Київський національний університет імені Тараса Шевченка
Факультет радіофізики, електроніки та комп’ютерних систем
Кафедра комп’ютерної інженерії

«Розробка демонстраційного макету пристрою, що видає решту»

Кваліфікаційна робота бакалавра
студента спеціальності
123 «комп'ютерна інженерія»
Владислава ШАПОВАЛЕНКА
_____(підпис)

Науковий керівник,
канд. фіз.-мат. наук, доцент
Олександр БАУЖА
_____(підпис)

Рецензент
канд. фіз.-мат. наук, доцент
Богдан СУСЬ
_____(підпис)

До захисту допускаю
Протокол засідання кафедри від
“ ” 2022р. №

Завідувач кафедри
к.ф.-м.н., доцент
Юрій БОЙКО

КИЇВ 2022

РЕФЕРАТ

Кваліфікаційна робота бакалавра: – 60 сторінок, 23 рисунки, 4 таблиці, 12 джерел, 3 додатки.

В даній кваліфікаційній роботі бакалавра було розроблено операційний та керуючий автомат для моделі пристрою, що видає решту. Розроблені автомати можуть бути використані для збірки пристрою на базі мікроконтролера з використанням логічних мікросхем 74-ї серії. Побудований пристрій в майбутньому може слугувати для демонстрації практичного застосування автоматів у курсах «Прикладна теорія цифрових автоматів» та «Комп'ютерна логіка». Автомати розроблялись у середовищі побудови електронних схем Proteus.

КЕРУЮЧИЙ АВТОМАТ, ОПЕРАЦІЙНИЙ АВТОМАТ, МІКРОСХЕМА, МІКРОКОНТРОЛЕР, КОМБІНАЦІЙНА СХЕМА, СЕРЕДОВИЩЕ PROTEUS

Перелік умовних скорочень

КС – комбінаійна схема

КЛ – комбінаційна логіка

СЛ – секвенційна логіка

ОА – операційний автомат

У(К)А – управляючий (керуючий) автомат

«0» – логічний нуль, або ж низький рівень напруги

«1» – логічна одиниця, або ж високий рівень напруги

ГСА – граф-схема алгоритму

ДК – доповняльний код

Вступ

Поява перших монетоприймачів пов’язана з винаходом різного роду автоматів взагалі. Монетоприймачі бувають найрізноманітніші: механічні, електричні програмовані, що використовуються найчастіше, еталонні. Це класифікація цих пристроїв за функцією розпізнавання монети, що туди потрапила. Іншим чином монетоприймачі можна класифікувати за наявністю функції видачі решти. Тож розглянемо цю функцію більш докладно.

Прагнучи догодити зрозумілим потребам клієнтів різного роду торгових автоматів, їх власник звертає увагу на опціональні пристрої видачі решти. Цей пристрій, що зберігає і видає монети чи жетони називають хопером. Найперші такі пристрої використовувались в розмінних апаратах дрібних грошей при декількох автоматах продажу газованої води. Однак пізніше, коли торгові автомати почали продавати більш дорогі напої – такі як кава – виникла потреба встановлення окремого пристрою видачі решти на кожен апарат.

Хопери використовуються у всіх видах торгових і ігрових автоматів, а також автономних пристроях послуг. Вони розрізняються габаритними розмірами та універсальністю видачі монет. Спеціальні лічильники імпульсів, що вбудовані в хопери, дозволяють вести облік виданих монет.

Зазвичай хопери бувають програмовані, однак в рамках даної роботи було вирішено розробити автоматику пристрою дискретно, а не програмно, а саме з використанням мікросхем 74-ї серії.

Мікросхеми – це електронні схеми, що реалізовані у вигляді напівпровідникових кристалів. Вони виконують найрізноманітніші функції. З їх розвитком з’явилась можливість реалізації на одному кристалі не лише окремі вузли, а цілі електронно-обчислювальні пристрої. Подальший їх розвиток призвів до появи мікроконтролерів.

Мета кваліфікаційної роботи бакалавра

Метою даної роботи є розробка електронної схеми моделі пристрою, що видає решту, який складається з трьох частин:

- Схема керуючого автомата на основі автомата Мура.
- Схема операційного автомата.
- Демонстраційна частина схеми.

Розроблена схема зможе слугувати у якості демонстрації спільної роботи керуючого та операційного автоматів для курсових робіт з курсу «Прикладна теорія цифрових автоматів», а також для демонстрації віднімання чисел у ДК для лабораторних робіт з курсу «Комп’ютерна логіка». Саме ця демонстраційна універсальність і робить роботу актуальною.

Подальший розвиток схеми може передбачати деяке спрощення схеми за допомогою приєднання до мікроконтролера. Спрощення може полягати в введенні кількісних показників за допомогою мобільного додатку чи цифрової матриці, що буде частиною схеми.

1. Теоретична частина

Комбінаційна та секвенційна логіка

При побудові будь-яких логічних схем на рівні з булевою алгеброю, теорією скінчених автоматів слід згадати про комбінаційну та секвенційну логіку. Тож розглянемо ці типи логіки більш докладно.

Комбінаційна схема

Комбінаційна схема (КС) – це логічна схема, де застосовується двійкові змінні, а певна двійкова комбінація на виході відповідає кожному двійковому коду на вході в конкретний момент часу. В теорії цифрових пристроїв комбінаційною логікою (КЛ) називають логіку функціонування пристроїв комбінаційного типу.

До комбінаційних належать наступні схеми: дешифратори, суматори, демультиплексори, компаратори та інші схеми. Саме згадані вище схеми використовувались при розробці схем в роботі.

Стан комбінаційної схеми характеризується логічними функціями – складними виразами, що містять декілька простих, об'єднаних сполучниками:

$Y = f(x_1, x_2, \dots, x_n)$, де x – двійкова змінна.

Логічні функції представляються таблицями істинності.

Таблиця істинності — це представлення логічної функції у вигляді таблиці, в одній частині якої записують вхідні набори, а в другій – відповідні значення функції.

Склавши таблицю істинності можна отримати значення, при яких набори (кортежі) змінних будуть дорівнювати 1 і записати загальну функцію для виходу КС у формі ДКНФ чи ДДНФ. За загальною функцією в заданому базисі будується схема КС. [1][2]

Секвенційна логіка

Секвенійна логіка (СЛ) – її також називають “послідовною”, являється логікою пам’яті в цифрових пристроях. Саме термін “послідовна” використовується в основному при зв’язку з логічними автоматами.

Головною відмінністю цієї логіки від комбінаційної є моделювання цифрових пристроїв з урахуванням їх попередніх станів. Тобто тут вже з’являється поняття пам’яті. Пристрої можуть поділятися на синхронні та асинхронні в залежності від типу їх спрацювання.

При синхронній секвенційній логіці увага приділяється часу, що в такого типу схемах рахується тактовим імпульсом. Ці такти визначають моменти зміни станів автомата. Тобто функція стає синхронною до тактів визначення моментів змін стані. Моделі автоматів Мура та Мілі задаються саме апаратом синхронної логіки

Асинхронна логіка в даній роботі не використовувалась, однак слід сказати, що така логіка виражає запам’ятовування використовуючи моменти зміни станів виходячи з принципу «раніше-пізніше». Тобто не потребується зв’язок з віртуальним чи реальним часом. [3]

Теорія автоматів

Для побудови логічної схеми використовуються два типи автоматів, що тісно пов’язані між собою. е керуючий та операційний автомат. Їх взаємодія показана на Рисунку 1. [4]

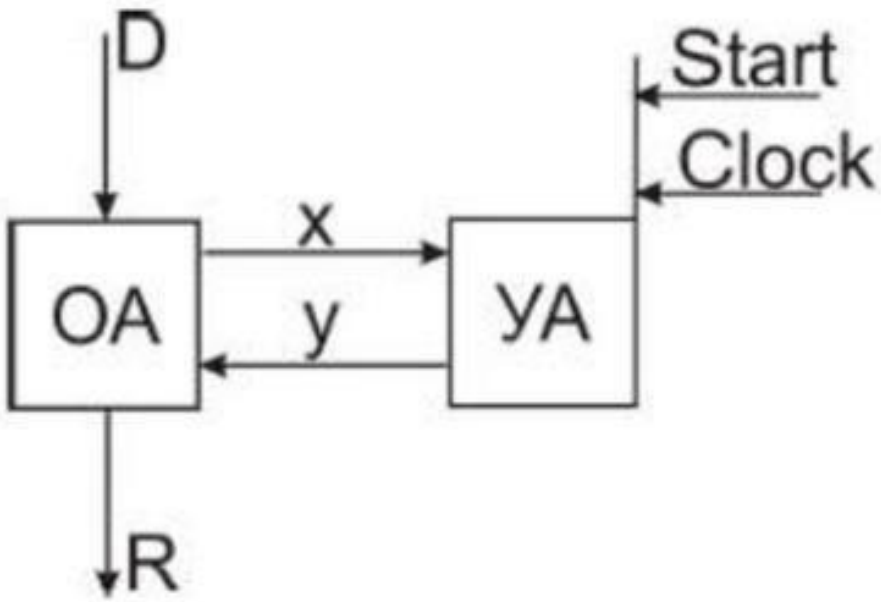


Рисунок 1. Схема з’вязку керуючого та операційного автомату

Зі схеми зрозуміло, що ОА отримує дані на вхід (D), обробляє їх відповідно до алгоритму свого функціонування та формує результат на виході (R). Він посилає сигнали умов (X) на УА, а той у свою чергу відсилає керуючі сигнали (Y), що дають вказівки, яким частинам ОА слід спрацювати у той чи інший момент часу. Докладніше розглянемо поняття «Операційний автомат» та «Керуючий автомат».

1.2.1. Операційні та керуючі автомати

Операційний автомат являє собою пристрій цифрової електронної обчислювальної машини, що перетворює коди чисел або слова.

Він складається із набору регістрів з певною комбінаційною логікою у на входах тригерів регістрів. На вхід ОА приходять вихідні сигнали УА, вони ж сигнали мікрооперацій. Ці сигнали визначають перетворення множини станів ОА. Вихідними сигналами ОА є рядки значень логічних умов, які характеризують стани його регістрів.

Керуючі (управляючі) автомати виконують керуючі функції над даними. На вхід він отримує набори запитів від ОА і відповідно реагує на результати перевірки цих запитів. Далі він формує набір керуючих сигналів, що спрямовані до ОА. Це відбувається на кожному кроці очікування

Мікросхеми 74-ї серії

При збірці моделі в програмному забезпеченні Proteus використовувався широкий спектр різноманітних мікросхем 74-ї серії. Це мікросхеми на ТТЛ-логіці, що є першим широко розповсюдженим сімейством інтегральних мікросхем з ТТЛ-логікою. [5][6]

74 серія має сотні пристроїв, що забезпечують як базові логічні операції, тригери та лічильники, так і шинні формувачі, передатчики сигналів та ін. Нині ці мікросхеми використовуються в споживчій електроніці, а також у якості узгоджувальної логіки в комп'ютерах та промисловій техніці. Розглянемо мікросхеми, що були використані при побудові схем ОА та УА.

Булеві функції NOT, OR, AND, XOR

Мікросхема 7432 містить чотири окремі логічні елементи OR з двома входами на кожному. Всі елементи працюють незалежно. При подачі «1» на один або обидва входи кожного елемента мікросхеми 7432 на виході встановлюється «1». Якщо два входи подається «0», то на виході формується «0».

Мікросхема 7408 містить чотири окремі логічні елементи AND з двома входами на кожному. Всі елементи працюють незалежно. При подачі «0» на один або обидва входи кожного елемента на виході 7408 мікросхеми встановлюється «0». Якщо два входи подається «1», то на виході формується «1».

Мікросхема 7486 містить чотири окремі логічні елементи XOR з двома входами на кожному. Всі елементи працюють незалежно. На виході кожного елемента мікросхеми 7486 формується «1», якщо на один із двох входів (але не на обидва) подається така напруга. Якщо два входи подається «1» чи «0», то виході формується «0».

Мікросхема 7404 містить шість окремих інверторів При подачі «0» на вхід кожного з них на виході мікросхеми 7404 встановлюється «1» та навпаки.

Суматор

Мікросхема 7483 містить повний суматор, який підсумовує два 4-розрядні двійкові числа з урахуванням переносу. Маємо на увазі, що молодший розряд позначений «1», старший – «4».

Перший операнд подається на входи мікросхеми 7483 A1-A4. Другий операнд подається на входи B1-B4. Сума обох чисел формується на виходах ∑1 - ∑4. Коли результат підсумовування перевищить у двійковій системі 1111, на виході сигналу перенесення C4 з'являється «1».

Вхід сигналу перенесення C0 мікросхеми 7483 має бути заземлений на корпус, якщо використовуються лише 4-розрядні числа. Якщо використовується 8-розрядне число (старші розряди), то вхід C0 мікросхеми 7483 з'єднується з виходом C4 попереднього ступеня (молодші розряди).

Мікросхеми пам’яті

Мікросхема 7474 містить два окремі D-тригери, які запускаються позитивним фронтом тактового імпульсу і мають роздільні входи установки та скидання. Обидва тригери можна використовувати незалежно один від одного.

Інформація, що надходить на вхід D, передається на вихід Q (і в інверсному вигляді на вихід Q), коли напруга на вході тактових імпульсів змінюється з низького рівня на високий. Без позитивного фронту тактового імпульсу на синхронізуючому вході неможливе ніяке послідовне перенесення інформації з входу D на вихід Q.

Дані, що надходять на вхід D мікросхеми 7474, можуть змінюватися будь-якої миті. Інформація вважається значимою лише тоді, коли тактовий імпульс переходить із низького рівня напруги на високий. У цей час вона передається на тригер.

У нормальному режимі роботи на входи Preset та Reset подається напруга високого рівня. Якщо на вхід Reset мікросхеми 7474 подається напруга низького рівня, то на виході Q формується напруга низького рівня. Якщо на вхід Preset мікросхеми 7474 подається напруга низького рівня, то на виході Q формується напруга високого рівня. Не слід подавати напругу низького рівня одночасно на обидва ці входи, оскільки в цьому випадку виходи будуть перебувати в нестійкому стані, який зміниться, коли зміниться стан входів Preset або Reset.

Мікросхема 74198 містить реверсивний 8-розрядний регістр зсуву даних з паралельним та послідовним введенням-виведенням інформації.

Якщо на вхід скидання Clear подається «0», то на виходах Q0 - Q7 встановлюється «0» незалежно від логічного стану решти всіх входів. Якщо на вхід скидання Clear подається «1», режим роботи визначається станами входів S0 і S1 (режим управління).

Зсув даних мікросхеми вліво відбувається, коли на вхід S0 подається «0», а на вхід S1 «1», при цьому дані послідовно надходять на вхід DSL.

Зсув даних мікросхеми вправо відбувається, коли на вхід S0 подається «1», а на вхід S1 «0», при цьому дані послідовно надходять на вхід DSR.

Якщо на обидва входи S0 і S1 мікросхеми подається «1», то можливе послідовне завантаження даних з входів P0 - P7. Під час паралельного введення даних послідовні входи замкнені.

Послідовно та паралельно введені дані надходять у регістр синхронно по позитивному фронту на вході Clock (вхід тактових імпульсів). При цьому дані повинні знаходитись на інформаційних входах до початку формування позитивного фронту тактового імпульсу.

Компаратор

Наступними будуть описані мікросхеми, що так чи інакше пов’язані з певним вибором.

Мікросхема 7485 порівнює два 4-розрядні слова і визначає, рівні вони чи ні. Обидва порівнювані слова А надходять на відповідні входи мікросхеми 7485.

Якщо необхідно порівняти лише 4-розрядні слова, то на вхід перенесення мікросхеми 7485 A = B «1», а на входи перенесення A>B та A<B – «0». Якщо обидва слова дорівнюють за величиною, на виході A = B формується «1». Якщо слово А більше слова, то на виході A> B формується «1». Якщо слово А менше за слово, на виході A<B встановлюється «1». На решті виходів формується «0».

Коли мікросхема 7485 порівнює 8-розрядні слова, виходи першого ступеня 4-розрядного компаратора з'єднуються з перенесеннями другого ступеня. У цьому випадку результат порівняння одержують на виходах 4-розрядного компаратора.

Дешифратор/демультиплексор

Мікросхема 74154 перетворює 4-розрядний двійковий код в сигнал низького рівня на одному з 16 виходів. Коли 4-розрядний двійковий код надходить на адресні входи мікросхеми 74154, то відповідному виході встановлюється напруга низького рівня, а на інших – високого.

Однак це відбувається лише в тому випадку, якщо на обидва роздільні входи мікросхеми E1 і E2 подається напруга низького рівня. При такому підключенні схема працюватиме як дешифратор.

Якщо на один з роздільних входів мікросхеми 74154 подається напруга низького рівня, а інший розглядається як інформаційний вхід, то вибраний через адресні входи вихід матиме той же логічний рівень, що і другий вхід. Таким чином, мікросхема 74154 використовується як демультиплексор або багатоканальний комутатор даних.

Селектор

Мікросхема 74157 містить чотири селектори даних 2-в-1.

За допомогою мікросхеми 74157 можна з даних, що надходять на чотири пари входів 1A/1B - 4A/4B, вибрати необхідну інформацію та передати її на один із відповідних виходів 1Q - 4Q.

Роздільний вхід Enable дозволяє відключити вихід незалежно від стану входу вибірки даних Select. Якщо на Enable подається «1», то на всіх виходах встановлюється «0» незалежно від стану інших входів. Якщо цей вхід подається «0», стан виходів залежить від стану входу вибірки Select.

Якщо на вхід вибірки подається «0». то виходи приймають той самий рівень напруги, який надходить на входи А. Якщо на вхід вибірки подається «1», то виходи приймають той самий логічний рівень, який надходить на входи В.[7]

Лічильники

Мікросхема 74163 містить програмований синхронний 4-розрядний двійковий лічильник, який рахує імпульси в двійковому коді у прямому напрямку та має синхронний вхід скидання показань.

При кожному перепаді напруги тактового імпульсу з низького рівня на високий показник лічильника 74163 синхронно збільшуються на одиницю. Отже, схема включається позитивним фронтом тактового імпульсу. Логічний рівень виходів QO, Q1, Q2 та Q3 визначається двійковим кодом.

Для скидання показань лічильника 74163 на вхід Mr подається напруга низького рівня. Скидання показань відбувається при наступному фронті тактового імпульсу, тим часом на усіх виходах встановлюється напруга низького рівня. Якщо на вхід завантаження Load подається напруга низького рівня, то при наступному позитивному фронті тактового імпульсу код, що знаходиться на входах PO - P3, завантажується в лічильник.

Для синхронного рахунку чисел з декількома розрядами без застосування додаткових логічних елементів використовують входи PE і TE мікросхеми 74163 в якості дозволяючих, а також вихід CO (вихід сигналу перенесення).

Мікросхема 74176 містить дільники на два та на п'ять, а також входи попередньої установки та вхід скидання.

Оскільки мікросхема 74176 складається з двох окремих дільників (на два і на п'ять) з роздільними входами тактових імпульсів, вона дозволяє працювати в наступних режимах:

- дільники на 2 та на 5 (не був використаний)
- дільник на 10 (не був використаний)
- десятковий лічильник: вихід QA з'єднаний із входом тактових імпульсів 2Clock. Тактова частота подається на вход 1Clock. Лічильник працює у двійково-десятковому коді

Лічильник 74176 послідовно включається при перепаді напруги тактового імпульсу з високого рівня на низький (негативний фронт). Скидання та встановлення лічильника 74176 відбуваються асинхронно, тобто незалежно від тактових імпульсів.

У нормальному режимі роботи на вхід скидання Clear подається напруга високого рівня. Якщо на цей вхід подається короткочасний імпульс напруги низького рівня, то на всіх виходах також формується така напруга.

Через інформаційні входи A - D мікросхеми 74176 здійснюють попереднє встановлення лічильника, подаючи на ці входи необхідний код, а на вхід Load - короткочасний імпульс напруги низького рівня.

Демонстраційна частина

Мікросхема 4511 являє собою регістр перетворювач двійкового коду в семисегментний драйвер.

Сегме́нтний індика́тор [8] — індикатор, елементи відображення якого є сегментами, що згруповані в одне або кілька знакомісць.

Сегментом є елемент відображення інформації індикатора, контур якого являє собою прямі та (або) криві лінії. На відміну від матричного індикатора, в якому всі елементи зображення однакові за формою, в сегментному індикаторі кожен сегмент унікальний. Форма і положення сегментів на індикаторі розробляється спеціально для передачі певного набору символів або знаків. Символи на таких індикаторах формуються сукупністю кількох сегментів. Основна відмінність сегментного індикатора від матричного — це порівняно невелика кількість елементів індикації і відповідно спрощена схема управління.

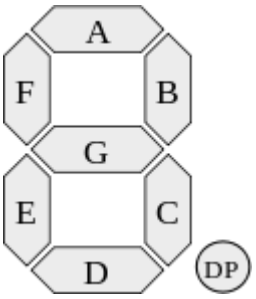


Рисунок 2. Індивідуальні сегменти семи-сегментного дисплея

2. Практична частина

Вибір середовища розробки

Існує багато програм для проектування електронних схем, однак було вибрано саме середовище Proteus. Демо-версія є абсолютно безкоштовною, однак без функції збереження побудованих схем. Вибрано саме пакет програм Proteus, оскільки ця програма використовувалась мною в курсах «Комп'ютерна схемотехніка», «Прикладна теорія цифрових автоматів» та «Комп'ютерна логіка», то ж вибір робився, покладаючись на майже дворічний досвід роботи в цьому середовищі проектування. Proteus має набір бібліотек, елементи яких повністю задовольняють потреби побудови схем в роботі.

Proteus 8 Professional

Пакет є системою моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice.

PSpice (Personal Simulation Program with Integrated Circuit Emphasis) — програма симуляції аналогової та цифрової логіки, описаної мовою SPICE, яка призначена для персональних комп'ютерів. Розроблена компанією MicroSim та використовується в автоматизації проектування електронних приладів.

Відмінною рисою пакету PROTEUS VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів та ін. Причому в Proteus повністю реалізована концепція наскрізного проектування, коли інженер змінює щось у логіці роботи схемотехніки і програмний пакет тут же «підхоплює» дані зміни в системі трасування. Бібліотека компонентів містить довідкові дані. Пакет Proteus складається з двох частин, двох підпрограм: ISIS – програма синтезу та моделювання безпосередньо електронних схем та ARES –

програма розробки друкованих плат. Разом із програмою встановлюється набір демонстраційних проектів для ознайомлення.

Пакет є комерційним. Безкоштовна ознайомча версія характеризується повною функціональністю, але не має можливості збереження файлів.

Примітною особливістю є те, що ARES можна побачити 3D-модель друкованої плати, що дозволяє розробнику оцінити свій пристрій ще на стадії розробки. [10]

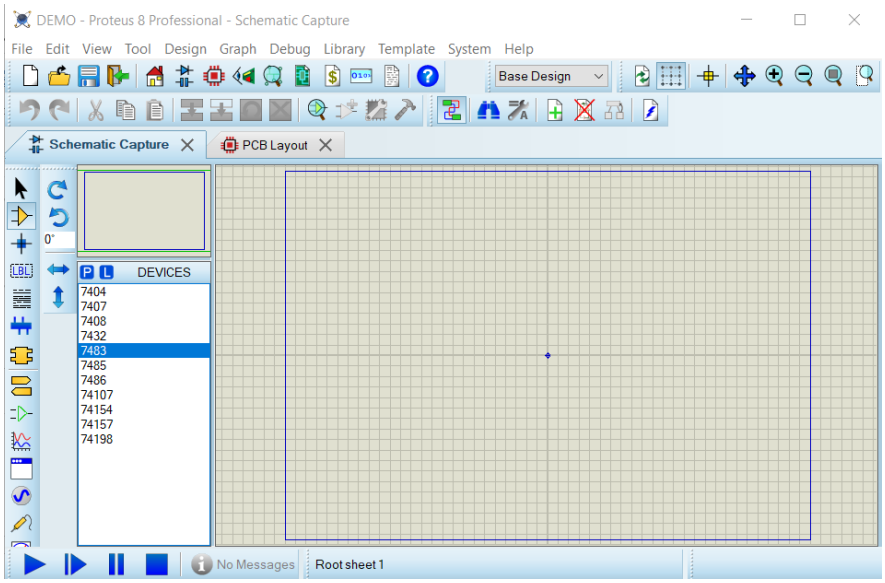


Рисунок 3. Система автоматизованого проєктування Proteus

Mermaid-js

Для побудови граф-схеми переходів керуючого автомата використовувались діаграми mermaid. [11]

У GitHub з'явилася можливість додавати до md-файлів динамічні діаграми за допомогою генератора Mermaid. До цього діаграми вставлялися як зображень чи «малювалися» з допомогою символів з ASCII-таблицы. Тепер повноцінну підтримку схем додали в синтаксис розмітки Markdown.

Mermaid – інструмент для побудови діаграм та графіків, що базується на JavaScript. З його допомогою можна динамічно створювати блок-схеми, UML-

діаграми, графіки комітів та діаграми Ганта. Команда GitHub об'єдналася з розробниками CommonMark і додала нативну підтримку синтаксису Mermaid на платформу.

Щоразу, коли в md-файлі зустрічається блок коду, відзначений як mermaid, система створює новий кадр iframe, бере необроблений код з блоку, передає його в Mermaid.js і перетворює код на діаграму. Все це відбувається локально у браузері користувача.

Однак на практиці через великий об’єм коду побудова блок-схеми виявилася недоречною. Тим не менш результати роботи продемонстровані нижче, а код блок-схеми знаходиться в ДОДАТКУ А.

«Розумний» алгоритм

Простий алгоритм видачі монет не враховує кількість монет. Тобто в параметрах ми маємо лише ту суму, що має видати пристрій. Це ідеальні умови, при яких кількості монет завжди вистачає, щоб покрити ту, чи іншу суму. Однак, в цій роботі така особливість як кількість монет врахована.

Для початку слід зауважити, що описаний вище алгоритм - лінійний, тобто постійно порівнює номінал монети і решту. Однак, якщо ми маємо обмежену кількість монет, то цей алгоритм не завжди працюватиме правильно.

Візьмемо для прикладу наступну ситуацію в автоматі в наявності нема одиниць, але є необмежена кількість п’ятірок та двійок. потрібно видати «8». В старому варіанті, автомат видав би спочатку «5», потім «2» і завершив роботу. Однак «8» можна видати чотирма двійками. Цей варіант повинен передбачатися.

Може здатися, що таких варіантів безліч, однак для випадку розряду одиниць випадків, коли лінійний алгоритм не впорається, а потрібно передбачити видачу іншим способом усього лише 4.

Варіанти можливих видач коректних сум наведені в Таблиці 1.

Зеленим позначені видачі монет лінійним способом, при чому на виході буде та сума, яку захоче користувач.

Помаранчевим та жовтим позначені видачі монет, що слід передбачити, оскільки лінійний метод буде видавати неповну суму.

Червоним позначені видачі монет, при яких сума буде неповна, оскільки з наявними монетами покрити її неможливо.

В наявності	9	8	7	6	5	4	3	2	1
5 2 1	5 2 2	5 2 1	5 2	5 1	5	2 2	2 1	2	1
5 2	5 2 2	2 2 2 2	5 2	2 2 2	5	2 2	5 2 2 2 2	2	5 2 2 2
5 1	5 1 1 1 1	5 1 1 1	5 1 1	5 1	5	1 1 1 1	1 1 1	1 1	1
2 1	2 2 2 2 1	2 2 2 2	2 2 2 1	2 2 2 2	2 2 1	2 2	2 1	2	1
5	5	5	5	5	5	0	0	0	0
2	2 2 2 2	2 2 2 2	2 2 2	2 2 2	2 2	2 2	2	2	0
1	1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1	1 1 1 1 1 1	1 1 1 1 1	1 1 1 1	1 1 1	1 1	1

Таблиця 1 «Розумний» алгоритм

Розробка керуючого автомата Мура

Необхідно побудувати керуючий автомат Мура для реалізації наступного алгоритму:

Крок 1 Ввести кількість всіх монет (10-ки, 5-ки, 2-ки, 1-ці)

Крок 2 Ввести значення решти, присвоїти це значення стартовому буферу

Крок 3 Перевірити наявність монет «10»

Крок 3.1 Якщо вони наявні, призначити змінній монет значення «10»

Крок 3.2 Якщо ні, перейти до Кроку 9

Крок 4 Перевірити, чи значення монети менше чи дорівнює решті та перевірити наявність монет «10»

Крок 4.1 Якщо так, перевірити відсутність «1» та чи дорівнює решта «13» або «11»

Крок 4.1.1 Якщо так, то перейти до Кроку 9

Крок 4.1.2 Якщо ні, то Кроки 5 – 8

Крок 4.2 Якщо ні, Крок 9

Крок 5 Відняти від решти значення номіналу

Крок 6 Подати сигнал «TEN»

Крок 7 Додати до фінішного буферу значення номіналу

Крок 8 Декрементувати кількість десяток та перейти до Кроку 4

Крок 9 Перевірити наявність монет «5»

Крок 9.1 Якщо вони наявні, призначити змінній монет значення «5»

Крок 9.2 Якщо ні, перейти до Кроку 15

Крок 10 Перевірити, чи значення монети менше чи дорівнює решті та перевірити наявність монет «5»

Крок 10.1 Якщо так, перевірити відсутність «1» та чи дорівнює решта «8» або «6»

Крок 10.1.1 Якщо так, то перейти до Кроку 15

Крок 10.1.2 Якщо ні, то Кроки 11 – 14

Крок 10.2 Якщо ні, Крок 15

Крок 11 Дії Кроку 5

Крок 12 Подати сигнал «FIVE»

Крок 13 Дії Кроку 7

Крок 14 Декрементувати кількість п’ятірок та перейти до Кроку 10

Крок 15 Перевірити наявність монет «2»

Крок 15.1 Якщо вони наявні, призначити змінній монет значення «2»

Крок 15.2 Якщо ні, перейти до Кроку 21

Крок 16 Перевірити, чи значення монети менше чи дорівнює решті та перевірити наявність монет «2»

Крок 16.1 Якщо так, то Кроки 17-20

Крок 10.2 Якщо ні, Крок 21

Крок 17 Дії Кроку 5

Крок 18 Подати сигнал «TWO»

Крок 19 Дії Кроку 7

Крок 20 Декрементувати кількість двійок та перейти до Кроку 16

Крок 21 Перевірити наявність монет «1»

Крок 21.1 Якщо вони наявні, призначити змінній монет значення «1»

Крок 21.2 Якщо ні, перейти до Кроку 27

Крок 22 Перевірити, чи значення монети менше чи дорівнює решті та перевірити наявність монет «1»

Крок 22.1 Якщо так, то Кроки 23-26

Крок 22.2 Якщо ні, Крок 27

Крок 23 Дії Кроку 5

Крок 24 Подати сигнал «ONE»

Крок 25 Дії Кроку 7

Крок 26 Декрементувати кількість одиниць та перейти до Кроку 22

Крок 27 Перевірити чи значення стартового буфера дорівнює фінішному буферу

Крок 27.1 Якщо так, запалити зелений світлодіод

Крок 27.2 Якщо ні, запалити червоний світлодіод

Крок 28.1 Якщо йдемо на новий круг, то перейти до Кроку 2

Крок 28.2 Якщо ні, то закінчуємо програму

Програмна демонстрація алгоритму, що написана мовою C#, наведена у ДОДАТКУ Б.

Побудова ГСА

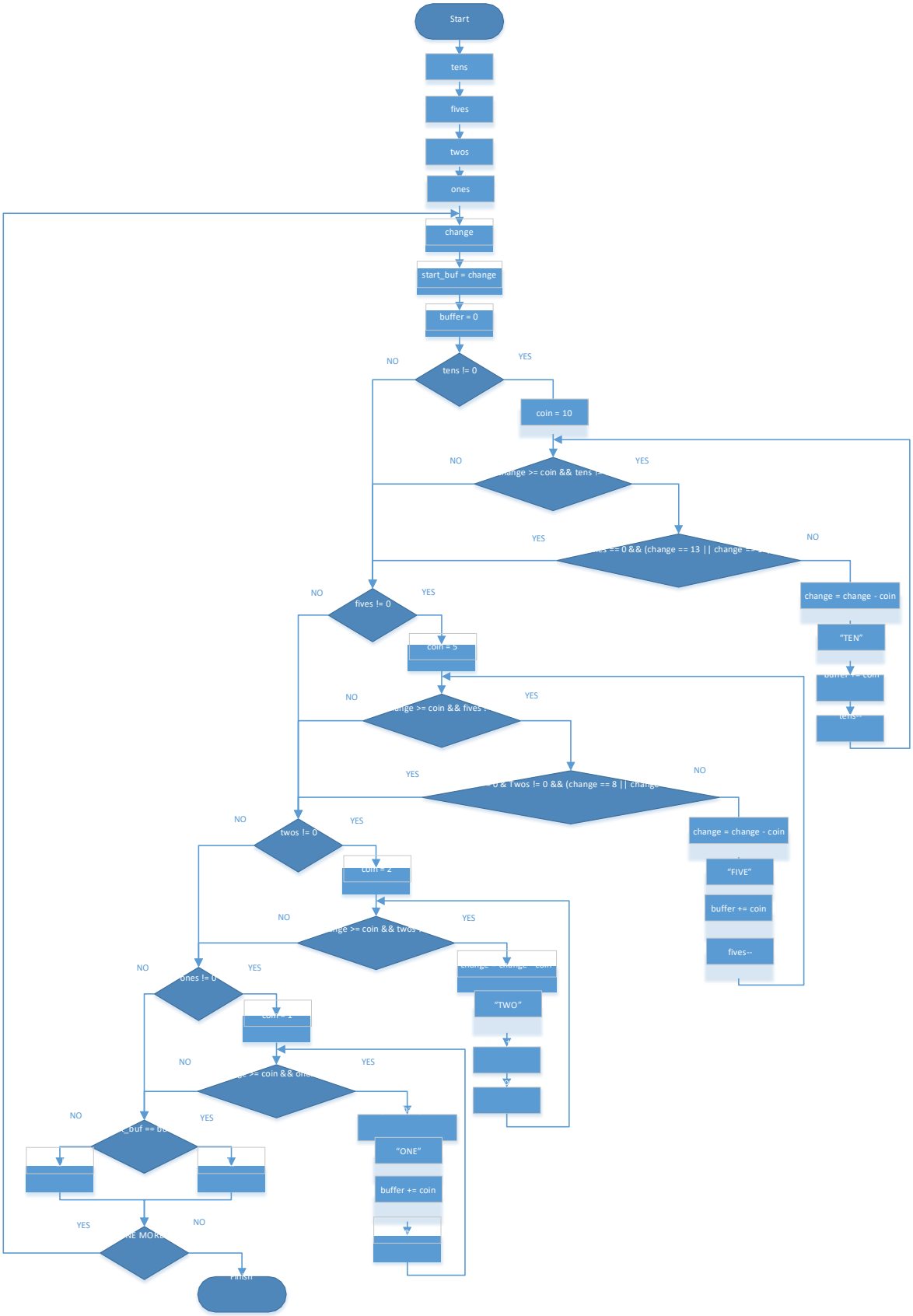


Рисунок 4. Блок-схема алгоритму

Побудова змістовної ГСА

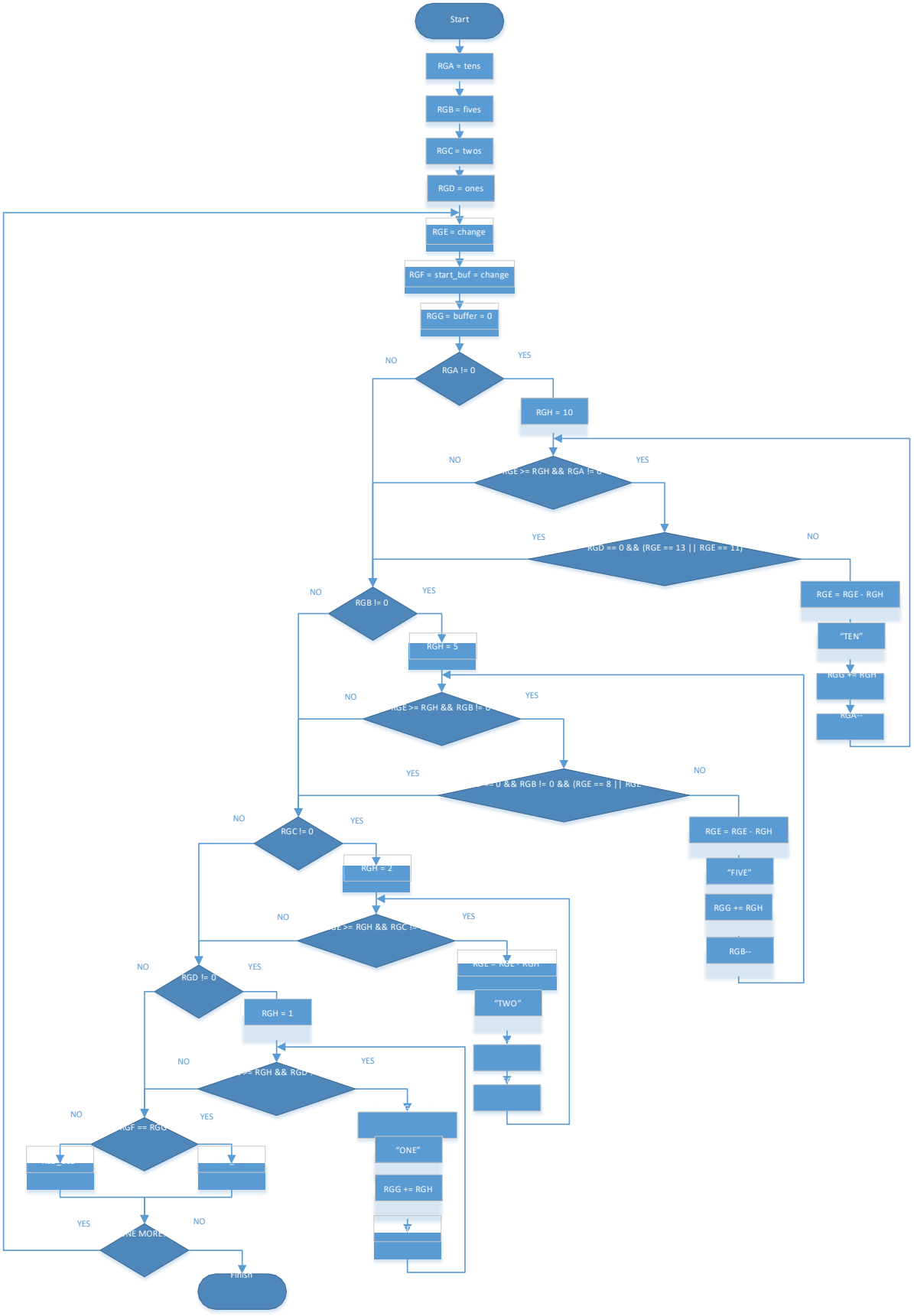


Рисунок 5. Змістовна схема алгоритму

Побудова блок-схеми закодованого алгоритму

Кодуємо умовні та операційні вершини блок-схеми. Якщо операції повторюються, то вони кодуються однаково. В цьому випадку віднімання відбувається 4 рази. Тому цю вершину ми кодуватимемо одним і тим самим кодом. Це ж саме стосується випадку, коли до буферу додається номінал монети. Таблиця кодування зображена в Таблиці 2.

Код	**Зміст**	**Примітки**
mY1	RGA = tens	Ввід кількості десяток і запис у RGA
mY2	RGB = fives	Ввід кількості п’ятірок і запис у RGA
mY3	RGC = twos	Ввід кількості двійок і запис у RGA
mY4	RGD = ones	Ввід кількості одиниць і запис у RGA
mY5	RGE = change	Ввід решти і запис у RGA
mY6	RGF = start_buf = change	В RGF записуємо стартовий буфер, якому надаємо значення решти
mY7	RGG = buffer = 0	В RGF записуємо значення фінального буферу, поки він дорівнює нулю
mY8	RGH = 10	Запис номіналу монети 10 у RGH
mY9	RGE = RGE – RGH	Віднімання від решти номіналу монети і запис результату в регістр решти
mY10	“TEN”	Сигнал про видачу монети 10
mY11	RGG += RGH	Додаємо до фінального буферу номінал
mY12	RGA--	Декрементуємо кількість «10»
mY13	RGH = 5	Запис номіналу монети 5 у RGB
mY14	“FIVE”	Сигнал про видачу монети 5
mY15	RGB--	Декрементуємо кількість «5»
mY16	RGH = 2	Запис номіналу монети 2 у RGB

mY17	“TWO”	Сигнал про видачу монети 2
mY18	RGC--	Декрементуємо кількість «2»
mY19	RGH = 1	Запис номіналу монети 1 у RGB
mY20	“ONE”	Сигнал про видачу монети 1
mY21	RGD--	Декрементуємо кількість «1»
mY22	GRN_LED	Запалюємо зелений світлодіод
mY23	RED_LED	Запалюємо червоний світлодіод
X1	RGA != 0	В наявності є «10»
X2	RGE >= RGH	Решта більша за номінал
X3	RGD == 0	В наявності нема «1»
X4	RGE == 13	Решта = 13
X5	RGE == 11	Решта = 11
X6	RGB != 0	В наявності є «5»
X7	RGE == 8	Решта = 8
X8	RGE == 6	Решта = 6
X9	RGC != 0	В наявності є «2»
X10	RGF == RGG	Стартовий буфер = фінальному буферу
X11	ONE MORE?	Чи йдемо на нове коло?

Таблиця 2. Таблиця кодування вершин

Закодована схема (Рисунок 6) будується на основі змістовної схеми (Рисунок 5) та таблиці кодувань (Таблиця 2). Блоки схеми замінюються відповідними блоками таблиці.

Об’єднання мікрооперацій в одну можливо лише за умови, коли мікрооперації є незалежними. Тобто результати виконання однієї мікрооперації не залежить від іншої. [12]

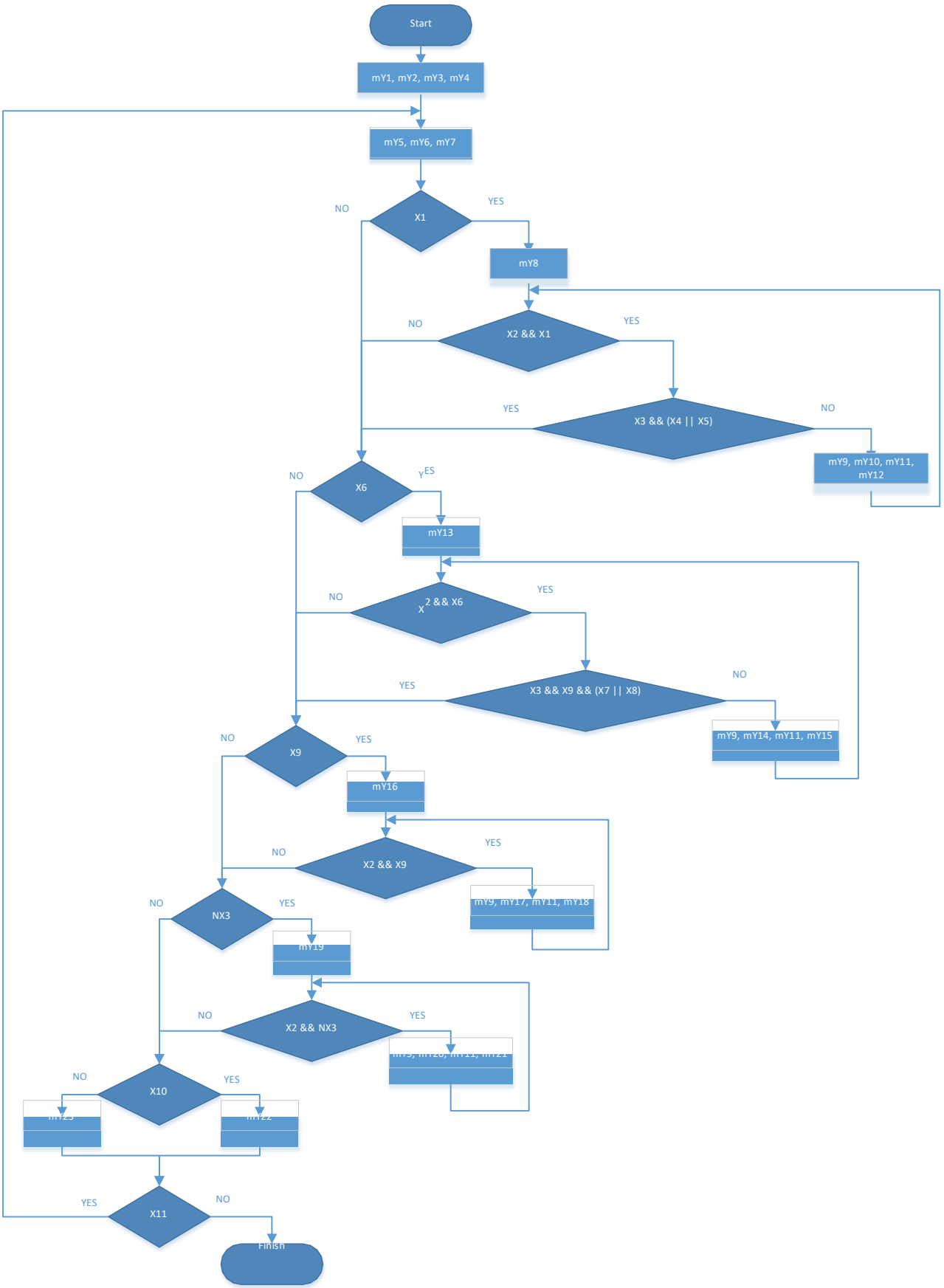


Рисунок 6. Блок-схема закодованого алгоритму

Побудова відміченої ГСА

Для автомата Мура на етапі побудови відміченої ГСА розмітка проводиться відповідно до наступних правил:

1. Символом s0 відзначається початкова так кінцева вершини (у цьому випадку початок і кінець позначені по-різному, щоб автомат не був зациклением).
2. Різні операційні вершини позначаємо різними символами.
3. Потрібно відзначити абсолютно всі операційні вершини.

Відповідно до правил, побудуємо відмічену ГСА для автомата, при цьому потрібно замінити всі мікрооперації відповідно на керуючі сигнали. [12]

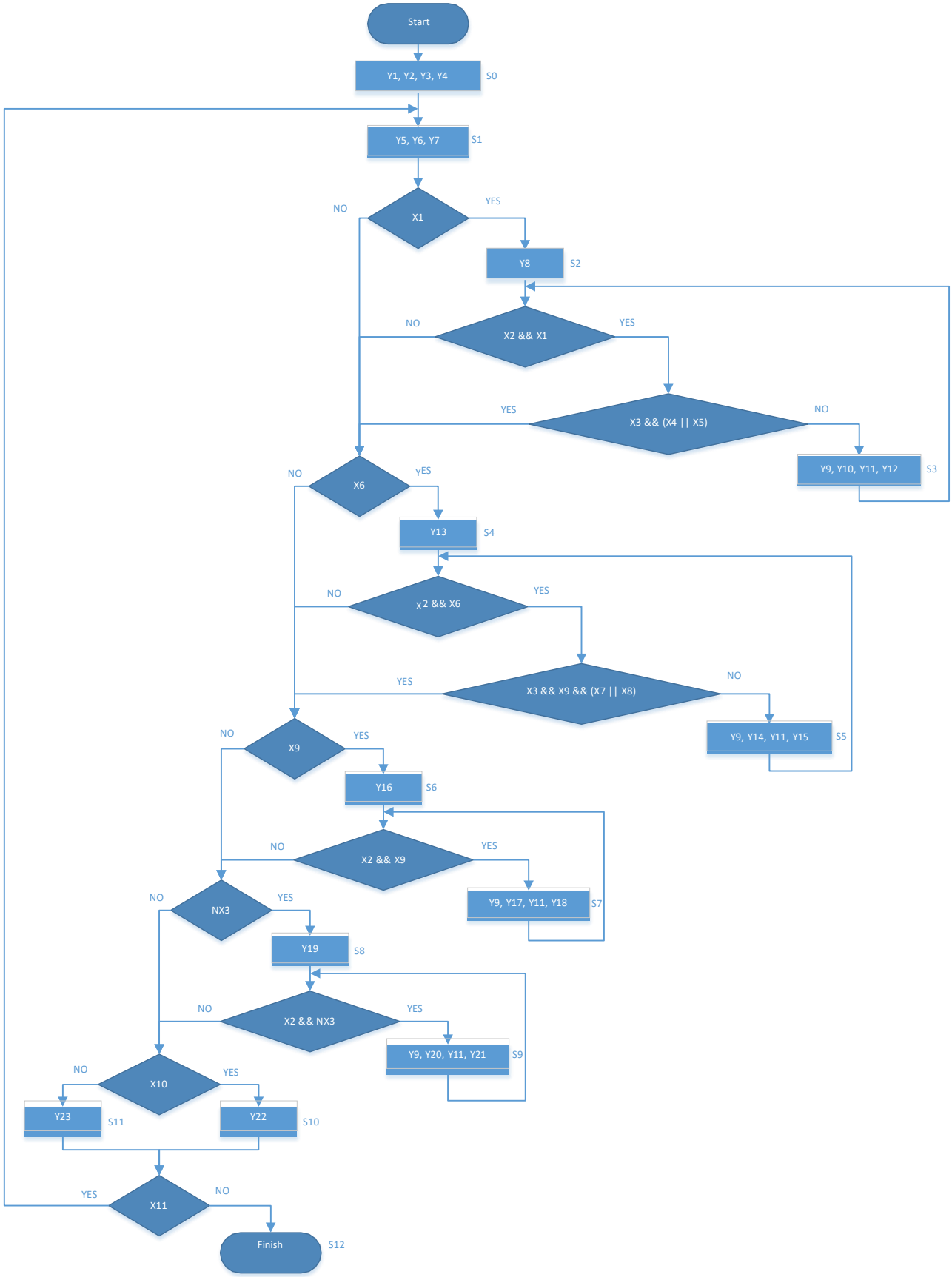


Рисунок 7. Відмічена граф-схема автомата

Побудова граф-схеми переходів

Граф-схема переходів (Рисунок 8) будується на основі відміченої граф-схеми автомата (Рисунок 7). Вершини графа позначають можливі стани автомата, а стрілки, у свою чергу, - на переходи. Над стрілкою вказується умова переходу, вона ж вхідний сигнал. А біля станів відмічаються вихідні сигнали. [12]

Схема побудована за допомогою діаграм mermaid. Код та сама схема (для зручності перегляду) наведені нижче в ДОДАТКУ А.

Побудова структурної таблиці переходів-виходів автомата Мура

Будуємо таблицю переходів-виходів. Для мікропрограмних автоматів така таблиця будується у вигляді списку. Буває пряма та зворотна таблиця. Для даного автомата була побудована пряма таблиця. Вона представлена в Таблиці 4.

У автоматі кількість станів M = 13, тому число елементів пам’яті

m = $]log_2 M[=]log_2 13[= 4$

Складність комбінаційної схеми залежить від обраного кодування станів автомата. Застосуємо метод кодування для D-Тригерів.

sm	Nm	Code
s10	9	0000
s11	9	0001
s8	7	0010
s6	5	0100
s1	3	1000
s4	3	0110
s0	2	1100
s3	2	0011
s5	2	1010
s7	2	0101
s9	2	1001
s2	1	1110
s12	2	1111

Таблиця 3. Таблиця кодування станів

Будуємо структурну таблицю переходів-виходів автомата (Таблиця 4). У стовпцях Ksm і Ksk вказуються коди вихідного стану та стану переходу. [12] У стовпці ФЗ, тобто функцій збудження вказуються, який із D тригерів повинен спрацювати. Нагадаємо, що D тригер при подачі сигналу на вхід на виході подає сигнал із входу.

В якості комірок пам’яті КА взято D-тригери. На схемі, це мікросхема 7474

sm	Ksm	sk	Ksk	Y	x	ФЗ
s10	0000	s12	1100	Y1Y2Y3Y4	nx11	D1D2D3D4
s11	0001				nx11	
s0	1100	s1	1000	Y5Y6Y7		D1
s10	0000				x11	
s11	0001				x11	
s1	1000	s2	1110	Y8	x1	D1D2D3
s2	1110	s3	0011	Y9Y10Y11Y12	x2x1n(x3(x4+x5))	D3D4
s3	0011				x2x1n(x3(x4+x5))	
s1	1000	s4	0110	Y13	nx1x6	D2D3
s2	1110				(n(x2x1)+(x2x1x3(x4+x5))x6	
s3	0011				(n(x2x1)+(x2x1x3(x4+x5))x6	
s4	0110	s5	1010	Y9Y14Y11Y15	x2x6n(x3x9(x7+x8))	D1D3
s5	1010				x2x6n(x3x9(x7+x8))	
s1	1000	s6	0100	Y16	nx1nx6x9	D2
s2	1110				(n(x2x1)+(x2x1x3(x4+x5))nx6x9	
s3	0011				(n(x2x1)+(x2x1x3(x4+x5))nx6x9	
s4	0110				(n(x2x6)+x2x6x3x9(x7+x8))x9	
s5	1010				(n(x2x6)+x2x6x3x9(x7+x8))x9	
s6	0100	s7	0101	Y9Y17Y11Y18	x2x9	D2D4
s7	0101				x2x9	
s1	1000	s8	0010	Y19	nx1nx6nx9nx3	D3
s2	1110				(n(x2x1)+(x2x1x3(x4+x5))nx6nx9nx3	
s3	0011				(n(x2x1)+(x2x1x3(x4+x5))nx6nx9nx3	
s4	0110				(n(x2x6)+x2x6x3x9(x7+x8))nx9nx3	
s5	1010				(n(x2x6)+x2x6x3x9(x7+x8))nx9nx3	
s6	0100				n(x2x9)nx3	
s7	0101				n(x2x9)nx3	
s8	0010	s9	1001	Y9Y20Y11Y21	x2nx3	D1D4
s9	1001				x2nx3	
s1	1000	s10	0000	Y22	nx1nx6nx9x3nx10	
s2	1110				(n(x2x1)+(x2x1x3(x4+x5))nx6nx9x3nx10	
s3	0011				(n(x2x1)+(x2x1x3(x4+x5))nx6nx9x3nx10	
s4	0110				(n(x2x6)+x2x6x3x9(x7+x8))nx9x3nx10	
s5	1010				(n(x2x6)+x2x6x3x9(x7+x8))nx9x3nx10	
s6	0100				n(x2x9)x3nx10	
s7	0101				n(x2x9)x3nx10	
s8	0010				n(x2nx3)nx10	
s9	1001				n(x2nx3)nx10	
s1	1000	s11	0001	Y23	nx1nx6nx9x3x10	D4
s2	1110				(n(x2x1)+(x2x1x3(x4+x5))nx6nx9x3x10	
s3	0011				(n(x2x1)+(x2x1x3(x4+x5))nx6nx9x3x10	
s4	0110				(n(x2x6)+x2x6x3x9(x7+x8))nx9x3x10	
s5	1010				(n(x2x6)+x2x6x3x9(x7+x8))nx9x3x10	
s6	0100				n(x2x9)x3x10	
s7	0101				n(x2x9)x3x10	
s8	0010				n(x2nx3)x10	
s9	1001				n(x2nx3)x10	

Таблиця 4. Структурна таблиця переходів-виходів автомата Мура

Створення системи рівнянь переходів та системи рівнянь виходів

Рівняння переходів

D1 = (s10+s11)*nx11 + s0 + (s10+s11)*x11 + s1x1 + (s4+s5)*(x2x6n(x3x9(x7+x8))) + (s8+s9)*(x2nx3)

D2 = (s10+s11)*nx11 + s1x1 + s1*(nx1x6) + (s2+s3)*((n(x2x1)+(x2x1x3(x4+x5))x6) + s1*(nx1nx6x9) + (s2+s3)*((n(x2x1)+(x2x1x3(x4+x5))nx6x9) + (s4+s5)*((n(x2x6)+x2x6x3x9(x7+x8))x9) + (s6+s7)*(x2x9)

D3 = s1x1 + (s2+s3)*(x2x1n(x3(x4+x5))) + s1*(nx1x6) + (s2+s3)*((n(x2x1)+(x2x1x3(x4+x5))x6) + (s4+s5)*(x2x6n(x3x9(x7+x8))) + s1*(nx1nx6nx9nx3) + (s2+s3)*((n(x2x1)+(x2x1x3(x4+x5))nx6nx9nx3) + (s4+s5)*((n(x2x6)+x2x6x3x9(x7+x8))nx9nx3) + (s6+s7)*(n(x2x9)nx3) + (s10+s11)*nx11

D4 = (s2+s3)*(x2x1n(x3(x4+x5))) + (s6+s7)*(x2x9) + (s8+s9)*(x2nx3) + s1*(nx1nx6nx9x3x10) + (s2+s3)*((n(x2x1)+(x2x1x3(x4+x5))nx6nx9x3x10) + (s4+s5)*((n(x2x6)+x2x6x3x9(x7+x8))nx9x3x10) + (s6+s7)*(n(x2x9)x3x10) + (s8+s9)*(n(x2nx3)x10) + (s10+s11)*nx11

Рівняння виходів

$Y1 = Y2 = Y3 = Y4 = s0$

$Y5 = Y6 = Y7 = s1$

$Y8 = s2$

$Y9 = s3 + s5 + s7 + s9$

$Y10 = Y12 = s3$

$Y11 = s3 + s5 + s7 + s9$

$Y13 = s4$

$Y14 = Y15 = s5$

$Y16 = s6$

$Y17 = Y18 = s7$

$Y19 = s8$

$Y20 = Y21 = s9$

$Y22 = s10$

$Y23 = s11$

Побудова функціональної схеми автомата

Для побудови функціональної схеми керуючого автомата потрібно отримати сигнал відповідний s на певній ітерації. Використовується дешифратор станів, на вхід йдуть D1D2D3D4. Для побудови керуючого автомату використовуємо схему 74154. А в якості елементів пам’яті використовуємо схеми 7474 – D-тригери.

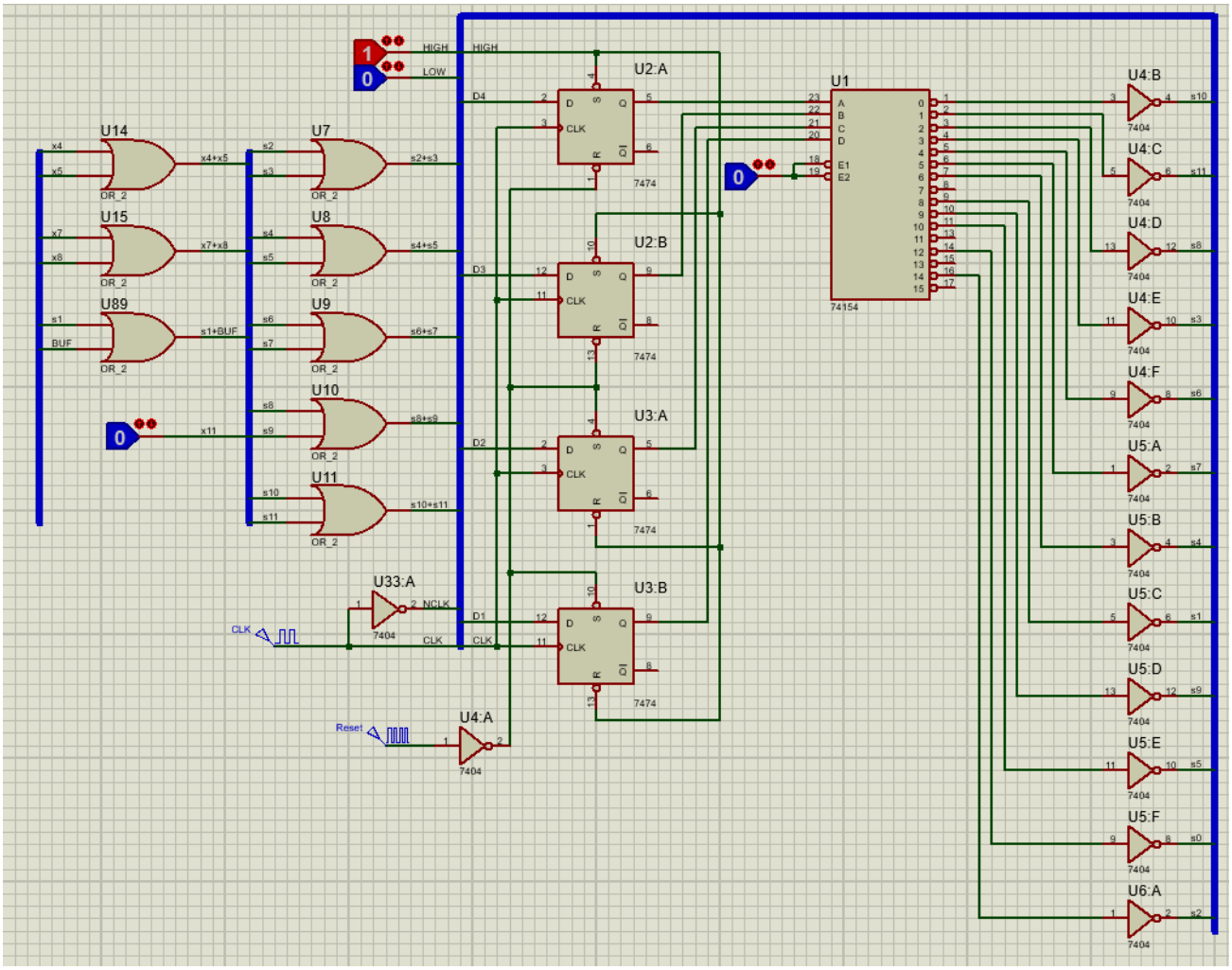


Рисунок 8. Частина функціональної схеми, що складається з комірок пам’яті (7474), дешифратора (74154) та допоміжних сигналів через OR

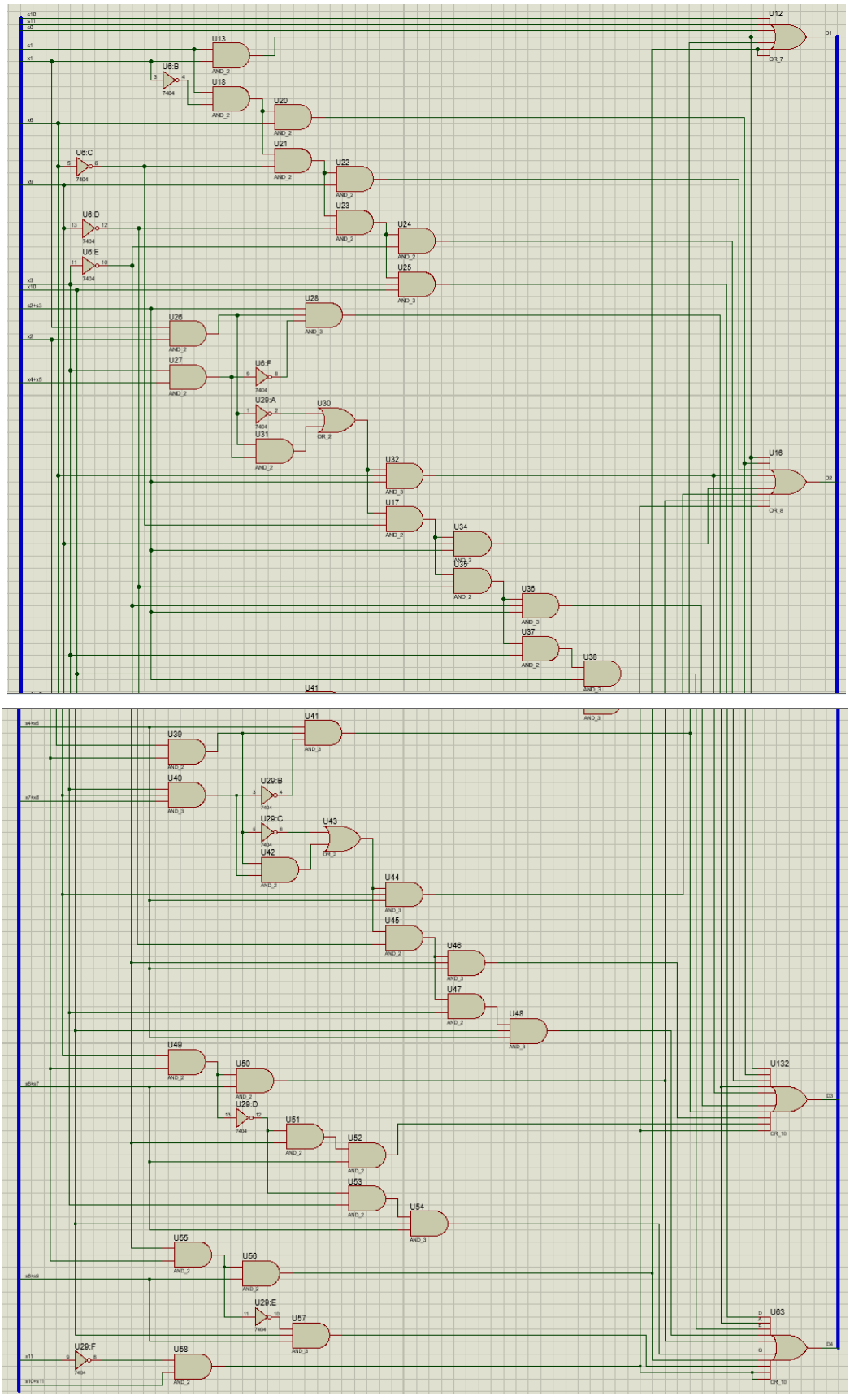


Рисунок 9. Частина функціональної схеми, що складається з комбінаційних схем визначення функцій переходів та функцій виходів

Основні частини операційного автомата

Регістри

На рисунку 10 зображені регістри RGA, RGB, RGC, RGD. Це регістри 74198. В них на початку роботи записуються кількості монет, що вводить користувач.

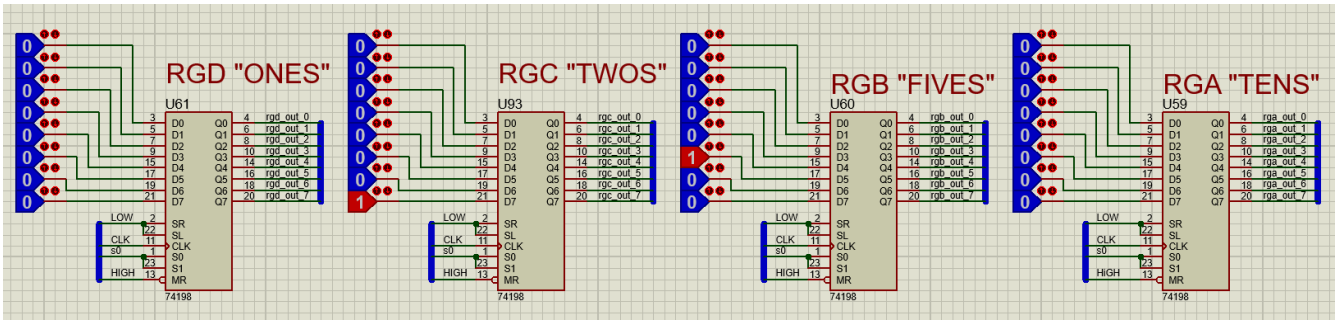


Рисунок 10. Регістри RGA, RGB, RGC, RGD

На рисунку 11 зображені регістри RGE та RGH. В RGE на початку роботи записується число, що вводить користувач (входи схеми 74157 1A-4A), а потім число, що йде з виходу суматора (входи схеми 74157 1B-4B), опис якого наведений нижче (Рисунок). Регістр RGH є початковим буфером, в якомц на початку фіксується значення решти.

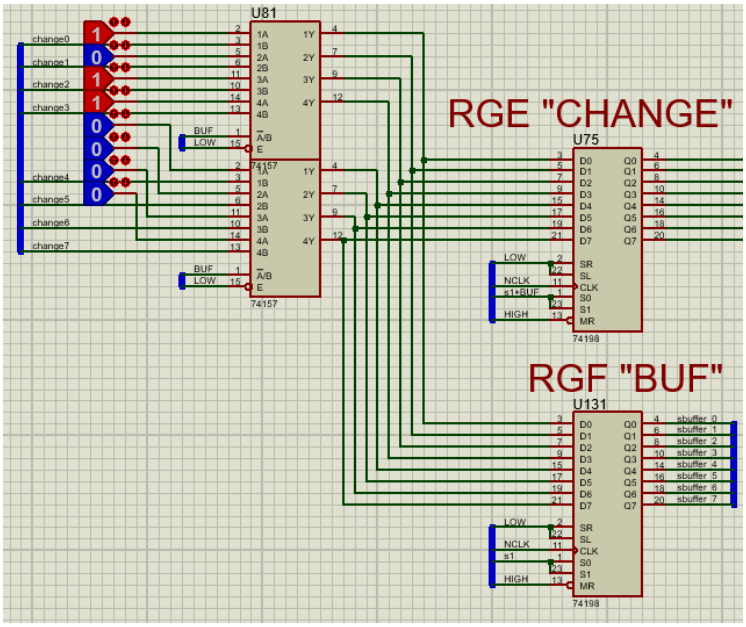


Рисунок 11. Регістри RGE, RGF

В регістр RGH покроково записуються числа-номінали монет.

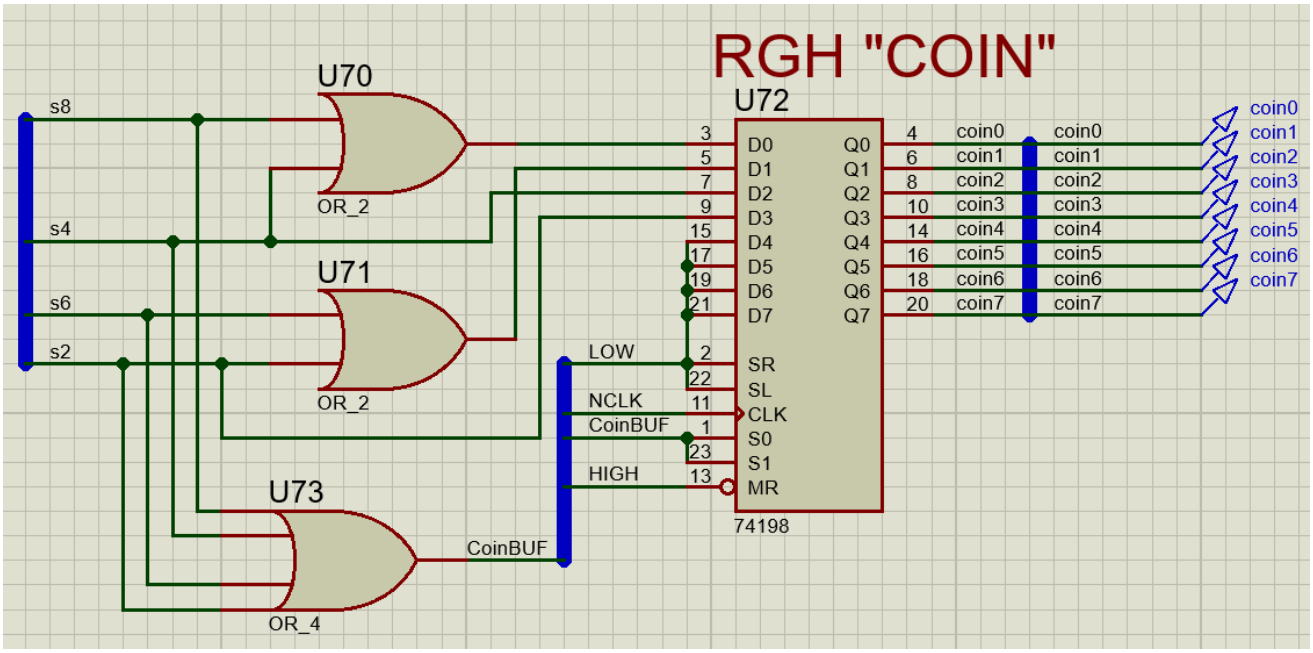


Рисунок 12. Регістр RGH

В регістрі RGG рахується сума монет, що їх під час роботи видає автомат. Використовується при порівнянні запиту на монети та тієї суми, що вдалося идати автомату.

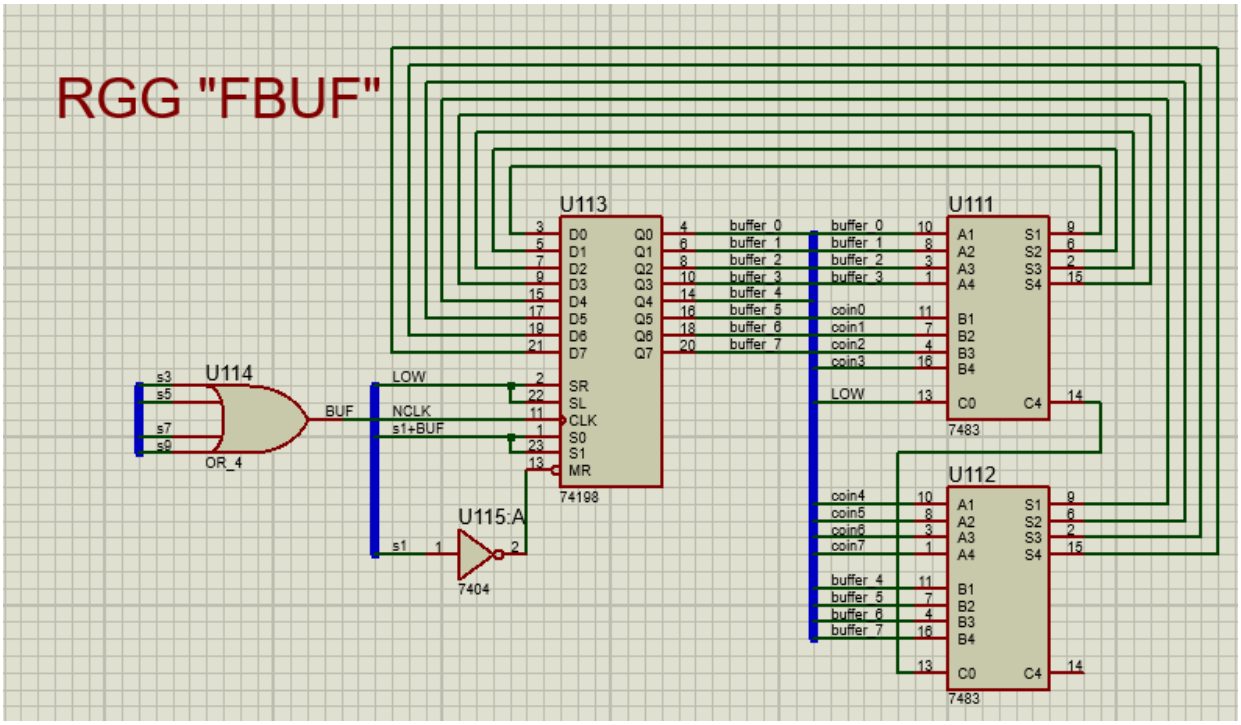


Рисунок 13. Регістр RGG

Реалізація віднімання в зворотному коді

На рисунку зображена комбінаційна схема, що інвертує вхідний сигнал номіналу монети для подачі його на суматор. Це буде віднімання у зворотному коді. Інверсія відбувається за допомогою матриці виключних АБО.

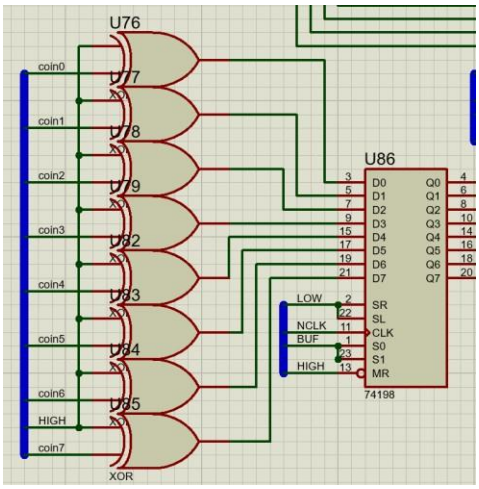


Рисунок 14. Запис в регістр інвертованого числа-номіналу монети

На рисунку зображений суматор, що виконує операцію RGE = RGE-RGH. Цю операцію віднімання суматор виконує у зворотному коді.

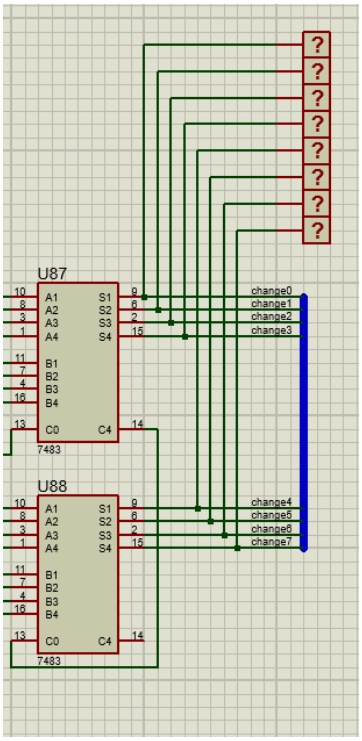


Рисунок 14. Суматор

Реалізація умовних вершин

В схемі є декілька видів умовних вершин, що відрізняються певними ознаками. Наприклад, елементною базою або принципом роботи. Нижче розглянуті вони всі.

Перевірка наявності монет

На рисунку 15 зображений компаратор та лічильники, що слугують для з’ясування правдивості умови RGA != 10. Вихід QA < B реалізує X1. В ході розробки схеми було вирішено, що доцільним є заміна декременту кількості монет на її інкремент, що реалізується лічильником. Знак порівняння в такому випадку змінюється на протилежний.

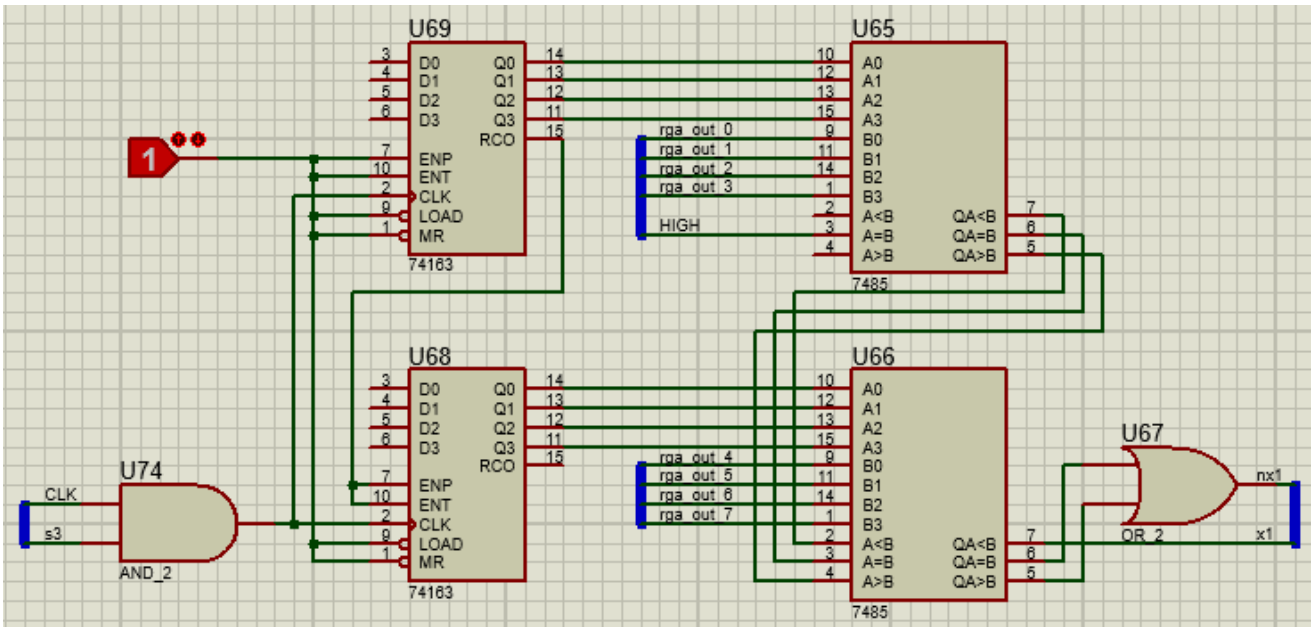


Рисунок 15. Лічильник та компаратор для X1

Аналогічно до опису Рисунку 15 на Рисунках 16, 17 зображені схожі схеми, однак вони стосуються підрахунку використаних монет номіналів «п’ять», «два» та «один».

Слід зауважити, що при підрахунку одиниць на Рисунку 16, X3 буде позитивним, коли за умовою кількість одиниць дорівнює нулю.

Для п’ятірок та двійок ситуація однакова з випадком дев’яток.

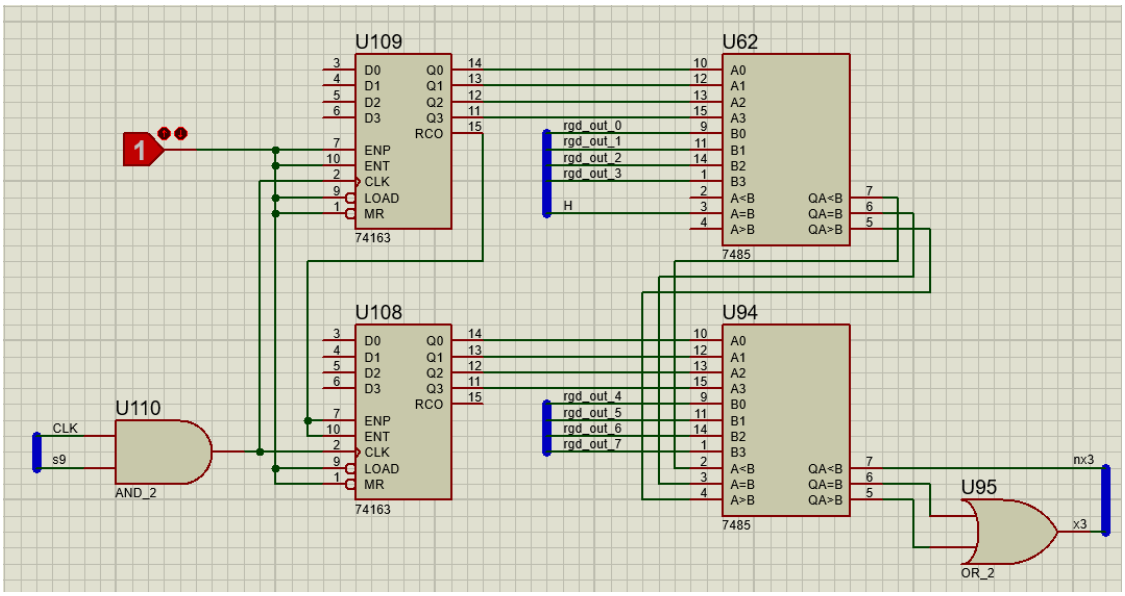


Рисунок 16. Лічильник та компаратор для X3

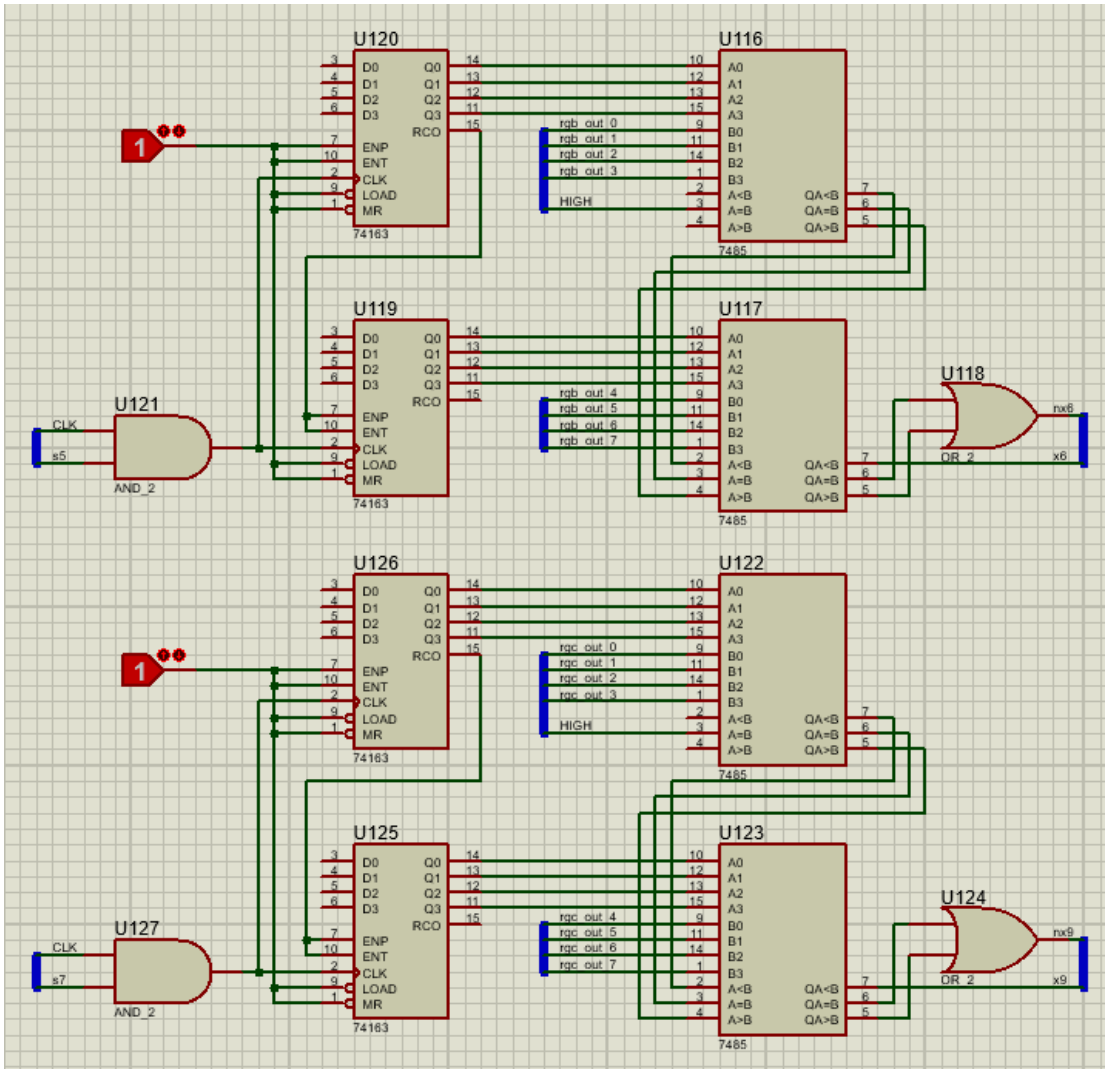


Рисунок 16. Лічильники та компаратори для X6 та X9

Порівняння решти та номіналу

На рисунку 17 зображений компаратор, що слугує для з’ясування правдивості умови RGE >= RGH, тобто умови, що решта більша чи дорівнбє номіналу. Вихід QA < B реалізує NX2. Виходи QA > B та QA = B, що об’єднані елементом але реалізують умову X2, а саме RGE >= RGH.

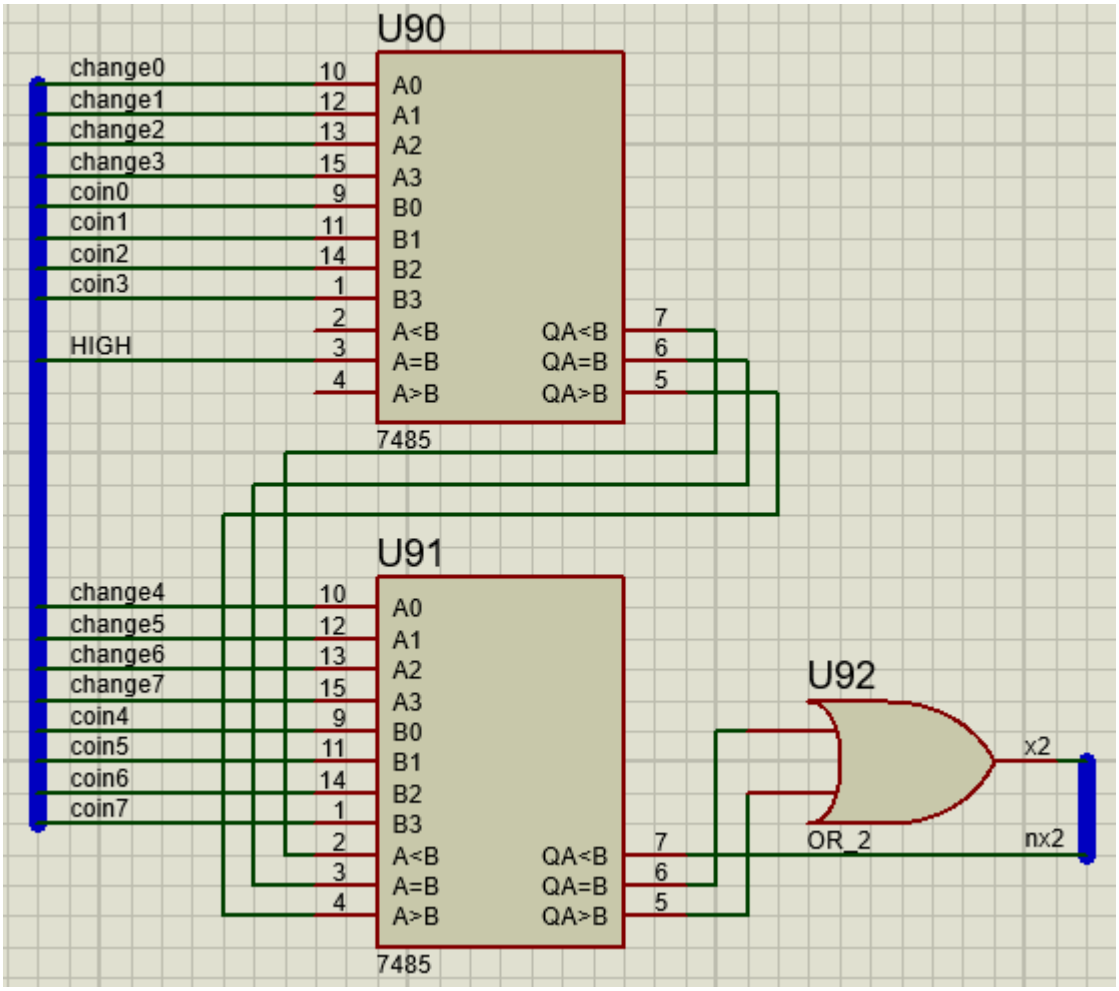


Рисунок 17. Компаратор для X2

Порівняння початкового та фінального буферів

На рисунку 18 зображений компаратор, що слугує для з’ясування правдивості умови RGF == RGG, тобто умови, що початковий буфер дорівнює фінальному. В залежності від виконання цієї умови можна зробити висновок, чи видав автомат решту повністю. Вихід QA = B реалізує X10.

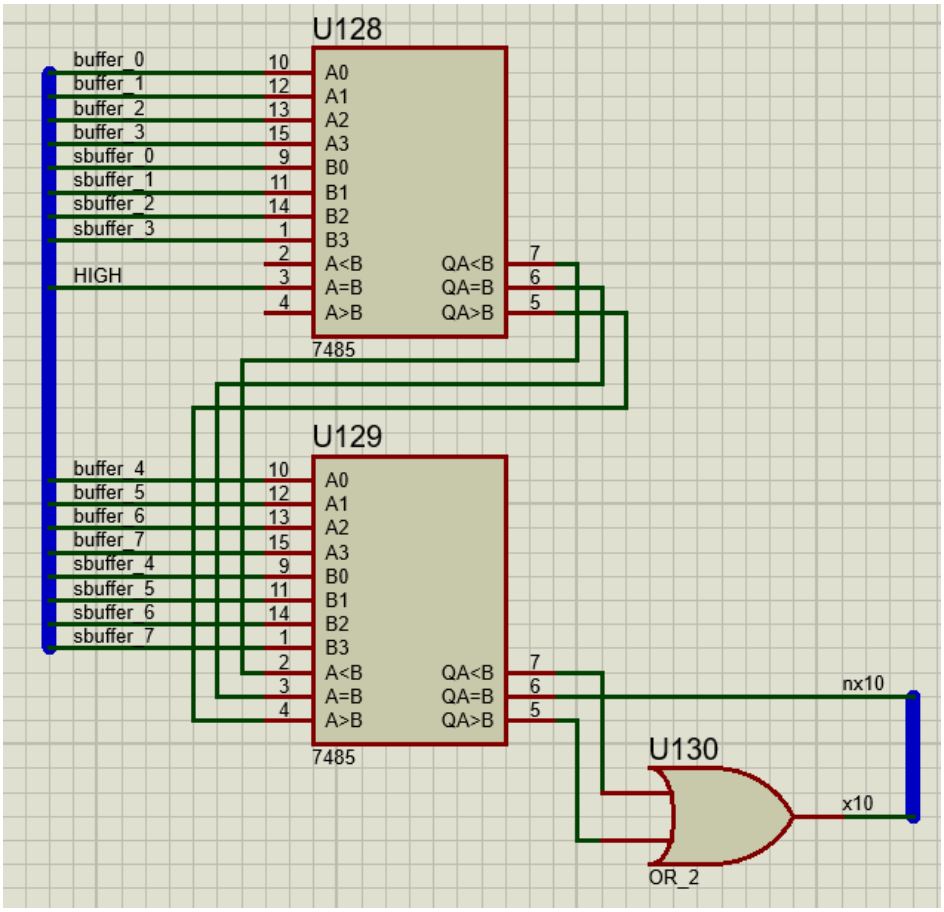


Рисунок 18. Компаратор для X10 («Висновка»)

Порівняння «передбачуваних проблем»

На рисунках 19 – 23 зображені компаратори, що слугують для виявлення чисел, що за відсутності одиниці в попередній (лінійній) версії алгоритму могли бути видані неправильно.

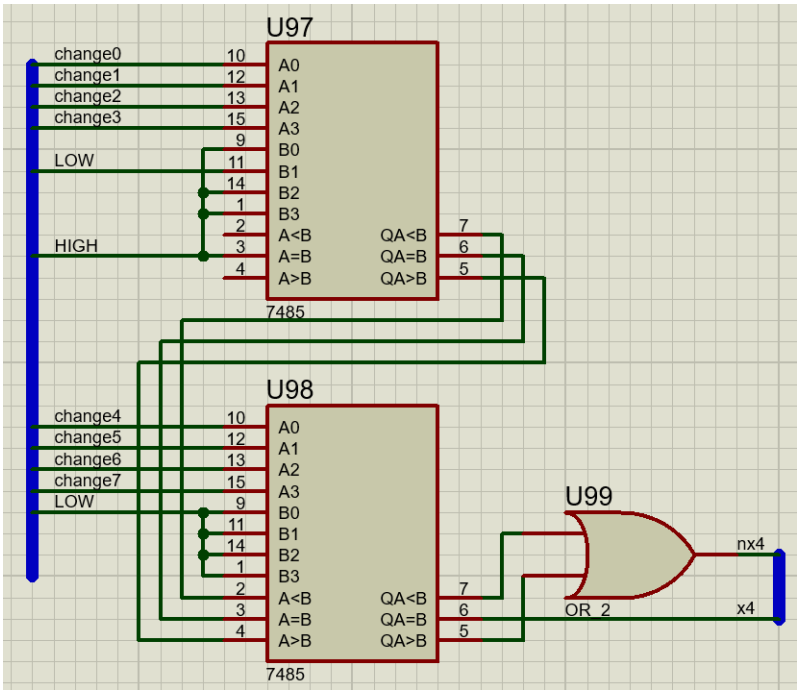


Рисунок 19 Перевірка, чи решта дорівнює «13» (умова X4)

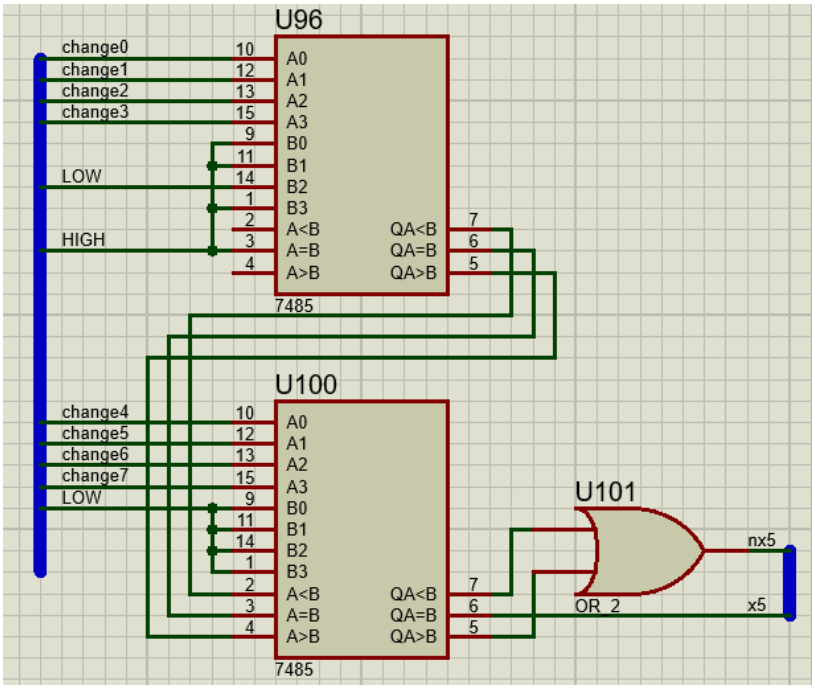


Рисунок 20 Перевірка, чи решта дорівнює «11» (умова X5)

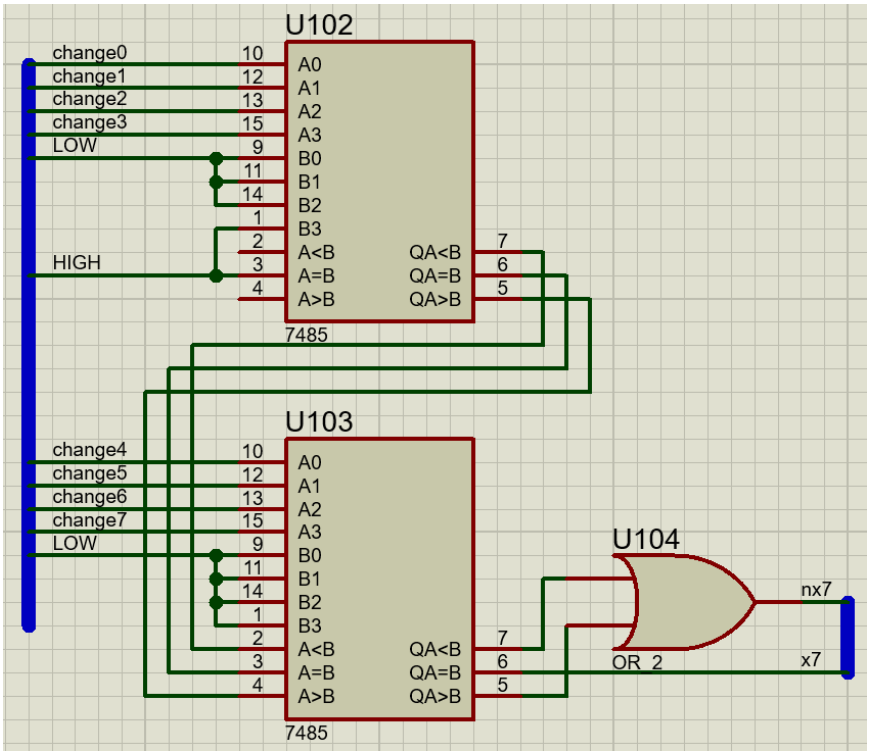


Рисунок 21 Перевірка, чи решта дорівнює «8» (умова X7)

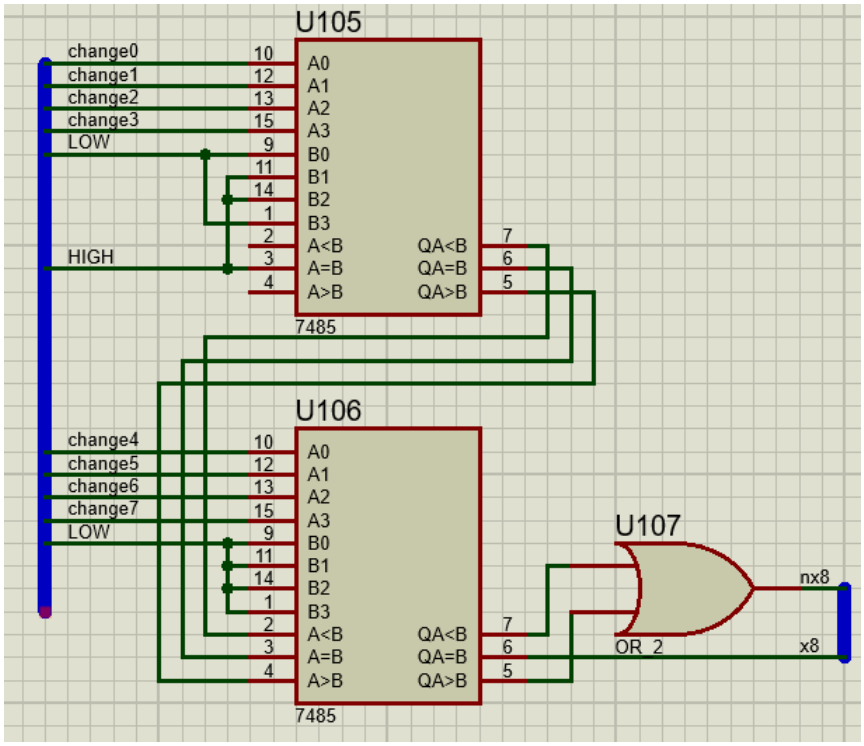


Рисунок 22 Перевірка, чи решта дорівнює «6» (умова X8)

Демонстраційна частина схеми

Сигнали автомату, що на блок-схемі позначались як “TEN”, “FIVE”, “TWO”, “ONE” відповідають за оповіщення, що монета того чи іншого номіналу була віднята від суми решти один раз. Тобто це означає, що при одному сигналі “TEN”, автомат повинен видати одну монету номіналом в 10. Побудована демонстраційна схема показує скільки монет того чи іншого номіналу видає автомат.

Демонстраційна частина схеми

На рисунку зображена частина схеми, що відповідає за демонстрацію результату. Ця схема являє собою чотири лічильника, що відміряють кількість імпульсів, що сигналізують про видачу монети того чи іншого номіналу. Схема 74176 – лічильник. Схема 4511 переводить двійковий код в набір бітів для відображення цифри на семи-сегментному індикаторі.

Також на демонстраційній схемі є два світлодіоди. Зелений світиться, коли автомат видав повну решту, що запросив користувач. Червоний, коли сума виданих монет не співпадає із запрошеною.

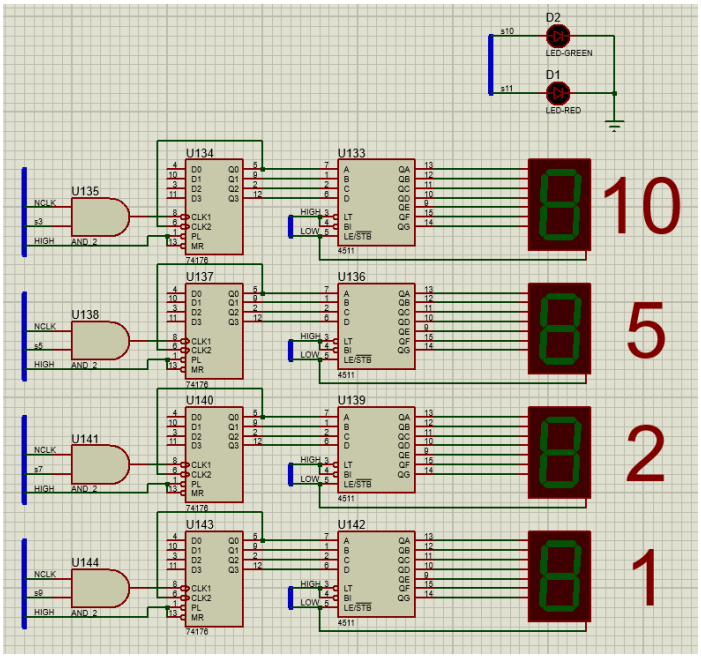


Рисунок 23. Демонстраційна частина схеми

3. Виконана робота

У ході виконання кваліфікаційної роботи бакалавра, використовуючи програмний пакет Proteus 8 Professional, було розроблено схему макету для пристрою, що видає решту. Цю схему можна розділити на три:

- Схема керуючого автомата Мура з комірками пам’яті у вигляді D-тригерів (схема 7474).
- Схема операційного автомата, головною частиною якого є суматор, що виконує віднімання у ЗК.
- Демонстраційна схема, що показує кількість монет за допомогою семисегментних індикаторів.

Найбільш трудомістким процесом в роботі було підключення операційного автомату до керуючого, а також створення та підключення демонстраійної схеми. Однак проблема була вирішена шляхом додавання до схеми допоміжних сигналів для коректної роботи (наприклад, сигнали “coin”, “buf” та інших на Рисунку 8).

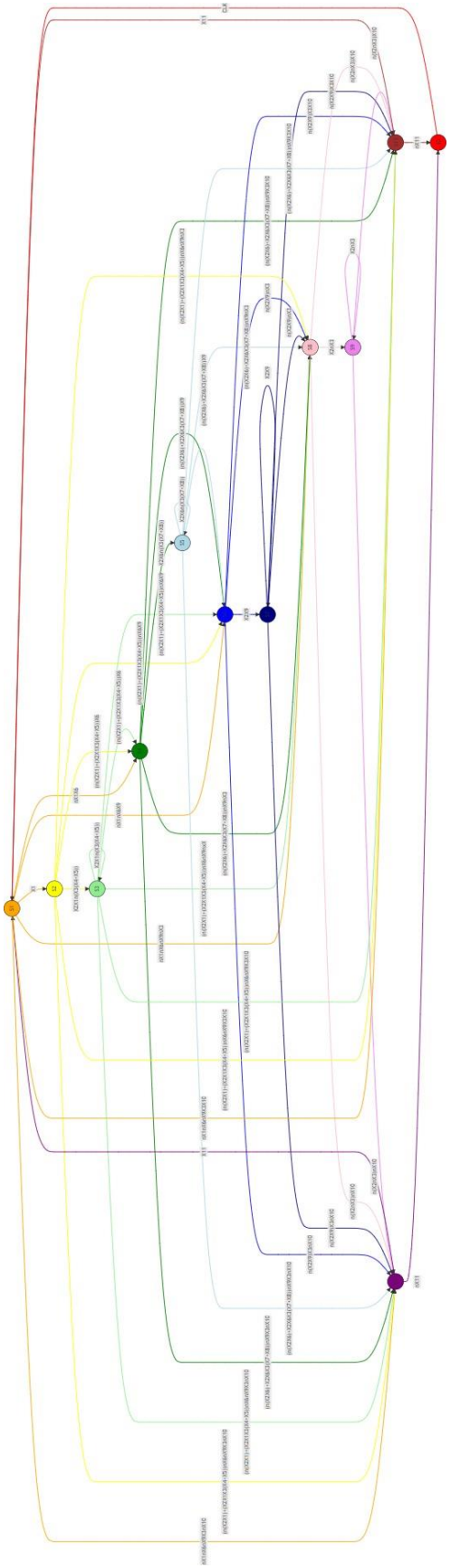
Висновки

- Під час виконання кваліфікаційної роботи бакалавра було з’ясовано, що система автоматизованого проектування Proteus 8 Professional є зручним інструментом для розробки та моделювання електронних схем.
- Генератор діаграм Mermaid є зручним інструментом при побудові різного роду граф- та блок-схем, однак при великому об’ємі даних її використання є недоречним.
- Мікросхеми 74 серії інтегральних мікросхем в якості першого широко розповсюдженого сімейства інтегральних мікросхем є універсальною, оскільки містить сотні пристроїв для побудови найрізноманітніших схем.
- Схема вже може використовуватись як приклад застосування алгоритмів віднімання в курсі «Комп’ютерна логіка»
- Також схема може слугувати для демонстрації злагодженої роботи операційного та керуючого автоматів у курсі «Прикладна теорія цифрових автоматів».

Перелік посилань

1. Прикладная теория цифровых автоматов. К.Г. Самофалов, А.М. Романкевич, В.Н. Валуйский, Ю.С. Каневский, М.М. Пиневич.
2. Омельчук Н.А - Методические указания по курсовому проектированию по дисциплине «Цифровые автоматы».-Запорожье: ЗГИА, 2001.-17 с.
3. Теорія автоматів [електронний ресурс]. – Режим доступу: URL: http://www.mathnet.ru/links/2b5ea8db4d2d3ccc69a960dcf58cfb0a/intv28.pdf
4. Канонічний синтез цифрових автоматів [електронний ресурс]. – Режим доступу: URL: https://ppt-online.org/112446
5. The Computer History Museum, 1963 Standard Logic Families Introduced, retrieved 2008 April 16 [електронний ресурс]. – Режим доступу: URL: http://www.computerhistory.org/semiconductor/timeline/1963-TTL.html
6. Don Lancaster, «TTL Cookbook», Howard W. Sams and Co., Indianapolis, 1975, ISBN 0-672-21035-5 , preface
7. Микросхемы серии К155 К561 К176 74 40 транзисторы [електронний ресурс]. – Режим доступу: URL: https://www.microshemca.ru
8. Вуколов Н. И., Михайлов А. Н. Знакосинтезирующие индикаторы / Под ред. В. П. Балашова. — М. : Радио и связь, 1987. — 592 с.
9. Proteus Design Suite [електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/Proteus_Design_Suite
10. Баужа О.С., Загороднюк С.П., Коновалов А.М. – Методичний посібник до лабораторних робіт з дисципліни «Прикладна теорія цифрових автоматів».
11. В GitHub добавили поддержку диаграмм Mermaid [електронний ресурс]. – Режим доступу: URL: https://habr.com/ru/news/t/651569
12. Прикладна теорія цифрових апаратів / В.І. Жабін, І.А. Жуков, І.А. Клименко, В.В. Ткаченко

ДОДАТОК А



Код для побудови діаграми:

```
graph TD

    S1 --> |X1| S2((S2))
    linkStyle 0 stroke: orange

    S1 --> |"nX1nX6nX9nX3"|S8
    linkStyle 1 stroke: orange
    S3 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6nX9nxX"|S8
    linkStyle 2 stroke: lightgreen
    S4 --> |"(n(X2X6)+X2X6X3(X7+X8))nX9nX3"|S8
    linkStyle 3 stroke: green
    S5 --> |"(n(X2X6)+X2X6X3(X7+X8))nX9nX3"|S8
    linkStyle 4 stroke: lightblue
    S5 --> |"X2X6n(X3(X7+X8))"|S5
    linkStyle 5 stroke: lightblue
    S7 --> |"n(X2X9)nX3"|S8((S8))
    linkStyle 6 stroke: navy
    S7 --> |"X2X9"|S7
    linkStyle 7 stroke: navy

    S6 --> |"X2X9"|S7((S7))
    linkStyle 8 stroke: blue
    S6 --> |"n(X2X9)nX3"|S8
    linkStyle 9 stroke: blue
    S1 --> |"nX1nX6X9"|S6
    linkStyle 10 stroke: orange
    S2 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6X9"|S6
    linkStyle 11 stroke: yellow
    S3 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6X9"|S6
    linkStyle 12 stroke: lightgreen
    S4 --> |"(n(X2X6)+X2X6X3(X7+X8))X9"|S6
    linkStyle 13 stroke: green
    S5 --> |"(n(X2X6)+X2X6X3(X7+X8))X9"|S6((S6))
    linkStyle 14 stroke: lightblue

    S1 --> |"nX1nX6nX9X3nX10"|S10
    linkStyle 15 stroke: orange
    S2 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6nX9X3nX10"|S10
    linkStyle 16 stroke: yellow
    S3 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6nX9X3nX10"|S10
    linkStyle 17 stroke: lightgreen
    S4 --> |"(n(X2X6)+X2X6X3(X7+X8))nX9X3nX10"|S10
    linkStyle 18 stroke: green
    S5 --> |"(n(X2X6)+X2X6X3(X7+X8))nX9X3nX10"|S10
    linkStyle 19 stroke: lightblue
    S6 --> |"n(X2X9)X3nX10"|S10
    linkStyle 20 stroke: blue
    S7 --> |"n(X2X9)X3nX10"|S10
    linkStyle 21 stroke: navy
    S8 --> |"n(X2nX3)nX10"|S10
    linkStyle 22 stroke: pink
    S9 --> |"n(X2nX3)nX10"|S10((S10))
    linkStyle 23 stroke: violet
    S10 --> |"X11"|S1
    linkStyle 24 stroke: purple
    S10 --> |"nX11"|S0
    linkStyle 25 stroke: purple
```

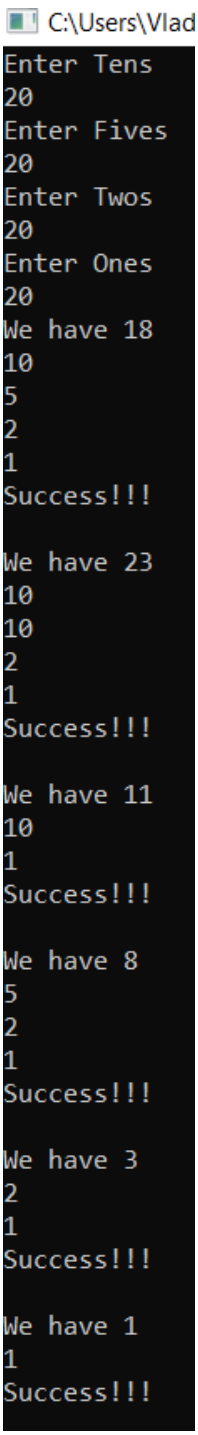
```
S1 --> |"nX1nX6nX9X3X10"|S11
linkStyle 26 stroke: orange
S2 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6nX9X3X10"|S11
linkStyle 27 stroke: yellow
S3 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6nX9X3X10"|S11
linkStyle 28 stroke: lightgreen
S4 --> |"(n(X2X6)+X2X6X3(X7+X8))nX9X3X10"|S11
linkStyle 29 stroke: green
S5 --> |"(n(X2X6)+X2X6X3(X7+X8))nX9X3X10"|S11
linkStyle 30 stroke: lightblue
S6 --> |"n(X2X9)X3X10"|S11
linkStyle 31 stroke: blue
S7 --> |"n(X2X9)X3X10"|S11
linkStyle 32 stroke: navy
S8 --> |"n(X2nX3)X10"|S11
linkStyle 33 stroke: pink
S9 --> |"n(X2nX3)X10"|S11((S11))
linkStyle 34 stroke: violet
S11 --> |"X11"|S1
linkStyle 35 stroke: brown
S11 --> |"nX11"|S0
linkStyle 36 stroke: brown


S4 --> |"X2X6n(X3(X7+X8))"|S5((S5))
linkStyle 37 stroke: green
S1 --> |"nX1X6"|S4
linkStyle 38 stroke: orange
S2 --> |"(n(X2X1)+(X2X1X3(X4+X5))X6"|S4
linkStyle 39 stroke: yellow
S3 --> |"(n(X2X1)+(X2X1X3(X4+X5))X6"|S4((S4))
linkStyle 40 stroke: lightgreen


S2 --> |"X2X1n(X3(X4+X5))"|S3((S3))
linkStyle 41 stroke: yellow
S2 --> |"(n(X2X1)+(X2X1X3(X4+X5))nX6nX9nX3"|S8
linkStyle 42 stroke: yellow
S3 --> |"X2X1n(X3(X4+X5))"|S3
linkStyle 43 stroke: lightgreen
S8 --> |"X2nX3"|S9((S9))
linkStyle 44 stroke: pink
S9 --> |"X2nX3"|S9
linkStyle 45 stroke: violet
S0((S0)) --> |"CLK"|S1((S1))
linkStyle 46 stroke: red

style S0 fill:red, stroke:black
style S1 fill:orange, stroke:black
style S2 fill:yellow, stroke:black
style S3 fill:lightgreen, stroke:black
style S4 fill:green, stroke:black
style S5 fill:lightblue, stroke:black
style S6 fill:blue, stroke:black
style S7 fill:navy, stroke:black
style S8 fill:pink, stroke:black
style S9 fill:violet, stroke:black
style S10 fill:purple, stroke:black
style S11 fill:brown, stroke:black
```

ДОДАТОК Б

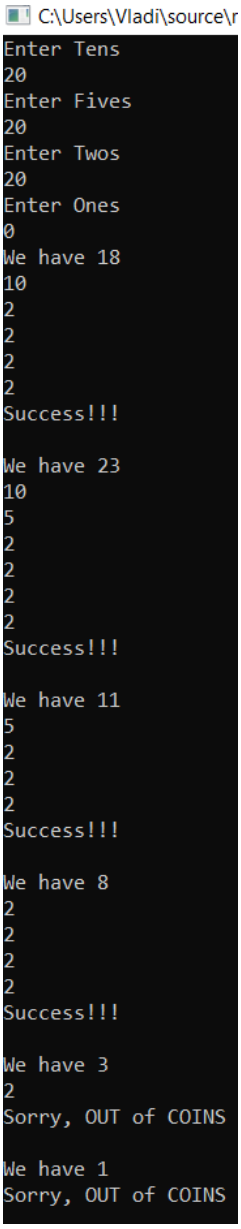
Програмний код мовою C#, що демонструє роботу алгоритму. Та результат його виконання для чисел 28, 23, 11, 8, 3, 1 при різних наборах наявних монет.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomaSmart
{
    class Program
    {
        static void GetChange(int change, int tens, int fives, int twos, int ones)
        {
            Console.WriteLine("We have " + change.ToString());

            int start = change;
            int buffer  =  0;
            int coin = 0;


            if (tens != 0)
            {
                coin = 10;
                while (change >= coin && tens != 0)
                {
                    if (ones == 0 && (change == 13 || change == 11))
                    {
                        break;
                    }
                    else
                    {
                        change = change - coin;
                        Console.WriteLine("10");
                        buffer += 10;
                        tens--;
                    }
                }
            }

            if (fives != 0)
            {
                coin = 5;
                while (change >= coin && fives != 0)
                {

                    if (ones == 0 && twos != 0 && (change == 8 || change == 6))
                    {
                        break;
                    }
                    else
                    {
                        change = change - coin;
                        Console.WriteLine("5");
                        buffer += 5;
                        fives--;
                    }
                }
```

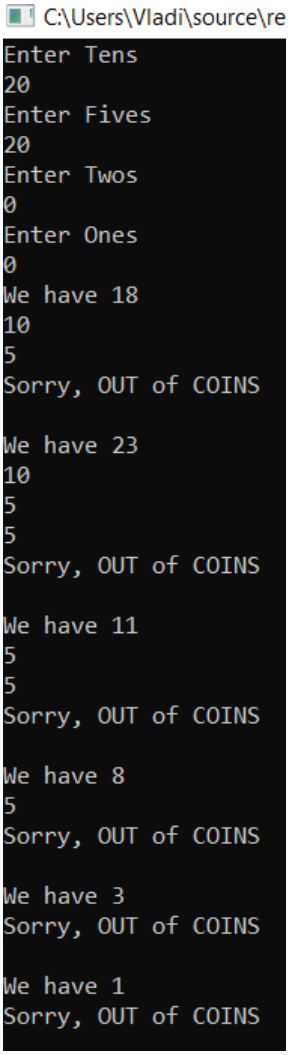
```
            }

            if (twos != 0)
            {
                coin = 2;
                while (change >= coin && twos != 0)
                {
                    change = change - coin;
                    Console.WriteLine("2");
                    buffer += 2;
                    twos--;
                }
            }

            if (ones != 0)
            {
                coin = 1;
                while (change >= coin && ones != 0)
                {
                    change = change - coin;
                    Console.WriteLine("1");
                    buffer += 1;
                    ones--;
                }
            }

            if (start == buffer)
            {
                Console.WriteLine("Success!!!");
            }
            else
            {
                Console.WriteLine("Sorry, OUT of COINS");
            }
            Console.WriteLine();



        }

        static void Main(string[] args)
        {
            Console.WriteLine("Enter Tens");
            int tens = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Fives");
            int fives = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Twos");
            int twos = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Ones");
            int ones = Convert.ToInt32(Console.ReadLine());

            GetChange(18, tens, fives, twos, ones);
            GetChange(23, tens, fives, twos, ones);
            GetChange(11, tens, fives, twos, ones);
            GetChange(8, tens, fives, twos, ones);
            GetChange(3, tens, fives, twos, ones);
            GetChange(1, tens, fives, twos, ones);

            Console.ReadLine();
        }
    }
}
```



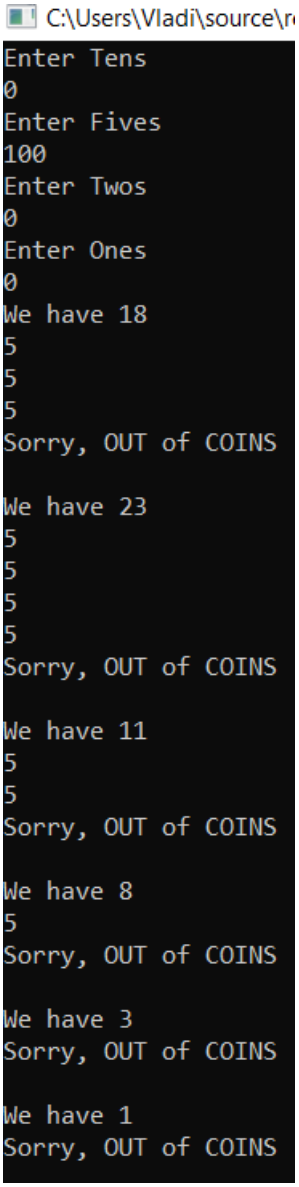
Є всі номінали



Відсутні одиниці



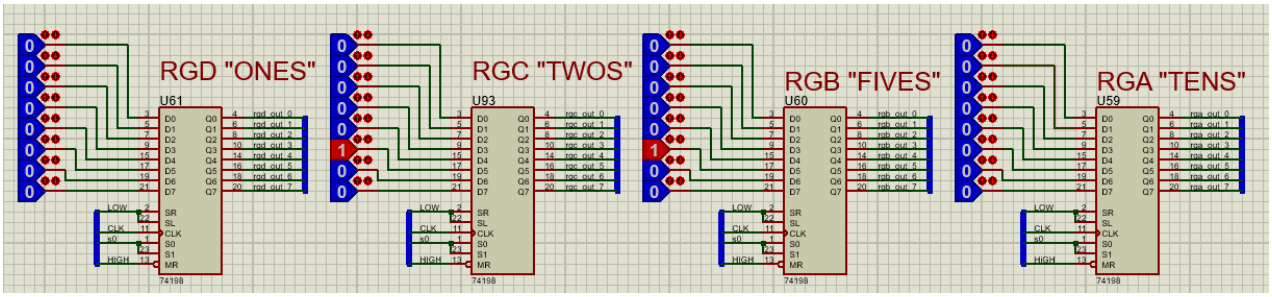
Відсутні одиниці та двійки



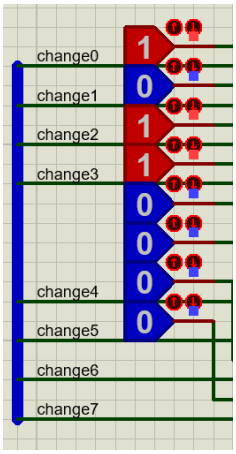
Наявні лише п’ятірки

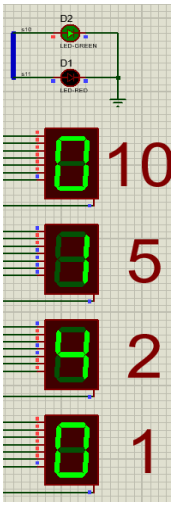
ДОДАТОК В

Проведемо симуляцію роботи схеми для чисел 13 та 21 при різних наборах монет. Результати виконання коду та сам код можна знайти у ДОДАТКУ Б

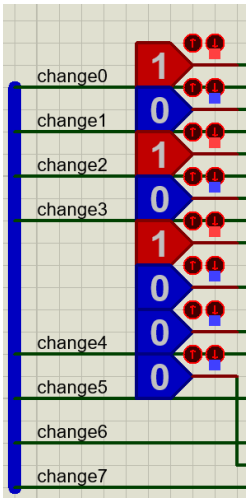


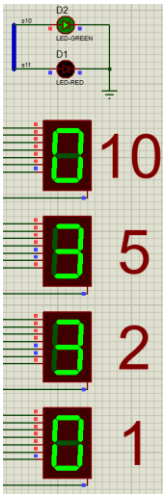
Присутні лише п’ятірки та двійки



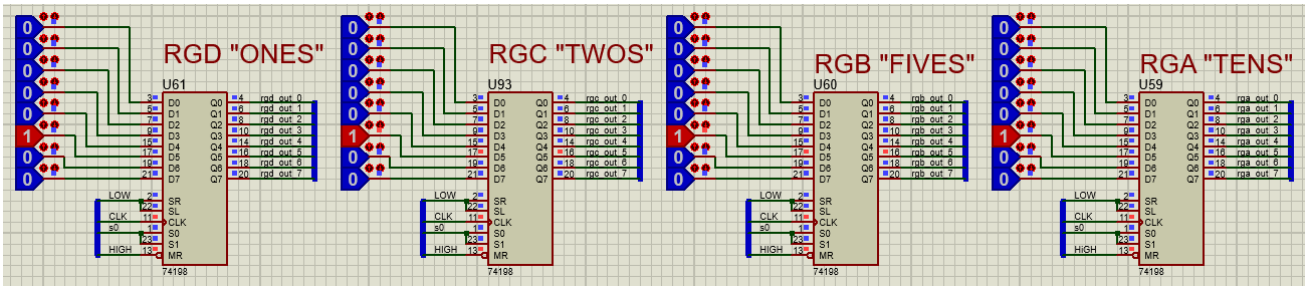


$13_{10} = 1101_2$ = 0*10 + 1*5 + 4*2 + 0*1

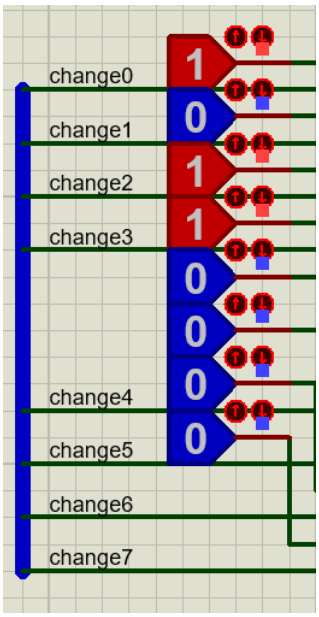


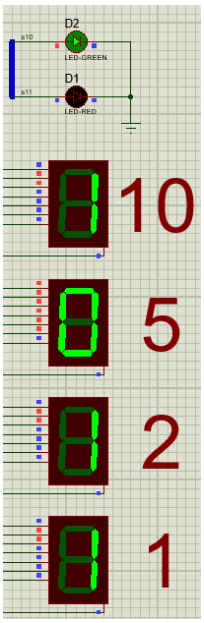


$21_{10} = 10101_2$ = 0*10 + 3*5 + 3*2 + 0*1

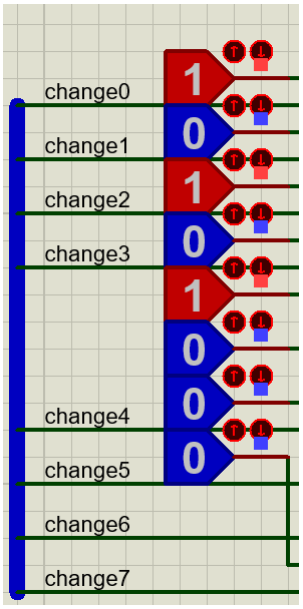


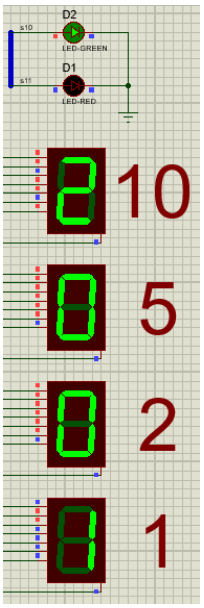
Присутні всі номінали



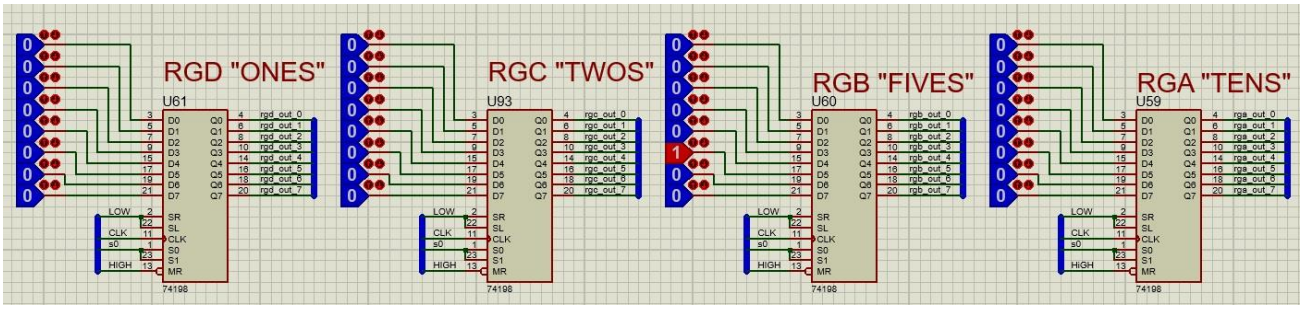


$13_{10} = 1101_2$ = 1*10 + 0*5 + 1*2 + 1*1

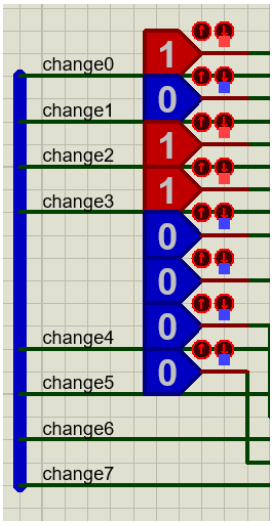


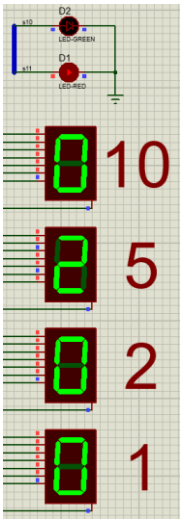


$21_{10} = 10101_2$ = 2*10 + 0*5 + 0*2 + 1*1

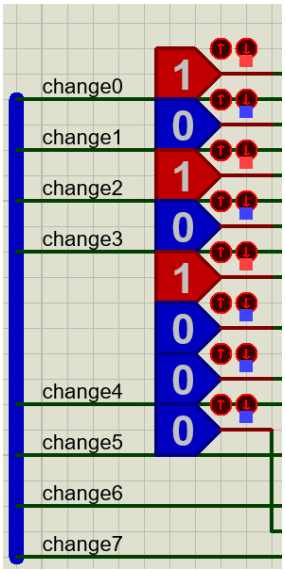


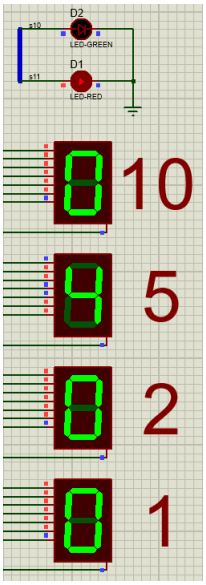
Наявні лише п’ятірки





$13_{10} = 1101_2$!= 0*10 + 2*5 + 0*2 + 0*1





$21_{10} = 10101_2$!= 4*10 + 4*5 + 0*2 + 0*1

Можемо спостерігати належну роботу зібраної схеми у всіх трьох варіантах запуску.