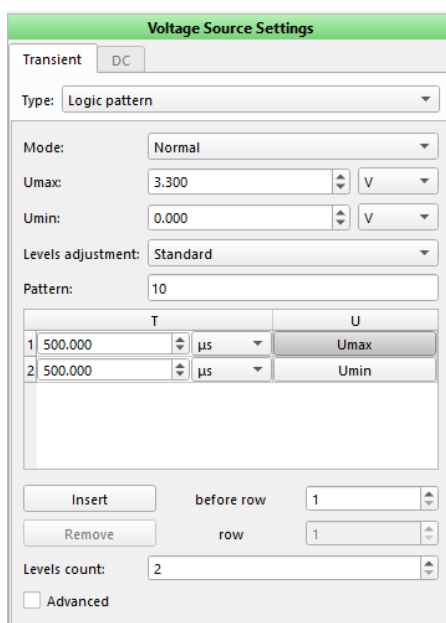


Рисунок 3.32 – Меню налаштування джерела напруги із типом вихідного сигналу Logic pattern

У меню відображеному на рис. 3.32 ми можемо налаштувати:

- Mode, яким буде сигнал по відношенню до налаштувань, прямим або інверсним;
- максимальний й мінімальний рівень напруги та їх одиниці вимірювання;
- патерн сигналу у вигляді двійкового числа, яке відображає стан сигналу, мінімальний («0») або максимальний («1») та задає порядок зміни станів;
- час який сигнал буде перебувати у кожному з станів патерну.

Оскільки у даному прикладі у нас всього три входи налаштуємо джерела напруги таким чином, щоб перебрати усі комбінації вхідних сигналів. Для цього буде достатньо збільшувати тривалість усіх станів кожного наступного сигналу удвічі. Тобто якщо встановити для сигналу А тривалість кожного стану у п'ять мілісекунд, то для сигналу В тривалість станів буде мати значення десять мілісекунд, а для сигналу С – 20 мілісекунд. Результат такого налаштування можна побачити на рис. 3.33. Через те, що період сигналу С буде 40 мілісекунд нам необхідно збільшити тривалість симуляції, щоб вона була як мінімум ці 40 мілісекунд і жоден з патернів не було перервано передчасно.

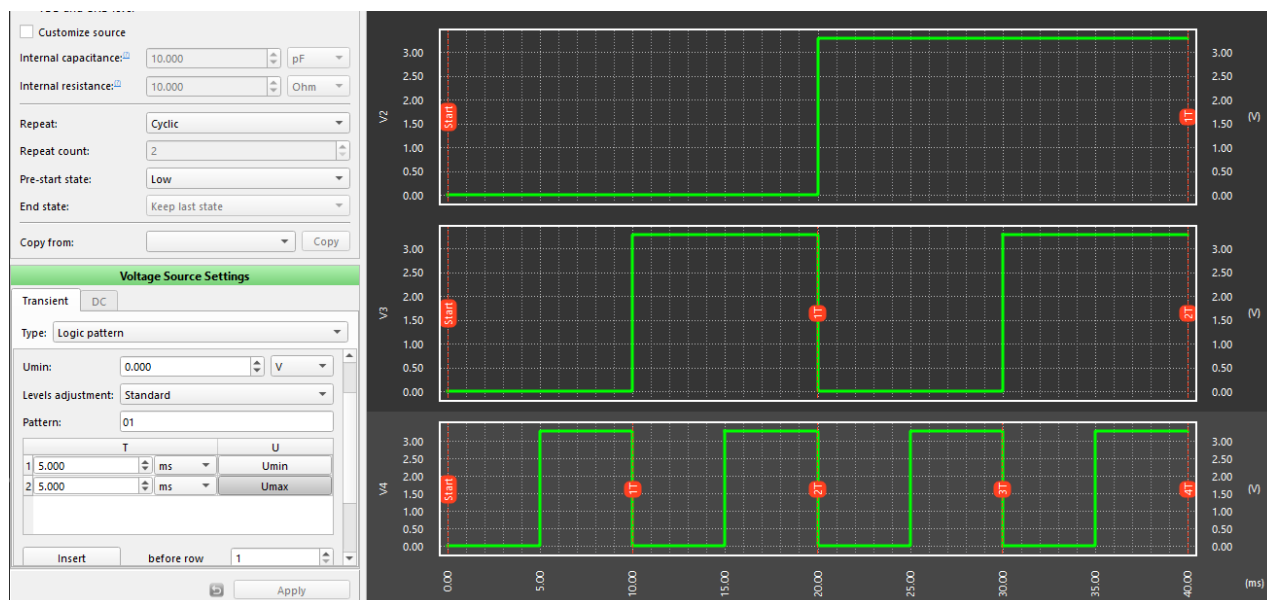


Рисунок 3.33 – Приклад налаштування сигналів

Тепер коли ми налаштували вхідні сигнали нам залишилось запустити симуляцію натиснувши на Start simulation яку ми можемо побачити на рис. 3.27. Після завершення симуляції її результат буде відображено у вигляді часових діаграм кожного з поставлених нами до цього «щупів», за їх допомогою ми зможемо проаналізувати зміни кожного з сигналів що надалі може допомогти виявити помилки, якщо такі є. Результат роботи схеми прикладу наведено на рис. 3.34.

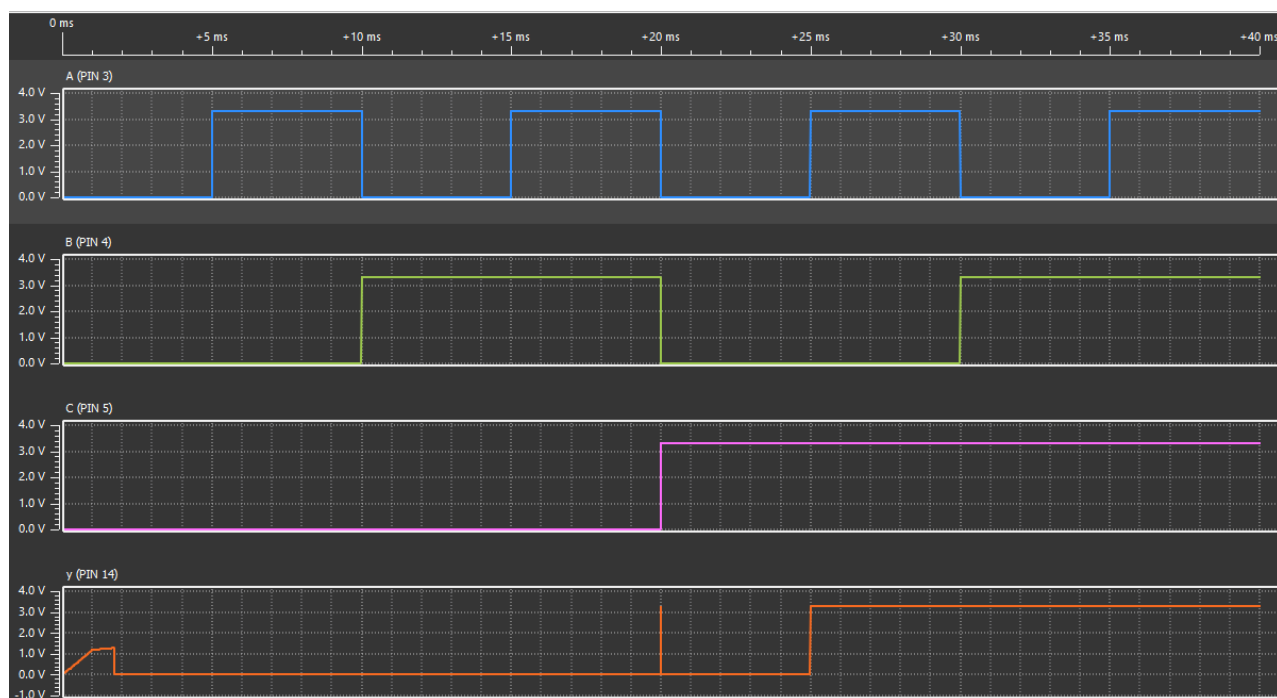


Рисунок 3.34 – Часова діаграма роботи схеми прикладу

Тепер можемо переходити до тестування безпосередньо тих пристроїв які моделювались протягом даного розділу.

3.4.1 Тестування операційного автомата

Для проведення тестування операційного автомата шляхом симуляції його роботи нам буде необхідно провести налаштування усіх вхідних сигналів та розставити «щупи» на контактах вхідних та вихідних сигналів. Розстановку «щупів» зображено на рис. 3.35.

У вікні налаштування симуляції тривалість симуляції виставимо у 100 мілісекунд, чого має вистачити для тестування усіх мікрооперацій, а значення максимального часового інтервалу та температуру залишимо стандартними.

Окремо зазначимо про сигнали CLK(сигнал синхронізації) та a_3 - a_0 (біти вхідного числа). Для сигналу CLK виставимо тип сигналу Clock generator, у налаштуваннях якого виставимо частоту у 100 Гц. Для сигналів a_3 - a_0 налаштуємо комбінацію «0110» яка дозволить нам протестувати усі операції які може виконувати операційний автомат. Налаштування більшості вхідних сигналів відображено у табл. 3.9.

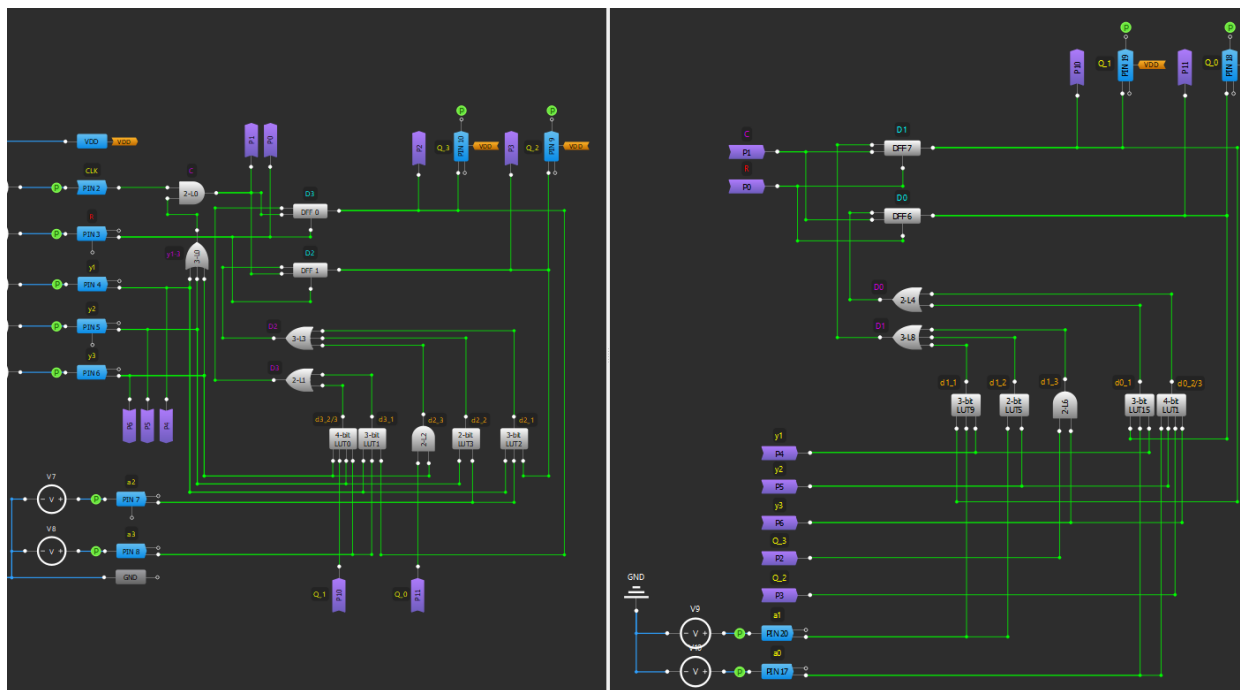


Рисунок 3.35 – Розстановка «щупів» для тестування операційного автомата

Таблиця 3.9 – Налаштування вхідних сигналів операційного автомата

Сигнал	Патерн	Тривалість станів, ms		
		стан 1	стан 2	стан 3
R	01	9	91	—
y ₁	010	10	20	70
y ₂	010	40	20	40
y ₃	010	70	20	10
a ₃	00	50	50	—
a ₂	01	10	90	—
a ₁	01	10	90	—
a ₀	00	50	50	—

Тепер залишається ввімкнути симуляцію, дочекатись її завершення та проаналізувати отриману у ході симуляції часову діаграму. Цю часову діаграму зображено на рис. 3.36. Завдяки аналізу цієї діаграми ми зможемо зрозуміти чи правильно операційний автомат виконує кожну з мікрооперацій.

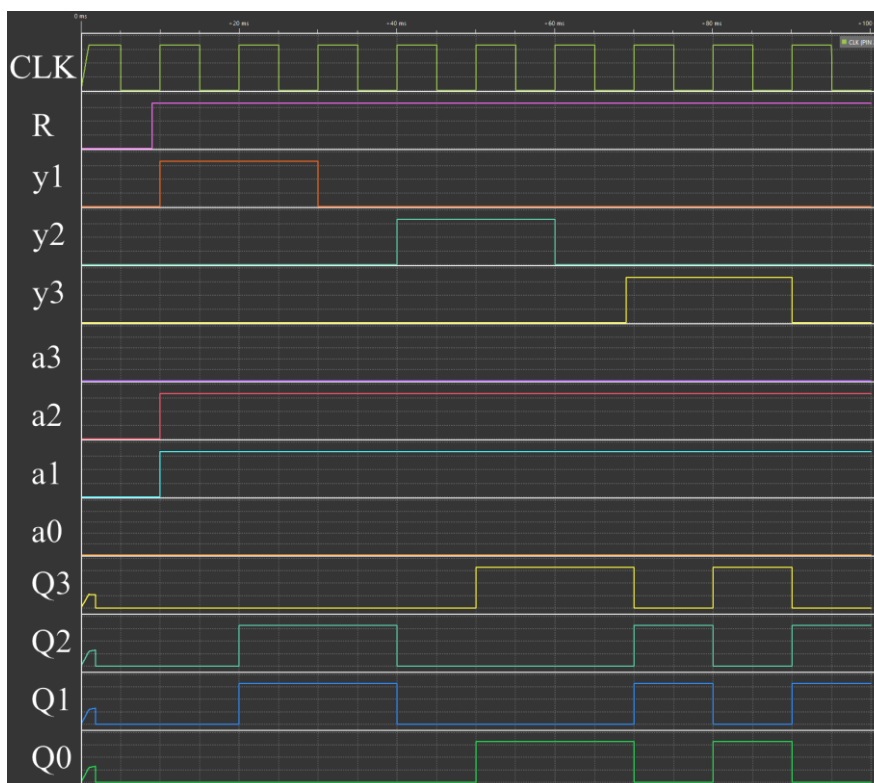


Рисунок 3.36 – Часова діаграма симуляції роботи операційного автомата

3.4.2 Тестування компараторів

Оскільки компаратори вбудовані у схему керуючого автомату нам необхідно спершу протестувати саме їхню роботу перед тим як тестувати керуючий автомат.

Для тестування компараторів шляхом проведення симуляції їх роботи нам буде достатньо налаштувати чотири вхідні сигнали, які являють собою біти вхідного числа A , та розставити «щупи» на контактах вхідних та вихідних сигналів. У вікні налаштування симуляції тривалість симуляції виставимо у 65 мілісекунд, а значення максимального часового інтервалу та температуру залишимо стандартними. Розстановка «щупів» продемонстрована на рис. 3.37. Згідно з умовами які реалізують ці компаратори можемо дійти висновку, що нам буде достатньо налаштувати вхідні сигнали на три різні комбінації, «1110» для задовільнення умови першого компаратора, «0110» для отримання логічних нулів від обох компараторів та «0001» для задовільнення умови другого компаратора. Налаштування вхідних сигналів відображено у табл. 3.10.

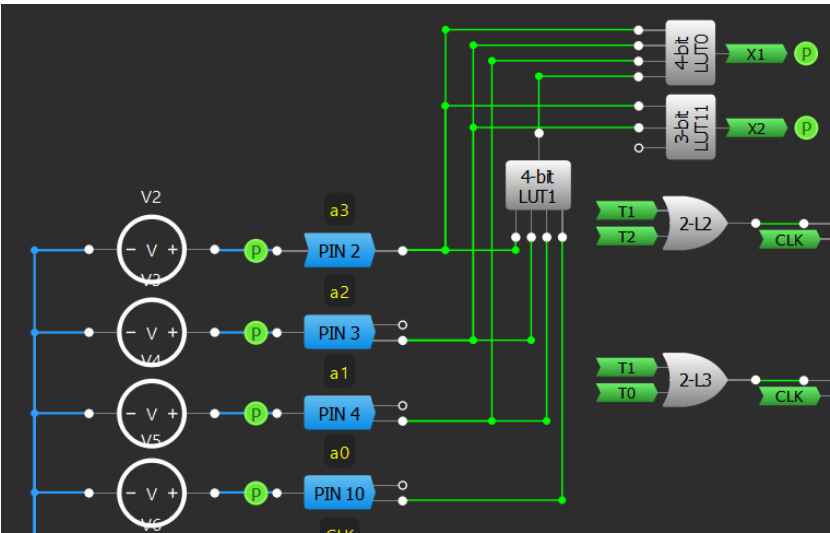


Рисунок 3.37 – Розстановка «щупів» для тестування компараторів

Таблиця 3.10 – Налаштування вхідних сигналів компараторів

Сигнал	Патерн	Тривалість станів, ms		
		стан 1	стан 2	стан 3
a ₃	010	5	20	40
a ₂	010	5	40	20
a ₁	010	5	40	20
a ₀	01	45	20	—

Налаштувавши сигнали можемо запускати симуляцію. Результуючу часову діаграму зображено на рис. 3.38.

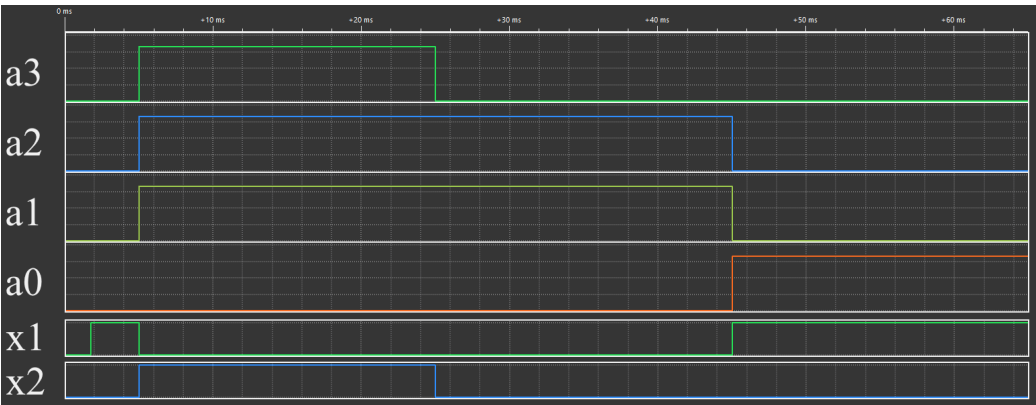


Рисунок 3.38 – Часова діаграма симуляції роботи компараторів

3.4.3 Тестування керуючого автомата

Для проведення тестування керуючого автомата шляхом симуляції його роботи нам буде необхідно провести налаштування усіх вхідних сигналів та розставити «щупи» на контактах вхідних та вихідних сигналів. Розстановку «щупів» зображено на рис. 3.39.

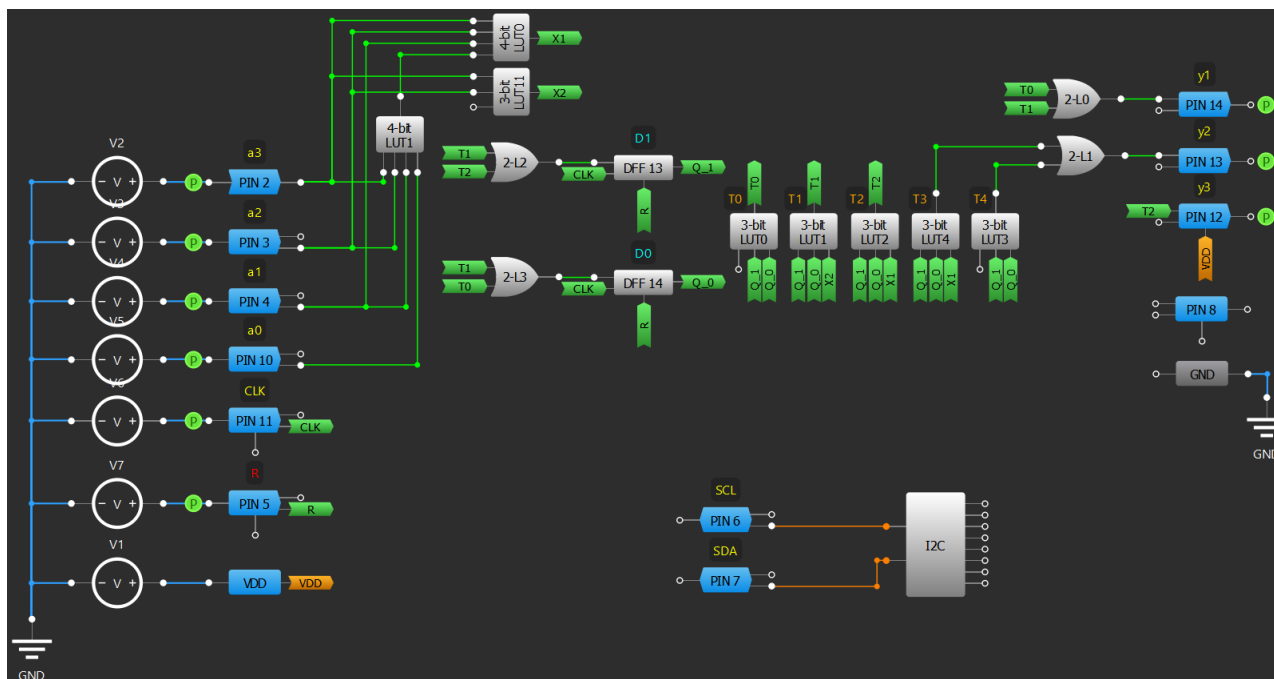


Рисунок 3.39 – Розстановка «щупів» для тестування керуючого автомата

У вікні налаштування симуляції виставимо її тривалість у 165 мілісекунд, чого має вистачити для тестування усіх мікрооперацій, а значення максимального часового інтервалу та температуру залишимо стандартними.

Окремо зазначимо про сигнали CLK(сигнал синхронізації) та a_3 - a_0 (біти вхідного числа). Для сигналу CLK виставимо тип сигналу Clock generator, у налаштуваннях якого виставимо частоту у 100 Гц. Для сигналів a_3 - a_0 необхідно налаштувати три різні комбінації які дозволять нам протестувати усі «гілки» алгоритму керуючого автомата. Спираючись на змістовий граф керуючого автомата, зображений на рис. 2.1, визначимо ці комбінації та запишемо їх разом з очікуваним результатом у табл. 3.11.

Таблиця 3.11 – Таблиця тестування керуючого автомату

Комбінація значень сигналів				Очікуваний результат
a_3	a_2	a_1	a_0	
1	1	0	1	$y_1 \rightarrow y_2$
1	0	0	1	$y_1 \rightarrow y_3$
0	0	1	0	$y_1 \rightarrow y_3 \rightarrow y_1 \rightarrow y_2$

Тепер знаючи комбінації які треба буде налаштувати для сигналів a_3 - a_0 ми можемо скласти таблицю налаштування сигналів керуючого автомата яку відображено у табл. 3.12.

Таблиця 3.10 – Налаштування вхідних сигналів компараторів

Сигнал	Патерн	Тривалість станів, ms						
		стан 1	стан 2	стан 3	стан 4	стан 5	стан 6	стан 7
R	0101010	15	30	10	30	10	60	10
a_3	010	15	70	80	—	—	—	—
a_2	010	15	30	120	—	—	—	—
a_1	010	95	60	10	—	—	—	—
a_0	010	15	70	80	—	—	—	—

Тепер, звершивши налаштування сигналів, ми можемо запустити симуляцію роботи керуючого автомата та отримати у результаті часову діаграму, дані з якої порівняємо з очікуваними значеннями з табл. 3.11. Отриману часову діаграму зображено на рис. 3.40.

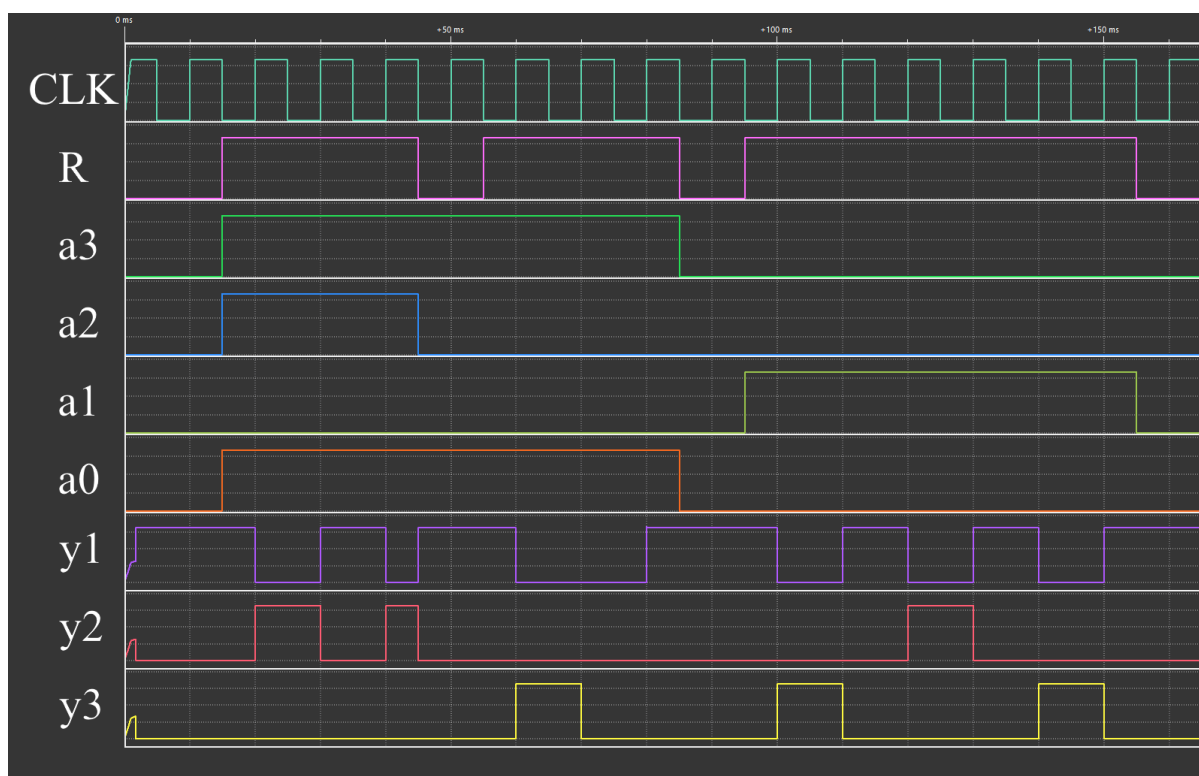


Рисунок 3.40 – Часова діаграма симуляції роботи керуючого автомата

Після проведення симуляції роботи кожного структурного елементу елемента мікрообчислювача ми отримали часові діаграми зображені на рис. 3.36, 3.38 та 3.40. Проаналізувавши кожну з цих діаграм можемо дійти висновку що усі структурні елементи мікрообчислювача працюють відповідно до виконаних у другому розділі розрахунків, та відповідно до алгоритму цього мікрообчислювача.

ВИСНОВКИ

У ході роботи над дипломним проєктом було проаналізовано наявні у мережі Інтернет віртуальні лабораторії інших компаній за для визначення концепції та найбільш оптимального способу розробки віртуальної лабораторії Renesas Electronics.

Було визначено тип пристрою, процес проєктування якого буде використано у якості прикладу необхідного для розробки віртуальної лабораторії. У якості такого пристрою було обрано мікрообчислювач, для якого було розроблено алгоритм та проведено усі розрахунки, необхідні для моделювання цього мікрообчислювача.

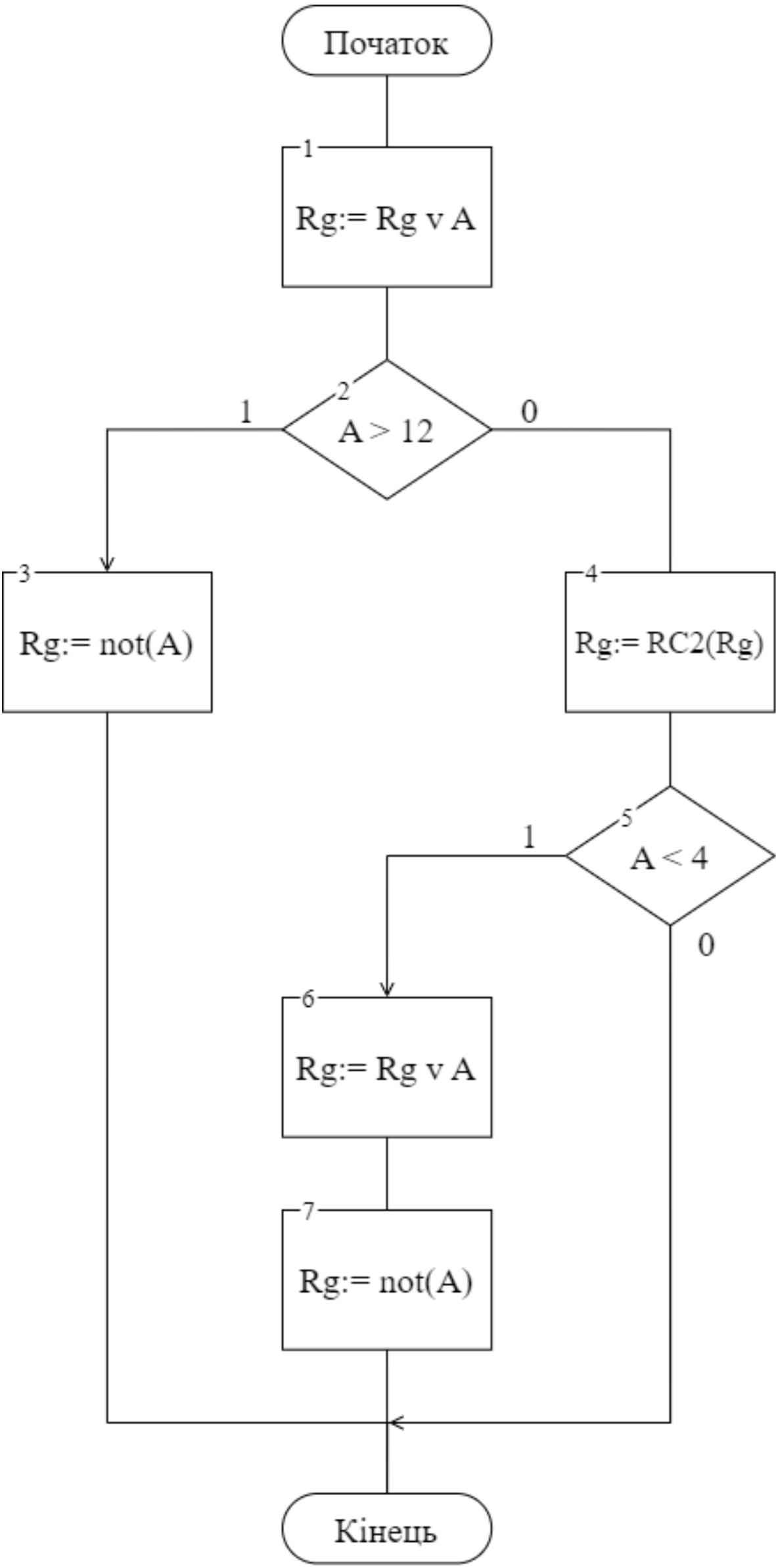
Виконано моделювання розробленого мікрообчислювача у середовищі програми Go Configure Software Hub що належить компанії Renesas Electronics. У ході цього моделювання було докладно описано процес та особливості роботи із цим програмним пакетом.

Виконано тестування усіх структурних елементів мікрообчислювача засобами наявними у програмному пакеті Go Configure Software Hub. Процес тестування цих структурних елементів було докладно описано, що становитиме значну цінність, через відсутність подібного опису у вільному доступі. За результатами тестування можемо дійти висновку що мікрообчислювач працює відповідно до поставленого завдання.

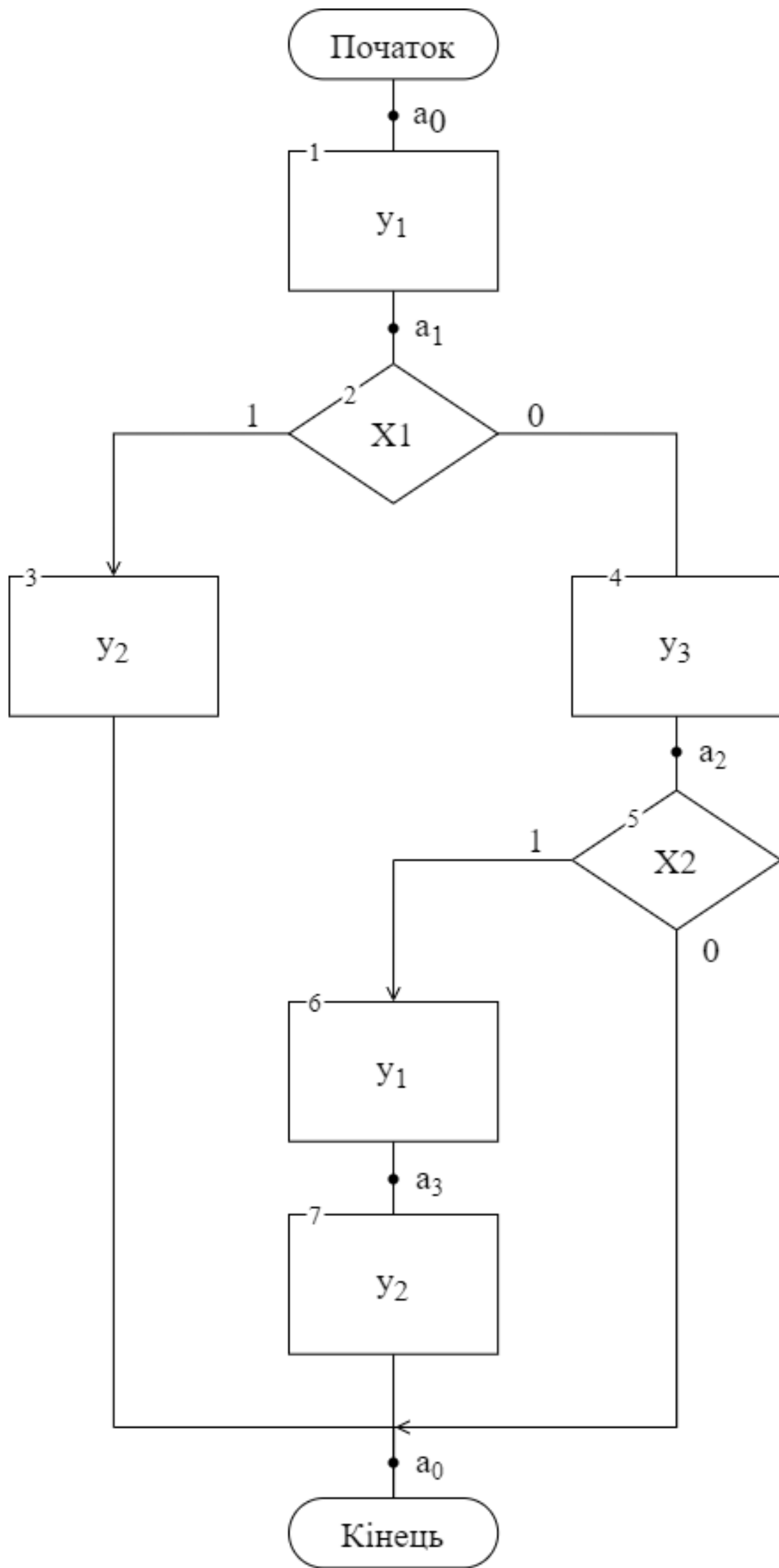
Спираючись на усе вище зазначене можемо зробити висновок, що підготовчий етап для створення віртуальної лабораторії Renesas Electronics завершено.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 The Grid of Online Laboratory Devices Ilmenau. URL: <https://www.goldilabs.net>
- 2 FPGAs in your browser!! URL: <https://medium.com/@m.baladhinesh/fpgas-in-your-browser-bb92be1c1fa3>
- 3 LabsLand - FPGA Laboratory. URL: <https://labsland.com/en/labs/fpga-llstd1>
- 4 Renesas Electronics Corporation. URL: <https://www.renesas.com/eu/en>
- 5 GreenPAK™ Programmable Mixed-signal Products. URL: <https://www.renesas.com/us/en/products/programmable-mixed-signal-asic-ip-products/greenpak-programmable-mixed-signal-products>
- 6 ForgeFPGA Low-density FPGAs. URL: <https://www.renesas.com/us/en/products/programmable-mixed-signal-asic-ip-products/forgefpga-low-density-fpgas>
- 7 Go Configure™ Software Hub. URL: <https://www.renesas.com/us/en/software-tool/go-configure-software-hub>
- 8 Рябенський В. М., Жуйков В. Я., Гулий В. Д. Цифрова схемотехніка: Навч. посібник. — Львів: "Новий Світ-2000", 2009. — 736 с.
- 9 Кочубей О.О. Прикладна теорія цифрових автоматів. Логічні основи : Навч. посіб./ О.О. Кочубей, О.В. Сопільник. - Д.: РВВ ДНУ; Вид-во ДНУ, 2009. - 264 с.
- 10 Бабич М. П., Жуков І. А. Комп'ютерна схемотехніка: Навчальний посібник. – К.: «МК-Прес», 2004. – 412 с.



					13.02070849.51911 A1						
					Розробка підсистеми тренувальної лабораторії Renesas Electronics Змістовий граф алгоритму мікрообчислювача	Лит.			Маса	Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата							
Розроб.		Кухтін К. І.									
Перевір.		Зеленьова І.Я.									
Т.контр.											
						Лист 1			Листів 1		
Н.контр.		Щербак Н.В.				НУ "Запорізька політехніка" КНТ 519					
Затвердж.		Кудерметов Р.К.									



					13.02070849.51911 A2				
					Розробка підсистеми тренувальної лабораторії Renesas Electronics				
Зам.	Лист.	№ докум.	Підп.	Дата	Розмічена граф-схема алгоритму мікрообчислювача				
Розроб.		Кухтін К. І.	<i>Kuch</i>						
Перевір.		Зеленьова І.Я.	<i>Zel</i>						
Т.контр.									
Н.контр.		Щербак Н.В.	<i>Shch</i>						
Затвердж.		Кудерметов Р.К.	<i>Kud</i>						
					Лист 1				
					Листів 1				
					НУ "Запорізька політехніка" КНТ 519				

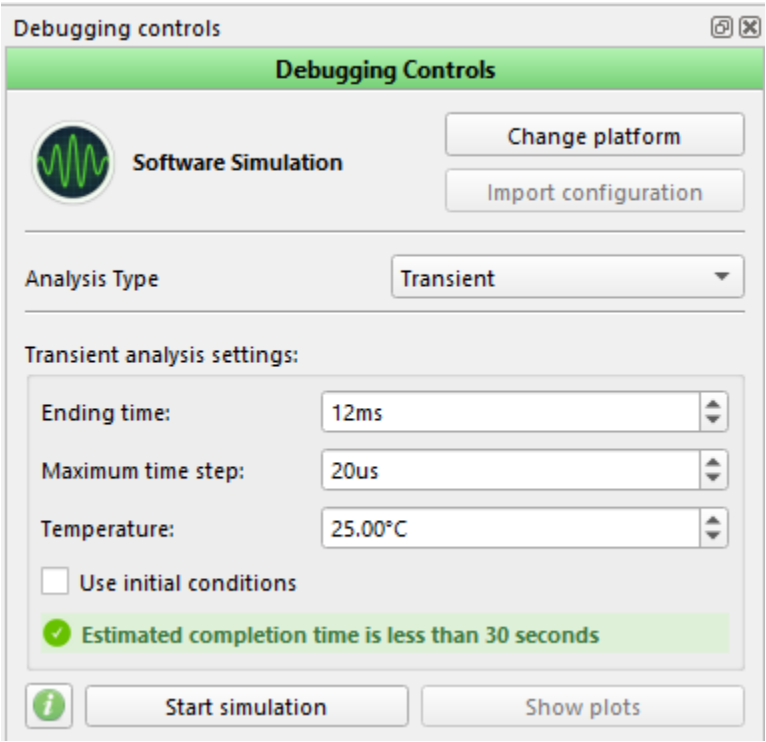


Рисунок 1 – Вікно налаштування симуляції

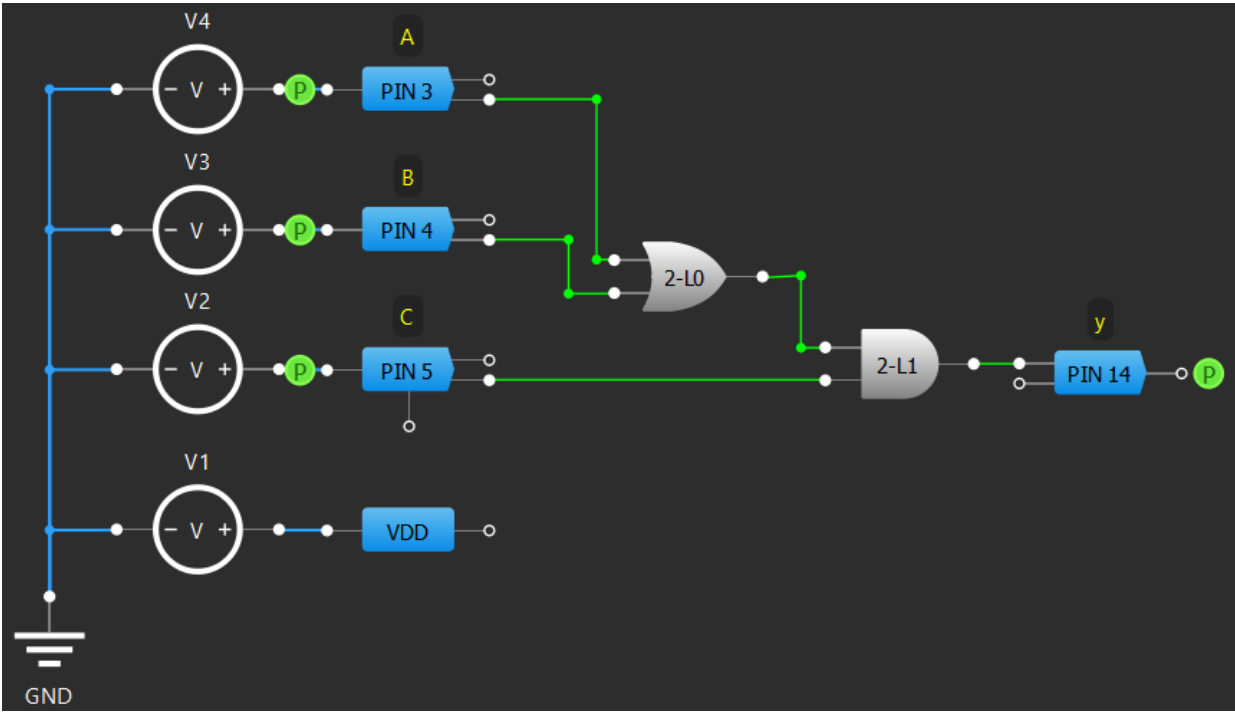


Рисунок 2 – Розстановка «щупів» на контактах схеми

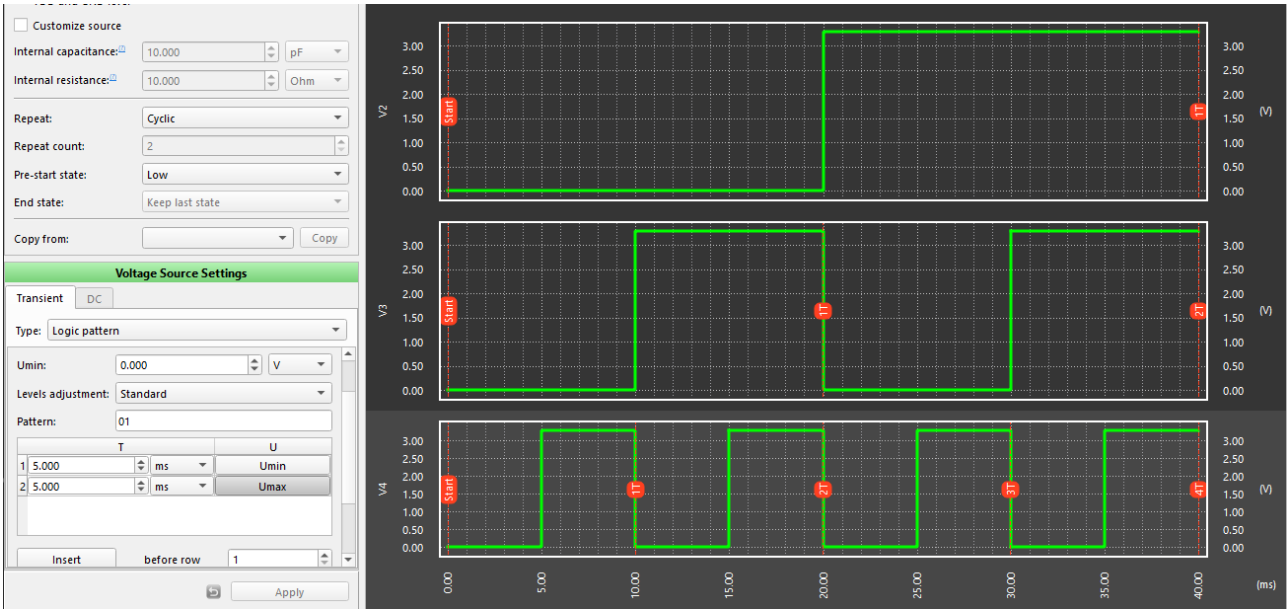


Рисунок 3 – Вікно налаштування сигналів джерел напруги

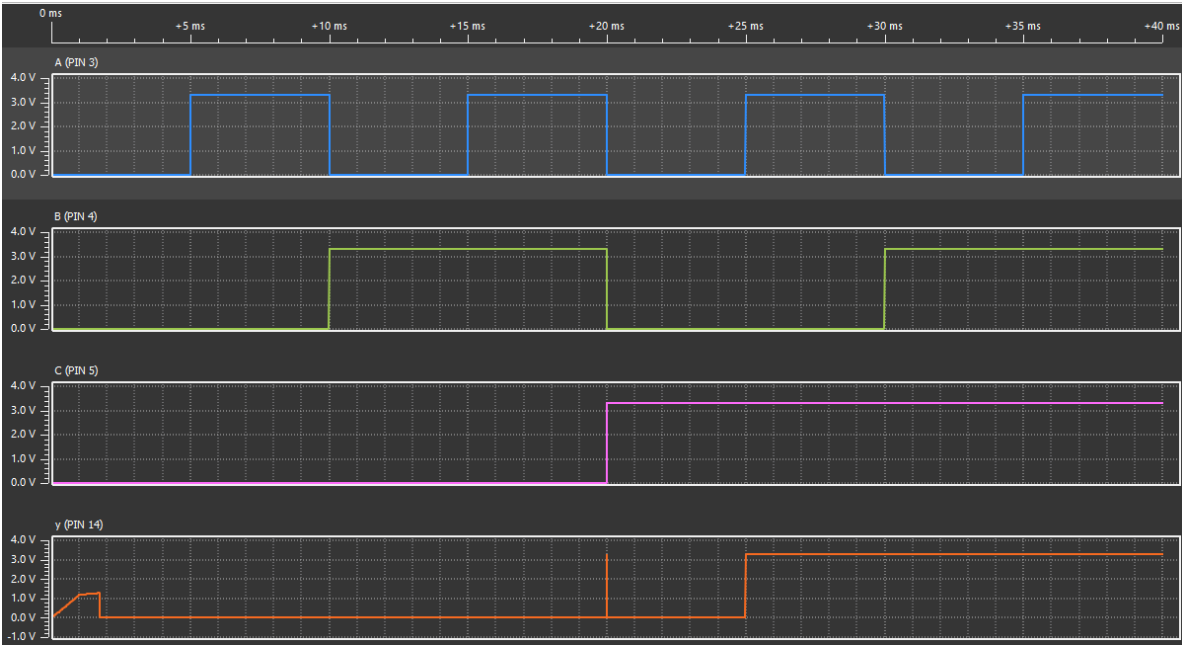


Рисунок 4 – Часова діаграма симуляції роботи пристрою

					13.02070849.51911 ПЛ1							
					Розробка підсистеми тренувальної лабораторії Renesas Electronics	Лит.			Маса		Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата								
Розроб.		Кухтін К. І.	<i>Kuch</i>									
Перевір.		Зеленьова І.Я.	<i>Zel</i>									
Т.контр.					Приклад тестування пристроїв	Лист 1			Листів 1			
						НУ "Запорізька політехніка" КНТ 519						
Н.контр.		Щербак Н.В.	<i>Sh</i>									
Затвердж.		Кудерметов Р.К.	<i>Kud</i>									

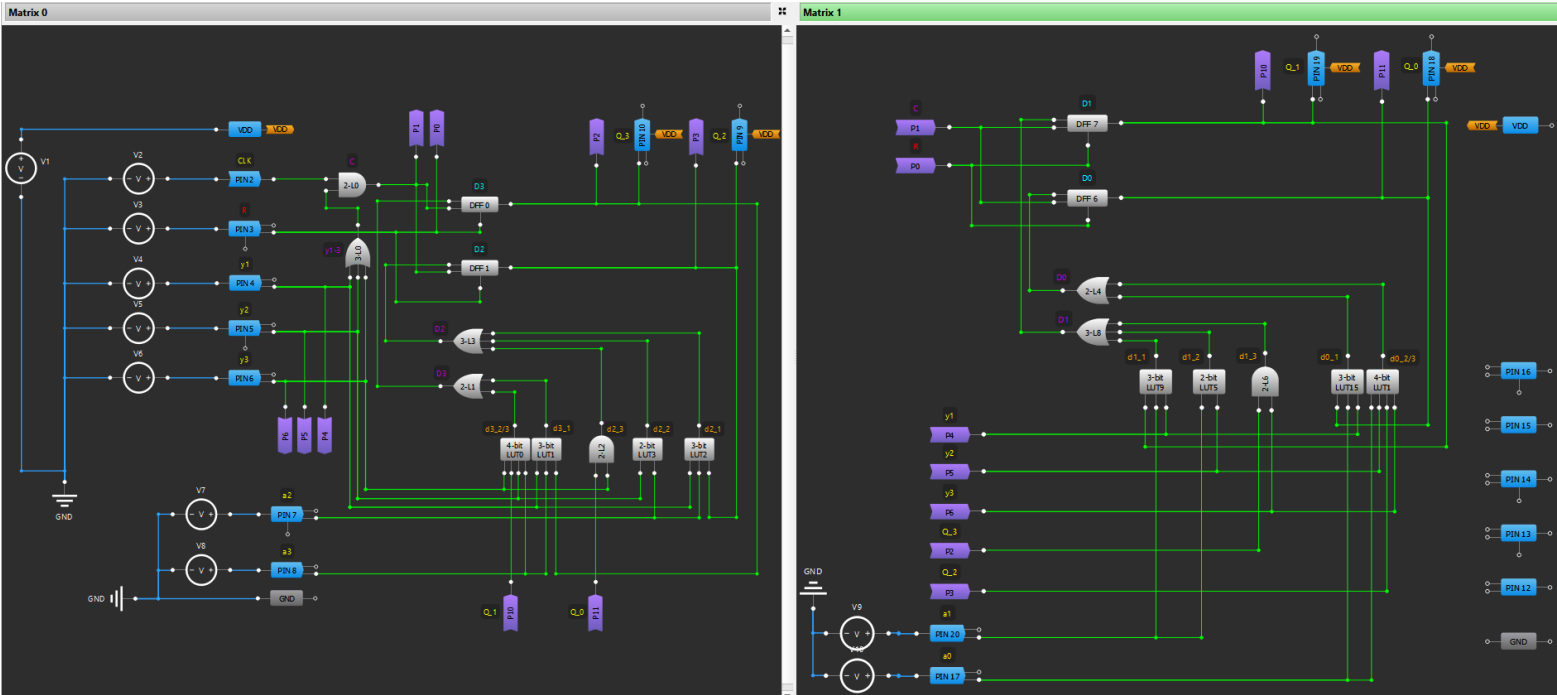


Рисунок 1 – Схема операційного автомата

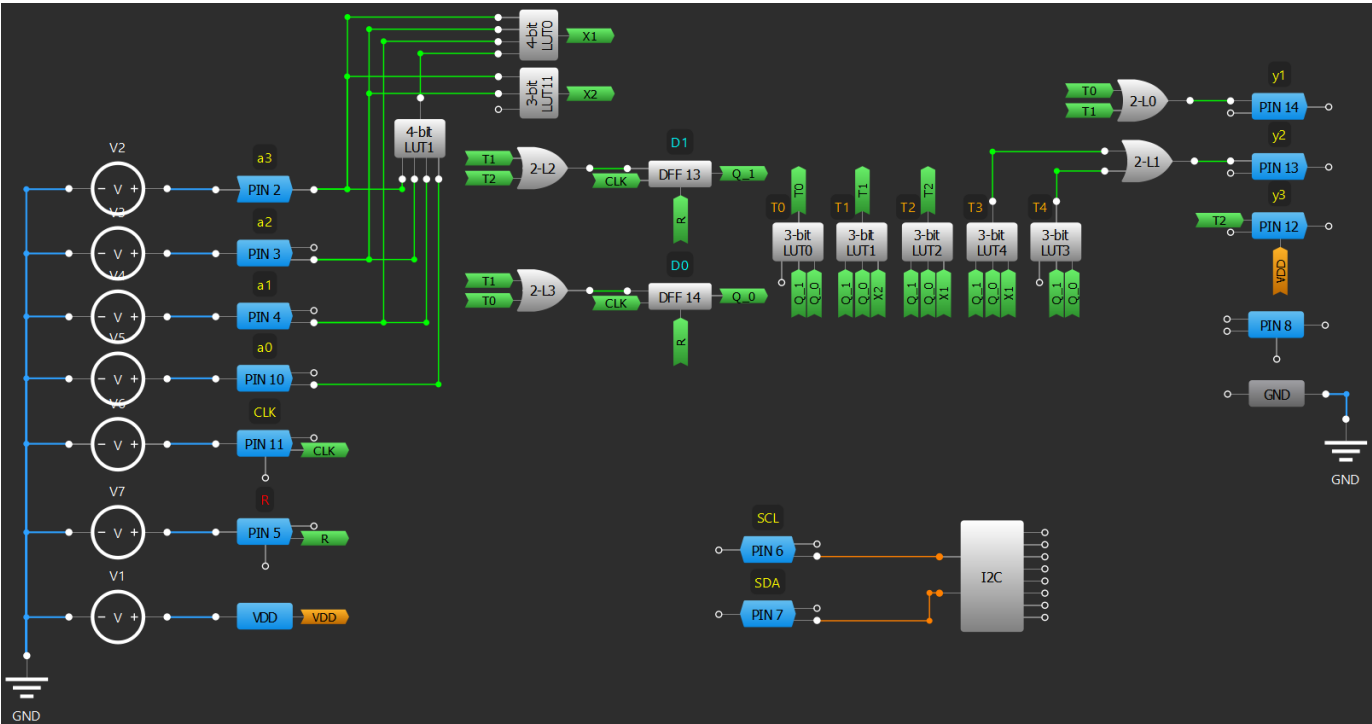


Рисунок 2 – Схема керуючого автомата

					13.02070849.51911 ПЛ2					
					Розробка підсистеми тренувальної лабораторії Renesas Electronics Операційний і керуючий автомати. Схеми електричні функціональні.	Лит.		Маса	Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата						
Розроб.		Кухтін К. І.	<i>Kuch</i>							
Перевір.		Зеленьова І.Я.	<i>Ze</i>							
Т.контр.										
						Лист 1		Листів 1		
Н.контр.		Щербак Н.В.	<i>Sh</i>			НУ "Запорізька політехніка" КНТ 519				
Затвердж.		Кудерметов Р.К.	<i>Kud</i>							

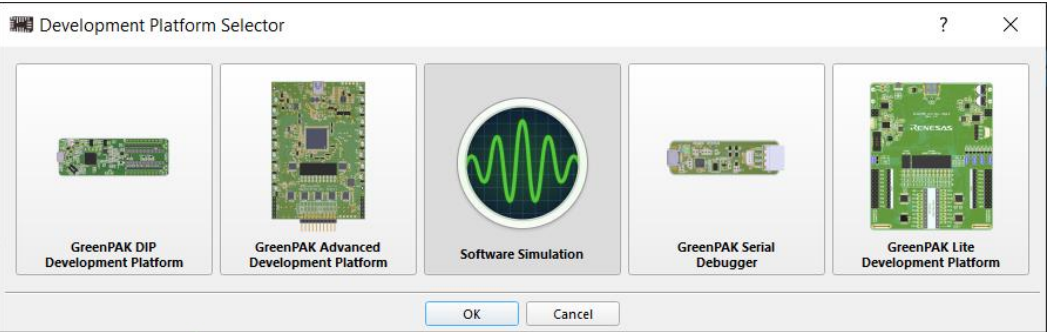


Рисунок 1 – Вікно вибору способу тестування

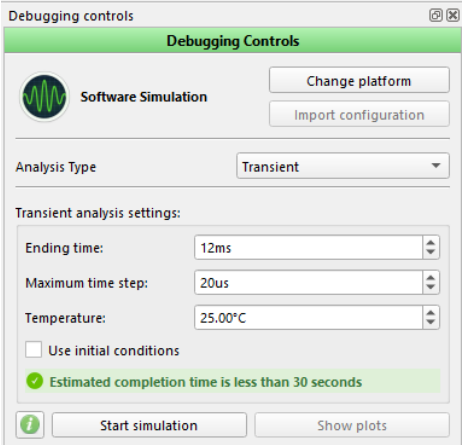


Рисунок 2 – Вікно налаштування симуляції

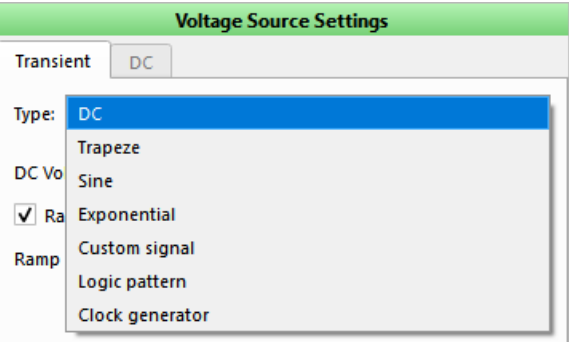


Рисунок 4 – Вікно вибору типу сигналу

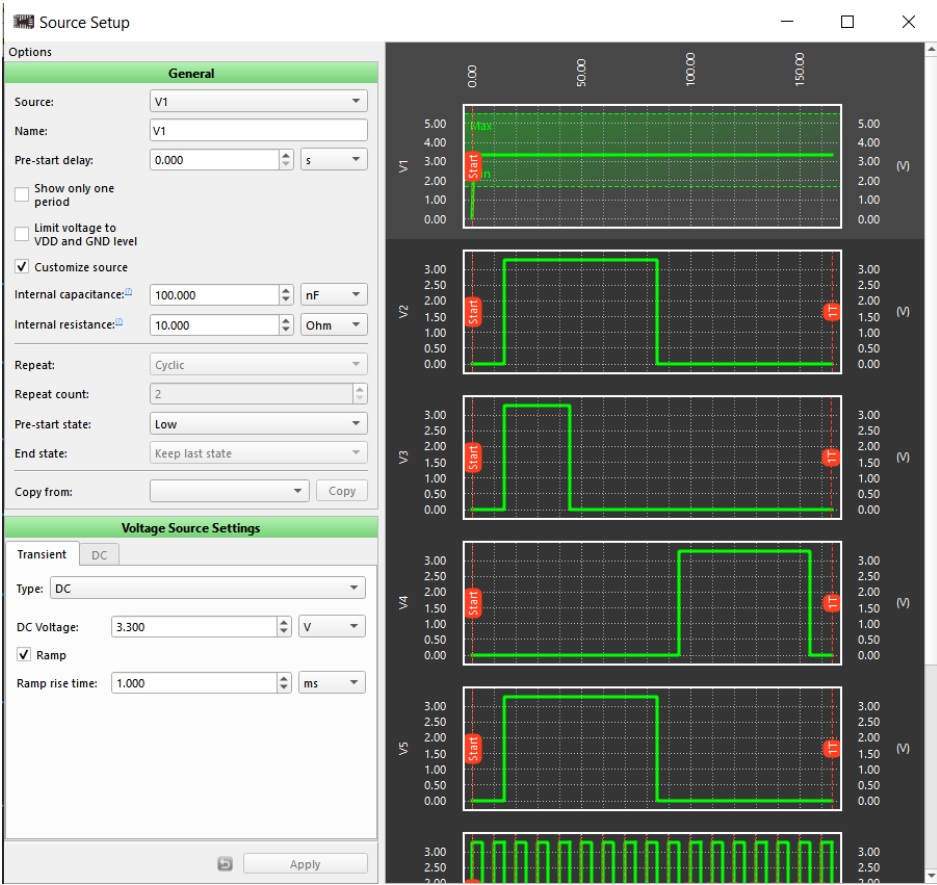


Рисунок 3 – Вікно налаштування сигналів

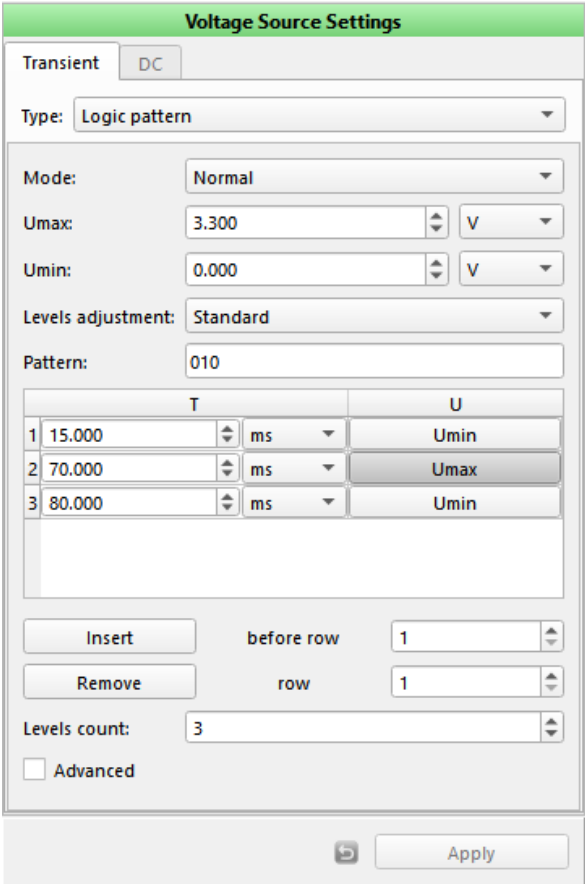


Рисунок 5 – Меню налаштування джерела напруги із типом вихідного сигналу Logic pattern

					13.02070849.51911 ПЛЗ							
					Розробка підсистеми тренувальної лабораторії Renesas Electronics Інструменти для налаштування тестування	Лит.			Маса		Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата								
Розроб.		Кухтін К. І.	<i>Kuch</i>									
Перевір.		Зеленьова І.Я.	<i>Ze</i>									
Т.контр.												
						Лист 1			Листів 1			
Н.контр.		Щербак Н.В.	<i>Sh</i>			НУ "Запорізька політехніка"						
Затвердж.		Кудерметов Р.К.	<i>Kud</i>			КНТ 519						