МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
"ЛЬВІВСЬКА ПОЛІТЕХНІКА"

НАВЧАЛЬНИЙ ПОСІБНИК

«Комп'ютерна схемотехніка: курсове проектування»

для студентів галузі знань «12 Інформаційні технології»
спеціальності 123 «Комп'ютерна інженерія»

Рекомендовано Науково-методичною радою
Національного університету "Львівська політехніка"

Львів 2022

Автор: Клушин Ю.С., к.т.н., доцент кафедри ЕОМ

Рецензенти:
Оліярник Б. О. – головний конструктор Державного підприємства “Львівський державний завод “ЛОРТА”, д.т.н., професор, Лауреат Державної премії України в галузі науки і техніки;
Лукенюк А. А. – директор Львівського центру Інституту космічних досліджень НАН та ДКА України, к.т.н.;
Глухов В.С., д.т.н., професор кафедри ЕОМ

Рекомендовано Науково-методичною радою
Національного університету “Львівська політехніка”
(протокол № від « » 2022 року)

Навчальний посібник «Комп’ютерна схемотехніка: курсове проектування» для студентів галузі знань «12 Інформаційні технології» спеціальності 123 «Комп’ютерна інженерія» / Автор Клушин Ю.С. – Львів: Видавництво Національного університету “Львівська політехніка”, 2022. – 154 с.

До посібника увійшли методичні вказівки до курсового проектування з навчальної дисципліни **«Комп’ютерна схемотехніка»**, індивідуальні завдання, практичні приклади, література і термінологічний словник.

Навчальний посібник призначений для студентів спеціальності “Комп’ютерна інженерія” галузі знань “Інформаційні технології”.

ISBN

ЗМІСТ

ВСТУП

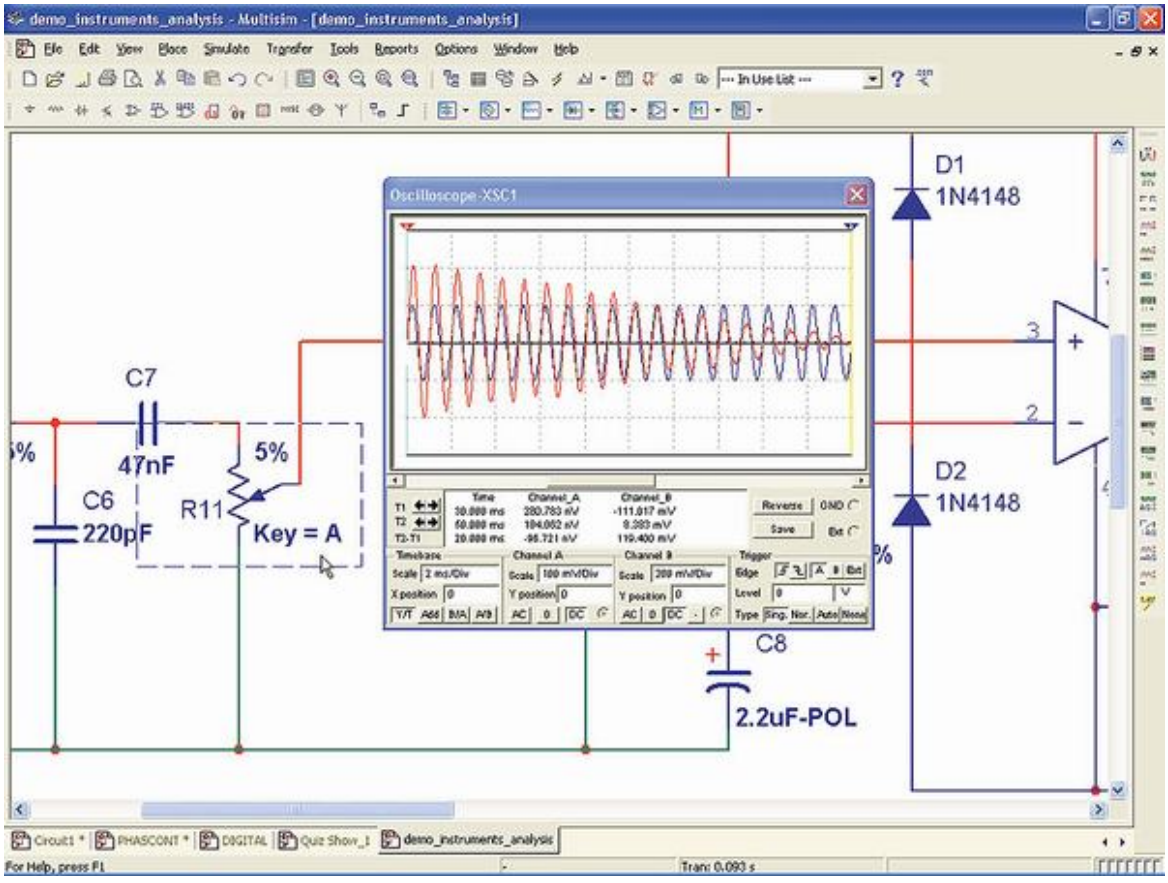
Курсове проектування обчислювального пристрою з дисципліни «Комп'ютерна схемотехніка», що викладається на кафедрі Електронні обчислювальні машини, складаються з чотирьох частин – з обзору та опису існуючих програмних середовищ для модулювання електронних пристроїв, з опису основних типів схем, з теоретичної частини пов'язаною з вибором електронних компонентів для проетування обчислювача, та практичної частини, де показані приклади проектування обчислювального пристрою в системах Multisim, Active HDL та Proteus. Такі електронні компоненти, як регістори, лічильники, суматори, арифметико – логічні пристрої, автомати керування, пам'ять входять до складу практично будь-яких сучасних обчислювальних приладів і систем. Знання та практичні навички, отримані при проектуванні обчислювального пристрою, будуть використовуватись студентами для вирішення практичних задач розробки та розрахунку різних обчислювальних пристроїв і систем. Методичні вказівки з курсового проектування електронного обчислювача розраховані на студентів, які навчаються за спеціальністю 123 – «Комп'ютерна інженерія». Також можуть бути корисними і для студентів інших спеціальностей, пов'язаних з проектуванням електронних пристроїв. Проектування електронного обчислювача може виконуватися на персональному комп'ютері в Proteus, Active HDL та Multisim. Основу навчального посібника складають три основних програмних середовищ, в яких необхідно синтезувати та промоделювати основні електронні пристрої, зробити висновки щодо отриманих результатів. Навчальний посібник оформлений доступно для сприйняття та відтворення і забезпечений відповідними детальними інструкціями та ілюстраціями, завдяки цьому студенти зможуть проектувати та модулювати роботу електронного обчислювача без особливих ускладнень та отримати задоволення від здобутих знань та практичних навичок.

1. СИСТЕМИ ДЛЯ СХЕМОТЕХНІЧНОГО ПРОЕКТУВАННЯ ЕЛЕКТРОННИХ ЗАСОБІВ

MULTISIM

При проектуванні сучасних радіоелектронних пристроїв неможливо обійтися без комп'ютерних методів розробки через складність та об'ємність виконуваних робіт. Створення радіоелектронних пристроїв вимагає високої точності та глибокого аналізу, у зв'язку з чим виникає потреба застосування на стадії проектування сучасних програмних засобів.

Multisim використовується у світі програмного забезпечення для проектування електричних схем, їх тестування та налагодження. У комплект продуктів NI Circuit Design Suite входять засоби для створення електричних схем, а також розробки та трасування друкованих плат на професійному рівні.



Програма Multisim призначена для схемотехнічного проектування електронних засобів та містить практично всі основні елементи

електронних ланцюгів. Зручність застосування цього пакету при моделюванні електронних пристроїв полягає у відображенні на екрані монітора схеми досліджуваного пристрою та контрольно-вимірювальних приладів, передні панелі яких з органами управління максимально наближені до їх промислових аналогів. Концепція віртуальних приладів – простий та швидкий спосіб побачити результат за допомогою імітації реальних подій.

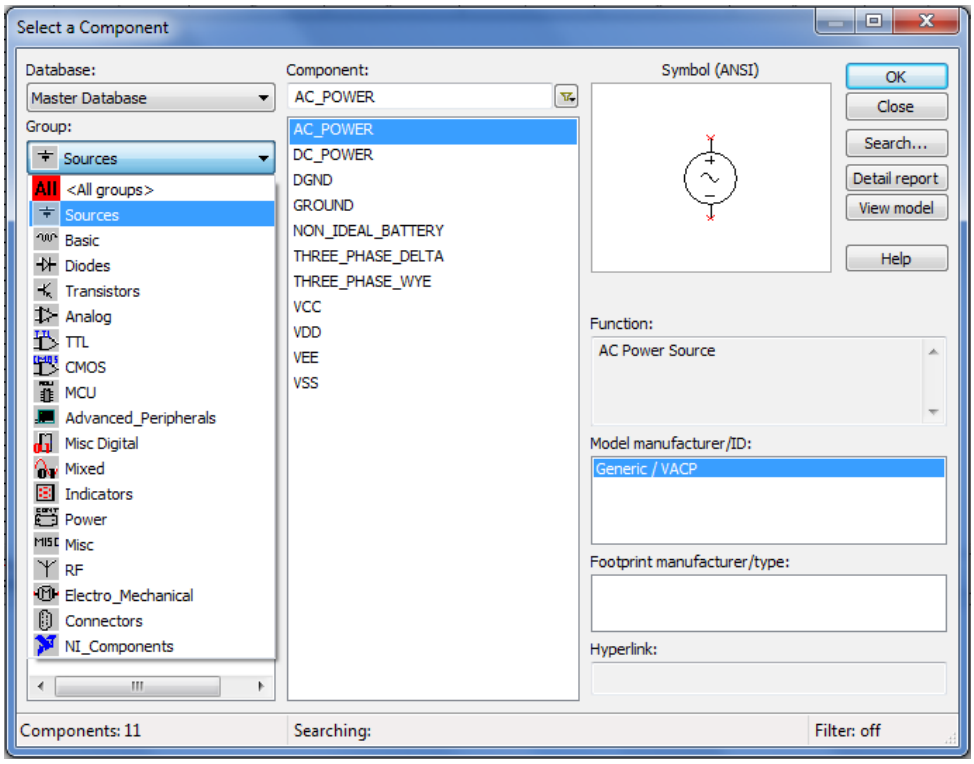
Multisim використовується у світі програмного забезпечення для проектування електричних схем, їх тестування та налагодження. У комплект продуктів NI Circuit Design Suite входять засоби для створення електричних схем, а також розробки та трасування друкованих плат на професійному рівні.

Програма Multisim є справжньою лабораторією схемотехнічного моделювання, яка завдяки простому та зручному інтерфейсу дозволяє з легкістю моделювати складні принципові схеми та проектувати багатошарові друковані плати. У розпорядженні користувачів широкий набір бібліотечних компонентів, параметри та режими роботи яких можна змінювати у широкому діапазоні значень. Під час підготовки даного лабораторного практикума, використовувалося програмне середовище Multisim 12.0. У цій версії значно збільшено обсяг і якість бібліотек компонентів: по відношенню до версій 10.0 та 11.0 було додано більш ніж 1500 компонентів, нові біполярні джерела струму та напруги, рідкокристалічні графічні індикатори. У пакеті присутні також джерела струму та напруги з впливом різної форми, функціональні перетворювачі сигналів (перемножувачі, дільники сигналів), пристрої на основі операційних підсилювачів, цифрові елементи, електромеханічні та ВЧ-компоненти, які не завжди є в інших подібних програмах, таких як Proteus та Crocodile Technology. Multisim 12.0 стійко працює під керуванням Windows XP/Vista/7/10 (32/64 біт).

Програма, що розглядається, дозволяє підключати до схеми, розробленої в її середовищі, віртуальні прилади — програмні моделі контрольно-вимірювальних приладів, які відповідають реальним. Використання віртуальних приладів в Multisim (осциллографів, генераторів сигналів, мережевих аналізаторів тощо) - це простий і зрозумілий метод взаємодії зі схемою, що майже не відрізняється від традиційного при тестуванні або створенні радіоелектронного пристрою.

За замовчуванням використовується база елементів Master Database. Компоненти, що містяться в ній, розділені на такі групи:

- Sources містить джерела живлення, заземлення;
- Basic - резистори, конденсатори, котушки індуктивності і т.д.
- Diodes – містить різні види діодів.
- Transistors – містить різні види транзисторів.
- Analog – містить усі види підсилювачів: операційні, диференціальні, інвертуючі та ін.
- TTL - містить елементи транзисторно-транзисторної логіки
- CMOS – містить елементи КМОП-логіки.
- MCU Module – керуючий модуль багатоточкового зв'язку.
- Advanced_Peripherals – зовнішні пристрої, що підключаються.
- Misc Digital – різні цифрові пристрої.



- Mixed – комбіновані компоненти
- Indicators - містить вимірювальні прилади та ін.

Але все ж таки у даної системи є і недолік - невеликий вибір компонентів бібліотеки мікроконтролерів, до складу якої входять лише такі представники: x8051, x8052, PIC16F84, PIC16F84A. Причому склад цієї бібліотеки не змінювався, починаючи з версії 10.

Multisim пропонує достатню кількість віртуальних інструментів, які ви можна використовувати для вимірювання та дослідження поведінки схеми. Ці інструменти встановлюються, використовуються і показують подібно до їхніх реальних еквівалентів. Використання віртуальних інструментів – найпростіший спосіб перевірити поведінку схеми та побачити результати симуляції. На додаток до стандартних інструментів, які є в Multisim, можна створити свої власні, використовуючи LabVIEW - графічне оточення для створення гнучких та масштабованих стендів, вимірювачів та керуючих додатків.

PROTEUS

Пакет є системою моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice. Відмінною рисою пакету PROTEUS VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP 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. Додатково до пакету PROTEUS VSM входить система проектування друкованих плат. 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.

Також до складу восьмої версії входить середовище розробки VSM Studio, що дозволяє швидко написати програму для мікроконтролера, що використовується в проекті, та скомпілювати.

Пакет є комерційним. Безкоштовна ознайомча версія характеризується повною функціональністю, але немає 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.

Система підтримує підключення нових елементів (SPICE) та підключення різних компіляторів (PICOLO, ARM-подібні, AVR і далі).

Proteus VSM включає ряд віртуальних інструментів, включаючи осцилограф, логічний аналізатор, генератор функцій, генератор шаблонів і віртуальний термінал, а також прості вольтметри і амперметри. Крім того, симулятор може відображати логічний стан кожного виводу за

допомогою кольорових точок. Це може бути надзвичайно корисним при однокроковому коді введення/виводу.

Якщо ви хочете провести докладні вимірювання на графіках або виконати інші типи аналізу, такі як аналіз частоти, спотворень, шуму або розгортки аналогових ланцюгів, ви можете зробити це опціями розширеного моделювання.

Proteus VSM використовує схематичне захоплення ISIS, основні додатки якого включають «Спливаючий кошик для деталей». Кошик деталей та оглядове вікно мають новий режим, в якому їх можна приховати, щоб максимально збільшити простір, відведений вікну редагування.

Найбільш цікавою та важливою особливістю Proteus VSM є його здатність моделювати взаємодію між програмним забезпеченням, що працює на мікроконтролері, та будь-якою підключеною до нього аналоговою чи цифровою електронікою.

Модель мікроконтролера знаходиться на схемі разом із іншими елементами дизайну вашого продукту. Він імітує виконання вашого об'єктного коду (машинного коду) як справжній чіп. Якщо програмний код записує до порту, відповідно змінюються логічні рівні у схемі, і якщо схема змінює стан висновків процесора, це буде видно вашому програмному коду, як і в реальному житті.

Моделі ЦП VSM повністю імітують порти вводу-виводу, переривання, таймери, USART та інші периферійні пристрої, присутні у кожному підтримуваному процесорі. Це зовсім не простий програмний симулятор, оскільки взаємодія всіх цих периферійних пристроїв із зовнішньою схемою повністю моделюється до рівня сигналу.

VSM може навіть моделювати проекти, що містять кілька процесорів, оскільки досить просто розмістити два або більше процесорів на схемі та з'єднати їх разом.

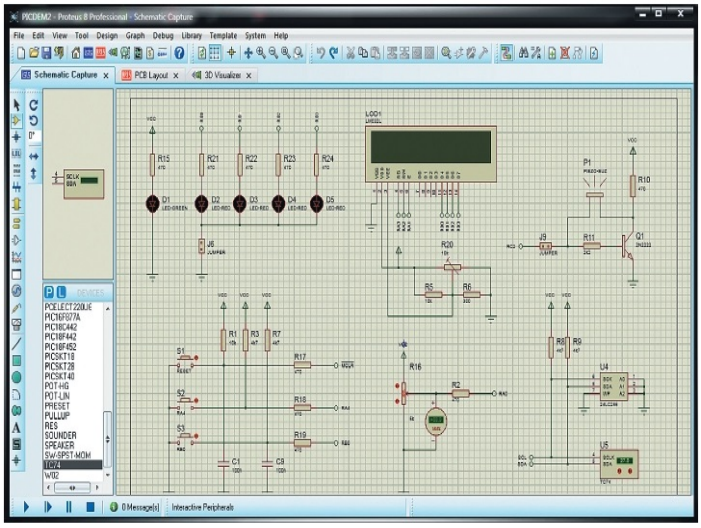
У той час як Proteus VSM вже унікальний своєю здатністю запускати симуляції повних мікроконтролерних систем у режимі, близькому до реального, його реальна міць полягає у здатності виконувати ці симуляції у покроковому режимі. Це працює так само, як ваш улюблений програмний наладчик, за винятком того, що покроково виконуючи код, ви можете спостерігати ефект на всій конструкції, включаючи всю електроніку, зовнішню по відношенню до мікроконтролера.

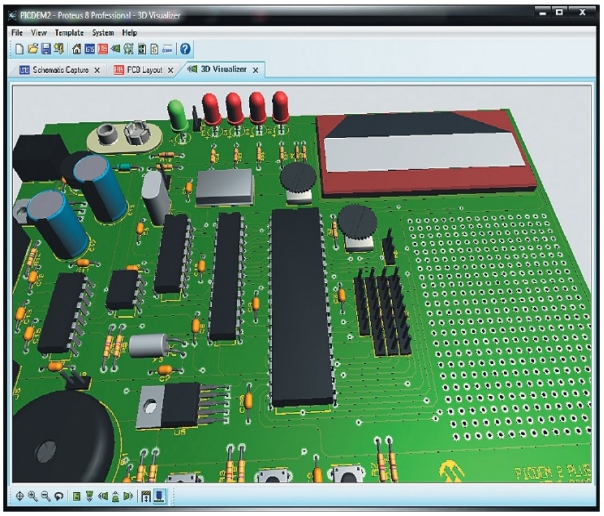
На додаток до моделей мікропроцесорів для кожного підтримуваного сімейства, кожен пакет Proteus VSM включає бібліотеку з більш ніж 6000 моделей пристроїв, таких як: Різні стандартні електронні компоненти, включаючи резистори, конденсатори, діоди, транзистори, тиристори, операційні розв'язки, операційні. 555 таймерів і т.д.

У комплекті продуктів Proteus передбачено засоби для формування електричних схем (редактор ISIS), а також для розробки та трасування друкованих плат (редактор ARES). Редактор ISIS використовується для проектування електричних схем, їх тестування та налагодження. Редактор ARES - це PCB-додаток програми Proteus, що застосовується для створення друкованих плат, виконання певних функцій CAD-систем та підготовки результатів проектування до виробництва. ARES має можливість автоматизованого розміщення компонентів на платі та автоматичного трасування і дозволяє фахівцям працювати в його середовищі так само, як у системі

3D-моделювання, в результаті друкована плата та її компоненти будуть відображені у реальному вигляді. Засоби ARES здатні формувати тривимірні моделі компонентів із плоских графічних даних із бібліотек топологічних посадкових місць. Особливість редактора ISIS – наявність віртуальних вимірювальних приладів, що імітують реальні аналоги. До складу ISIS входять ефективні засоби графічної обробки результатів моделювання. Крім того, програма дозволяє проводити аналіз цифро-

аналогових та цифрових схем великого ступеня складності. Наявні у програмі бібліотеки містять великий набір поширених електронних компонентів. Солідний комплект приладів допомагає виконувати вимірювання різних величин, задавати вхідні дії, будувати графіки. Всі прилади зображуються у вигляді максимально наближеному до реального, тому працювати з ними просто і зручно. У ISIS є великий набір віртуальних інструментів щодо вимірювань, робота з якими було розглянуто в [2], [3]. Результати моделювання можна вивести на принтер або передати до текстового або графічного редактора для подальшої обробки. На рис. 1 показані розроблені в Proteus схема електрична принципова (рис. 1а), розведення друкованої плати (рис. 1б) та 3D-вид пристрою (рис. 1в).



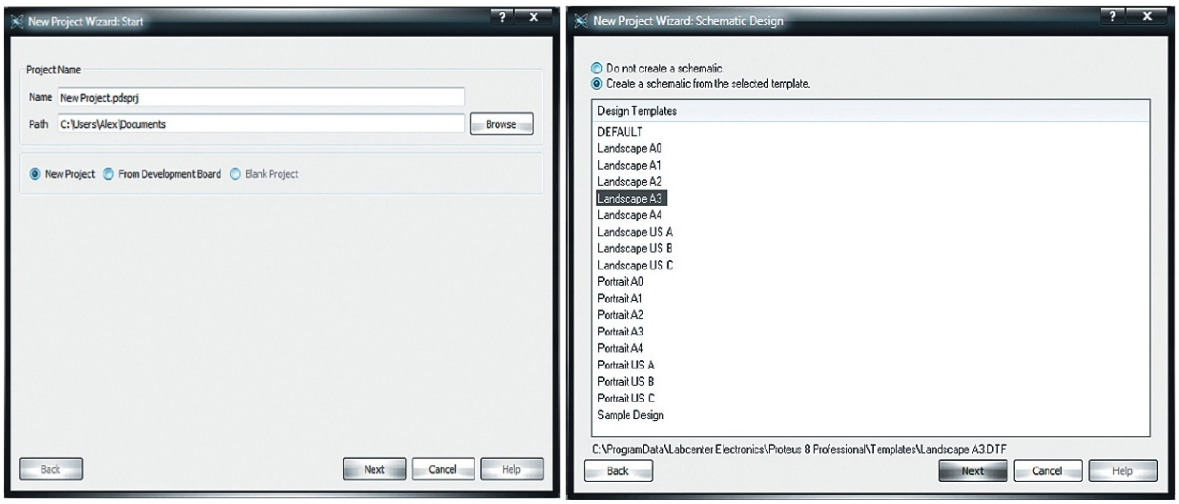


Proteus є так званим середовищем наскрізного проектування, що дозволяє створювати пристрій, починаючи з проектування його принципової схеми і закінчуючи виготовленням друкованої плати, і контролювати кожен етап його виробництва. p align="justify"> Першим етапом проектування вузла друкованої плати в системі Proteus є розробка схеми електричної принципової, яка виконується в редакторі ISIS. На цій стадії відбувається вибір необхідних компонентів, їх розміщення у робочому полі креслення, зв'язок компонентів за допомогою ланцюгів та шин. При необхідності можна модифікувати властивості компонентів,

додавати текстові написи. p align="justify"> Принципова електрична схема - це графічне відображення реального електричного пристрою за допомогою умовних графічних і буквено-цифрових позначень, а також зв'язків між елементами. На відміну від розведення друкованої плати принципова схема не показує фізичного розташування компонентів, лише вказує взаємне з'єднання висновків умовних графічних позначень реальних компонентів (наприклад, мікросхем) друкованої плати. При цьому допускається об'єднання групи ліній зв'язку в шини, проте необхідно чітко вказувати номери ліній, що входять до шини та виходять із неї. Зазвичай розробки радіоелектронного пристрою процес створення принципової схеми стає проміжним ланкою між стадіями розробки функціональної схеми і проектуванням друкованої плати. У ГОСТ 2.701-2008 принципову схему описано як «схему, що визначає повний склад елементів і зв'язків між ними і, як правило, дає детальне уявлення про принципи роботи виробу».

Складні електричні схеми можуть складатися із сотень елементів. І помилка, допущена під час проектування принципової схеми, неминуче повторюватиметься переважають у всіх наступних документах конструкторської документації. У результаті розробнику доведеться знову повертатися до читання принципової схеми, щоб виявити допущену помилку. Програми розробки електричних схем помітно полегшують працю інженера. ISIS – це редактор для створення на комп'ютері електричних схем та моделювання подій, що відбуваються у реальних електричних ланцюгах. За допомогою цієї програми можна зібрати елементарну схему та розробити досить складний проект. Складання електричних ланцюгів здійснюється у графічному режимі, що значно полегшує та прискорює побудову. Усі компоненти, з яких виконується складання схеми, зображені схематичними значками, що набагато спрощує їх пошук у бібліотеці компонентів. Після створення порожнього аркуша схеми його потрібно заповнити символами

необхідних компонентів із бібліотеки. У Proteus створити новий проект схеми можна за допомогою команди File/New Project. Слід зазначити, що за умовчанням під час створення нового проекту запускається майстер New Project Wizard (рис. 2). Робота майстра складається з декількох етапів, на яких вказуються назва проекту та його місце розташування на диску комп'ютера (можна створити проект з чистого аркуша або використовувати запропоновані системою шаблони), задається необхідність створення розробки ISIS (при цьому позначається формат креслення) та/або ARES та включення у

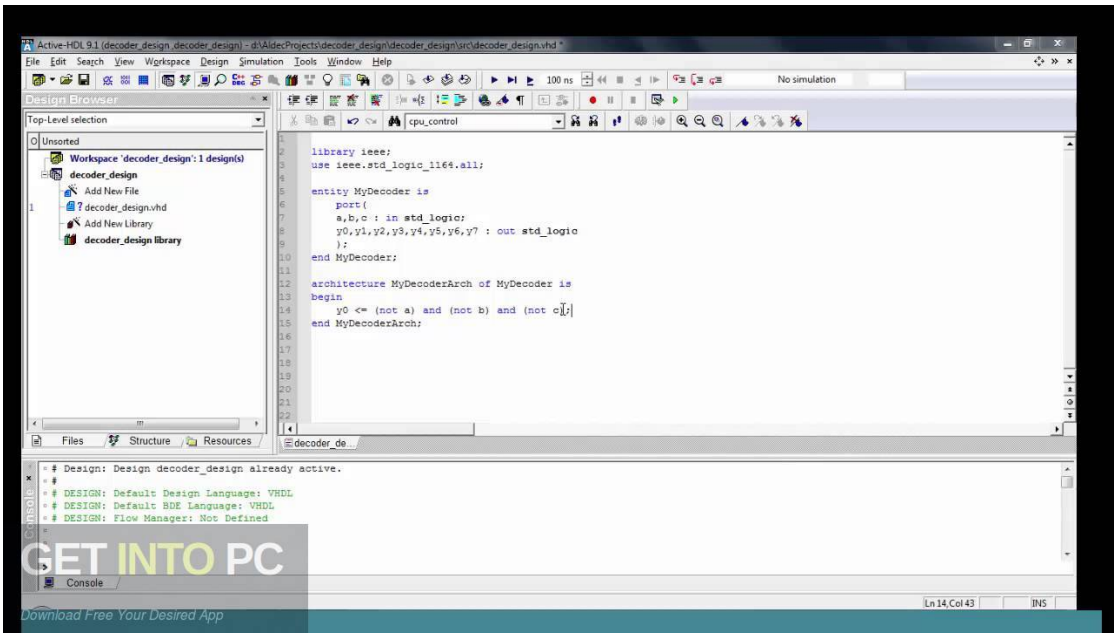


проект певного мікроконтролера. Після закінчення роботи майстра система, застосовуючи задані установки, сформує новий проект, який може містити робоче поле креслення (рис. 3а), контур друкованої плати (рис. 3б), заготівлю програмного коду мікроконтролера (рис. 3в). Слід зазначити, що під час одного сеансу роботи над проектуванням пристрою можуть бути паралельно відкриті проект схеми, проект плати, 3D-вид пристрою та редактор написання програмного коду (якщо у схемі використовується мікроконтролер). Причому для кожного редактора буде створено окрему вкладку.

ALDEC ACTIVE-HDL

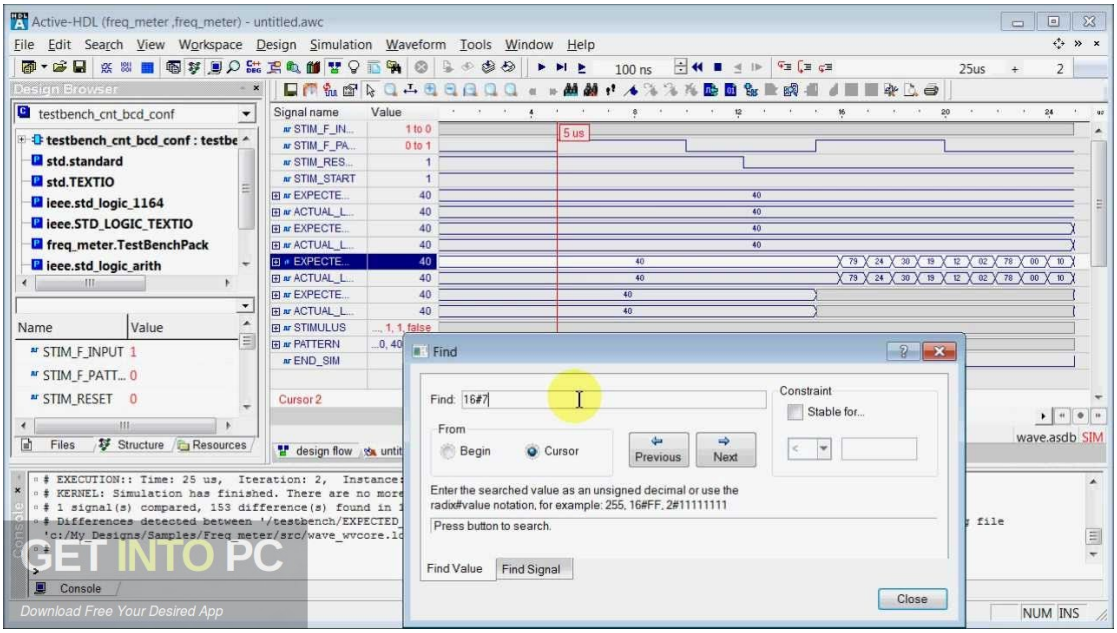
Active-HDL – це програмне забезпечення на базі Windows для створення, проектування та моделювання програмованих на місці клапанних масивів (FPGA) у командному середовищі. Інтегроване

середовище проектування Active-HDL надає повний набір графічних та багатомовних інструментів моделювання (з підтримкою листа праворуч наліво) для швидкої розробки, перевірки та модифікації проектів ПЛІС. Управління потоком програмного забезпечення у програмному забезпеченні вимагає понад 120 інструментів для створення автоматизації проектування електроніки (EDA) та ПЛІС при проектуванні, моделюванні, об'єднанні та реалізації потоків та дозволяє команді брати участь у всьому процесі розробки. FPGA, залишайтеся на спільній платформі. Active-HDL підтримує всі провідні в галузі ПЛІС, включаючи Altera, Atmel, Lattice, Microsemi, Quick Logic та Xilinx. Ви також можете завантажити CSiEDA



Процес перевірки проекту зазвичай включає багато стомлюючої роботи, а також хороше розуміння поведінки моделі. Однак, оскільки мова VHDL і Verilog гарантує, що будь-який опис моделі не залежить від постачальника і може бути замінений, зусилля щодо перевірки проекту можуть мати довгострокові результати. Оскільки людині властиво час від часу робити помилки, вам необхідно перевірити дизайн на наявність помилок. Active-HDL надає кілька механізмів для налагодження коду VHDL та Verilog: Active-HDL надає інтерактивний налагоджувач коду VHDL та Verilog. Після кожного виконання команди Compile у вікні

консолі з'являється список помилок. Кожна помилка з'являється з додатковою інформацією. Active-HDL може багато чого запропонувати як початківцям, і більш просунутим користувачам. Деякі розробники вважають за краще використовувати команди, що викликаються з меню, а деяким простіше використовувати комбінації клавіш.



Особливості ALDEC ACTIVE-HDL

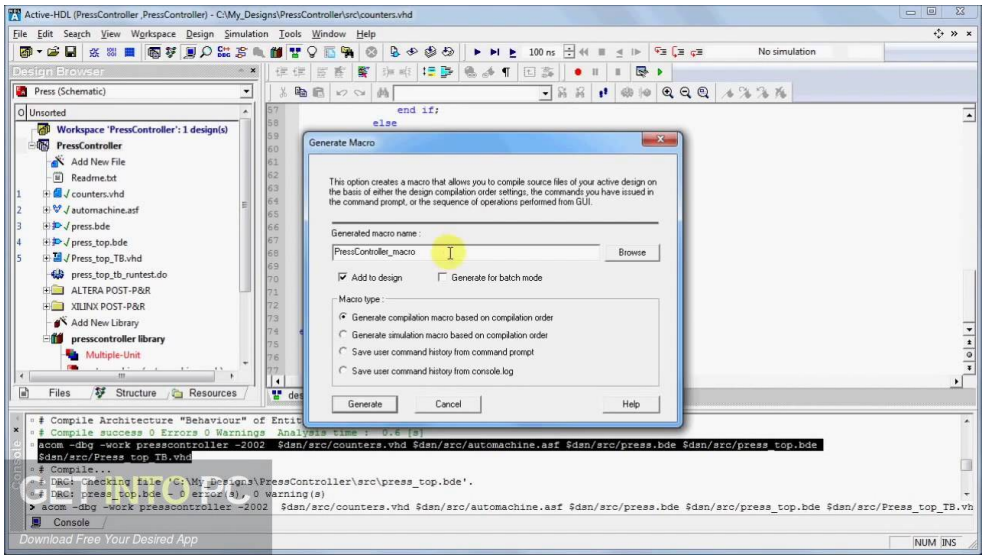
• Командне та єдине командне керування.

• Швидке розширення дизайнів текстами, діаграмами та режимами пристроїв

• IP-адреси більш безпечні та надійні завдяки послідовному та стандартному шифруванню.

• Потужний та багатомовний симулятор загального ядра з підтримкою VHDL, Verilog, System Verilog та System

• Забезпечення якості та достовірності коду за допомогою інтерактивних графічних засобів налагодження та контролю якості коду.



Технічні подробиці встановлення ALDEC ACTIVE-HDL

• Повне ім'я програмного забезпечення: ALDEC ACTIVE-HDL

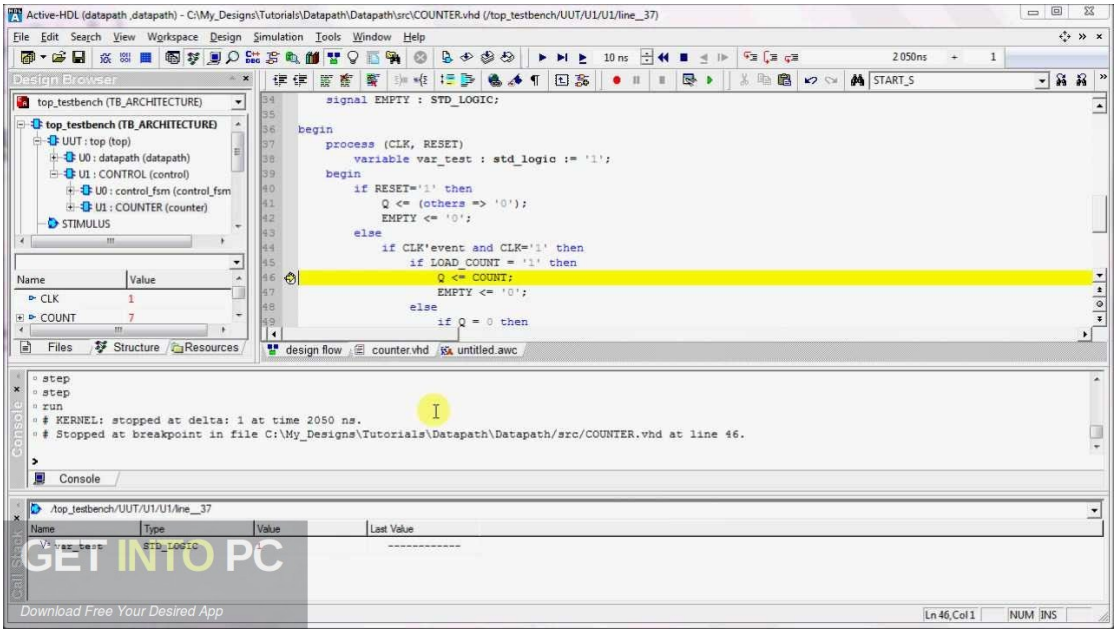
• Ім'я файлу встановлення: Aldec_Active-HDL_10.1.3088.5434.rar

• Повний розмір установки: 850 МБ

• Тип установки: автономний установник/повна автономна установка

• Архітектура сумісності: 32-розрядна (x86) / 64-розрядна (x64)

• Розробники: Aldec Active-HDL



Системні вимоги до ALDEC ACTIVE-HDL

• Операційна система: Windows 7/8/8.1/10

• Пам'ять: потрібно 2 ГБ ОЗУ.

• Місце на жорсткому диску: 500 МБ вільного місця.

• Процесор: двоядерний процесор Intel або новіша версія.

VHDL зазвичай використовується для написання текстових моделей, що описують логічну схему. Така модель обробляється програмою синтезу, якщо вона є частиною логічного проекту. Програма моделювання використовується для тестування логічного проекту за допомогою імітаційних моделей для представлення логіки схеми, які взаємодіють із проектом. Цей набір імітаційних моделей зазвичай називають тестовим стендом.

VHDL має конструкції для обробки паралелізму, властивого апаратним проектам, але ці конструкції (процеси) відрізняються за синтаксисом від паралельних конструкцій Ada (завдання). Як і Ada, VHDL має строго типізований і без урахування регістру. Для прямого представлення операцій, які є загальними для обладнання, існує безліч функцій VHDL, яких немає в Ada, наприклад розширений набір логічних операторів, включаючи nand та .

VHDL. має можливості введення та виведення файлів і може використовуватися як універсальна мова для обробки тексту, але файли частіше використовуються засобами моделювання для тестування стимулів або даних перевірки. Є кілька компіляторів VHDL, які створюють двійкові файли, що виконуються. У цьому випадку можна було б використовувати VHDL для написання тестового середовища для перевірки функціональності проекту з використанням файлів на головному комп'ютері для визначення стимулів, взаємодії з користувачем та порівняння результатів з очікуваними. Однак більшість дизайнерів залишають цю роботу симулятору.

Можна проектувати обладнання в VHDL IDE (для Реалізація FPGA, така як Xilinx ISE, Altera Quartus, Synopsys Synplify або Mentor

Graphics HDL Designer) для створення схеми RTL необхідної схеми. Після цього згенеровану схему можна перевірити за допомогою програмного забезпечення для моделювання, яке показує форми вхідних та вихідних сигналів схеми після створення відповідного випробувального стенду. Щоб згенерувати відповідний стенд для конкретної схеми або VHDL коду, вхідні дані повинні бути визначені правильно. Наприклад, для введення годинника потрібен процес циклу або ітеративний оператор.

Останнім моментом є те, що коли модель VHDL перетворюється на «заліро», яке відображається на програмованому логічному пристрої, такому як CPLD або FPGA, то це вже реальне обладнання, що конфігурується.

Переваги.

Ключовою перевагою VHDL, коли він використовується для проектування систем, є те, що дозволяє описувати (моделювати) і перевіряти (моделювати) поведінку необхідної системи до того, як інструменти синтезу переведуть конструкцію в справжню фурнітуру («залізо»).

Інша перевага полягає в тому, що VHDL дозволяє описувати паралельну систему. VHDL - це мова потоку даних, в якому кожен оператор розглядається для одночасного виконання, на відміну від мовних процедурних обчислень, таких як BASIC, C та асемблерний код, де послідовність операторів виконується послідовно за однією інструкцією за раз.

Проект VHDL багатоцільовий. Розрахунковий блок, що створюється одного разу, може використовуватися в багатьох інших проектах. Тим не менш, багато параметрів формальних та функціональних блоків можуть бути налаштовані (параметри ємності, розмір пам'яті, елементна база, склад блоку та структура взаємозв'язків).

Проект VHDL переносимо. Створений для однієї елементної бази проект обчислювального пристрою може бути перенесений на іншу елементну базу, наприклад НВІС з різними технологіями.

Великою перевагою VHDL порівняно з вихідним Verilog і те, що VHDL має повну систему типів . Дизайнери можуть використовувати систему типів для написання більш структурованого коду (особливо шляхом оголошення record типів). VHDL означає мову опису обладнання дуже високошвидкісної інтегральної схеми. Це мова програмування, яка використовується для моделювання цифрової системи за допомогою потоку даних, поведінкового та структурного стилю моделювання.

Опис дизайну.

У VHDL конфігурація використовується для опису апаратного модуля. Зміст може бути описан за допомогою:

- Декларація суб'єкта
- Архітектура
- Конфігурація
- Декларація на пакет
- Корпус упаковки

Декларація суб'єкта

Він визначає імена, вхідні вихідні сигнали та режими апаратного модуля.

Синтаксис.

```
entity entity_name is
  Port declaration;
end entity_name;
```

Оголошення конфігурації має починатися з "entity" і закінчуватися ключовими словами "end". Напрямок буде вхідним, вихідним або вхідним/вихідним .

I	Порт можна прочитати

n	
Out	Порт можна записувати
In out	Порт можна читати та записувати
Buffer	Порт можна читати і записувати, він може мати лише одне джерело.

Архітектура.

Архітектуру можна описати за допомогою структурного стилю, стилю потоку даних, поведінкового або змішаного стилю.

Синтаксис.

```
architecture architecture_name of entity_name
architecture_declarative_part;


begin
  Statements;
end architecture_name;
```

Тут необхідно вказати назву конфігурації, для якої пишимо тіло архітектури. Вирази архітектури мають бути всередині ключових слів 'begin' і 'énd'. Декларативна частина архітектури може містити змінні, константи або оголошення компонентів.

Моделювання процедури потоку даних.

У цьому стилі моделювання потік даних через об'єкт виражається за допомогою одночасного (паралельного) сигналу. Одночасними операторами у VHDL є WHEN і GENERATE.

Крім них, для побудови коду також можна використовувати присвоєння, що використовують лише оператори (AND, NOT, +, *, sll тощо).

Нарешті, особливий вид призначення, який називається BLOCK, також може бути використаний у цьому типі коду.

У паралельному коді можна використовувати наступне:

- Оператори
- Оператор WHEN (WHEN/ELSE або WITH/SELECT/WHEN);
- Оператор GENERATE;
- Оператор BLOCK

Моделювання процедури поведінки коду.

У цьому стилі моделювання набір операторів виконується послідовно у вказаному порядку. Лише оператори, розміщені всередині ПРОЦЕСУ, ФУНКЦІЇ або ПРОЦЕДУРИ, є послідовними.

ПРОЦЕСИ, ФУНКЦІЇ та ПРОЦЕДУРИ це єдині розділи коду, які виконуються послідовно.

Важливим аспектом коду поведінки є те, що він не обмежується послідовною логікою. За його можна будувати послідовні схеми, а також комбінаційні схеми.

Оператори поведінки: IF, WAIT, CASE і LOOP. ЗМІННІ також обмежені, і їх можна використовувати лише в послідовному коді. VARIABLE ніколи не може бути глобальною, тому її значення не можна передати безпосередньо.

Структурне моделювання.

У цьому моделюванні конфігурація описується як набір взаємопов’язаних компонентів. Оператор створення екземпляра компонента є паралельним оператором. Тому порядок цих тверджень не важливий. Структурний стиль моделювання описує лише взаємозв’язок компонентів (розглядаються як чорні скриньки), не маючи на увазі жодної поведінки самих компонентів або змісту, який вони разом представляють.

У структурному моделюванні тіло архітектури складається з двох частин: декларативної частини (перед ключовим словом begin) і операторної частини (після ключового слова begin).

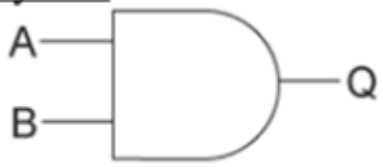
ПРИКЛАДИ ДИЗАЙНУ VHDL

У VHDL дизайн складається як мінімум з об'єкту, який описує інтерфейс, і архітектуру, яка містить фактичну реалізацію. Крім того, більшість проектів імпортують бібліотечні модулі. Деякі проекти також містять кілька архітектур та конфігурацій.

Логічна операція – AND (I)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

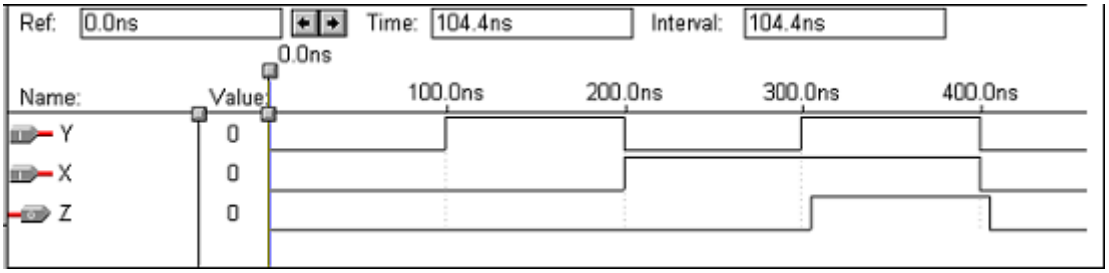
Symbol:



VHDL Code:

```
Library ieee;
use ieee.std_logic_1164.all;

entity and1 is
  port(x,y:in bit ; z:out bit);
end and1;

architecture virat of and1 is
begin
  z<=x and y;
end virat;
```

Осцилограми



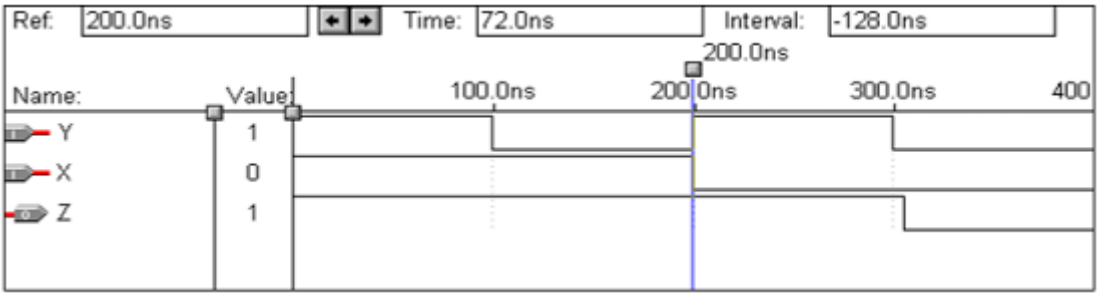
Логічна операція – OR (АБО)

X	Ю	З
0	0	0
0	1	1
1	0	1
1	1	1

Symbol:

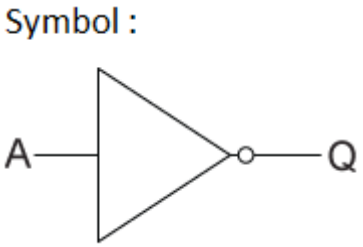

```
VHDL Code:
Library ieee;
use ieee.std_logic_1164.all;

entity or1 is
   port(x,y:in bit ; z:out bit);
end or1;

architecture virat of or1 is
begin
   z<=x or y;
end virat;
```

Осцилограми



Логічна операція – NOT (НЕ)

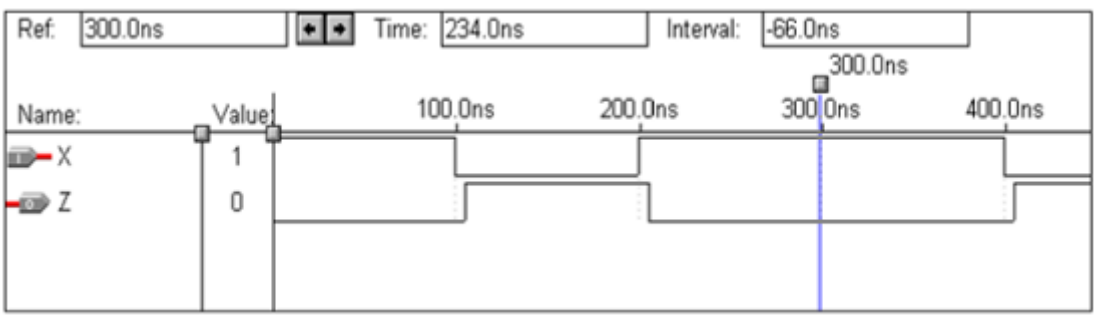
X	Ю
0	1
1	0



VHDL Code:

```
Library ieee;
use ieee.std_logic_1164.all;

entity not1 is
   port(x:in bit ; y:out bit);
end not1;

architecture virat of not1 is
begin
   y<=not x;
end virat;
```

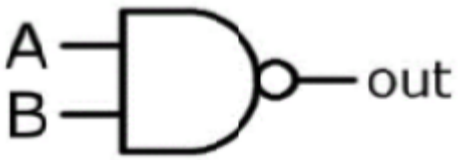
Осцилограми



Логічна операція – NAND (І-НЕ)

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

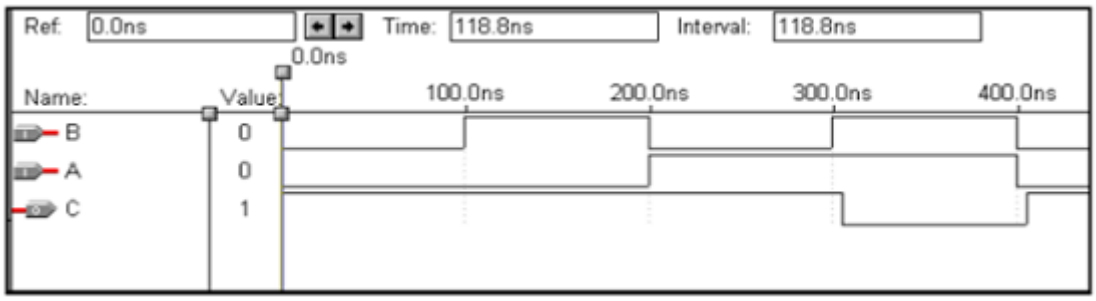
Symbol:



VHDL Code:

```
Library ieee;
use ieee.std_logic_1164.all;

entity nand1 is
  port(a,b:in bit ; c:out bit);
end nand1;

architecture virat of nand1 is
begin
  c<=a nand b;
end virat;
```

Осцилограми



Логічна операція – NOR (АБО-НЕ)

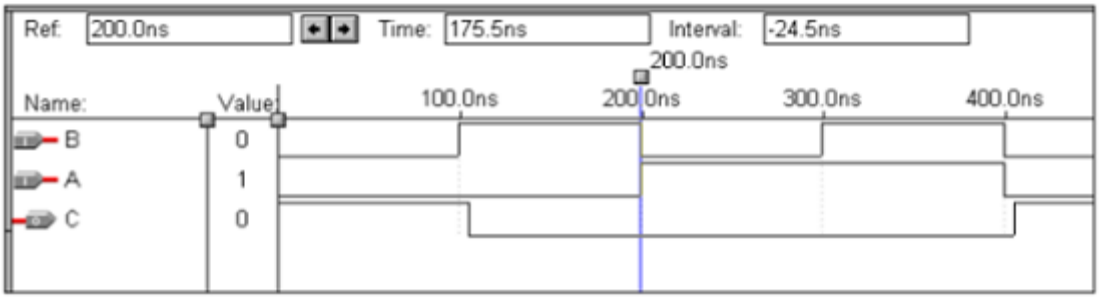
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Symbol:

VHDL Code:

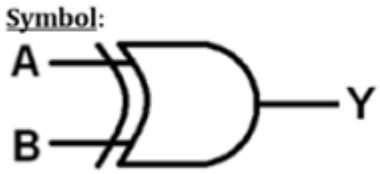
```
Library ieee;
use ieee.std_logic_1164.all;

entity nor1 is
  port(a,b:in bit ; c:out bit);
end nor1;

architecture virat of nor1 is
begin
  c<=a nor b;
end virat;
```

Осцилограми


Ref:
200.0ns
Time:
175.5ns
Interval:
-24.5ns
200.0ns
Name:
Value:
100.0ns
200.0ns
300.0ns
400.0ns
B
0
A
1
C
0

Логічна операція – XOR (Виключне АБО)

X	Ю	З
0	0	0
0	1	1
1	0	1
1	1	0

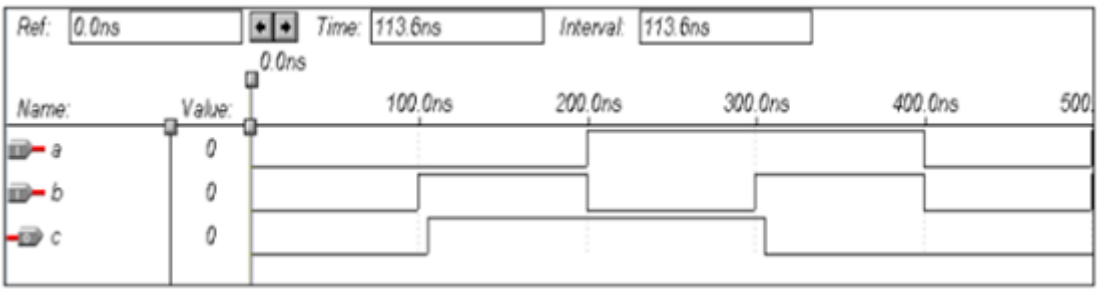


VHDL Code:

VHDL Code:

```
Library ieee;
use ieee.std_logic_1164.all;

entity xor1 is
  port(a,b:in bit ; c:out bit);
end xor1;

architecture virat of xor1 is
begin
  c<=a xor b;
end virat;
```

Осцилограми


Ref:
0.0ns
Time:
113.6ns
Interval:
113.6ns
0.0ns
Name:
Value:
100.0ns
200.0ns
300.0ns
400.0ns
500
a
0
b
0
c
0

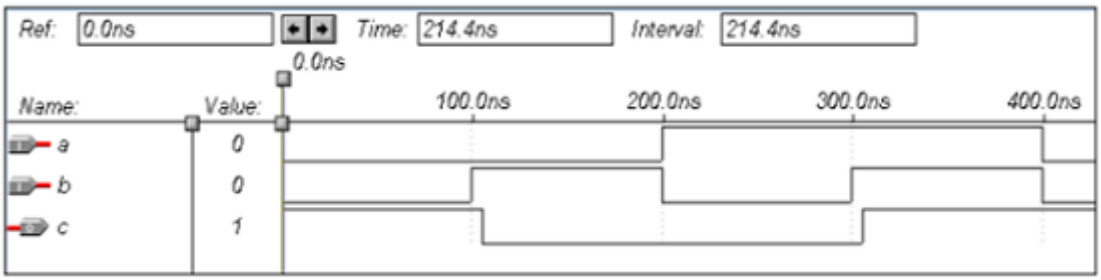
Логічна операція – X-NOR

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

Symbol:


```
Library ieee;
use ieee.std_logic_1164.all;

entity xnor1 is
   port(a,b:in bit ; c:out bit);
end xnor1;

architecture virat of xnor1 is
begin
   c<=not(a xor b);
end virat;
```

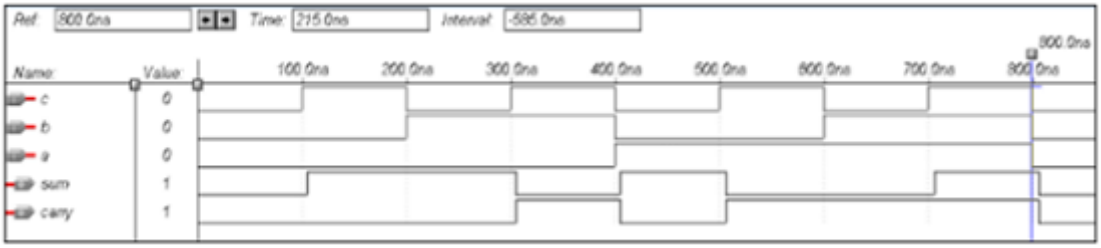
Осцилограми



Код VHDL для повного суматора

```
Library ieee;
use ieee.std_logic_1164.all;

entity full_adder is port(a,b,c:in bit; sum,carry:out bit);
end full_adder;

architecture data of full_adder is
begin
  sum<= a xor b xor c;
  carry <= ((a and b) or (b and c) or (a and c));
end data;
```

Осцилограми

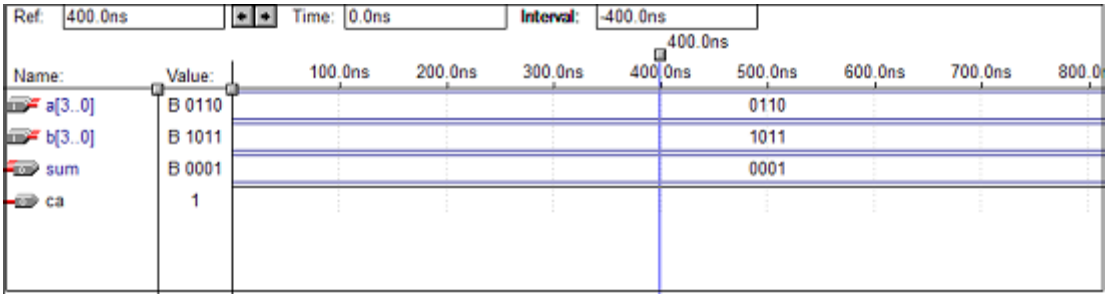


Код VHDL – 4-бітний паралельний суматор

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity pa is
  port(a : in STD_LOGIC_VECTOR(3 downto 0);
    b : in STD_LOGIC_VECTOR(3 downto 0);
    ca : out STD_LOGIC;
    sum : out STD_LOGIC_VECTOR(3 downto 0)
  );
end pa;

architecture vcgandhi of pa is
  Component fa is
    port (a : in STD_LOGIC;
      b : in STD_LOGIC;
      c : in STD_LOGIC;
      sum : out STD_LOGIC;
      ca : out STD_LOGIC
    );
  end component;
  signal s : std_logic_vector (2 downto 0);
```

```
  signal temp: std_logic;
begin
  temp<='0';
  u0 : fa port map (a(0),b(0),temp,sum(0),s(0));
  u1 : fa port map (a(1),b(1),s(0),sum(1),s(1));
  u2 : fa port map (a(2),b(2),s(1),sum(2),s(2));
  ue : fa port map (a(3),b(3),s(2),sum(3),ca);
end vcgandhi;
```

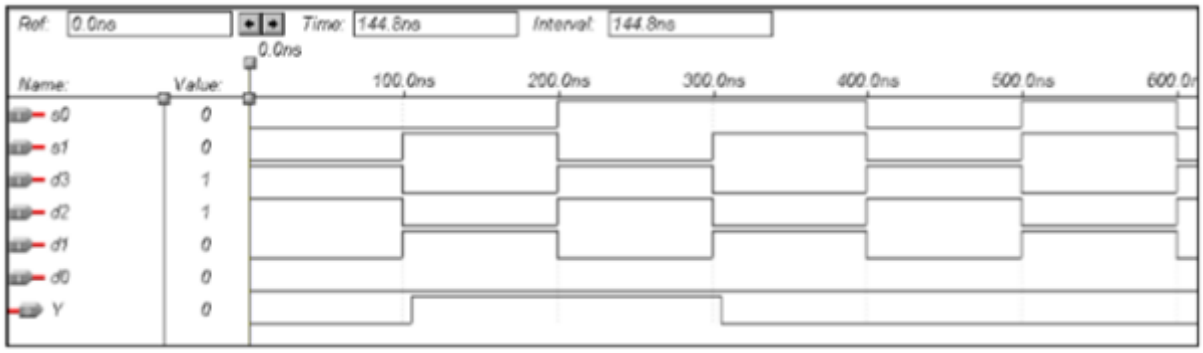
Осцилограми



Код VHDL для мультиплексора

```
Library ieee;
use ieee.std_logic_1164.all;

entity mux is
  port(S1,S0,D0,D1,D2,D3:in bit; Y:out bit);
end mux;

architecture data of mux is
begin
  Y<= (not S0 and not S1 and D0) or
    (S0 and not S1 and D1) or
    (not S0 and S1 and D2) or
    (S0 and S1 and D3);
end data;
```

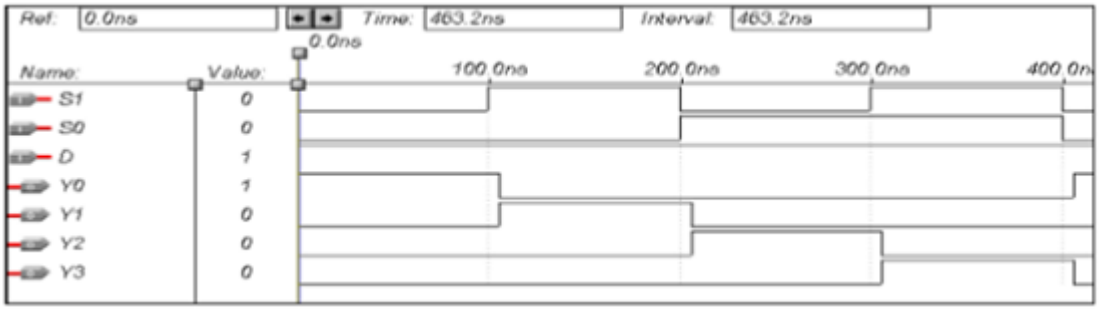
Осцилограми



Код VHDL для демультиплексора

```
Library ieee;
use ieee.std_logic_1164.all;

entity demux is
  port(S1,S0,D:in bit; Y0,Y1,Y2,Y3:out bit);
end demux;

architecture data of demux is
begin
  Y0<=  ((Not S0) and (Not S1) and D);
  Y1<=  ((Not S0) and S1 and D);
  Y2<=  (S0 and (Not S1) and D);
  Y3<=  (S0 and S1 and D);
end data;
```

Осцилограми

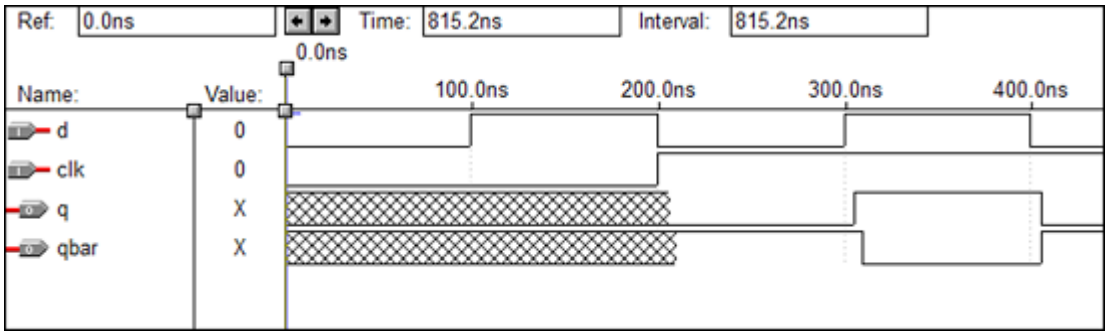


Код VHDL для D тригера

```
Library ieee;
use ieee.std_logic_1164.all;

entity dflip is
  port(d,clk:in bit; q,qbar:buffer bit);
end dflip;

architecture virat of dflip is
```

```
  signal d1,d2:bit;
begin
  d1<=d nand clk;
  d2<=(not d) nand clk;
  q<= d1 nand qbar;
  qbar<= d2 nand q;
end virat;
```

Осцилограми

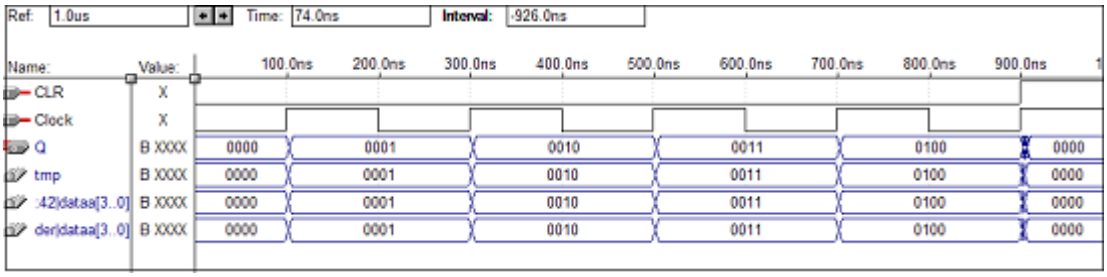


Код VHDL для 4-розрядного лічильника

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
  port(Clock, CLR : in std_logic;
    Q : out std_logic_vector(3 downto 0)
  );
end counter;

architecture virat of counter is
  signal tmp: std_logic_vector(3 downto 0);
begin
  process (Clock, CLR)

  begin
    if (CLR = '1') then
      tmp < = "0000";
```

```
    elsif (Clock'event and Clock = '1') then
      mp <= tmp + 1;
    end if;
  end process;
  Q <= tmp;
end virat;
```

Осцилограми

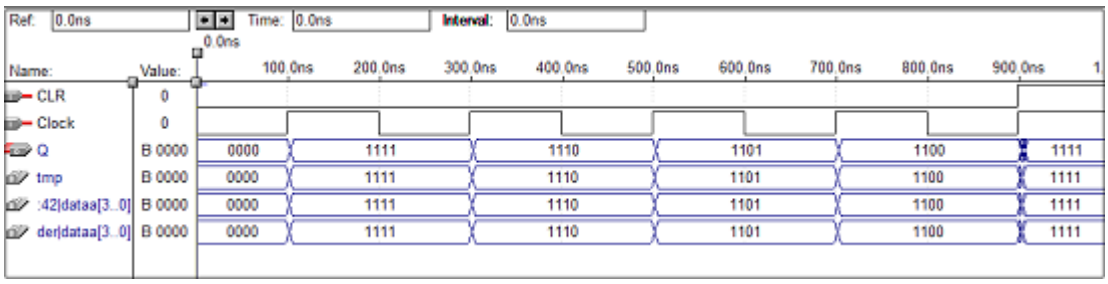


Код VHDL для 4-бітного лічильника, що віднімає

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity dcounter is
  port(Clock, CLR : in std_logic;
    Q : out std_logic_vector(3 downto 0));
end dcounter;

architecture virat of dcounter is
  signal tmp: std_logic_vector(3 downto 0);

begin
  process (Clock, CLR)
  begin
    if (CLR = '1') then
      tmp <= "1111";
    elsif (Clock'event and Clock = '1') then
```

```
            tmp <= tmp - 1;
        end if;
    end process;
    Q <= tmp;
end virat;
```

Осцилограми



2. ОСНОВНІ ТИПИ СХЕМ

Для зображення електронних пристроїв і їхніх вузлів застосовується три основних типи схем:

- принципова схема;
- структурна схема;
- функціональна схема.

Розрізняються вони своїм призначенням і ступенем деталізації зображення пристроїв.

Принципова схема – найбільш докладна. Вона відображає всі використані в пристрої елементи й всі зв'язки між ними. Якщо схема будується на основі мікросхем, то повинні бути показані номери виводів всіх входів і виходів цих мікросхем. Принципова схема повинна дозволяти повністю відтворити пристрій. Позначення принципової схеми найбільш жорстко стандартизовані, відхилення від стандартів не рекомендуються.

Структурна схема – найменш докладна. Вона призначена для відображення загальної структури пристрою, тобто його основних блоків, вузлів, частин і головних зв'язків між ними. Зі структурної схеми повинно бути зрозуміло, навіщо потрібний даний пристрій і що він робить в основних режимах роботи, як взаємодіють його частини. Позначення структурної схеми можуть бути досить довільними, хоча загальноприйняті правила потрібно виконувати.

Функціональна схема являє собою гібрид структурної й принципової. Деякі найбільш прості блоки, вузли, частини пристрою відображаються на ній, як на структурній схемі, а інші – як на принциповій схемі. Функціональна схема дає можливість зрозуміти всю логіку роботи пристрою, всієї його відмінності від інших подібних пристроїв, але не дозволяє без додаткової самостійної роботи відтворити цей пристрій. Що стосується позначень, використовуваних на функціональних схемах, то в частині, показаної як структура, вони не

стандартизовані, а в частині, показаної як принципова схема, – стандартизовані.

У технічній документації обов'язково наводяться структурна або функціональна схема, а також обов'язково принципова схема.

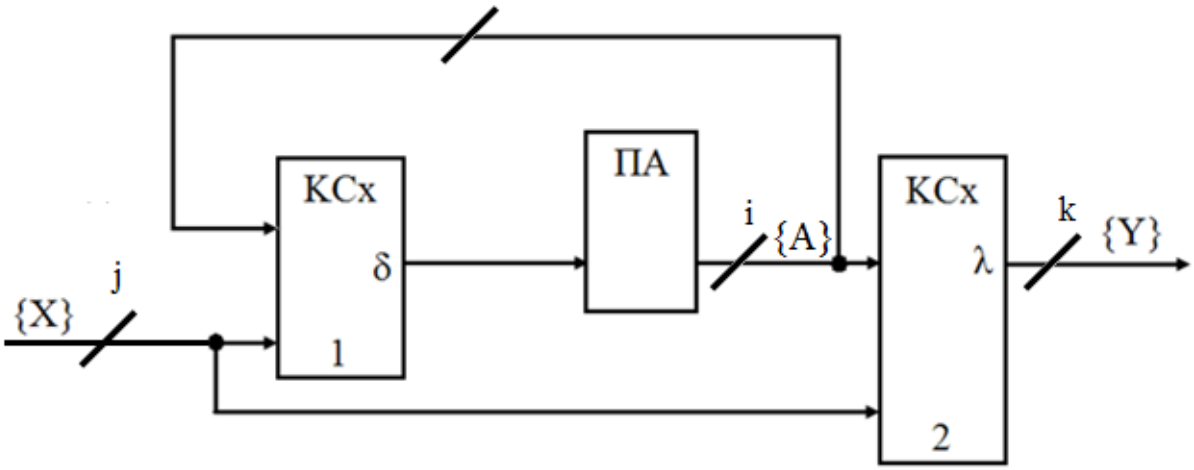
3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис автомата Мілі

Автомат Мілі — скінчений автомат, чиї вихідні символи визначаються його станом, та символами на вході (на відміну від автомату Мура, вихідні символи якого визначаються тільки його станом). На ребрах в діаграмі станів позначають вхідні та вихідні символи (а в автоматі Мура вихідні символи позначають на вершинах).

Автомат Мілі названий на честь Джорджа Мілі, який представив ідею в роботі 1955 року, «A Method for Synthesizing Sequential Circuits.»

Автомат Мілі може бути примітивною математичною моделлю шифрувальної машини. Якщо взяти за вхідний та вихідний алфавіти наприклад символи латинки, то можна сконструювати автомат Мілі, який буде для кожного вхідного рядка давати на виході зашифровану послідовність. Тим не менш, хоча його й можна використати для опису наприклад шифрувальної машини Енігма, діаграма станів буде занадто складною для конструювання відповідного автомата. На рис. 1 показана загальна структура автомата Мілі.



КСх - комбінаційна схема
ПА - пам'ять автомата
δ - функція входів
λ - функція виходів
j - кількість вхідних сигналів
k - кількість вихідних сигналів

{X} - множення вхідних сигналів
{Y} - множення вихідних сигналів
{A} - множення внутрішніх сигналів
i - розрядність зворотнього звязку кількість тригерів у памяті автомата

Рис. 3.1.1 Загальна структурна схема автомата Мілі

Автомат Мілі це шестірка, (S, S0, X, Y, T, G), що складається з:

1. скінченної множини станів (S)
2. початкового (ініціального) стану S0 який є елементом (S)
3. вхідного алфавіту (X)
4. вихідного алфавіту (Y)
5. функції переходів (T : S × X → S) що відображує пару стан, вхідний символ, на інший стан в який здійснюється перехід за цим символом.
6. функції виходів (G : S × X → Y), яка відображає пари стан, вхідний символ, у вихідні символи.

За способом формування функції виходів виділяють три типи абстрактних автоматів: автомат Мілі, автомат Мура та С-автомат.

В абстрактному автоматі Мілі значення функції виходу в момент t залежить не лише від стану автомата, але і від набору значень вхідних сигналів.

Довільний абстрактний автомат Мілі має один вхідний і один вихідний канали.

Таблиця 1. Двійкового кодування станів автомата

№	**Стан автомата**				
	позначення	Код			
	a_i	Q3	Q2	Q1	Q0
0	a_0	0	0	0	0
1	a_1	0	1	0	1
2	a_2	1	0	1	0
…		…	…	…	…
8	a_8	1	0	0	0
9	a_9	1	0	0	1

Автомат Мілі характеризується системою рівнянь:

$$\begin{cases} y(t) = \lambda[s(t), x(t)]; \\ \varphi \end{cases} \quad (1)$$

де $X = \{x_1, x_2, \ldots, x_m\}$ – множина вхідних сигналів автомата (вхідний алфавіт);

$S = \{s_1, s_2, \ldots, s_n\}$ – множина станів автомата (алфавіт станів);

$Y = \{y_1, y_2, \ldots, y_l\}$ – множина вихідних сигналів (вихідний алфавіт).

λ – функція виходів автомата;

φ – функція переходів автомата.

Іншими словами, функція виходів λ задає відображення (X × S)→Y, тобто ставить у відповідність будь-якій парі елементів декартового добутку множин (X × S) елемент множини S.

3.2. Кодування граф-схеми автомата (ГСА)

В автоматі Мілі початок і кінець мікропрограми представляється початковим станом автомата A0. Якщо декілька дуг, позначені певними станами Ai, входить до одного блоку графа мікропрограми, то всі вони помічаються одинаковим символом стану Ai.

ГСА - це орієнтований зв'язковий граф, що задає послідовність виконання операцій даного алгоритму і містить ряд операторних та умовних вершин, а також одну початкову і кінцеву вершини. Операторною називається вершина, якій ставиться у відповідність одна або кілька мікрооперацій і відзначається відповідними керуючими сигналами У, а умовною – вершина, якій ставиться у відповідність деяка логічна умова X. Будь-яка вершина ГСА, крім вершини «Початок», має по одному входу. Вершина "Початок" входів не має. Вершина «Початок» та будь-яка операторна вершина мають по одному виходу. Вершина "Кінець" виходів не має. Будь-яка умовна вершина має два виходи, що позначаються символами «Так» та «Ні»: Замість цих символів можуть

бути використані цифри «1» та «0» відповідно. Зображення вершин «Початок», «Кінець», операторної вершини та умовної вершини ГСА складають так, щоб забезпечити виконання необхідних операцій та перевірку логічних умов відповідно до словесного опису алгоритму. На підставі переліку мікрооперацій та функціональних вузлів, що їх реалізують, складається структурна схема ОА. На ній широкими стрілками показують шини, за якими передається інформація, а тонкими - сигнали, що управляють роботою окремих вузлів або передачею інформації по шинах. ГСА повинна задовольняти такі умови:

1. Входи та виходи вершин з'єднуються один з одним за допомогою спрямованих завжди від виходу до входу.
2. Кожен вихід з'єднаний лише з одним входом.
3. Будь-який вхід з'єднується принаймні з одним виходом.
4. Будь-яка вершина ГСА лежить принаймні на одному шляху з вершини «Початок» у вершину «Кінець».

Один із виходів умовної вершини може з'єднуватися з її входом, що є неприпустимим для операторної вершини. Такі умовні вершини іноді називають поворотними.

У кожній умовній вершині записується логічна умова з багатьох логічних умов. Дозволяється у різних умовних вершинах записувати однакові логічні умови.

У кожній операторній вершині записується оператор, що є вихідним сигналом або сукупністю вихідних сигналів управляючого автомата. Дозволяється в різних операторних вершинах записувати однакові оператори.

Мікрокомандою називають групу операторів, що формують на виходах автомата сигнали ініціації зовнішніх пристроїв з урахуванням значень умовних операторів на його входах. Кожна мікрокоманда має поля: К2 розрядів умовних операторів, К1-розрядного коду поточної

мікрокоманди, К1-розрядного коду наступної мікрокоманди та (К4+К6) виходів сигналів ініціації зовнішніх пристроїв. Усі оператори мікрокоманди виконуються протягом лише одного мікротакту. Мікротактом називають період між сусідніми синхронізуючими імпульсами тактового генератора.

Мікропрограмою називають послідовність мікрокоманд, що виконують закінчену дію над зовнішнім пристроєм.

3.3. Побудова таблиці переходів

Умови переходу по мікропрограмі від одного стану до іншого задають функцію переходів автомата.

Таблицю переходів (виходів) являє собою таблицю з подвійним входом, рядки якого пронумеровані вхідними буквами, а стовпці – станами. На перетині вказується стан, у який переходить автомат (в таблиці переходів) або вихідний сигнал, що видається ним (у таблиці виходів).

Іноді при заданні автоматів Мілі використовують одну суміщену таблицю переходів і виходів, в якій на перетині стовпця am і рядка xj записуються у вигляді as / yg наступний стан і вихідний сигнал, що видається.

3.4. Синтез керуючого автомата

Керуючі пристрої складаються із окремих логічних схем елементів, які виробляють керуючі сигнали в заданій послідовності. Такий керуючий пристрій можна розглядати як керуючий автомат типу Мура чи Мілі.

Для автомату Мілі вихідний сигнал залежить не лише від внутрішнього стану, а й від зовнішнього стану схеми. Можна побудувати граф переходів автомата Мура, де вершинами являються стани автомата, а дугами умови переходу з одного стану в інший.

Автомати можна задати також у вигляді графів, таблиць виходів та переходів, суміщеної таблиці переходів і виходів. Управляючий пристрій

складається із окремих логічних схем, що створюють управляючі сигнали в заданій послідовності. Такий управляючий пристрій можна розглядати як керуючий автомат типу Мура чи Мілі.

Після побудови автомата Мілі функціонування керуючого автомата представляють у вигляді таблиць переходів і виходів. Для цього спочатку виробляють кодування станів автомата двійковими кодами, визначають тип та кількість тригерів. Потім по таблиці переходів визначають функції збудження тригерів і виконують їх мінімізацію (спрощення). По знайдених виразах будується схема управляючого автомата на вибраних елементах.

Мікропрограмні автомати є наступний крок по шляху ускладнення інтелекту цифрових схем. На основі мікропрограмних автоматів можна будувати пристрої, які працюють за досить складних алгоритмах, виконують різні функції, які визначаються вхідними сигналами, видають складні послідовності вихідних сигналів. При цьому алгоритм роботи мікропрограмного автомата може бути легко змінений заміною прошивки ПЗП.

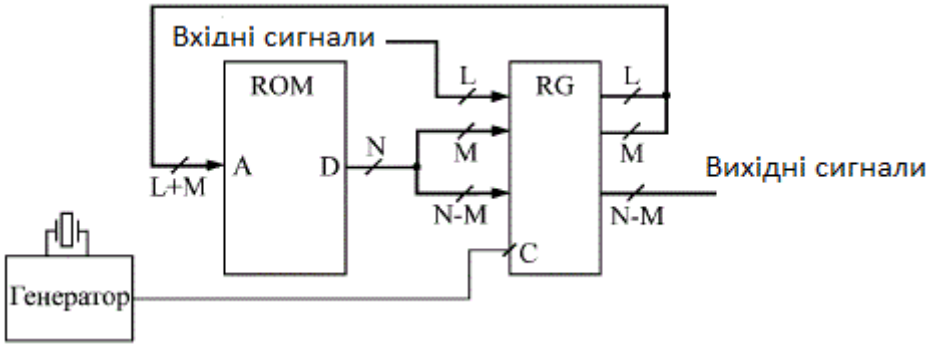


Рис. 3.1.2. Структура мікропрограмного автомата

На відміну від розглядалися раніше пристроїв на "жорсткої" логіки, принцип роботи яких однозначно визначається використовуваними елементами і способом їх з'єднання, мікропрограмні автомати за допомогою однієї і тієї ж схеми можуть виконувати найрізноманітніші функції. Тобто вони набагато гнучкіші, ніж схеми на "жорсткої" логіки. До того ж проектувати мікропрограмні автомати з точки зору схемотехніки досить просто. Недоліком будь-якого

мікропрограмного автомата в порівнянні зі схемами на "жорсткої" логіки є

менше граничне швидкодію і необхідність складання карти прошивки ПЗП з мікропрограмами, часто досить складними.

Найбільш поширена структура мікропрограмного автомата (рис. 3.1.2) включає в себе всього лише три елементи: ПЗП, регістр, що спрацьовує по фронту, і тактовий генератор.

ПЗП має (L + M) адресних розрядів і N розрядів даних. Регістр застосовується з кількістю розрядів (N + L). Розряди даних ПЗП записуються в регістр по позитивному фронту тактового сигналу з генератора. Частина цих розрядів (M) використовується для освіти адреси ПЗП, інша частина (N-M) служить для формування вихідних сигналів. Вхідні сигнали (L) надходять на входи регістра і використовуються спільно з частиною вихідних розрядів ПЗП для отримання адреси ПЗП.

Схема працює в такий спосіб. У кожному такті ПЗП видає код даних, тим самим визначаючи не тільки стан вихідних сигналів схеми, а й адресу ПЗП, який встановиться в наступному такті (після наступного позитивного фронту тактового сигналу). На цей наступну адресу впливають також і вхідні сигнали. Тобто на відміну від формувача послідовності сигналів, розглянутого в попередньому розділі, в даному випадку адреси можуть перебиратися не тільки послідовно (за допомогою лічильника), а й в довільному порядку, який визначається прошивкою ПЗП, званої мікропрограмою.

Умовою правильної роботи схеми буде наступне. За один період тактового сигналу повинні встигнути спрацювати регістр і ПЗП. Інакше кажучи, сума затримки регістра і затримки вибірки адреси ПЗП не повинна перевищувати періоду тактового сигналу. Відзначимо також, що вхідні сигнали мікропрограмного автомата можна подавати безпосередньо на адресні входи ПЗП (без регістра), так як їх асинхронне (по відношенню до тактовою сигналу) зміна може викликати перехідний

процес на виходах даних ПЗП саме в той момент, коли вихідні сигнали ПЗП записуються в регістр. В результаті в регістр може записатися невірна інформація, що порушить роботу всієї схеми. Регістр ж синхронізує зміни вхідних сигналів з тактовим сигналом, в результаті чого всі розряди коду адреси ПЗП змінюється одночасно - по позитивному фронту тактового сигналу. Регістр повинен обов'язково спрацьовувати по фронту, використання регістра-засувки не допускається, так як він може викликати лавиноподібний перехідний процес.

Можливості такої простої схеми виявляються дуже великими. Наприклад, мікропрограмний автомат може видавати послідовності вихідних сигналів у відповідь на певну зміну вхідних сигналів. Він може також тимчасово зупинити видачу вихідних сигналів до приходу вхідних сигналів. Він може аналізувати тривалість вхідного сигналу і в залежності від неї видавати ті чи інші вихідні сигнали. Він може також і багато іншого.

3.5. Опис компонентів

3.5.1. Лічильник (CT)

Лічильник — пристрій для підрахунку кількості сигналів, які надходять на його вхід. Двійкові лічильники реалізують лічбу вхідних імпульсів у двійковій системі числення.

У двійковому підсумовуючому лічильнику перенесення Pi в сусідній старший розряд Qi+1 виникає в тому випадку, коли в момент надходження чергового лічильного імпульсу U+ всі молодші розряди мають високий рівень, тобто Pi=U+QiQi–1…Q1=1. Після вироблення перенесення старший розряд перемикається на високий рівень, а всі молодші розряди — на низький рівень.

До синхронних лічильників відносяться лічильники, в яких переключення розрядів відбувається одночасно. Це досягається подаванням на всі тригери синхронізуючих імпульсів, які додатнім або від’ємним перепадом викликають переключення тригерів у відповідності із логікою роботи лічильника. Завдяки такій синхронізації досягається мінімальний час встановлення лічильника, який не перевищує час встановлення одного тригера, чим забезпечується максимальна частота зміни станів лічильника.

Лічильник DM74LS193

Мікросхема DM74LS193 - це синхронний 4-бітний бінарний лічильник інкрементації/декрементації. Синхронна робота забезпечується одночасним синхронізацією всіх тригерів, так що виходи обмінюються разом, коли це вказує логіка рульового управління. Цей режим роботи виключає вихідні лічильники, що зазвичай асоціюються з асинхронними лічильниками.

У нормальному режимі роботи на вхід LOAD подається напруга високого рівня, а на вхід скидання CLEAR - низького рівня.

Значення, що зберігається в лічильнику, послідовно збільшується на 1 при кожному перепаді напруги на вході прямого рахунку тактових

імпульсів UP з низького рівня на високий (позитивний фронт). Кожен позитивний фронт тактового імпульсу на вході зворотного рахунку DOWN зменшує показання лічильника. У будь-якому випадку на один з двох входів тактових імпульсів має

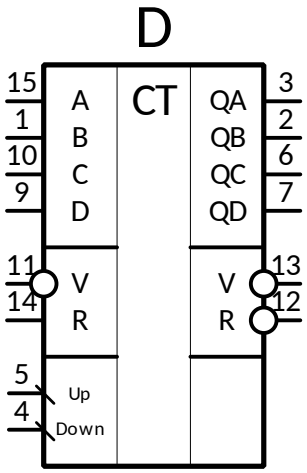
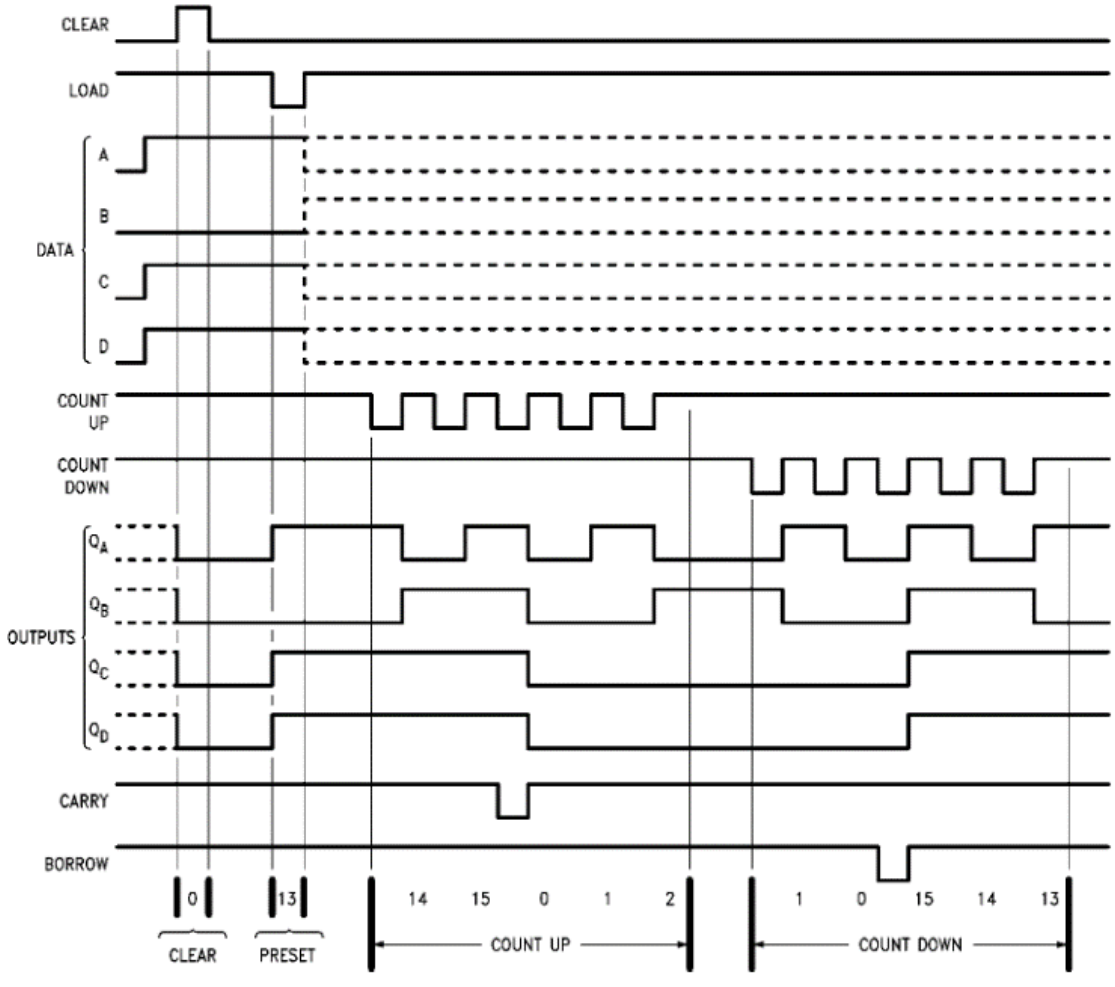


Рис. 3.5.1.1. Умовне графічне позначення лічильника DM74LS193

Timing Diagram



Note A: Clear overrides load, data, and count inputs

Note B: When counting up, count-down input must be HIGH; when counting down, count-up input must be HIGH.

Рис. 3.5.1.2. Часові діаграми вхідних та вихідних сигналів лічильника DM74LS193

подаватися напруга високого рівня.

При програмуванні необхідні дані в двійковій коді подаються на входи P0 - P3 мікросхеми 74193, а на вхід LOAD - короткочасний імпульс напруги низького рівня.

Щоб асинхронно скинути лічильник, на вхід CLEAR (очищення) подається короткочасний імпульс напруги високого рівня.

Таблиця. 3.5.1.1. Таблиця істинності лічильника DM74LS193

UP	DOWN	CLEAR	LOAD	Функція
↑	1	0	1	Додавання
1	↑	0	1	Віднімання
X	X	1	X	Скидання
X	X	0	0	Загрузка

Лічильник 74LS163

Мікросхема 74LS163 - це 4-розрядний двійковий синхронний прямий лічильник. В якому вхід MR використовується для скидання лічильника в 0, а вхід LOAD – для попереднього запису в лічильник інформації, що надходить з входів D0, D1, D2, D3.

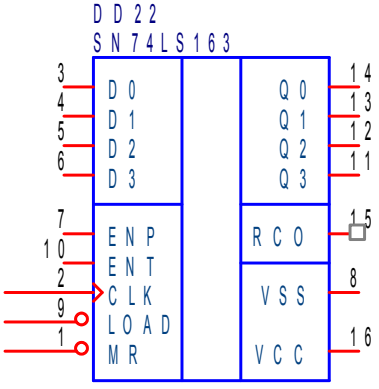


Рис. 3.5.1.3 Умовне графічне позначення лічильника SN74LS163

Таблиця 3.5.1.2. Таблиця істинності лічильника

Стан мікросхеми 74LS163				
MR	ENP	ENT	LOAD	Зростаючий імпульс
H	X	X	X	Скид лічильника
B	X	X	H	Завантаження
B	B	B	B	Лічба
B	X	H	B	Без змін
B	H	X	B	Без змін

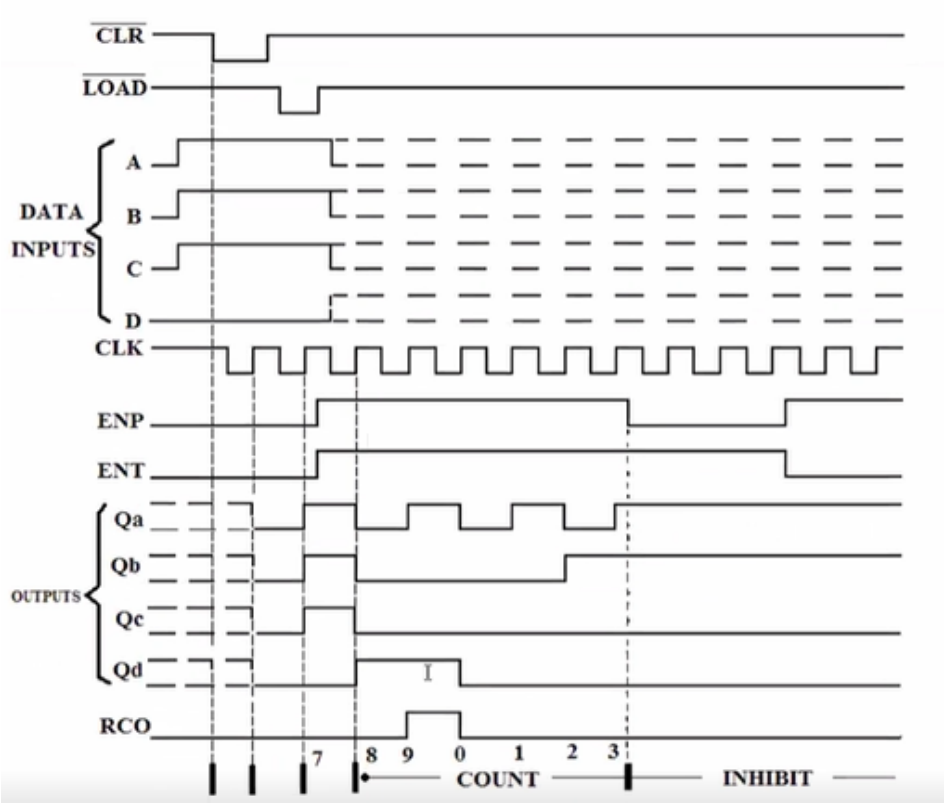


Рис. 3.5.1.4. Часова діаграма роботи лічильника

Скидання лічильника в 0 відбувається при подаванні лог.0 на вхід CLR, при цьому на вході LD, повинна бути лог.1. Максимальний коефіцієнт перерахунку 74LS163 складає 16.

Лічильник 74LS169

Мікросхема 74LS169 - це 4-розрядний двійковий синхронний реверсивний лічильник. В якому вхід MR використовується для скидання лічильника в 0, а вхід LD – для попереднього запису в лічильник інформації, що надходить з входів D0, D1, D2, D3.

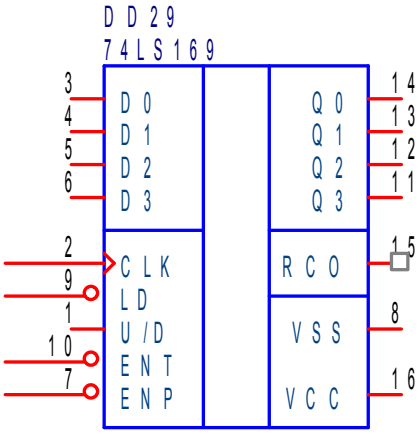


Рис. 3.5.1.5. Умовне графічне позначення лічильника SN74LS169

Таблиця 3.5.1.3. Таблиця істинності лічильника

PE	CEP	CET	U/D	Action on Rising Clock Edge
L	X	X	X	Load (Pn Qn)
H	L	L	H	Count Up (increment)
H	L	L	L	Count Down (decrement)
H	H	X	X	No Change (Hold)
H	X	H	X	No Change (Hold)

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

3.5.2. Оперативно запам’ятовуючий пристрій (RAM)

Операти́вна па́м'ять (англ. Random Access Memory, дослівно — пам'ять з довільним доступом, ПДД, первинна пам'ять) — пам'ять ЕОМ, призначена для зберігання коду та даних програм під час їхнього виконання. В неї можна записати інформацію скільки завгодно разів. Інформація в пам’яті пропадає після включення її живлення

Основна відмінність оперативної пам'яті (RAM) від постійної (ROM) полягає в можливості оперативної зміни вмісту всіх елементів пам'яті за допомогою додаткового управляючого сигналу запису WR. Кожний елемент оперативної (статичної) пам'яті є, по суті, регістром з комірок тригерів, в який може бути записана інформація і з якого можна інформацію читати. Вибір того або іншого елемента пам'яті проводиться за допомогою коду адреси пам'яті. Тому при виключенні живлення вся

інформація з оперативної пам'яті пропадає (стирається), а при включенні живлення інформація в оперативній пам'яті може бути довільною.

Оперативні запам’ятовуючі пристрої (ОЗП) поділитися на дві великі групи: статичні і динамічні. В накопичувачах статичних ОЗП застосовуються тригерні елементи пам’яті. В ОЗП динамічного типу запам’ятовуючим елементом служить конденсатор, в якому інформація зберігається в форматі наявності або відсутності заряду.

Перевагою статичних ОЗП перед динамічними є відсутність схеми регенерації інформації, що значно спрощує керування. Крім того, швидкодія статичних ОЗП суттєво перевищує швидкодію динамічних ОЗП. З іншого боку, ємність мікросхем динамічних ОЗП набагато вища ніж статичних.

ОЗП IDT6116SA(RAM).

Мікросхема IDT6116SA(RAM) використовується для запису, зберігання та читання 8-розрядних слів.

Особливості:

◆ Високошвидкісний доступ і час вибору чіпа – Військовий: 20/25/35/45/55/70/90/120/150 нс (макс.) – Промисловий: 20/25 нс (макс.) – Комерційний : 15/20/25 нс (макс.)

◆ Низьке енергоспоживання

◆ Резервне живлення від батареї – напруга збереження даних 2 В (лише для версії LA)

◆ Виготовлено за допомогою вдосконаленої високопродуктивної технології CMOS

◆ Процес CMOS практично усуває частоту м’яких помилок альфа-частинок

◆ Вхід і вихід напряму сумісні з TTL

◆ Статична робота: не потрібні годинник або оновлення

◆ Доступно 4 керамічних pin DIP, керамічний і пластиковий 24-контактний Thin Dip і 24-контактний SOIC

◆ Військовий продукт, сумісний з MIL-STD-833, клас B

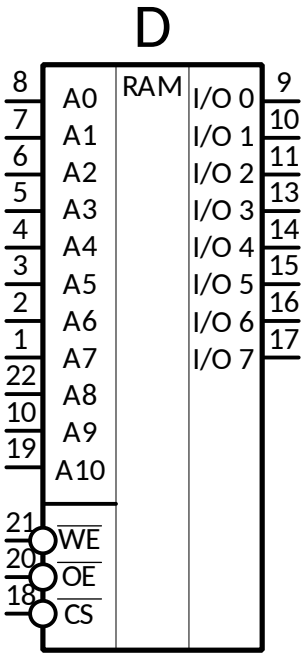


Рис. 3.5.2.1. Умовне графічне позначення ОЗП IDT6116SA

Functional Block Diagram

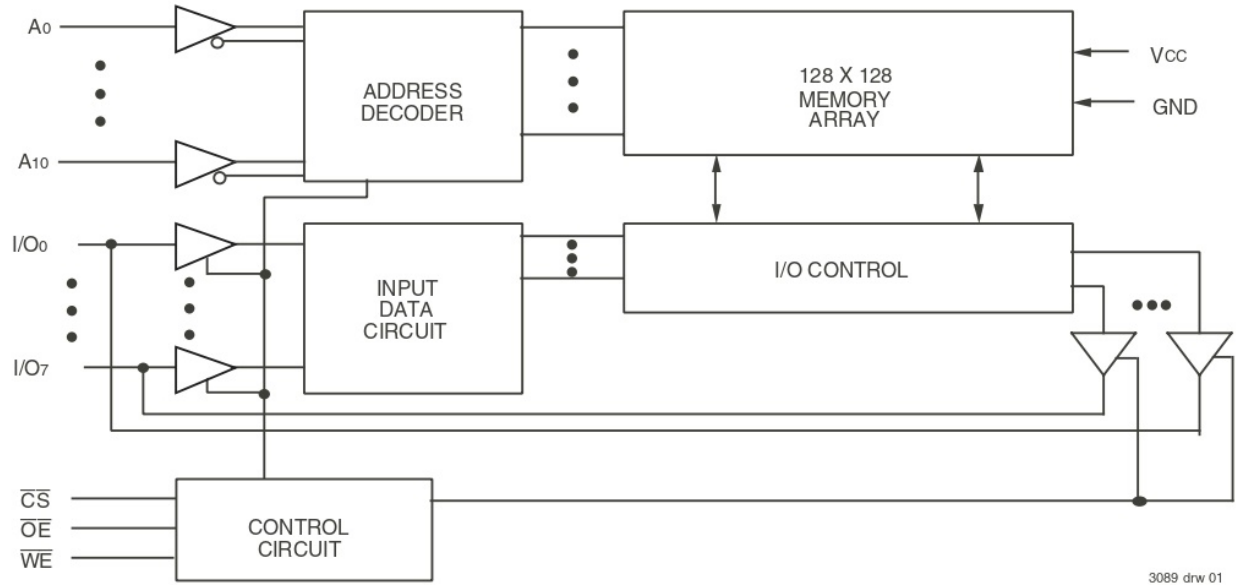


Рис. 3.5.2.2. Функціональна схема ОЗП IDT6116SA

Пам'ять категорії => SRAM => Async. SRAM => 16 Кб

Опис CMOS Static RAM 16k : 2kx8

Компанія Integrated Device Technology, Inc.Part IDT6116SA

Входи та виходи мікросхема IDT6116SA :

- A0…A10 – Адресні входи;

- WE – Вхід дозволу запису та зчитування;
- OE – Вхід управління станом вхідного буфера;
- CE – Вхід вибору кристалу;
- I/O0…I/O7 – Входи/виходи даних

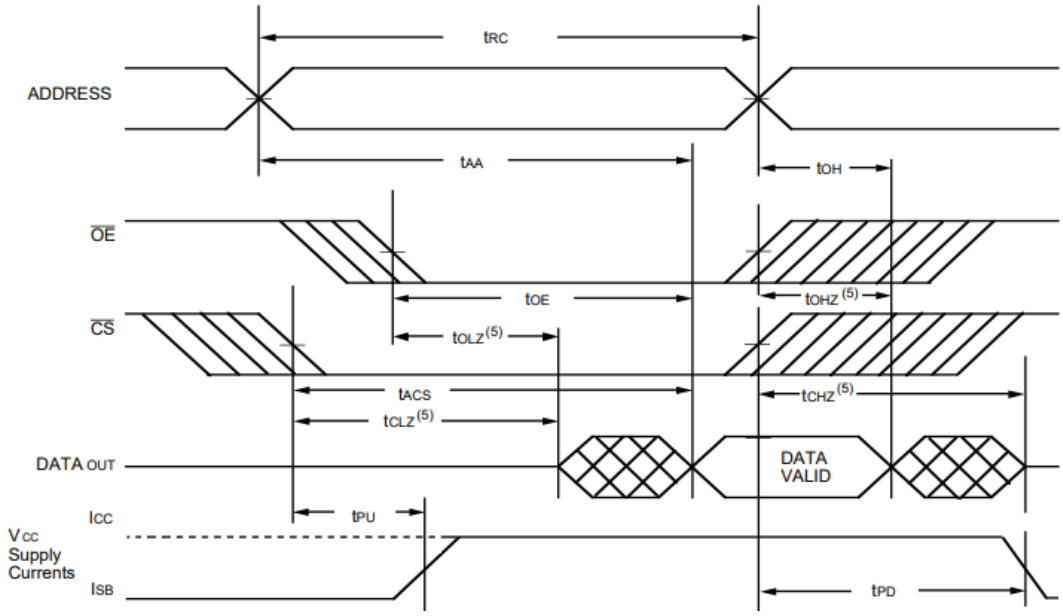


Рис. 3.5.2.2. Цикл читання 1.

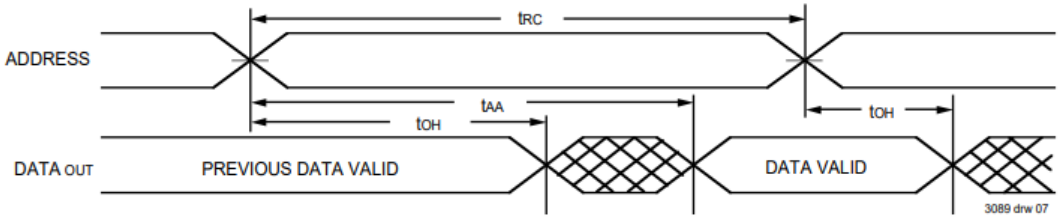


Рис. 3.5.2.3. Цикл читання 2.

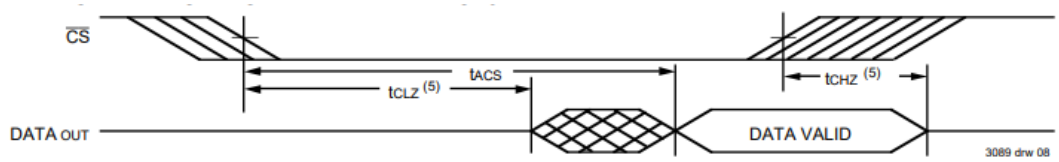


Рис. 3.5.2.4. Цикл читання 3.

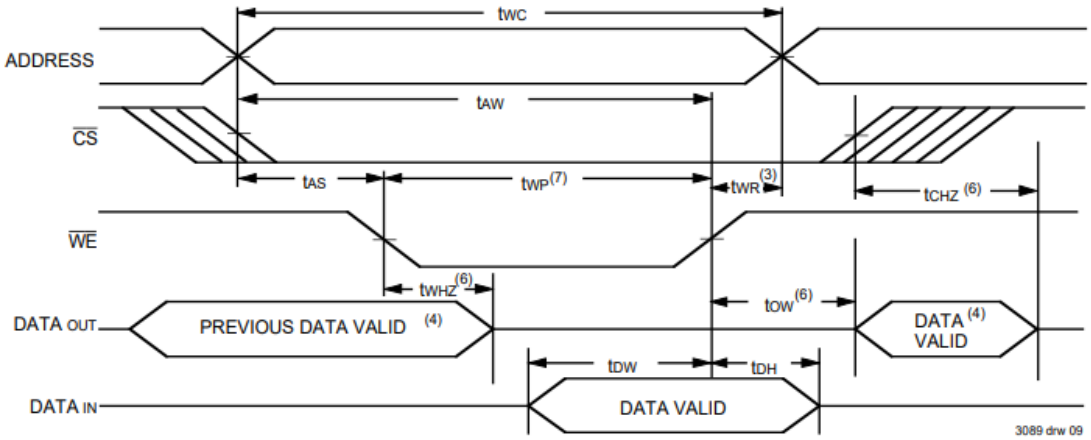


Рис. 3.5.2.5. Цикл запису 1.

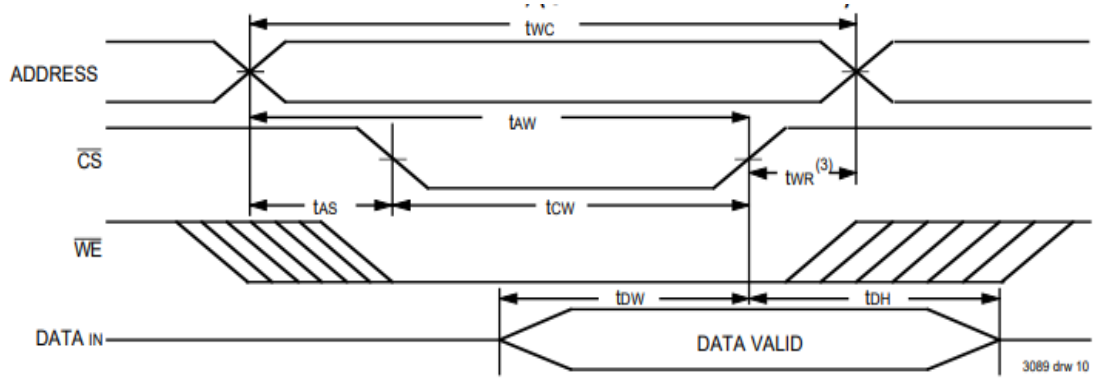


Рис.3.5.2.6 Цикл запису 2.

ОЗП CY7C128A(RAM)

Мікросхема моделі CY7C128A використовується для запису, зберігання та читання 8-розрядних слів.

Особливості:

• Автоматичне відключення живлення, якщо цей параметр скасовано

• CMOS для оптимальної швидкості/потужності

• Висока швидкість — 15 нс

• Низька активна потужність — 440 мВт (комерційний) — 550 мВт (військовий) • Низька потужність у режимі очікування — 110 мВт

• TTL-сумісні входи та виходи

• Здатність витримувати електростатичний розряд понад 2001 В

• V IH 2,2 В

A10…A0 – адресні входи,

WE – Вхід дозволу запису та зчитування

OE – вхід управління станом вхідного буфера

CE - Вхід вибору кристала

Пам'ять категорії => SRAM => Async. SRAM => 16 Кб

Опис CMOS Static RAM 16k : 2kx8

Компанія CYPRESS [Cypress Semiconductor]

Входи та виходи мікросхема CY7C128A :

- A0…A10 – Адресні входи;
- WE – Вхід дозволу запису та зчитування;
- OE – Вхід управління станом вхідного буфера;
- CE – Вхід вибору кристалу;
- DQ0…DQ7 – Входи/виходи даних

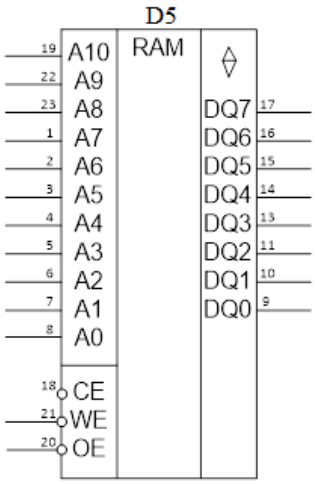


Рис. 3.5.2.7. Умовне графічне позначення ОЗП CY7C128A

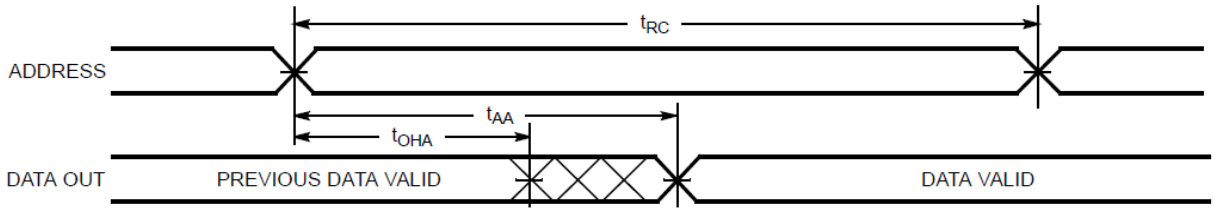


Рис. 3.5.2.8. Цикл читання 1

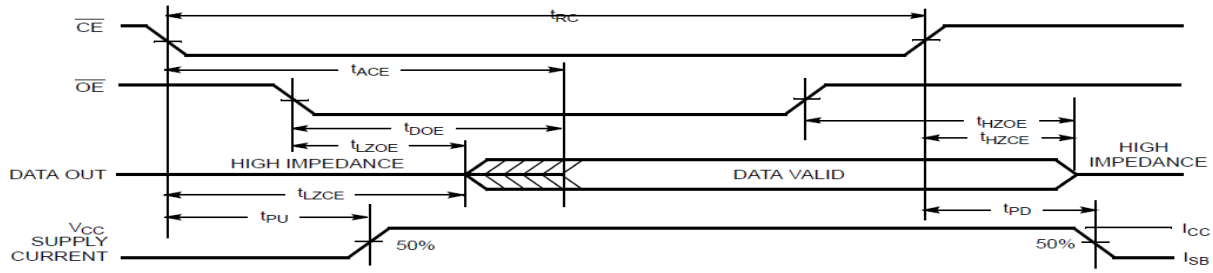

CE
OE
DATA OUT
VCC
SUPPLY
CURRENT
tRC
tACE
tDOE
tLZOE
HIGH IMPEDANCE
tLZCE
tPU
50%
DATA VALID
tHZOE
tHZCE
HIGH
IMPEDANCE
tPD
50%
ICC
ISB

Рис. 3.5.2.9. Цикл читання 2

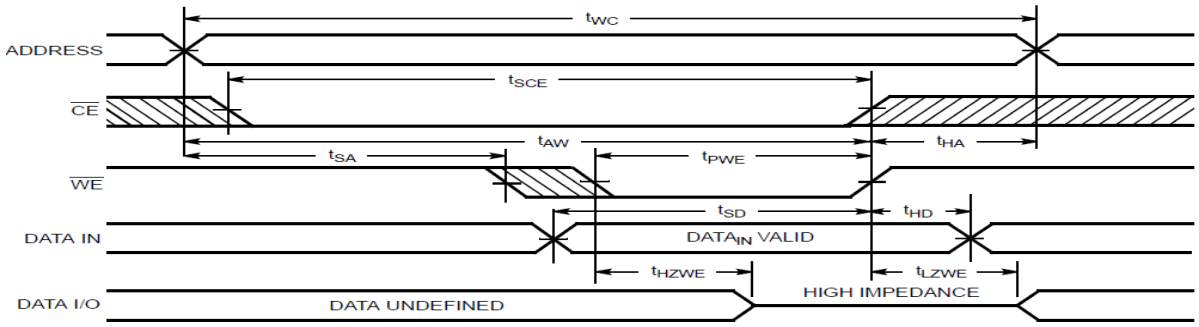


Рис. 3.5.2.10. Цикл запису WE

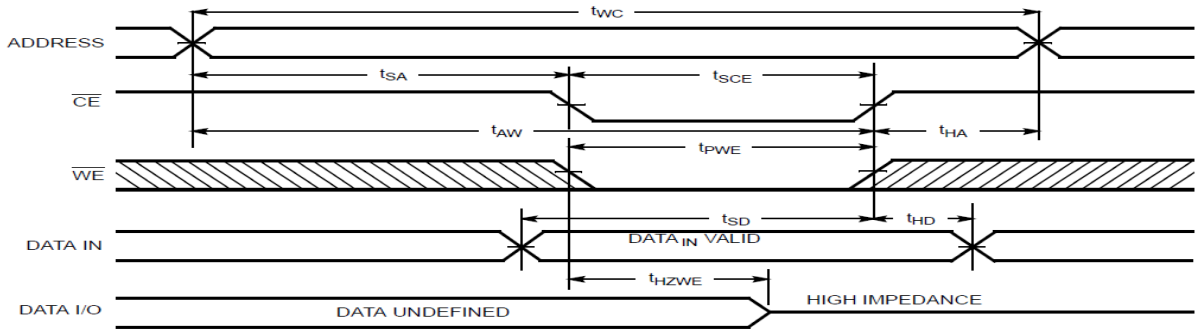


Рис. 3.5.2.11. Цикл запису CE

3.5.3. Двійковий суматор

Суматори – це комбінаційні пристрої, що здійснюють основну арифметичну операцію – додавання чисел в двійковому коді. Найбільш поширені двійкові повні суматори.

Кожний розряд двійкового повного суматора має три входи (A і B – для доданків, P0 – сигнал переносу від попереднього розряду) і два виходи (S – суми і P – сигналу переносу в наступний розряд). Входи A, B і P0 в загальному рівноправні. Сигнал суми S приймає значення лог.1 при непарній кількості одиниць на входах A, B і P0, і лог.0 при парній. Сигнал переносу P рівень лог.1 при кількості одиниць на входах, рівній 2 або 3. Логічні функції виходів повного одно-розрядного суматора:

$$S = A \oplus B \oplus \mathrm{P0}$$
$$P = A \cdot B \vee B \cdot \mathrm{P0} \vee A \cdot \mathrm{P0}$$

Суматор SN74S283

Мікросхема SN74S283 – це швидкодіючий повний 4-розрядний двійковий суматор. В якому використовуються такі входи та виходи :

A1…A4 - входи даних

B1…B4 –входи даних

P0 – вхід переносу

P – вихід сигналу переносу

S1…S4 – виходи на яких появляються суми вхідних слів

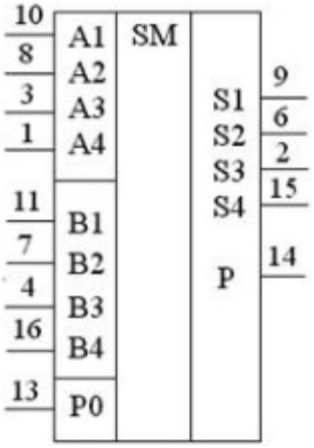


Рис. 3.5.3.1 Умовне графічне позначення суматора SN74S283(SN74LS283)

Мікросхема SN74LS283 - швидкодіючий повний 4-розрядний двійковий суматор. Логіка його роботи відповідає логіці роботи суматора SN74S283. В зв'язку із симетрією двійкової логіки SN74LS283 також можна використовувати як з високорівневою, так і з низькорівневою логіками.

3.5.4. Арифметико-логічний пристрій (ALU)

Арифметико-логічний пристрій - блок процесора, що служить для виконання арифметичних та логічних перетворень над даними, що

іменуються операндами. Цей пристрій є фундаментальною частиною будь-якого обчислювача, навіть найпростіші мікроконтролери мають його в складі свого ядра.

ALU SN74LS181

Мікросхема SN74LS181 – 4-розрядний швидкісний АЛП. Він може працювати в двох режимах, виконуючи або 16 логічних, або 16 арифметичних операцій. Для отримання максимальної швидкодії при обробці багаторозрядних цифрових слів в схемі АЛП присутня внутрішня схема прискореного переносу.

На входи A1..A4 подається 4-розрядне слово A (операнд A), на входи B1..B4 – слово-операнд B. АЛП має 4 входи вибору C0..C3, за допомогою яких можна вибрати 2^4 = 16 функцій пристрою. За допомогою входу M (Mode) АЛП переключається в режим виконання логічних (M=1) або арифметичних (M=0) функцій двох змінних. Таким чином загальна кількість функцій, які виконуються АЛП складає 32.

На вхід $\overline{P0}$ приймається вхідний сигнал переносу (активний рівень – лог.0). Результат виконання однієї з 32 вибраних функцій з'являється на виходах S1..S4. На виході $\overline{P4}$ формується сигнал переносу після чотирьох розрядів (активний рівень – лог.0). Цей сигнал подається на вхід $\overline{P0}$ наступного АЛП при побудові схем АЛП великої розрядності.

Мікросхема SN74LS181 має три додаткових виходи: A=B – вихід компаратора, який відображає рівність операндів (має вихідний каскад з відкритим колектором), GRG – вихід генерації переносу і GRP – вихід розповсюдження переносу, які використовуються при побудові багаторозрядних АЛП з прискореним переносом.

Мікросхема SN74LS181 керується паралельними входами вибору C1..C4 і входом керування режимом M. Якщо на вході M лог.1, забороняються всі внутрішні переноси і пристрій буде виконувати логічні операції порозрядно. При лог.0 на вході M переноси дозволяються і будуть виконуватись арифметичні операції над двома 4-розрядними

словами. Завдяки повній внутрішній схемі прискореного переносу сигнал переносу на виході $\overline{P4}$ з'являється при кожному вхідному сигналі переносу, що надходить на вхід $\overline{P0}$. Для організації переносу між корпусами АЛП, об'єднаними в багаторозрядну схему, використовуються виходи GRG і GRP. Дані, що з'являються на них, не залежать від стану входу переносу $\overline{P0}$.

Якщо від багатокорпусного АЛП не вимагається максимальної швидкодії, можна використати простий режим пульсуючого переносу. Для цього вихід переносу $\overline{P4}$ з'єднують з входом переносу $\overline{P0}$ наступного АЛП. Для забезпечення високошвидкісних операцій необхідно включати між АЛП SN74LS181 спеціальну мікросхему прискореного переносу (наприклад SN74LS182).

На виході компаратора, тобто на виході відображення еквівалентності A=B буде лог.1, якщо на усіх чотирьох виходах S1..S4 з'явились лог.1. Цей вихід застосовується для відображення логічної еквівалентності 4-розрядних слів, якщо АЛП працює в режимі віднімання. Сигнал A=B можна використовувати разом з сигналом $\overline{P4}$ для визначення співвідношень A>B або A<B.

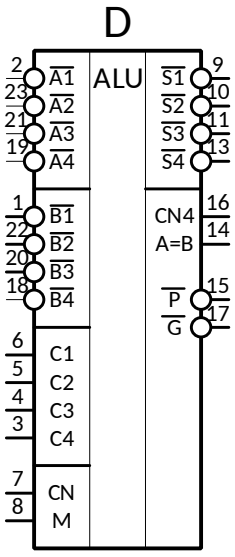


Рис. 3.5.4.1. Умовне графічне позначення

В мікросхемі SN74LS181 використовуються такі основні входи та виходи :

- A0…A4 - входи даних;

- B0…B4 –входи даних;
- C1…C4 – входи вибору ;
- M – вхід керування режимом;
- CN – вхід переносу;
- S1…S4 – виходи даних;
- CN4 – вихід переносу

Табл. 3.5.4.1. Таблиця істинності

Вибір функції				Пряма логіка		Інверсна логіка	
C3	C2	C1	C0	Логічні функції (M=1)	Арифметичні функції (M=0)	Логічні функції (M=1)	Арифметичні функції (M=0)
0	0	0	0	$\overline{A}$	$A+\overline{P0}$	$\overline{A}$	$A-1+P0$
0	0	0	1	$\overline{A \vee B}$	$(A \vee B)+\overline{P0}$	$\overline{A \bullet B}$	$A \bullet B-1+P0$
0	0	1	0	$\overline{A} \bullet B$	$(A \vee \overline{B})+\overline{P0}$	$\overline{A} \vee B$	$A \bullet \overline{B}-1+P0$
0	0	1	1	0	$-1+\overline{P0}$	1	$-1+P0$
0	1	0	0	$\overline{A \bullet B}$	$A+A \bullet \overline{B}+\overline{P0}$	$\overline{A \vee B}$	$A+(A \vee \overline{B})+P0$
0	1	0	1	$\overline{B}$	$(A \vee B)+A \bullet \overline{B}+\overline{P0}$	$\overline{B}$	$A \bullet B+(A \vee \overline{B})+P0$
0	1	1	0	$A \oplus B$	$A-B-1+\overline{P0}$	$\overline{A \oplus B}$	$A-B-1+P0$
0	1	1	1	$A \bullet \overline{B}$	$A \bullet \overline{B}-1+\overline{P0}$	$A \vee \overline{B}$	$(A \vee \overline{B})+P0$
1	0	0	0	$\overline{A} \vee B$	$A+A \bullet B+\overline{P0}$	$\overline{A} \bullet B$	$A+(A \vee B)+P0$
1	0	0	1	$\overline{A \oplus B}$	$A+B+\overline{P0}$	$A \oplus B$	$A+B+P0$
1	0	1	0	B	$(A \vee \overline{B})+A \bullet B+\overline{P0}$	B	$A \bullet \overline{B}+(A \vee B)+P0$
1	0	1	1	$A \bullet B$	$A \bullet B-1+\overline{P0}$	$A \vee B$	$(A \vee B)+P0$
1	1	0	0	1	$A+A+\overline{P0}$	0	$A+A+P0$
1	1	0	1	$A \vee \overline{B}$	$A+(A \vee B)+\overline{P0}$	$A \bullet \overline{B}$	$A+A \bullet B+P0$
1	1	1	0	$A \vee B$	$A+(A \vee \overline{B})+\overline{P0}$	$A \bullet B$	$A+A \bullet \overline{B}+P0$
1	1	1	1	A	$A-1+\overline{P0}$	A	$A+P0$

3.5.5. Регістр (RG)

Регістр — послідовний або паралельний логічний пристрій, який виконує функцію приймання, запам'ятовування і передавання інформації.

Інформація в регістрі зберігається за видом числа (слова), зображеного комбінацією сигналів низького та високого рівня. Кожному розряду числа, що записаний в регістр, відповідає свій розряд, побудований, як правило, на базі тригерів RS-, D- або JK- типу.

Регістр 74374

Мікросхема 74374 містить вісім D-тригерів, що запускаються фронтом тактового імпульсу, з виходами, що мають три стани.

Дані, що надходять на входи D0 — D7 мікросхеми 74374 зберігаються в тригерах при перепаді напруги на вході тактових

імпульсів Clock з низького рівня на високий (позитивний фронт імпульсу). Вхід тактових імпульсів виконаний на тригері Шмітта.

Записані дані надходять на виходи Q мікросхеми 74374, коли на вхід OE (дозвіл видачі вихідних сигналів) подається напруга низького рівня. Якщо на цьому вході встановлюється напруга високого рівня, то всі виходи переходять у високоомний (третій) стан.

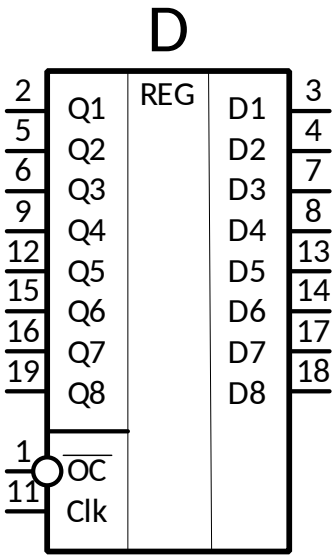


Рис. 3.5.5.1. Умовне графічне позначення мікросхема 74374

Табл. 3.5.5.1. Таблиця істинності

Входи			Вихід Q
OE	Clock	D	
0	⎍	0	0
0	⎍	1	1
0	0, 1, ⎏	X	Немає змін
1	X	X	Z

Регістр зсуву (RG_S) SN74198

Мікросхема SN74198 (74F198) містить реверсивний 8-розрядний регістр зсуву даних з паралельним і послідовним введенням-виведенням інформації.

Якщо на вхід скидання Clear мікросхеми подається напруга високого рівня, то режим роботи визначається станами входів S0 і S1 (режим управління).

Зсув даних вліво відбувається, коли таке напруга подається на входи S0 і S1. При цьому дані послідовно надходять на вхід L. Зсув даних мікросхеми вправо відбувається, коли на вхід S0 подається напруга високого, а на вхід S1 — низького рівня, при цьому дані послідовно надходять на вхід R.

Якщо на обидва входи S0 і S1 мікросхеми подається напруга високого рівня, то можливе паралельне завантаження даних з входів A — H.

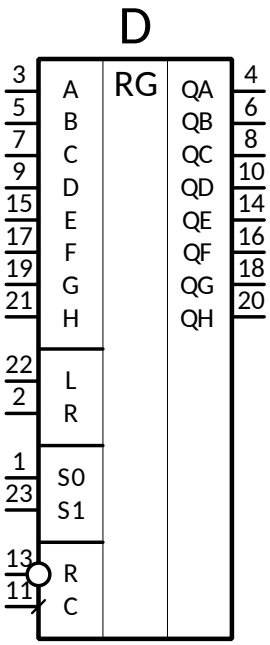


Рис. 3.5.5.2. Умовне графічне позначення мікросхеми SN74198

Табл. 3.5.5.2. Таблиця істинності

Входи				Функція
Clear	Clock	S0	S1	
0	X	X	X	Асинхронне скидання
1	⎍	1	1	Паралельне завантаження
1	⎍	0	1	Зсув вправо
1	⎍	1	0	Зсув вліво
1	X	0	0	Зберігання даних

3.5.6. Постійно запам’ятовуючий пристрій (ROM)

Постійний запам’ятовуючий пристрій — енергонезалежна пам'ять, з якої може проводитись тільки зчитування даних. Прикладом ПЗП є мікросхема ROM, яка використовується для керування роботою апаратної частини комп'ютера. Ця мікросхема містить BIOS (Basic Input/Output System — базова система вводу/виводу). Комп'ютер звертається до BIOS одразу ж після подачі живлення на центральний процесор, ще до завантаження операційної системи. Мікросхема ПЗП здатна тривалий час зберігати інформацію, навіть при вимкненому комп'ютері. Мікропрограми, які знаходяться в ПЗП, «зашиті» у ньому — вони записуються туди на етапі виготовлення мікросхеми. Основне призначення BIOS полягає в тому, щоб перевірити склад та працездатність системи та забезпечити взаємодію з клавіатурою, монітором, жорсткими та гнучкими дисками. Внутрішня схема постійного запам’ятовуючого пристрою показана на рис. 3.5.6.1.

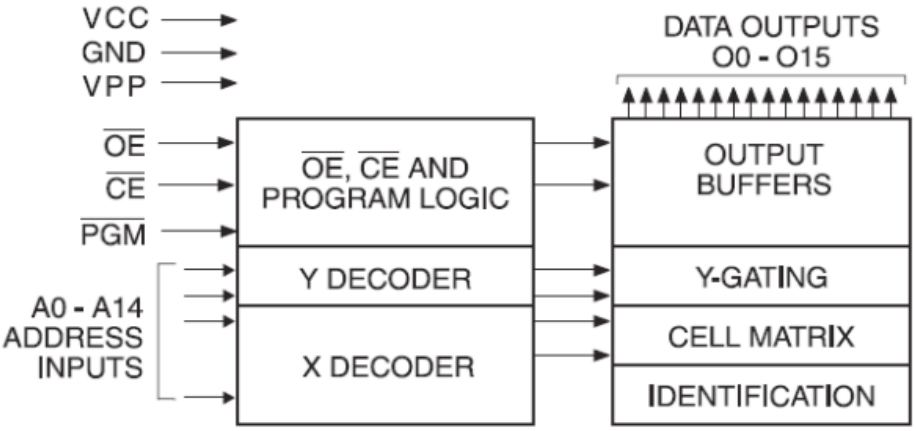


Рис. 3.5.6.1. Внутрішня схема постійного запам’ятовуючого пристрою

Сучасні ПЗП будуються на основі МОН-структур і мають значно більші можливості, оскільки інформація в них може бути замінена багаторазово.

Вони розділяються на такі групи:

- *EPROM* (***E**lectrically **P**rogrammable **ROM***) або РПЗП-УФ (репрограмовані ПЗП з ультрафіолетовим стиранням записаної інформації);
- *EEPROM* (***E**lectrically **E**rasable **PROM***) або **РПЗП-ЕС** (репрограмовані ПЗП з електричним стиранням записаної інформації);
- пам'ять типу *flash* (флеш).

Прикладом паралельних EEPROM є мікросхеми сім'ї AT28C фірми ATMEL.

Особливості приладів цієї сім'ї полягають у наступному:

- використана одна напруга живлення для всіх операцій;
- 8-розрядна організація;
- вбудовані фіксатори адреси та даних;
- вбудований автомат програмування;
- забезпечується побайтове і секторне програмування;
- типова кількість циклів програмування – не менш 10000;
- виконання як для побутової (діапазон робочих температур 0…+ 70° С), так і промислової (– 40…+ 85° С) апаратури.

Сім'я 8-розрядних EEPROM AT28C має діапазон інформаційних ємностей від 16 Кбіт до 4 Мбіт. В усіх режимах мікросхеми працюють від одного джерела живлення, що забезпечує просте внутрисистемне перепрограмування. Частина МС забезпечує побайтове програмування як одного (кількох) байтів, так і всього обсягу основної пам'яті, інші мають секторну організацію і за одну операцію програмуються як в межах одного сектора, так і декількох байтів. Програмування забезпечується вбудованим автоматом, не вимагає зовнішнього тактування і операцій попереднього стирання інформації. Звернення до пристроїв при операціях читання / запису аналогічне зверненню до статичних ОЗП. Мікросхеми містять вбудовані регістри на весь об'єм даних – байт або сторінку в залежності від типу приладу. На час циклу запису адреса і дані фіксуються вбудованими регістрами-фіксаторами, звільнюючи шину для інших операцій керуючого процесора.

Додатковий механізм програмного захисту даних зберігає дані від випадкового перепрограмування. У приладах сім'ї виділена особлива область пам'яті ємністю від 32 до 128 байт для зберігання користувальницьких даних ідентифікації.

Одним з представників сім'ї AT28C з паралельною адресною організацією вводу-виводу є EEPROM AT28C64 та її модифікації для різних використань з організацією 8K × 8 ємністю 64K. Умовне позначення мікросхеми приведено на рис. 3.5.6.1.

Призначення виводів приводиться у табл. 3.5.6.1.

Мікросхема має час доступу до даних (час зчитування), що не перевищує 120 нс, а інтервал часу запису коливається у межах від 200 мкс до 1 мс. Цикли запису та зчитування забезпечуються сигналами мікропроцесора без допоміжних елементів.

Таблиця 3.5.6.1.

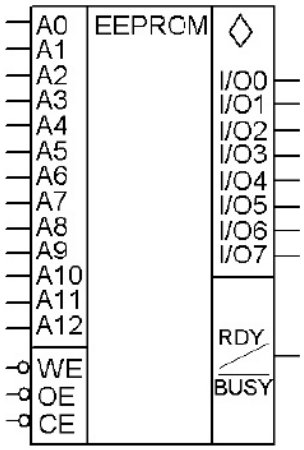


Рис. 3.5.6.1

Назва виводу	Призначення
$A_0 \dots A_{12}$	*Address Inputs* (адресні входи)
$\overline{CE}$	*Chip Enable* (пристрій активований)
$\overline{OE}$	*Output Enable* (вихід дозволено)
$\overline{WE}$	*Write Enable* (запис дозволено)
$I/O_0 \dots I/O_7$	*Data Inputs / Outputs* (інформаційні входи/виходи)
$RDY/\overline{BUSY}$	*Ready*/$\overline{Busy}$ *Output* (вихід готовності/зайнятості)

Оскільки інтервал часу запису набагато більший, ніж часу читання, то при записі даних передбачено, що код адреси і дані попередньо запам'ятовуються, звільнюючи шини процесора для іншого використання. Після ініціалізації циклу запису мікросхема переходить у стан зайнятості, і перезапис введених даних забезпечується за допомогою внутрішнього керуючого таймера. Для визначення кінця циклу запису використовуються два способи. Перший забезпечується

рівнем сигналу на виході *RDY* , а другий – рівнем сигналу на *I/O*$_7$.

Мікросхема має режим енергозбереження. Споживаний струм в активному режимі не перевищує 30 мА при напрузі живлення 5 В. У режимі енергозбереження величина споживаного струму обмежується величиною 100 мкА. Мікросхеми фірми ATMEL забезпечують внутрішню корекцію даних, що дає можливість підвищити надійність і покращити якість даних, які зберігаються.

Блок-схема мікросхеми EEPROM AT28C64 приведена на рис. 3.5.6.2.

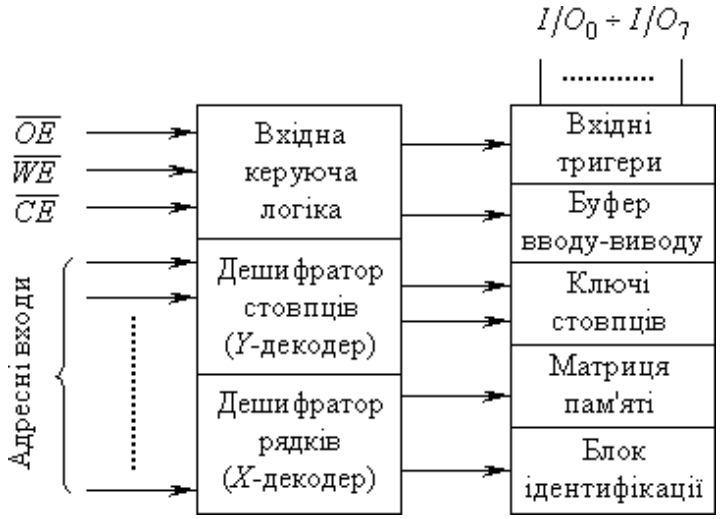


Рис. 3.5.6.2. Блок-схема мікросхеми EEPROM AT28C64

Таблиця 3.5.6.2

Тип приладу	Ємність, біт	Організація	Обсяг сторінки, байт	Час вибірки, нс	Тривалість циклу запису байту {сторінки}, мкс	I_{AC}, мА	I_{SB}, мА
AT28C16-T	16 К	2К x 8			1	30	0,1
AT28C64 AT28C64E	64 К	8К x 8		120	1 0.2	30	0,1
AT28C256F	256 К	32К x 8	64		{3}	50	0,2
AT28C010 AT28C010E	1 М	128К x 8	128	150	{1	40	0,

					0 }		2

AT28C040	4 M	512K x 8	256	280	{
					1
					0
					}
						80	0
							,
							3

У мікросхемі AT28C64B рис. 3.5.6.3. передбачені контроль розміру сторінки даних і індикація кінця циклу запису для забезпечення якості запису інформації. До того ж, у мікросхемі використані апаратні та програмні засоби захисту даних.

Апаратні засоби з невеликим розширенням практично повторюють засоби, застосовані в попередній мікросхемі. Програмні засоби захисту даних (*SDP – Software Data Protection*) призначені для упередження ненавмисного запису. Ці програмні засоби є внутрішнім алгоритмом роботи мікросхеми і користувачем можуть не використовуватись.

У табл. 3.5.6.2 приведені основні параметри мікросхем сім'ї AT28C.

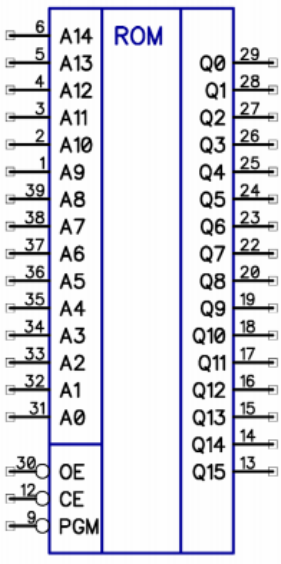


Рис. 3.5.6.3. Умовне графічне позначення AT28C64B

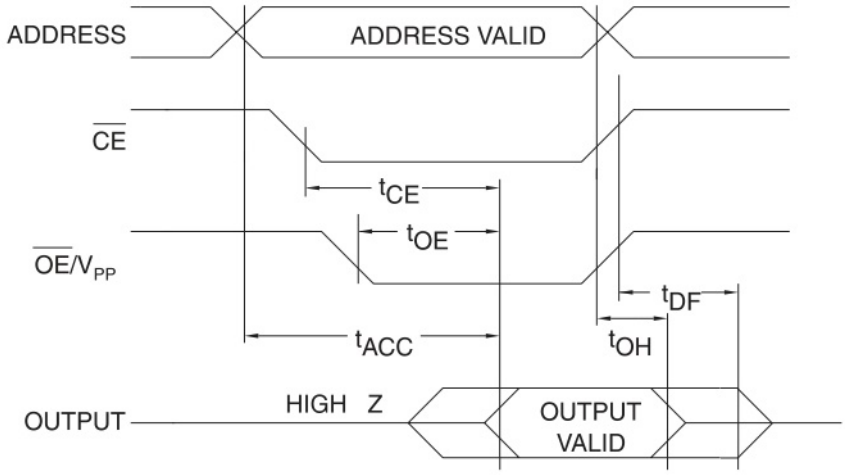


Рис. 3.5.6.4. Цикл читання AT28C64B

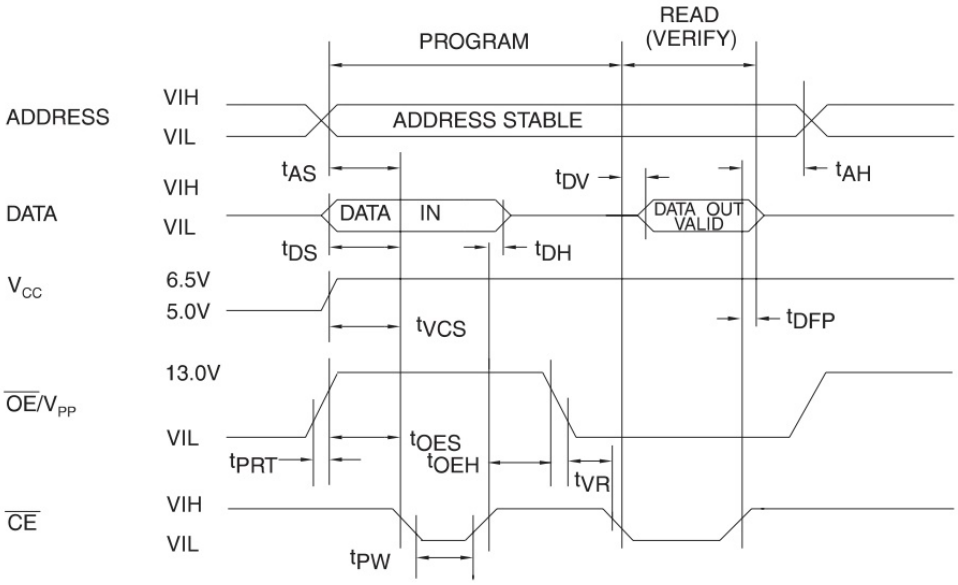


Рис. 3.5.6.5. Цикл запису AT28C64B

Основні параметри мікросхем пам'яті:

a) інформаційна ємність. Здатність зберігати кілька біт двійкової інформації;
b) організація мікросхем ЗП. Вона може бути різною при тому самому об'ємі пам'яті. Наприклад, 65536 біт можуть виглядати як 4096 х 16, або як 8192 х 8, або в іншому поєднанні. Внутрішня організація матриці, що запам'ятовує, при цьому залишається незмінною, змінюється тільки зовнішній інтерфейс і, відповідно, число зовнішніх висновків;
c) час вибірки. Час від подачі останнього з сигналів, що дозволяють читання до появи на виході стійких даних;
d) споживана потужність. Як завжди, існує компроміс між споживаною потужністю та швидкодією мікросхеми;
e) напруга живлення. Загальна тенденція до зниження напруги живлення призвела до появи мікросхем ЗП, що працюють за 3,3, 2,5 і навіть 1,8 вольт;
f) температурний діапазон. Комерційний, індустріальний чи розширений.

До специфічних параметрів ЗУ відносяться такі як час зберігання (годин, років), число циклів перезапису, час стирання та інші.

3.5.7. Шинний формувач (BD)

Шинний формувач представляє 8 - розрядний паралельний передатчик з три стабільними виходами і використовується як буферний пристрій шини даних в мікропроцесорних системах.

Пристрій передає данні з шини А в шину В або з В шини у А шину , в залежності від логіки рівня в напрямку контролю (DIR - вхід керування напрямком передачі). OUT-Enable (OE) вхід може бути використаний для відключення пристрою, щоб шина була надійно ізольована.

Мікросхема DM74ALS245A - Цей вдосконалений пристрій містить 8 пар 3-СТАНОВИХ логічних елементів, виконаних як восьмеричні передавальні шини. Ці схеми призначені для використання в пам'яті,

мікропроцесорних системах і в асинхронних двонаправлених шинах даних. Двосторонній зв'язок між шинами контролюється входом (DIR). Дані передаються або з шини А на шину В, або з шини В на шину А. І виходи драйвера, і приймача можна відключити через вхід (G), що приводить до того, що виходи переходять у режим високого опору, щоб шини були ефективно ізольованими.

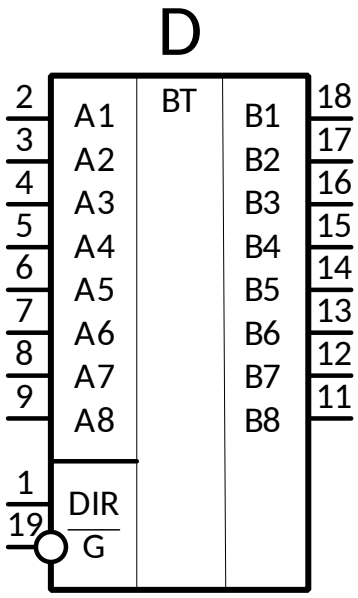


Рис. 3.5.7.1. Умовне графічне позначення

Табл. 2.5.7.1. Таблиця істинності

Control Inputs		Operation
$\overline{G}$	DIR	
L	L	B Data to A Bus
L	H	A Data to B Bus
H	X	Hi-Z

3.5.8. Групи комбінаційної логіки

У залежності від технології виготовлення інтегральні мікросхеми (ІМС) підрозділяються на серії (сімейства), що розрізняються фізичними параметрами базових елементів і їхнім функціональним призначенням. Найбільше поширення одержали ІМС, виготовлені по ТТЛ- і КМДН-

технологіям, де ТТЛ – транзисторно-транзисторна логіка з використанням біполярних транзисторів, КМДН – з використанням комплементарних МДН- транзисторів (метал-діелектрик-напівпровідник). Типовий МДН-транзистор складається з МД/ОН- структури (метал-діелектрик/окисел- напівпровідник, наприклад n типу), та двох p-карманів для електродівджерела та стоку.

Кожна технологія виготовлення мікросхем безперервно удосконалюється у напрямку збільшення швидкодії, зменшення споживаної потужності та зростання ступеня інтеграції.

У табл. 1.1. зведені усереднені характеристики мікросхем, що виготовлені за найбільш поширеними в практиці технологіями.

Таблиця 1.1. Характеристики мікросхем виконаних в різних технологій

Тип тех-нології	Серія мікросхем	Параметри одного логічного елемента				Зарубіжний аналог
		$t_{ЗС}$, нс	$P_{ССТ}$, мВА	$t_{ЗС} P_{ССТ}$, нДж	E, В	
ТТЛ	К155, КМ155 К131 К134, КР134	10 11 66	10 23 1	100 263 66	5 5 5	SN74 SN74 SN74
ТТЛШ	К551 К1531, КР1531 К555, КМ555, 533 К1533, КР1533	3 3 10 4	20 4 2 2	60 12 20 8	5 5 5 5	SN74S SN74F SN74LS SN74ALS
ЕЗЛ	К100, К500, 700 К1500 К1800	2 0,75 1,5	25 40 20	50 30 30	-5,2 -4,5 -5,2	MC10K F00K MC10800
КМОН	К176 К561, 564, 1561	100 15-50	10^{-3} 10^{-3}	0,1 3	9 3-15	CD4000B CD4000
$I^2Л$	К583, КР 584	5	0,2	1	5-9	-

Кожна мікросхема серій сімейства 74 має своє позначення, і система позначень вітчизняних серій істотно відрізняється від прийнятої за кордоном.

Серія К155 (SN74) – це найбільш стара серія, що поступово зніметься з виробництва. Вона відрізняється гіршими параметрами в порівнянні з іншими серіями. Із цією класичною серією прийнято порівнювати всі інші.

Серія К555 (SN74LS) відрізняється від серії К155 малими вхідними струмами й меншою споживаною потужністю (струм споживання – майже втроє менше, ніж у К155). По швидкодії (часом затримок) вона близька до К155.

Серія КР531 (SN74S) відрізняється високою швидкодією (її затримки приблизно в 3-4 рази менше, ніж у серії К155), але більшими вхідними струмами (на 25% більше, ніж у К155) і великою споживаною потужністю (струм споживання – більше в півтора рази в порівнянні з К155).

Серія КР1533 (SN74ALS) відрізняється підвищеною вдвічі в порівнянні з К155 швидкодією й малою споживаною потужністю (у чотири рази менше, ніж у К155). Вхідні струми ще менше, ніж у К555.

Серія КР1531 (SN74F) відрізняється високою швидкодією (на рівні КР531), але малою споживаною потужністю. Вхідні струми й струм споживання приблизно вдвічі менше, ніж у К155.

Серія КР1554 (SN74AC) відрізняється від всіх попередніх тим, що вона виконана за КМОП-технологією. Тому вона має малі вхідні струми й мале споживання при малих робочих частотах. Затримки приблизно вдвічі менші, ніж у К155.

Найбільшою розмаїтістю наявних мікросхем відрізняються серії К155 і КР1533, найменшим – КР1531 і КР1554.

Мікросхеми різних серій звичайно легко сполучаються між собою, тобто сигнали з виходів мікросхем однієї серії можна подавати на входи мікросхем іншої серії.

При виборі тієї або іншої серії мікросхем варто також ураховувати, що мікросхеми потужної й швидкої серії КР531 створюють високий рівень завад по шинах живлення, а мікросхеми малопотужної серії К555 дуже чутливі до таких завад. Тому серію КР531 рекомендується використати тільки в крайніх випадках, при необхідності одержання дуже високої швидкодії. Не рекомендується також застосовувати в одному пристрої потужні швидкодіючі мікросхеми й малопотужні мікросхеми.

Логічний елемент І

І (англ. AND) — логічний вентиль, який реалізує операцію кон'юнкція. Активний сигнал («логічна 1», «істина») на виході цього вентиля присутній тоді, коли на обох його входах присутній активний сигнал. Якщо ж на хоча б одному із входів сигнал пасивний («логічний 0», «хиба»), на виході також буде пасивний сигнал.

Таблиця істинності

Входи		Вихід
A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Мікросхема 7408 виконує логічну функцію 2І

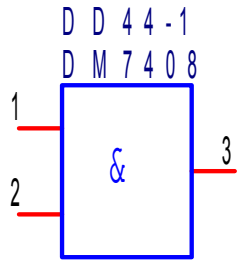


Рис. 3.5.8.1. Умовне графічне позначення 7408

Мікросхема 4011 виконує функцію 2І-НЕ

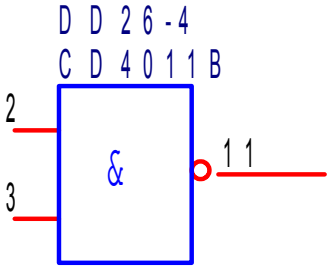


Рис. 3.5.8.2. Умовне графічне позначення 7408

Мікросхема 74S30 виконує функцію 12І-НЕ

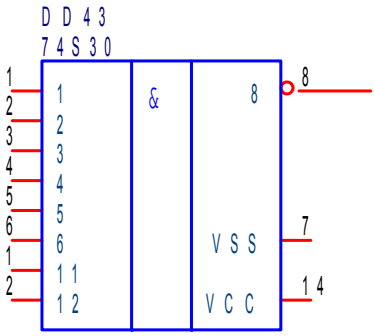


Рис. 3.5.8.3. Умовне графічне позначення 74S30

Мікросхема 4073 виконує функцію 3І

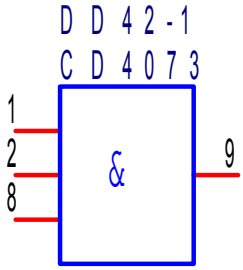


Рис. 3.5.8.4. Умовне графічне позначення 4073

Мікросхема SN74ALS11AN – містить в собі 3 незалежних елементів 3І.

Табл. 2.5.8.1. Таблиця істинності

A	B	C	f
0	X	X	0
X	0	X	0
X	X	0	0
1	1	1	1

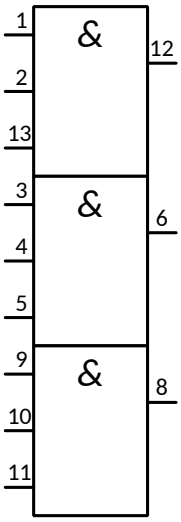


Рис. 3.5.8.5. Умовне графічне позначення

Мікросхема 7410 виконує функцію 3І-НЕ

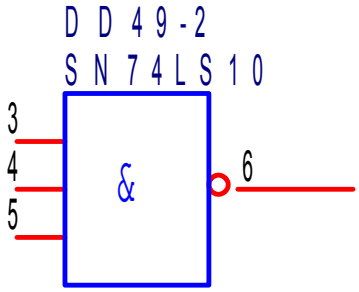


Рис. 3.5.8.6. Умовне графічне позначення 7410

Логічний елемент АБО

АБО (англ. OR) — логічний вентиль, який реалізує операцію диз'юнкція. Активний сигнал («логічна 1», «істина») на виході цього вентиля присутній тоді, коли на хоча б одному вході присутній активний сигнал. Лише коли на обох його входах сигнал пасивний («логічний 0», «хиба»), на виході також буде пасивний сигнал.

Таблиця істинності

Входи		Вихід
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Мікросхема 7432 виконує логічну функцію 2АБО

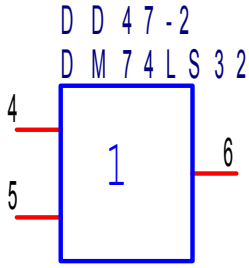


Рис. 3.5.8.7. Умовне графічне позначення 7432

Мікросхема 74LS02 виконує логічну функцію 2АБО-НЕ

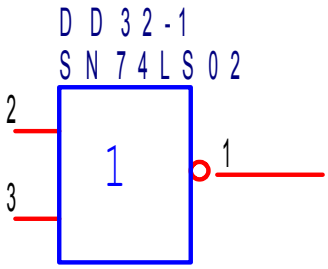


Рис. 3.5.8.8. Умовне графічне позначення 74LS02

Інвертор

Логічний елемент, який реалізує функцію інверсії. Мнемонічне правило для НЕ звучить так: На виході буде: Високий рівень тоді і лише тоді, коли на вході низький рівень, інакше на виході діє низький рівень.

Мікросхема SN7404N - містить в собі 6 незалежних елементів НЕ.

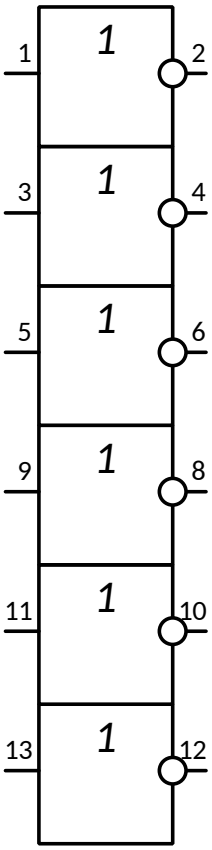


Рис. 3.5.8.10. Умовне графічне позначення

Таблиця істинності	
Вхід	Вихід
A	$\overline{A}$
0	1
1	0

Мікросхема 4075 виконує логічну функцію ЗАБО

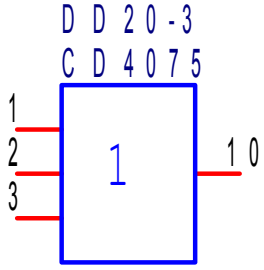


Рис. 3.5.8.9. Умовне графічне позначення 4075

4. ПРИКЛАДИ МОДУЛЮВАННЯ.

4.1. Курсове проектування спеціалізованого обчислювача в программному середовище MULTISIM

4.1.1. Мікропрограма роботи МПА.

Згідно індивідуального завдання робоча формула має вигляд:

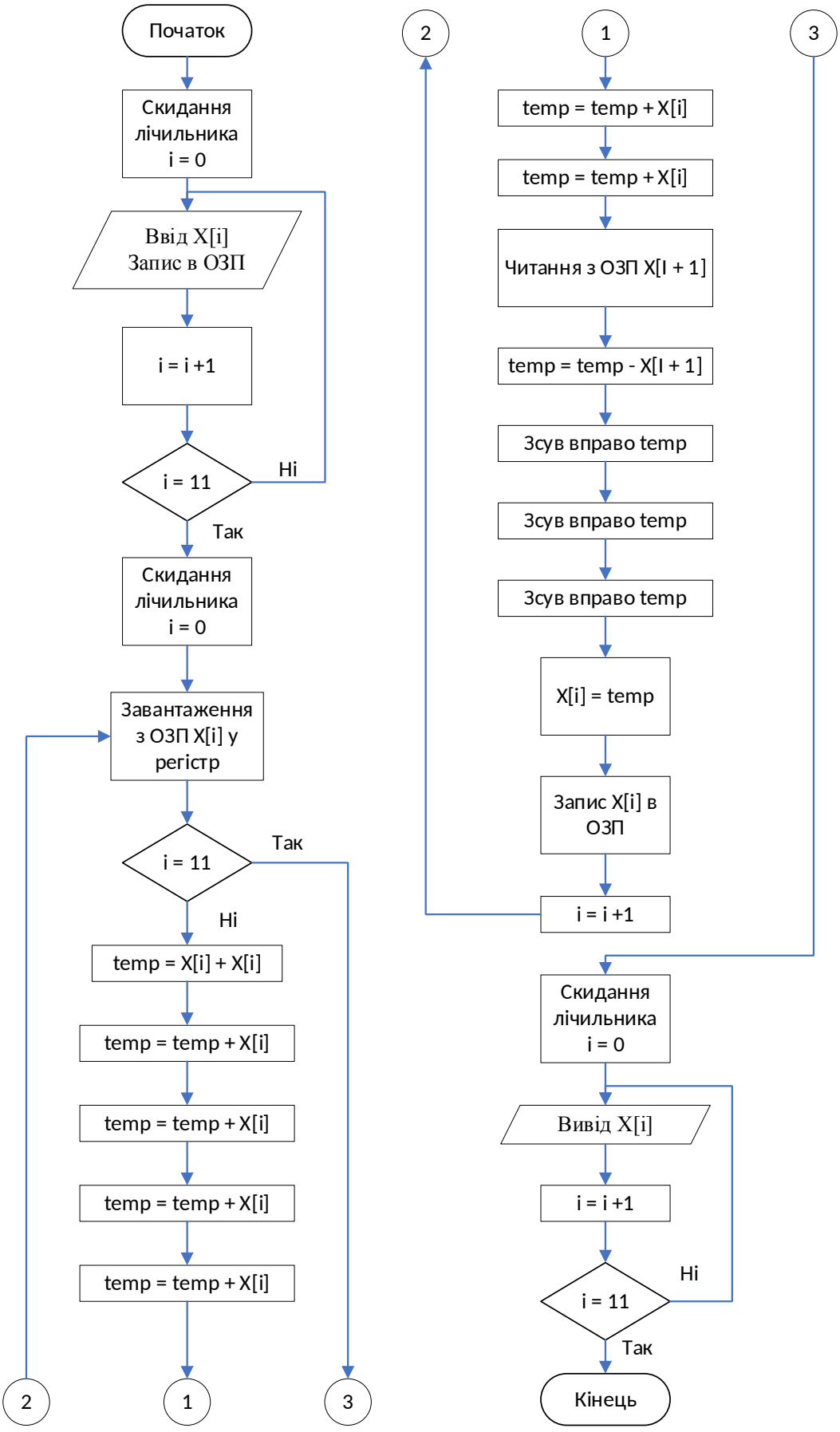
$Y_i=(7*X_i-X_{i+1})/8$; N = 11;

A0 – A16 – стани мікропрограммного автомату.

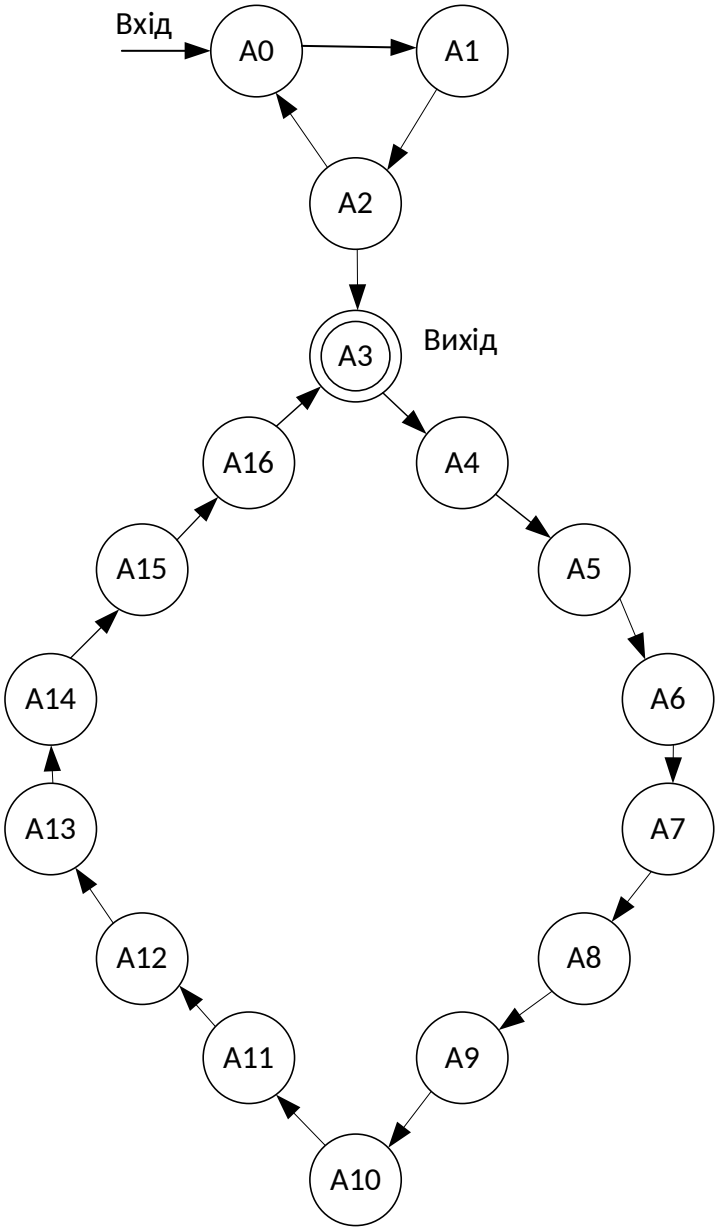
Стан	Дія, що виконується	Наступний стан
A0	WE, OE, DIR, G1, G2, UP, DOWN,	A1
A1	OE, DIR, G1, G2, UP, DOWN,	A2
A2	WE, OE, DIR, G1, G2, DOWN,	Якщо лічильник дійшов до 11 перейти у A3, інакше у A0
A3	WE, G1, S, S0, S1, UP, DOWN,	A4
A4	WE, G1, S, S0, S1, DOWN	A5
A5	WE, G1, S, S0, S1, DOWN	A6
A6	WE, G1, S, S0, S1, DOWN	A7
A7	WE, G1, S, S0, S1, DOWN	A8
A8	WE, G1, S, S0, S1, DOWN	A9
A	WE, G1, S, S0, S1, DOWN	A10

9		
A 10	WE, G1, S0, S1, UP, DOWN	A11
A 11	G, G1, G2, S0, UP	A12
A 12	G, G1, G2, S0, UP	A13
A 13	G, G1, G2, S0, UP	A14
A 14	WE, OE, UP, DOWN, G, S	A15
A 15	OE, UP, DOWN, G, S	A16
A 16	WE, OE, DOWN, G, S	A3

4.1.2. Блок-схема алгоритму



4.1.3. Граф-схема МПА



4.1.4. Формування прошивки ПЗП

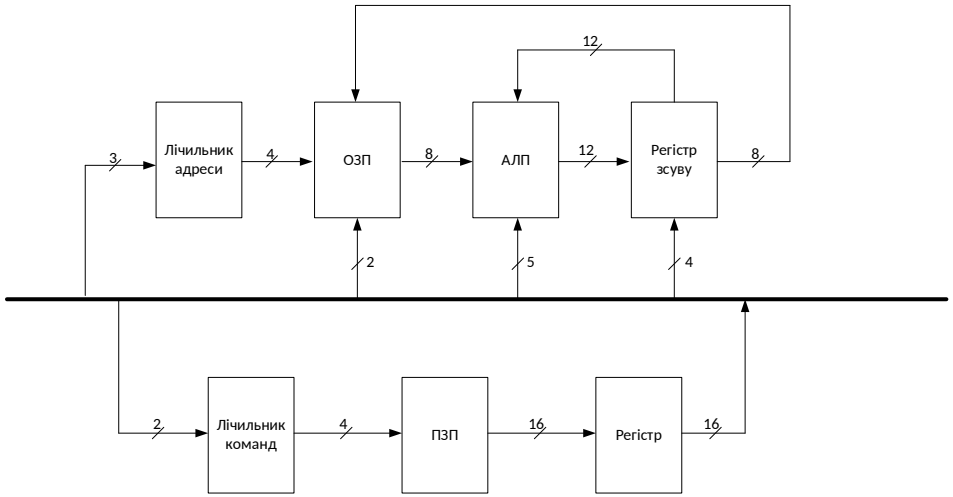
Для занесення інформації в ПЗП необхідно скласти таблицю прошиття, яка встановлює відповідність між адресами і даними. Занесення інформації в ПЗП здійснюється користувачем за допомогою програматора.

4.3.5 Результати моделювання у вигляді часових діаграм

Часові діаграми роботи схеми наведені у додатку.

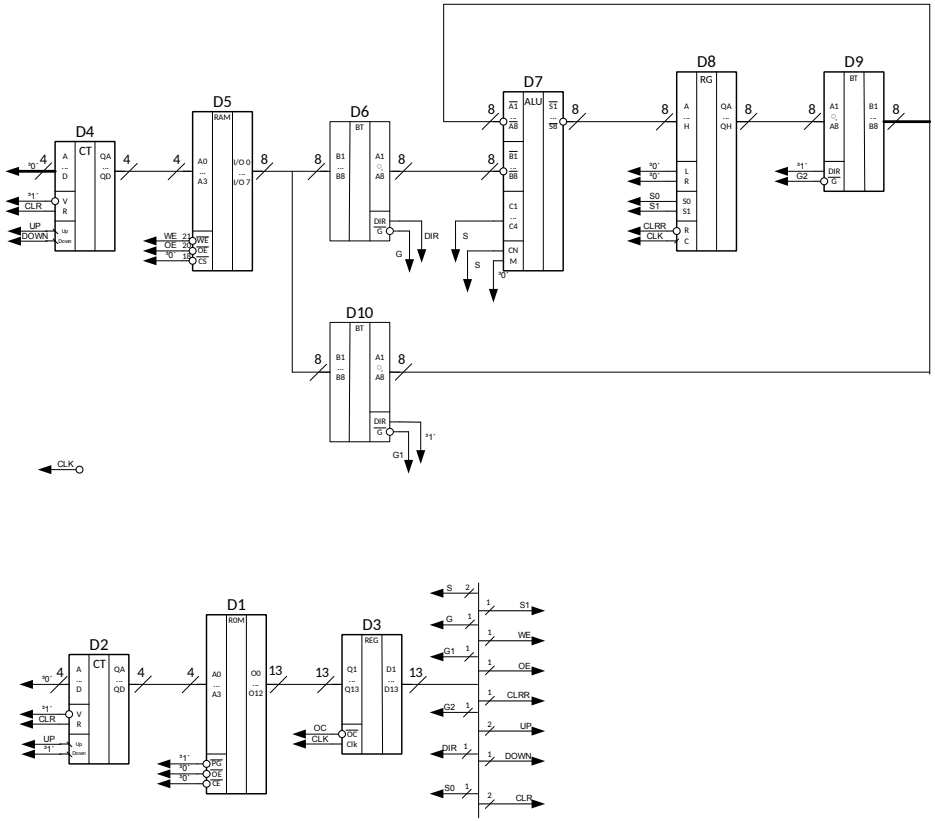
Додатки

Структурна схема



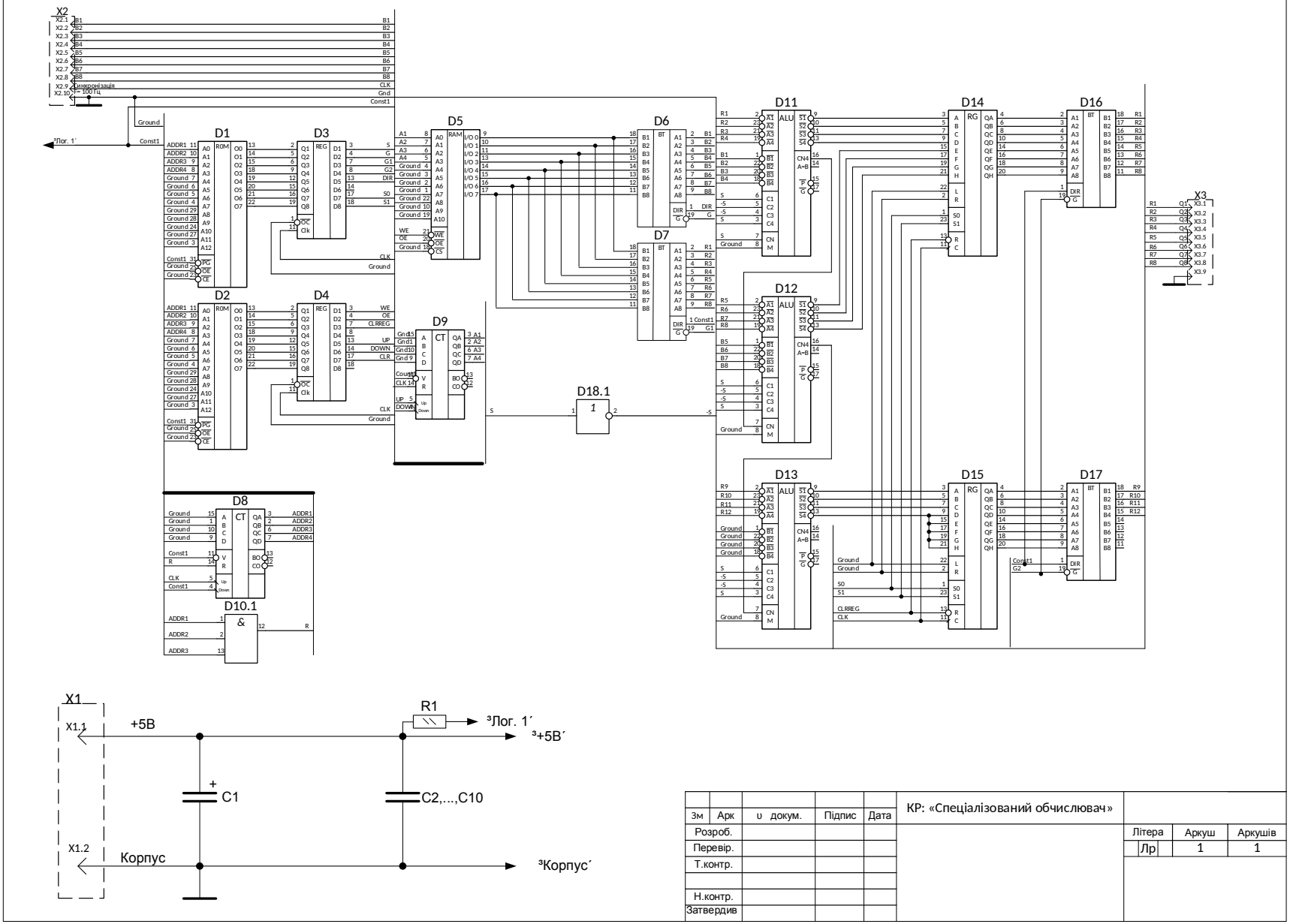
Зм	Арк	ü докум.	Підпис	Дата	КР: «Спеціалізований обчислювач»		
Розроб.						Літера	Аркуш	Аркушів
Перевір.						Лр	1	2
Т.контр.								
Н.контр.								
Затвердив								

Функціональна схема



Зм	Арк	ט докум.	Підпис	Дата	КР: «Спеціалізований обчислювач»		
Розроб.						Літера	Аркуш	Аркушів
Перевір.						Лр	1	2
Т.контр.								
Н.контр.								
Затвердив								

Принципова схема



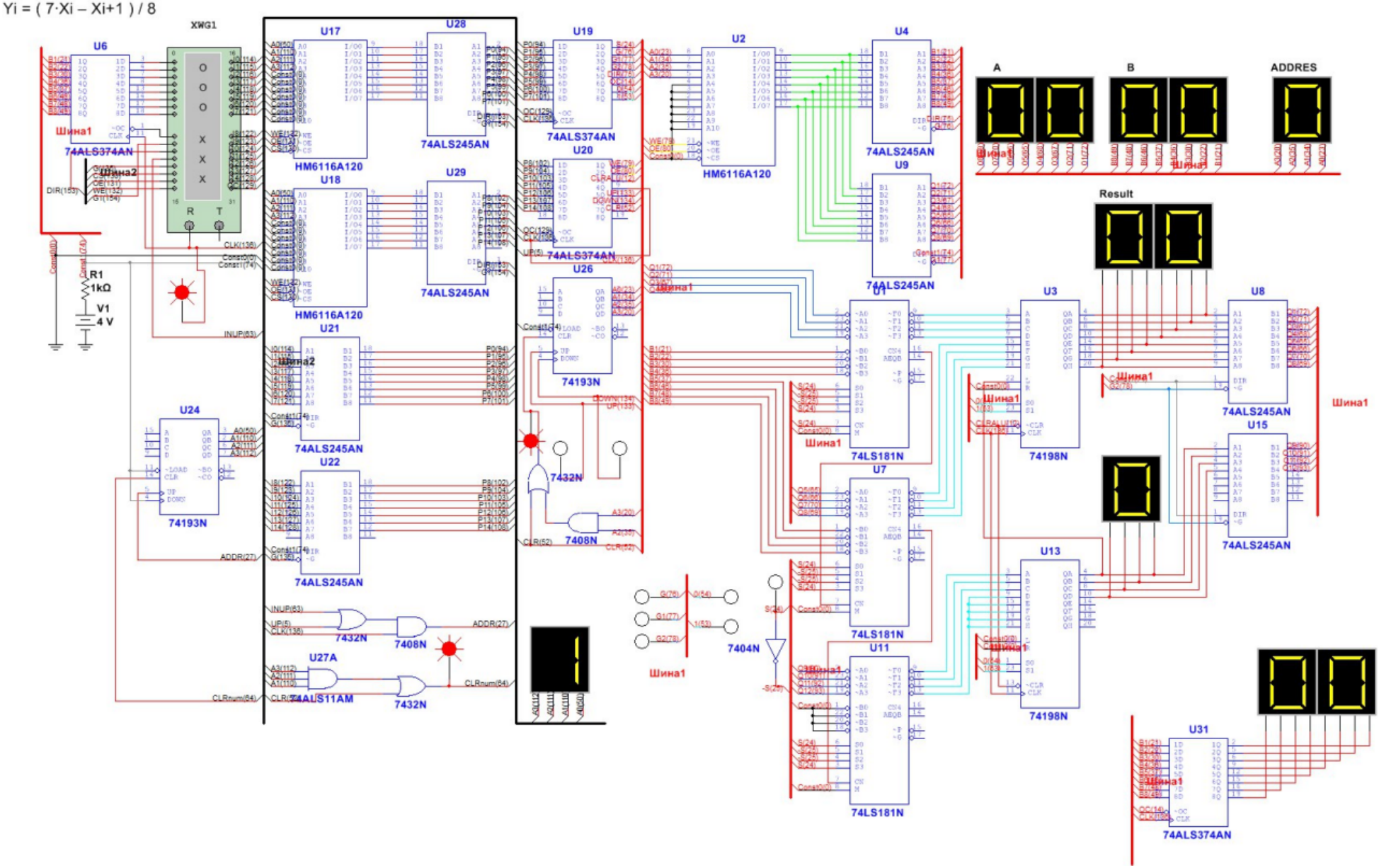
Зм	Арк	∪ докум.	Підпис	Дата	КР: «Спеціалізований обчислювач»		
Розроб.						Літера	Аркуш	Аркушів
Перевір.						Лр	1	1
Т.контр.								
Н.контр.								
Затвердив								

Таблиця прошиття ПЗП

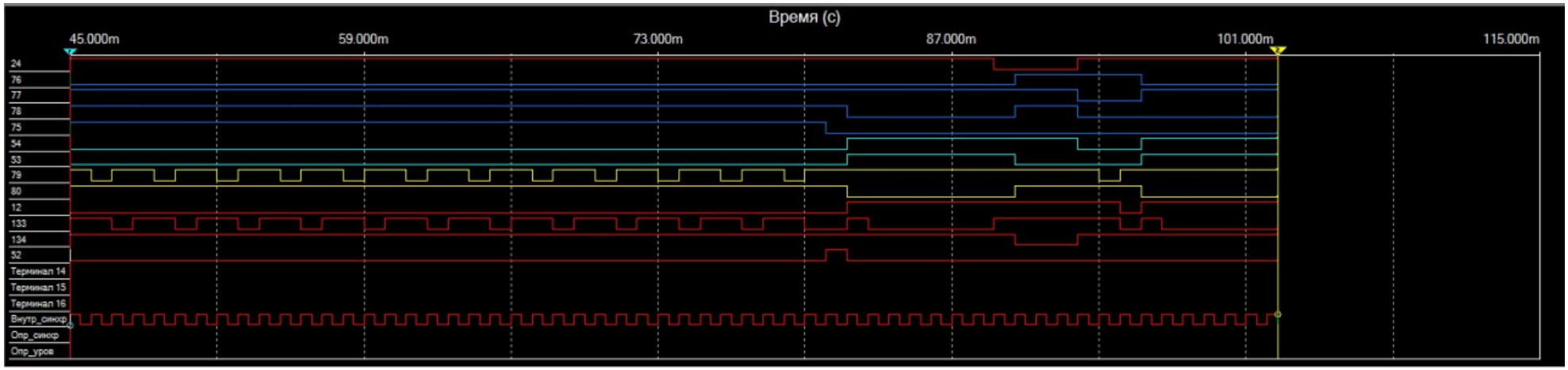
	Поточний стан					Керуючі сигнали													Наступний стан					Опис дії
	A4	A3	A2	A1	A0	S	G	G1	G2	DIR	S0	S1	WE	OE	CLRR	UP	DOWN	CLR	A4	A3	A2	A1	A0	
0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	0	0	0	1	Ввід X[i]
1	0	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0	1	0	0	0	0	1	0	Ввід X[i]
2	0	0	0	1	0	0	0	1	1	1	0	0	1	1	0	1	1	0	0	0	0	1	1	Ввід X[i]
3	0	0	0	1	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	0	0	temp += X[i]
4	0	0	1	0	0	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	0	1	temp += X[i]
5	0	0	1	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	temp += X[i]
6	0	0	1	1	0	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	temp += X[i]
7	0	0	1	1	0	1	0	1	0	0	1	1	1	0	0	0	1	0	0	1	0	0	0	temp += X[i]
8	0	1	0	0	0	1	0	1	0	0	1	1	1	0	0	0	1	0	0	1	0	0	1	temp += X[i]
9	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	1	0	1	0	temp += X[i]
10	0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	1	1	0	0	1	0	1	1	temp -= X[i+1]
11	0	1	0	1	1	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1	1	0	0	Зсув вправо
12	0	1	1	0	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1	1	0	1	Зсув вправо
13	0	1	1	0	1	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1	1	1	0	Зсув вправо
14	0	1	1	1	0	0	1	0	0	0	1	1	1	1	0	1	1	0	0	1	1	1	1	Запис Y[i]
15	0	1	1	1	1	0	1	0	0	0	1	1	0	1	0	0	1	0	1	0	0	0	0	Запис Y[i]
16	1	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	1	Запис Y[i]
17	1	0	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0	1	1	CLR REG

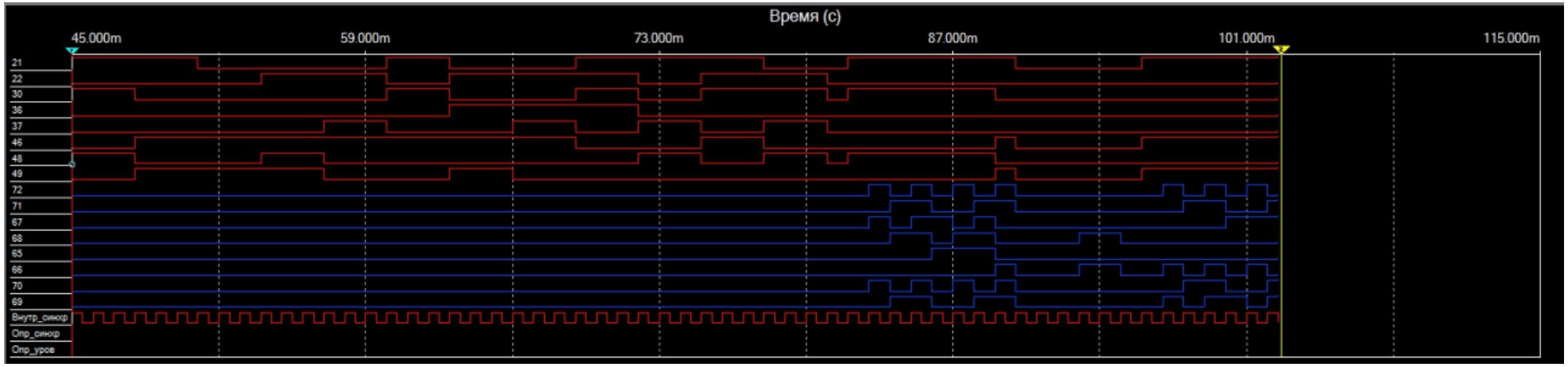
Комбінаційна схема

$Yi = (7 \cdot Xi - Xi+1) / 8$



Часова діаграма





Перелік елементів

Познач.	Найменування	Кл.	Примітки
	Резистори		
R1	1206 (1кОм)	1	
	Конденсатори		
C1	1206 (4.7 мкФ х 35 В)	1	
C2,..,C10	0806 (0,1 мкФ)	9	
	Мікросхеми		
D1, D2	27C64	2	8Кх8 ROM
D3, D4	SN74ALS374AN	2	Register
D5	HM6116A120	1	2Кх8 RAM
D6,D7 D16, D17	DM74ALS245A	4	Bus Tranceiver
D8, D9	DM74LS193N	2	4-bit Counter
D10	SN74F11N	1	3AND
D11,D12, D13	SN74LS181	3	ALU
D14, D15	SN74198N	2	Shift Register
D18	SN7404N	1	NOT
	Роз¶єми		
X1	Роз©ём WH-2 (WF-02S)	1	
X2	Роз©ём WH-10 (WF-10S)	1	
X3	Роз©ём WH-9 (WF-09S)	1	

Зм.	Арк.	υ докум.	Підп.	Дата	
Розробив		Пилипів Р. С.			**Комбінаційна схема** **Схема електрична принципова** **Перелік елеменів**
Керівник		Клушин Ю.С.			
Консульт.					
Н.контр.					
Зав. каф.					

Літера	Аркуш	Аркушів
Лр	2	2

ІКТА, каф.ЕОМ, гр.КІ-25

4.2. Курсове проектування спеціалізованого обчислювача в программному середовище ACTIVE HDL

4.2.1 Мікропрограма роботи МПА.

№	N	Формула
10	11	$Yi = 5 \cdot X_i / 8 – X_{i+1} / 4$

В даному випадку:

A0-A28– стани мікро програмного автомату;

S0 – S18– дія, яка виконується при переході по мікропрограмі з одного стану в інший;

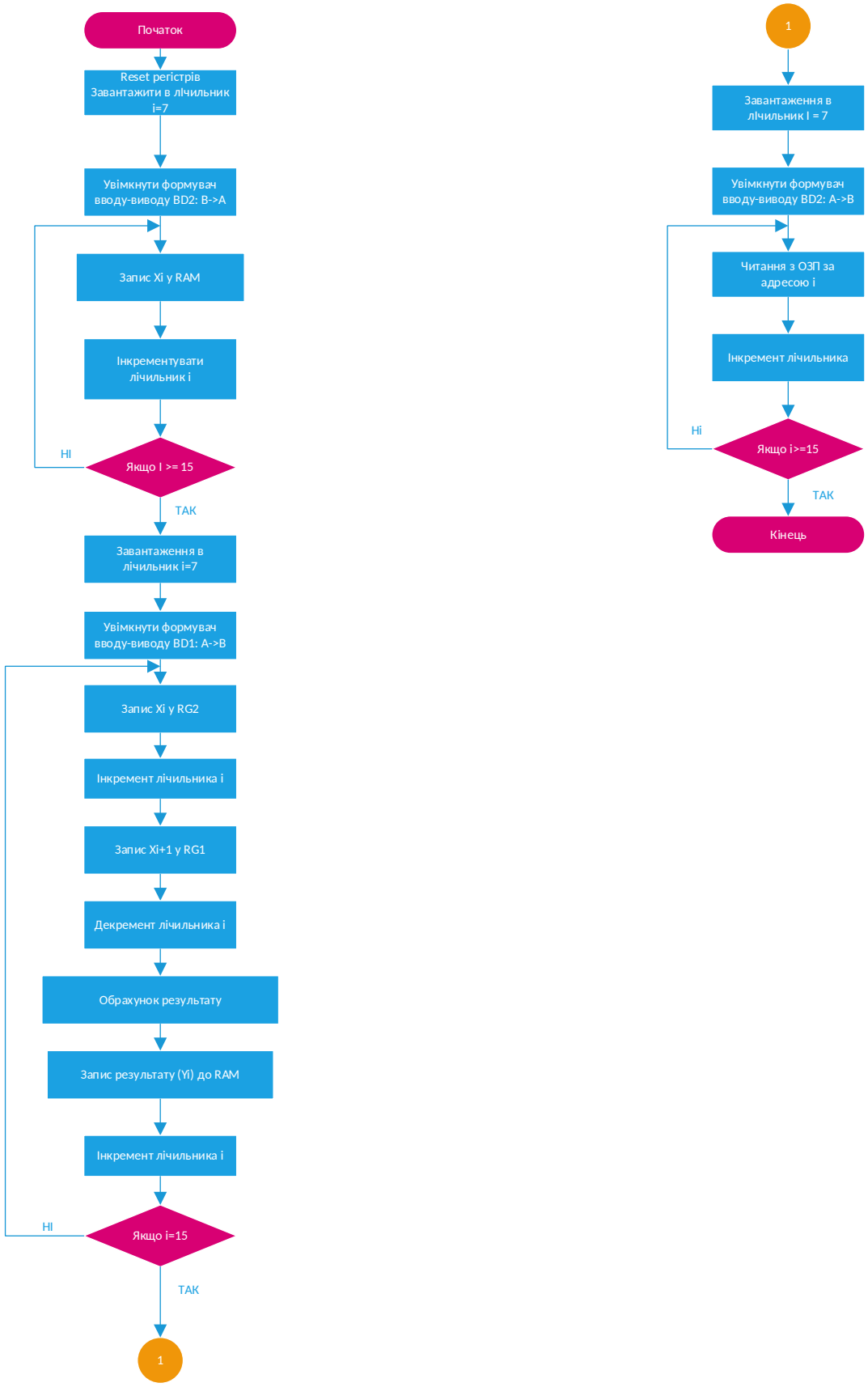
C0 – C2 – умови переходу з одного стану в інший.

Мікропрограма обчислювального автомата

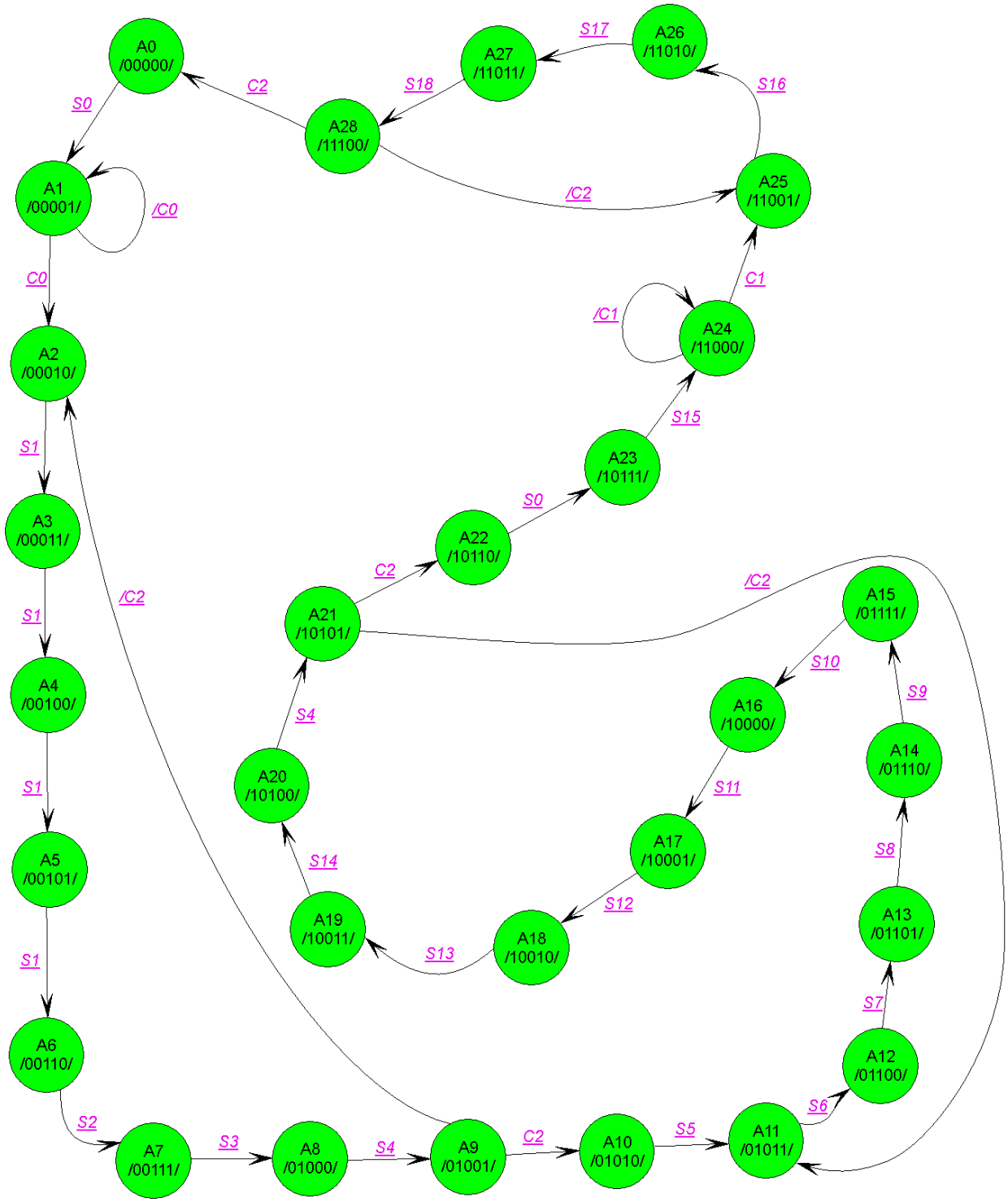
A0:	/load	йти до A1
A1:	якщо wr_rdy	йти до A2, інакше йти до A1
A2:	/WEnotRE, /EZA_2, WR_ACK	йти до A3
A3:	/WEnotRE, /EZA_2, WR_ACK	йти до A4
A4:	/WEnotRE, /EZA_2, WR_ACK	йти до A5
A5:	/WEnotRE, /EZA_2, WR_ACK	йти до A6
A6:	WEnotRE, EZA_2, /WR_ACK	йти до A7
A7:	INC	йти до A8
A8:	/INC	йти до A9
A9:	Якщо P_15	йти до A10, інакше A2
A10:	/load, EZA_2	йти до A11
A11:	Load	йти до A12
A12:	ER2	йти до A13
A13:	/ER2,INC	йти до A14
A14:	/DEC	йти до A15
A15:	ER1	йти до A16
A16:	/ER1	йти до A17
A17:	DEC	йти до A18

A18:	/INC,/WEnotRE, /EZB_1	Йти до A19
A19:	INC,WEnotRE	Йти до A20
A20:	/INC	Йти до A21
A21:	Якщо P_15	Йти до A22, інакше A11
A22:	/load	Йти до A23
A23:	Load, rd_rdy, WEnotRE	Йти до A24
A24:	Якщо rd_ack,	Йти до A25, інакше A24
A25:	/EZB_2, WEnotRE	Йти до A26
A26	/EZB_2, WEnotRE, INC	Йти до A27
A27	/EZB_2, WEnotRE, /INC	Йти до A28
A28	Якщо P_15	Йти до A0, інакше A25

4.1.2. Блок-схема алгоритму



4.2.3. Граф-схема МПА



S0:	/load
S1:	/WEnotRE, /EZA_2, WR_ACK
S2:	WEnotRE, EZA_2, /WR_ACK
S3:	INC
S4:	/INC
S5:	/load, EZA_2
S6:	Load
S7:	ER1
S8:	/ER1,INC
S9:	/DEC
S10:	ER2
S11:	/ER2
S12:	DEC
S13:	/INC,/WEnotRE, EZB_1
S14:	INC,WEnotRE
S15:	Load, rd_rdy, WEnotRE
S16:	/EZB_2, WEnotRE
S17:	/EZB_2, WEnotRE, INC
S18:	/EZB_2, WEnotRE, /INC
C0:	Wr_rdy
C1:	Rd_ack
C2:	P_15

4.2.4 Формування прошивки ПЗП керуючого автомату

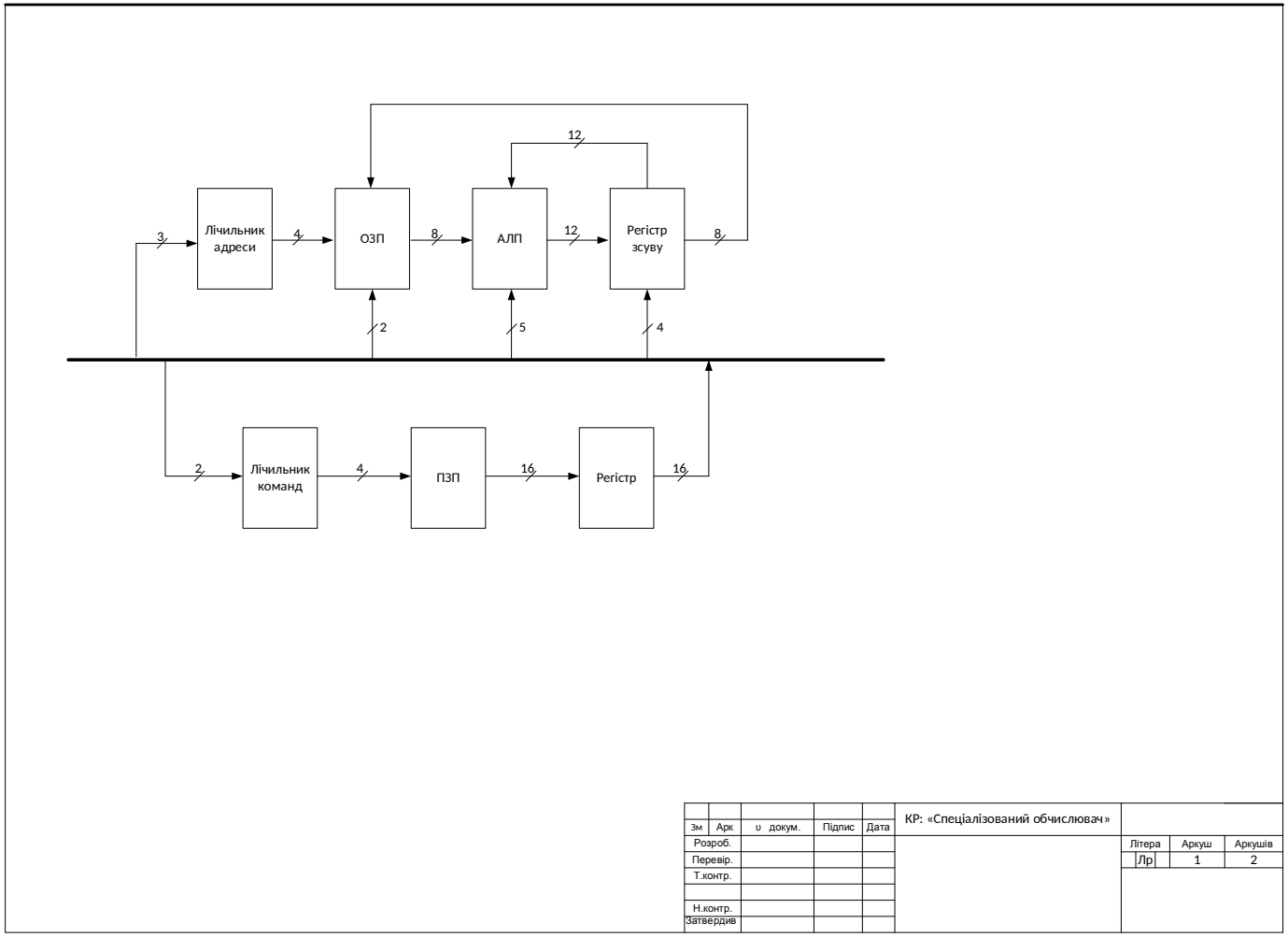
Для занесення інформації в ПЗП необхідно скласти таблицю прошиття, яка встановлює відповідність між адресами і даними. Занесення інформації в ПЗП здійснюється користувачем за допомогою програматора.(див. додаток).

4.2.5 Результати моделювання у вигляді часових діаграм

Часові діаграми роботи схеми наведені у додатку.

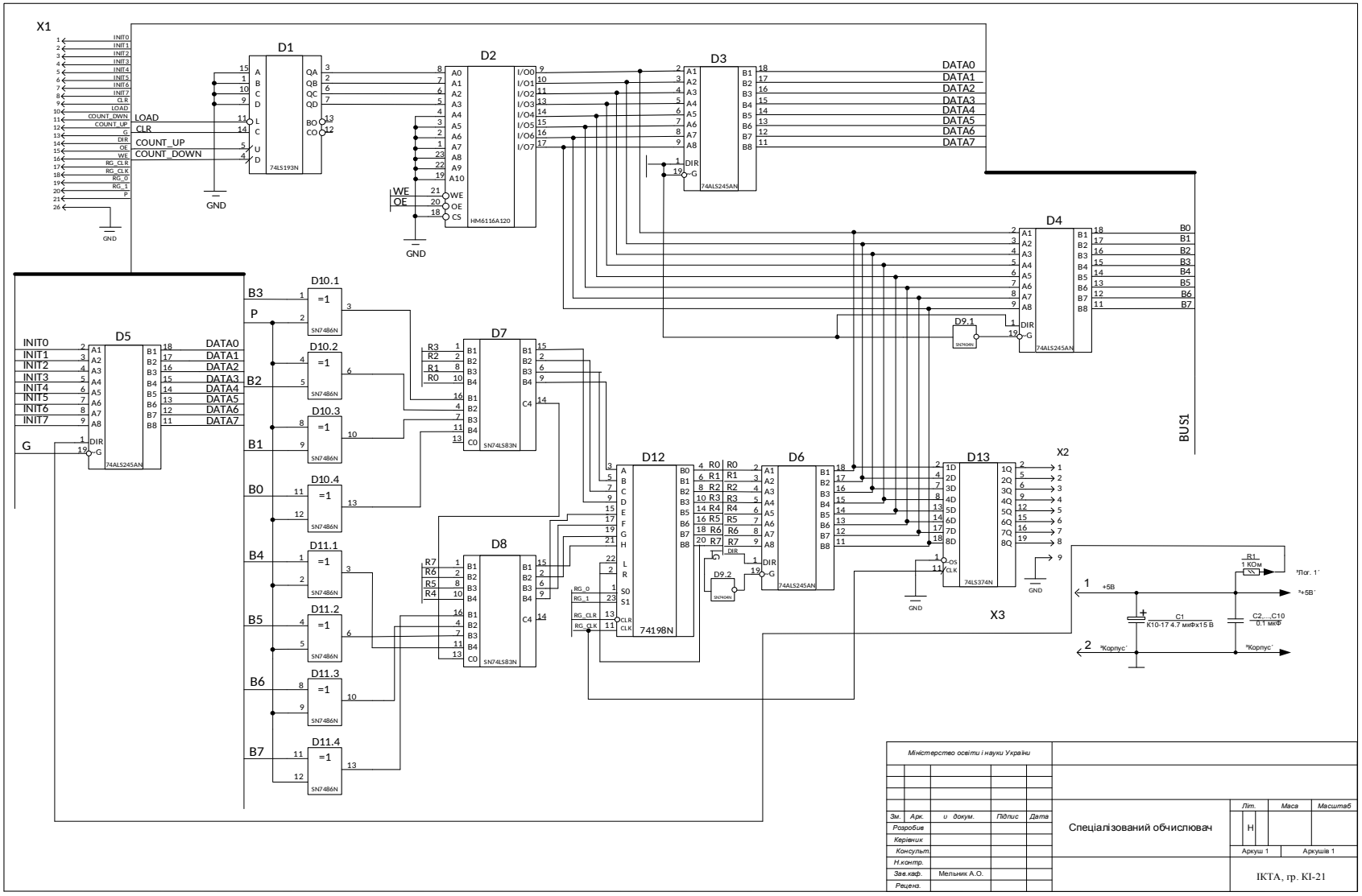
Додатки

Структурна схема



Зм	Арк	ц докум.	Підпис	Дата	КР: «Спеціалізований обчислювач»		
Розроб.						Літера	Аркуш	Аркушів
Перевір.						Лр	1	2
Т.контр.								
Н.контр.								
Затвердив								

Принципова схема



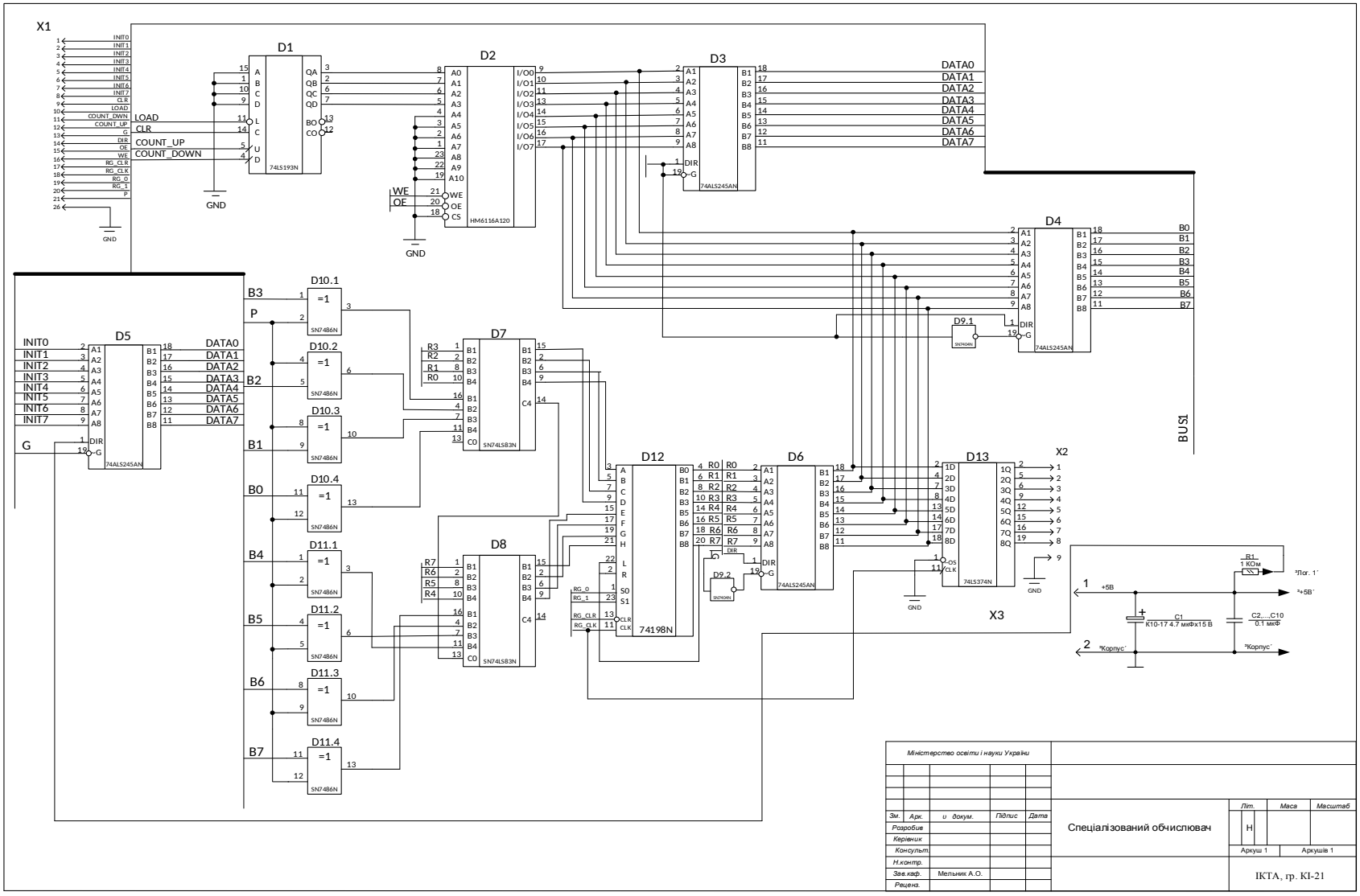
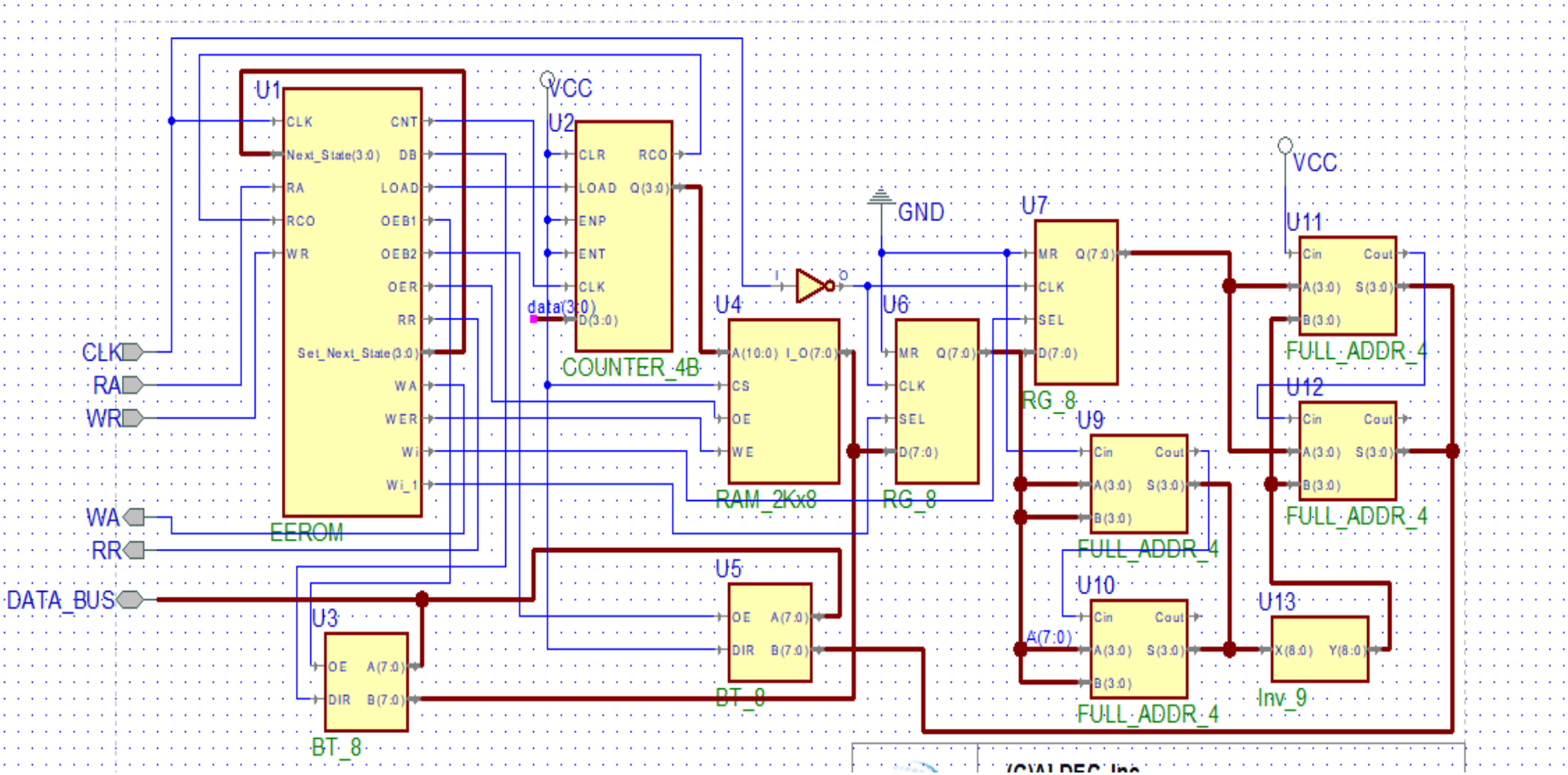
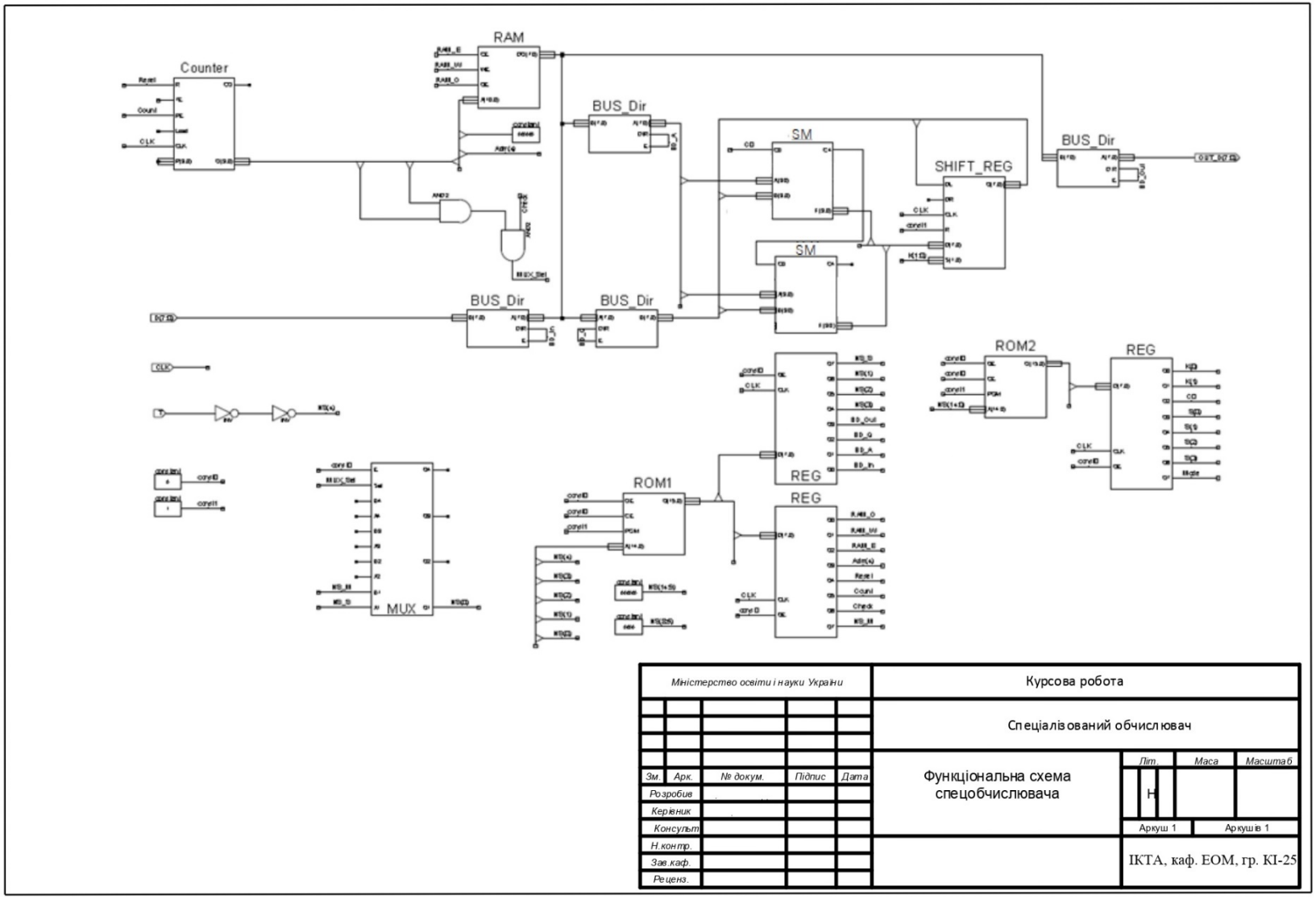

X1
D1
D2
D3
D4
D5
D7
D8
D9.1
D9.2
D10.1
D10.2
D10.3
D10.4
D11.1
D11.2
D11.3
D11.4
D12
D13
X2
X3
LOAD
CLR
COUNT_UP
COUNT_DOWN
GND
WE
OE
DATA0
DATA1
DATA2
DATA3
DATA4
DATA5
DATA6
DATA7
BUS1
74LS193N
HM6116A120
74ALS245AN
SN74LS83N
SN7486N
74198N
74LS374N
Міністерство освіти і науки України
Спеціалізований обчислювач
ІКТА, гр. КІ-21

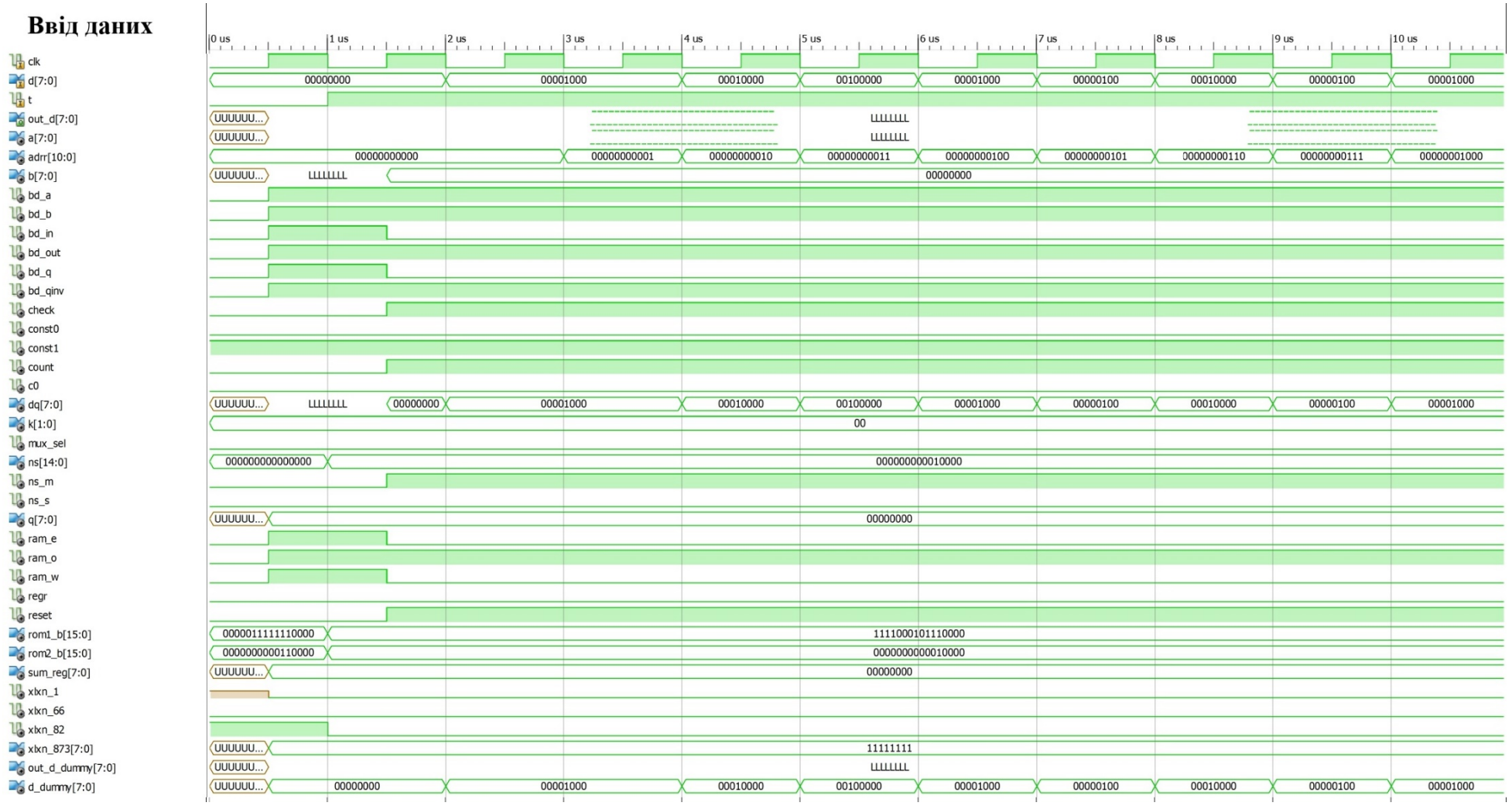
Схема у Active-HDL



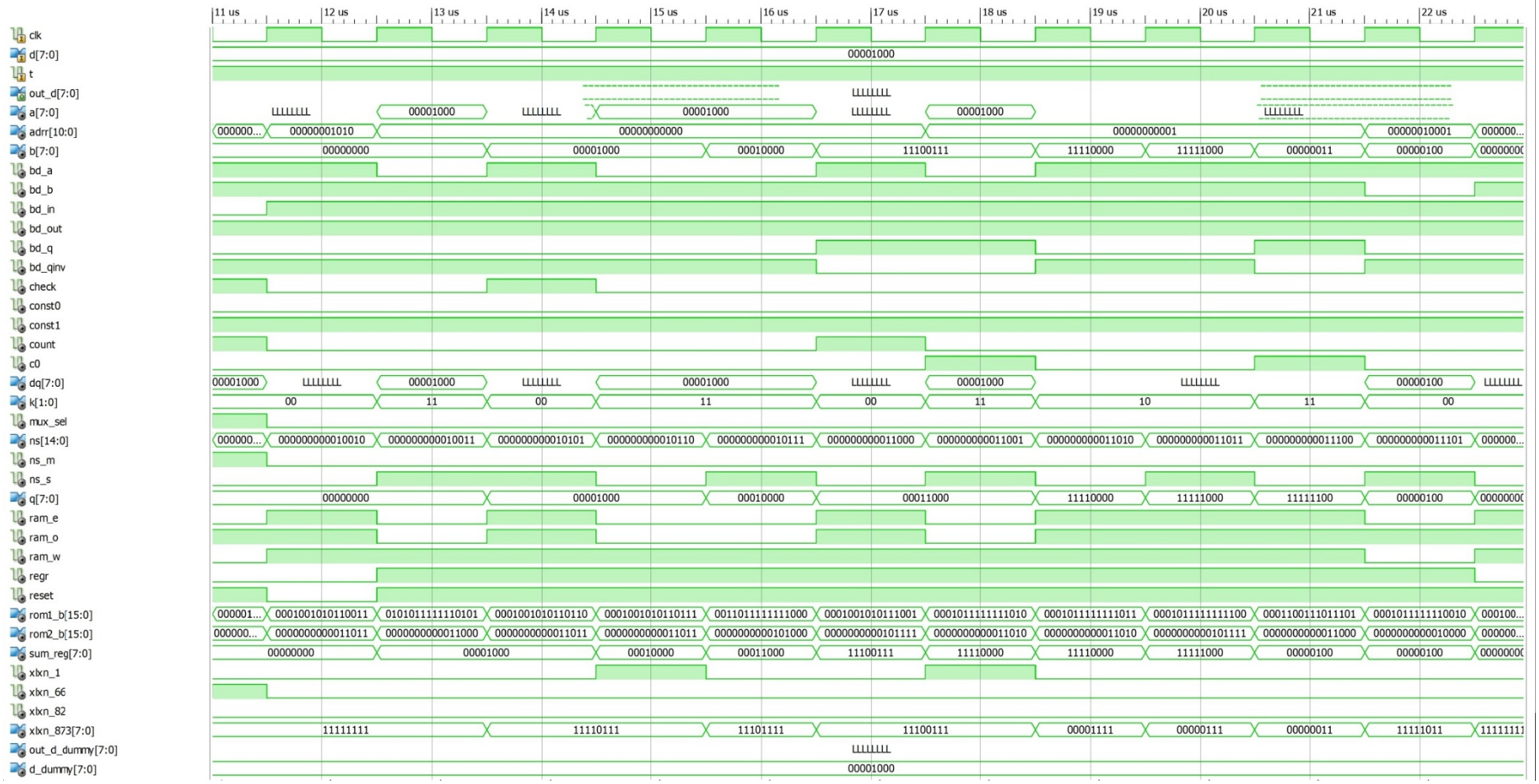
Модель VHDL

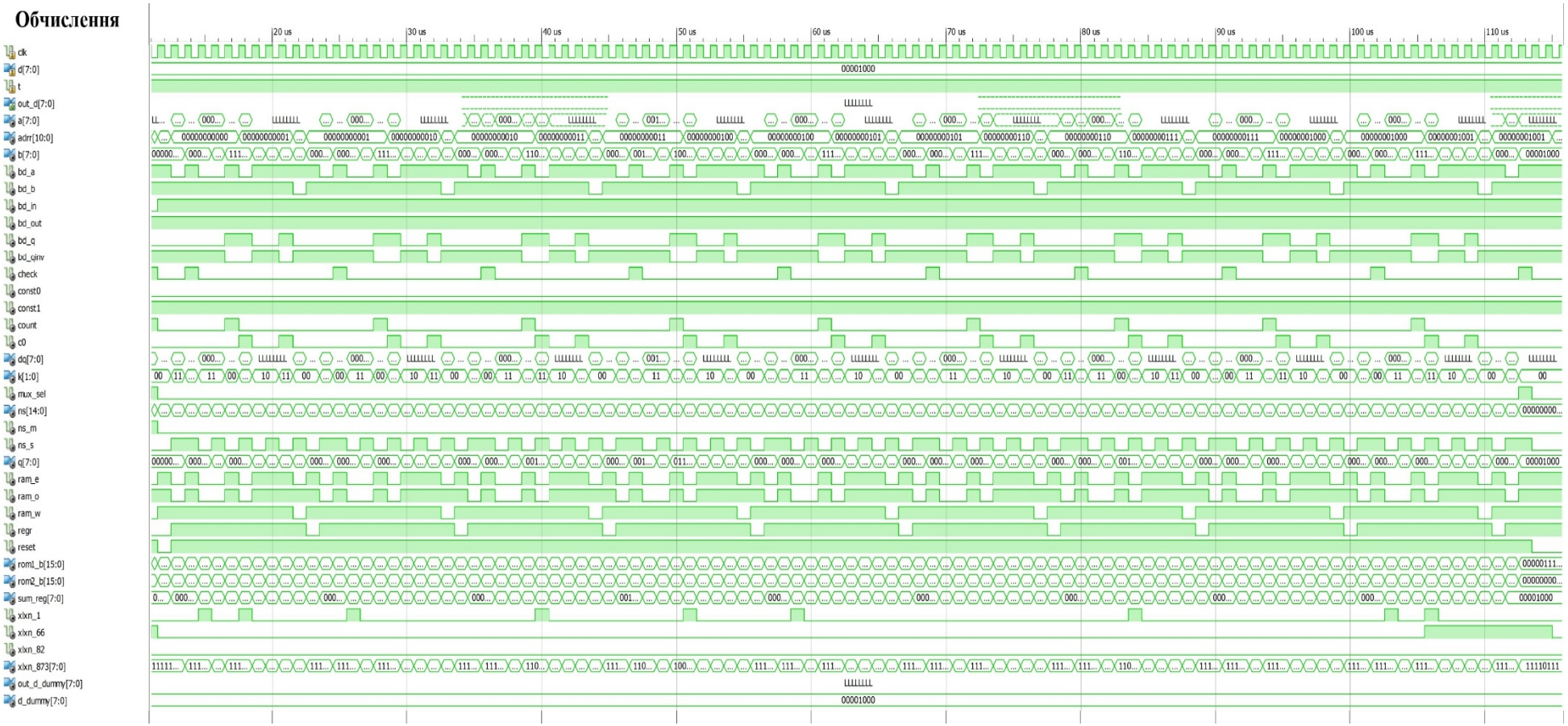

RAM
Counter
BUS_Dir
SM
SHIFT_REG
BUS_Dir
BUS_Dir
BUS_Dir
SM
ROM2
REG
ROM1
REG
MUX
Міністерство освіти і науки України
Курсова робота
Спеціалізований обчислювач
Функціональна схема спецобчислювача
Літ.
Маса
Масштаб
Зм.
Арк.
№ докум.
Підпис
Дата
Розробив
Керівник
Консультт
Н.контр.
Зав.каф.
Реценз.
Аркуш 1
Аркушів 1
ІКТА, каф. ЕОМ, гр. КІ-25

Результати моделювання у вигляді часових діаграм

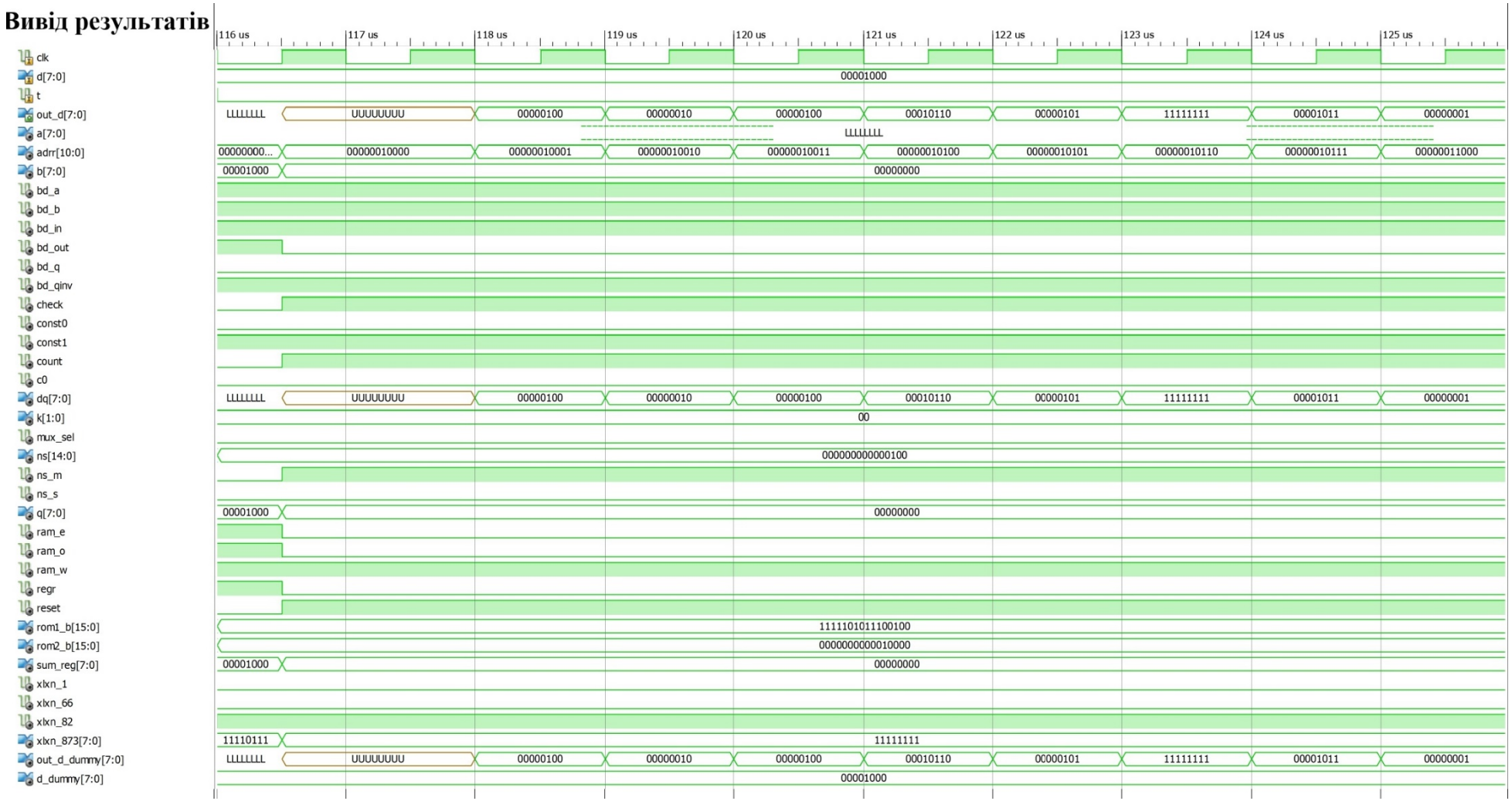


Обчислення одного виразу




Обчислення
20 us
30 us
40 us
50 us
60 us
70 us
80 us
90 us
100 us
110 us
clk
d[7:0]
t
out_d[7:0]
a[7:0]
adrr[10:0]
b[7:0]
bd_a
bd_b
bd_in
bd_out
bd_q
bd_qinv
check
const0
const1
count
c0
dq[7:0]
k[1:0]
mux_sel
ns[14:0]
ns_m
ns_s
q[7:0]
ram_e
ram_o
ram_w
regr
reset
rom1_b[15:0]
rom2_b[15:0]
sum_reg[7:0]
xlxn_1
xlxn_66
xlxn_82
xlxn_873[7:0]
out_d_dummy[7:0]
d_dummy[7:0]
00001000

Вивід результатів



VHDL-код:

counter.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity COUNTER_4B is
	port(
		CLR  : in  std_logic;
		LOAD : in  std_logic;
		ENP  : in  std_logic;
		ENT  : in  std_logic;
		CLK  : in  std_logic;
		D    : in  std_logic_vector(3 downto 0);
		RCO  : out std_logic;
		Q    : out std_logic_vector(3 downto 0));
end COUNTER_4B;

architecture COUNTER_4B of COUNTER_4B is
signal val	: std_logic_vector (3 downto 0);-- <= (others => "0000000");
begin
	process (clk,clr,load,enp,ent,d)
	begin
		if (clr = '0') then val <="0000";
		elsif (clk'event  and clk = '1') then

			if(ENP = '1' and ENT ='1') then val<=val+'1';
			end if;
		elsif (LOAD = '0') then val <= D;
		end if;
	end process;
	process (val)
	begin
		Q <= val;
		if (val = "1111") then RCO <= '1';
		else RCO <= '0';
		end if;
	end process;

end COUNTER_4B;
```

ram.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity RAM_2Kx8 is
	port(
```

```
                A : in std_logic_vector (10 downto 0);
                I_O : inout std_logic_vector (7 downto 0);
                CS : in std_logic;
                OE : in std_logic;
                WE : in std_logic
        );
end RAM_2Kx8;

architecture RAM_2Kx8 of RAM_2Kx8 is
type ram2Kx8 is array (2047 downto 0) of std_logic_vector (7 downto 0);
signal RAM    : ram2Kx8;-- <= (others => "0000000");
begin
        process (cs, a, I_O, oe, we)
        begin
                if (cs ='1'        or (oe = '1' and we ='1')) then
                        I_O <= (others => 'Z');
                elsif (cs='0' and oe = '0' and we = '1') then
                        I_O <= RAM (conv_integer(A));
                elsif (cs='0' and we = '0')       then
                        RAM (conv_integer(A)) <=  I_O;
                end if;
        end process;
end RAM_2Kx8;
```

rg.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity RG_8 is
        port(
                MR  : in  std_logic; --reset
                CLK : in  std_logic; --clock
                SEL : in  std_logic; --load
                D       : in  std_logic_vector (7 downto 0);  --Input
                Q   : out std_logic_vector (7 downto 0) );    --Output
end RG_8;

architecture RG_8 of RG_8 is
begin
        process (CLK,D)
        begin
                if rising_edge(clk) then
                        if(MR = '1') then Q <= (others => '0');
                        elsif SEL='0' then Q <=D;
        end if;
        end if;
        end process;
```

```
--      process (clk)
--      begin
--              Q <=val;
--      end process;
end RG_8;
```

full addr.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity RG_8 is
        port(
                MR  : in  std_logic; --reset
                CLK : in  std_logic; --clock
                SEL : in  std_logic; --load
                D       : in  std_logic_vector (7 downto 0);  --Input
                Q   : out std_logic_vector (7 downto 0) );    --Output
end RG_8;

architecture RG_8 of RG_8 is
begin
        process (CLK,D)
        begin
                if rising_edge(clk) then
                        if(MR = '1') then Q <= (others => '0');
                        elsif SEL='0' then Q <=D;
        end if;
        end if;
        end process;
--      process (clk)
--      begin
--              Q <=val;
--      end process;
end RG_8;
```

bt.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity BT_8 is
        port(
                A   : inout std_logic_vector (7 downto 0):="ZZZZZZZZ";
                B   : inout std_logic_vector (7 downto 0);
                OE  : in std_logic:='1';
                DIR : in std_logic );
end BT_8;
```

```
architecture BT_8 of BT_8 is
begin
        process (OE, DIR,A,B)
        begin
                if (OE = '1') then A <= (others =>'Z');          B <= "ZZZZZZZZ";
                elsif (oe='0' and DIR = '1') then A <=B;
                elsif (oe='0' and dir='0') then B <= A;
                end if;
        end process;
end BT_8;
```

eprom.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity EPROM_64Kx16b is
        port(
        A  : in  std_logic_vector (15 downto 0);
        O  : out std_logic_vector (0 to 15);
        CS : in  std_logic;
        OE : in  std_logic );
end EPROM_64Kx16b;

architecture EPROM_64Kx16b of EPROM_64Kx16b is
--type ram2Kx8 is array (2047 downto 0) of std_logic_vector (7 downto 0);
--signal RAM: ram2Kx8;-- <= (others => "0000000");
--      constant EROM array  of std_logic
begin

        O <=
                "0011011111110000"when A(3 downto 0) = "0000"          and A(4)='1' else
                "0010011111110001"when A(3 downto 0) = "0000" and A(4)='0' else
                "0011011111110001"when A(3 downto 0) = "0001" and A(4)='1' else
                "0001011011110010"when A(3 downto 0) = "0001" and A(4)='0' else
                "0001001011110011"when A(3 downto 0) = "0010" else
                "0001011011110100"when A(3 downto 0) = "0011" else
                "0011111111110101"when A(3 downto 0) = "0100" else
                "0011011111110001"when A(3 downto 0) = "0101" and A(6)='0'else
                "0010011111110110"when A(3 downto 0) = "0101" and A(6)='1'else
                "0011010111010111"when A(3 downto 0) = "0110" else
                "0011010110011000"when A(3 downto 0) = "0111" else
                "0011010111011001"when A(3 downto 0) = "1000" else
```

```
                "0011110111011010"when A(3 downto 0) = "1001" else
                "0011011111111011"when A(3 downto 0) = "1010" else
                "0011011101111011"when A(3 downto 0) = "1011" and A(5)='1'else
                "0011011101101100"when A(3 downto 0) = "1011" and A(5)='0'else
                "0011011110111101"when A(3 downto 0) = "1100" and A(5)='1'else
                "0011011101101100"when A(3 downto 0) = "1100" and A(5)='0'else
                "0011011111111110"when A(3 downto 0) = "1101" else
                "0011111111111111"when A(3 downto 0) = "1110" else
                "0011010111011010"when A(3 downto 0) = "1111" and A(6)='0'else
                "0011011111110000"when A(3 downto 0) = "1111" and A(6)='1'else
                "0011011111110000";
end EPROM_64Kx16b;
```

invertor.vhdl

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Inv_9 is
        port(
                X       : in  std_logic_vector (7 downto 2);  --Input
                Y   : out std_logic_vector (7 downto 2) );    --Output
end Inv_9;

architecture Inv_9 of Inv_9 is
begin
        process (X)
        begin
                Y<=not(X);
        end process;
end Inv_9;
```

top_scheme.vhdl

```
        library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity top_design is
port(
        CLK : in STD_LOGIC;
        RA : in STD_LOGIC;
        WR : in STD_LOGIC;
        RR : out STD_LOGIC;
        WA : out STD_LOGIC;
        DATA_BUS : inout STD_LOGIC_VECTOR (7 downto 0)          );
end entity top_design;
```

```
architecture top_design of top_design is
signal cntrl_signals : std_logic_vector (15 downto 0);
signal i_o_b         : std_logic_vector (7 downto 0);
signal sram_b        : std_logic_vector (7 downto 0):="ZZZZZZZZ";
signal xi1_b         : std_logic_vector (7 downto 0);
signal xi_rg_b       : std_logic_vector (7 downto 0);
signal xi_s_b        : std_logic_vector (16 downto 0);
signal result_b      : std_logic_vector (7 downto 0);
signal sram_adr_b    : std_logic_vector (10 downto 0):="00000000000";
signal cur_state_b   : std_logic_vector (15 downto 0):="0000000000000000";
signal xi1_5_b               : std_logic_vector (8 downto 0);
signal xi_crr        : std_logic;
signal res_crr       : std_logic;
signal RCO                   : std_logic;
signal inv_clk       : std_logic;
constant vcc         : std_logic :='1';
constant gnd         : std_logic :='0';
signal ggnd          : std_logic;
signal start_reset   : std_logic :='0';
-----------------------------------
signal rg_input              : std_logic_vector (7 downto 0);
signal Xi1_AddH_A    : std_logic_vector (3 downto 0);
signal Xi1_AddH_B    : std_logic_vector (3 downto 0);
signal Res_addH              : std_logic_vector (3 downto 0);
-----------------------------------
component EPROM_64Kx16b is
  port(
        A  : in  std_logic_vector (15 downto 0);
             O  : out std_logic_vector (0 to 15);
             CS : in  std_logic;
             OE : in  std_logic
        );
end component;
-----------------------------------
component BT_8
  port (
     DIR : in STD_LOGIC;
     OE : in STD_LOGIC;
     A : inout STD_LOGIC_VECTOR(7 downto 0);
     B : inout STD_LOGIC_VECTOR(7 downto 0)
  );
end component;
-----------------------------------
component COUNTER_4B
  port (
     CLK : in STD_LOGIC;
```

```
      CLR : in STD_LOGIC;
      D : in STD_LOGIC_VECTOR(3 downto 0);
      ENP : in STD_LOGIC;
      ENT : in STD_LOGIC;
      LOAD : in STD_LOGIC;
      Q : out STD_LOGIC_VECTOR(3 downto 0);
      RCO : out STD_LOGIC
  );
end component;
-----------------------------------
component FULL_ADDR_4
  port (
      A : in STD_LOGIC_VECTOR(3 downto 0);
      B : in STD_LOGIC_VECTOR(3 downto 0);
      Cin : in STD_LOGIC;
      Cout : out STD_LOGIC;
      S : out STD_LOGIC_VECTOR(3 downto 0)
  );
end component;
-----------------------------------
component Inv_9
  port (
      X : in STD_LOGIC_VECTOR(7 downto 2);
      Y : out STD_LOGIC_VECTOR(7 downto 2)
  );
end component;
-----------------------------------
component RAM_2Kx8
  port (
      A : in STD_LOGIC_VECTOR(10 downto 0);
      CS : in STD_LOGIC;
      OE : in STD_LOGIC;
      WE : in STD_LOGIC;
      I_O : inout STD_LOGIC_VECTOR(7 downto 0)
  );
end component;
-----------------------------------
component RG_8
  port (
      CLK : in STD_LOGIC;
      D : in STD_LOGIC_VECTOR(7 downto 0);
      MR : in STD_LOGIC;
      SEL : in STD_LOGIC;
      Q : out STD_LOGIC_VECTOR(7 downto 0)
  );
```

```
        end component;

        --*************************************************************************--
        begin


        ------------------------------------------------
--      process(clk)
--      begin
--              if not(start_reset = '1') then
--                      cur_state_b<=(others =>'0');
--                      start_reset <='1';
--              end if;
--      end     process;
        ------------------------------------------------
        process (clk)
        begin
                inv_clk <=not clk;
        end process;
        ------------------------------------------------
        process (cntrl_signals)
        begin
                WA<=cntrl_signals(8);
                RR<=cntrl_signals(7);
        end process;
        ------------------------------------------------
        process (RCO , RA , WR , cntrl_signals (3 downto 0))
        begin
                rg_input <=     ('0' & RCO & RA & WR & cntrl_signals (3 downto 0));
        end process;
        DD1_st_reg : rg_8
        port map(
                CLK => CLK,
                D       => rg_input,
                MR  => GND,
                SEL     => GND,
                Q       => cur_state_b (7 downto 0));
        ------------------------------------------------
        DD2_eprom : EPROM_64Kx16b
        port map(
                A  => cur_state_b,
                O  => cntrl_signals,
                cs => gnd,
                oe => gnd);
        ------------------------------------------------
        DD3_adr_counter :  COUNTER_4B
```

```
port map(
        CLK  => cntrl_signals(11),
        CLR  => VCC,
        D    => "0011",
        ENP  => VCC,
        ENT  => VCC,
        LOAD => cntrl_signals(12),
        Q        =>     sram_adr_b(3 downto 0),
        RCO  =>       RCO);
-------------------------------------------------
DD4_sram : RAM_2Kx8
port map(
        A       => sram_adr_b,
        CS      => GND,
        OE  => cntrl_signals(9),
        WE  => cntrl_signals(10),
        I_O => sram_b);
-------------------------------------------------
DD5_bt_inp : BT_8
port map(
        DIR => GND,
        OE  => cntrl_signals(13),
        A       => data_bus,
        B       => sram_b);
-------------------------------------------------
DD6_Xi1_reg : RG_8
port map(
        CLK => inv_clk,
        D       => sram_b,
        MR      => GND,
        SEL => cntrl_signals(5),
        Q       => xi1_b);
-------------------------------------------------
DD9_Xi_reg : RG_8
port map(
        CLK => inv_clk,
        D       => xi1_b,
        MR      => GND,
        SEL => cntrl_signals(6),
        Q       => xi_rg_b);
-------------------------------------------------
DD7_Xi_addL : FULL_ADDR_4
port map(
        A       => xi_rg_b(6 downto 3),
        B       => xi_rg_b(4 downto 1),
        Cin     => GND,
```

```
        Cout => xi_crr,
        S       => xi_s_b(3 downto 0));
-------------------------------------------------
DD8_Xi_addH : FULL_ADDR_4
port map(
        A(3) => '0',
        A(2) => '0',
        A(1) => '0',
        A(0) => xi_rg_b(7),
        B(3) => '0',
        B(2) => xi_rg_b(7),
        B(1) => xi_rg_b(6),
        B(0) => xi_rg_b(5),
        Cin  => xi_crr,
        Cout => ggnd,
        S       => xi_s_b(7 downto 4));
-------------------------------------------------
DD10_inv : Inv_9
port map(
        X(7 downto 2) => xi1_b(7 downto 2),
        Y(7 downto 2) => xi_s_b (13 downto 8));
-------------------------------------------------
DD11_Res_addL : FULL_ADDR_4
port map(
        A       => xi_s_b(11 downto 8),
        B       => xi_s_b(3 downto 0),
        Cin     => VCC,
        Cout => res_crr,
        S       => result_b(3 downto 0));
-------------------------------------------------
process (xi_rg_b)
begin
        Res_addH<="000" & xi_rg_b(7);
end process;
-------------------------------------------------
DD12_Res_addH : FULL_ADDR_4
port map(
        A(3) => '1',
        A(2) => '1',
        A(1) => xi_s_b(13),
        A(0) => xi_s_b(12),
        B       => xi_s_b(7 downto 4),
        Cin  => res_crr,
        Cout => ggnd,
        S       => result_b(7 downto 4));
-------------------------------------------------
```

```
DD13_bt_out : BT_8
port map(
        DIR => GND,
        OE  => cntrl_signals(4),
        A       => result_b,
        B       => data_bus);
-------------------------------------------------
end architecture top_design;
```

test.vhdl

```
        library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.NUMERIC_STD.all;


entity testbench is
end entity testbench;

architecture testbench of testbench is
----------------------------
component top_design is
port(
        CLK : in STD_LOGIC;
        RA : in STD_LOGIC;
        WR : in STD_LOGIC;
        RR : out STD_LOGIC;
        WA : out STD_LOGIC;
        DATA_BUS : inout STD_LOGIC_VECTOR (7 downto 0)  );
end component;
signal count1,count2 :  integer :=0;
signal WR,WA,RR,RA, CLK : std_logic:='1';
signal io : std_logic_vector (7 downto 0):=(others =>'Z');
signal data : std_logic_vector (7 downto 0):="11110000";
type ram32x8 is array (32 downto 0) of std_logic_vector (7 downto 0);
type ram16x8 is array (11 downto 0) of std_logic_vector (7 downto 0);
signal calc_val          : ram16x8:=(others =>"00000000");
signal inpt_val          : ram16x8:=(others =>"00000000");
type ramx8 is array (26 downto 0) of integer ;
signal tmp  : ramx8:=(others =>0);

begin
        process
```

```
begin
        wait for 10 ns;
        clk<=not clk;
end process;
comp_syst : top_design
port map(
        CLK=> CLK,
        RR=>RR,
        RA=>RA,
        WR=>WR,
        WA=>WA,
        DATA_BUS=>IO);
process
begin
        if count1 = 0 then
                wait for 100ns;
        end if;
        wait for 100ns;
        if(count1<12) then
                WR <= '0';
                wait on wa;
                if wa = '0' then
                        io <= data - count1*3;
                        tmp(count1) <= to_integer(unsigned(data - count1*3));
                        count1<=count1 +1;
                        wait on wa;
                        if wa='1' then
                                io <=(others=>'Z');
                                wr <='1';
                        end if;
                end if;
        end if;
        if(count1 = 12 and count2 <11) then

                        calc_val(count2)  <= std_logic_vector(to_signed(((tmp(count2))/2 +
tmp(count2)/8 - tmp(count2+1)/4),8));
                        wait on rr;
                        if rr='0' then
                                ra <='0';
                                wait on io;
                                inpt_val(count2) <= io;
                                wait on clk;
                                wait on clk;
                                wait on clk;
                                wait on clk;
                                ra<='1';
```

```
                    count2<=count2+1;
                end if;
            end if;
            end process;
end testbench;
```

4.3 Курсове проектування спеціалізованого обчислювача в програмному середовище PROTEUS

4.3.1 Мікропрограма роботи МПА

Згідно індивідуального завдання робоча формула має вигляд:

$Y_i=(X_i-5X_{i+1})/8$

N=12

В даному випадку

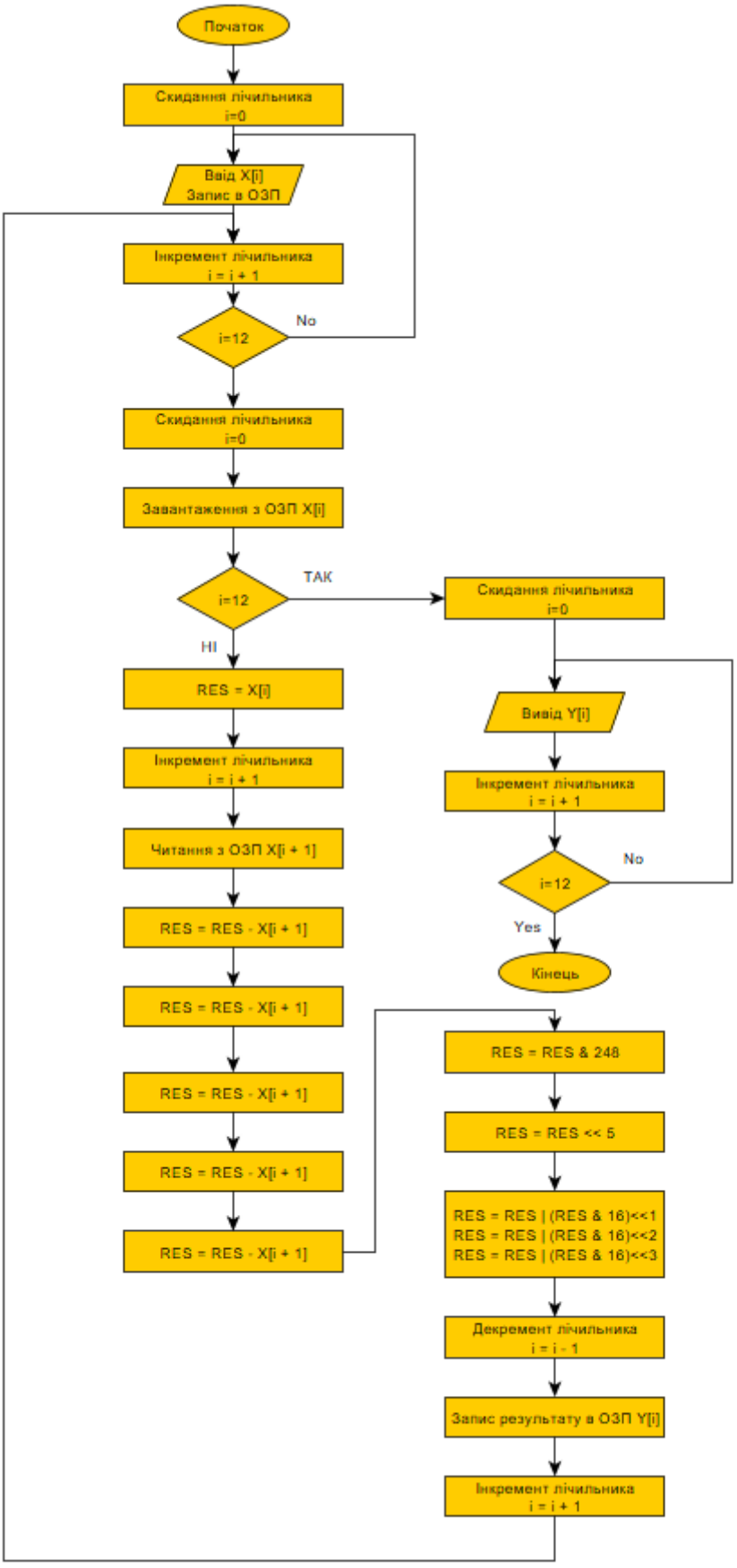
A0 – A39 – стани мікропрограмного автомата

Якщо наступний стан без вказаного рівня сигналу ENTER, то перехід відбувається при його рівні в лог. 1;

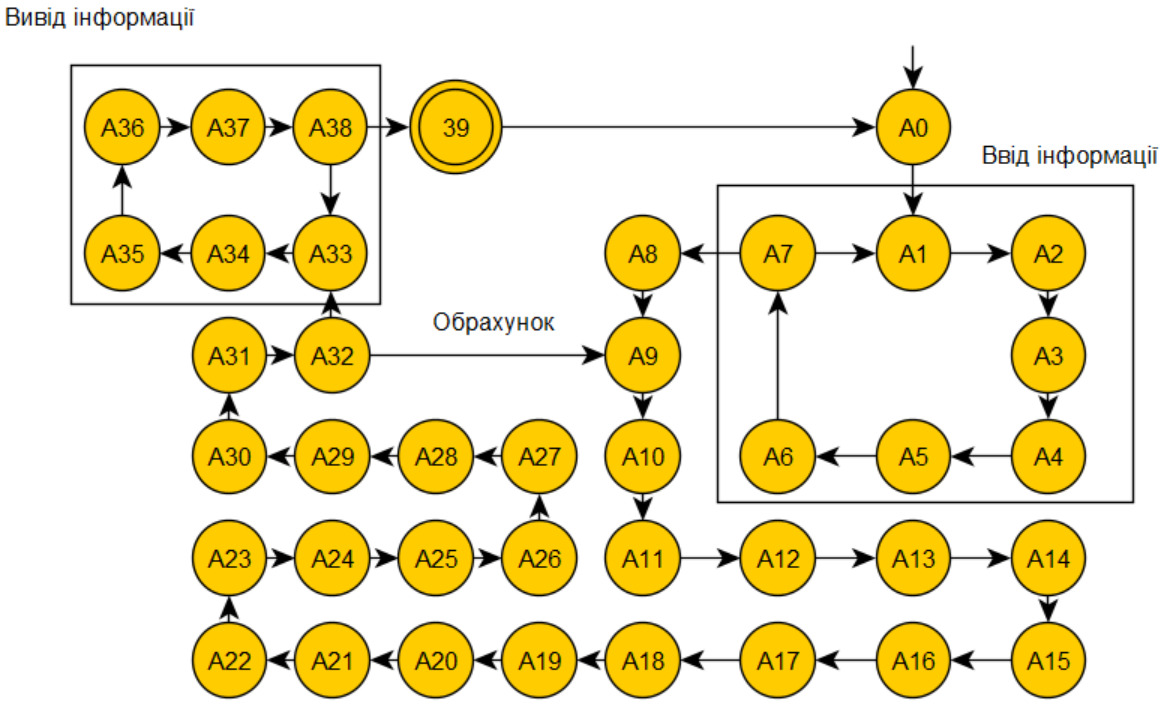
Стан	Дія, що виконується	Наступний стан
A0	CNT,RAM_UP,RAM_DN,RAM_CLR,WE,RE,ALU_IN, ALU_OUT,MSK	A1
A1	RAM_UP,RAM_DN,WE,RE,ALU_IN,ALU_OUT,MSK	В A2 коли подали сигнал ENTER в лог. 0
A2	CNT,RAM_UP,RAM_DN,RE,ALU_IN,ALU_OUT,MSK	A3
A3	CNT,RAM_UP,RAM_DN,WE,RE,ALU_IN,ALU_OUT, MSK	A4
A4	CNT,RAM_UP,RAM_DN,WE,RE,ALU_IN,ALU_OUT, MSK	A5
A5	CNT,RAM_DN,WE,RE,ALU_IN,ALU_OUT,MSK	A6
A6	CNT,RAM_UP,RAM_DN,WE,RE,ALU_IN,ALU_OUT, MSK	A7
A7	CNT,RAM_UP,RAM_DN,WE,RE,ALU_IN,ALU_OUT, MSK	Якщо лічильник = 12 в A8, інакше в A1
A8	RAM_UP,RAM_DN,RAM_CLR,MODE,S1,S3,WE, ALU_OUT,INPUT,MAIN_DR,MSK	В A9 коли подали сигнал ENTER в лог. 0
A9	CNT,RAM_UP,RAM_DN,MODE,S1,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A10
A10	CNT,RAM_DN,MODE,S1,S3,WE,ALU_OUT,INPUT, MAIN_DR,MSK	A11
A11	CNT,RAM_UP,RAM_DN,CN,S1,S2,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A12
A12	CNT,RAM_UP,RAM_DN,CN,S1,S2,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A13
A13	CNT,RAM_UP,RAM_DN,CN,S1,S2,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A14
A14	CNT,RAM_UP,RAM_DN,CN,S1,S2,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A15
A15	CNT,RAM_UP,RAM_DN,CN,S1,S2,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A16
A16	CNT,RAM_UP,RAM_DN,MODE,S0,S2,S3,WE, ALU_OUT,INPUT,MAIN_DR,MAIN_G,MSK1	A17

A17	CNT,RAM_UP,RAM_DN,SHIFT,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A18
A18	CNT,RAM_UP,RAM_DN,SHIFT,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A19
A19	CNT,RAM_UP,RAM_DN,SHIFT,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A20
A20	CNT,RAM_UP,RAM_DN,SHIFT,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A21
A21	CNT,RAM_UP,RAM_DN,SHIFT,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A22
A22	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S3,WE, ALU_OUT,INPUT,MAIN_DR,MAIN_G,MSK2	A23
A23	CNT,RAM_UP,RAM_DN,S0,S1,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,SIGN_DIV,MSK	A24
A24	CNT,RAM_UP,MODE,S0,S1,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A25
A25	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,WE, ALU_OUT,INPUT,MAIN_DR,MSK	A26
A26	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,WE,RE, ALU_IN,INPUT,MSK	A27
A27	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,RE, ALU_IN,INPUT,MSK	A28
A28	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,WE,RE, ALU_IN,INPUT,MSK	A29
A29	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,WE,RE, ALU_IN,INPUT,MAIN_DR,MSK	A30
A30	CNT,RAM_DN,MODE,S0,S1,S2,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A31
A31	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,WE, ALU_OUT,INPUT,MAIN_DR,MSK	A32
A32	CNT,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3,WE, ALU_OUT,INPUT,MAIN_DR,MSK	Якщо лічильник = 12 в A33, інакше в A9
A33	CNT,RAM_UP,RAM_DN,RAM_CLR,MODE,S0,S1,S2, S3,WE,ALU_OUT,INPUT,MAIN_DR,MSK	A34
A34	CNT,RAM_UP,RAM_DN,MODE,S1,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A35
A35	RAM_UP,RAM_DN,MODE,S1,S3,WE,ALU_OUT, INPUT,MAIN_DR,OUT_SIGNAL,MSK	В A36 коли подали сигнал ENTER в лог. 0
A36	CNT,RAM_UP,RAM_DN,MODE,S1,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	A37
A37	CNT,RAM_DN,MODE,S1,S3,WE,ALU_OUT,INPUT, MAIN_DR,MSK	A38
A38	CNT,RAM_UP,RAM_DN,MODE,S1,S3,WE,ALU_OUT, INPUT,MAIN_DR,MSK	Якщо лічильник = 12 в A39, інакше в A34
A39	CNT,RESET,RAM_UP,RAM_DN,MODE,S0,S1,S2,S3, WE,ALU_IN,ALU_OUT,INPUT,MAIN_DR,MSK	A0

4.3.2 Блок-схема алгоритму



4.3.3 Граф-схема МПА



4.3.4 Формування прошивки ПЗП керуючого автомату

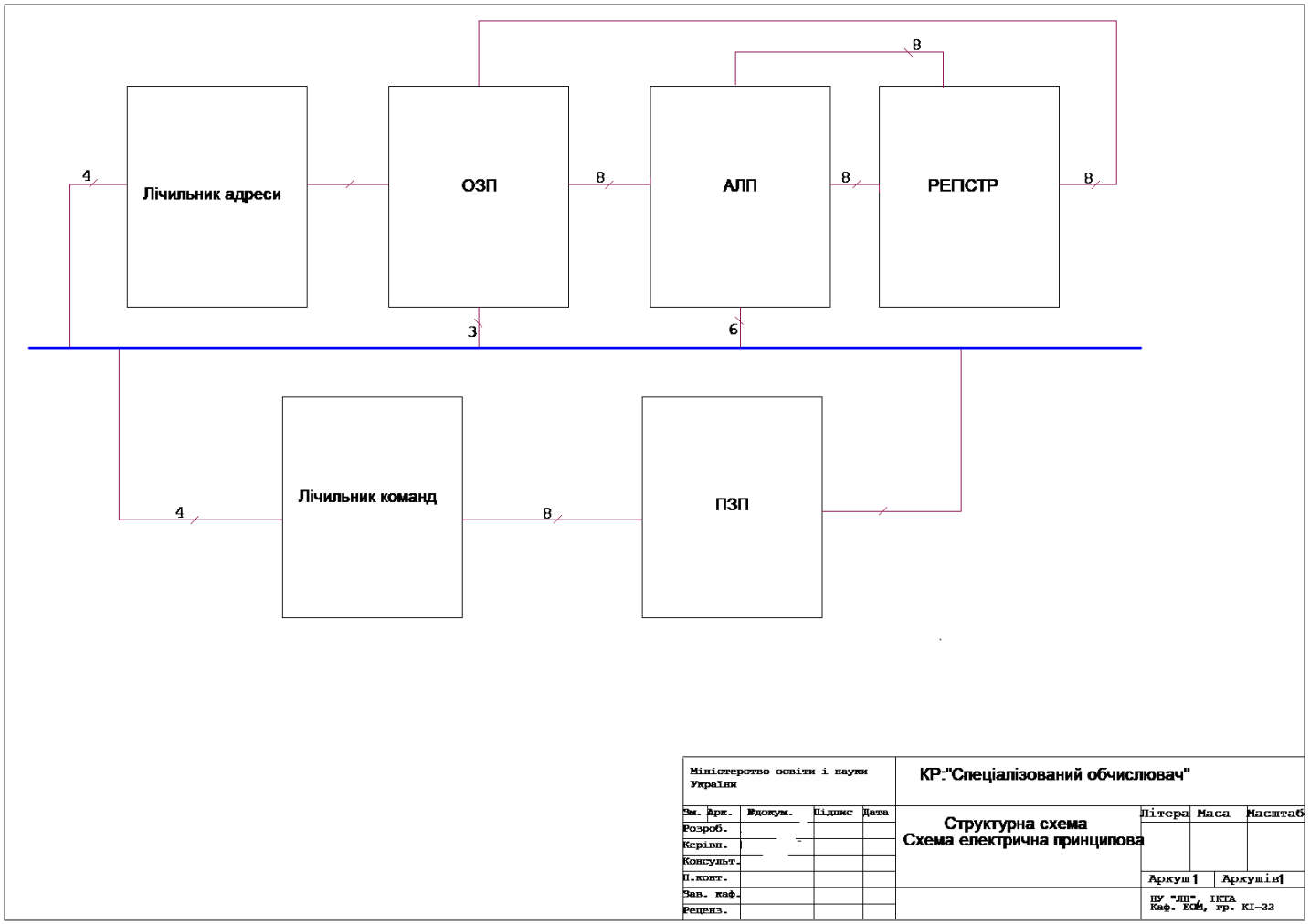
Для занесення інформації в ПЗП необхідно скласти таблицю прошиття, яка встановлює відповідність між адресами і даними. Занесення інформації в ПЗП здійснюється користувачем за допомогою програматора.(див. додаток).

4.3.5 Результати моделювання у вигляді часових діаграм

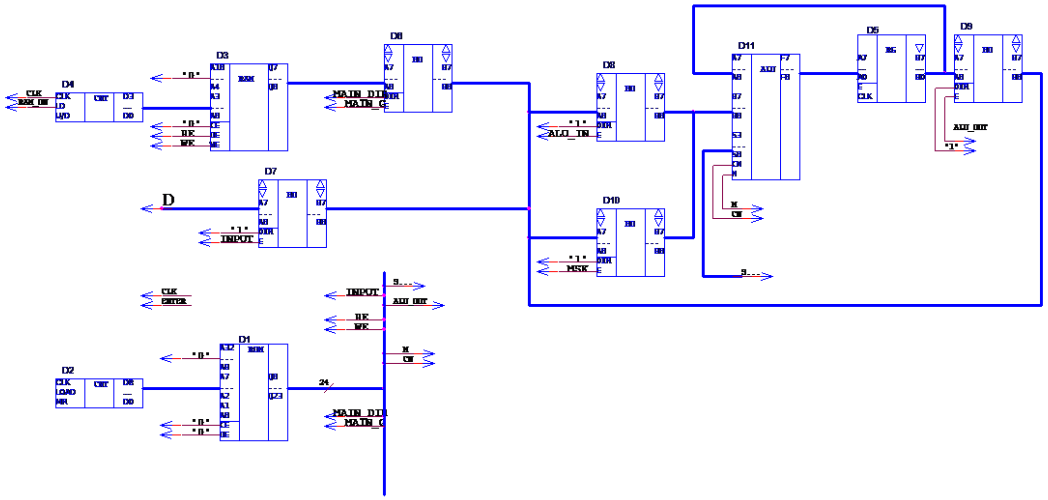
Часові діаграми роботи схеми наведені у додатку.

Додатки

Структурна схема

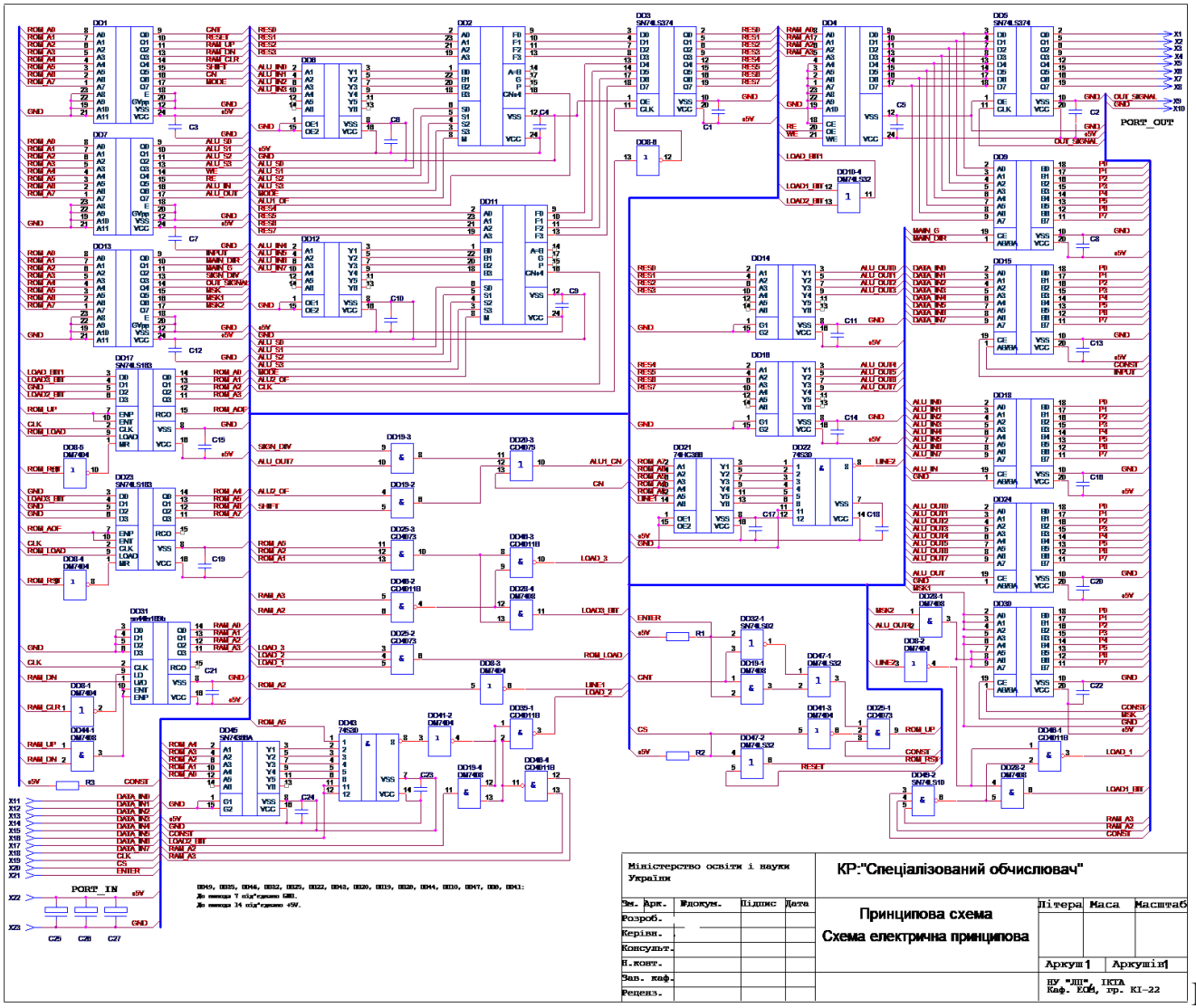


Функцональна схема

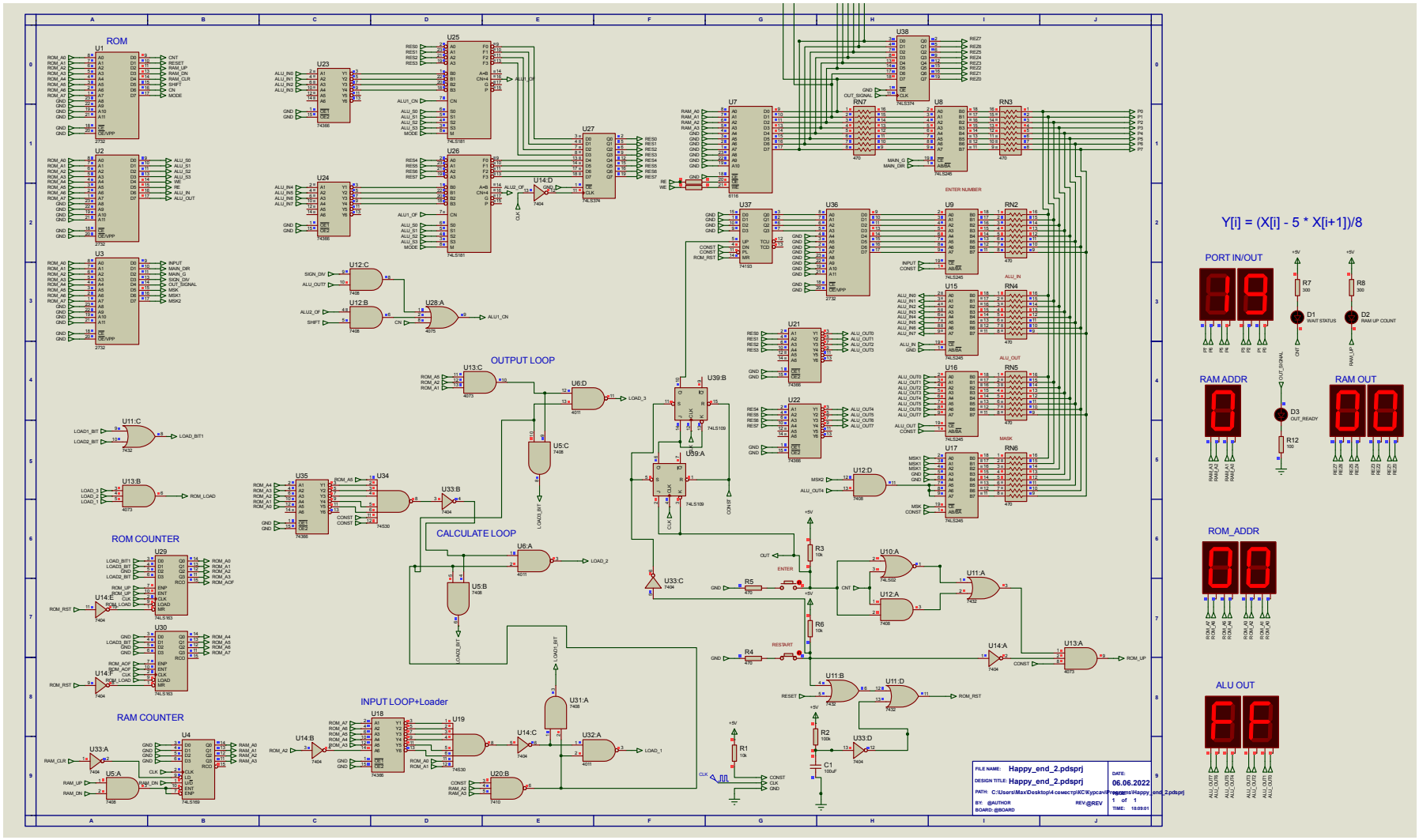


Міністерство освіти і науки України					КР:"Спеціалізований обчислювач"			
Зм.	Арк.	№докум.	Підпис	Дата	Функціональна схема Схема електрична принципова	Літера	Маса	Масштаб
Розроб.								
Керівн.						Аркуш 1	Аркушів 1	
Консульт.								
Н.контр.								
Зав. каф.						НУ "ЛП", ІКТА Каф. ЕОМ, гр. КІ-22		
Реценз.								

Принципована схема



Комбінаційна схема


ROM
OUTPUT LOOP
CALCULATE LOOP
ROM COUNTER
INPUT LOOP+Loader
RAM COUNTER
ENTER NUMBER
ALU_IN
ALU_OUT
MASK
Y[i] = (X[i] - 5 * X[i+1])/8
PORT IN/OUT
RAM ADDR
RAM OUT
ROM_ADDR
ALU OUT
FILE NAME: Happy_end_2.pdsprj
DESIGN TITLE: Happy_end_2.pdsprj
DATE: 06.06.2022
TIME: 18:09:01

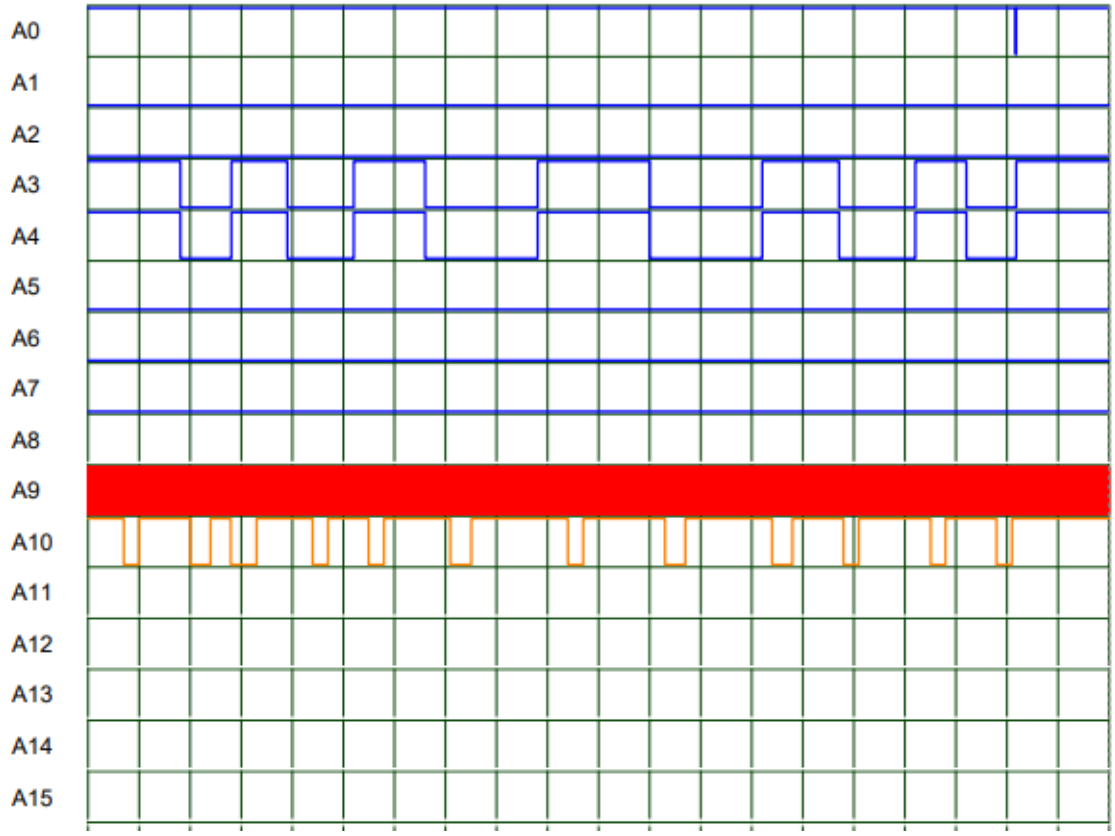
Таблиця прошиття

	Поточний стан								Керуючі сигнали																							Наступний стан								Опис дії
	A7	A6	A5	A4	A3	A2	A1	A0	CNT	RST	RAM_UP	RAM_DN	RAM_CLR	SHIFT	CN	MODE	S0	S1	S2	S3	WE	RE	ALU_IN	ALU_OUT	INPUT	MAIN_DR	MAIN_G	SIGN_DIV	OUT_SIGNAL	MSK	MSK1	MSK2	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	Підготовка
1	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	Ввід X[i]
2	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	
3	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
4	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	
5	0	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	
6	0	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	
7	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
8	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	i = 0
9	0	0	0	0	1	0	0	1	1	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0	1	
10	0	0	0	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	1	Y[i] = X[i]
11	0	0	0	0	1	0	1	1	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0	Y[i] -= X[i+1]
12	0	0	0	0	1	1	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1	Y[i] -= X[i+1]
13	0	0	0	0	1	1	0	1	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	1	0	Y[i] -= X[i+1]
14	0	0	0	0	1	1	1	0	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	1	1	Y[i] -= X[i+1]
15	0	0	0	0	1	1	1	1	1	0	1	1	0	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	Y[i] -= X[i+1]
16	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	1	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	Y[i]&=/3
17	0	0	0	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	Циклічний здвиг Y[i] вліво на 5 позицій
18	0	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
19	0	0	0	1	0	0	1	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	
20	0	0	0	1	0	1	0	0	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1	
21	0	0	0	1	0	1	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	1	1	0	
22	0	0	0	1	0	1	1	0	1	0	1	1	0	0	0	1	1	1	0	1	1	0	0	1	1	1	1	0	0	0	0	1	0	0	0	1	0	1	1	1	Повернення знаку
2	0	0	0	1	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	1	0	1	0	0	0	0	0	1	1	0	0	0	

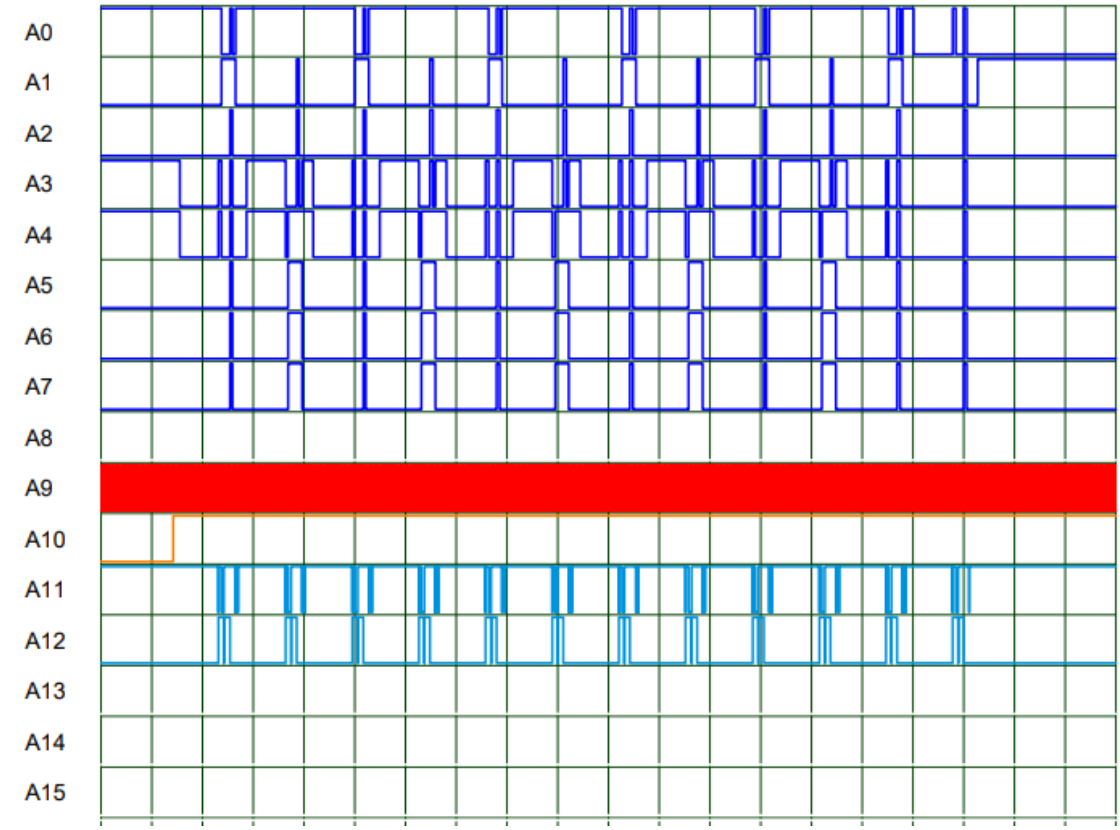
3																																							
24	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	
25	0	0	0	1	1	0	0	1	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	для старших
26	0	0	0	1	1	0	1	0	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	3-х знаків
27	0	0	0	1	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	Запис Y[i]
28	0	0	0	1	1	1	0	0	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	1	
29	0	0	0	1	1	1	0	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0	1	1	1	1	0	
30	0	0	0	1	1	1	1	0	1	0	0	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	
31	0	0	0	1	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	i = i + 1	
32	0	0	1	0	0	0	0	0	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1		
33	0	0	1	0	0	0	0	1	1	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	i = 0	
34	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	1	Вивід Y[i]	
35	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0	0	1	0	0	1	0	0	Вивід Y[i]	
36	0	0	1	0	0	1	0	0	1	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	1	Вивід Y[i]	
37	0	0	1	0	0	1	0	1	1	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	1	1	0	i = i + 1	
38	0	0	1	0	0	1	1	0	1	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	1	1	1		
39	0	0	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	Скид автомата	

Часові діаграми

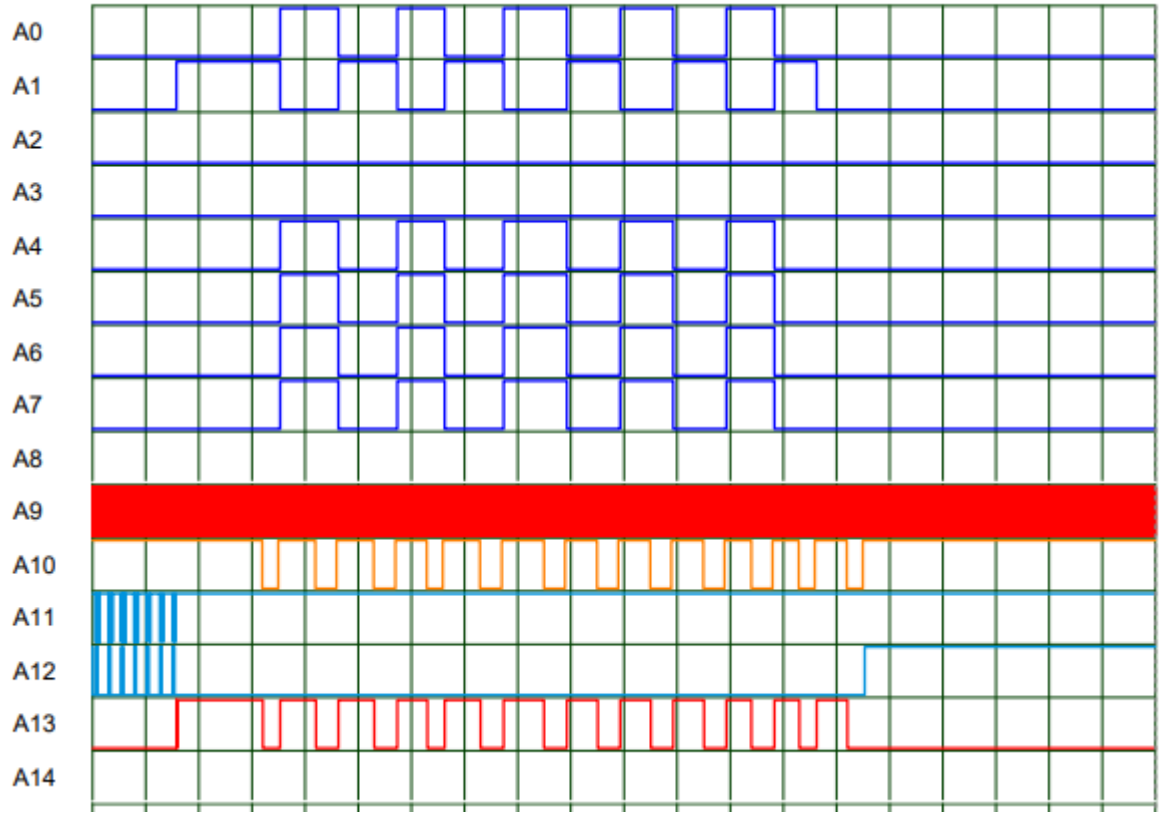
Діаграма вводу даних



Діаграма проведення розрахунку



Діаграма виведення даних



Діаграма одного обчислення

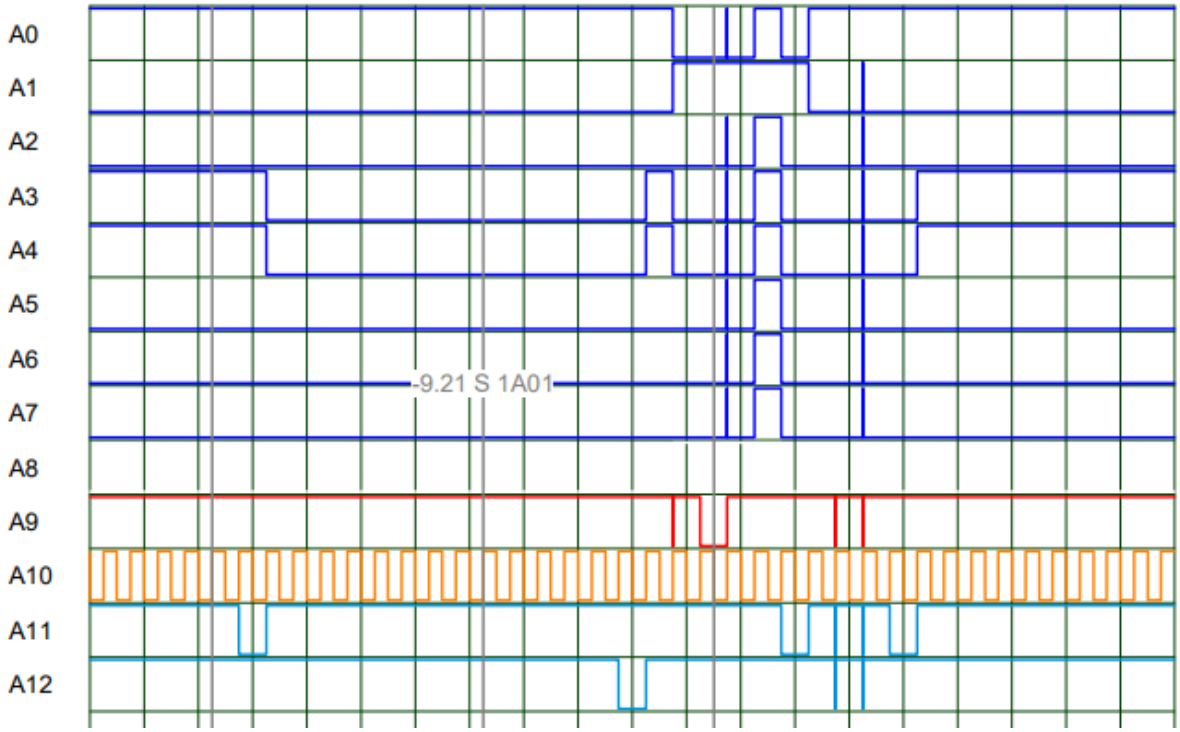
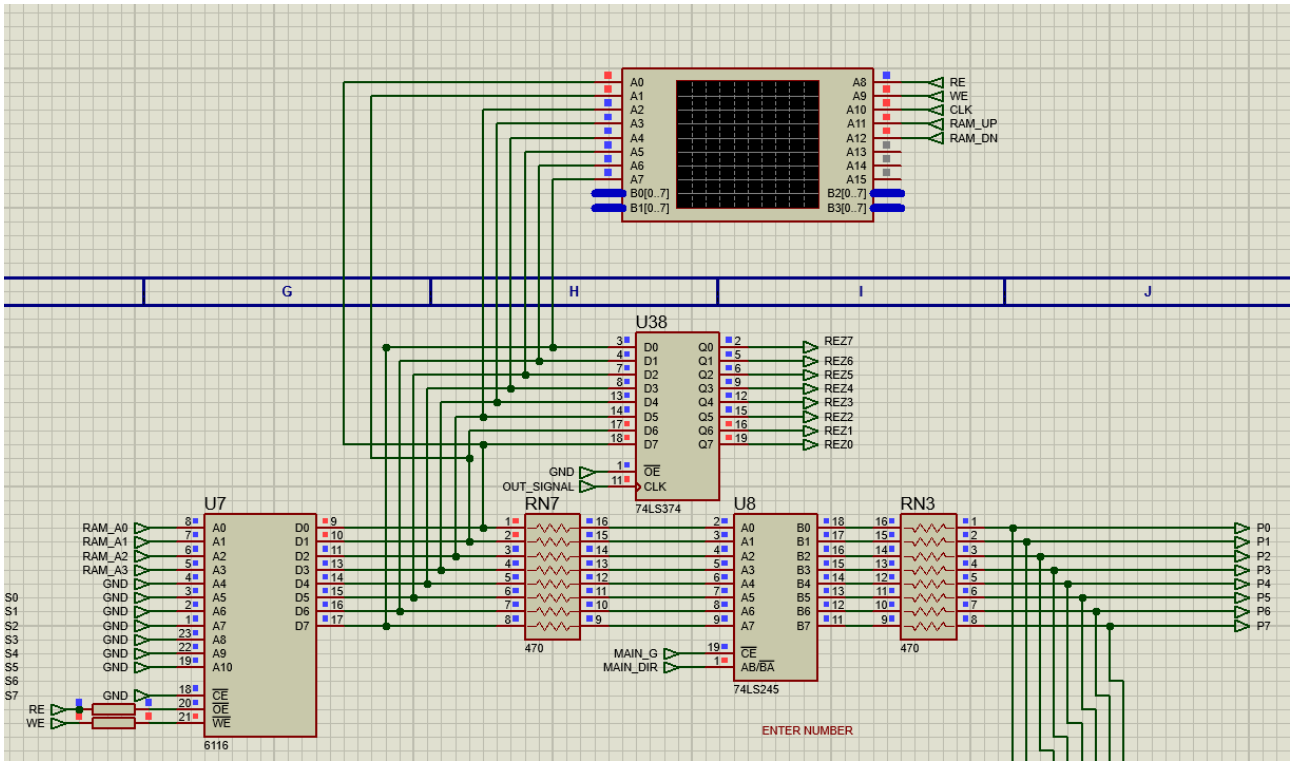


Схема підключення для аналізу одного обчислення



Перелік елементів

Поз. Позн.	Найменування	Кільк.	Примітка
	Резистори		
R1,R2,R3	10Вт, 10кОм	3	
	Конденсатори		
C25..C27	293D106X9016A2T, 10μF/16V ±10%	3	
C1..C24	K104 0.1мкФ, Y5V, 50В,+80-20%	24	
	Мікросхеми		
DD1,DD7, DD13	2732	3	
DD17,DD23	74LS163	2	
DD31	74LS169	1	
DD45,DD2 DD16,DD14 DD6,DD12	74366	6	
DD43,DD22	7430	2	
DD4	6116	1	
DD3, DD5	74LS374	2	
DD9, DD15 DD18,DD24 DD30	74LS245	5	
DD2, DD11	74LS181	2	
DD49	SN74LS10	1	
DD35, D46	4011	2	
DD32	SN74LS02	1	
DD25	CD4073	1	
DD22, D43	74S30	2	
DD20	CD4075	1	
DD19, DD28,DD44	DM7408	3	
DD10,DD47	DM74LS32	2	
DD8,DD41	DM7404	2	

Підпис і дата | Інв. № дубл. | Взім.інв № | Інв. № подл

Зм.	Лист	№ докум.	Підп.	Дата	Аркуш
					1

ЗАВДАННЯ НА КУРСОВУ РОБОТУ.

Розробити спеціалізований обчислювач, що має відповідати наступним вимогам :

1. структурна схема обчислювача : мікропрограмний автомат Мілі;
2. робоча формула : див. таблицю.
3. формат даних : 8 бітний доповняльний двійковий код;
4. інформаційний обмін здійснюється через паралельну 8-ми розрядну двонаправлену шину даних за допомогою додаткових сигналів синхронізації (рівні сигналів сумісні з ТТЛ);
5. керуючий автомат реалізувати на основі ПЗП та регістра;
6. напруга живлення та тактові імпульси надходять від зовнішнього джерела, а сигнал початкового скидання формується локально.

В результаті виконання курсової роботи мають бути розроблені :

1. схема структурна спеціалізованого обчислювача;
2. схема електрична функціональна спеціалізованого обчислювача;
3. мікропрограма у вигляді тексту, блок – схеми, граф-схеми та таблиці прошиття ПЗП;
4. схема електрична принципова спеціалізованого обчислювача (з переліком елементів);
5. Часові діаграми: вводу даних, виводу даних, обчислення одного результату, повне обчислення.
6. Вимоги до оформлення пояснювальної записки

Вимоги до оформлення аналогічні вимогам до пояснювальній записки БКР чи дипломного проекту.

Вимоги до оформлення графічної частини курсового проекта.

Всі креслення мають бути виконані у відповідності до вимог ЄСКД.

Спосіб виконання креслень – ручний або машинний.

ВАРІАНТИ ЗАВДАНЬ НА КУРСОВУ РОБОТУ

1. Курсовий проект розробити в програмному середовищі Multisim, Proteus, Active-HDL.

2. 1-16 варіанти на суматорах, 16-10 варіанти на ALU.

№	n	Формула	№	n	Формула
1	9	$Yi = (3 \cdot X_i - X_{i+1}) / 4$	17	9	$Yi = (-X_i + 5 \cdot X_{i+1}) / 8$
2	10	$Yi = 3 \cdot X_i / 4 - X_{i+1} / 8$	18	10	$Yi = -X_i / 4 + 5 \cdot X_{i+1} / 8$
3	11	$Yi = (X_i - 3 \cdot X_{i+1}) / 4$	19	11	$Yi = (7 \cdot X_i - 3X_{i+1}) / 8$
4	12	$Yi = X_i / 8 - 3 \cdot X_{i+1} / 4$	20	12	$Yi = 7 \cdot X_i / 8 - X_{i+1} / 16$
5	13	$Yi = (-3 \cdot X_i + X_{i+1}) / 4$	21	13	$Yi = (X_i - 7 \cdot X_{i+1}) / 8$
6	14	$Yi = -3 \cdot X_i / 4 + X_{i+1} / 8$	22	14	$Yi = X_i / 16 - 7 \cdot X_{i+1} / 8$
7	15	$Yi = (-X_i + 3 \cdot X_{i+1}) / 4$	23	15	$Yi = (-7 \cdot X_i + X_{i+1}) / 8$
8	9	$Yi = -5 \cdot X_i / 8 + X_{i+1} / 4$	24	9	$Yi = -7 \cdot X_i / 8 + X_{i+1} / 16$
9	10	$Yi = (5 \cdot X_i - X_{i+1}) / 8$	25	10	$Yi = (-X_i + 7 \cdot X_{i+1}) / 8$
10	11	$Yi = 5 \cdot X_i / 8 - X_{i+1} / 4$	26	11	$Yi = -X_i / 16 + 7 \cdot X_{i+1} / 8$
11	12	$Yi = (X_i - 5 \cdot X_{i+1}) / 8$	27	12	$Yi = (9 \cdot X_i - X_{i+1}) / 16$
12	13	$Yi = X_i / 4 - 5 \cdot X_{i+1} / 8$	28	13	$Yi = 9 \cdot X_i / 16 - X_{i+1} / 8$
13	14	$Yi = (-5 \cdot X_i + X_{i+1}) / 8$	29	14	$Yi = (X_i - 9 \cdot X_{i+1}) / 16$
14	15	$Yi = -5 \cdot X_i / 8 + X_{i+1} / 4$	30	15	$Yi = X_i / 8 - 9 \cdot X_{i+1} / 16$
15	16	$Yi = (6 \cdot X_i - 2 \cdot X_{i+1}) / 8$	31	16	$Yi = (X_i - 7 \cdot X_{i+1}) / 4$
16	17	$Yi = (- 2 \cdot X_i - 4 \cdot X_{i+1}) / 4$	32	17	$Yi = (9 \cdot X_i - 5 \cdot X_{i+1}) / 8$

ЛІТЕРАТУРА ДО КУРСОВОЇ РОБОТИ

1. Павлов В.Н., Ногин В.Н. Схемотехника аналоговых электронных устройств: Учебник для вузов.- М.: Горячая Линия, Телеком, 2001;

2. Бойко В.И. Схемотехника электронных систем. Цифровые устройства.- СПб.: БХВ-Петербург, 2004;

3. Угрюмов Е.П. Цифровая схемотехника. — СПб.: БХВ — Санкт – Петербург, 2000;

4. Титце У., Шенк К. Полупроводниковая схемотехника. 12-е изд. Том II: Пер. с нем.- М.: ДМК Пресс, 2007.

5. Рябенький В.М., Жуйков В.Я., Гулий В.Д. Цифрова схемотехніка: Навчальний посібник. – Львів: «Новий світ-2000», 2012. – 736с.

6. Бабіч М.П., Жуков І.А. Комп'ютерна схемотехніка. К. МК- Прес, 2004.

7. П. Хоровиц, У.Хилл. Искусство схемотехники. – М. Мир, 2003.

8. Угрюмов Е.П. Проектирование элементов и узлов ЭВМ. – М.: Высшая школа 2000.

9. Horowitz, Hill W. The Art of Electronics. Cambridge University Press, 2015. 1125 p.

10. Новиков Ю.В. Н73 Введение в цифровую схемотехнику. М: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. — 343 с: ил., табл. — (Серия «Основы информационных технологий»).

11. Потехин В.А. Схемотехника цифровых устройств: учеб. пособие для вузов [Текст] / В.А. Потехин. – Томск: В-Спектр, 2012 . – 250 с.

12. Електроніка і мікросхемотехніка: Підручник для студентів вищ. закл. освіти у 4-х т. Під ред. В. І. Сенька. – Т.1: Елементна база електронних пристроїв. – К.: ТОВ "Видавництво Обереги", 2000.– 300 с.

13. Колонтаєвський, Ю. П., Сосков А.Г. Електроніка і мікросхемотехніка. Київ «Каравела », 2009. – 416 с.

14. Євчук, О. В. Комп'ютерна електроніка : конспект лекцій / О. В. Євчук. - Івано-Франківськ : ІФНТУНГ, 2017. - 134 с.

15. Шило В.Л. Популярньїе цифровьіе микросхемьі. - М.: Металлур-гия, 1988. - 352 с.

16. Токкейм Р. Основи цифровой электроники: Пер. с англ. — М.: Мир, 1988. - 392 с.

17. Сенько В. L, Панасенко М. В., Сенько Є. В. та ін. Електроніка і мікросхемотехніка. - Т. 2. Аналогові та імпульсні пристрої. - Харків.: Фоліо, 2002.-510с.

18. Прянишников В. А. Электроника: Курс лекций. - СПб.: КОРОНА принт, 1998. - 400 с.

19. ISIS Help. Labcenter Electronics, 2014.

20. Филатов М. Разработка схемы электрической принципиальной в программной среде Proteus 8.1// Компоненты и технологии. 2015. № 6.

21. http://www.aldec.com/

22. Сафронов В. Практика математического синтеза микропрограммных управляющих автоматов на основе ПЗП и ПЛМРазработка схемы электрической принципиальной в программной среде Proteus 8.// Компоненты и технологии. 2015. № 6.

ТЕРМІНОЛОГІЧНИЙ СЛОВНИК.

2D - структура ЗП з однокоординатною вибіркою слів шляхом порушення лінії вибірки від дешифратора адреси.

2DM — структура ЗП (модифікація структури 2D), в якій слова вибираються поетапно — спочатку вибираються "довгі" слова за допомогою дешифрації однієї частини адреси, а потім слова потрібної розрядності за допомогою дешифрації іншої частини адреси.

3D — структура ЗП з двокоординатною вибіркою елементів, що запам'ятовують, на перетині двох ліній вибірки, що збуджуються виходами двох дешифраторів адреси.

x86 - сімейство процесорів фірми Intel, сумісних за системою команд: 8086, 80186, 80286, 80386, 80486, Pentium-Pentium 4.

А

Автомат Милі — автомат із пам'яттю, вихідні сигнали якого залежать як від стану, так і від вхідних сигналів.

Автомат Мура — це автомат із пам'яттю, вихідні сигнали якого залежать тільки від стану автомата.

Адресація абсолютна - адресація, при якій осередку пам'яті або зовнішньому пристрої відповідає одна-єдина адреса.

Адресація неабсолютна — адресація, коли комірці пам'яті чи зовнішньому пристрою відповідає деяка зона адрес.

Адресне ЗП - ЗП, в якому доступ до одиниць зберігання інформації здійснюється за їх адресою (розташування в пам'яті).

Адресний простір — це діапазон адрес, до яких може звертатися процесор.

Аналогове моделювання — моделювання, що спирається на дійсні значення параметрів і сигналів у пристроях і системах (струмів і напруг в електронних схемах) на відміну від моделювання зі спрощеним дискретним поданням цих параметрів і сигналів.

Асинхронність - поведінка (властивість) пристроїв і процесів, що

полягає в обробці ними інформації (сигналів) у міру їх надходження без участі зовнішніх тактуючих (виконавчих, командних) сигналів.

Асоціативне ЗП (CAM, Content Addressable Memory) — ЗП, в якому доступ до одиниць зберігання інформації здійснюється не за їх адресою, а за спеціальною ознакою (ключом).

Б

Базовий матричний кристал (БМК) — напівзамовна БІС/НВІС, що містить нескомутовані схемні елементи, основа для створення необхідного пристрою шляхом реалізації міжз'єднань елементів методом масочного програмування металізації.

Безканальний БМК - базовий матричний кристал, внутрішня область якого суцільно заповнена базовими осередками і не містить вільних каналів, заздалегідь відведених для трасування (цей тип БМК називають кристалами типу "море вентилів" або "море транзисторів").

БМК блокової структури — базовий матричний кристал, що містить спеціалізовані області (логічної обробки, пам'яті, реалізації окремих операцій і т. п.).

Бібліотека функціональних осередків - сукупність функціональних осередків, що використовуються при проектуванні на основі БМК, створюється при його розробці.

В

Вектор переривання — відомості про місцезнаходження в пам'яті підпрограми обслуговування цього переривання, що надсилаються в процесор джерелом запиту переривання або контролером переривань.

Векторне переривання — переривання, обслуговування якого потрібно передати в процесор вектор переривання.

Вентильна матриця (ВМ) – синонім поняття БМК (див. раніше).

Верифікація - перевірка за формальними методиками збігу проектної

специфікації та результатів синтезу пристрою.

Віта пара — одна з поширених конструкцій ліній передачі сигналів, що представляє собою два скручені проводи.

Внутрішньокристалічна налагодження — методи налагодження, що базуються на розміщенні всередині ІС засобів тестування, що налагоджується, та їх з'єднання з основним тестуючим обладнанням спеціальними протоколами.

Внутрішньосхемна емуляція - об'єднання ресурсів системи, що налагоджується, з ресурсами налагоджувальної системи.

Г

Гарвардська архітектура процесора - архітектура з роздільною пам'яттю програм та даних.

Д

Двійковий дешифратор - пристрій, що перетворює двійковий код код "1 з N".

Двійковий лічильник — лічильник, модуль рахунку якого дорівнює цілого ступеня числа 2, а стани кодуються двійковими числами.

Двонаправлений висновок — висновок, який залежно від програмування може бути використаний як вхід або вихід мікросхеми.

Двопортове ЗУ - ЗУ, в якому можливі одночасне читання за однією адресою і запис за іншою.

Декомпозиція - процедура зведення завдання верхнього рівня ієрархії до групи найпростіших завдань нижчого рівня.

Демультиплексор - пристрій, що передає вхідну величину в один із декількох вихідних каналів залежно від вхідного коду, що адресує.

Динамічна реконфігурація (Run-Time Reconfiguration) — швидка зміна налаштувань у схемах програмованої логіки, орієнтованих на використання в апаратурі з багатофункціональним використанням тих самих

ІС.

Довга лінія - (1) лінія, час поширення сигналу в якій з вимірно з тривалістю фронтів переданих імпульсів, що вимагає узгодження хвильових опорів в тракті передачі сигналів; (2) неперервна лінія міжз'єднань, що проходить по всій довжині або ширині кристала ВІС/СВІС програмованої логіки для швидкої передачі сигналів на великі відстані.

ДНФ – диз'юнктивна нормальна форма подання логічної функції, диз'юнкція кон'юнктивних термів.

ДОЗП (DRAM) — динамічний оперативний ЗП, запам'ятовуючим елементами якого є конденсатори.

Дрібнення контактів — наслідки пружних властивостей механічних контактів, що призводять до появи серій перемикань замість одного при однократній зміні положення контакту.

Е

Еквівалентний вентиль - група схемних елементів, що відповідає можливості реалізації на ній функції вентиля (найчастіше 2І-НЕ, 2АБО-НЕ)

І

Інформаційна ємність ЗУ — максимальний об'єм ЗУ інформації, що зберігається. '

Інтерфейс — сукупність апаратних та програмних засобів, що уніфікують процеси обміну між модулями системи.

Інтерфейс із загальною шиною — інтерфейс, у якому адреси осередків пам'яті та зовнішні пристрої мають спільний адресний простір.

Інтерфейс з роздільною шиною - інтерфейс, в якому для адрес зовнішніх пристроїв є окремий адресний простір.

Інтерфейс з подвійною швидкістю передач (DDR-технологія) — організація передач лініями зв'язку, при якій операції прийому (видачі) інформації здійснюються обома фронтами синхросигналів.

Інтерфейс із вченою швидкістю передач (QDR-технологія) — термін, що застосовується для позначення пристроїв, в яких DDR-технологія поєднується з двопортовістю.

З

Згортка за модулем — додавання по модулю значень розрядів кодової комбінації.

Зовнішня синхронізація — процес координації подій у схемі з станами тактуючої системи, що не належить самій схемі.

К

Канал трасування - вільна зона на кристалі БМК, виділена для реалізації міжз'єднань осередків.

Канальний БМК - базовий матричний кристал, у конструкції якого передбачені певні канали трасування.

Клонування проектів — несанкціоноване копіювання проектів без усвідомленого розкриття їхньої внутрішньої структури та принципів роботи.

Код - сукупність кодових комбінацій; використовуються для представлення інформації. Цей же термін використовується як синонім поняття "кодова комбінація" в тих випадках, коли це не може викликати будь-яких непорозумінь.

Код Грея - код, в якому сусідні кодові комбінації відрізняються одна від одної лише в одному розряді.

Код Хеммінгу — код, кодові комбінації якого містять кілька контрольних розрядів для перевірки на парність/непарність ваг певних груп розрядів. Має властивості не тільки виявлення, а й виправлення помилок одиничної кратності.

Кодова комбінація — набір символів прийнятого алфавіту.

Командний цикл - інтервал часу, що відповідає виконанню однієї

команди програми.

Комбінаційний ланцюг - схема, що встановилися значення вихідних сигналів якої залежать тільки від поточних значень вхідних сигналів.

Компаратор (цифровий) – пристрій, що визначає відносини між двома словами.

Компіляція - етап автоматизованого проектування електронних пристроїв, що полягає в їх синтезі.

Конвеєризація — спосіб підвищення частоти тактування в тракті обробки даних, для реалізації якого комбінаційні ланцюги тракту розбиваються на щаблі.

Конвертація проектів - переклад проектів на реалізацію на інших засобах. Найбільш широко використовуються переходи від ПЛІС до БМК і від ПЛІС до реалізації на основі стандартних осередків.

Контролер ПДП - контролер прямого доступу до пам'яті, пристрій, що керує обміном даними між пам'яттю та зовнішніми пристроями без участі процесора.

Контроль за парністю/непарністю — контроль з перевіркою парності/ непарності ваги кодових комбінацій. Має властивість виявлення помилок одиничної кратності.

Контрольний розряд — додатковий розряд, який вводиться в інформаційне слово для забезпечення парності/непарності його ваги або ваги окремих груп розрядів при контролі за модулем два або за допомогою коду Хеммінга.

Конфігурований логічний блок (Configurable Logic Block) — логічний блок мікросхем програмованої логіки, що настроюється (програмується) на відтворення необхідних функцій.

Коефіцієнт розгалуження — число входів, які можуть бути підключені до одного виходу у схемі з однотипних елементів. Максимально допустимий коефіцієнт розгалуження визначається паспортом елемента.

Кратність помилки — кількість неправильних розрядів у цій кодовій

комбінації.

Критичний шлях - маршрут поширення сигналу у пристрої (схемі), що визначає його (її) максимально можливу швидкодію.

Кеш-пам'ять — особливо швидкодіюча пам'ять, що зберігає копії інформації, яка використовується в поточних операціях обміну з процесором.

Кеш-пам'ять набірно-асоціативного типу — варіант кеш-пам'яті, проміжний щодо варіантів з повною асоціацією та прямим розміщенням.

Кеш-пам'ять із повною асоціацією — асоціативна кеш-пам'ять із довільним завантаженням даних.

Кеш-пам'ять з прямим розташуванням — кеш-пам'ять, в якій одна або кілька сторінок основної пам'яті суворо відповідають одному рядку кеш-пам'яті.

Кеш першого рівня (L1) – внутрішньопроцесорна кеш-пам'ять, розміщена на одному кристалі з процесором.

Кеш другого рівня (L2) - кеш-пам'ять, розташована поза кристалом, на якому розміщений процесор. Місткість кеш-пам'яті другого рівня, як правило, перевищує ємність кеш-пам'яті першого рівня.

Л

ЛІЗМОН - МОН-транзистор з лавинною інжекцією заряду. Має "плаваючий затвор", тобто ізольовану область над каналом, в якій можна створювати або не створювати електричний заряд, відображаючи тим самим логічні стани 1 і 0. Крім того, може мати або не мати звичайний затвор, що управляє (варіанти " з плаваючим затвором" та "з подвійним за твором").

Лічильник асинхронний - лічильник, розряди якого при переході до нового стану формуються не одночасно.

Лічильник Джонсона (лічильник Мебіуса, що зсуває регістр з перехресним зворотним зв'язком) - лічильник, що працює в коді Лібау-Крейга.

Лічильник синхронний — лічильник, розряди якого при переході в

новий стан перемикаються одночасно під впливом вхідного (тактовного) сигналу.

М

Магістрально-модульна структура — структура мікропроцесорної системи, в якій до тих самих шин підключаються різні модулі.

Мажоритарний елемент - логічний елемент з непарним числом входів, вихідна величина якого визначається тим, які сигнали (0 або 1) становлять більшість серед вхідних сигналів.

Маскування запитів - вплив на сигнали запитів переривання, прямого доступу до пам'яті та ін, забороняє обслуговування цих запитів.

Масочне програмування — запис даних у ПЗП або завдання міжз'єднань у БМК, які здійснюються під час виробництва кристалів методами інтегральної технології (за допомогою шаблонів металізації).

Матричний базовий осередок - базовий осередок внутрішньої області БМК, призначений для реалізації на її основі функціональних осередків.

Машинний цикл - інтервал часу, що становить частину командного циклу, що відповідає в основному зверненню процесора до пам'яті або зовнішнього пристрою та передачі байта (слова) у процесор або з нього.

Метастабільний стан - аномальний стан тригера, в якому він тривалий час знаходиться поблизу рівноважного стану. Викликається порушенням умов попереднього встановлення та витримки інформаційних сигналів щодо тактуючого або іншими факторами, що вводять тригер у режим, близький до рівноважного (симетричного).

Мікроконтролер — однокристальна мікроЕОМ, орієнтована на виконання щодо простих алгоритмів управління технічними об'єктами та технологічними процесами.

Мікропроцесор - реалізований на одному або декількох кристалах програмно-керований пристрій, що здійснює процес обробки інформації та управління ним.

Мікропроцесорний комплект БІС — набір мікросхем, придатних для спільного застосування при побудові мікропроцесорної системи.

Мікропроцесорна система - система, в якій реалізований закінчений процес виконання заданої програми, що містить як основні блоки (модулі) процесор, пам'ять, зовнішні пристрої та інтерфейсні схеми.

Мінімізація логічних функцій - таке перетворення логічних функцій, яке спрощує їх у сенсі заданого критерію.

МНОН - транзистор зі структурою "метал-нітрид-оксид-напівпровідник", в якому при програмуванні можна створювати або усувати заряд на межі шарів "нітрид-оксид", відображаючи тим самим логічні стани (О і 1).

Моделювання - відтворення в модельному часі поведінки моделі (математичної або програмної), що відповідає поведінці цільової системитемному часі.

Моделювання з одиничною затримкою (Unit Delay) — найпростіша форма тимчасового моделювання цифрових схем, коли затримки всіх елементів вважаються однаковими і рівними модельної одиниці.

Модуль рахунку - число станів, яке може мати лічильник, тобто ємність лічильника.

Мультиплексування - передача сигналів від різних джерел по одних і тих же лініях у режимі розподілу часу.

Мультиплексор - схема, що передає на вихід одну з декількох вхідних величин під керуванням коду, що адресує.

Н

Навантажувальна здатність — параметр мікросхеми, який визначається величинами допустимих вихідних струмів у станах логічного нуля та одиниці. Значення максимально допустимого ємнісного навантаження зазвичай визначається окремо.

Наскрізний струм — короткочасний імпульс струму споживання

мікросхеми, характерний для елементів ТТЛ(Ш) і КМОП і що виникає при їх перемиканні.

О

Однофазна синхронізація — система синхронізації, в якій на всі елементи пам'яті (тригери) подаються одні й ті сигнали, що тактують.

Операція монтажної логіки — логічна операція, що реалізується шляхом з'єднання в одній точці виходів кількох логічних елементів з відкритим колектором або емітером.

Організація ЗУ - параметр ЗУ, що виражається добутком максимально можливої кількості слів, що зберігаються, на їх розрядність.

Основна пам'ять - пам'ять, що працює в режимі оперативного обміну даними з процесором і, на відміну від кеш-пам'яті, що зберігає весь обсяг

П

Паралельний периферійний адаптер (PPI, Parallel Peripheral Interface) - пристрій, що обслуговує обмін паралельними даними між процесом сорому і зовнішніми пристроями.

Передній фронт - перепад сигналу під час його переходу від пасивного рівня до активного. Для Н-активного сигналу переднім є позитивний фронт, для L-активного - негативний.

Перехресна перешкода - перешкода, що породжується взаємним впливом біля спраглих сигнальних ліній.

Периферійне сканування - синонім терміна "Граничне сканування", в цій книзі не використовується.

Поведінкова модель — високорівневе уявлення електронного проекту, яке описує поведінку різних модулів або підсистем проекту (зазвичай без урахування технології, що використовується при його реалізації).

Поліноміальний лічильник - зсувний регістр з лінійними зворотними зв'язками, тобто зв'язками, реалізованими за допомогою елементів додавання по

модулю два. Використовуються як генератори псевдовипадкових після тривалостей.

Повністю замовлена BIC/CBIC — мікросхема, яка повністю проектується за конкретним замовленням і виготовляється за допомогою індивідуального набору фотошаблонів для всіх етапів процесу виробництва.

Порогова напруга — значення вхідної напруги елемента, перехід через який трактується як перехід сигналу з одного логічного стану до іншого.

Послідовні репрограмовані ЗУ — репрограмовані запам'ятовуючі пристрої типів EEPROM і Flash з послідовним інтерфейсом, які приймають і видають команди та дані по однорозрядних лініях.

Принстонська архітектура процесора (архітектура фон Неймана) — архітектура із загальною пам'яттю команд та даних.

Пріоритетний шифратор - пристрій, що виробляє двійковий номер старшого з наявних на входах запитів (переривання, прямого доступу до пам'яті та ін.).

Програмована логічна матриця (PLA, Programmable Logic Array) — мікросхема для реалізації системи перемикальних функцій, представлених у ДНФ і складаються з єдиного набору кон'юнктивних термів. Основа ПЛМ - послідовно включені програмовані матриці елементів І та АБО.

Програмована матрична логіка (PAL, Programmable Array Logic) — мікросхема для реалізації системи перемикальних функцій, представлених у ДНФ, кожна з яких складається з індивідуального набору щодо невеликої кількості кон'юнктивних термів. Основа ПМЛ - послідовне включення програмованої матриці елементів І та фіксованої матриці елементів АБО.

Програмування в системі (In System Programmable) — властивість БІС/НВІС програмованої логіки конфігуруватися безпосередньо в системі, тобто без вилучення зі схеми.

Програмований інтервальний таймер (Programmable Interval Timer) - мікросхема, що виконує в системі операції, пов'язані з часом, частотами та інтервалами.

Програмований контролер переривань (Programmable Interrupt Controller) - мікросхема, що обслуговує векторні переривання за запитами багатьох джерел. Реалізує різноманітні способи арбітражу та маскування запитів.

Програмований зв'язковий адаптер (Programmable Communication Interface) - мікросхема, що обслуговує обмін даними між процесором і зовнішнім пристроєм, що оперує послідовними даними. Виконує перетворення паралельних даних на послідовні, і навпаки, і необхідні інтерфейсні функції.

Проектування методом "стандартних осередків" - проектування BIC/ CBIC, що виготовляються за допомогою повного набору фотошаблонів, фрагменти яких можуть запозичуватися з бібліотеки готових рішень.

Прототипування — як правило, використання програмованих або конфігурованих стандартно схем, що випускаються для роботи з проектом (налагодження) перед його замовною реалізацією в конструктивно закінченій формі.

Псевдовипадкова послідовність — детермінована і, як правило, циклічна послідовність, що складається з нулів і одиниць, характеристики якої близькі до характеристик істинно випадкової послідовності.

Подієво кероване моделювання — моделювання, при якому визначення стану моделі відбувається тільки при виникненні події (зміні стану будь-яких внутрішніх або зовнішніх елементів, таких як новий такт роботи). Відрізняється від тимчасового моделювання, при якому обчислення стану системи здійснюється через однакові кванти астрономічного часу.

Р

Радіальне переривання - переривання, місцезнаходження підпрограми про служіння якого заздалегідь відомо, і передача в процесор відомостей про нього не потрібно.

Реверсивний лічильник — лічильник, напрямок рахунку в якому може змінюватися під впливом сигналу, що управляє.

Регенерація даних - необхідний для динамічних ЗУ режим відновлення збережених даних, періодична реалізація якого запобігає втраті інформації внаслідок перезаряду запам'ятовують конденсаторів струмами витоку.

Регістр — типовий функціональний вузол цифрових пристроїв, що виконує операції прийому, зберігання та видачі даних, причому прийом та видача можуть здійснюватися для паралельних та (або) послідовних даних.

Регістровий файл - запам'ятовуючий пристрій, реалізований на основі набору регістрів.

Резистор-термінатор — резистор, що має опір, що дорівнює хвильовому опору лінії передачі сигналу, що включається в її кінці для придушення відбитих хвиль.

Реінжиніринг - відтворення (несанкціоноване) проекту з тим чи іншим ступенем розкриття його внутрішньої структури та принципів функціонування.

С

Самовідновлення після збою - властивість автомата входити в робочий цикл після попадання в "зайві" (не використовуються) стани без впливу спеціальних сигналів установок.

Сегментована система міжз'єднань — система комутації, властива головним чином схемам FPGA, в якій лінії зв'язків складаються з окремих сегментів, тобто провідних ділянок, що не містять програмованих ключів. Самі сегменти з'єднуються один з одним програмованими ключами.

Семисегментний індикатор - індикатор для візуального сприйняття символів, в якому ці символи відображаються за допомогою семи відрізків прямих (сегментів).

Синдром помилки — слово, складене з розрядів, значення яких визначаються результатами перевірок груп, що входять до кодових комбінацій коду Хеммінга. Синдром вказує номер неправильного розряду, що підлягає виправленню.

Симуляція - моделювання поведінки системи (цільової) за допомогою

системи з іншою фізичною природою.

Синтез - процес перекладу опису проекту від одного рівня абстракції до іншого (як правило, у напрямку до певної фізичної реалізації).

Синхронізатор одиночних імпульсів — схема виробітку за командою одиничного імпульсу, що належить тактовій послідовності системи.

Синхронність — скоординована зміна стану кількох елементів схеми.

Синхронний автомат — автомат, елементи пам'яті якого приймають інформацію тільки в певні моменти часу, що задаються синхронними сигналами.

Системний інтерфейс — інтерфейс міжмодульного обміну в межах мікропроцесорної системи.

Системний еквівалентний вентиль - одиниця виміру складності програмованих БІС / НВІС. Визначення "системний" означає, що через число таких еквівалентних вентилів виражаються і складності блоків, що не належать до логічних, перш за все блоків пам'яті.

Досконала диз'юнктивна нормальна форма (СДНФ) — форма подання перемикальних (логічних) функцій, диз'юнкція кон'юнкцій однакової розмірності, що включають літерали всіх аргументів.

Статична завадостійкість — стійкість до дії перешкод, тривалість яких не обмежується. Визначається амплітудами перешкод, які не порушують роботу елемента.

Статичний ризик - короткочасні "хибні" сигнали, що з'являються в перехідних процесах на виходах схем у ситуаціях, в яких згідно з логічними рівняннями вихідні сигнали повинні залишатися незмінними. Виникають як наслідок затримок сигналів у ланцюгах схеми.

Статичне ОЗУ (SRAM) - оперативне запам'ятовуючий пристрій, основою запам'ятовуючого елемента якого є тригер. Відрізняється високою швидкодією.

Сторожовий таймер — таймер, призначений для відстеження ситуацій, в яких програма виконується неправильно, і викликає перезапуск програми у

цих випадках.

Сторінкова організація пам'яті — організація пам'яті, у якій адреса осередку сприймається як з двох частин, причому старша вказує сторінку (субмодуль), а молодша є адресою слова даної сторінці (у цьому субмодулі).

Стробуючий сигнал - сигнал, своїм рівнем визначальний інтервал часу, на якому виконується та чи інша операція.

Схема прискореного множення - у цьому контексті схема, що реалізує алгоритм множення "відразу на два розряди".

Т

Табличний функціональний перетворювач (LUT, Look-Up Table) — логічний блок програмованих ВІС/СВІС, реалізований на основі схем програмованої пам'яті.

Тег - додаткові дані, що супроводжують одиницю інформації, що зберігається в кеш-пам'яті, і визначають, копією вмісту якої комірки основної пам'яті є ця одиниця інформації.

Терм — у цій книзі під цим терміном розуміється кон'юнктивний терм, т. е. логічний твір змінних (їх прямих чи інверсних значень).

Тестування - перевірка збігу необхідної та реальної поведінки пристроїв та систем, що проводиться з використанням певних методів та засобів.

Третій стан - стан "відключено", в якому вихід логічного

елемента практично від'єднується від навантаження. Елементи з трьома станами виходу (0, 1 і "відключено") можуть підключатися до магістралей систем з магістрально-модульною структурою.

Тимчасова діаграма - графічне представлення сигналів електричних схем, що показує, як змінюються сигнали в часі і як вони взаємно співвідносяться.

Тимчасове моделювання - моделювання електричних схем з урахуванням дійсних часових співвідношень між вхідними та вихідними

сигналами всіх елементів схеми.

Тригер - елементарний автомат, що містить елемент пам'яті з ємністю один біт і схему керування записом цього елемента пам'яті.

Тригер асинхронний — тригер, який сприймає вплив інформаційних вхідних сигналів безпосередньо в моменти їх змін.

Тригер-клацанка - тригер типу D, що має режим "прозорості" при одному рівні керуючого сигналу і режим зберігання при іншому.

Тригер, керований фронтом - тригер, для якого сигналом дозволу прийому інформації є перепад керуючого (тактуючого) сигналу. Такий тригер називають також синхронним тригером з динамічним управлінням.

Тригер D — синхронний тригер з одним інформаційним входом, який приймає стан, відповідний вхідному сигналу, за дозволом тактуючого сигналу.

Тригер JK — тригер, який має інформаційні входи установки та скиду, а також режим лічильного тригера.

Тригер RS - тригер, що має інформаційні входи установки та скидання.

Турбо-біт — біт, програмуванням якого в схемах вибирається один з двох режимів — більш швидкодіючий (при підвищенні споживаної схемою потужності) або менш швидкодіючий (економічніший за споживаною потужністю).

У

Універсальний логічний модуль - пристрій, який відтворює будь-яку функцію заданої кількості аргументів.

Ф

Фіксований пріоритет - пріоритет, наданий даному запиту (входу) і не змінюється у процесі роботи системи.

Флеш-пам'ять — пам'ять, що репрограмується, в якій стирання даних проводиться електричними сигналами для всього кристала або для окремих блоків (симетричних або несиметричних).

Флеш-пам'ять із дзеркальним бітом (Mirror-bit Memory) — флеш-пам'ять із зберіганням двох бітів в одному запам'ятовуючому елементі за допомогою дорожньогоого поділу областей зберігання двох зарядів у плаваючому затворі транзистора.

Флеш-пам'ять з несиметричними блоками (Boot-Block Flash Memory) - флеш-пам'ять для зберігання програм та інших рідко змінюваних даних, структурі якої виділяються спеціалізовані блоки різного значення.

Флеш-пам'ять із симетричними блоками (Flash-File Memory) — флеш-пам'ять для заміни електромеханічних пристроїв на основі жорстких дисків.

Функції збудження тригера - функції, що визначають такі впливи на тригери автомата, які переводять автомат з одного стану в інший згідно з необхідним графом переходів.

Функціональний осередок - типове схемне рішення, що входить до складу бібліотеки БМК і реалізується на основі одного або декількох базових осередків кристала.

Функціональне моделювання — моделювання цифрових систем, при якому не враховуються затримки розповсюдження сигналів усередині окремих компонентів.

Функція генерації — допоміжна функція, яка використовується при синтезі суматорів та деяких інших пристроїв, які використовують сигнали переносу. Приймає одиничне значення для тих розрядів або груп розрядів, на виходах яких сигнал перенесення виникає незалежно від наявності або відсутності вхідного переносу.

Функція прозорості — допоміжна функція, яка використовується при синтезі суматорів та деяких інших пристроїв, у яких використовуються сигнали перенесення. Приймає одиничне значення тих розрядів чи груп розрядів, на виходах яких сигнал перенесення виникає лише за наявності вхідного переносу.

X

Хвильовий опір - параметр лінії передачі сигналів, що трактується як "довга лінія".

Час витримки (Hold Time) - (1) для тригера - інтервал часу після надходження синхросигналу, протягом якого вхідні інформаційні сигнали повинні залишатися незмінними; (2) у більш загальному сенсі для двох сигналів А і це інтервал часу між початком сигналу А і закінченням сигналу (цей час називають також часом утримання).

Ц

Цикл ЗУ - мінімальний інтервал часу між сусідніми однотипними зверненнями до ЗУ. Відповідно до типу звернення розрізняють цикли читання, записи та ін.

Циклічний (круговий) пріоритет - порядок обслуговування запитів (переривання, прямого доступу до пам'яті та ін), для якого джерела просів рівноправні. Рівноправність джерел запитів досягається тим, що їхні пріоритети змінюються під час роботи системи — після обслуговування джерело отримує нижчий пріоритет, який поступово підвищується з обслуговуванням інших джерел запитів.

Ч

Час попереднього встановлення (Set-Up Time) - (1) для тригера - інтервал часу до надходження синхросигналу, протягом якого вхідні інформаційні сигнали повинні залишатися незмінними; (2) у більш загальному сенсі для двох сигналів А і це інтервал часу між початком сигналу А і початком сигналу.

Ш

Швидкий сторінковий доступ (FPM, Fast Page Mode) — прискорений доступ до даних в динамічних ЗУ, можливий за умови "куповості" їх адрес, коли дані, що запитуються, належать одній і тій же сторінці (рядку матриці

запам'ятовуючих елементів).

Доповнення.

Словник іноземних скорочень та термінів

AGP – Advanced Graphics Port – стандартний інтерфейс графічних систем.

AHDL – Altera Hardware Description Languages – мова опису апаратури фірми Altera.

ASCII – American Standard Code for Information Interchange – стандартний американський код обміну інформацією у послідовних інтерфейсах.

ASICs - Application Specific Integrated Circuits - спеціалізовані ІС, що виготовляються тим чи іншим способом за індивідуальним технічним завданням (для конкретного проекту).

AVR – сімейство мікроконтролерів фірми Atmel.

Back Annotation — коригування схеми проекту, яка спирається на інформацію, отриману після моделювання або будь-яких інших видів роботи проектної інформації, і відображає вимоги зміни схеми в залежності від результатів обробки (зміна контактів, обмін вентильних груп).

BEDORAM - Burst Extended Data Out RAM - варіант динамічних ОЗУ, близький до EDORAM і відрізняється від нього пакетним доступом до даних, що дозволяє скоротити цикл звернення всередині пакета.

BIST - Built In Self Test - засоби тестування інтегральних схем, вбудовані в цю ж схему.

BSCs - Boundary Scan Cells - див. Комірки граничного сканування.

CD – Coder – шифратор.

CDR - Clock-Data Recovery - автоматична взаємна тимчасова під будівництво переданих даних і синхросигналів.

CDRAM — Cached DRAM — динамічна ОЗУ підвищеної швидкодії, що досягається шляхом кешування.

CISC - Complex Instruction Set Computer - архітектура процесора, що має широкий набір різноманітних команд.

Clock Boost — збільшення частоти тактових імпульсів, одна з функцій, що виконуються блоками PLL.

Clock Lock — корекція тимчасового становища тактових імпульсів, одне з функцій, виконуваних блоками PLL.

Clock Skew - тимчасове зсув тактового імпульсу щодо заданого положення, викликаний паразитними затримками в ланцюгах тактування.

Clock Tree — деревоподібна побудова синхронних схем для мінімізації ефекту тактового перекосу (Clock Skew).

CPLD — Complex PLD — БІС/СВІС програмованої логіки, структура якої є сукупністю блоків типу PAL або GAL, об'єднаних матрицею програмованих з'єднань. Програмується користувачем.

CPU - Central Processor Unit - центральний процесор.

CRU – Carry Unit – блок прискорених переносів.

CSoC - Configurable System on Chip - мікросхема типу "конфігурована система на кристалі".

DC – Decoder – дешифратор.

DCM — Digital Clock Manager — блок керування параметрами синхронізації.

DDR – Double Data Rate – передача сигналів з подвійною швидкістю.

DIMM — Dual In Line Memory Module — модуль пам'яті з дворядним розташуванням висновків.

DLL – Delay Locked Loop – схема корекції параметрів синхросигналів.

DMA - D irect Memory Access – прямий доступ до пам'яті.

DRDRAM - Direct RDRAM - варіант динамічного ОЗУ високого швидкодії типу RDRAM, в якому скорочено характерне для RDRAM запізнення при доступі до першого слова пакета даних (латентність).

DSP – Digital Signal Processing – цифрова обробка сигналів.

EAB - Embedded Array Block - вбудований блок пам'яті/обробки в

мікросхемах програмованої логіки.

EDA – Electronic Design Automation – загальна назва програмних засобів автоматизації проектування складних електронних систем.

EDIF – Electronic Design Interchange Format – формат обміну проектів при розробці електронних схем. Список ланцюгів у цьому стандарті можна отримати з описів проекту мовами VHLD чи Verilog HDL за допомогою стандартних програм. Файли у форматі EDIF можуть формуватися пакетами програмних засобів ряду САПР для цілей моделювання за допомогою стандартного пакета моделювання EDIF.

EDORAM — Extended Data Out RAM — варіант динамічної ОЗП підвищеної швидкодії.

EEPROM - Electrically Erasable Read Only Memory - електрично стирає моє ЗП, що репрограмується.

EPROM - Electrically Programmable Read Only Memory - репрограмує моє ЗП з ультрафіолетовим стиранням даних.

EPROM-OTP - одноразово програмоване ЗП з програмуванням станів плаваючих затворів МОН-транзисторів.

ESB — Embedded System Block — вбудований системний блок пам'яті/обробки в мікросхемах програмованої логіки.

FACM – Full Associative Cache Memory – повністю асоціативна кеш пам'ять.

FCRAM — Fast Cycle Random-Access Memory — динамічне ЗП із скороченою тривалістю циклу.

FIFO - First-In - First-Out - ЗП з послідовним доступом до даних типу "черга" (за правилом "перший увійшов - перший вийшов").

Firm-ядро — різновид soft-ядра з меншим ступенем свободи реалізації в мікросхемі з програмованою структурою.

FPM — Fast Page Mode — див. Швидкий сторінки.

FRAM - Ferroelectric RAM - фероелектричне ОЗУ.

GA – Gate Array – базовий матричний кристал (вентильна матриця).

GRM – General Routing Matrix – головна матриця з'єднань.

Hard-ядро - область кристала з програмованою структурою, в якій реалізовано той чи інший пристрій жорсткої структури, що не дозволяє репрограмування.

Hit - сигнал "попадання" в схемах кеш-пам'яті, що свідчить про наявність запитуваної одиниці інформації в цій пам'яті.

HLD – Hardware Description Language – мова опису апаратури.

HSTL – High Speed Transceiver Logic – стандарт сигналів інтерфейсу.

12C - Inter-Inter Computer — інтерфейс синхронного послідовного обміну для з'єднання між мікроЕОМ або між мікроЕОМ і периферійними пристроями.

IP - Intellectual Property - інтелектуальна власність, термін, прийнятий для найменування програмних ядер (soft-ядер) мікросхем з програмованими структурами.

ISP - In-System Programmable - див. Програмування в системі.

JEDEC — Joint Electronic Device Engineering Council — об'єднана інженерна рада з електронних пристроїв, в галузі програмованої логіки позначає текстовий файл, що містить інформацію про програмування схеми у стандартній формі JEDEC.

JTAG — Joint Test Action Group — об'єднана група з питань тестування, на ім'я якої названо методи тестування БІС/НВІС без фізичного доступу до кожного їх висновку та програмування мікросхем програмованої логіки за допомогою JTAG-інтерфейсу.

LIFO - Last-In - Fist-Out - ЗУ з послідовним доступом до даних стекового типу (за правилом "останній увійшов - перший вийшов").

LPGA – Laser-Programmable Gate Array – базовий матричний кристал із лазерним програмуванням міжз'єднань.

LUT - Look Up Table - див. Табличний функціональний перетворювач.

MDAC — Multiplying Digital-Analog Converter — множинний цифровий перетворювач.

MDRAM - Multibank DRAM - багатобанківська динамічна пам'ять, варіант підвищення швидкодії ЗУ за допомогою розбиття пам'яті на частини (банки), що при купівлі адрес послідовних звернень до пам'яті дозволяє звертатися до банків по черзі з вищою швидкістю.

MIPS - (1) Mega Instructions Per Second - мільйон команд в секунду, міра продуктивності процесора; (2) Microprocessor without Interlocked Pipe line Stages - назва відомої фірми, що означає, по суті, "мікропроцесор без затримок очікування конвеєра".

MLC — Multilevel Cell — багаторівневий осередок пам'яті, що дозволяє зберігати в ньому більше одного біта.

MPGA – Mask-Programmable Gate Array – базовий матричний кристал із масковим програмуванням міжз'єднань.

MRAM - Magnetic RAM - перспективне ОЗП, робота якого заснована на застосуванні магнітних матеріалів.

MUX – Multiplexer – мультиплексор.

Net List — перелік усіх компонентів проекту та всіх сигналів, що з'єднують ці компоненти між собою.

NoBL - No Bus Latency - варіант ОЗУ з усуненням затримки при реверсі шин.

Node – одиночний контакт у електричних схемах.

NtRAM - No Turnaround RAM - варіант ОЗУ з усуненням затримки при реверсі шин.

NvSRAM - Non-Volatile SRAM - статичне ОЗУ з властивістю енергонезалежності.

OUM — Ovonics Unified Memory — перспективне ЗП фірми Ovonics, робота якого заснована на застосуванні матеріалів із змінними фазовими станами.

PAL - Programmable Array Logic - див. Програмована матрична логіка.

PCI – Peripheral Component Interconnect – популярна шина розширення

для з'єднання периферійних компонентів системи.

PFRAM - Polimeric FRAM - полімерна фероелектрична пам'ять.

PIC - Programmable Interruption Controller - програмований контролер переривань.

PIP - Programmable Interconnect Point - програмована точка з'єднань.

PLA - Programmable Logic Array - див. Програмована логічна матриця.

PLD - Programmable Logic Device - загальне найменування для схем PAL та PLA.

PLL - Phase Locked Loop - схема стежить системи з чутливим елементом, що реагує на різницю фаз імпульсних послідовностей, що використовується для управління часовими параметрами синхросигналів цифрових пристроїв.

PREP — Programmable Electronics Performance Corporation — консорціум компаній, що запропонував набір еталонних схем та методику оцінки складності BIC/CBIC програмованої логіки.

PSOC - Programmable System On Chip - програмована система на кристалі.

Pull-down Resistor - "заземлювальний" резистор, що фіксує низький рівень потенціалу на виведенні у відсутності на ньому активного сигналу.

Pull-up Resistor - "підтягуючий" резистор, що фіксує високий рівень ень потенціалу на виведенні відсутність у ньому активного сигналу.

RAM - Random-Access Memory - оперативне ЗУ (ЗУ з довільним доступом).

RDRAM – Rambus DRAM – динамічна ОЗУ високої швидкодії, розроблена фірмою Rambus.

RISC - Reduced Instruction Set Computer - варіант архітектури швидко діючих процесорів зі скороченим набором команд.

RLDRAM - Reduced Latency DRAM - динамічна ОЗУ, що швидко діє.

SERDES - Serialiser-Deserialiser - блок перетворення паралельних даних у послідовні, і навпаки.

SIMM – Single In-line Memory Module – модуль пам'яті з однорядним розташуванням висновків.

SOC — System On Chip — ВІС/НВІС вищого рівня складності, в якій можна реалізувати цілу систему, тобто сукупність різних модулів, що утворюють цілісну систему обробки інформації.

SOI - Silicon On Insulator - схемотехнологія інтегральних схем, що забезпечує мінімальність паразитних параметрів схеми, що, в кінцевому рахунку, призводить до поліпшення її технічних характеристик.

SOPC - System On Progammable Chip - БІС/НВІС програмованої логіки вищого рівня складності, на якій можна реалізувати цілу систему, тобто сукупність різних модулів, що утворюють цілісну систему обробки інформації.

SPI - Serial Peripheral Interconnect - синхронний послідовний інтерфейс.

SPLD – Simple Programmable Logic Device – програмована логічна схема невисокої складності.

SRAM – Static RAM – статичне ОЗП.

SSTL - Stub Series Terminated Logic - стандарт сигналів інтерфейсу, що використовується для мікросхем швидкодіючих динамічних ОЗП.

StrataFlash — запам'ятовуючий пристрій флеш-типу із запам'ятовуванням двох бітів в одному елементі, що запам'ятовує, за допомогою багаторівневого заряду плаваючих затворів транзисторів ЛІЗМОН.

TAP - Test Access Port - див. Порт тестування.

Terminator - резистор в кінці лінії зв'язку, що забезпечує для неї узгодження хвильового опору з навантаженням.

UART - Universal Asynchronous Receiver - Transmitter - програмований зв'язковий адаптер, що реалізує асинхронні протоколи передачі послідовних даних.

VLIW - Very Long Instruction Word - архітектура процесора, що має набір дуже довгих (багатобайтових) команд.

WDT – Watchdog Timer – сторожовий таймер.

XACT- пакет програмних засобів для проектування БІС/НВІС для

грамованої логіки компанії Xilinx.

ZBT – Zero Bus Turnaround – варіант ОЗУ з усуненням затримки реверсу шини.

НАВЧАЛЬНЕ ВИДАННЯ

НАВЧАЛЬНИЙ ПОСІБНИК

«Комп’ютерна схемотехніка: курсове проектування»

для студентів
спеціальності 123 “Комп’ютерна інженерія”

Автор Клушин Юрій Сергійович

Редактор

Комп’ютерне верстання

Підписано до друку . .2021.
Формат 60x84 1/16. Папір офсетний. Друк на різографі.
Умовн. друк. арк. . Обл.-вид. арк. .
Наклад прим. Зам. .

Видавництво Національного університету “Львівська політехніка”
Реєстраційне свідоцтво ДК № 751 від 27.12.2001 р.

Поліграфічний центр Видавництва
Національного університету “Львівська політехніка”

вул.Ф. Колесси, 2, Львів, 79000