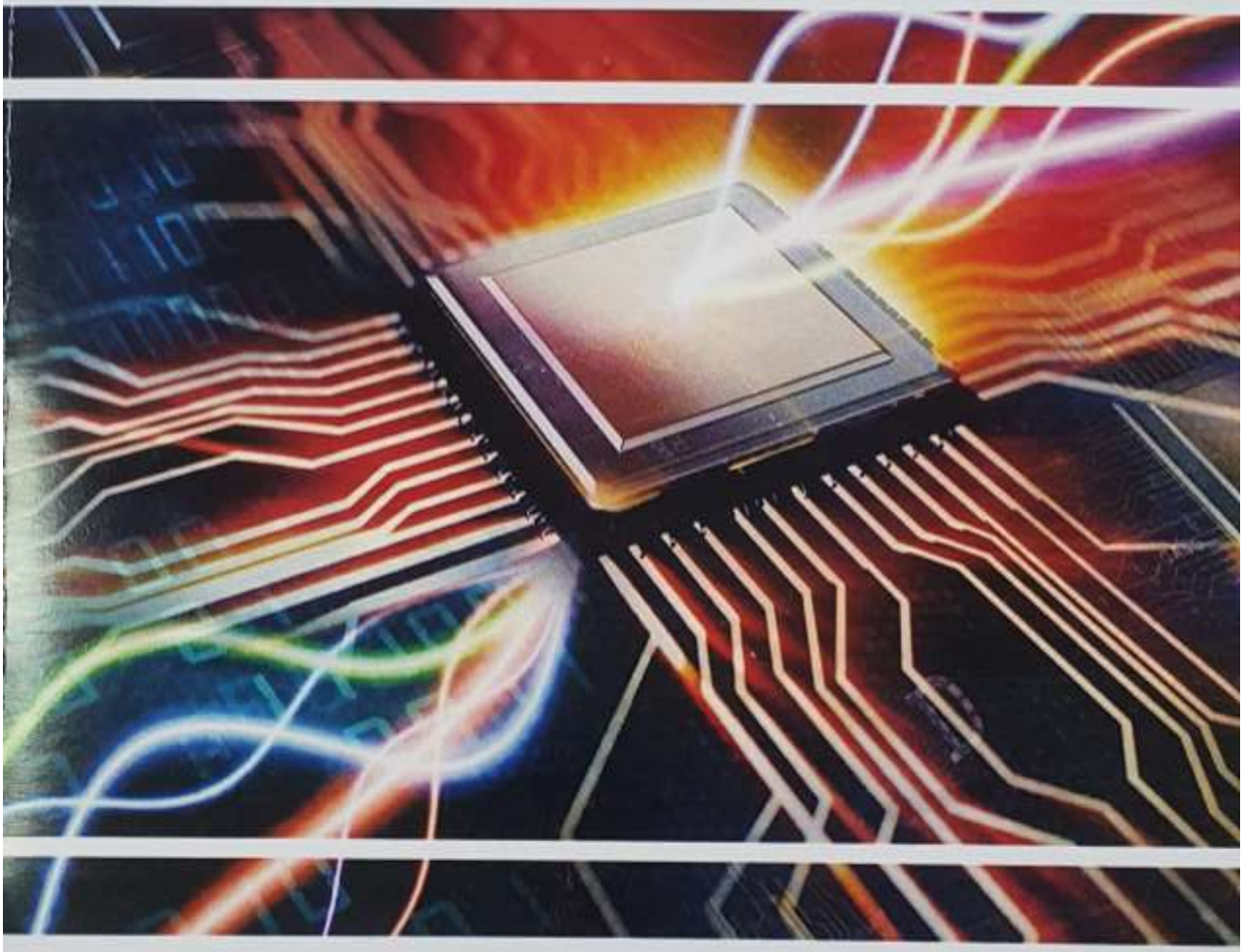
Лахно В.А., Гусєв Б.С., Смолій В.В.,
Місюра М.Д, Касаткін Д.Ю.



Технології проектування комп'ютерних систем

частина 1

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Кафедра комп'ютерних систем і мереж

Лахно Валерій Анатолійович
Гусєв Борис Семенович
Смолій Віктор Вікторович
Місюра Максим Дмитрович
Касаткін Дмитро Юрійович

Технології проектування комп'ютерних систем

ЧАСТИНА 1. ПРОЕКТУВАННЯ ЦИФРОВИХ СИСТЕМ У САПР QUARTUS II ТА ЛАБОРАТОРНОМУ СТЕНДІ DE0

Навчальний посібник із дисципліни «Технології проектування комп'ютерних систем»
для студентів
спеціальності 123 «Комп'ютерна інженерія»

КИЇВ 2019
КОМПРІНТ

УДК 004.31
ББК 32.973.26-018
Л29

Рекомендовано до друку Вченою радою Національного університету біоресурсів і природокористування України
(протокол № 4 від 26.11.2019р..)

Рецензенти:

Казмірчук С.В. - доктор технічних наук, доцент, завідувач кафедри компютерізованих систем захисту інформації, Національний авіаційний університет;

Криворучко О.В. - доктор технічних наук, професор, завідувач кафедри програмної інженерії та кібербезпеки Київського національного торговельно-економічного університету;

Хиленко В.В. - доктор технічних наук, професор кафедри компютерних наук Національного університету біоресурсів і природокористування України.

Л29 **Лахно В.А., Гусєв Б.С., Смолій В.В., Місюра М.Д., Касаткін Д.Ю.**
Технології проектування комп'ютерних систем (частина 1) - К.: НУБіП України, 2019. – 205 с.

У навчальному посібнику, відповідно до навчального плану, містяться теми, які в певній мірі дозволяють студентам оволодіти методами аналізу й синтезу комп'ютерних систем за допомогою найбільш поширених CASE-засобів.

Навчальний посібник містить ґрунтовний теоретичний матеріал, що супроводжується достатньо великою кількістю графічних зображень, схем та прикладів розв'язання конкретних задач з використанням CASE-засобів для проектування комп'ютерних систем, дослідження та проектування цифрових систем на лабораторному стенді DE0 та у САПР Quartus II:

Позитивним фактом для навчання студентів однойменної дисципліни є наявність у посібнику достатньої кількості прикладів та задач із використанням програмно-технічних засобів

Слід зазначити, що необхідність такого видання вкрай гостра, у зв'язку з відсутністю україномовного контенту та інших підручників із такої дисципліни.

Посібник являється актуальним, написаний на сучасному високому рівні, відповідає усім основним вимогам щодо такого роду робіт для ЗВО, якісно сприятиме підготовці фахівців зі спеціальності 123 – «Комп'ютерна інженерія».

ЗМІСТ

ВСТУП

На сьогодні область діяльності людини, де використовують цифрові пристрої дуже широка. Нас оточує велика кількість об'єктів в різних сферах діяльності, що потребують їх вивчення, прогнозування та можливого впливу на їх розвиток. Інформацію про об'єкти перетворюють у електричні сигнали, які все частіше мають дискретну (цифрову) форму. Це дозволяє обробляти сигнали за допомогою різних обчислювальних пристроїв. Основу таких пристроїв і систем складають цифрові схеми (пристрої). Сучасні методи проектування таких пристроїв базуються на використання *програмованих логічних інтегральних схем* (ПЛІС), що дозволяє зменшити час від розробки принципової схеми пристрою до його реалізації, поліпшити його масо-габаритні показники та надійність, виконати моделювання його роботи на персональному комп'ютері та стенді для відладки цифрових пристроїв на ПЛІС фірми "Altera" QUARTUS II, проектування та моделювання для перевірки працездатності і оцінки отриманих параметрів цифрових пристроїв на ПЛІС.

Дисципліна «Технології проектування комп'ютерних систем» є одною з базових в системі знань та вмінь, що формують бакалавра та спеціаліста із спеціальності 123 "Комп'ютерна інженерія". Навчальний посібник містить ґрунтовний теоретичний матеріал, що супроводжується достатньо великою кількістю графічних зображень, схем та прикладів розв'язання конкретних задач з використанням CASE-засобів для проектування комп'ютерних систем.

Особливою метою вивчення окремих розділів дисципліни «Технології проектування комп'ютерних систем» є набуття студентом навичок у розробці систем дискретних пристроїв із використанням сучасних технологій та високотехнологічних форм навчання, зокрема пов'язаних із використовуванням стендів для на ПЛІС: 1) Програмована логічна інтегральна схема (Лахно В.А.); комп'ютерна система на базі лабораторного стенда DE0; дослідження логічних елементів на стенді DE0 (Лахно В.А.) система автоматизованого проектування QUARTUS II розробка та дослідження комбінаційних схем цифрових автоматів у Quartus II; симуляція проекту в Quartus II (Гусєв Б.С.); призначення та можливості системи MATHCAD (Касаткін Д.Ю.); теоретичне підґрунтя бінарної логіки комп'ютерних систем (смолій в.в.); технології проектування комп'ютерних систем. Інформаційний (програмний) рівень (Місюра М.Д.).

Слід зазначити, що необхідність такого видання вкрай гостра, у зв'язку з відсутністтю україномовного контенту та інших підручників із такої дисципліни.

Розділ 1. ПЛІС - Програмована логічна інтегральна схема

Програмована логіка

Програмована логіка є однією з найбільш динамічних галузей ринку електроніки, зокрема цифрових систем захисту інформації.

Програмована логічна інтегральна схема, ПЛІС (*programmable logic device, PLD*) - електронний компонент, що використовується для створення цифрових інтегральних схем. На відміну від звичайних цифрових мікросхем, логіка роботи ПЛІС не визначається при виготовленні, а задається за допомогою програмування. Для програмування можуть бути використані спеціальні пристрої - програматори, або налагоджувальні середовища, наприклад, системи автоматизованого проектування MAX+PLUS II, Quartus II, Xilinx Foundation Series та ін.

САПР дозволяють задати бажану структуру цифрового пристрою у вигляді принципової електричної схеми або програми на спеціальних мовах опису апаратури Verilog, VHDL, AHDL та ін.

За статистичними даними провідних фірм Aldec Inc., США та Xilinx Inc., США обсяг виробництва програмованих логічних інтегральних мікросхем (ІМС) з 1990 по 2010 роки зріс у понад 20 разів і продовжує зростати.

Через невелику швидкодію й малу кількість еквівалентних логічних вентилів ПЛІС довго займали досить скромну нішу на ринку електронних компонентів. З появою сучасних швидкодіючих ПЛІС надвисокої інтеграції, що працюють на високих тактових частотах, їх вплив на світовий ринок значно розширився.

Сфера застосування ПЛІС постійно розширюється. Діаграма розподілу обсягів споживання ПЛІС в окремих галузях наведена на рис. 1.1. Найбільшим споживачем ПЛІС є галузь телекомунікацій і зв'язку (41 % від всього обсягу виробництва). Друге місце обіймає галузь комп'ютерних мереж, що використовує 26 % обсягу виробництва ПЛІС. Крім того, програмовані логічні ІМС застосовуються в галузі цифрової обробки даних (19 %) та в індустріальному виробництві (14 %). Серед споживачів цих мікросхем можна назвати такі відомі фірми і концерни, як Alcatel, Aser, Asus, IBM, Lockheed, Hewlett Packard, Fujitsu, Hitachi, Silicon Graphics, Texas Instruments, Motorola, Rockwell, Kodak та багато інших.

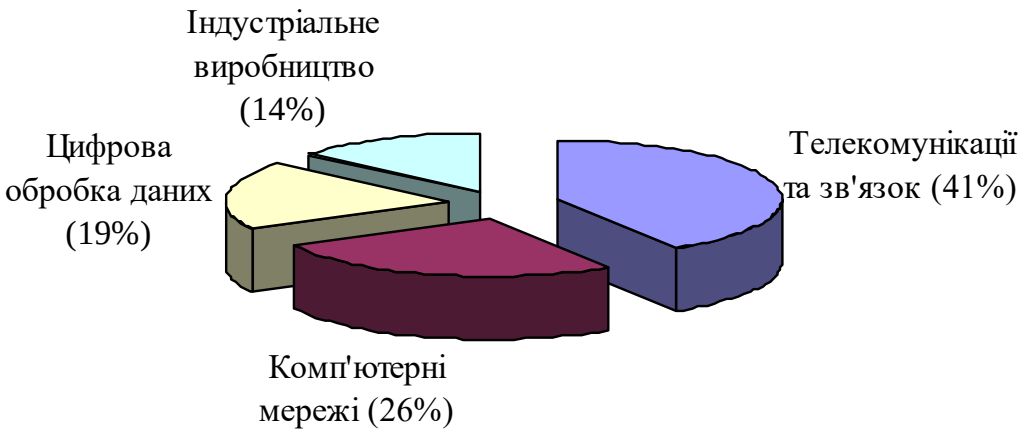


Рис. 1.1. Розподіл обсягів споживання ПЛІС в окремих галузях

ПЛІС діляться на наступні класи, див. рис. 2:

- стандартні (*Standard Programmable Logic Devices - SPLD*) або класичні;
- складні (*Complex PLD*);
- решітки програмованих елементів (*Field Programmable Gate Array - FPGA*);
- програмовані матриці (PLM).

В схемах PLA (*Programmable Logic Array*) обидві логічні матриці є програмованими, що робить їх найбільш гнучкими серед інших ПЛІС.

В PAL (*Programmable Array Logic*) програмованою є логічна матриця «І», а матриця «АБО» є фіксованою. Ці пристрої поєднують гнучкість, властиву PLA, із швидкодією PROM.

В PROM (*Programmable Read Only Memory*) логічна матриця «І» є фіксованою, а матриця «АБО» - може бути програмованою.

До основних характеристик ПЛІС належать [5, 6]:

- кількість системних вентилів;
- швидкодія;
- системна тактова частота;
- максимальне число повторів вводу-виводу;
- енергоспоживання.

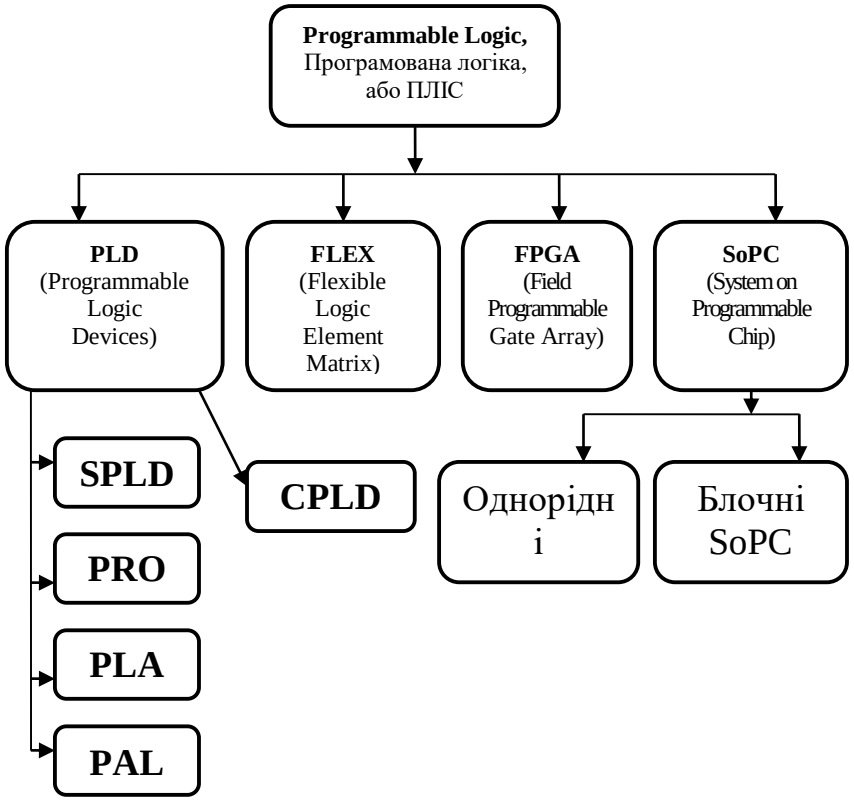


Рис. 1.2. Класифікація пристроїв із програмованою логікою

На рис. 1.3 показані умовні позначення стандартного логічного елементу «І» та його еквівалент у *PLD.*

Одиночна лінія на вході «І» використовується для представлення декількох входів. Вертикальні лінії відповідають сигналам (аргументами) *A*, *B* і *C*. Точка на перетині ліній позначає програмоване з'єднання між вхідними сигналами *A*, *B* і *C* із входом елемента «І». Програмування з'єднання відбувається за допомогою спеціальної технології.

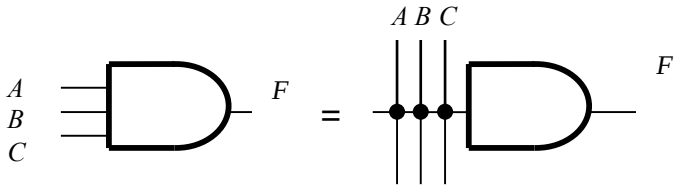


Рис. 1.3. Умовні позначення стандартного логічного елементу «І» та його еквівалент у PLD

Ще один приклад (рис. 1.4) ілюструє реалізацію логічної функції трьох аргументів у ПЛІС (*PLD*).

$$F = A \cdot \overline{B} + \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C}.$$

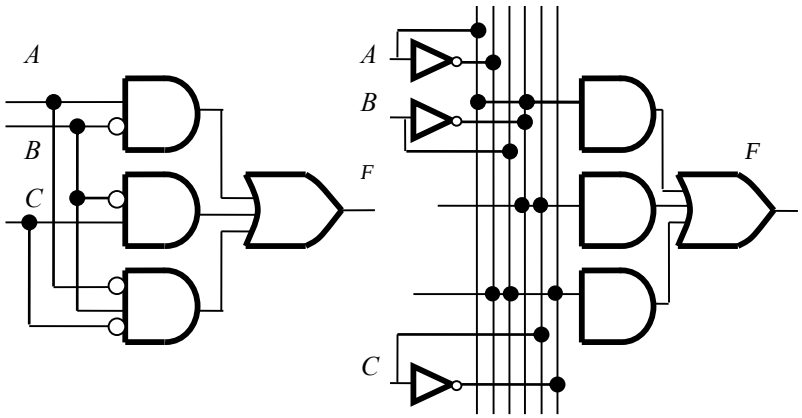


Рис. 1.4. Приклад реалізації логічної функції у PLD

Сьогодні ПЛІС проектують і виробляють кілька десятків провідних фірм. Лідерами у цій галузі є фірми Xilinx (створена у США у 1984 р., займає приблизно 60% ринку), Altera (створена у США у 1983 р., 19%) та Actel (створена у США у 1988 р., 15%).

До перших ПЛІС, які з'явилися на початку 70-х років, відносяться програмовані постійні запам'ятовуючі пристрої (ППЗП - Programmable Read Only Memory - *PROM*). Спочатку PROM використовували винятково для зберігання інформації, пізніше їх стали використовувати для реалізації логічних функцій.

Структура *PROM* містить дві матриці: матрицю DC, яка реалізує функції повного дешифратора, і програмовану матриця «АБО» («OR»). Конструкція *PROM* дозволяє реалізувати логічні функції, представлені в довершеній диз'юнктивній нормальній формі.

З 1971 р. стали випускатися програмовані логічні матриці (ПЛМ – Programmable Logic Array – PLA), які містять дві програмовані матриці (рис. 1.5), одна з яких реалізує функції «І» («AND»), а інша - функції «АБО» («OR»).

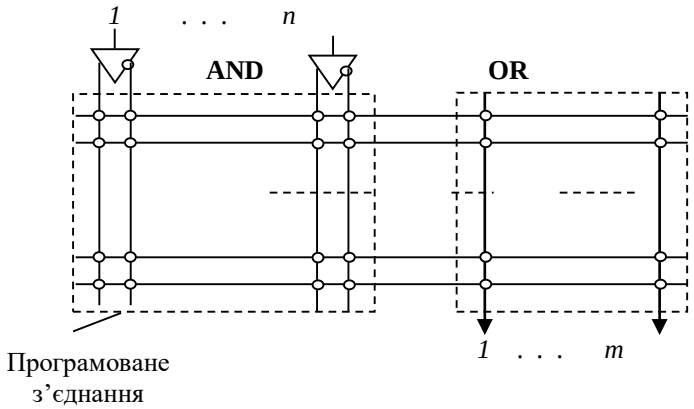


Рис. 1.5. Структура ПЛМ – Programmable Logic Array (PLA)

Схеми PAL для проектування комбінаційної логіки

Приклад реалізації структури *PLA* для логічної функції чотирьох аргументів наведено на рис. 1.6.

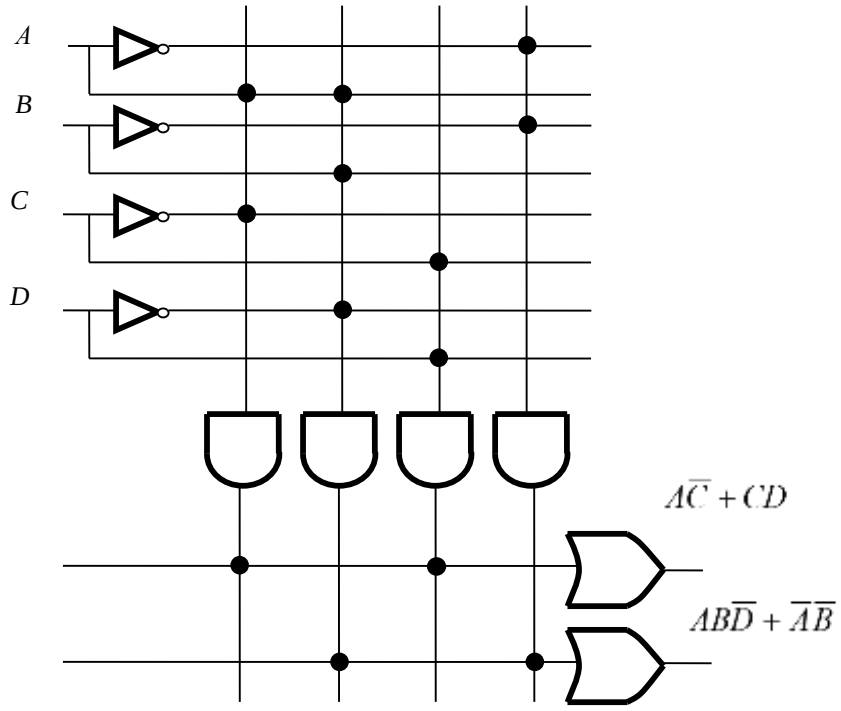


Рис. 1.6. Приклад структури ПЛМ (PLA)

Удосконалювання структури *PLA* привело до створення програмованих матриць логіки (ПМЛ - Programmable Array Logic - *PAL*), у яких, на відміну від *PLA*, програмується тільки матриця «І», а матриця «АБО» має попередньо налагоджену та фіксовану структуру, при якій q проміжних шин зв'язується з одним виходом (рис. 7). Це дозволяє реалізувати матрицю «АБО» у вигляді сукупності q-входових диз’юнкторів. Вихідні буфери, що визначають архітектуру *PAL*, є програмованими макроосередками, які можуть включати інвертор із трьома станами, тригери різних типів, елементи «виключне АБО» та ін.

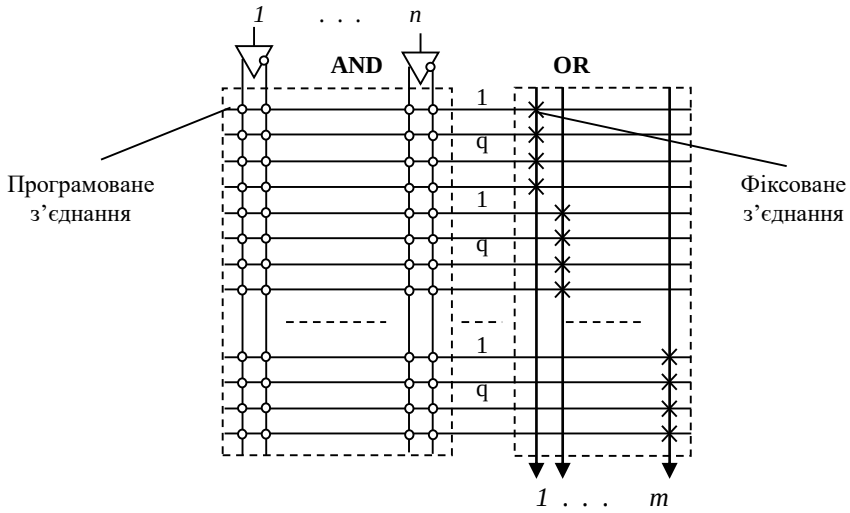


Рис. 1.7. Структура ПМЛ – Programmable Array Logic (PAL)

Удосконалювання технології виробництва ПЛІС призвело до можливості реалізації на одному кристалі декількох *PAL*, що поєднуються у єдину конструкцію програмованими з'єднаннями. Такі ПЛІС одержали назву складних ПЛП (*Complex Programmable Logic Devices, CPLD*).

Загальна структура *CPLD* містить матрицю функціональних блоків *FB* і програмовану матрицю перемикань (*Switch Matrix, SM*). Основні логічні перетворення виконуються в *PAL*-блоках, а матриця перемикань служить лише для передачі сигналів між ними. Також у структурі *CPLD* присутні спеціалізовані входи, зв'язані як з матрицею перемикань, так і з усіма *PAL*-блоками. Ці входи звичайно використовують для передачі глобальних сигналів синхронізації і керування пристроєм.

Проектування за допомогою булевих рівнянь

Перший підхід до розробки ПЛІС ґрунтувався на проектуванні за допомогою булевих рівнянь. Такий спосіб проектування передбачає формування булевих рівнянь чи таблиць істинності для кожного тригера або блока вентилів. Приклад реалізації структури PAL для функції чотирьох аргументів наведено на рис. 1.8.

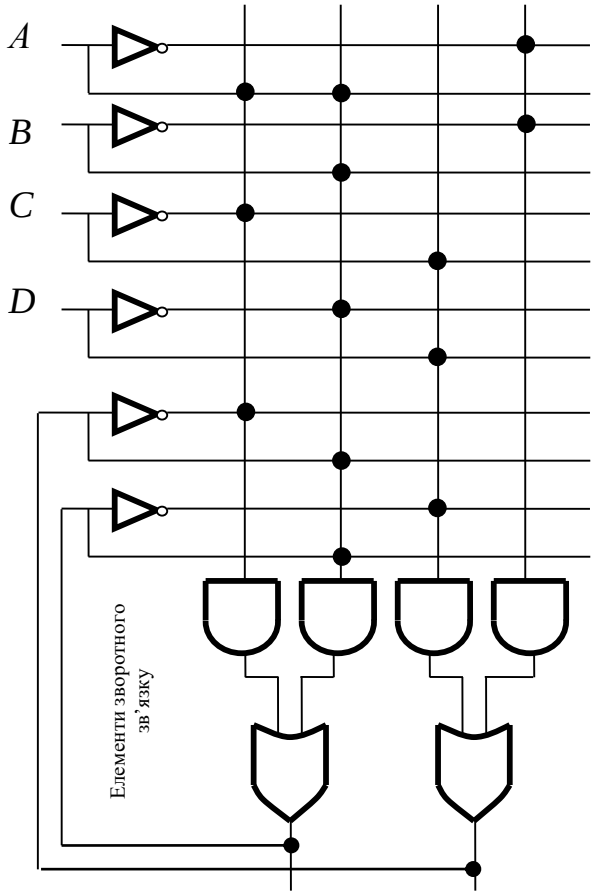


Рис. 1.8. Приклад реалізації структури PAL

Схеми PAL дуже зручні для проектування комбінаційної логіки. Але вони не можуть бути використані для проектування схем без застосування зовнішніх тригерів. Тому в конструкції *PLD* (простих *PLD - simple PLD*) додають тригери, як це показано на рис. 1.9. За допомогою мультиплексорів здійснюється вибір виходу.

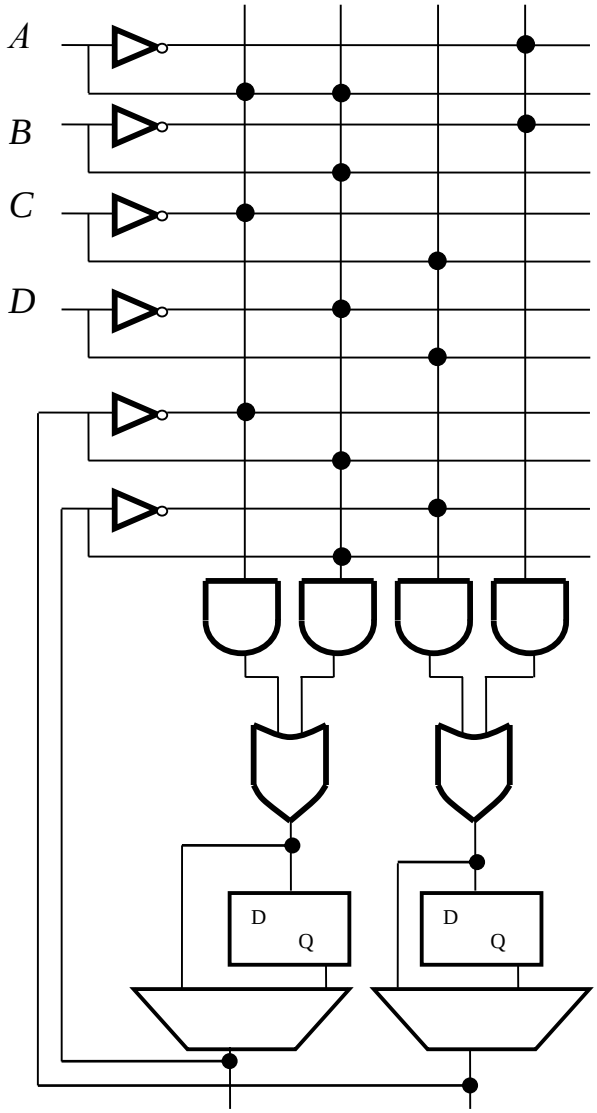


Рис. 1.9. Структура SPLD

Суть проектування ПЛІС полягає в розробці спеціальної програми (у вигляді двійкового потоку "bit-stream"), при завантаженні якої в ПЛІС остання виконує функції спроектованого електронного пристрою.

Опис проектних схем за допомогою булевих рівнянь є прийнятним для малих проектів, що включають десятки або навіть сотні вентилів, але для схем, що базуються на тисячах вентилів, такий опис стає надто громіздким і неефективним [6].

Крім того, схема, що формалізована за допомогою булевих рівнянь, важко піддається декомпозиції.

У традиційній формі схемне проектування є модифікацією методу проектування булевими рівняннями. Цей підхід ґрунтується на застосуванні попередньо синтезованих логічних блоків, таких, як регістри, лічильники, дешифратори та ін. Довгий час метод схемного проектування вважався найкращим вибором, однак з розвитком елементної бази мікросхемотехніки кількість елементарних схем у нових ПЛІС постійно зростала.

Найсучасніший підхід до проектування ПЛІС ґрунтується на застосуванні спеціалізованих мов опису обладнання (*HDL - Hardware*

Description Language). Ці мови, крім звичних конструкцій, таких, як розгалуження, цикли, підпрограми тощо, мають також спеціалізовані засоби - робота з портами, забезпечення паралельності виконання процесів тощо. Найпоширенішими серед HDL стали мови ABEL, Verilog та VHDL.

ABEL (*Advanced Boolean Equation Language* – базова мова логічних рівнянь) є стандартом промислових виробництв, яка розробленим Data I/O Corp. для програмованих логічних пристроїв. ABEL може застосовуватися для опису поведінки систем (за допомогою С-подібних операторів) у різних формах: на основі логічних рівнянь, таблиць істинності, діаграм стану. Порівняно з ABEL мови VHDL та Verilog є складнішими, але більш придатними для опису комп'ютерних систем, які майже рівні за своїми технічними можливостями. В Європі та США ширше застосовується VHDL, в країнах Азії - Verilog. VHDL базується на мові високого рівня Ada. З цієї мови розробниками VHDL було запозичені синтаксис та основні структури.

Наприклад, синтаксис операторів у Verilog подібний синтаксису мови програмування С++. Існують арифметичні (+, -, *, /), логічні (&& - logical **AND**; || - logical **OR**; = - logical **EQUALITY**; != - logical **INEQUALITY**) оператори, корисні для моделювання цифрових схем. При однаковому синтаксисі (\ ~| ^ ~^ & ~&) дані оператори можуть бути бітовими (*bitwize*) і працювати з двома операндами або операторами згортки (*reduction*) і працювати з одним операндом. Тип операції визначається по місту оператора у виразі.

У якості приклада в таблиці 1 показані варіанти опису стандартного **D** – тригеру на мовах VHDL та Verilog. Більш детальному опису цих мов присвячено багато спеціальної літератури [5, 6].

Відповідно до сучасних вимог системи автоматизованого проектування ПЛІС повинні забезпечувати:

• реалізацію однієї чи більше HDL-мов з можливістю введення, редагування та відлагодження початкового тексту програм;

• реалізацію засобів графічного введення проектної схеми, наприклад, за допомогою редактора скінчених автоматів та засобів компіляції графічного представлення в HDL-код;

• реалізацію засобів моделювання поведінки описаного об'єкта;

• реалізацію засобів синтезу бітового потоку з підтримкою широкого класу серій ІМС;

• реалізацію засобів моделювання об'єкта на рівні вентилів;

• реалізацію засобів програмування ІМС.

Таблиця 1.1

Приклади опису D – тригеру на мовах VHDL та Verilog

VHDL	Verilog
library IEEE; use IEEE.std_logic_1164.all;	module dff (data, clk, q); input data, clk;

entity dff is
port (data, clk : in std_logic;
q :out std_logic);
end dff;
architecture behav of dff is
begin
process (clk) begin
if (clk'event and clk = '1') then
q <= data;
end if;
end process;
end behav;
	output q;
	reg q;
	always @(posedge clk)
	q = data;
	endmodule

Розробкою такого програмного забезпечення займається значна кількість фірм, серед яких можна назвати Aldec, Altera, Xilinx, Viewlogic, Mentor, Synopsis, Vantage та ін.

На рис. 10, 11 та 12, відповідно, показані вікна програмних пакетів для синтезу мікросхем ПЛІС - Active-HDL 8.2 (фірма Aldec), Quartus II 9.1 (Altera) та Xilinx ISE 10.1 (Xilinx).

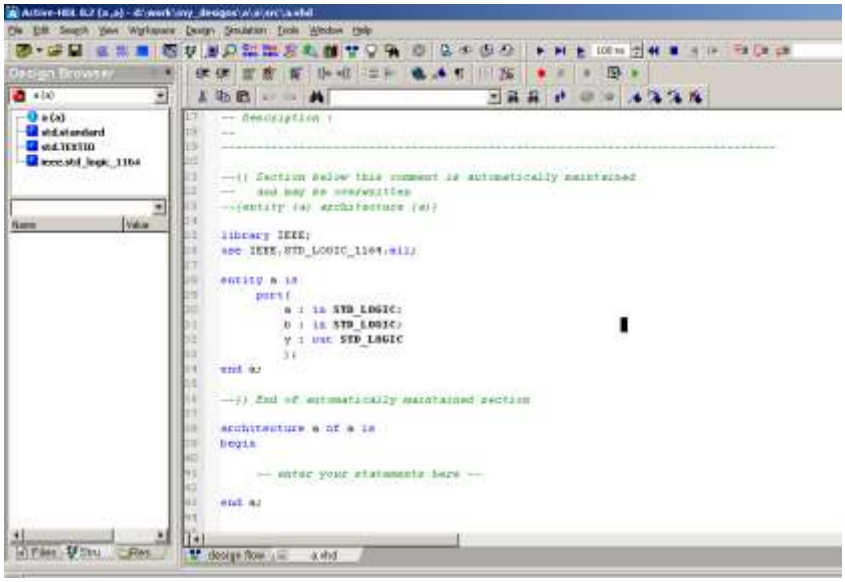


Рис. 1.10. Загальний вигляд пакету Active-HDL 8.2 (фірма Aldec)

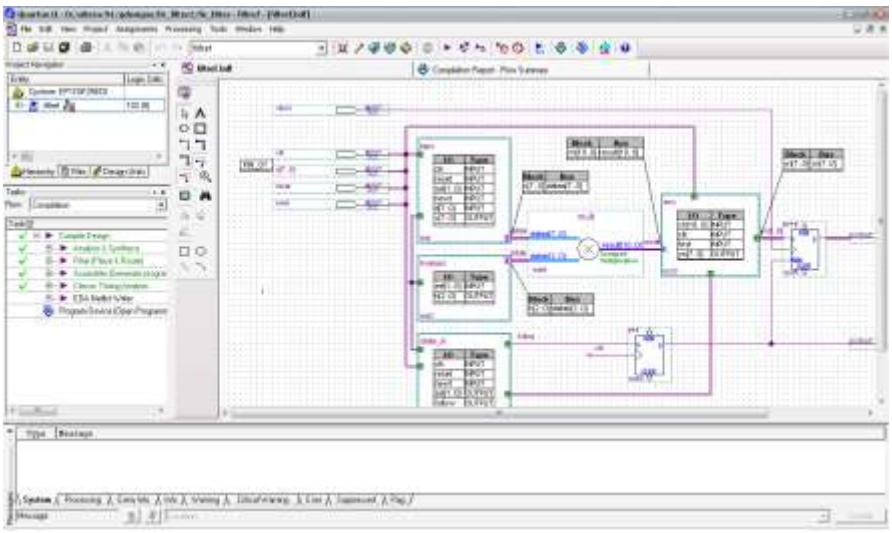


Рис. 1.11. Загальний вигляд пакету Quartus II 9.1 (Altera)

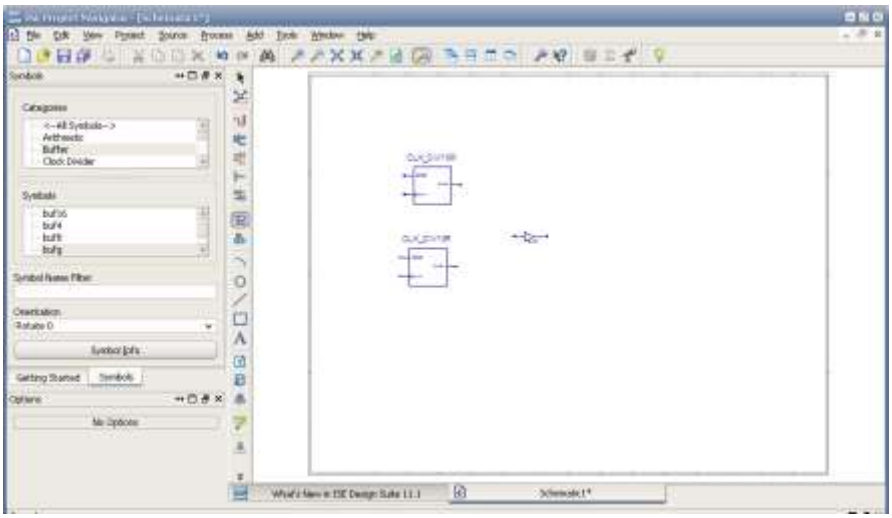


Рис. 1.12. Загальний вигляд пакету Xilinx ISE 10.1 (Xilinx)

Основним обмеженням цих програмних пакетів є те, що вони не мають засобів синтезу для мікросхем інших виробників ПЛІС.

Крім того, розроблено ряд програмних продуктів, що реалізують ту чи іншу частину загальних вимог до САПР ПЛІС. Застосування цих продуктів вимагає в кожному конкретному випадку розв'язання проблем сумісності між пакетами, тому перевагу слід надавати пакетам, що реалізують процес проектування ПЛІС.

Серед номенклатури ПЛІС найбільшого поширення набули мікросхеми, що виготовлені за технологіями *FPGA* та *CPLD*. Абревіатура FPGA розшифровується як *Field Programmable Gate Array* - програмована користувачем вентильна матриця. Загальну структуру кристала FPGA-мікросхеми наведено на рис. 1.13.

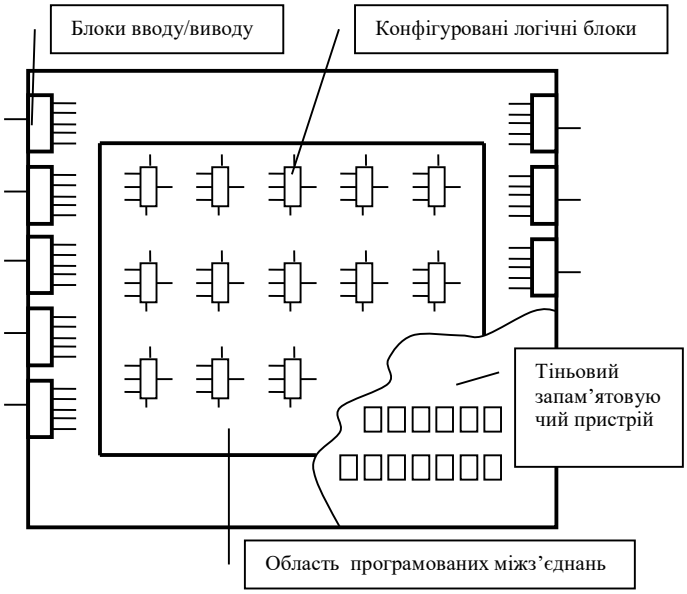


Рис. 1.13. Загальна структура кристала FPGA-мікросхеми

По периферії верхнього шару кристала розміщуються блоки вводу/виводу, що можуть бути запрограмовані для виконання функцій буферів:

- вхідного;
- вихідного;
- запам'ятовуванням;
- ін.

У деяких серіях FPGA-IMC рівень напруги на блоках вводу/виводу може відрізнятись, що дає змогу зв'язувати різні за живленням інтерфейси FPGA. У центрі кристала у вигляді матриці розміщено конфігуровані логічні блоки. Швидкодія мікросхем визначається часовою затримкою "вхід-вихід" одного конфігурованого логічного блоку.

Конфігуровані логічні блоки FPGA-IMC можуть генерувати будь-яку логічну функцію чотирьох аргументів або дві логічні функції трьох аргументів. Аргументи для логічних функцій можуть надходити у вигляді сигналів з чотирьох входів та виходу запам'ятовуючого елемента. Область між конфігурованими логічними блоками називається областю програмованих з'єднань і являє собою розвинену ієрархію ліній зв'язку, в місцях перетину яких розміщено спеціальні швидкодіючі транзистори.

Функція області міжз'єднань полягає в забезпеченні зв'язку між будь-якими виводами конфігурованих логічних блоків та блоками вводу/виводу.

Нижній шар кристала займає тіньовий запам'ятовуючий пристрій, інформація в елементах якого і визначає логічні функції конфігурованих логічних блоків, конфігурацію блоку вводу/виводу та маршрути міжз'єднань.

Широкий діапазон FPGA-технології дозволяє проектувати на їх основі широкий спектр електронних пристроїв, серед яких: мікропрограмні пристрої керування, скінченні автомати, універсальні та спеціалізовані

процесори, пристрої цифрової обробки сигналів, засоби поєднання різних за живленням інтерфейсів, перетворювачі кодів, периферійні контролери, тощо.

Архітектура ПЛІС типу *CPLD* зображена на рис. 1.14. ПЛІС типу *CPLD* - XC9500 має три групи виводів:

- виводи JTAG-порта для програмування та периферійного сканування;
- порти вводу/виводу;
- керуючі виводи: сигнал тактування - GCK, установлення/скидання - GSR, керування третім станом - GTS.

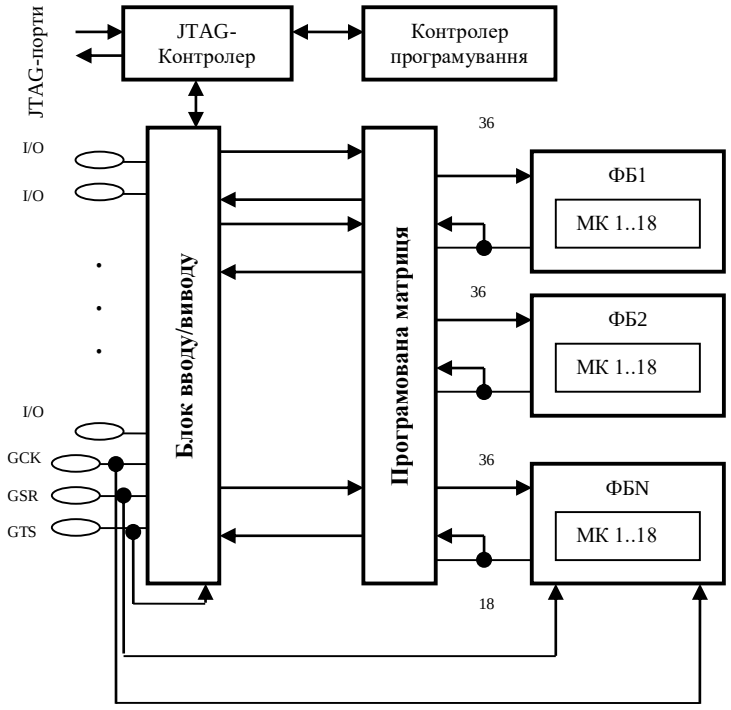


Рис. 1.14. Архітектура ПЛІС типу CPLD - XC9500

Блоки вводу/виводу забезпечують буферизацію всіх входів та виходів ПЛІС. Кожен функціональний блок (ФБ) містить 18 макрокомірок із структурою «36 входів - 1 вихід» і дозволяє формувати 18 логічних функцій для будь-якої комбінації з 36 аргументів. Матриця перемикань забезпечує подавання будь-яких вхідних сигналів та вихідних сигналів ФБ на входи ФБ, а також подавання вихідних сигналів ФБ на блоки вводу/виводу.

CPLD-технології широко застосовуються для проектування нестандартних арифметико-логічних пристроїв, дешифраторів, мультиплексорів та ін.

У порівнянні з *FPGA*, до недоліків *CPLD* слід віднести: малу кількість системних вентилів та значне енергоспоживання. Переваги технології *CPLD* полягають у вищій швидкодії та забезпеченні можливості встановлення

захисту від копіювання. Важливою перевагою також є те, що програмні засоби для розробки та синтезу систем на базі *CPLD* поширюються вільно.

Основу функціонування сучасних цифрових систем технічних комплексів захисту інформації з обмеженим доступом складає принцип мікропрограмного керування, відповідно до якого координування взаємодії всіх блоків системи виконує керуючий автомат. На практиці широко застосовується реалізація керуючих автоматів у вигляді автомата з «жорсткою» логікою, тобто автомати Мілі і Мура. Вихідними даними для синтезу схеми керуючого автомату служать алгоритми керування операційним пристроєм, задані, як правило, у вигляді граф-схем. Зараз для реалізації схем керуючих автоматів використовуються різноманітні програмовані логічні пристрої, які дозволяють суттєво підвищити швидкодію, надійність, компактність цифрових систем.

В даний час для реалізації логічних схем цифрових автоматів широко використовуються програмувальні логічні пристрої - *PLA*, *PAL*, *PLD*, *PROM*, і схеми середнього ступеню інтеграції – дешифратори та мультиплексори. Однією з важливих задач, що виникають при синтезі керуючих автоматів, є оптимізація апаратурних витрат в логічній схемі.

Під областю ефективної реалізації структури керуючого автомату розуміється таке сполучення характеристик ГСА, обумовлених її розмірністю і конфігурацією, при якому реалізація автомата з даною структурою забезпечує найменші апаратурні витрати в його логічній схемі.

Сучасні мікросхеми ПЛІС, виконані за 0,22-мікронної технології, здатні працювати на частотах до 300 Мгц і реалізують до 3 млн. еквівалентних логічних вентилів. Різке збільшення потужності сучасних ПЛІС створило передумови для реалізації не тільки простих контролерів й інтерфейсних вузлів, але й систем цифрової обробки сигналів, пристроїв управління в реальному часі найскладнішими технологічними процесами, інтелектуальних контролерів і нейроних мереж.

Поява ПЛІС із наднизьким рівнем енергоспоживання відкриває широкі можливості за їх використання в мобільних мережах та портативних комп'ютерах.

Недоліком ПЛІС є їх енергозалежність. Програма зазвичай зберігається в енергозалежній пам'яті, при кожнім включенні живлення мікросхеми необхідно заново конфігурувати її. Такі мікросхеми виробляють фірми Xilinx й Altera.

Завантаження конфігурації відбувається за допомогою завантажника, що може бути убудований й у саму *FPGA*. Фірми виробники пропонують конфігураційні ПЗУ що при включенні живлення завантажує конфігурацію в ПЛІС. Фірми Actel й Lattice Semiconductor пропонують мікросхеми *FPGA*, в яких конфігурація зберігається енергонезалежній Flash-пам'яті, в таких ПЛІС програма зберігається при зникненні електроживлення.

Альтернативою ПЛІС є: програмовані логічні контролери; базові матричні кристали, що вимагають заводського виробничого процесу для програмування; ASIC - спеціалізовані замовні великі інтегральні схеми, які при багатосерійному та одиничному виробництві істотно дорожче; спеціалізовані комп'ютери або спеціалізовані процесори (наприклад, цифровий сигнальний процесор).

Сучасний рівень розвитку ПЛІС, їх швидкодія та ємність, дозволяє реалізувати системи на одному кристалі, що дає більший виграш у продуктивності, ніж за використання класичних мікроконтролерів та ВІС, а також дозволяє прискорити швидкість процесів передачі даних, що зокрема є критичним для різноманітних сучасних цифрових систем захисту інформації. Застосування ПЛІС обумовлює простоту розробки та налагодження системи, а також дозволяє адаптувати систему під визначений об'єкт управління з максимальною продуктивність та швидкодією.

Питання для самоперевірки

1. Дайте визначення ПЛІС.
2. На яки класи поділяють ПЛІС?
3. У яких галузях використовуються ПЛІС?
4. Які програмні пакети використовують для розробки ПЛІС?
5. Які мови програмування використовують для розробки пристроїв із програмованою логікою?
6. Які фірми є лідерами на ринку програмованих логічних пристроїв?
7. Які переваги та недоліки мають програмовані логічні пристрої?
8. Яким чином виконується програмування з'єднань у матрицях ПЛІС.
9. Як розшифровуються абревіатури *PLA, PAL, PLD, PROM*?

Розділ 2. Комп’ютерна система на базі лабораторного стенда DE0 з дослідження схем різного ступеню складності

2.1. Технологічна структура системи

Стенд DE0 має багато особливостей, які дозволяють користувачеві виконувати широкий спектр дій з дослідження схем різного ступеню складності, від простих до складних мультимедійних проектів.

На рис 2.1. наведена фотографія налагоджувального стенду DE0, а також показані роз'єми та ключові компоненти стенду.

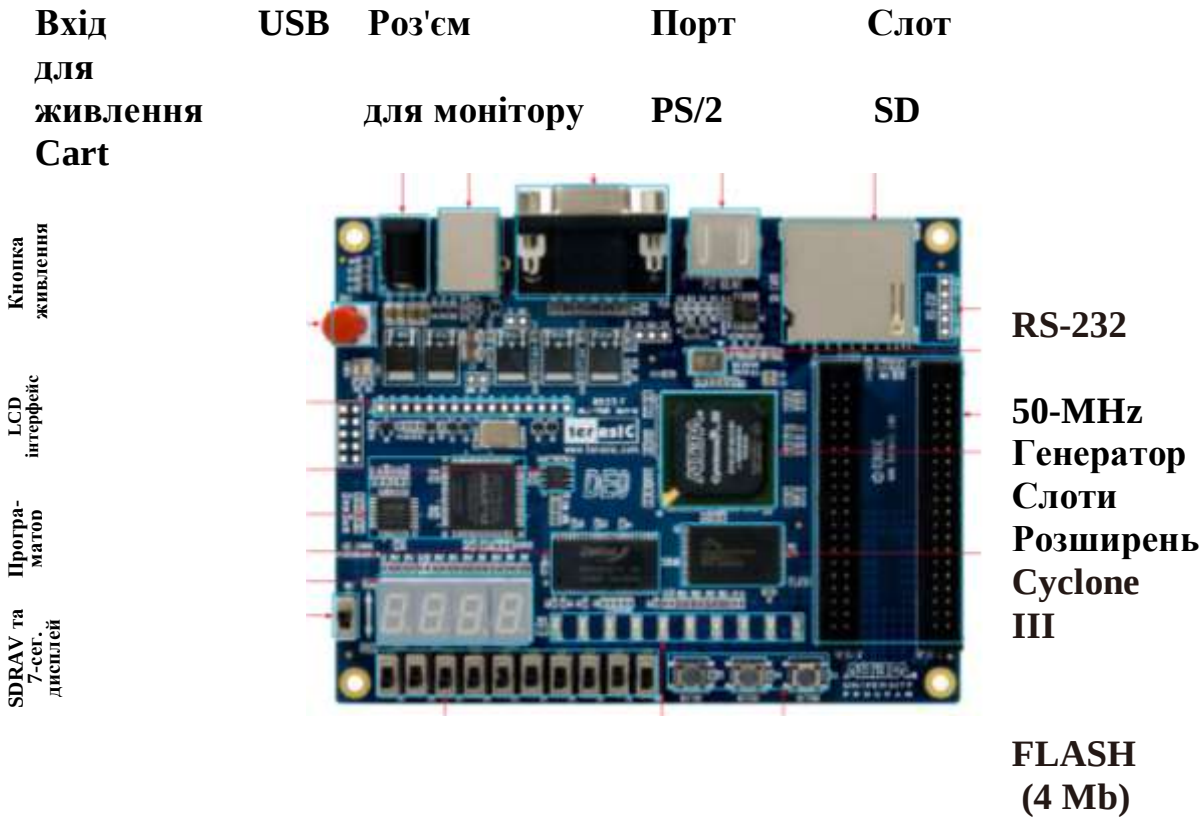


Рис. 2.1. Роз’єми та ключові компоненти стенду DE0

Стенд DE0 поставляється з відповідною панеллю управління, див. рис. 2.2, яка дозволяє користувачам отримувати доступ до різних компонентів на платі від комп'ютера. Комп'ютер з’єднується DE0 через USB-порт. Стенд може використовуватися для перевірки функціональності компонентів на платі або як інструмент налагодження при розробці RTL коду.

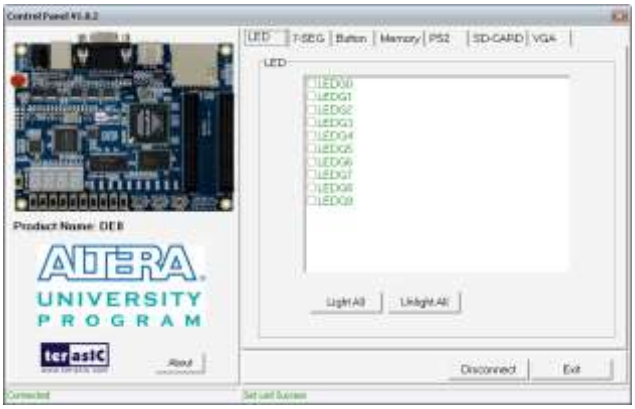


Рис. 2.2. Загальний вигляд панелі управління стендом DE0

Наприклад, вибір перемикачів та кнопок з контрольної панелі здійснюється у вікні наведеному на наступному рис. див. рис. 2.3.

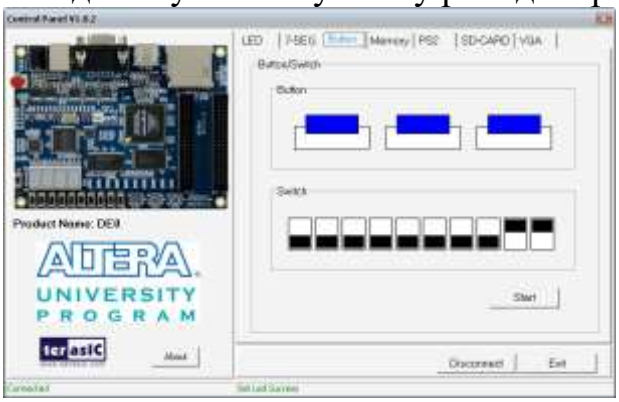


Рис. 2.3. Управляння перемикачами та кнопками

Ці кнопки дозволяють контролювати статус перемикачів і кнопок у реальному часі і показують статус у графічному інтерфейсі користувача. Це може бути використане, щоб перевірити функціональність вимикачів і кнопок.

На платі DE0 є три кнопки вимикачів. Три виходи Button 0, Button 1, Button2 безпосередньо підключені до Cyclone III FPGA. Кожна з кнопок (комутатор) забезпечує високий логічний рівень (3,3 вольта), коли вона не натиснута, і забезпечує низький логічний рівень (0 вольт) при вимкнені.

На стенді є також 10 перемикачів (повзунки). Коли перемикач знаходиться в положенні DOWN (ближче до краю плати) він забезпечує низький логічний рівень (0 вольт), а коли перемикач знаходиться у верхньому положенні він забезпечує високий логічний рівень (3,3 вольта) .

Також на платі розташовані 10 світло діодів. На рис. 2.4 і рис. 1.5 показано зв'язки між кнопками, перемикачами, і Cyclone III.

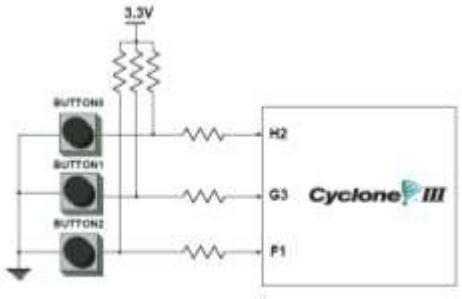


Рис. 2.4. Зв'язки між кнопками та Cyclone III

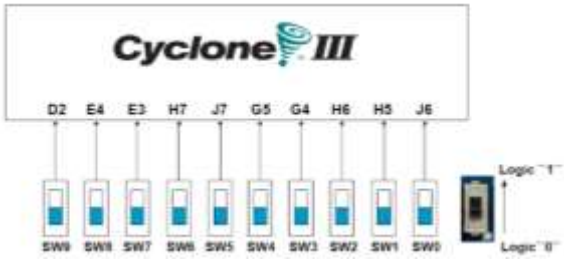


Рис. 2.5. Зв'язки між перемикачами та Cyclone III

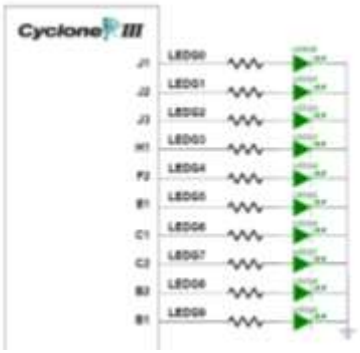


Рис. 2.6. Зв'язки між світло діодами та Cyclone III

Список контактних імен Cyclone III, які пов'язані з перемикачами наведено в таблиці 2.1. Крім того, контакти використовуються для підключення до кнопкових перемикачів та світло діодів, див. табл. 2.2 та табл. 2.3, відповідно.

Таблиця 2.1

Призначення контактів (вихід) для перемикачів

Перемикач	Контакт	Опис
SW[0]	PIN_J6	Перемикач SW[0]
SW[1]	PIN_H5	Перемикач SW[1]
SW[2]	PIN_H6	Перемикач SW[2]
SW[3]	PIN_G4	Перемикач SW[3]
SW[4]	PIN_G5	Перемикач SW[4]
SW[5]	PIN_J7	Перемикач SW[5]
SW[6]	PIN_H7	Перемикач SW[6]
SW[7]	PIN_E3	Перемикач SW[7]

SW[8]	PIN_E4	Перемикач SW[8]
SW[9]	PIN_D2	Перемикач SW[9]

Таблиця 2.2

Призначення контактів (вихід) для кнопок

Кнопка	Контакт	Опис
Button [0]	PIN_H2	Кнопка [0]
Button [1]	PIN_G3	Кнопка [1]
Button [2]	PIN_F1	Кнопка [2]

Таблиця 2.3

Призначення контактів (вихід) для світло діодів

Перемикач	Контакт	Опис
LEDG [0]	PIN_J1	Світло діод LEDG [0]
LEDG [1]	PIN_J2	Світло діод LEDG [1]
LEDG [2]	PIN_J3	Світло діод LEDG [2]
LEDG [3]	PIN_H1	Світло діод LEDG [3]
LEDG [4]	PIN_F2	Світло діод LEDG [4]
LEDG [5]	PIN_E1	Світло діод LEDG [5]
LEDG [6]	PIN_C1	Світло діод LEDG [6]
LEDG [7]	PIN_C2	Світло діод LEDG [7]
LEDG [8]	PIN_B2	Світло діод LEDG [8]
LEDG [9]	PIN_B1	Світло діод LEDG [9]

DE0 плата має чотири 7-сегментний дисплеї. Ці дисплеї розташовані по два у групі, з метою відображення чисел різних розмірів. Як показано на рис. 2.7, сім сегментів підключені до контактів на Cyclone III. Застосовуючи логічний рівень «0» відповідний сегмент світиться, та навпаки коли застосовується високий логічний, тобто «1» сегмент вимикається.

Кожен сегмент на дисплей ідентифікується індексом від 0 до 6, див. рис. 2.7. Крім того, десяткові точки визначається як DP. У таблиці 2.4 показані зв'язки між контакти FPGA у 7-сегментному дисплеї.

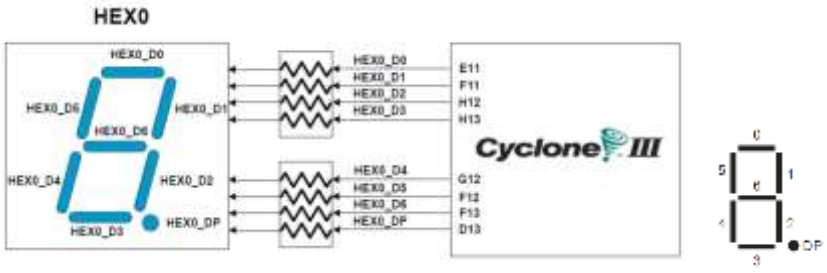


Рис. 2.7. Зв'язки між 7-сегментним дисплеєм та Cyclone III

Таблиця 2.4

Призначення контактів (вихід) для 7- ссегментного дисплею

Перемикач	Контакт	Опис
HEX0_D[0]	PIN_E11	Індекс дисплею 0 [0]
HEX0_D[1]	PIN_F11	Індекс дисплею 0 [1]
HEX0_D[2]	PIN_H12	Індекс дисплею 0 [2]
HEX0_D[3]	PIN_H13	Індекс дисплею 0 [3]
HEX0_D[4]	PIN_G12	Індекс дисплею 0 [4]
HEX0_D[5]	PIN_F12	Індекс дисплею 0 [5]
HEX0_D[6]	PIN_F13	Індекс дисплею 0 [6]
HEX0_DP	PIN_D13	Крапка дисплею 0 DP
HEX1_D[0]	PIN_A13	Індекс дисплею 1 [0]
HEX2_D[1]	PIN_B13	Індекс дисплею 1 [1]
HEX3_D[2]	PIN_C13	Індекс дисплею 1 [2]
HEX4_D[3]	PIN_A14	Індекс дисплею 1 [3]
HEX5_D[4]	PIN_B14	Індекс дисплею 1 [4]
HEX6_D[5]	PIN_E14	Індекс дисплею 1 [5]
HEX1_D[6]	PIN_A15	Індекс дисплею 1 [6]
HEX1_DP	PIN_B15	Крапка дисплею 1 DP
HEX2_D[0]	PIN_D15	Індекс дисплею 2 [0]
HEX2_D[1]	PIN_A16	Індекс дисплею 2 [1]
HEX2_D[2]	PIN_B16	Індекс дисплею 2 [2]
HEX2_D[3]	PIN_E15	Індекс дисплею 2 [3]
HEX2_D[4]	PIN_A17	Індекс дисплею 2 [4]
HEX2_D[5]	PIN_B17	Індекс дисплею 2 [5]
HEX2_D[6]	PIN_F14	Індекс дисплею 2 [6]
HEX2_DP	PIN_A18	Крапка дисплею 2 DP
HEX3_D[0]	PIN_B18	Індекс дисплею 3 [0]
HEX3_D[1]	PIN_F15	Індекс дисплею 3 [1]
HEX3_D[2]	PIN_A19	Індекс дисплею 3 [2]
HEX3_D[3]	PIN_B19	Індекс дисплею 3 [3]
HEX3_D[4]	PIN_C19	Індекс дисплею 3 [4]
HEX3_D[5]	PIN_D19	Індекс дисплею 3 [5]
HEX3_D[6]	PIN_G15	Індекс дисплею 3 [6]
HEX3_DP	PIN_G16	Крапка дисплею 3 DP

До плати DE0 входить генератор із тактовою частотою 50 МГц. Цей генератор підключено до ПЛІС для синхронізації роботи елементів комбінаційної логіки. У таблиці 2.5 наведені контакти генератора.

Таблиця 2.5

Призначення контактів (вихід) для генератора кнопок

Генератор	Контакт	Опис
CLOCK_50	PIN_G21	Вхід 50 МГц
CLOCK_50_2	PIN_B12	Вхід 50 МГц

Більш детальний опис стенду DE0 можливо знайти на сайті компанії Terasic Technologies - http://www.terasic.com.tw/en/

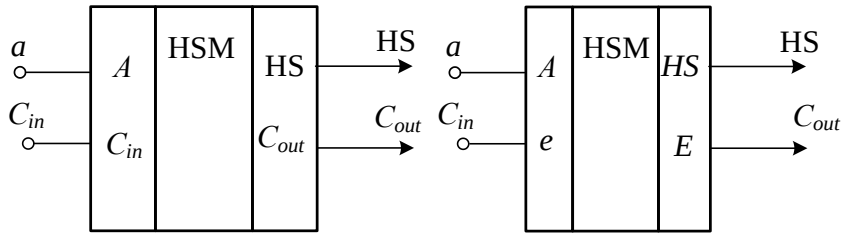
Питання для самоперевірки

1. Основні компоненти плати стенду DE0?
2. Які інтерфейси має стенд DE0?

2.2. Розробка та дослідження двійкових суматорів в середовищі QUARTUS II

Серед найпростіших арифметичних пристроїв можна виділити напівсуматори, які відносяться до класу комбінаційних цифрових пристроїв. Напівсуматори використовуються для побудови повних двійкових суматорів, для організації лічильників на базі лічильної схеми, для виконання мікрооперації «інкремент», тощо.

Умовне графічне позначення (УГП) напівсуматора на функціональних схемах приведено на рис.2.8. Відповідно до рис.2.8,а напівсуматор (*HSM – half-adder*) має два входи (*A* і C_{in}) та два виходи (*HS* і C_{out}). Вхід *A* призначений для підключення операнду, а вхід C_{in} – для підключення вхідного переносу (*input carry*). Вихід *HS* є виходом напівсуми (*half-sum*), а вихід C_{out} – виводом вихідного переносу (*output carry*). Для більш компактного позначення на логічних і функціональних схемах вивід вхідного переносу будемо позначати літерою «*e*», а вихідний перенос – літерою «*E*» (рис.2.8,б).



а) *б)*

Рис.2.8. Умовні графічні позначення напівсуматора

Таблиця істинності напівсуматора приведена в табл.2.6.

Таблиця 2.6

Таблиця істинності напівсуматора

a	*e* (C_{in})	*HS*	*E* (C_{out})
0	*0*	*0*	*0*
0	*1*	*1*	*0*
1	*0*	*1*	*0*
1	*1*	*0*	*1*

З таблиці можна отримати логічні вирази в булевому базисі, за допомогою яких можна реалізувати схему напівсуматора:

$$HS = \bar{a}e + a\bar{e} = \overline{\bar{a}\bar{e} + ae}; \quad E = C_{out} = ae.$$

За допомогою логічної функції «додавання за модулем два» (*XOR*) можна також отримати логічні формули для напівсуми:

$$HS = a \oplus e = \bar{a} \oplus \bar{e} = \overline{a \oplus e}; = \overline{\bar{a} \oplus e} \ .$$

Один з варіантів логічної схеми напівсуматора приведено на рис.2.9

Крім того, логічна схема напівсуматора може бути побудована на базі типових комбінаційних схем дешифратора або мультиплексора. На рис.2.10 приведена логічна схема напівсуматора на базі дешифратора 2→4.

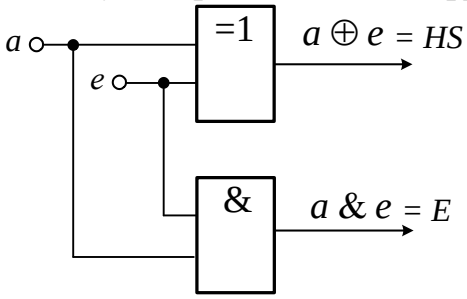


Рис.2.9. Логічна схема напівсуматора

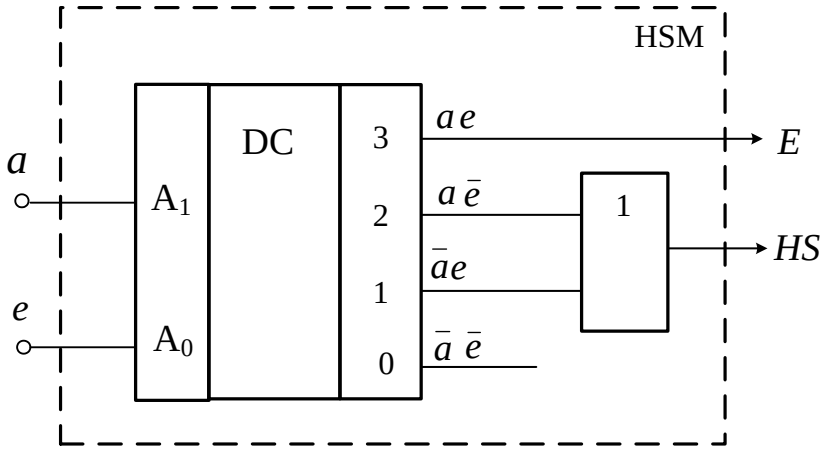


Рис.2.10. Логічна схема напівсуматора на базі дешифратора 2→4

Аналогічним чином відбувається побудова і функціонування напіввіднімачів (*half-subtractor*), які призначені для виконання мікрооперації «декремент». УГП і таблиця істинності напіввіднімача приведено відповідно на рис.2.11 і в табл.2.8. На УГП напіввіднімача використовуються такі позначення:

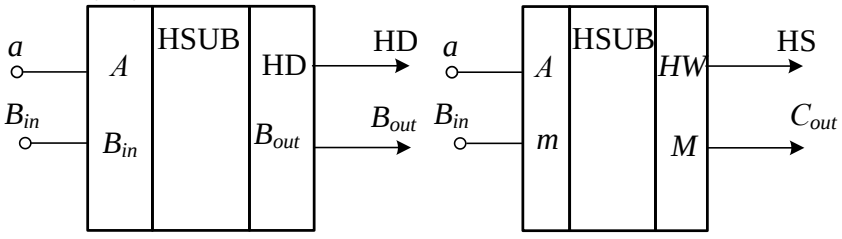
- *a* – операнд;

- *m, B_{in}* – позначення вхідної позики з сусіднього молодшого розряду (*input borrow*);

- *M, B_{out}* – позначення вихідної позики з сусіднього старшого розряду (*output borrow*);

- *HD, HW* – вихід напіврізниці (*half-difference*).

Під час розглядання принципів функціонування арифметичних пристроїв літера «*B*» буде використовуватися для позначення другого операнду арифметичних операцій, тому далі виоди вхідної, вихідної позики та напіврізниці будуть позначатися відповідно *m, M* та *HW*.



а) *б)*

Рис.2.11. Умовні графічні позначення напіввіднімача

Таблиця 2.8

Таблиця істинності напіввіднімача

a	*m* (B_{in})	*HW (HD)*	*M* (B_{out})
0	*0*	*0*	*0*
0	*1*	*1*	*1*
1	*0*	*1*	*0*
1	*1*	*0*	*0*

Функціонування напіввіднімачів характеризується логічними виразами, що можуть бути отримані з таблиці істинності (табл.2.8):

$$HW = \bar{a}m + a\bar{m} = \overline{\bar{a}\bar{m} + am};\ \ M = B_{out} = \bar{a}e.$$

$$HW = a \oplus m = \bar{a} \oplus \bar{m} = \overline{a \oplus m};= \overline{\bar{a} \oplus m}\ .$$

Аналізуючи формули реалізації напівсуматора і напіввіднімача, можна зробити висновок, що логічні вирази для виводів напівсуми і напіврізниці описуються однаковими логічними формулами, а формули для реалізації вихідних переносу і позики відрізняються тільки інверсією змінної «*a*». Це необхідно враховувати при проектуванні комбінованого арифметичного пристрою напівсуматора - напіввіднімача, в якому при виконанні мікрооперації «інкремент» на вивід підключення операнду надходить змінна «*a*», а при виконанні мікрооперації «декремент» на цей вхід необхідно підключити $\bar{a}$.

Аналіз функціонування цифрових пристроїв можна виконувати з використанням моделювання поведінки таких пристроїв, тобто визначення реакції пристрою на зміни вхідних сигналів за допомогою будь-якої системи логічного моделювання. В якості альтернативи САПР QUARTUS II, наприклад, нижче приведені схеми та результати моделювання напівсуматора, логічна схема якого приведена на рис.2.9. Моделювання виконано з використанням системи схемотехнічного проектування MicroCap.

При цьому необхідно відмітити, що для завантаження логічної схеми до плати стенда DE0 необхідно спочатку розробити поведінкову модель пристрою. Ця модель часто будується на функціональному, а не на логічному рівні опису. Тому для того, щоб бачити взаємозв’язок між логічною схемою, поведінковою моделлю та результатами функціонування пристрою, що досліджується, далі опис досліджуваних пристроїв буде приведений в такій послідовності:

- логічна схема пристрою в середовищі MicroCap;
- поведінкова модель в середовищі САПР QUARTUS II;
- результати моделювання пристрою в середовищі MicroCap;
- результати моделювання пристрою в середовищі САПР QUARTUS II.

Два останні пункти використовуються для підтвердження відповідності поведінкової моделі результатам моделювання.

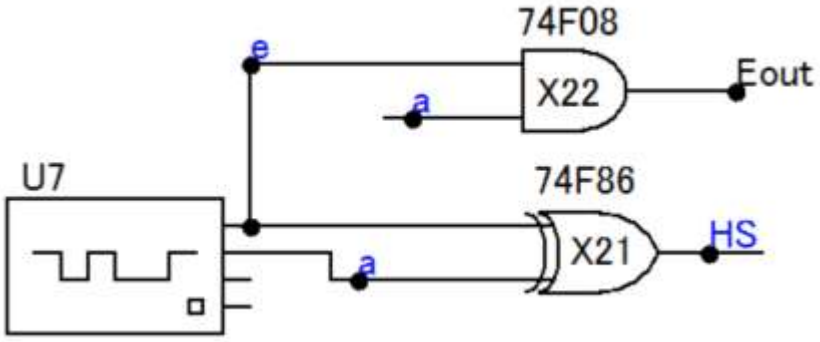


Рис.2.12. Схема напівсуматора, що моделюється

Відповідно до схеми напівсуматор реалізований на базі логічних елементів транзисторно-транзисторної логіки (ТТЛ) серії SN74F08 (КР1531ЛИ1) і SN74F86 (КР1531ЛП5). Результати моделювання в статичному режимі приведені на рис.2.13. Моделювання статичного режиму функціонування пристрою передбачає використання нульових затримок логічних елементів або використання затримок, значення яких набагато менше за тривалість вхідних сигналів ($t_\Delta << t_{вх}$, де t_Δ – час затримки логічного елемента; $t_{вх}$ – тривалість вхідних сигналів). Таким чином, під час моделювання статичних режимів перевіряється відповідність функціонування цифрового пристрою таблиці істинності або таблиці переходів.

Моделювання динамічних режимів функціонування цифрових пристроїв передбачає використання затримок логічних елементів, сумірних з тривалістю вхідних сигналів. В цьому режимі визначаються значення часу спрацьовування, встановлення, підготовки пристрою, виявляється наявність гонок сигналів в схемі, тощо.

На рис.2.13 представлені результати моделювання напівсуматора в динамічному режимі. З часової діаграми функціонування пристрою можна визначити час спрацьовування, який на рис.2.14 показаний за допомогою виносок.

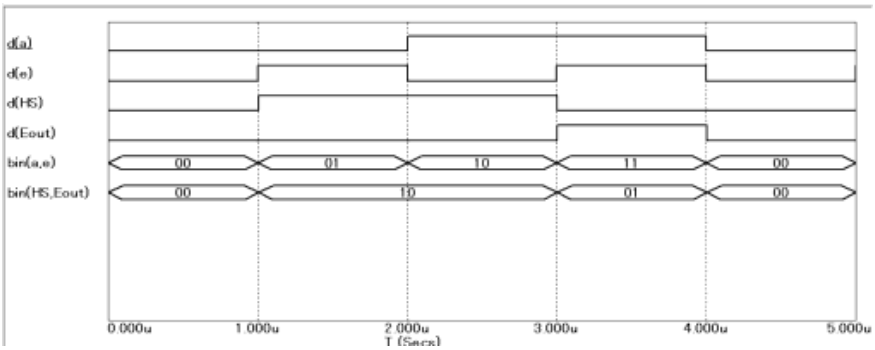


Рис.2.13. Результати моделювання напівсуматора в статичному режимі

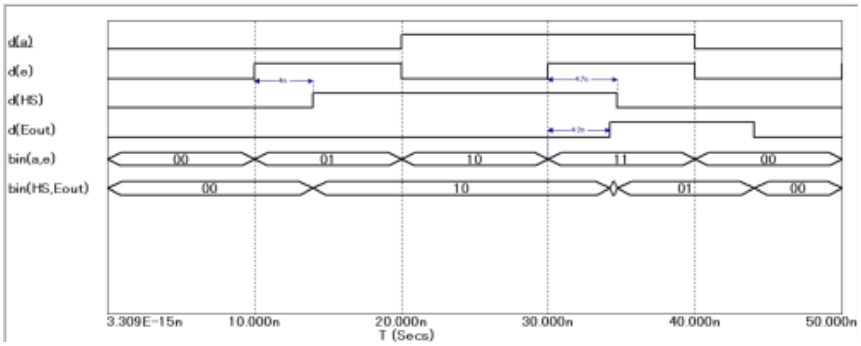


Рис.2.14. Результати моделювання напівсуматора в динамічному режимі

Для виконання арифметичних операцій в комп'ютерних системах використовуються повні двійкові суматори.

Під час виконання арифметичної операції додавання двох n- розрядних операндів A (a_{n-1}, …, a_0) та B (b_{n-1}, …, b_0) в i- тому розряді результат операції може бути визначений за формулою:

$$W \cdot E_i + S_i = a_i + b_i + e_i,$$

де n – кількість розрядів операндів;

a_i, b_i – значення i-тих двійкових цифр операндів A і B;

e_i – значення переносу в i-тий біт з молодшого сусіднього розряду;

S_i – сума двійкових операндів в i-тому розряді;

E_i – вихідний перенос з i-того розряду до сусіднього старшого розряду;

W – вага вихідного переносу E ($W = 2^n$).

При цьому E_i визначається за формулою:

$$E_i = \begin{cases} 0, \text{ якщо } (a_i + b_i + e_i) < W; \\ 1, \text{ якщо } (a_i + b_i + e_i) \geq W. \end{cases}$$

Очевидно, що для двійкового однорозрядного суматора ($n = 1$) $W = 2$.

УГП однорозрядного двійкового суматора, яке використовується на функціональних схемах, наведено на рис.2.15.

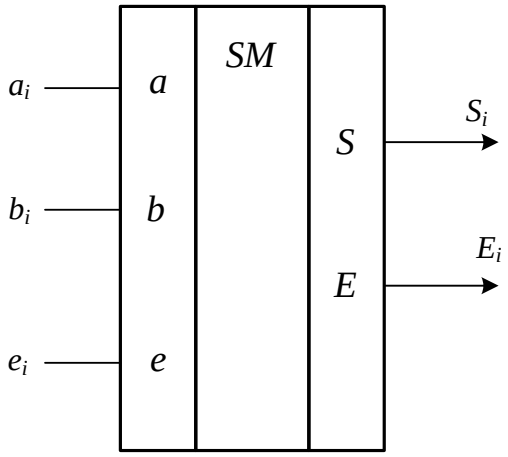


Рис.2.15. УГП однорозрядного суматора

Принцип функціонування однорозрядного повного суматора може бути описаний за допомогою таблиці істинності (табл.2.9).

Таблиця 2.19

Таблиця істинності однорозрядного

a_i	b_i	e_i	E_i	S_i	Десяткове значення суми (a_i+b_i+e_i)
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Відповідно до таблиці 2.9 досконала диз'юнктивна нормальна форма (ДДНФ) виглядає таким чином:

$$S = \bar{a}\,\bar{b}\,e \vee \bar{a}\,b\,\bar{e} \vee a\,\bar{b}\,\bar{e} \vee a\,b\,e;$$
$$E = a\,b\,\bar{e} \vee a\,\bar{b}\,e \vee \bar{a}\,b\,e \vee a\,b\,e. \qquad (2.1)$$

Двійковий суматор, реалізований за формулами (5.1, будемо називати канонічним суматором.

В отриманих вище логічних виразах кожний терм ДДНФ (кожна кон'юнкція в ДДНФ) відповідає рядкам таблиці істинності (табл. 5.3), в яких функції суми і вихідного переносу приймають одиничні значення (нагадаємо, що ДДНФ являє собою диз'юнкцію конституент одиниці).

Для реалізації логічної схеми суматора на базі ДДНФ доцільно використовувати готові інтегральні схеми дешифратора або мультиплексора. Наприклад, дешифратор є комбінаційною логічною схемою, яка формує на виходах конституенти одиниці. При цьому кількість адресних входів дешифратора відповідає кількості логічних змінних суматора, тобто необхідно використовувати дешифратор 3→8. Функціональна логічна схема канонічного суматора, яка реалізована за формулами (5.1), приведена на рис. 5.9. В цій схемі дешифратор у складі суматора виконує формування необхідних термів функцій S і E: $\bar{a}\,\bar{b}\,e$, $\bar{a}\,b\,\bar{e}$, $\bar{a}\,b\,e$, $a\,\bar{b}\,\bar{e}$, $a\,\bar{b}\,e$, $a\,b\,\bar{e}$ та $a\,b\,e$. Таким чином, на першому виході дешифратора реалізовано терм $\bar{a}\,\bar{b}\,e$, на виході 2 – терм $\bar{a}\,b\,\bar{e}$ і т.д.

Аналізуючи, таблицю істинності суматора, можна зробити висновок, що логічні функції суми і переносу характеризуються властивістю самоподвійності (самодвійковості). Ця властивість полягає в тому, що

інвертування значень аргументів функції призводить до інвертування значення самої функції, тобто:

$$S = F_S(a,b,e) = \overline{F_S(\bar{a},\bar{b},\bar{e})}; \qquad E = F_E(a,b,e) = \overline{F_E(\bar{a},\bar{b},\bar{e})}.$$

Самоподвійність функцій суматора широко застосовується при побудові багаторозрядних арифметичних пристроїв.

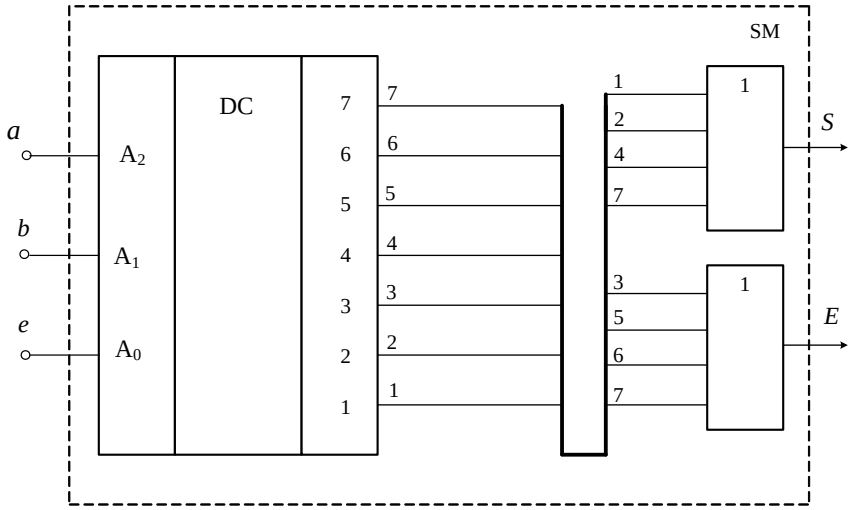


Рисунок 2.16 – Функціональна схема канонічного суматора на базі дешифратора

Схема для моделювання канонічного суматора на основі дешифратора в середовищі MicroCap приведена на рис. 2.17.

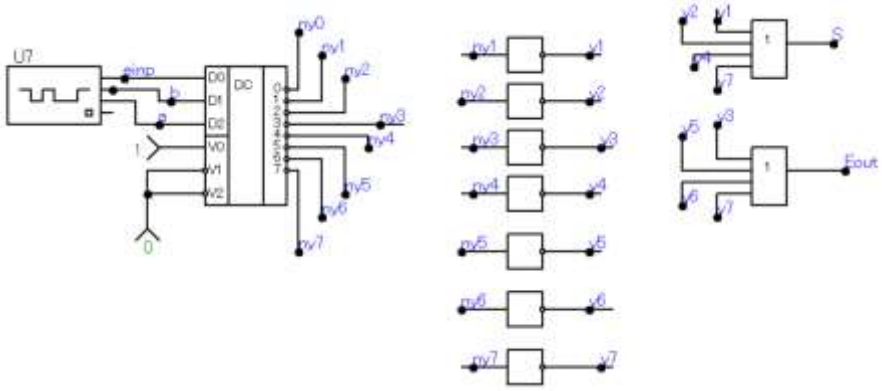


Рисунок 2.17 – Схема для моделювання канонічного суматора на базі дешифратора

Нижче приведена поведінкова модель суматора на базі дешифратора на мові *Verilog*:

```
module ProjectNULES (
 a,b,einp,res, S, Eout
);
        input a;
        input b;
```

```
        input einp;

        wire [2:0] bus;
        output [7:0] res;
        output S;
        output Eout;
        reg [7:0] res;

        assign bus[0] = einp;
        assign bus[1] = b;
        assign bus[2] = a;

        always @(bus)
        begin
            case (bus)
                3'b000 : res = 8'b11111110;
                3'b001 : res = 8'b11111101;
                3'b010 : res = 8'b11111011;
                3'b011 : res = 8'b11110111;
                3'b100 : res = 8'b11101111;
                3'b101 : res = 8'b11011111;
                3'b110 : res = 8'b10111111;
                default : res = 8'b01111111;
            endcase
        end

        assign S = !res[1] | !res[2] | !res[4] | !res[7] ;
        assign Eout = !res[3] | !res[5] | !res[6] | !res[7] ;
    endmodule
```

Результати моделювання схеми, приведеної на рис.5.10, в середовищі MicroCap приведена на рис. 2.18,а.

На часових діаграмах використовуються наступні скорочення: *d(a), d(b)* – часові діаграми доданків *a, b*; *d(einp), d(Eout)* – часові діаграми відповідно вхідного і вихідного переносів; *d(S)* – часова діаграма суми.

На рис.2.18,б приведені результати моделювання канонічного суматора на базі дешифратора в середовищі САПР QUARTUS II.

Результати моделювання на рис.5.11,а,б вказують на тотожну реакцію пристрою на вхідні сигнали в статичному режимі (в динамічному режимі результати відрізняються в зв'язку з різними затримками, які використовуються в моделях логічних елементів систем моделювання).

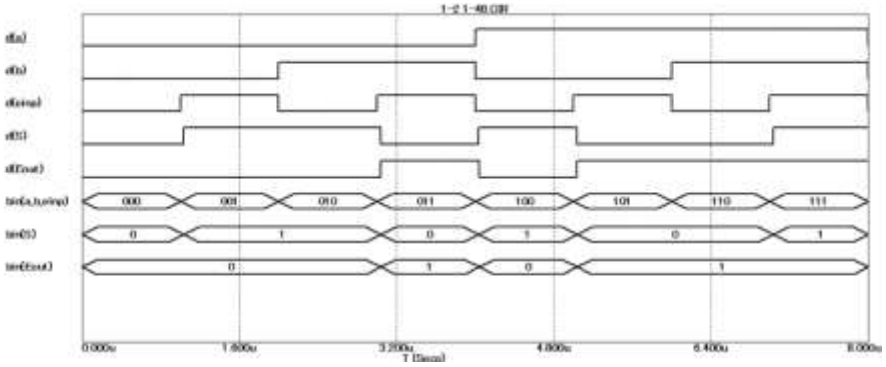


Рисунок 2.18,а – Результати моделювання канонічного суматора на базі дешифратора в середовищі MicroCap

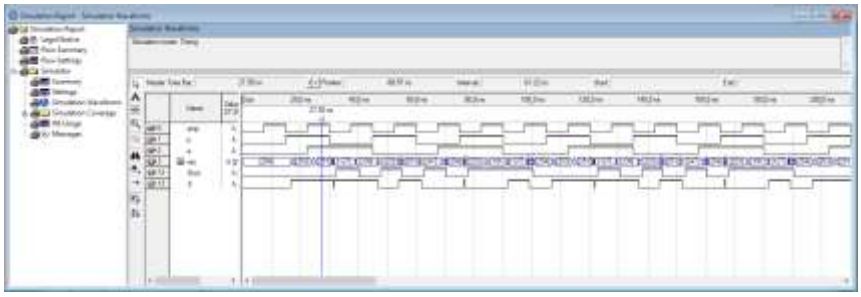


Рисунок 2.18,б – Результати моделювання канонічного суматора на базі дешифратора в середовищі САПР QUARTUS II

На рис. 2.19,а приведена схема для моделювання і часові діаграми функціонування канонічного суматора на основі дешифратора з інверсними виходами в середовищі MicroCap. В схемі логічні функції цього суматора реалізовані за формулами:

$$S = \bar{a}\bar{b}e \vee \bar{a}b\bar{e} \vee a\bar{b}\bar{e} \vee abe = \overline{\bar{a}\bar{b}e}\ \overline{\bar{a}b\bar{e}}\ \overline{a\bar{b}\bar{e}}\ \overline{abe};$$

$$E = ab\bar{e} \vee \bar{a}be \vee a\bar{b}e \vee abe = \overline{ab\bar{e}}\ \overline{\bar{a}be}\ \overline{a\bar{b}e}\ \overline{abe}.$$

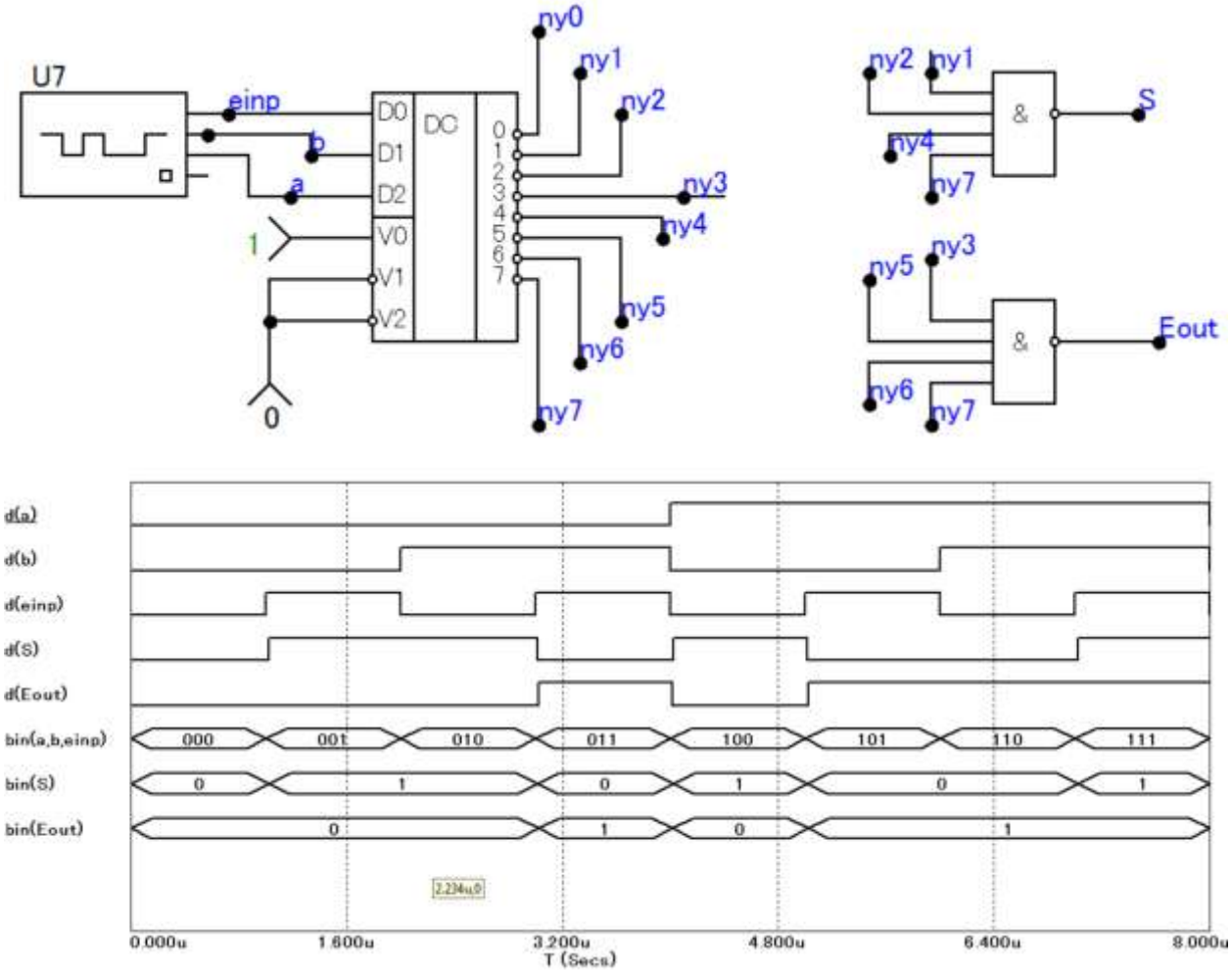


Рисунок 2.19,а – Результати моделювання канонічного суматора на основі дешифратора з інверсними виходами в середовищі MicroCap

Поведінкова модель суматора на базі дешифратора з інверсними виходами на мові *Verilog* приведена нижче:

```
module ProjectNULES (
 a,b,einp,res, S, Eout
);
        input a;
        input b;
        input einp;
        wire [2:0] bus;
        output [7:0] res;

        output S;
        output Eout;
        reg [7:0] res;
```

```
        assign bus[0] = einp;
        assign bus[1] = b;
        assign bus[2] = a;

        always @(bus)
        begin
            case (bus)
                3'b000 : res = 8'b11111110;
                3'b001 : res = 8'b11111101;
                3'b010 : res = 8'b11111011;
                3'b011 : res = 8'b11110111;
                3'b100 : res = 8'b11101111;
                3'b101 : res = 8'b11011111;
                3'b110 : res = 8'b10111111;
                default : res = 8'b01111111;
            endcase
        end

        assign S = !(res[1] & res[2] & res[4] & res[7]) ;
        assign Eout = !(res[3] & res[5] & res[6] & res[7]) ;
    endmodule
```

Часові діаграми функціонування пристрою, логічна схема якого приведена на рис.2.20,а), в середовищі САПР QUARTUS II приведена на рис.2.20,б).

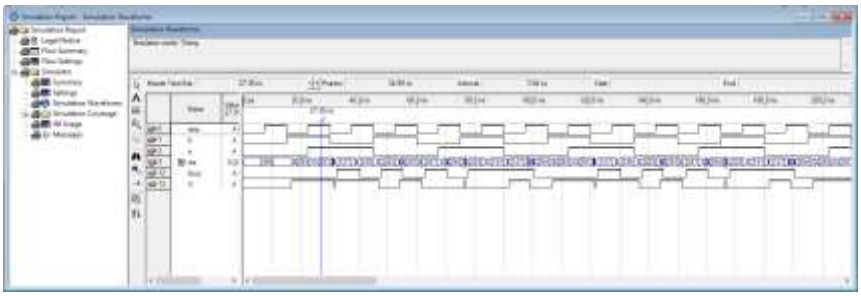


Рисунок 2.20,б – Часові діаграми канонічного суматора на основі дешифратора з інверсними виходами в середовищі САПР QUARTUS II

Результати моделювання канонічного суматора, побудованого в базисі І-АБО-НІ, в середовищах MicroCap і САПР QUARTUS II приведені на рис. 2.21а,б.

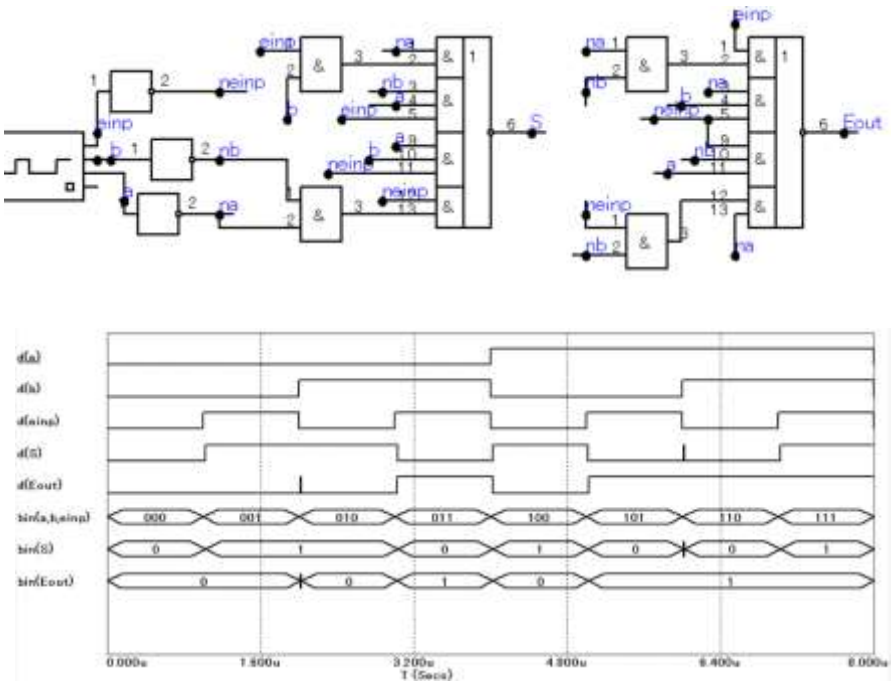


Рисунок 2.21,а – Часові діаграми і схема канонічного суматора в базисі І-АБО-НІ в середовищі MicroCap

Поведінкова модель такого суматора приведена нижче:

```
module ProjectNULES (a, b, einp, S, Eout);
        wire neinp;
        wire nb;
        wire na;

        input a;
        input b;
        input einp;

        output S;
        output Eout;

        assign neinp = ~einp;
        assign nb = ~b;
        assign na = ~a;

        assign S = !((na & einp & b) | (nb & a & einp) | (a & b & neinp) | (na & nb & neinp));
        assign Eout = !((na && nb && einp) || (na && b && neinp) || (a && nb && neinp) || (na && nb && neinp));

endmodule
```

Результати моделювання цього суматора в середовищі QUARTUS II приведені на рис.2.22,б.

Рисунок 2.22,б – Часові діаграми канонічного суматора в базисі І-АБО-НІ в середовищі QUARTUS II

Для спрощення логічної схеми, тобто для скорочення апаратних витрат під час побудови суматора, необхідно застосувати процедуру мінімізації логічних функцій з використанням карт Карно. Таблиця істинності суматора може бути представлена за допомогою карт Карно (рис. 2.23).

a \ *be*	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$S = f_S(a,b,e)$

a \ *be*	00	01	11	10
0	0	0	1	1
1	0	1	1	1

$E = f_E(a,b,e)$

Рисунок 2.23 – Карти Карно для мінімізації логічних функцій двійкового суматора

З карт Карно витікає, що функція суми не мінімізується і може бути представлена у вигляді ДДНФ (5.1). В результаті мінімізації функції переносу отримаємо формулу:

$$E = ab + ae + be;$$

Суматор, побудований за цим принципом, будемо називати мінімальним суматором. При використанні базису І-АБО-НІ (з урахуванням властивості самоподвійності функцій S і E) для реалізації схеми мінімального суматора можна записати:

$$S = \overline{a\bar{b}\bar{e} + \bar{a}b\bar{e} + \bar{a}\bar{b}e + \bar{a}\bar{b}\bar{e}};\quad E = \overline{\bar{a}\bar{b} + \bar{a}\bar{e} + \bar{b}\bar{e}} \qquad (2.2)$$

Логічна схема мінімального суматора в базисі І-АБО-НІ приведена на рис. 2.24.

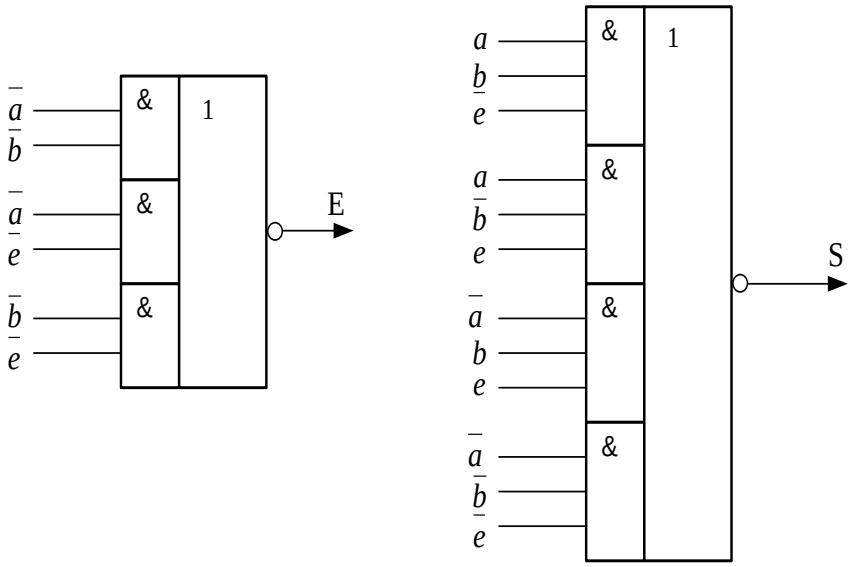


Рисунок 2.24 – Логічна схема мінімального суматора в базисі І-АБО-НІ

Логічна схема такого суматора в середовищі MicroCap (рис.2.25) та результати моделювання за допомогою MicroCap і САПР QUARTUS II приведені відповідно на рис.2.26,а,б.

Логічну схему мінімального суматора може бути реалізована за допомогою двох інших форм в елементному базисі «Виключне АБО» («XOR», («Додавання за модулем два») та в базисі І-АБО-НІ (відповідно рис.2.27 і рис.5.28.

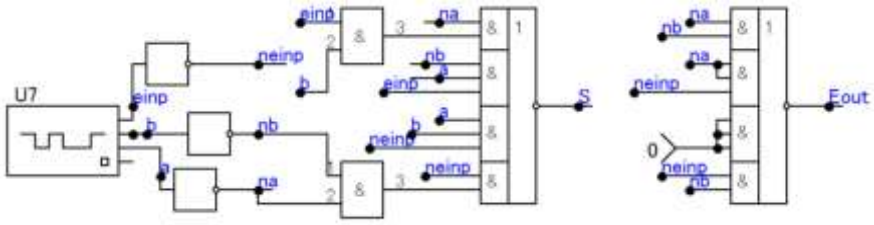


Рисунок 5.25 – Схема моделювання мінімального суматора

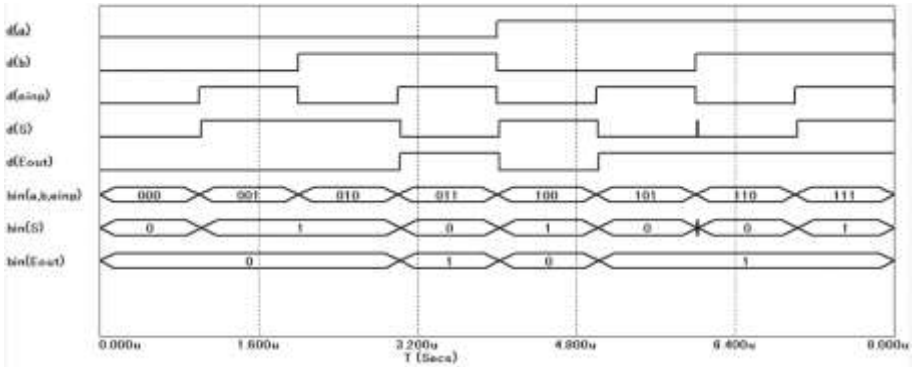


Рисунок 5.26,а – Часові діаграми мінімального суматора в середовищі MicroCap

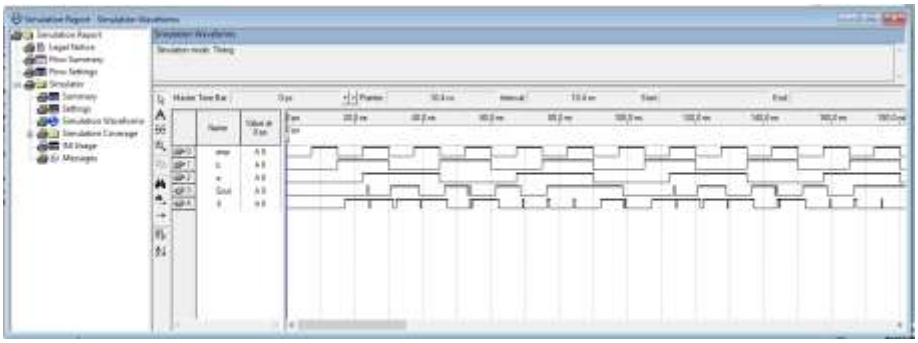


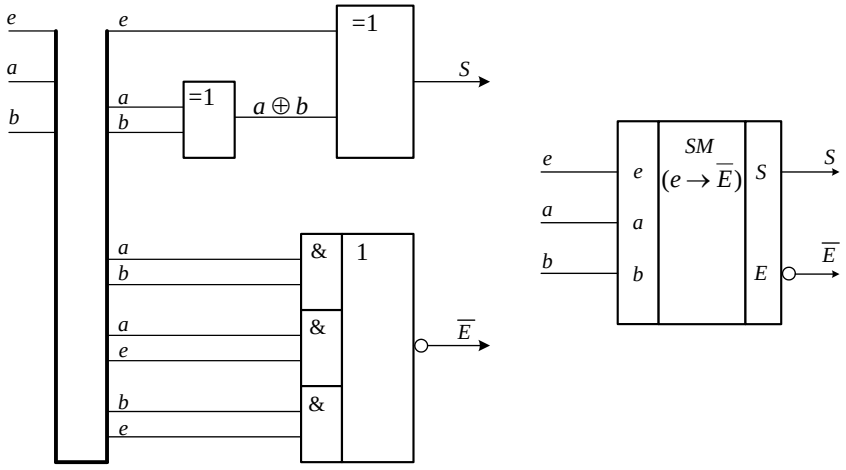
Рисунок 5.26,б – Часові діаграми мінімального суматора в середовищі QUARTUS II

Поведінкова модель мінімального суматора приведена нижчу:

```
module ProjectNULES (a, b, einp, S, Eout);
    wire neinp;
    wire nb;
    wire na;

    input a;
    input b;
    input einp;

    output S;
    output Eout;

    assign neinp = ~einp;
    assign nb = ~b;
    assign na = ~a;

    assign S = !((na & einp & b) | (nb & a & einp) | (a & b & neinp) | (na & nb & neinp));
    assign Eout = !((na && nb && einp) || (na && b && neinp) || (a && nb && neinp) || (na && nb && neinp));

endmodule
```

При цьому функція вихідного переносу E мінімального суматора (рис.5.27) реалізована з використанням тракту $(e \to \overline{E})$, тобто на входи суматора подається пряме значення вхідного переносу, а на виході формується інверсне значення вихідного переносу. Така структурна організація використовується під час проектування багаторозрядних суматорів. Для побудови логічної схеми формування суми цього пристрою використовується перетворення:

$$S = \bar{a}\bar{b}e + \bar{a}b\bar{e} + a\bar{b}\bar{e} + abe =$$
$$= e(\bar{a}\bar{b} + ab) + \bar{e}(\bar{a}b + a\bar{b}) =$$
$$= e(\overline{a \oplus b}) + \bar{e}(a \oplus b) = e \oplus (a \oplus b);$$
$$\bar{E} = \overline{ab + ae + be}.$$



а) *б)*

Рисунок 5.27 – Логічна схема (а) і УГП (б) мінімального суматора з трактом ($e \to \bar{E}$)

Схема модельного експерименту в середовищі MicroCap приведена на рис.5.28.

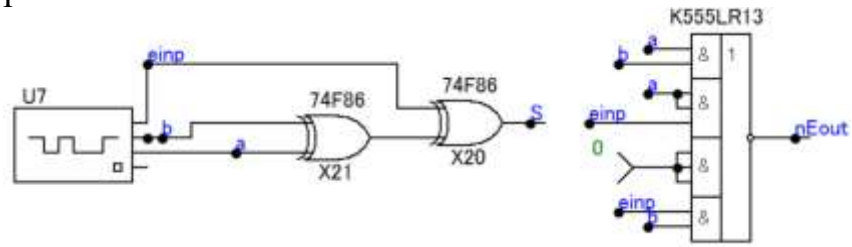


Рисунок 5.28 – Схема моделювання мінімального суматора з трактом ($e \to \bar{E}$)

Поведінкова модель на мові Verilog, яка реалізує функціонування мінімального суматора з трактом ($e \to \bar{E}$, приведена нижче:

```
module ProjectNULES (
  input a,
  input b,
  input einp,

  output   nEout,
  output   S
);
        assign S  = (a ^ b) ^ einp;
        assign nEout = !((a & b) | (a & einp) | (einp & b));
endmodule
```

Часові діаграми мінімального суматора з трактом $e \rightarrow \overline{E}$ в середовищі MicroCap і QUARTUS II приведені відповідно на рис.2.30,а і рис.2.30,б.

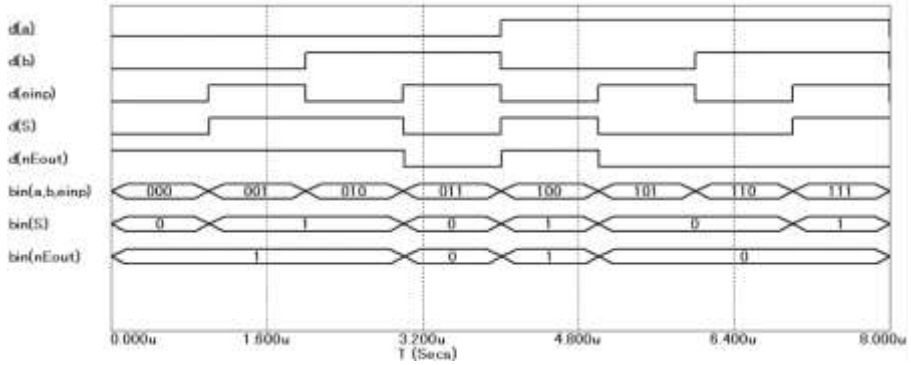


Рисунок 2.30,а – Часові діаграми мінімального суматора з трактом ($e \rightarrow \overline{E}$) в середовищі MicroCap

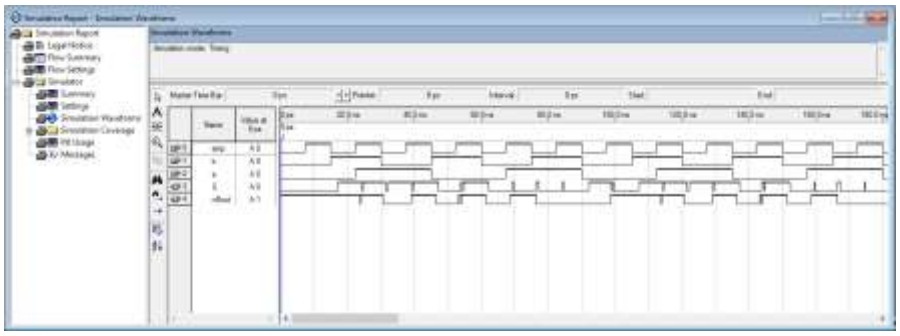


Рисунок 2.30,б – Часові діаграми суматора з трактом ($e \rightarrow \overline{E}$) в середовищі QUARTUS II

Логічна схема на рис. 2.31 реалізує мінімальний суматор з трактом розповсюдження переносу $(\overline{e} \rightarrow E)$ та побудована за допомогою перетворень:

$$S = (a \oplus b) \oplus e = (\overline{a}b + a\overline{b}) \oplus e =$$

$$= \overline{(\overline{a}b + a\overline{b})}e + (\overline{a}b + a\overline{b})\overline{e} =$$

$$= (\overline{a}\overline{b} + ab)e + \overline{(\overline{a}\overline{b} + ab)}\overline{e} = (\overline{a \oplus b}) \oplus (\overline{e}).$$

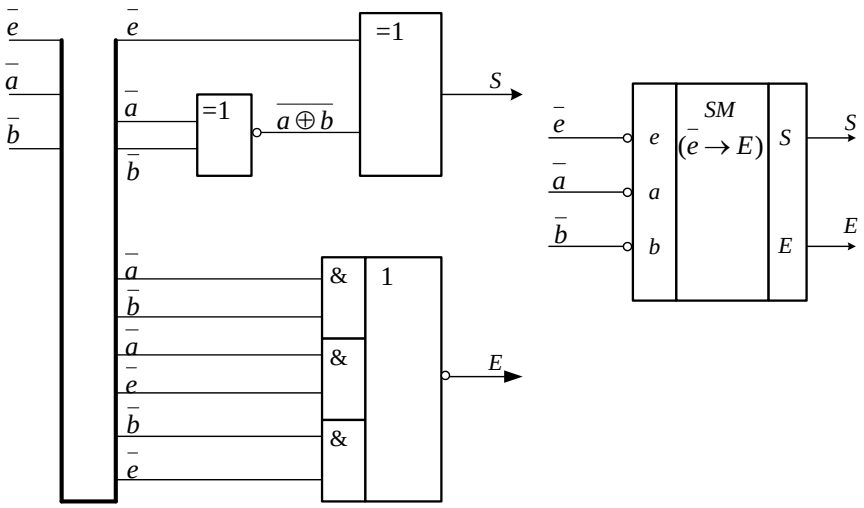
Таким чином, отримаємо

$$S = (\overline{a \oplus b}) \oplus (\overline{e}); \; E = \overline{\overline{a}\overline{b} + \overline{a}\overline{e} + \overline{b}\overline{e}}.$$

Поведінкова модель на мові Verilog реалізує функціонування мінімального суматора з трактом $(\overline{e} \rightarrow E)$ та приведена нижче:

```
module ProjectNULES (
  input a,
  input b,
  input einp,

  output  nEout,
  output  S
);
```

```
        wire na;
        wire nb;
        wire neinp;

        assign na = !a;
        assign nb = !b;
        assign neinp = !einp;
        assign S  = (na ^ nb) ^ neinp;
        assign nEout = !((na & nb) | (na & neinp) | (neinp & nb));
endmodule
```



а) *б)*

Рисунок 2.31 – Логічна схема (а) і УГП (б) мінімального суматора з трактом $(\bar{e} \to E)$

Результат моделювання у вигляді часових діаграм мінімального суматора з трактом $(\bar{e} \to E)$ в середовищі MicroCap і QUARTUS II приведені відповідно на рис.2.32,а і рис.2.32,б.

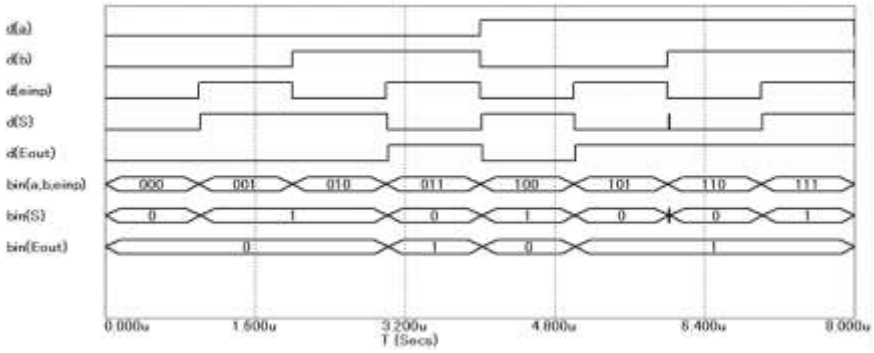


Рисунок 2.32,а – Часові діаграми мінімального суматора з трактом $(\bar{e} \to E)$ *в середовищі MicroCap*

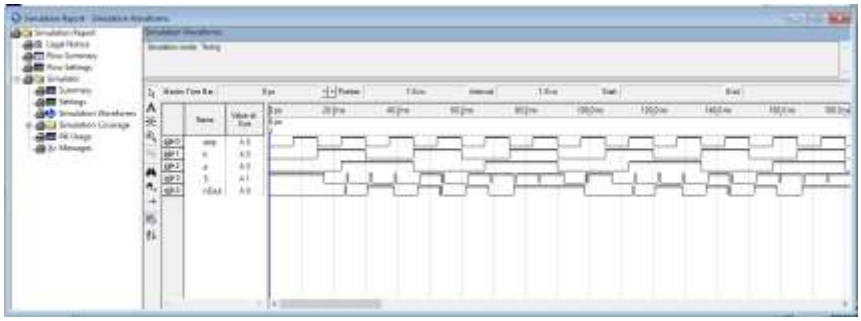


Рисунок 2.32,б – Часові діаграми мінімального суматора з трактом $(\bar{e} \to E)$ *в середовищі QUARTUS II*

Аналізуючи логічні схеми суматора з різними способами реалізації трактів розповсюдження переносу, можна відзначити, що ці суматори є еквівалентними з точки зору апаратурних витрат. Для реалізації логічної схеми формування суми використовується властивість функції додавання за модулем два: $a \oplus b = \bar{a} \oplus \bar{b}$.

Приклад виконання лабораторної роботи на стенді DE0 наведено у Додатку В.

Питання для самоперевірки

1. Назвіть етапи проектування комбінаційної схеми із застосуванням ПЛІС?
2. Як проводиться підключення комбінаційної схеми до зовнішніх контактів стенду?
3. Яким чином визначити реакцію пристрою на вхідні сигнали?
4. Як задавати вхідні сигнали на вхід пристрою?
5. Для чого використовується блок індикації стенду?
6. Для чого використовується задавальна частина стенду?
7. Як виконати моделювання логічної схеми в середовищі системи MicroCap?
8. Як виконати моделювання логічної схеми в середовищі САПР QUARTUS II?

Розділ 3. Система автоматизованого проектування QUARTUS II

На сьогодні найпоширенішими стали кілька потужних САПР для ПЛІС – MAX + PLUS II, Quartus II, Xilinx ISE. У даній лабораторній роботі розглядається основні особливості програмного пакету Quartus II. (фірма Altera).

Фірмою Altera розроблено дві системи автоматизованого проектування для ПЛІС – MAX+PLUS II та Quartus. Назва системи MAX+PLUS II є абревіатурою від Multiple Array Matri Programmable Logic User System (Користувальницька система програмування логіки впорядкованих структур). Система MAX+PLUS II забезпечує багатоплатформне архітектурно незалежне середовище створення дизайну, що легко пристосовується для конкретних вимог користувача. Система MAX+PLUS II має засоби зручного введення дизайну, швидкого прогону й безпосереднього програмування пристроїв.

САПР Quartus II, див. рис. 3.1 надає комплекс засобів для системного налагодження проектів, які дозволяють як замінити зовнішні прилади вимірювання та аналізу, так і полегшити підключення цих зовнішніх приладів.

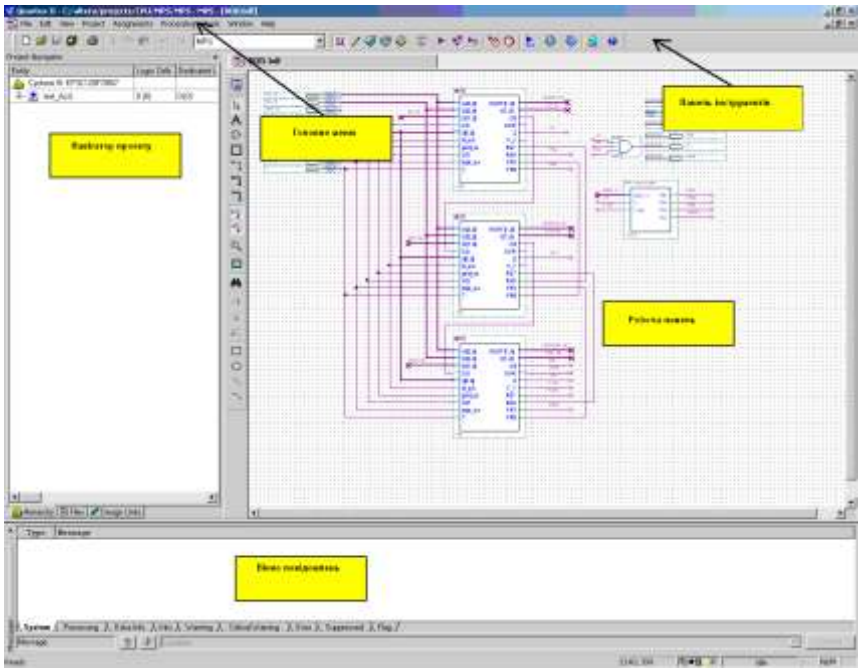


Рис. 3.1. Загальний вигляд САПР Quartus II

До засобів системного налагодження САПР Quartus II відносяться:

• редактор налагоджувальних виводів (SignalProbe Pins);

• редактор інтерфейсу для зовнішнього логічного аналізатора (Logic Analyzer Interface Editor);

• редактор вмісту пам'яті в системному середовищі (In-SystemMemory Content Editor);

• вбудований логічний аналізатор Signal Tap II (Signal Tap II Logic Analyzer).

Пакет Quartus II надає засоби для підключення сигналів проекту, доступних в Node Finder після синтезу, до виводів надвеликої інтегральної схеми програмованої логіки (НВІС ПЛ), які вільні від основних функцій проекту і призначені користувачем для контролю функціонування пристроїв в системі. До таких засобів відносяться налагодження виводів (SignalProbe Pins) і інтерфейс для підключення зовнішнього логічного аналізатора (LAI).LogicLock - це нова блокова методологія проектування, доступна лише в програмному забезпеченні Quartus II. Quartus II спільно з LogicLock - єдине програмне забезпечення для розробки пристроїв на основі програмованої логіки, яке включає в себе блочну методологіюпроектування як стандартну функцію. Це допомагає збільшити ефективність роботи розробок, зменшити час проектування та верифікації. Розробники можуть об'єднувати готові модулі в проект верхнього рівня, зберігаючи продуктивність кожного модуля в процесі об'єднання.

NativeLink - дозволяє здійснювати зв'язок між засобом розробки Quartus II і програмним забезпеченням інших виробників. Пряме перетворення скорочує час компіляції та звільняє від використання додаткових бібліотек трансляцій перетворень, які можуть обмежити продуктивність, досягнуту засобами проектування сторонніх виробників. NativeLink дозволяє розробникам використовувати Quartus II для розміщення елементів, а засоби проектування інших виробників - для оптимізації стратегій синтезу.

PowerFit. Технологія розміщення елементів і трасування з'єднань PowerFit в програмному забезпеченні Quartus II використовує тимчасові параметри, задані розробником, для оптимального складання схеми і розміщення логічних елементів. Критичні до часових параметрів з'єднання оптимізуються в першу чергу, для зменшення затримок і досягнення максимальної продуктивності (fMAX). Подальше поліпшення параметра fMAX досягається використанням новітньої архітектури, такий як в сімействі пристроїв Stratix. Ця передова технологія розміщення елементів і трасування з'єднань допомагає користувачам програмного забезпечення Quartus II досягти максимальної продуктивності, і володіє малим часом компіляції проекту серед подібних засобів розробки.

Верифікація. Перевірка або верифікація проекту може виявитися найтривалішою стадією в процесі розробки високопродуктивних систем на кристалі (SOPC). Однак, використовуючи Quartus II, можна скоротити час верифікації, оскільки це програмне забезпечення має набір власних засобів верифікації.

Аналіз. Altera розробила два методи SignalProbe і SignalTap, для того, щоб допомогти розробникам проаналізувати стан внутрішніх точок і входів / виходів пристрою.

SignalProbe. Доступна в останніх версіях програмного забезпечення Quartus II технологія апаратного налагодження SignalProbe дозволяє користувачам послідовно з'єднувати внутрішні точки пристрою з вільними зарезервованими виводами для аналізу за допомогою осцилографа або логічного аналізатора.

SignalTap. Для багатьох розробників, які використовують корпуса BGA з великою кількістю входів / виходів, верифікація системного рівня займає дуже багато часу і іноді сильно утруднена.

SignalTap дозволяє розробникам зібрати дані з будь-яких внутрішніх точок і входів / виходів пристрою в режимі реального часу при роботі системи. Quartus II вставляє в проект мегафункцію,

що містить логічний аналізатор. Дані збираються і зберігаються в блоках вбудованої пам'яті пристрою і направляються в програмне забезпечення Quartus II через завантажувальний кабель.

Розробники також можуть подати внутрішні сигнали на виводи пристрою для подальшого моніторингу. Логічний аналізатор SignalTap дозволяє істотно знизити час верифікації, що дозволяє в більш короткі терміни випускати нові продукти.

PowerGauge. Програмне забезпечення Quartus II включає технологію PowerGauge – перший інтегрований засіб аналізу енергоживлення, який використовує файли, створені в процесі моделювання для того, щоб скласти оцінку використання енергії із заданими параметрами пристрою.

САПР Quartus II має простий і дружній інтерфейс користувача (див. рис. 2.1). Основними елементами інтерфейсу є:

1) робоча панель - використовується для відображення і редагування схемних і текстових описів проекту, тимчасових діаграм, виведення звітів симуляції і т.п.;

2) панель інструментів, яка містить основні інструменти САПР по роботі з файлами і проектом (компілятор, симулятор і т.д.);

3) головне меню;

4) вікно виведення повідомлень, яке призначено для виведення повідомлень компілятора, симулятора і іншої подібної інформації;

5) навігатор проекту - містить три вкладки:

- Hierarchy (Ієрархія проекту);
- Files (список файлів проекту);
- Design Units (список модулів, використаних в проекті).

Після створення файлу проекту стає активною панель інструментів розташована зліва від робочої області файлу, див. рис. 3.2. На панелі інструментів знаходяться засоби для створення графічної схеми проекту (схемного опису).

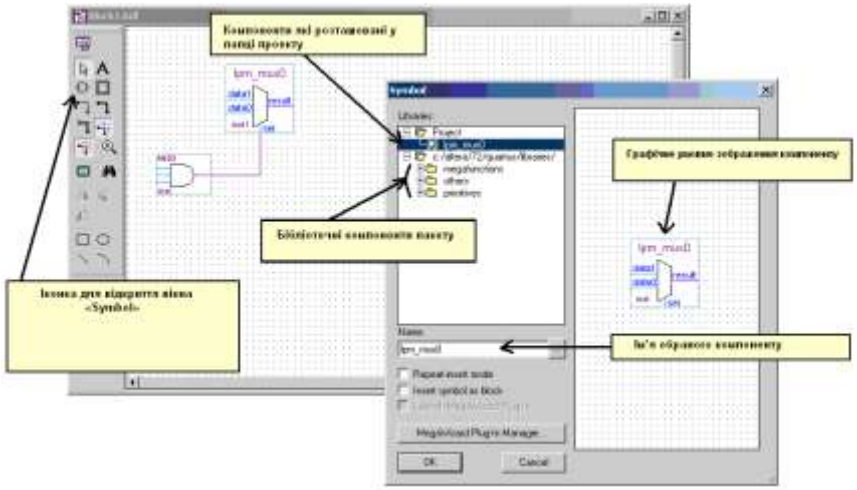


Рис. 3.2. Редактор схем і вікно вибору символу

Інструмент **Symbol Tool** зображений на рис. 3.2 у верхньому лівому куті панелі інструментів, натиснемо на його іконі лівим клацанням миші, в результаті нам відкриється вікно **Symbol** (див. рис. 3.3). В лівому верхньому кутку представлений список стандартних бібліотек основних типів елементів і модулів САПР Quartus II.

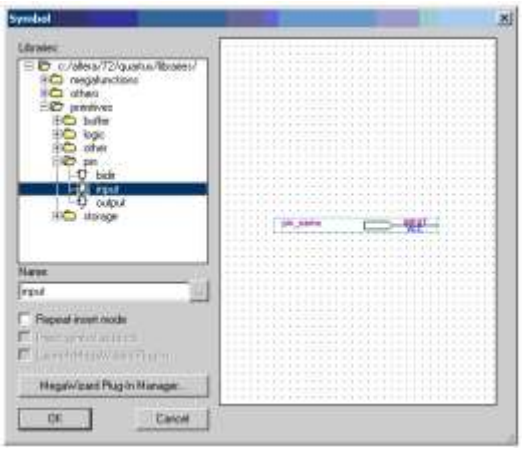


Рис. 3.3. Вікно вибору символу

Після розміщення компонентів схеми, необхідно провести з'єднання входів і виходів компонентів, помістивши курсор миші на один з входів елемента, затиснув ліву кнопку миші з'єднати його з вихідним кінцем елемента входу.

Для з'єднання компонентів на панелі інструментів є 3 інструменти, див. рис. 3.4.

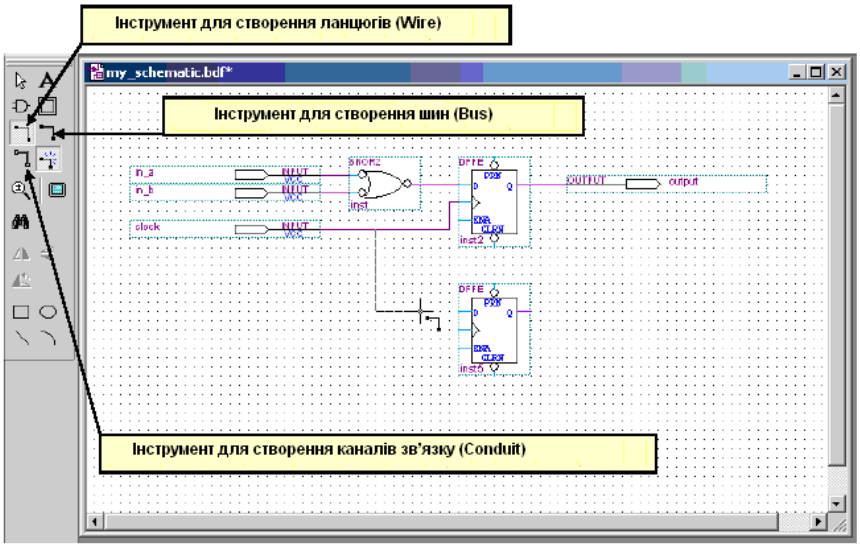


Рис. 3.4. Інструменти для з'єднання компонентів

Повідомлення компілятора (помилки і попередження) відображаються у вікні виведення повідомлень в нижній частині інтерфейсу САПР, див. рис. 3.5. У разі відсутності помилок з'єднань компонентів (немає непідключених виходів і т.п.) і не порушення інших правил проектування, компіляція буде завершена успішно.

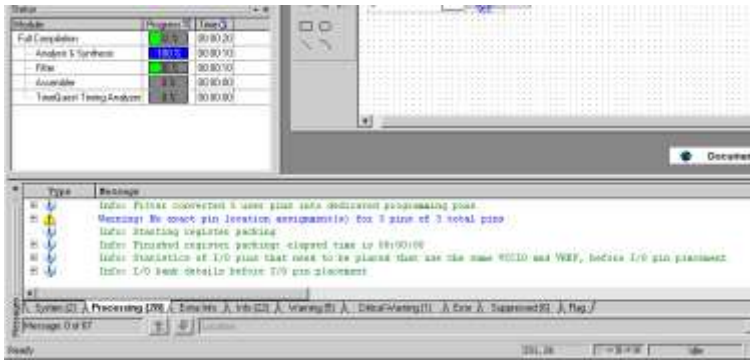


Рис. 3.5. Вікно виведення повідомлень

Інакше причину неуспішної компіляції можна переглянути у вікні повідомлень – меню **View > Utility Windows > Messages**. Потрібно виправити в схемі помилки і повторити компіляцію. Результати компіляції можна переглянути у вікні звіту компілятора – меню **Processing > Compilation Report**. З погляду на задачу дослідження продуктивності пристрою, що наприклад, розробляється, найцікавішими є результати тимчасового аналізатора. В гілці **Timing Analyzer > Summary** можна переглянути максимальний час розповсюдження сигналу t_{PD}, див. рис. 2.6.

Рис. 3.6. Вікно гілки Timing Analyzer > Summary

В даному випадку максимальний час за який сигнал проходить, на приклад, від входу **inv** до виходу **o[21] складає** - 15.214 нс. Це дозволяє працювати пристрою на частоті не більше 65,7 Мгц. Для перегляду аналогічного параметра для всіх пар входів і виходів потрібно переглянути гілку **Timing Analyzer > tpd** звіту, див. рис. 3.7.

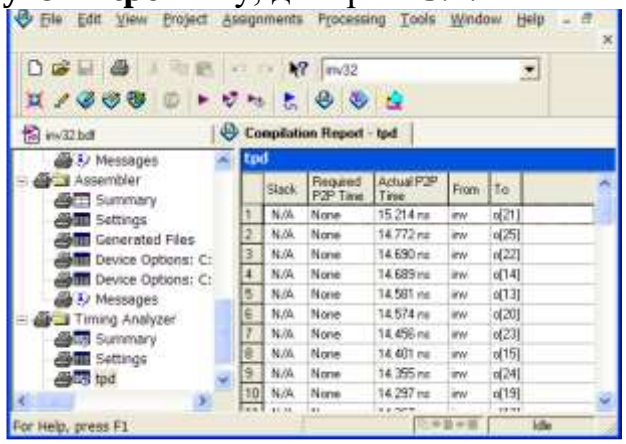


Рис. 3.7. Вікно гілки Timing Analyzer >tpd

Для дослідження апаратурної складності схеми пристрою – кількості логічних елементів, блоків пам'яті, зовнішніх виходів потрібно проглянути гілку Fitter > Summary звіту, див. рис. 3.8.

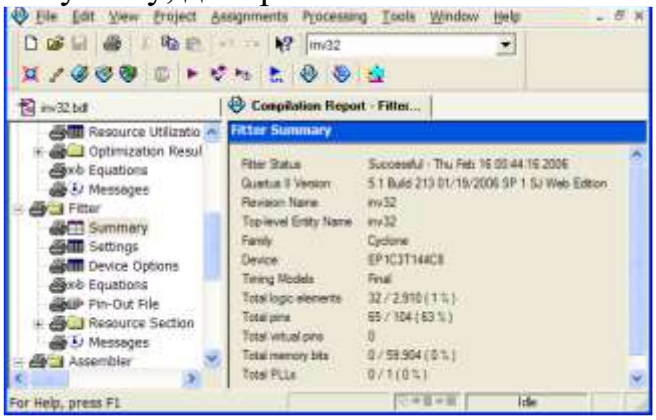


Рис. 3.8. Вікно гілки Fitter > Summary

Для тестування пристрою можна побудувати тимчасові діаграми (waveform) його роботи. Щоб відкрити вікно редактора тимчасових діаграм,

потрібно в діалоговій панелі **New** на закладці **Other Files** вибрати пункт **Vector Waveform Editor**, див. рис. 3.9.

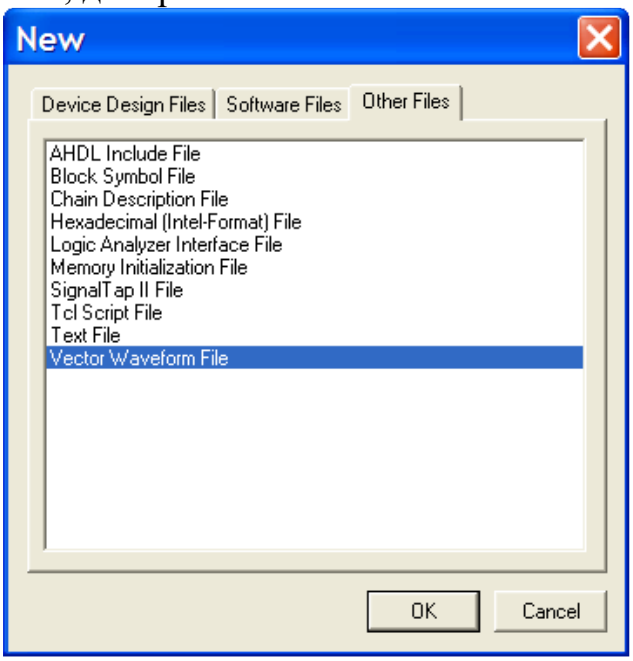


Рис. 3.9. Вікно New

Для запуску процесу симуляції необхідно обрати пункт меню **Processing > Start Simulation**. Результуючі тимчасові діаграми відображені у гілці **Simulator > Simulation Waveforms**, див. рис. 3.10.

Для зміни тривалості симуляції і кроку сітки використовуються, відповідно, пункти меню **Edit > End Time...** і **Edit > Grid Size...., див. рис. 3.10.**

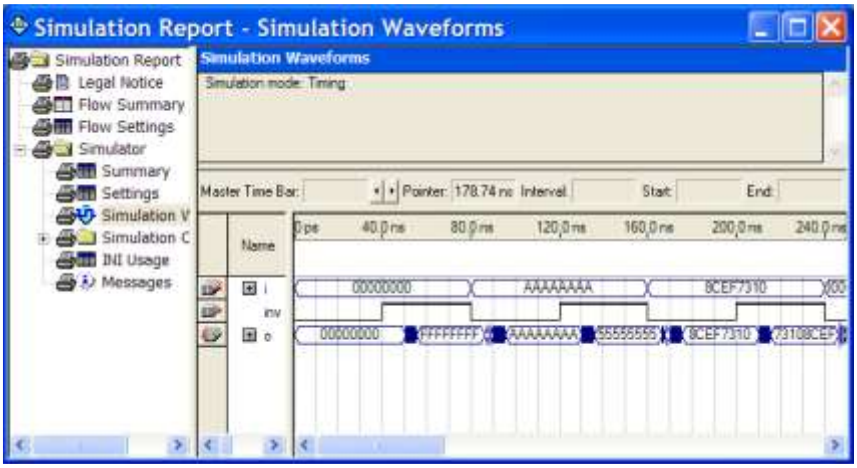


Рис. 3.10. Вікно Simulation Waveforms

Питання для самоперевірки

1. Призначення системи автоматизованого проектування Quartus II.
2. Етапи створення проекту в системі Quartus II.
3. Робота з графічним редактором Quartus II.

Розділ 4. Дослідження логічних елементів на стенді DE0.

Графічне введення схеми в САПР QUARTUS II.

Робота в середовищі Quartus II починається з дій, які називають створенням проекту. Перш за все, необхідно створити окремий каталог для зберігання файлів проекту. Ім'я каталогу необхідно вводити латинськими буквами. Потім слід запустити програму Quartus II. Відкривши пакет, Quartus II вибираємо з меню **File** пункт **New Project Wizard** - майстер створення нових проектів. У вікні, що відкрилося, натискуємо кнопку Next і потрапляємо у вікно для завдання поточної директорії проекту. Заповнюємо три рядки як, показано на рис. 4.1. В даному випадку поточний проект буде названий LAB1. Проект буде створений у папці LAB на робочому столі. Натискуємо кнопку **Finish** і підтверджуємо створення проекту.

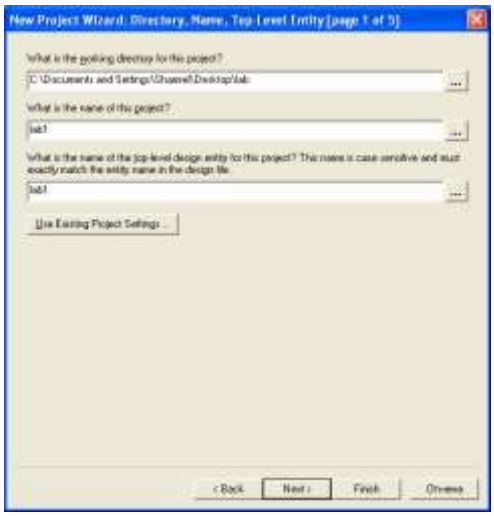


Рис. 4.1. Створення нового проекту у Quartus II

Графічний редактор призначений для введення принципової схеми цифрового пристрою. Для створення файлу, який міститиме схему пристрою (після створення проекту) слід виконати команду **New** меню **File**. У діалоговим вікні (рис. 4.2) на вкладці **Devise Design File** слід вибрати тип файлу **Block Diagram/Schematic File** і натиснути **OK**.

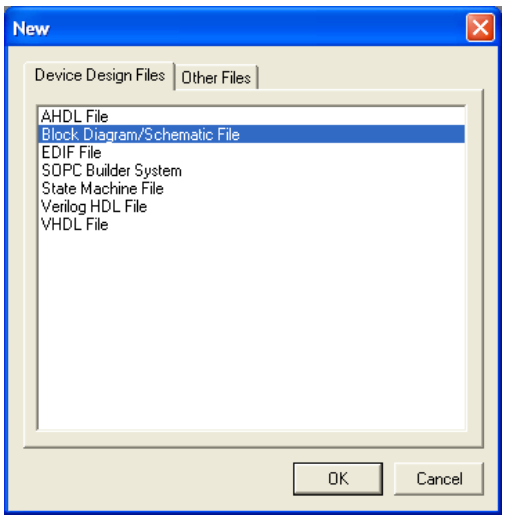


Рис. 4.2. Тип файлу проекту

В результаті відкриється вікно графічного редактора з файлом Block1.bdf, в якому створюється схема, див. рис. 4.3.

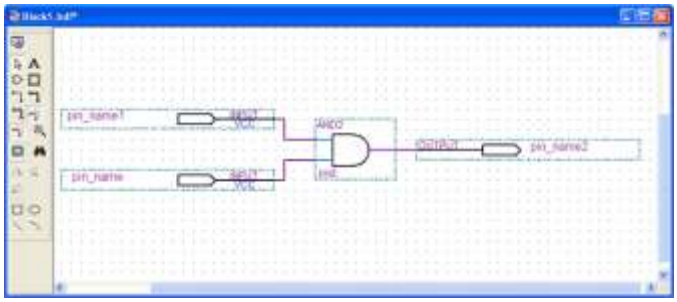


Рис. 4.3. Схема пристрою

Після створення файлу проекту стає активною панель інструментів, розташована зліва від робочої області вікна. Для введення елемента схеми слід «клацнути» по **Symbol Tool** (див. рис. 4.4). В результаті відкриється вікно з бібліотеками елементів. Обравши компонент натискаємо **Ok**.

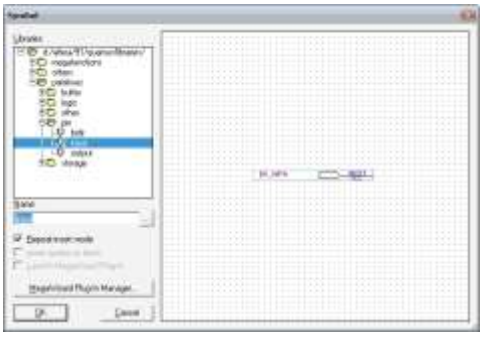


Рис. 4.4. Вікно Symbol

Наприклад, для введення логічного елемента «І» слід вибрати бібліотеку **primitives/logic**. Після розміщення компонентів на схемі слід розмістити вхідні (input) і вихідні (output) контакти, які знаходяться в теці

primitives/pin. З'єднання компонентів проводиться наступним чином: перемістити курсор в одну з двох точок схеми, які потрібно з'єднати, натискувати ліву кнопку миші і, не відпускаючи її, перемішати курсор до другої з крапок, що з'єднується. Далі слід перейменувати вхідні і вихідні контакти. Для цього двічі клацаємо лівою кнопкою миші на контакті і редагуємо його. В результаті одержимо наступну схему, див. рис. 4.5.

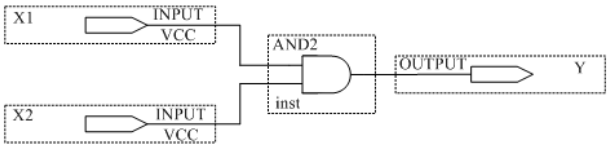


Рис. 4.5. Схема пристрою після редагування

Перед здійсненням компіляції проведемо вибір типу кристала. Для цього вибираємо пункт **Device**. з меню **Assignments**. У вікні **Settings**, див. рис. 3.6, в рядку **Family** виберемо сімейство **Cyclon III**, а у вікні **Available Devices** виберемо конкретний пристрій (див. рис. 4.6) **EP3C16F484C6**. Натиснемо кнопку **OK** та підтвердимо вибір кристала. Для запуску процесу компіляції виберемо пункт **Start Compilation** з меню **Processing**. Підтвердимо збереження поточного файлу і чекаємо закінчення процесу компіляції. Після закінчення компіляції з'являється вікно з повідомленням про результати компіляції і кількість помилок і попереджень, див. рис. 4.7.

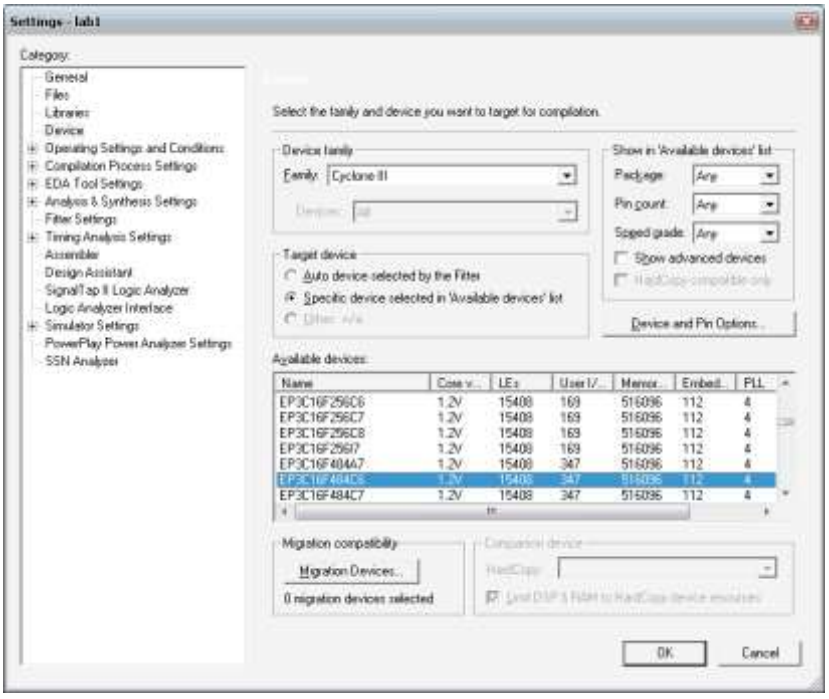


Рис. 4.6. Підключення до проекту кристалу

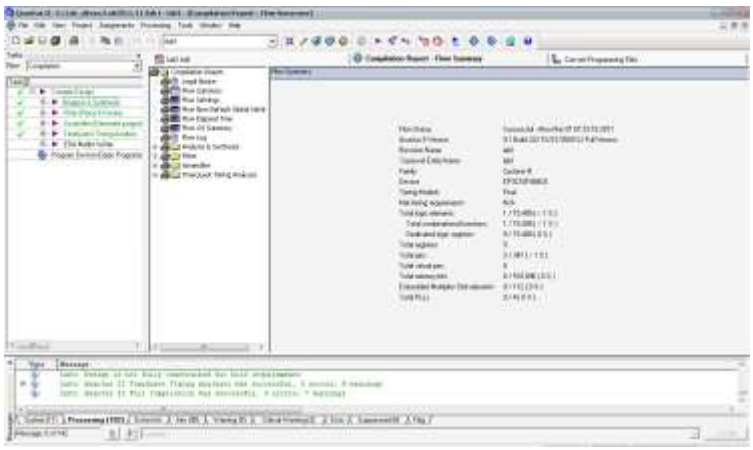


Рис. 4.7. Результат компіляції проекту

Після того, як проведена компіляція проекту слід підключити вхідні і вихідні контакти до зовнішніх контактів ПЛІС. Для цього слід вибрати **Assignments/Pins**. В результаті з'явиться вікно **Pin Planer**, див. рис. 4.8.

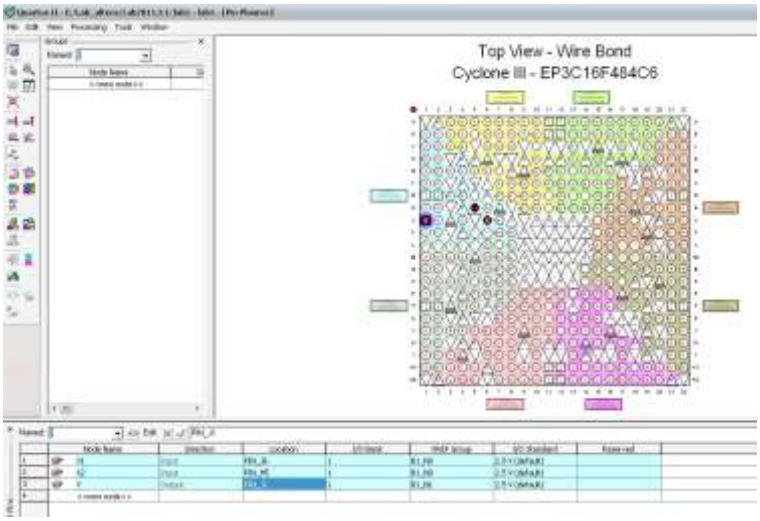


Рис. 4.8. Вікно Pin Planer

В колонці **Node Name** розташовуються імена входів і виходів схеми. Для їх підключення до контактів ПЛІС слідує двічі «клацнути» по відповідному елементу в колонці Location і вибрати контакт, до якого потрібно підключити вхід (вихід) електричної схеми. Можна також просто перемістити відповідне ім'я (наприклад, X1) на контакт, зображений на рис. ПЛІС. Після підключення всіх контактів слід ще раз скомпілювати проект. В результаті необхідна схема матиме наступний вигляд, див. рис. 4.9.

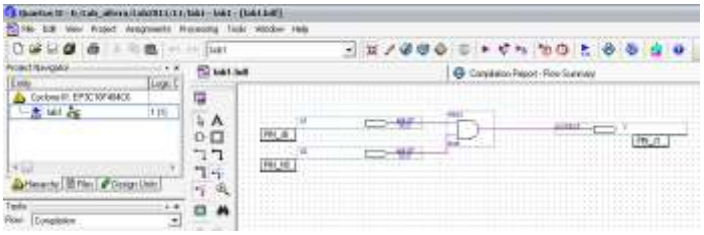


Рис. 4.9. Вигляд схеми після компіляції проекту

Підключення ПЛІС виконуємо у відповідності до наступної таблиці, див. табл. 4.1.

Таблиця 4.1

Призначення контактів для дослідження ЛЕ «І»

Зміна	Контакт	Опис
X1	PIN_J6	Перемикач SW[0]
X2	PIN_H5	Перемикач SW[1]
Y	PIN_J1	Світло діод LEDG [0]

В результаті компіляції проекту в середовищі Quartus формується файл конфігурації ПЛІС із розширенням *.sof. Для запису файлу конфігурації в пам'ять ПЛІС через порт USB персонального комп'ютера потрібно перетворити цей файл у формат з розширенням *.rbf. Для цього можна скористатися командою з меню **File –> Convert Programming Files**. У вікні **Convert Programming Files**, див. рис. 4.11, в розділі Output programming files слід вибрати тип файлу Raw Binary File (*.rbf) і задати шлях до вихідного файлу.

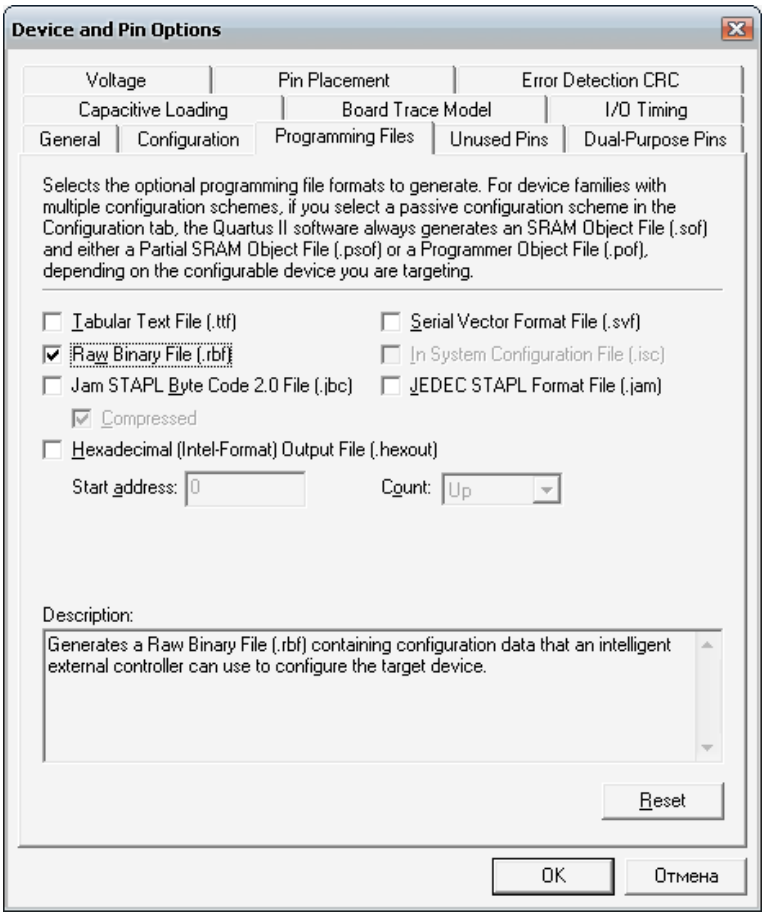


Рис. 4.10. Налаштування для автоматичної компіляції проекту

Примітка! У Quartus II є можливість налаштувати проект так, щоб файл *.rbf формувався автоматично в процесі компіляції. Для цього на закладці Settings в меню Assignments (див. рис. 4.10) потрібно натиснути кнопку Devise & pin Options. Далі в розділі Programming file необхідно встановити галочку Raw Binary File (*.rbf). Завантаження файлу *.rbf в ПЛІС проводиться за допомогою окремої програми – завантажувача.

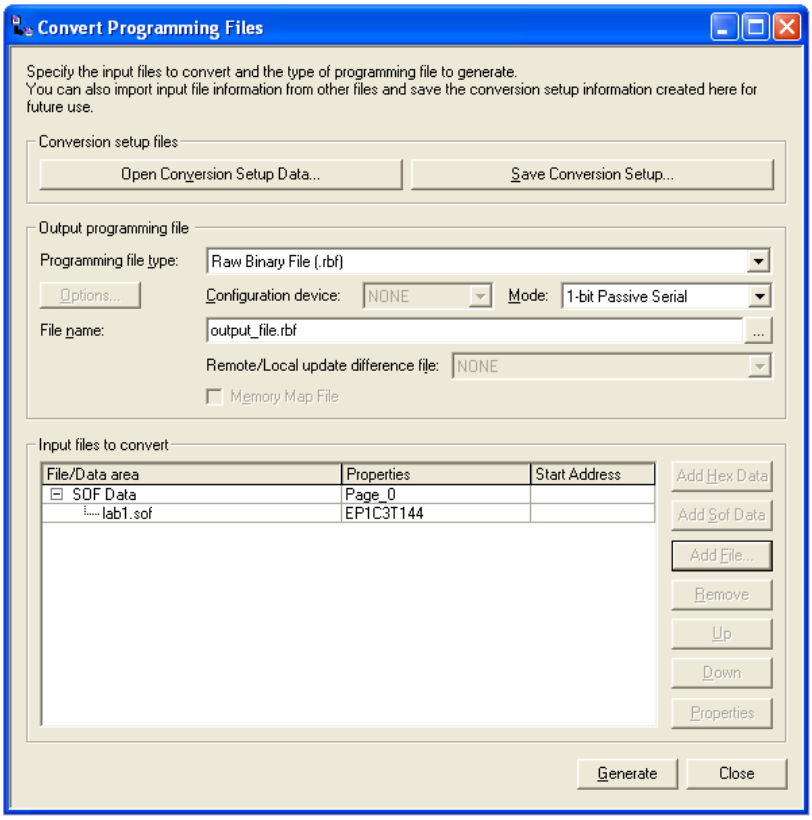


Рис. 4.11. Вікно Convert Programming Files

У розділі **Input files to convert** слід натиснути кнопку **Add File** і в меню, що відкрилося, вибрати шлях до початкового файлу з розширенням *.sof. Далі необхідно натискаємо кнопку **Generate**.

Далі натискакмо **Tools->Programmer**, див. рис. **4.12.**

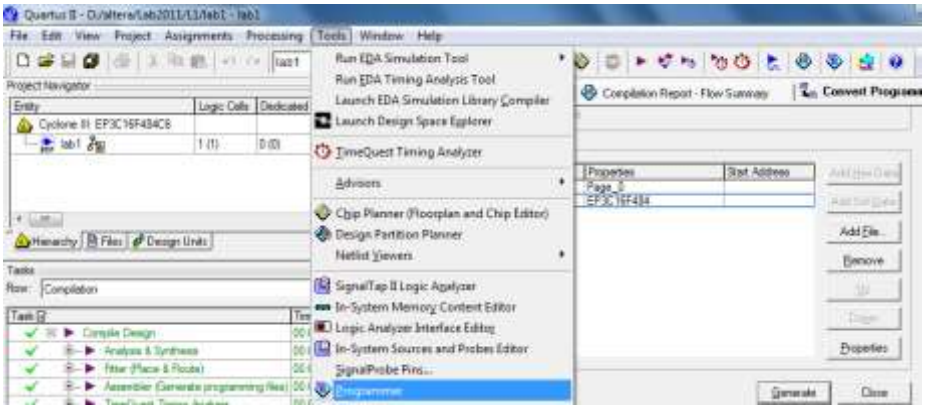


Рис. 4.12. Прошивка ПЛІС

Далі у вікні програма тора здійснюємо прошивку ПЛІС, див. рис. 4.13.

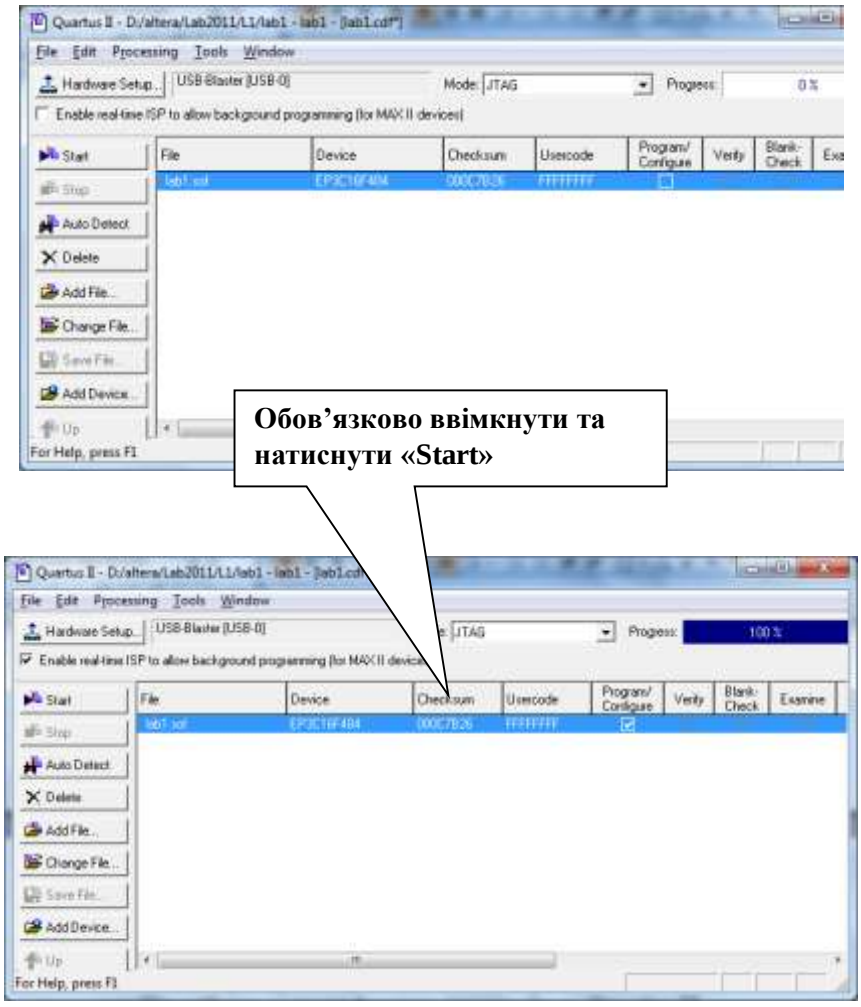


Рис. 4.13. Прошивка ПЛІС

Всі роботи виконуються на стенді DE0, основою якого є ПЛІС сімейства Cyclon III EP3C16F484C6. Принципова схема стенду розглянута у лабораторній роботі 1. Стенд підключається до персонального комп'ютера через порт USB. До контактів ПЛІС підключено 10 світло діодів (LEDG0-LEDG9), які можуть бути використані як індикатори логічних рівнів в різних точках схеми. Чотири 7-сегментні індикатори призначено для індикації цифр від 0 до 9. Світло діоди LEDG0-LEDG9 використовуються для сигналізації стану перемикачів.

Після завантаження файлу конфігурації в ПЛІС провести дослідження логічного елемента «І». Для цього за допомогою перемикачів SW[0] і SW[1] послідовно встановити можливі комбінації логічних рівнів на входах елемента «І». При цьому кожного разу контролювати логічний рівень на виході елемента «І». Якщо світло діод LEDG0 світиться – це логічна одиниця, інакше – логічний нуль, див. рис. 3.14. За наслідками дослідження заповнити таблицю 4.2.

Таблиця 4.2

Протокол дослідження логічного елементу «І»

X1	X2	У
0	0	
0	1	
1	0	
1	1	

Завдання для самостійної роботи:

1. За допомогою системи Quartus II зібрати схеми для дослідження логічних елементів «АБО», «АБО-НЕ», «І-НЕ».

2. Підключити схеми до кристалу та провести відповідні дослідження наведених логічних елементів. Скласти протоколи.

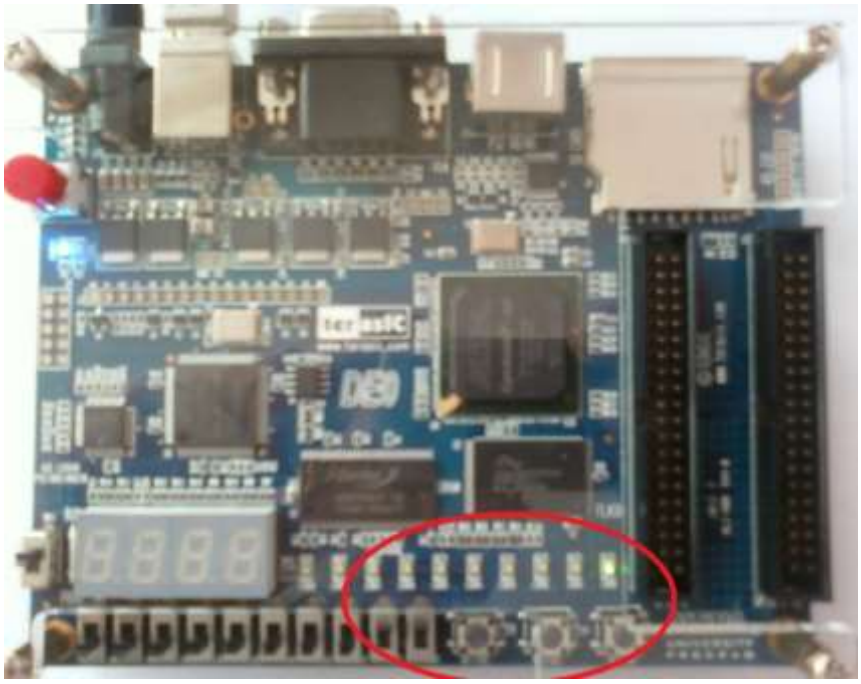


Рис. 4.14. Результат моделювання роботи логічного елемента «І»

Питання для самоперевірки

1. Що таке файл конфігурації ПЛІС?
2. Яким чином проводиться конфігурування ПЛІС?
3. Як проводиться підключення електричної схеми ПЛІС до зовнішніх контактів?

Розділ 5. Розробка та дослідження комбінаційних схем цифрових автоматів у Quartus II

5.1 Методології проектування комбінаційних схем цифрових автоматів із програмованою логікою

Необхідно спроектувати ПЛІС яка буде реалізовувати наступну логічну функцію.

$$f = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C + D.$$

Етап 1. Мінімізація.

$$f = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C + D = \overline{B} \cdot \overline{C} \cdot (A + \overline{A}) + \overline{A} \cdot B \cdot C + D =$$
$$= \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C + D.$$

Тобто необхідно розробити пристрій якій реалізує наступний вираз $f = \overline{A} \cdot B \cdot C + \overline{B} \cdot \overline{C} + D.$

Етап 2. Створення проекту.

Побудуємо наступну схему пристрою, див. рис. 5.1.

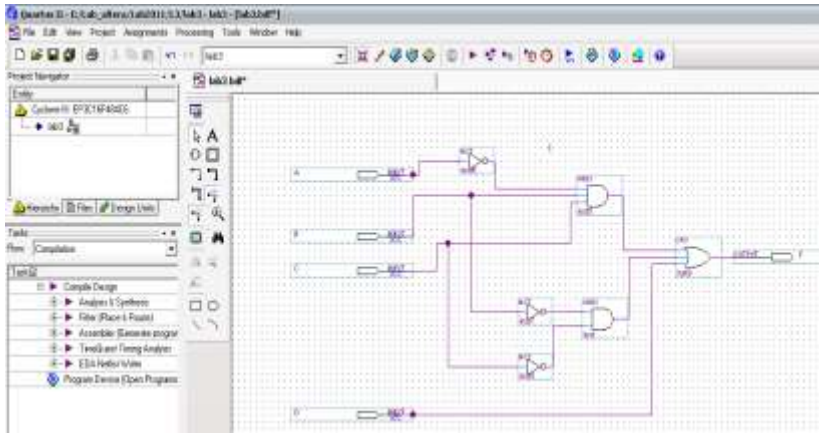


Рис. 5.1. Схема у Quartus II

Етап 3. Підключення кристалу.

Після того, як проведена компіляція проекту слід підключити вхідні і вихідні контакти до зовнішніх контактів ПЛІС. Для цього слід вибрати **Assignments/Pins**. В результаті з'явиться вікно **Pin Planer**, див. рис. 5.2.

Після завантаження файлу конфігурації в ПЛІС провести дослідження комбінаційної схеми.

Для цього за допомогою перемикачів SW[0], SW[1], SW[2] та SW[3] послідовно встановити можливі комбінації логічних рівнів на входах елементів схеми. При цьому кожного разу контролювати логічний рівень на виході F. Якщо світло діод **LEDG0** світиться – це логічна одиниця, інакше – логічний нуль.

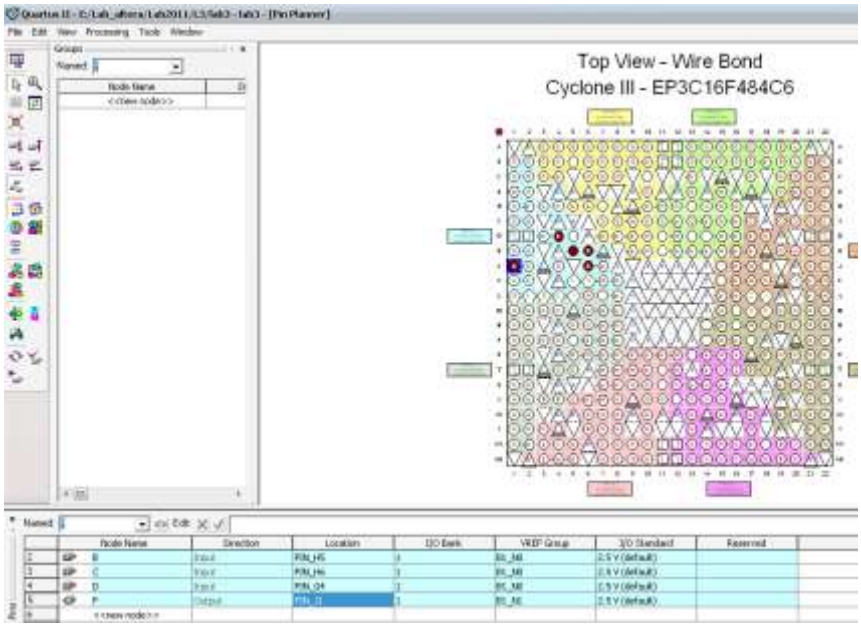


Рис. 5.2. Вікно Pin Planer

За результатами дослідження заповнити таблицю 5.1.

Таблиця 5.1

Протокол дослідження комбінаційної схеми

A	B	C	D	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Таблиця 5.2

Варіанти завдань для проектування

Варіант	Логічний вираз
1	$y = A\cdot(\overline{B}\cdot\overline{D}+D)+A\cdot\overline{B}\cdot C+\overline{D}+\overline{A\cdot\overline{B}\cdot C}.$
2	$y = \overline{\overline{A}\cdot\overline{B}\cdot\overline{D}+A\cdot B\cdot D}+\overline{A\cdot\overline{B}\cdot C}+\overline{D}\cdot\overline{B}\cdot C.$
3	$y = \overline{A}\cdot\overline{B}\cdot\overline{D}+A\cdot B\cdot D+\overline{D}+\overline{A\cdot\overline{B}\cdot\overline{C}}.$
4	$y = A\cdot\overline{B}\cdot C+A\cdot(\overline{\overline{B}\cdot\overline{D}+C})+\overline{D}+A\cdot\overline{B}\cdot C.$
5	$y = A\cdot\overline{B}+A\cdot(\overline{B}\cdot\overline{D}+C)+\overline{D}\cdot A\cdot\overline{B}\cdot C.$
6	$y = A\cdot\overline{B}+A\cdot(\overline{B}\cdot\overline{D}+C)+\overline{D\cdot\overline{A}\cdot\overline{B}\cdot\overline{C}}.$
7	$y = \overline{A}\cdot\overline{B}+A\cdot(\overline{B}\cdot\overline{D}+C\cdot\overline{B})+\overline{D}\cdot A\cdot\overline{B}\cdot C+\overline{A}\cdot B.$
8	$y = \overline{A}\cdot\overline{B}\cdot D+A\cdot(\overline{B}\cdot\overline{D}+C)+\overline{D}\cdot A\cdot\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot C.$
9	$y = A\cdot(\overline{B}\cdot\overline{D}+C)+\overline{D}\cdot A\cdot\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot\overline{C}+A\cdot\overline{D}+C.$
10	$y = \overline{C}(\overline{A}\cdot\overline{B}\cdot\overline{D}+D)+\overline{A\cdot\overline{B}\cdot C+\overline{D}}.$
11	$y = A\cdot(\overline{B}\cdot\overline{D}+C)+\overline{D}\cdot(A\cdot\overline{B}\cdot\overline{C}+\overline{\overline{A}\cdot B\cdot\overline{C}}).$
12	$y = A\cdot(\overline{B}\cdot\overline{D}+C)+D\cdot(A\cdot B\cdot\overline{C}+\overline{A}\cdot B\cdot\overline{C}).$
13	$y = B\cdot(A\cdot\overline{D}+C)+\overline{D}\cdot(\overline{B}\cdot\overline{C}\cdot A+\overline{A}\cdot B\cdot C)+D.$
14	$y = A\cdot\overline{B}\cdot\overline{D}\cdot C+A\cdot(\overline{B}\cdot\overline{D}+C\cdot D)+D\cdot\overline{A}\cdot\overline{B}\cdot\overline{C}.$
15	$y = C\cdot(\overline{A}\cdot\overline{D}+B)+\overline{D}\cdot(A\cdot\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot\overline{C}).$
16	$y = A\cdot\overline{B}\cdot\overline{D}+C\cdot\overline{D}+\overline{A\cdot\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot\overline{C}}.$
17	$y = (\overline{C+D})+\overline{A\cdot\overline{C}\cdot\overline{D}}+A\cdot\overline{B}\cdot\overline{C}+\overline{A}\cdot\overline{B}\cdot C\cdot D+A\cdot C\cdot D.$
18	$y = A\cdot B\cdot(\overline{\overline{C}\cdot D})+A\cdot\overline{B}\cdot D+\overline{B}\cdot\overline{C}\cdot\overline{D}.$
19	$y = A\cdot(\overline{\overline{B}\cdot\overline{D}+C})+\overline{D}\cdot A\cdot\overline{B}\cdot\overline{C}+\overline{\overline{A}\cdot B\cdot\overline{C}+A\cdot\overline{D}+C}.$
21	$y = \overline{A}\cdot\overline{B}+\overline{A\cdot(\overline{B}\cdot\overline{D}+C\cdot\overline{B})}+\overline{\overline{D}\cdot A\cdot\overline{B}\cdot C+\overline{A}\cdot B}.$
22	$y = A\cdot\overline{B}+\overline{A\cdot(\overline{B}\cdot\overline{D}+C)}+\overline{D\cdot\overline{A}\cdot\overline{B}\cdot\overline{C}}.$
23	$y = \overline{A\cdot(\overline{B}\cdot\overline{D}+D)+A\cdot\overline{B}\cdot C+\overline{D}+\overline{A\cdot\overline{B}\cdot C}}.$
24	$y = A\cdot B\cdot(\overline{\overline{C}\cdot D})+A\cdot\overline{B}\cdot D+\overline{B}\cdot\overline{C}\cdot\overline{D}+\overline{C\cdot\overline{B}\cdot D}.$

Питання для самоперевірки

1. Етапи проектування комбінаційної схеми із застосуванням ПЛІС?
2. Як проводиться підключення комбінаційної схеми до зовнішніх контактів стенду?

5.2. Розробка та дослідження суматорів у QUARTUS II

Створимо простий проект для плати «Суматор», який буде складатися усього з пари модулів, буде складати два двухбітних числа. На платі DE0 ми використовуємо 4 кнопки. Дві кнопки - одне двухбітное число, ще дві кнопки - друге двухбітное число. Відображати результат складання будемо на світлодіодах плати DE0.

Етап 1. Складання суматора для однобітних чисел.

На рис. 5.3 показана схема класичного суматора. Цей суматор складає два однобітних числа *a* і *b*. При виконанні складання однобітних чисел може трапитися «переповнення», тобто результат вже буде двухбітним (1+1=2 або в двійковому вигляді 11 =10). Тому у нас є вихідний сигнал перенесення *c_out*. Додатковий вхідний сигнал *c_in* служить для прийому сигналу переносу від суматорів молодших розрядів, при побудові многобітних суматорів.

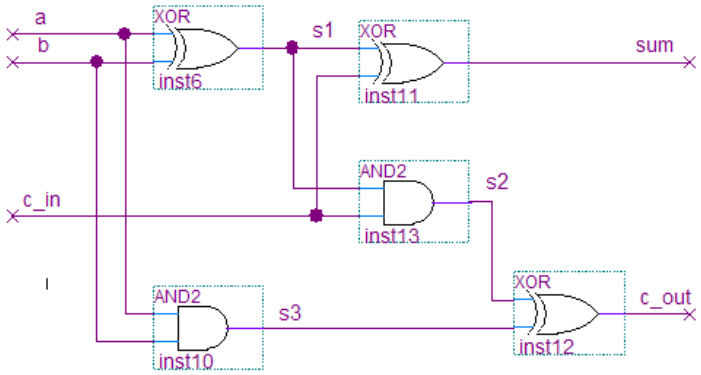


Рис. 5.3. Схема суматора

Етап 1. Коли ми отримали та дослідили одно бітний суматор, можемо зробити, наприклад, чотирьох бітний (із послідовним переносом), див. рис. 5.4.

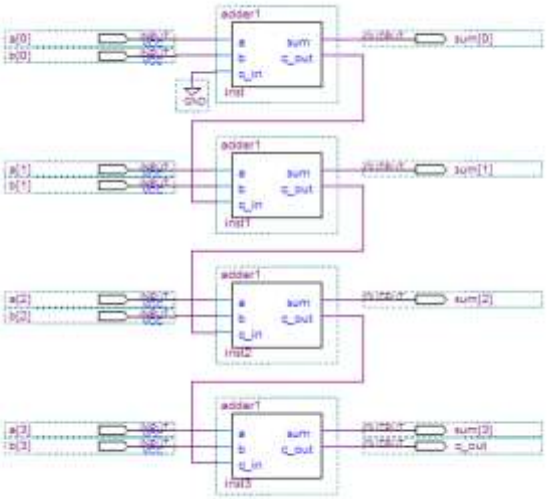


Рис. 5.4. Схема чотирьох бітного суматора

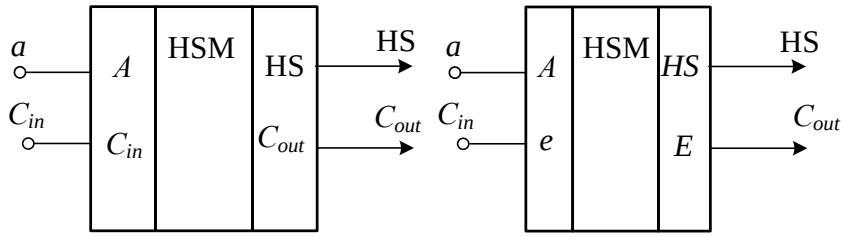
Питання для самоперевірки

1. Етапи проектування комбінаційної схеми із застосуванням ПЛІС?
2. Як проводиться підключення комбінаційної схеми до зовнішніх контактів стенду?

5.3. Розробка та дослідження канонічних двійкових суматорів в середовищі QUARTUS II

Серед найпростіших арифметичних пристроїв можна виділити напівсуматори, які відносяться до класу комбінаційних цифрових пристроїв. Напівсуматори використовуються для побудови повних двійкових суматорів, для організації лічильників на базі лічильної схеми, для виконання мікрооперації «інкремент», тощо.

Умовне графічне позначення (УГП) напівсуматора на функціональних схемах приведено на рис.5.1. Відповідно до рис.5.1,а напівсуматор (*HSM – half-adder*) має два входи (A і C_{in}) та два виходи (*HS* і C_{out}). Вхід A призначений для підключення операнду, а вхід C_{in} – для підключення вхідного переносу (*input carry*). Вихід *HS* є виходом напівсуми (*half-sum*), а вихід C_{out} – виводом вихідного переносу (*output carry*). Для більш компактного позначення на логічних і функціональних схемах вивід вхідного переносу будемо позначати літерою «*e*», а вихідний перенос – літерою «*E*» (рис.5.1,б).



а) *б)*

Рис.5.1. Умовні графічні позначення напівсуматора

Таблиця істинності напівсуматора приведена в табл.5.1.

Таблиця істинності напівсуматора Таблиця 5.1

a	*e* (C_{in})	*HS*	*E* (C_{out})
0	*0*	*0*	*0*
0	*1*	*1*	*0*
1	*0*	*1*	*0*
1	*1*	*0*	*1*

З таблиці можна отримати логічні вирази в булевому базисі, за допомогою яких можна реалізувати схему напівсуматора:

$HS = \bar{a}e + a\bar{e} = \overline{\bar{a}\bar{e} + ae};\ \ E = C_{out} = ae.$

За допомогою логічної функції «додавання за модулем два» (*XOR*) можна також отримати логічні формули для напівсуми:

$HS = a \oplus e = \bar{a} \oplus \bar{e} = \overline{a \oplus \bar{e}}; = \overline{\bar{a} \oplus e}$.

Один з варіантів логічної схеми напівсуматора приведено на рис.5.2.

Крім того, логічна схема напівсуматора може бути побудована на базі типових комбінаційних схем дешифратора або мультиплексора. На рис.5.3 приведена логічна схема напівсуматора на базі дешифратора 2→4.

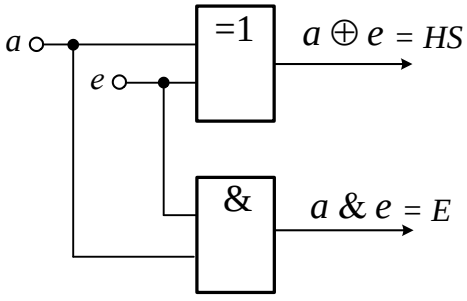


Рис.5.2. Логічна схема напівсуматора

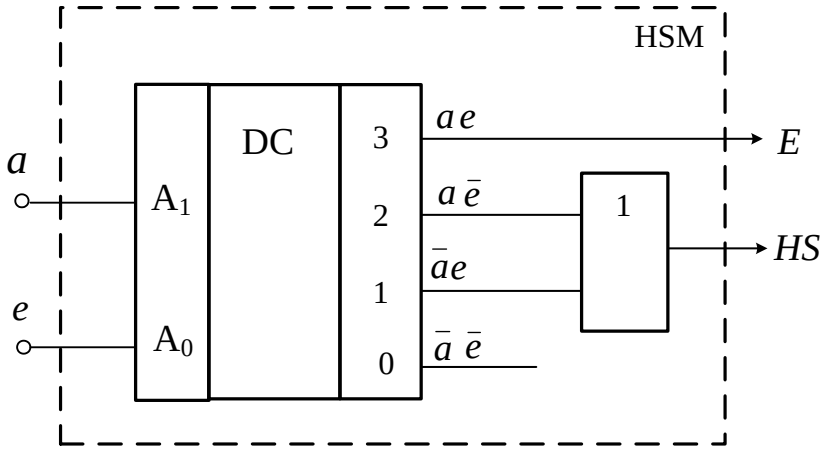


Рис.5.3. Логічна схема напівсуматора на базі дешифратора 2→4

Аналогічним чином відбувається побудова і функціонування напіввіднімачів (*half-subtractor*), які призначені для виконання мікрооперації «декремент». УГП і таблиця істинності напіввіднімача приведено відповідно на рис.5.4 і в табл.5.2. На УГП напіввіднімача використовуються такі позначення:

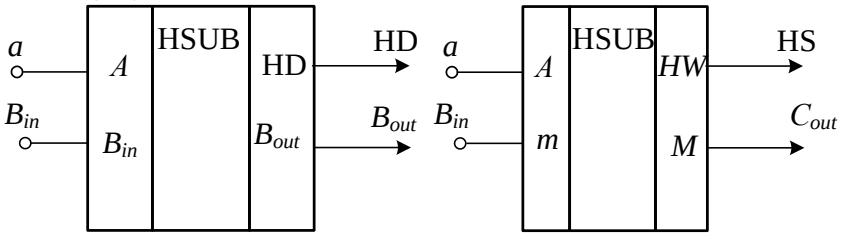
- *a* – операнд;

- *m, B_{in}* – позначення вхідної позики з сусіднього молодшого розряду (*input borrow*);

- *M, B_{out}* – позначення вихідної позики з сусіднього старшого розряду (*output borrow*);

- *HD, HW* – вихід напіврізниці (*half-difference*).

Під час розглядання принципів функціонування арифметичних пристроїв літера «*B*» буде використовуватися для позначення другого операнду арифметичних операцій, тому далі виоди вхідної, вихідної позики та напіврізниці будуть позначатися відповідно *m, M* та *HW*.



а) *б)*

Рис.5.4. Умовні графічні позначення напіввіднімача

Таблиця істинності напіввіднімача Таблиця 5.2

a	*m (B_{in})*	*HW (HD)*	*M (B_{out})*
0	*0*	*0*	*0*
0	*1*	*1*	*1*
1	*0*	*1*	*0*
1	*1*	*0*	*0*

Функціонування напіввіднімачів характеризується логічними виразами, що можуть бути отримані з таблиці істинності (табл.5.2):

$$HW = \bar{a}m + a\bar{m} = \overline{\bar{a}\bar{m} + am}; \quad M = B_{out} = \bar{a}e.$$

$$HW = a \oplus n = \bar{a} \oplus \bar{n} = \overline{a \oplus \bar{n}}; = \overline{\bar{a} \oplus n} \ .$$

Аналізуючи формули реалізації напівсуматора і напіввіднімача, можна зробити висновок, що логічні вирази для виводів напівсуми і напіврізниці описуються однаковими логічними формулами, а формули для реалізації вихідних переносу і позики відрізняються тільки інверсією змінної «*a*». Це необхідно враховувати при проектуванні комбінованого арифметичного пристрою напівсуматора - напіввіднімача, в якому при виконанні мікрооперації «інкремент» на вивід підключення операнду надходить змінна «*a*», а при виконанні мікрооперації «декремент» на цей вхід необхідно підключити $\bar{a}$.

Аналіз функціонування цифрових пристроїв можна виконувати з використанням моделювання поведінки таких пристроїв, тобто визначення реакції пристрою на зміни вхідних сигналів за допомогою будь-якої системи логічного моделювання. В якості альтернативи САПР QUARTUS II, наприклад, нижче приведені схеми та результати моделювання напівсуматора, логічна схема якого приведена на рис.5.2. Моделювання виконано з використанням системи схемотехнічного проектування MicroCap.

При цьому необхідно відмітити, що для завантаження логічної схеми до плати стенда DE0 необхідно спочатку розробити поведінкову модель пристрою. Ця модель часто будується на функціональному, а не на логічному рівні опису. Тому для того, щоб бачити взаємозв'язок між логічною схемою, поведінковою моделлю та результатами функціонування пристрою, що досліджується, далі опис досліджуваних пристроїв буде приведений в такій послідовності:

- логічна схема пристрою в середовищі MicroCap;
- поведінкова модель в середовищі САПР QUARTUS II;
- результати моделювання пристрою в середовищі MicroCap;
- результати моделювання пристрою в середовищі САПР QUARTUS II.

Два останні пункти використовуються для підтвердження відповідності поведінкової моделі результатам моделювання.

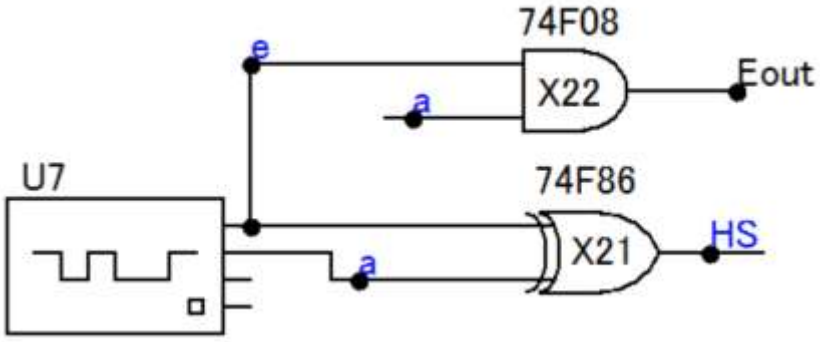


Рис.5.5. Схема напівсуматора, що моделюється

Відповідно до схеми напівсуматор реалізований на базі логічних елементів транзисторно-транзисторної логіки (ТТЛ) серії SN74F08 (КР1531ЛИ1) і SN74F86 (КР1531ЛП5). Результати моделювання в статичному режимі приведені на рис.5.6. Моделювання статичного режиму функціонування пристрою передбачає використання нульових затримок логічних елементів або використання затримок, значення яких набагато менше за тривалість вхідних сигналів ($t_\Delta << t_{вх}$, де t_Δ – час затримки логічного елемента; $t_{вх}$ – тривалість вхідних сигналів). Таким чином, під час моделювання статичних режимів перевіряється відповідність функціонування цифрового пристрою таблиці істинності або таблиці переходів.

Моделювання динамічних режимів функціонування цифрових пристроїв передбачає використання затримок логічних елементів, сумірних з тривалістю вхідних сигналів. В цьому режимі визначаються значення часу спрацьовування, встановлення, підготовки пристрою, виявляється наявність гонок сигналів в схемі, тощо.

На рис.5.7 представлені результати моделювання напівсуматора в динамічному режимі. З часової діаграми функціонування пристрою можна визначити час спрацьовування, який на рис.5.7 показаний за допомогою виносок.

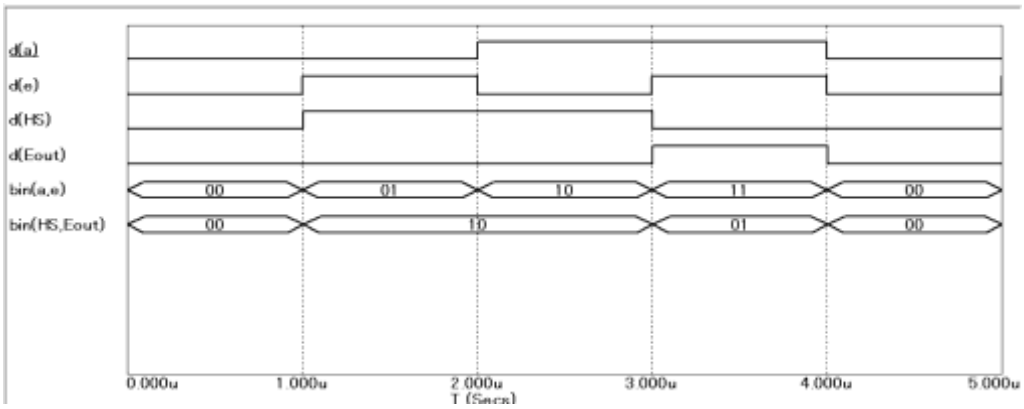


Рис.5.6. Результати моделювання напівсуматора в статичному режимі

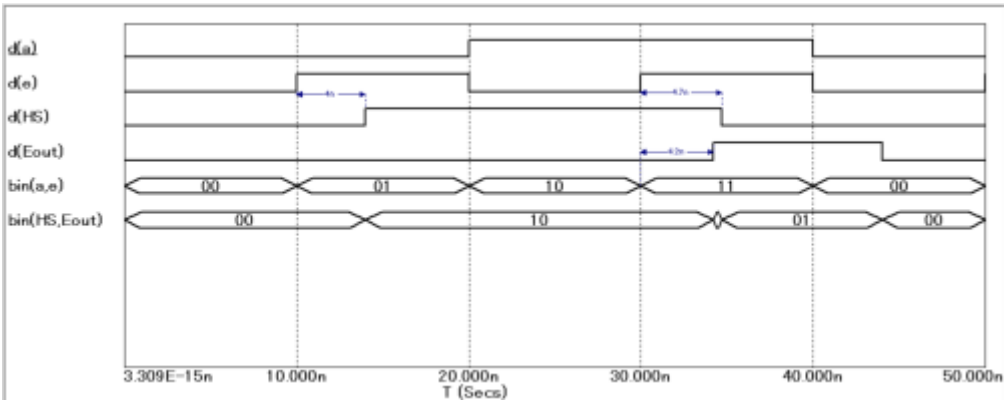


Рис.5.7. Результати моделювання напівсуматора в динамічному режимі

Для виконання арифметичних операцій в комп'ютерних системах використовуються повні двійкові суматори.

Під час виконання арифметичної операції додавання двох n- розрядних операндів A (a_{n-1}, …, a_0) та B (b_{n-1}, …, b_0) в i- тому розряді результат операції може бути визначений за формулою:

$$W \cdot E_i + S_i = a_i + b_i + e_i,$$

де n – кількість розрядів операндів;

a_i, b_i – значення i-тих двійкових цифр операндів A і B;

e_i – значення переносу в i-тий біт з молодшого сусіднього розряду;

S_i – сума двійкових операндів в i-тому розряді;

E_i – вихідний перенос з i-того розряду до сусіднього старшого розряду;

W – вага вихідного переносу E ($W = 2^n$).

При цьому E_i визначається за формулою:

$$E_i = \begin{cases} 0, \text{ якщо } (a_i + b_i + e_i) < W; \\ 1, \text{ якщо } (a_i + b_i + e_i) \geq W. \end{cases}$$

Очевидно, що для двійкового однорозрядного суматора ($n = 1$) $W = 2$.

УГП однорозрядного двійкового суматора, яке використовується на функціональних схемах, наведено на рис.5.8.

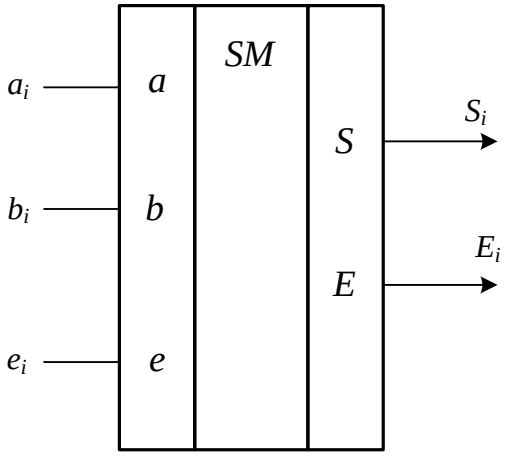


Рис.5.8. УГП однорозрядного суматора

Принцип функціонування однорозрядного повного суматора може бути описаний за допомогою таблиці істинності (табл.5.3).

Таблиця істинності однорозрядного Таблиця 5.3

a_i	b_i	e_i	E_i	S_i	Десяткове значення суми $(a_i+b_i+e_i)$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Відповідно до таблиці 5.3 досконала диз'юнктивна нормальна форма (ДДНФ) виглядає таким чином:

$$S = \bar{a}\,\bar{b}\,e \vee \bar{a}\,b\,\bar{e} \vee a\,\bar{b}\,\bar{e} \vee a\,b\,e;$$
$$E = a\,b\,\bar{e} \vee a\,\bar{b}\,e \vee \bar{a}\,b\,e \vee a\,b\,e. \qquad (5.1)$$

Двійковий суматор, реалізований за формулами (5.1, будемо називати канонічним суматором.

В отриманих вище логічних виразах кожний терм ДДНФ (кожна кон'юнкція в ДДНФ) відповідає рядкам таблиці істинності (табл. 5.3), в яких функції суми і вихідного переносу приймають одиничні значення (нагадаємо, що ДДНФ являє собою диз'юнкцію конституент одиниці).

Для реалізації логічної схеми суматора на базі ДДНФ доцільно використовувати готові інтегральні схеми дешифратора або мультиплексора. Наприклад, дешифратор є комбінаційною логічною схемою, яка формує на виходах конституенти одиниці. При цьому кількість адресних входів дешифратора відповідає кількості логічних змінних суматора, тобто необхідно використовувати дешифратор 3→8. Функціональна логічна схема канонічного суматора, яка реалізована за формулами (5.1), приведена на рис. 5.9. В цій схемі дешифратор у складі суматора виконує формування необхідних термів функцій S і E: $\bar{a}\,\bar{b}\,e$, $\bar{a}\,b\,\bar{e}$, $\bar{a}\,b\,e$, $a\,\bar{b}\,\bar{e}$, $a\,\bar{b}\,e$, $a\,b\,\bar{e}$ та $a\,b\,e$. Таким чином, на першому виході дешифратора реалізовано терм $\bar{a}\,\bar{b}\,e$, на виході 2 – терм $\bar{a}\,b\,\bar{e}$ і т.д.

Аналізуючи, таблицю істинності суматора, можна зробити висновок, що логічні функції суми і переносу характеризуються властивістю

самоподвійності (самодвійковості). Ця властивість полягає в тому, що інвертування значень аргументів функції призводить до інвертування значення самої функції, тобто:

$$S = F_S(a,b,e) = \overline{F_S(\bar{a},\bar{b},\bar{e})}; \qquad E = F_E(a,b,e) = \overline{F_E(\bar{a},\bar{b},\bar{e})}.$$

Самоподвійність функцій суматора широко застосовується при побудові багаторозрядних арифметичних пристроїв.

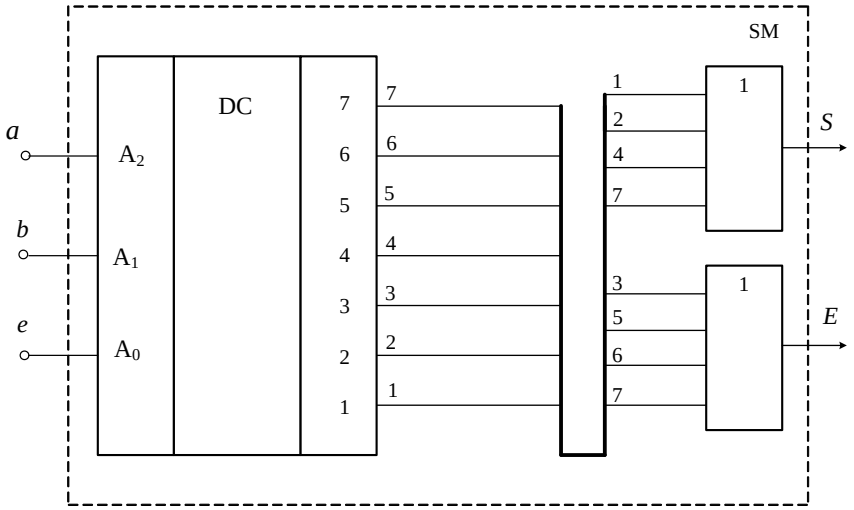


Рисунок 5.9 – Функціональна схема канонічного суматора на базі дешифратора

Схема для моделювання канонічного суматора на основі дешифратора в середовищі MicroCap приведена на рис. 5.10.

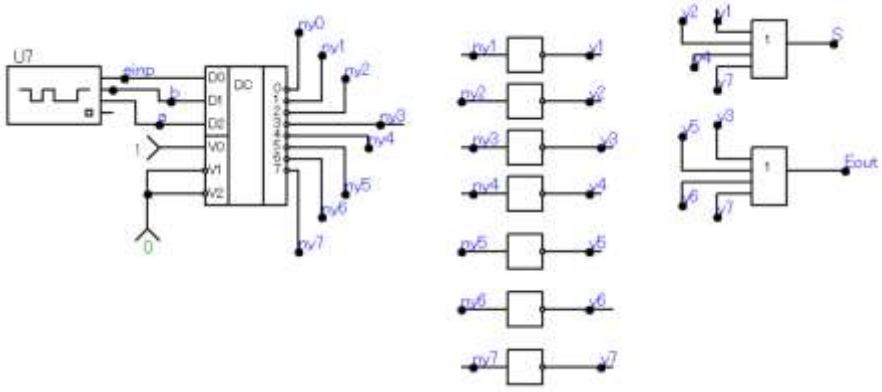


Рисунок 5.10 – Схема для моделювання канонічного суматора на базі дешифратора

Нижче приведена поведінкова модель суматора на базі дешифратора на мові *Verilog*:

```
module ProjectNULES (
 a,b,einp,res, S, Eout
);
        input a;
```

```
    input b;
    input einp;

    wire [2:0] bus;
    output [7:0] res;
    output S;
    output Eout;
    reg [7:0] res;

    assign bus[0] = einp;
    assign bus[1] = b;
    assign bus[2] = a;

    always @(bus)
    begin
        case (bus)
            3'b000 : res = 8'b11111110;
            3'b001 : res = 8'b11111101;
            3'b010 : res = 8'b11111011;
            3'b011 : res = 8'b11110111;
            3'b100 : res = 8'b11101111;
            3'b101 : res = 8'b11011111;
            3'b110 : res = 8'b10111111;
            default : res = 8'b01111111;
        endcase
    end

    assign S = !res[1] | !res[2] | !res[4] | !res[7] ;
    assign Eout = !res[3] | !res[5] | !res[6] | !res[7] ;
endmodule
```

Результати моделювання схеми, приведеної на рис.5.10, в середовищі MicroCap приведена на рис. 5.11,а.

На часових діаграмах використовуються наступні скорочення: *d(a), d(b)* – часові діаграми доданків *a, b*; *d(einp), d(Eout)* – часові діаграми відповідно вхідного і вихідного переносів; *d(S)* – часова діаграма суми.

На рис.5.11,б приведені результати моделювання канонічного суматора на базі дешифратора в середовищі САПР QUARTUS II.

Результати моделювання на рис.5.11,а,б вказують на тотожну реакцію пристрою на вхідні сигнали в статичному режимі (в динамічному режимі результати відрізняються в зв'язку з різними затримками, які використовуються в моделях логічних елементів систем моделювання).

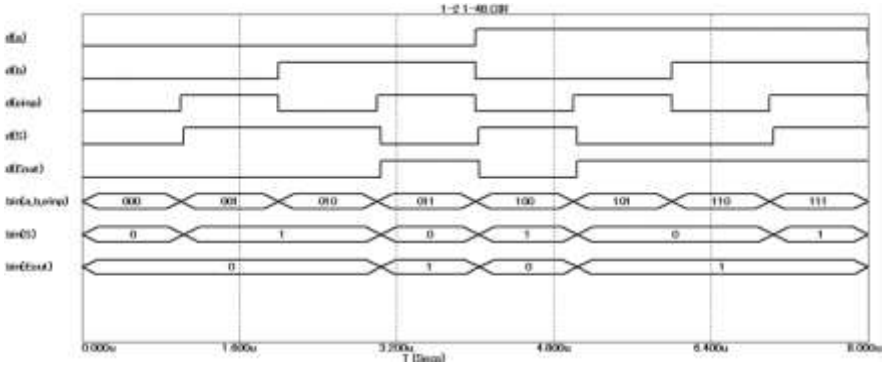


Рисунок 5.11,а – Результати моделювання канонічного суматора на базі дешифратора в середовищі MicroCap

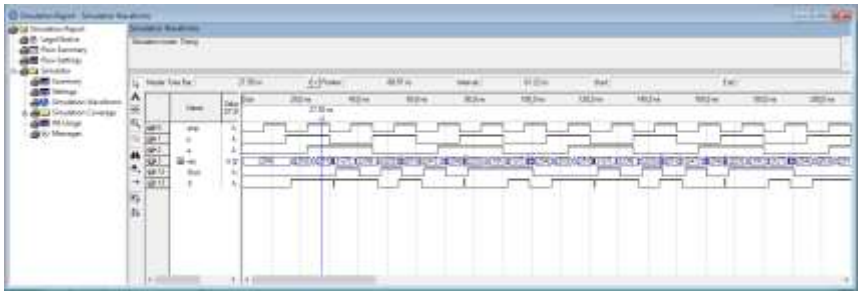


Рисунок 5.11,б – Результати моделювання канонічного суматора на базі дешифратора в середовищі САПР QUARTUS II

На рис. 5.12,а приведена схема для моделювання і часові діаграми функціонування канонічного суматора на основі дешифратора з інверсними виходами в середовищі MicroCap. В схемі логічні функції цього суматора реалізовані за формулами:

$$S = \bar{a}\bar{b}e \vee \bar{a}b\bar{e} \vee a\bar{b}\bar{e} \vee abe = \overline{\bar{a}\bar{b}e}\ \overline{\bar{a}b\bar{e}}\ \overline{a\bar{b}\bar{e}}\ \overline{abe};$$

$$E = ab\bar{e} \vee a\bar{b}e \vee \bar{a}be \vee abe = \overline{ab\bar{e}}\ \overline{a\bar{b}e}\ \overline{\bar{a}be}\ \overline{abe}.$$

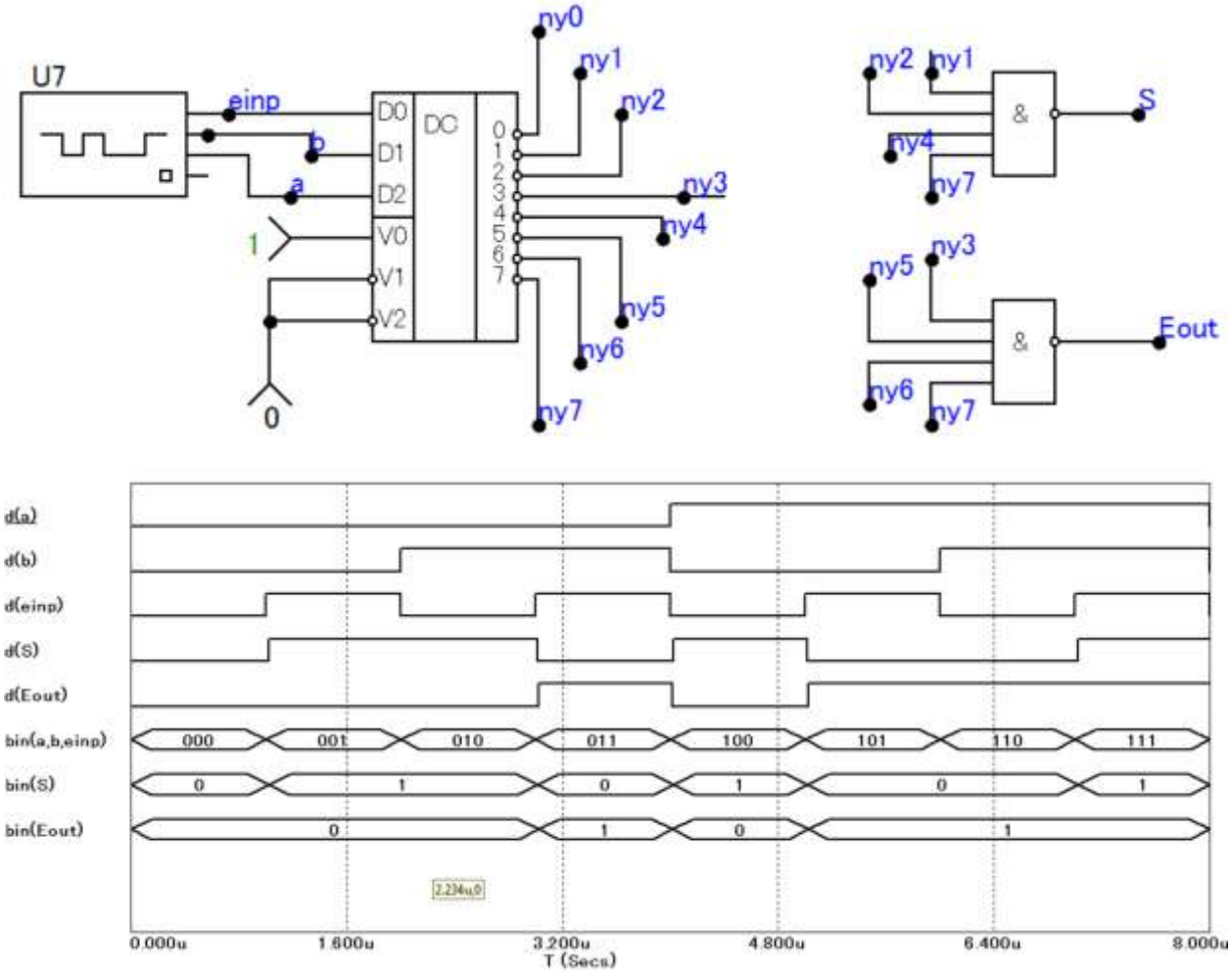


Рисунок 5.12,а – Результати моделювання канонічного суматора на основі дешифратора з інверсними виходами в середовищі MicroCap

Поведінкова модель суматора на базі дешифратора з інверсними виходами на мові *Verilog* приведена нижче:

```
module ProjectNULES (
 a,b,einp,res, S, Eout
);
        input a;
        input b;
        input einp;
        wire [2:0] bus;
        output [7:0] res;

        output S;
        output Eout;
        reg [7:0] res;
```

```
        assign bus[0] = einp;
        assign bus[1] = b;
        assign bus[2] = a;

        always @(bus)
        begin
            case (bus)
                3'b000 : res = 8'b11111110;
                3'b001 : res = 8'b11111101;
                3'b010 : res = 8'b11111011;
                3'b011 : res = 8'b11110111;
                3'b100 : res = 8'b11101111;
                3'b101 : res = 8'b11011111;
                3'b110 : res = 8'b10111111;
                default : res = 8'b01111111;
            endcase
        end

        assign S = !(res[1] & res[2] & res[4] & res[7]) ;
        assign Eout = !(res[3] & res[5] & res[6] & res[7]) ;
    endmodule
```

Часові діаграми функціонування пристрою, логічна схема якого приведена на рис.5.12,а, в середовищі САПР QUARTUS II приведена на рис.5.12,б.

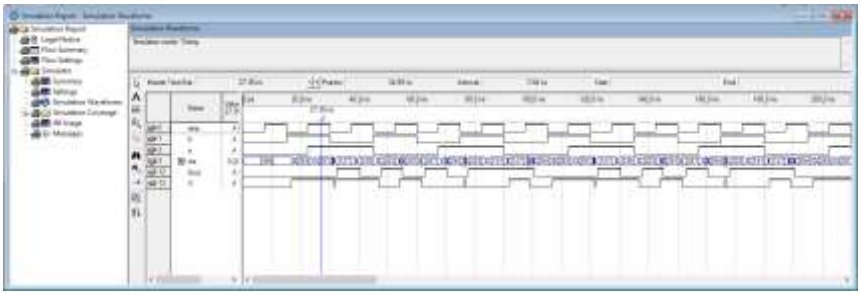


Рисунок 5.12,б – Часові діаграми канонічного суматора на основі дешифратора з інверсними виходами в середовищі САПР QUARTUS II

Результати моделювання канонічного суматора, побудованого в базисі І-АБО-НІ, в середовищах MicroCap і САПР QUARTUS II приведені на рис. 5.13а,б.

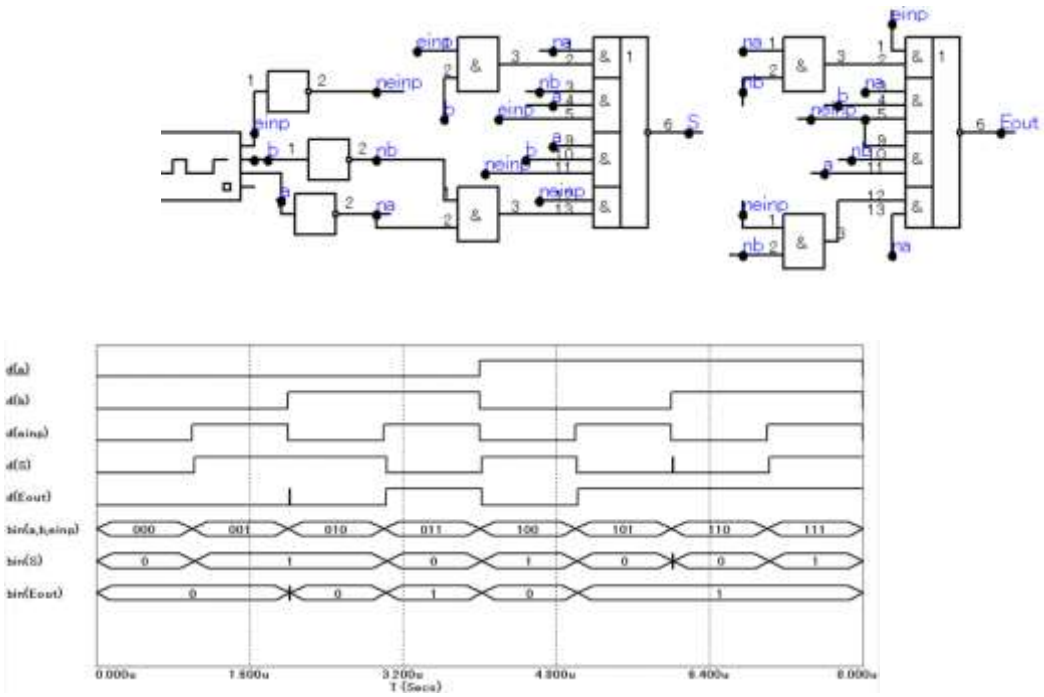


Рисунок 5.13,а – Часові діаграми і схема канонічного суматора в базисі І-АБО-НІ в середовищі MicroCap

Поведінкова модель такого суматора приведена нижче:

```
module ProjectNULES (a, b, einp, S, Eout);
        wire neinp;
        wire nb;
        wire na;

        input a;
        input b;
        input einp;

        output S;
        output Eout;

        assign neinp = ~einp;
        assign nb = ~b;
        assign na = ~a;

        assign S = !((na & einp & b) | (nb & a & einp) | (a & b & neinp)
| (na & nb & neinp));
        assign Eout = !((na && nb && einp) || (na && b && neinp) || (a
&& nb && neinp) || (na && nb && neinp));

endmodule
```

Результати моделювання цього суматора в середовищі QUARTUS II приведені на рис.5.13,б.

Рисунок 5.13,б – Часові діаграми канонічного суматора в базисі І-АБО-НІ в середовищі QUARTUS II

Для спрощення логічної схеми, тобто для скорочення апаратних витрат під час побудови суматора, необхідно застосувати процедуру мінімізації логічних функцій з використанням карт Карно. Таблиця істинності суматора може бути представлена за допомогою карт Карно (рис. 5.14).

a \ *be*	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$S = f_S(a,b,e)$

a \ *be*	00	01	11	10
0	0	0	1	1
1	0	1	1	1

$E = f_E(a,b,e)$

Рисунок 5.14 – Карти Карно для мінімізації логічних функцій двійкового суматора

З карт Карно витікає, що функція суми не мінімізується і може бути представлена у вигляді ДДНФ (5.1). В результаті мінімізації функції переносу отримаємо формулу:

$$E = ab + ae + be;$$

Суматор, побудований за цим принципом, будемо називати мінімальним суматором. При використанні базису І-АБО-НІ (з урахуванням властивості самоподвійності функцій S і E) для реалізації схеми мінімального суматора можна записати:

$$S = \overline{\bar{a}\bar{b}e + \bar{a}b\bar{e} + a\bar{b}\bar{e} + abe}; \quad E = \overline{\bar{\bar{a}}\bar{\bar{b}} + \bar{\bar{a}}\bar{\bar{e}} + \bar{\bar{b}}\bar{\bar{e}}} \qquad (5.2)$$

Логічна схема мінімального суматора в базисі І-АБО-НІ приведена на рис. 5.15.

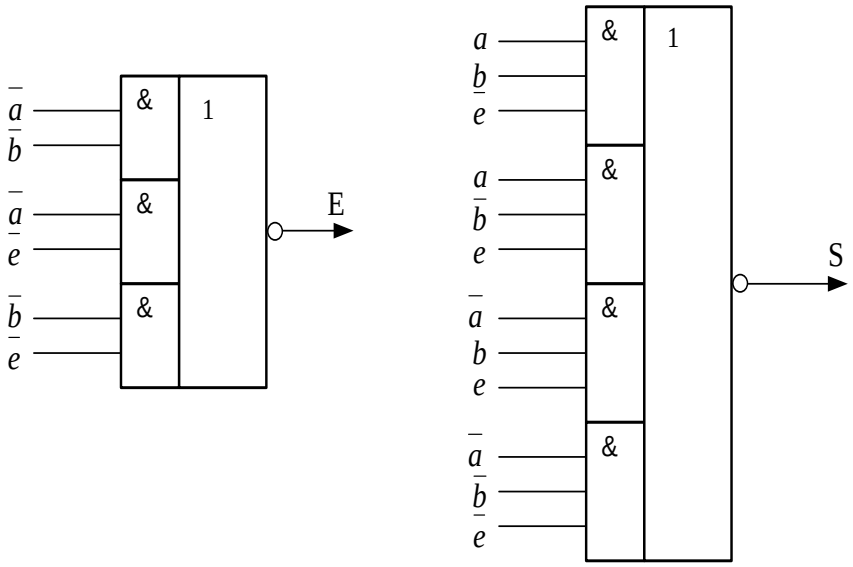


Рисунок 5.15 – Логічна схема мінімального суматора в базисі І-АБО-НІ

Логічна схема такого суматора в середовищі MicroCap (рис.5.16) та результати моделювання за допомогою MicroCap і САПР QUARTUS II приведені відповідно на рис.5.17,а,б.

Логічну схему мінімального суматора може бути реалізована за допомогою двох інших форм в елементному базисі «Виключне АБО» («XOR», («Додавання за модулем два») та в базисі І-АБО-НІ (відповідно рис.5.18 і рис.5.20.

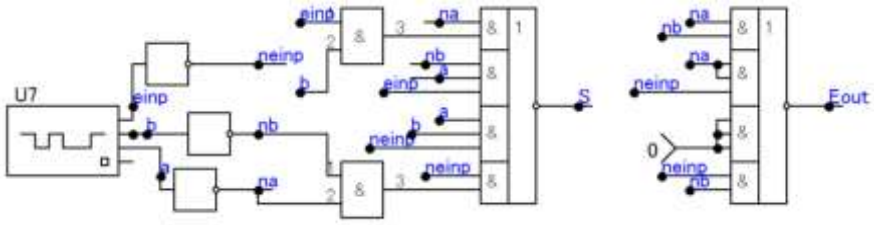


Рисунок 5.16 – Схема моделювання мінімального суматора

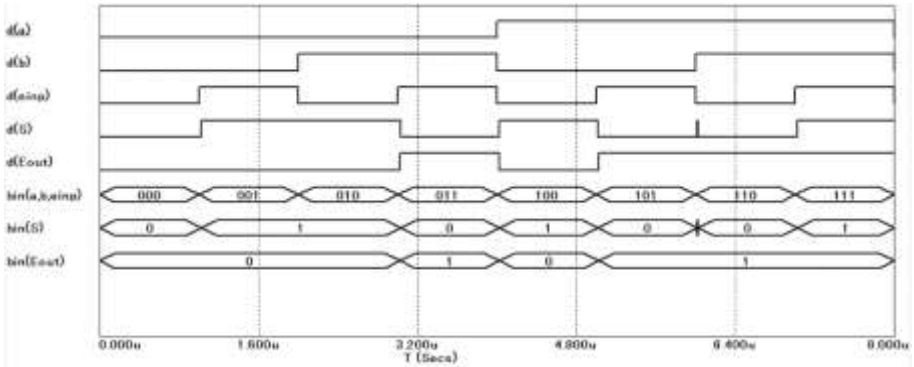


Рисунок 5.17,а – Часові діаграми мінімального суматора в середовищі MicroCap

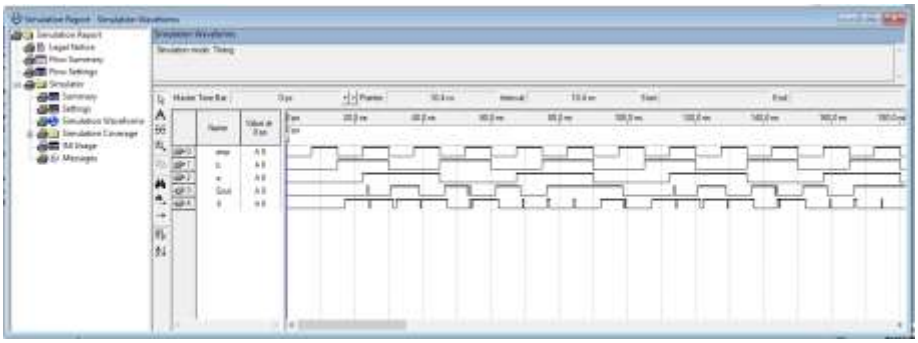


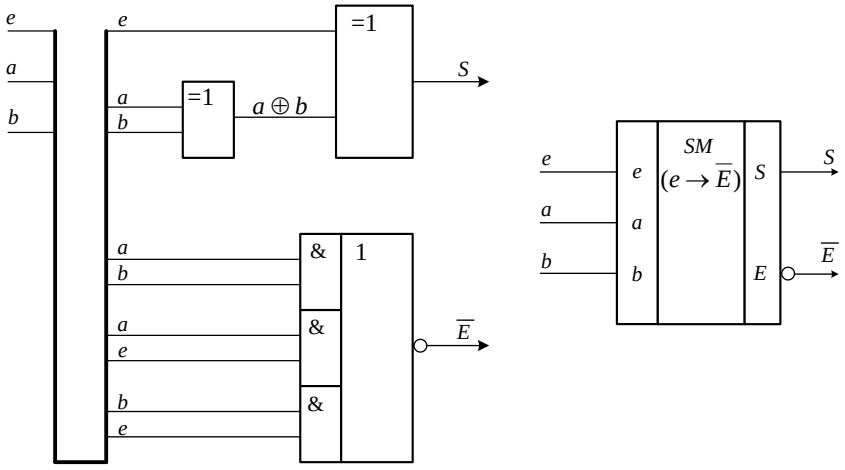
Рисунок 5.17,б – Часові діаграми мінімального суматора в середовищі QUARTUS II

Поведінкова модель мінімального суматора приведена нижчу:

```
module ProjectNULES (a, b, einp, S, Eout);
    wire neinp;
    wire nb;
    wire na;

    input a;
    input b;
    input einp;

    output S;
    output Eout;

    assign neinp = ~einp;
    assign nb = ~b;
    assign na = ~a;

    assign S = !((na & einp & b) | (nb & a & einp) | (a & b & neinp) | (na & nb & neinp));
    assign Eout = !((na && nb && einp) || (na && b && neinp) || (a && nb && neinp) || (na && nb && neinp));

endmodule
```

При цьому функція вихідного переносу E мінімального суматора (рис.5.18) реалізована з використанням тракту $(e \to \overline{E})$, тобто на входи суматора подається пряме значення вхідного переносу, а на виході формується інверсне значення вихідного переносу. Така структурна організація використовується під час проектування багаторозрядних суматорів. Для побудови логічної схеми формування суми цього пристрою використовується перетворення:

$$S = \bar{a}\bar{b}e + \bar{a}b\bar{e} + a\bar{b}\bar{e} + abe =$$
$$= e(\bar{a}\bar{b} + ab) + \bar{e}(\bar{a}b + a\bar{b}) =$$
$$= e(\overline{a \oplus b}) + \bar{e}(a \oplus b) = e \oplus (a \oplus b);$$
$$\bar{E} = \overline{ab + ae + be}.$$



а) *б)*

Рисунок 5.18 – Логічна схема (*а*) і УГП (*б*) мінімального суматора з трактом ($e \to \bar{E}$)

Схема модельного експерименту в середовищі MicroCap приведена на рис.5.19.

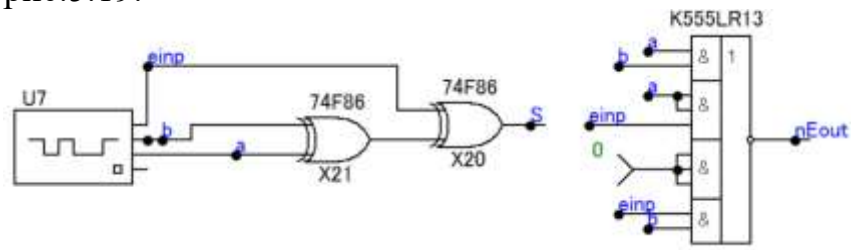


Рисунок 5.19 – Схема моделювання мінімального суматора з трактом ($e \to \bar{E}$)

Поведінкова модель на мові Verilog, яка реалізує функціонування мінімального суматора з трактом ($e \to \bar{E}$, приведена нижче:

```
module ProjectNULES (
  input a,
  input b,
  input einp,

  output   nEout,
  output   S
);
        assign S  = (a ^ b) ^ einp;
        assign nEout = !((a & b) | (a & einp) | (einp & b));
endmodule
```

Часові діаграми мінімального суматора з трактом $e \rightarrow \overline{E}$ в середовищі MicroCap і QUARTUS II приведені відповідно на рис.5.20,а і рис.5.20,б.

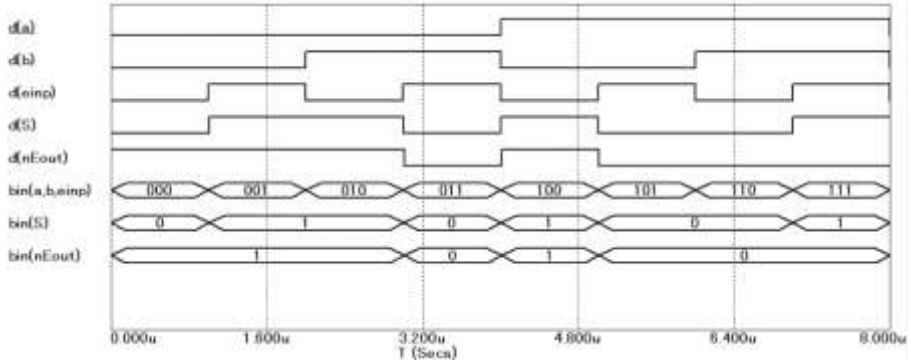


Рисунок 5.20,а – Часові діаграми мінімального суматора з трактом ($e \rightarrow \overline{E}$) в середовищі MicroCap

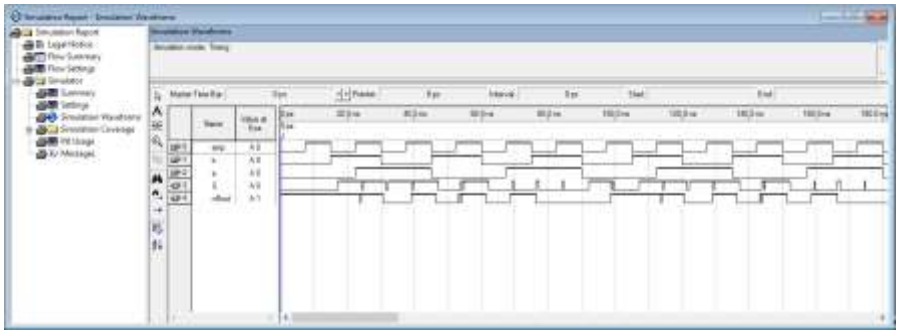


Рисунок 5.20,б – Часові діаграми суматора з трактом ($e \rightarrow \overline{E}$) в середовищі QUARTUS II

Логічна схема на рис. 5.21 реалізує мінімальний суматор з трактом розповсюдження переносу $(\bar{e} \rightarrow E)$ та побудована за допомогою перетворень:

$$S = (a \oplus b) \oplus e = (\bar{a}b + a\bar{b}) \oplus e =$$

$$= \overline{(\bar{a}b + a\bar{b})}\, e + (\bar{a}b + a\bar{b})\, \bar{e} =$$

$$= (\bar{a}\bar{b} + ab)\, e + \overline{(\bar{a}\bar{b} + ab)}\, \bar{e} = \overline{(a \oplus b)} \oplus (\bar{e}).$$

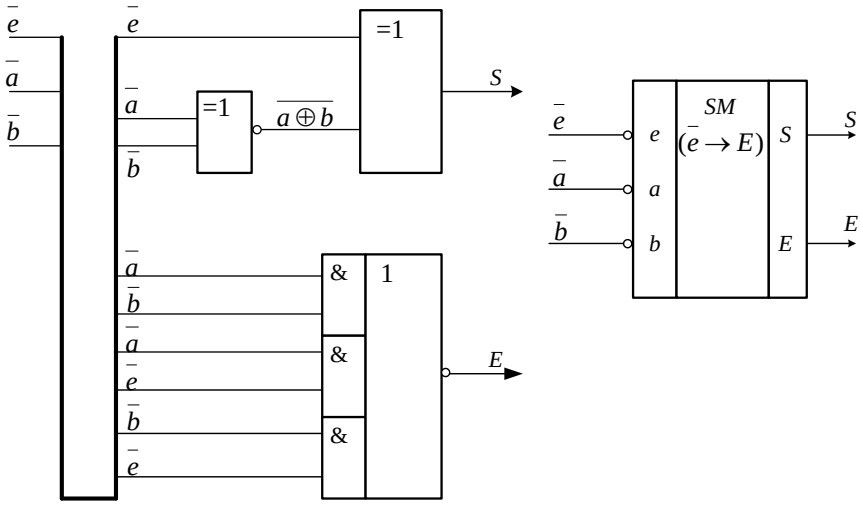
Таким чином, отримаємо

$$S = \overline{(a \oplus b)} \oplus (\bar{e});\ E = \overline{\bar{a}\bar{b}} + \overline{\bar{a}\bar{e}} + \overline{\bar{b}\bar{e}}.$$

Поведінкова модель на мові Verilog реалізує функціонування мінімального суматора з трактом $(\bar{e} \rightarrow E)$ та приведена нижче:

```
module ProjectNULES (
  input a,
  input b,
  input einp,

  output  nEout,
  output  S
);
        wire na;
```

```
        wire nb;
        wire neinp;

        assign na = !a;
        assign nb = !b;
        assign neinp = !einp;
        assign S  = (na ^ nb) ^ neinp;
        assign nEout = !((na & nb) | (na & neinp) | (neinp & nb));
endmodule
```



а) *б)*

Рисунок 5.21 – Логічна схема (*а*) і УГП (*б*) мінімального суматора з трактом $(\bar{e} \to E)$

Результат моделювання у вигляді часових діаграм мінімального суматора з трактом $(\bar{e} \to E)$ в середовищі MicroCap і QUARTUS II приведені відповідно на рис.5.22,а і рис.5.22,б.

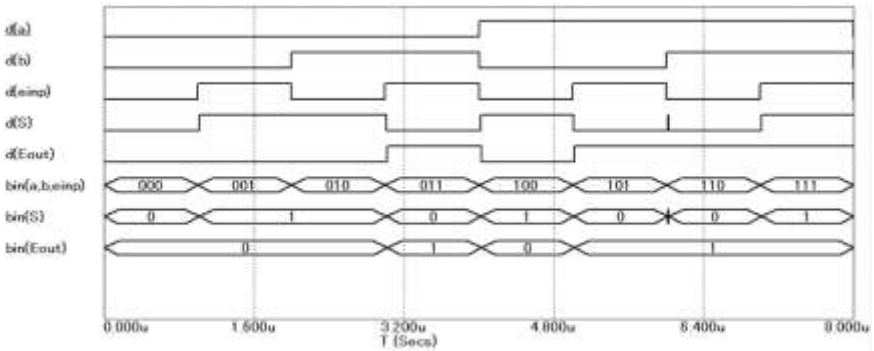


Рисунок 5.22,а – Часові діаграми мінімального суматора з трактом $(\bar{e} \to E)$ в середовищі MicroCap

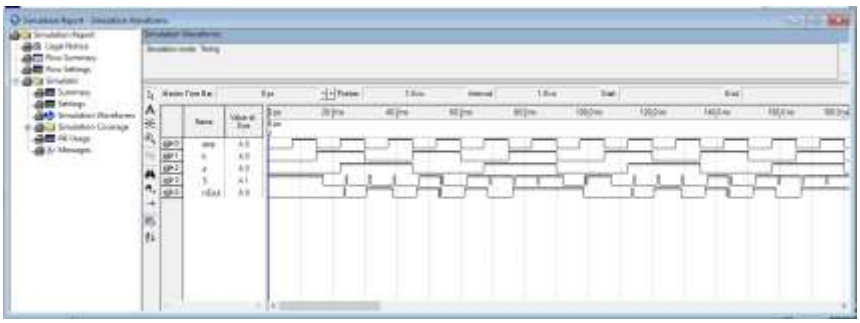


Рисунок 5.22,б – Часові діаграми мінімального суматора з трактом $(\bar{e} \to E)$ в середовищі QUARTUS II

Аналізуючи логічні схеми суматора з різними способами реалізації трактів розповсюдження переносу, можна відзначити, що ці суматори є еквівалентними з точки зору апаратурних витрат. Для реалізації логічної схеми формування суми використовується властивість функції додавання за модулем два: $a \oplus b = \bar{a} \oplus \bar{b}$.

Розділ 6. Симуляція проекту в Quartus II

Симуляція - це програмне тестування проекту, завжди робиться до його перевірки. Мікросхема має входи і виходи. Якщо ми поставимо, наприклад, послідовність і тривалість вхідних імпульсів, то система проектування зможе прорахувати результуючі сигнали на всіх виходах. І не тільки на виходах! За допомогою симуляції можна заглянути всередину всіх модулів проекту і подивитися на процеси, що відбуваються всередині чіпа. Так, ще до тестування в реальному чіпі можна зрозуміти правильно працює проект, чи ні.

В якості об'єкта обраний D-тригер на логічних елементах І-НЕ.

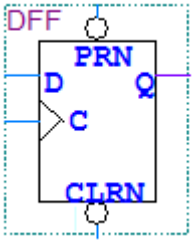


Рис. 6.1. Умовне позначення D-тригера

D-тригер дуже важливий пристрій. Без нього немає цифрових мікросхем. Елемент D-тригер запам'ятовує один біт інформації. У сучасних складних схемах їх багато тисяч. Тригери запам'ятовують стан вхідної лінії даних саме в момент фронту сигналу тактової частоти. Це властивість чутливості саме до перепаду рівня сигналу (від логічного нуля до логічної одиниці або навпаки) тактової частоти дуже цінно.

Стан цифрової схеми в кожен момент схеми описується значеннями записаними в тригера. По кожному фронту тактової частоти тригера перемикаються в новий стан, зберігаючи нове, обчислене комбінаторними функціями значення.

Так як запам'ятовування в тригерах відбувається відразу у всій схемі, то у комбінаторних схем є час на обчислення своїх функцій - час обчислення не більше періоду тактової частоти. Щоб зрозуміти, як же все таки працює тригер, ми і будемо його симулювати. Перша частина цього опису - створення проекту Quartus II для D-тригера.

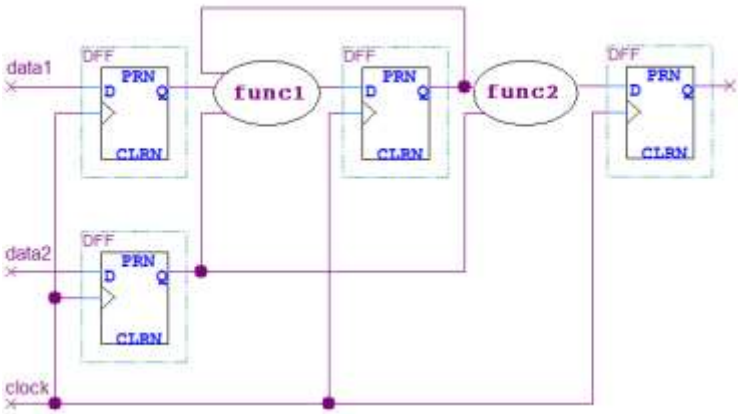


Рис. 6.2. Приклад комбінаційної схеми із використовуванням D-тригера

Крок 1. Зберігаємо наш новий файл під ім'ям **flipflop.bdf.**
Крок 2. У полі графічного редактора схем натискаємо праву кнопку миші і у меню вибираємо пункти **Insert \ Symbol -** вставити елемент**.**
Крок 3. Вибираємо з бібліотек **Quartus II.** Нам потрібен трехвходовий елемент **І-НЕ - це \ primitives \ logic \ nand3.**
Крок 4. Так само вибираємо потрібні нам входи і виходи **\ primitives \ pin \ input і \ primitives \ pin \ output.**
Крок 5. Ось всі елементи які будуть нам потрібні для схеми**. *Ctrl+C* і *Ctrl+V*** дозволяють копіювати і вставляти виділені ділянки схеми або окремі елементи.
Крок 6. Потрібно 6 трьох - входові елементів **І-НЕ,** див. рис. 6.3.

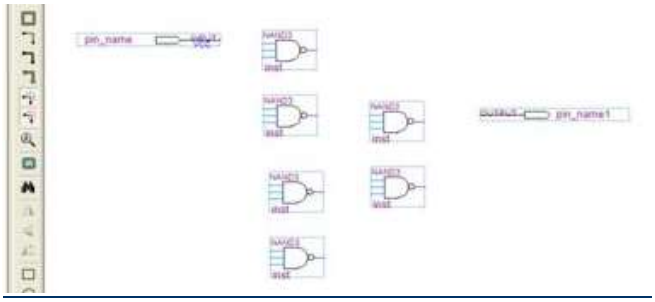


Рис. 6.3. Складання схеми

Крок 7. Додамо ще входи і виходи. Міняємо ім'я елемента. З'єднуємо залишилися входи і виходи всіх елементів ось так, як на цій схемі. Ось це - типова схема *D* - тригера. Вона містить вхід даних *D*, вхід для тактової частоти *C*, два асинхронних входу: скид R_n і установка S_n, два виходи: прямий *Q* та інверсний Q_n , див. рис. 6.4.

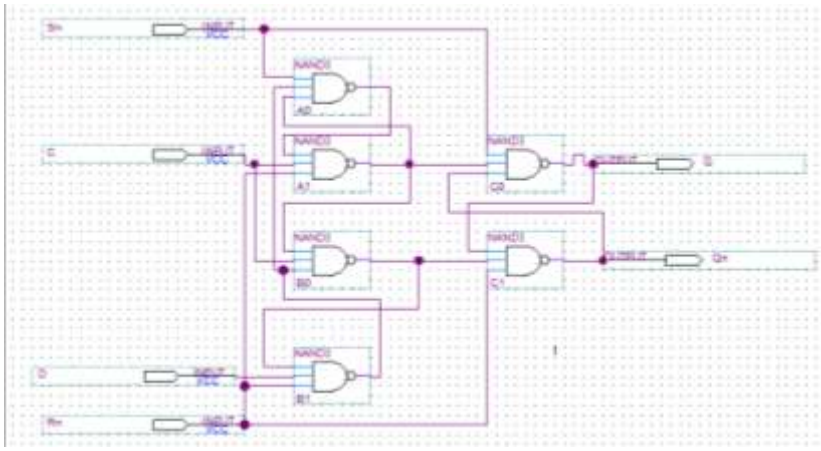


Рис. 6.4. Сформована схема

Крок 8. Перш ніж робити симуляцію потрібно відкомпілювати проект. Вибираємо пункти меню Processing \ Start Compilation, див. рис. 6.5.

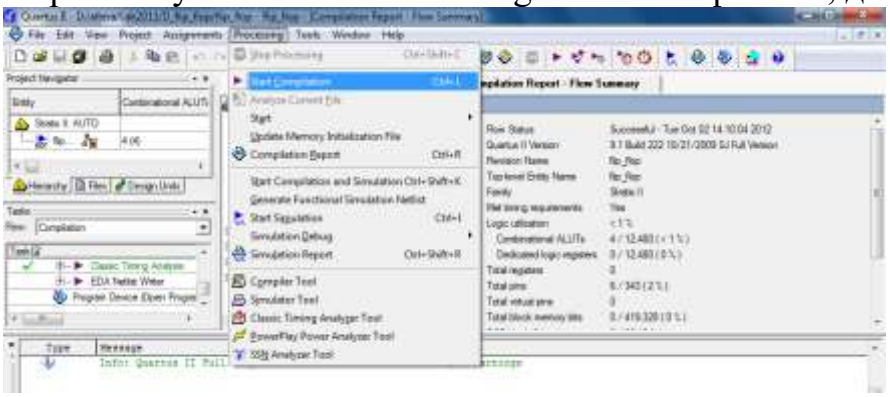


Рис. 6.5. Компіляція проекту

Крок 9. Після компіляції потрібно ще створити спеціальний netlist для функціональної симуляції. Вибираємо пункт меню **Processing \ Generate Functional Simulation Netlist, див. рис. 6.6.**

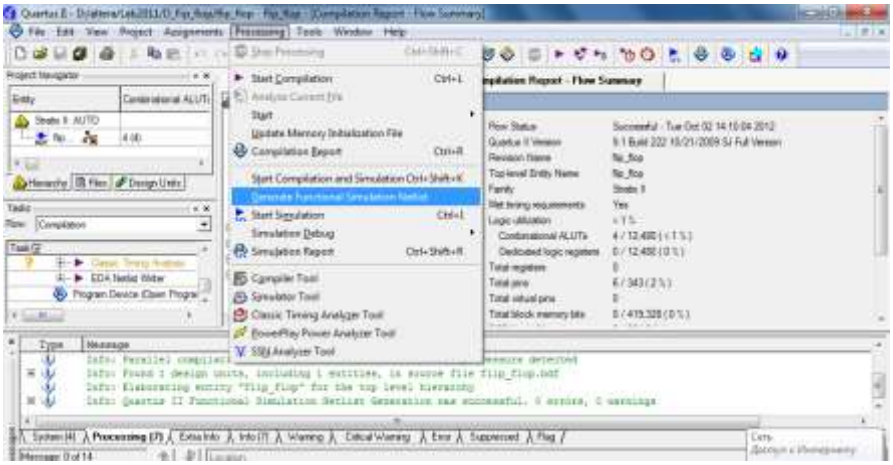


Рис. 6.6. Функціональна симуляція проекту

Симуляція буває тимчасова і функціональна. Функціональна симуляція дозволяє перевірити саме логіку роботи. З її допомогою ми можемо переконатися, що проект може і повинен працювати, так як задумано. Перш за все, потрібно робити саме функціональну симуляцію. Тимчасова симуляція дозволяє побачити сигнали з урахуванням всіх виникаючих затримок сигналу між елементами, входами і виходами. Тимчасову симуляцію роблять в останню чергу, вже після функціональної симуляції, щоб переконатися, що схема може працювати на потрібній заданій частоті. Для невеликих проектів цілком достатньо робити тільки функціональну симуляцію.

Крок 10. Створимо файл для симуляції. У ньому ми будемо описувати вхідні сигнали, і задавати вихідні сигнали, які хочемо переглядати. Вибираємо пункт меню File \ New і потім в діалозі Verification \ Debugging Files \ Vector Waveform File, див. рис. 6.7.

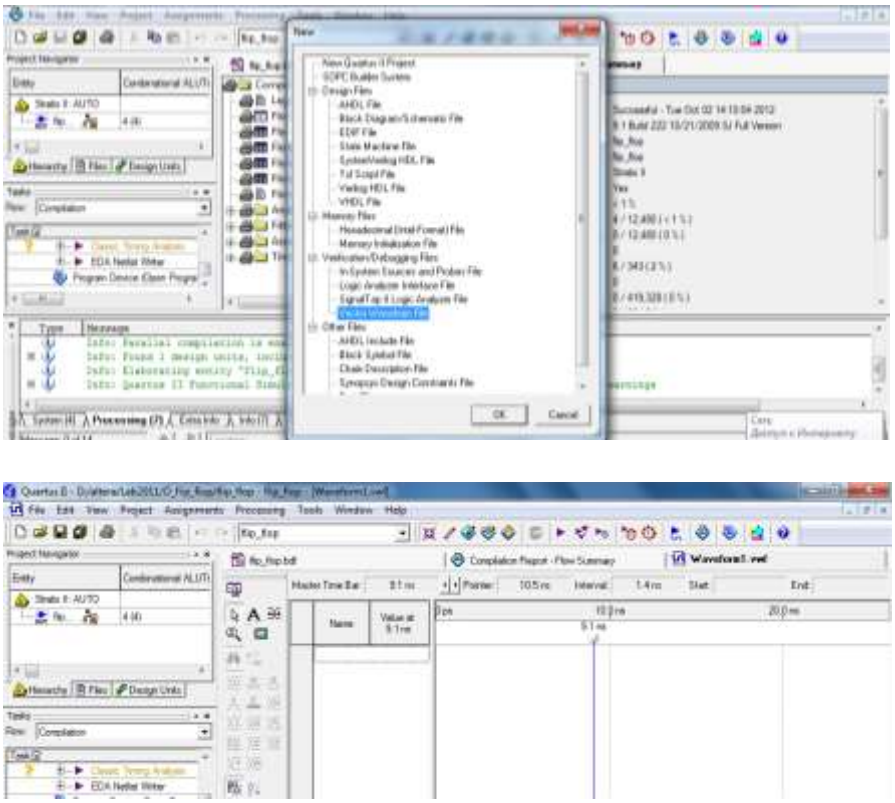


Рис. 6.7. Створення файлу для симуляція проекту

Крок 11. Збережемо наш файл симуляції під ім'ям, наприклад, flip_flop.vwf.

Крок 12. Ми повинні вибрати сигнали, які будемо дивитися, і які будемо визначати. Клік правою клавіші миші на лівій панелі Names і в випадаючому меню вибираємо пункт Insert \ Insert Node or Bus, див. рис. 6.8.

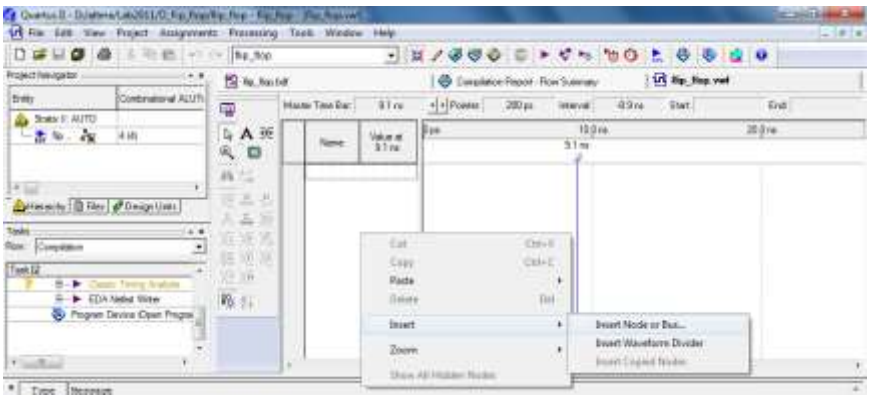


Рис. 6.8. Визначення сигналів для симуляції

Крок 13. З'явилося діалогове вікно. У ньому ви можете набрати ім'я виходу сигналу або елемента. Повне ім'я сигналу може бути дуже довгим і не завжди зрозумілим. Тому краще скористатися пошуком сигналу в проекті. Натискаємо кнопку Node Finder, див. рис. 6.9.

Рис. 6.9. Ім'я виходу сигналу або елемента

Крок 14. З'являється діалогове вікно для пошуку сигналів в проекті Node Finder. У полі Look вказано ім'я нашого проекту - в ньому буде пошук сигналу. Тут нічого змінювати не потрібно. У полі Named можна задати частину імені сигналу або проводу. Наприклад, ви шукайте в проекті вихід модуля з ім'ям data, але в проекті кілька модулів з виходами / входами з таким ім'ям. Тоді для пошуку в цьому полі Named вкажіть * data *. Потім натисните кнопку List (провести пошук) і в таблиці Nodes Found буде показаний список всіх модулів містять такий сигнал. Якщо вказати просто *, то це значить, що нас цікавлять взагалі всі сигнали, не залежно від імені. Ще важливе поле - Filter. Тут зі списку можна вибрати тип сигналів для пошуку. Ми вибираємо зараз Pins: All - тобто нас цікавлять фізичні входи і виходи мікросхеми. Натискаємо кнопку List і бачимо список наших входів і виходів, див. рис. 6.10.

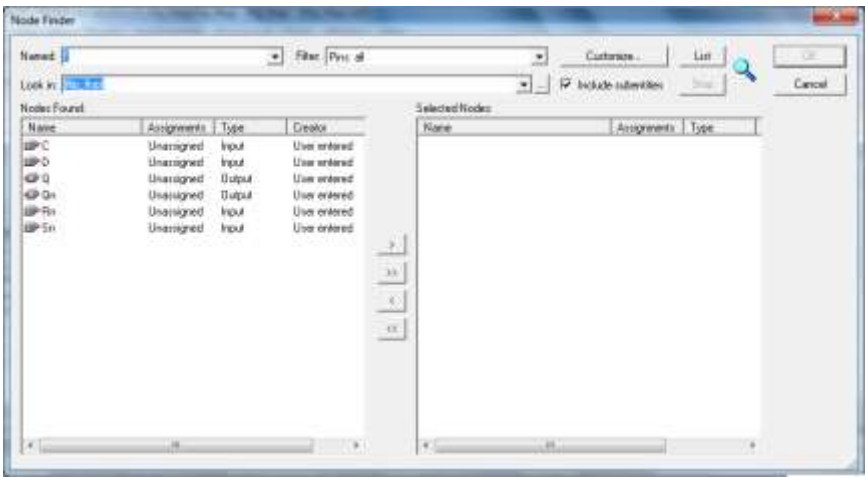


Рис. 6.10. Вікно для пошуку сигналів в проекті

Вибираємо всі потрібні нам сигнали зліва в таблиці знайдених сигналів Nodes Found і переносимо їх вправо в таблицю обраних сигналів Selected Nodes. Натискаємо кнопку OK.

Ось у нашому файлі симуляції потрібні нам сигнали, див. рис. 6.11.

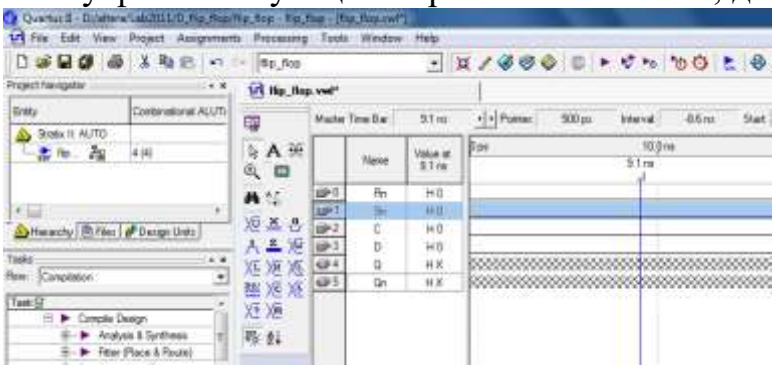
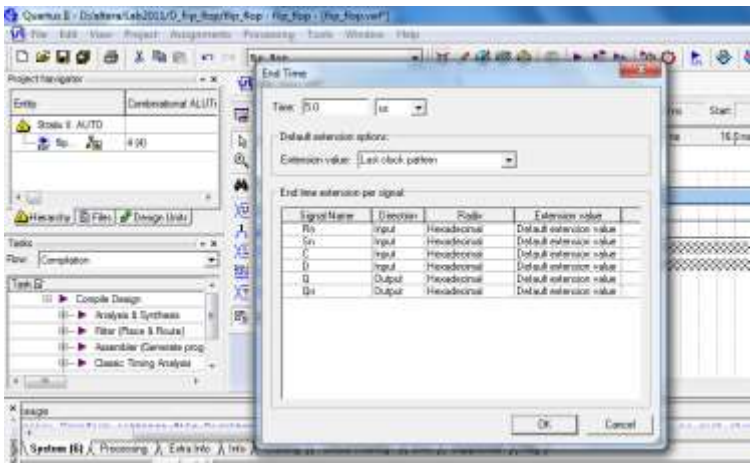


Рис. 6.11. Сигнали для симуляції

Крок 15. Перш ніж редагувати самі сигнали потрібно задати тимчасову сітку (це для зручності малювання). Для цього зайдіть в пункт меню Edit \ Grid Size. Ще потрібно задати тривалість симуляції. Для цього зайдіть в пункт меню Edit \ End Time, див. рис. 6.12.

а)



б)

Рис. 6.12. Завдання тривалості симуляції

Крок 16. Редагувати вхідні сигнали проекту можна за допомогою кнопок панелі інструментів. Наприклад, вибираємо сигнал R_n і натискаємо кнопку Forcing High (1). Таким чином, вхідний сигнал R_n стає логічною одиницею на всьому відрізку часу. Робимо те ж саме і з сигналом S_n. Логічна одиниця на цих сигналах робить їх неактивними для D-тригера, який ми збираємося редагувати, див. рис. 6.13.

Рис. 6.13. Панель інструментів для редагування вхідних сигналів проекту

Крок 17. Вибираємо сигнал C і натискаємо на панелі інструментів кнопку Overwrite Clock. Зараз у цьому діалоговому вікні ми зможемо задати поведінку сигналу як тактової частоти. Задаємо період тактової частоти 50 нс, див. рис. 6.14.

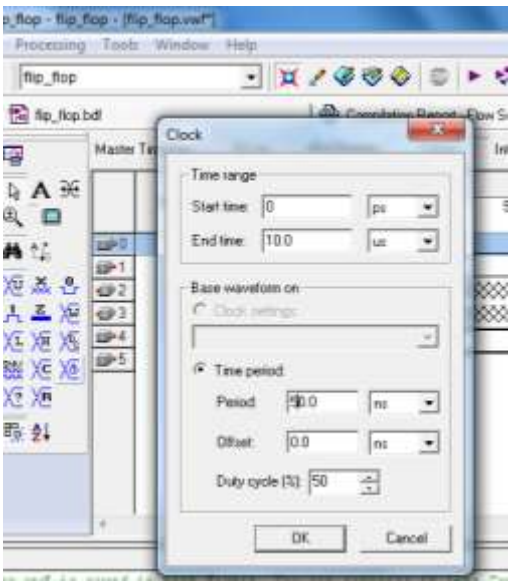


Рис. 6.14. Завдання поведінки сигналу

Крок 18. Намалюємо сигнал D. Для цього натиснемо на панелі інструментів кнопку Waveform Editing Tool - засіб для малювання. Тепер, за допомогою миші, ми можемо просто малювати сигнал див. рис. 6.15.

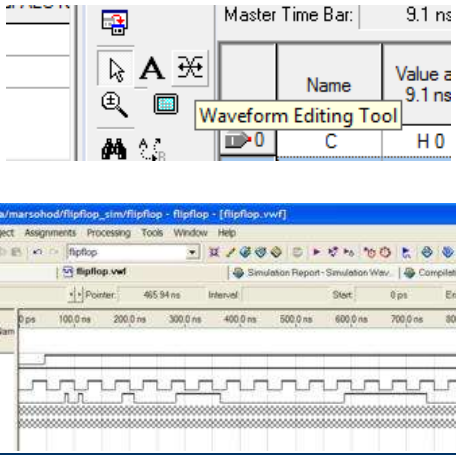


Рис. 6.15. Завдання сигналі

Зверніть увагу, як намальовано сигнал D. Я хочу перевірити його вплив на D-тригер у всіх можливих варіантах. Сигнал D у мене з'являється імпульсом коли C стабільно в одиниці, потім з'являється, коли C стабільно в нулі, потім тримається в одиниці під час спаду сигналу C і, наостанок, D тримається в одиниці, коли йде фронт сигналу C. Чим більше варіантів ви перевірите, тим більша ймовірність, що проект буде працювати правильно.

Крок 19. Потрібно поставити тип симуляції. Зайдіть в пункт меню Assignment \ Settings, див. рис. 6.16.

Рис. 6.16. Завдання типу симуляції

Крок 20. У діалоговому вікні Settings нас зараз цікавить розділ Simulator Settings. Тут потрібно задати тип симуляції - Functional, функціональна (нас зараз цікавить саме логіка роботи, а не тимчасові затримки всередині чіпа). Ну і звичайно задаємо ім'я вхідного файлу симуляції *flipflop.vwf*, натискаємо OK. Ось тепер все готово для симуляції, див. рис. 6.17.

Рис. 6.17. Тип симуляції - Functional

Крок 21. Запускаємо симулятор з пункту меню Processing \ Start Simulation, див. рис. 6.18.

Рис. 6.18. Запуск симуляції проекту

Крок 22. Після успішної симуляції можна розглянути сигнали на виході тригера Q або інверсного йому Q_n. Зміна вхідного сигналу D абсолютно не важливо, важливо тільки його стан у момент фронту C. Саме фронт сигналу C записує вхідний сигнал D на вихід Q, див. рис. 6.19.

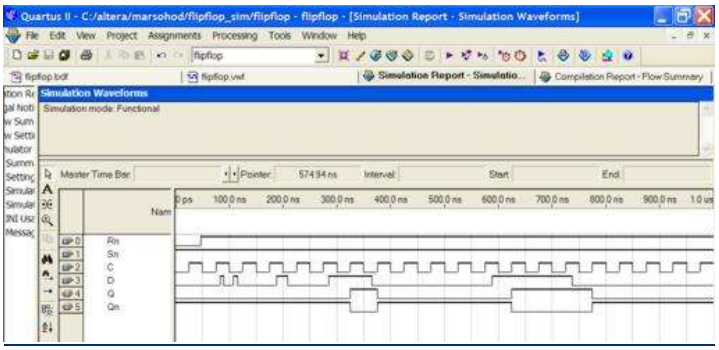


Рис. 6.19. Зміна сигналів проекту

Крок 23. Додамо ще внутрішні сигнали для симуляції. Ще раз запустимо діалог пошуку сигналів. У вікні пошуку встановимо фільтр Design Entity (all names) і натиснемо кнопку пошуку List. Тепер ми бачимо ще і внутрішні сигнали названі A0, A1, B0, B1, C0, C1 - це імена nand3 елементів **I-НЕ** і значить це імена їх вихідних сигналів. Звичайно, досить часто, при симулювання інших проектів ви можете зіткнутися з фактом, що не можна знайти якийсь сигнал, він пропав. Це пояснюється дуже легко. Компілятор синтезатор зробив оптимізацію і просто викинув сигнал як несуттєвий. Компілятор при оптимізації може викидати не тільки комбінаторну логіку, але і регістри! Що ж робити, якщо цікавить сигналу не виявилося, якщо він викинутий оптимізатором? Є два варіанти: або обмежитися тим, що є, або встановлювати в проект віртуальні Піни (Virtual Pins). При установці віртуального пина сигнал завжди буде існувати в проекті, і ви зможете дивитися симулятором сигнали в точці, де він встановлений. Установка віртуального пина загрожує збільшенням кількості використовуваної логіки, адже частина функцій могла б бути оптимізована, а ви не дали це зробити установкою віртуального пина!

У нашому прикладі D-тригера всі елементи залишилися місці. Їх не можна оптимізувати, адже вони з'єднані складними зворотними зв'язками, див. рис. 6.20.

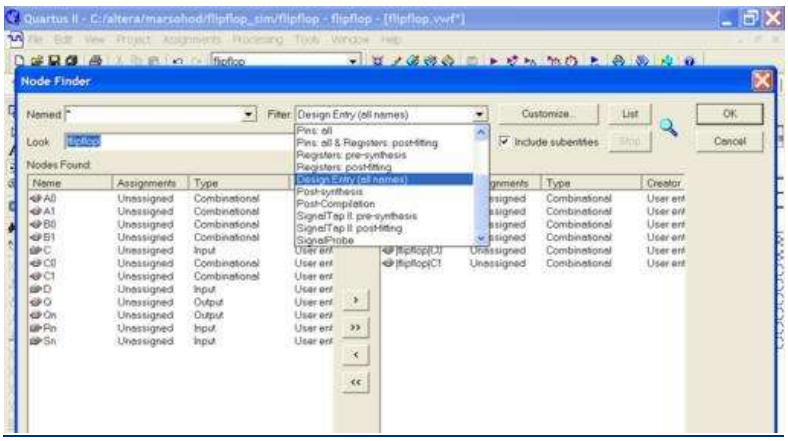


Рис. 6.20. Зміна сигналів проекту

Крок 24. Ось поведінку внутрішніх сигналів D-тригера. Тепер можна спробувати розібратися в принципі роботи D-тригера. Сигнали A0 і A1 - це виходи верхнього RS-тригера першого ступеня. Сигнали B0 та B1 - це виходи нижнього RS-тригера першого ступеня. Сигнали A1 і B0 діють як S і R для RS-тригера другого ступеня. З симуляції видно, що в залежності від поточного стану RS-тригерів першого ступеня в момент приходу фронту сигналу C виходи цих тригерів встановлюються або у 0,1,0,1 або в 1,0,1,0 (в порядку A0, A1, B0, B1). Одночасно з цим переключенням стан тригерів першого ступеня переписується у другій RS-тригер, див. рис. 6.21.

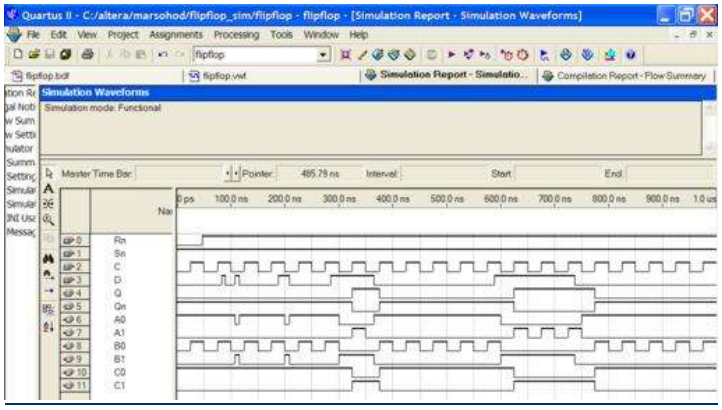


Рис. 6.21. Кінцевий вигляд симуляції проекту

Питання для самоперевірки

1. Етапи симуляції проекту?
2. Як проводиться підключення файлу симуляції до проекту?

Розділ 7. Призначення та можливості системи MATHCAD

7.1. Система розрахунків у Mathcad

Mathcad — система комп'ютерної алгебри з класу систем автоматизованого проектування, орієнтована на підготовку інтерактивних документів з обчисленнями і візуальним супроводженням, відрізняється легкістю використання і застосування для колективної роботи.

Незважаючи на те, що ця програма здебільшого орієнтована на користувачів-непрограмістів, Mathcad також використовується в складніших проектах, щоб візуалізувати результати математичного моделювання, шляхом використання найбільш поширених обчислень і традиційних мов програмування.

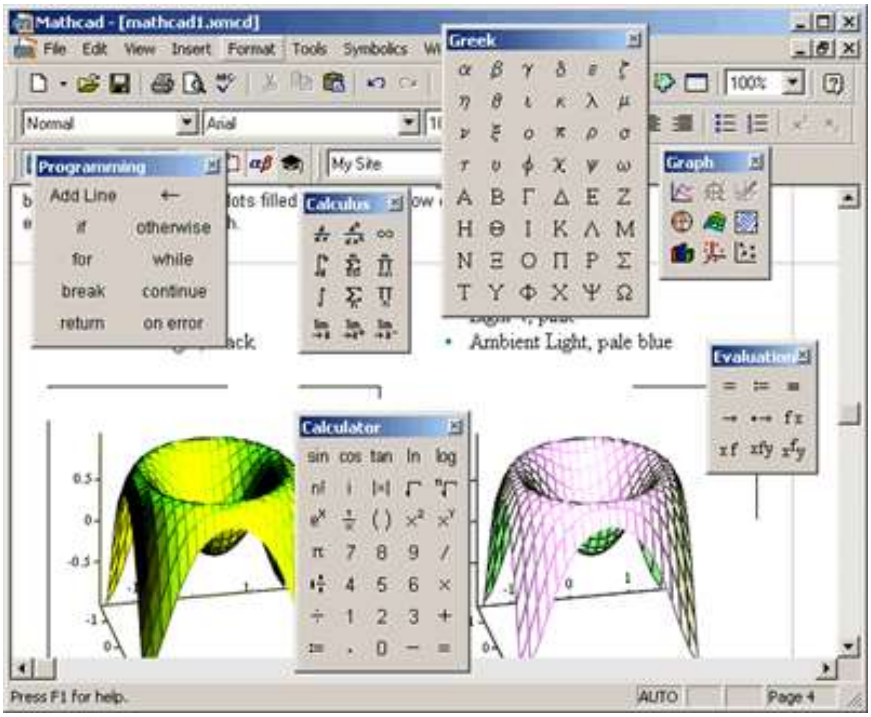


Рис. 7.1 - Головне вікно системи MATHCAD

Основна відмінність Mathcad від аналогічних програм — це графічний, а не текстовий режим вводу виразів. Для набору команд, функцій, формул можна використовувати як клавіатуру, так і кнопки на численних спеціальних панелях інструментів. В будь якому разі — формули будуть мати звичний, аналогічний книжковому, вигляд. Тобто особливої підготовки для набору формул, власне, й не потрібно. Обчислення із введеними формулами здійснюються за бажанням користувача або миттєво, одночасно із набором, або за командою. Звичайні формули обчислюються зліва-направо і зверху вниз (подібно читанню тексту). Будь-які змінні, формули, параметри можна змінювати, спостерігаючи наочно відповідні

зміни результату. Це дає можливість організації справді інтерактивних обчислювальних документів.

Mathcad містить сотні операторів і вбудованих функцій для вирішення різних технічних завдань. Програма дозволяє виконувати чисельні і символьні обчислення, проводити операції з скалярними величинами, векторами і матрицями, автоматично переводити одні одиниці вимірювання в інші.

Серед можливостей Mathcad є:

- ✓ Розв'язання диференціальних рівнянь, в тому числі і чисельними методами.
- ✓ Побудова двовимірних і тривимірних графіків (в різних системах координат, контурні, векторні тощо).
- ✓ Використання грецького алфавіту (верхній і нижній регістр) як в тексті, так і у рівняннях.
- ✓ Символьні обчислення.
- ✓ Операції з векторами і матрицями.
- ✓ Символьне розв'язання систем рівнянь.
- ✓ Згладжування кривих.
- ✓ Виконання підпрограм.
- ✓ Знаходження коренів функцій і поліномів.
- ✓ Статистичні функції і розподіли ймовірностей.
- ✓ Пошук власних значень і власних векторів.
- ✓ Обчислення з розмірностями.

За допомогою Mathcad інженери можуть документувати всі обчислення в процесі їх проведення.

7.2. Математичні вирази в MathCAD

MathCAD працює з *документами*. З погляду користувача, документ - це чистий аркуш паперу, на якому можна розміщати блоки трьох основних типів: математичні вирази, текстові фрагменти і графічні області.

Розташування нетекстових блоків у документі має принципове значення – *зліва направо і зверху вниз.*

До основних елементів математичних виразів MathCAD відносяться *типи даних, оператори, функції* і *керуючі структури.*

Оператори - елементи MathCAD, за допомогою яких можна створювати математичні вирази. До них, наприклад, відносяться символи

арифметичних операцій, знаки обчислення сум, добутків, похідної, інтегралу і т.д. Оператор визначає:

1. дію, що повинна виконуватися при наявності тих чи інших значень операндів;
2. скільки, де і які операнди повинні бути введені в оператор.

Операнд – число чи вираз, на яке діє оператор. Наприклад, у виразі **5! + 3** число **3** і вираз **5!** – операнди оператору **+** (плюс), а число **5** операнд оператору факторіал (!). Після вказівки *операндів* оператори стають блоками, що виконуються у документі. У Додатку 2 даного посібника наведено список операторів, що найбільш часто використовуються.

До *типів даних* відносяться числові константи, звичайні і системні змінні, масиви (вектори і матриці) і дані файлового типу.

Константами називають пойменовані об'єкти, що зберігають деякі значення, що не можуть бути змінені. ***Змінні*** є пойменованими об'єктами, що мають деяке значення, що може змінюватися по ходу виконання програми. Тип змінної визначається її значенням; змінні можуть бути числовими, рядковими, символьними і т.д. Імена констант, змінних і інших об'єктів називають *ідентифікаторами*. Ідентифікатори в MathCAD являють собою набір латинських чи грецьких букв і цифр.

У MathCAD міститься невелика група особливих об'єктів, які не можна віднести ні до класу констант, ні до класу змінних, значення яких визначені одразу після запуску програми. Їх вірніше вважати ***системними змінними***, що мають визначені системою початкові значення. Зміну значень системних змінних роблять у вкладці **Вбудовані змінні** діалогового вікна **Math Options** команди **Математика ⇒ Опції.**

Звичайні змінні відрізняються від системних тим, що вони повинні бути попередньо *визначені* користувачем, тобто їм необхідно хоча б один раз *присвоїти значення*. У якості *оператора присвоєння* використовується знак **:=**, тоді як знак = відведений для *виводу значення* чи константи змінної.

Якщо змінній присвоюється початкове значення за допомогою оператора **:=** викликається натисканням клавіші **:** (двокрапка) на клавіатурі, таке присвоєння називається *локальним*. До цього присвоєння змінна не визначена і її не можна використовувати. Однак за допомогою знака **≡** (клавіша **~** на клавіатурі) можна забезпечити *глобальне* присвоєння. MathCAD прочитує весь документ двічі зліва направо і зверху вниз. При першому проході виконуються всі дії, запропоновані глобальним оператором присвоєння (≡), а при другому – виробляються дії,

запропоновані локальним оператором присвоєння (**:=**), і відображаються всі необхідні результати обчислень (=).

Існують також жирний знак рівності **=** (комбінація клавіш **Ctrl** + =), що використовується, наприклад, як оператор наближеної рівності при розв'язку систем рівнянь, і символьний знак рівності → (комбінація клавіш **Ctrl** + **.**).

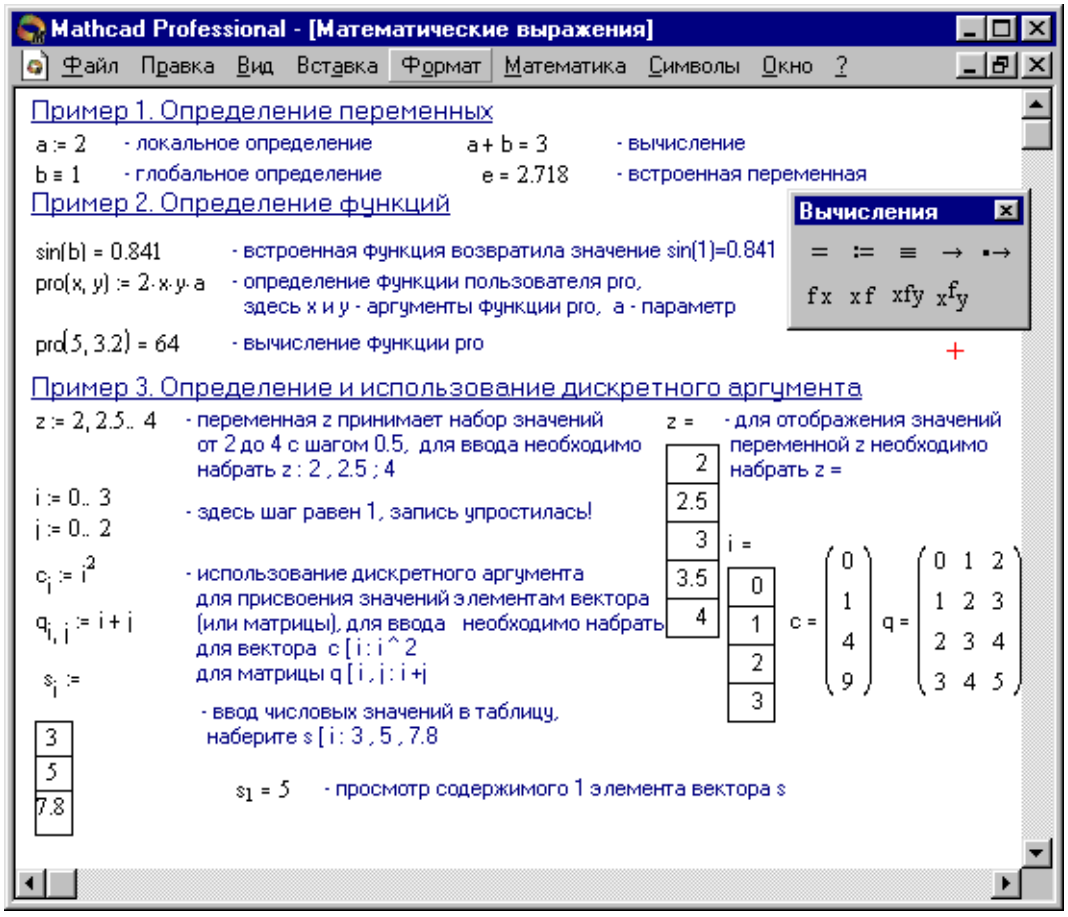


Рис 7.2 - Відображення математичних та текстових виразів в MathCAD

Дискретні аргументи - особливий клас змінних, який у пакеті MathCAD найчастіше заміняє ***керуючі структури***, називані циклами (однак повноцінною така змінна не є). Ці змінні мають ряд фіксованих значень, або цілочисельних (1 спосіб), або у вигляді чисел з визначеним кроком, що міняються від початкового значення до кінцевого (2 спосіб).

1. *Name* := *Nbegin* .. *Nend*,

де *Name* – ім'я змінної, *Nbegin* – її початкове значення, *Nend* – кінцеве значення, **..** – символ, що вказує на зміну змінної в заданих межах (вводиться

клавішею **;**). Якщо *Nbegin* < *Nend*, то крок змінної буде дорівнює +1, інакше –1.

2. *Name* := *Nbegin*, (*Nbegin* + *Step*) .. *Nend*

Тут *Step* – заданий крок зміни змінної (він повинний бути додатнім, якщо *Nbegin* < *Nend*, чи від'ємним в іншому випадку).

Дискретні аргументи значно розширюють можливості MathCAD, дозволяючи виконувати багаторазові обчислення чи цикли з повторними обчисленнями, формувати вектори і матриці.

Масив - сукупність, що має унікальне ім'я, кінцевого числа числових чи символьних елементів, впорядкованих деяким чином і що мають визначені адреси. У пакеті MathCAD використовуються масиви двох найбільш розповсюджених типів:

- одновимірні (вектори);
- двовимірні (матриці).

Порядковий номер елемента, що є його адресою, називається *індексом*. Індекси можуть мати тільки цілочисельні значення. Вони можуть починатися з нуля чи одиниці, у відповідності зі значенням системної змінний **ORIGIN**.

Вектори і матриці можна задавати різними способами:

- за допомогою команди **Вставка ⇒ Матриця**, чи комбінації клавіш **Ctrl + M**, чи щигликом на кнопці  панелі **Матриця,** заповнивши масив порожніх полів для не занадто великих масивів;
- з використанням дискретного аргументу, коли має місце деяка явна залежність для обчислення елементів через їхні індекси.

Вставка текстових фрагментів на лист MathCAD

Текстові фрагменти являють собою куски тексту, що користувач хотів би бачити у своєму документі. Існують два види текстових фрагментів:

- *текстова область* призначена для невеликих шматків тексту - підписів, коментарів і т.п. Вставляється за допомогою команди **Вставка ⇒ Текстова** або комбінації клавіш **Shift + "** (подвійні лапки);
- *текстовий абзац* застосовується в тому випадку, якщо необхідно працювати з абзацами чи сторінками. Вставляється за допомогою комбінації клавіш **Shift + Enter**.

7.3. Обчислення функцій в MathCAD та побудова графічних областей

Функція – вираз, відповідно до якого проводяться деякі обчислення з *аргументами* і визначається його числове значення.

Слід особливо зазначити різницю між *аргументами* і *параметрами* функції. Змінні, зазначені в дужках після імені функції, є її *аргументами* і замінюються при обчисленні функції значеннями з дужок. Змінні в правій частині визначення функції, не зазначені дужках у лівій частині, є *параметрами* і повинні задаватися *до* визначення функції.

Головною ознакою функції є *повернення значення*, тобто функція у відповідь на звернення до неї по імені з вказівкою її аргументів повинна повернути своє значення.

Функції в пакеті MathCAD можуть бути *вбудовані*, тобто завчасно введені розроблювачами, і *визначені користувачем*.

Способи вставки вбудованої функції:

1. Вибрати пункт меню **Вставка ⇒ Функція**.
2. Натиснути комбінацію клавіш **Ctrl + E**.
3. Клацнути на кнопці  .

В середовищі Mathcad фактично немає графіків функцій в математичному розумінні терміну, а є візуалізація даних, що знаходяться у векторах та матрицях (тобто здійснюється побудова як ліній, так і поверхонь по точках з інтерполяцією), хоча користувач може про це й не знати, оскільки у нього є можливість використання безпосередньо функцій однієї або двох змінних для побудови графіків чи поверхонь відповідно.

Графічні області поділяються на три основних типи - двовимірні графіки, тривимірні графіки й імпортовані графічні образи. Двовимірні і тривимірні графіки будуються самим MathCAD на підставі оброблених даних.

Для створення ***декартового графіка***:

1. Встановити візир у порожньому місці робочого документа.
2. Вибрати команду **Вставка ⇒ Графік ⇒ X-У графік,** чи натиснути комбінацію клавіш **Shift + @,** чи клацнути кнопку  панелі **Графіки**. З'явиться шаблон декартового графіка.
3. Введіть у середній мітці під віссю *X* першу незалежну змінну, через кому – другу і так до 10, наприклад *x*1, *x*2, …

4. Введіть у середній мітці ліворуч від вертикальної осі *Y* першу незалежну змінну, через кому – другу і т.д., наприклад *y*1(*x*1), *y*2(*x*2), …, чи відповідні вирази.

5. Клацніть за межами області графіка, щоб почати його побудову.

Тривимірні, чи ***3D-графіки***, відображають функції двох змінних виду *Z*(*X*, *Y*). При побудові тривимірних графіків у ранніх версіях MathCAD поверхню потрібно було визначити математично (Рис. 7.3., спосіб 2). Тепер застосовують функцію MathCAD *CreateMesh.*

CreateMesh(*F* (чи *G*, чи *f*1, *f*2, *f*3), *x*0, *x*1, *y*0, *y*1, *xgrid*, *ygrid*, *fmap*)

Створює сітку на поверхні, визначеною функцією *F*. *x*0, *x*1, *y*0, *y*1 – діапазон зміни змінних, *xgrid*, *ygrid* – розміри сітки змінних, *fmap* – функція відображення. Усі параметри, за винятком *F*, - факультативні. Функція *CreateMesh* за замовчуванням створює сітку на поверхні з діапазоном зміни змінних від –5 до 5 і із сіткою 20×20 точок.

Приклад використання функції *CreateMesh* для побудови 3D-графіків наведений на рисунку 7.3., спосіб 1. На рисунку 7.3. побудована та сама поверхня різними способами, з різним форматуванням, причому зображені поверхні і під ними ті ж поверхні у вигляді контурного графіка. Така побудова здатна додати малюнку велику наочність.

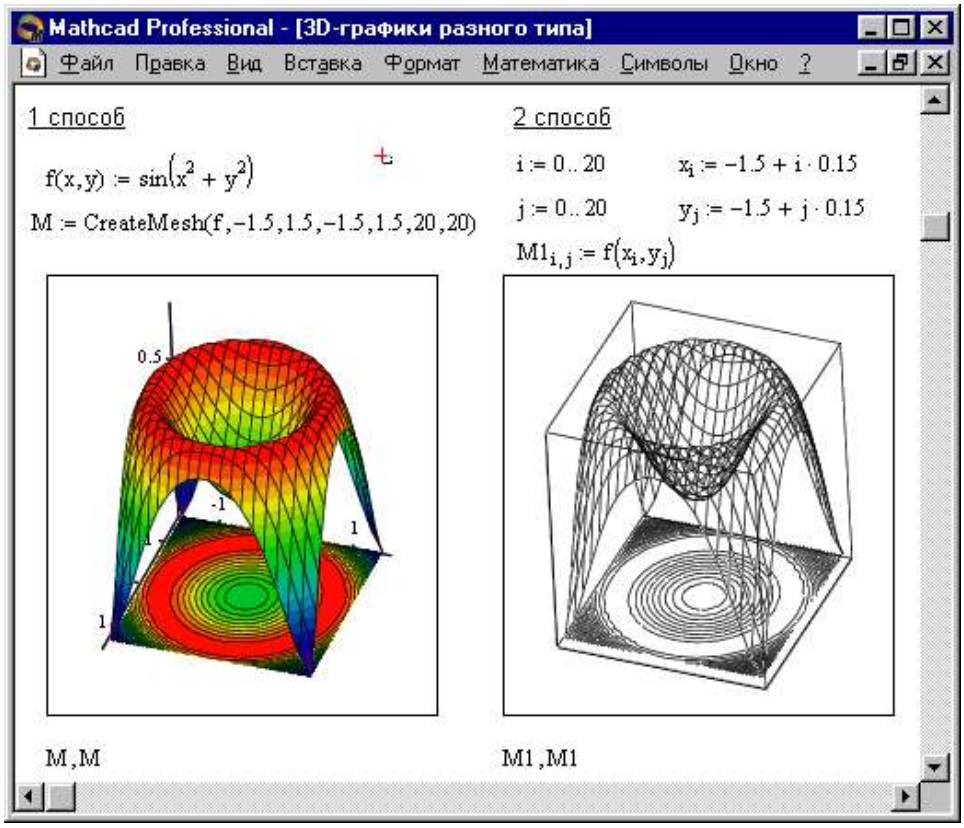


Рис. 7.3 - Приклад побудови на одному малюнку двох 3D-графіків різного типу в MathCAD

Нерідко поверхні і просторові криві представляють у вигляді крапок, чи кружечків або інших фігур. Такий графік створюється операцією **Вставка ⇒ Графік ⇒ 3D Точковий**.

Причому поверхня задається параметрично – за допомогою трьох матриць (*X, Y, Z*) (див. рисунок 7.4., спосіб 2), а не однієї, як у прикладі на рисунку 7.3. Для визначення вихідних даних для такого виду графіків використовується функція *CreateSpace* (див. рисунок 7.4., спосіб 1).

CreateSpace (*F* , *t*0, *t*1, *tgrid*, *fmap*)

Повертає вкладений масив трьох векторів, що представляють *x*-, *y*-, і *z*- координати просторової кривої, визначеною функцією *F*. *t*0 і *t*1 – діапазон зміни змінної, *tgrid* – розмір сітки змінної, *fmap* – функція відображення. Усі параметри, за винятком *F*, - факультативні.

Побудова фігур, що перетинаються

Особливий інтерес являє собою можливість побудови на одному графіку ряду різних фігур чи поверхонь з автоматичним обліком їхнього взаємного перетинання. Для цього треба роздільно задати матриці відповідних поверхонь і після виводу шаблона 3D-графіки перелічити ці матриці під ним з використанням як роздільник коми (див. рис. 7.5.).

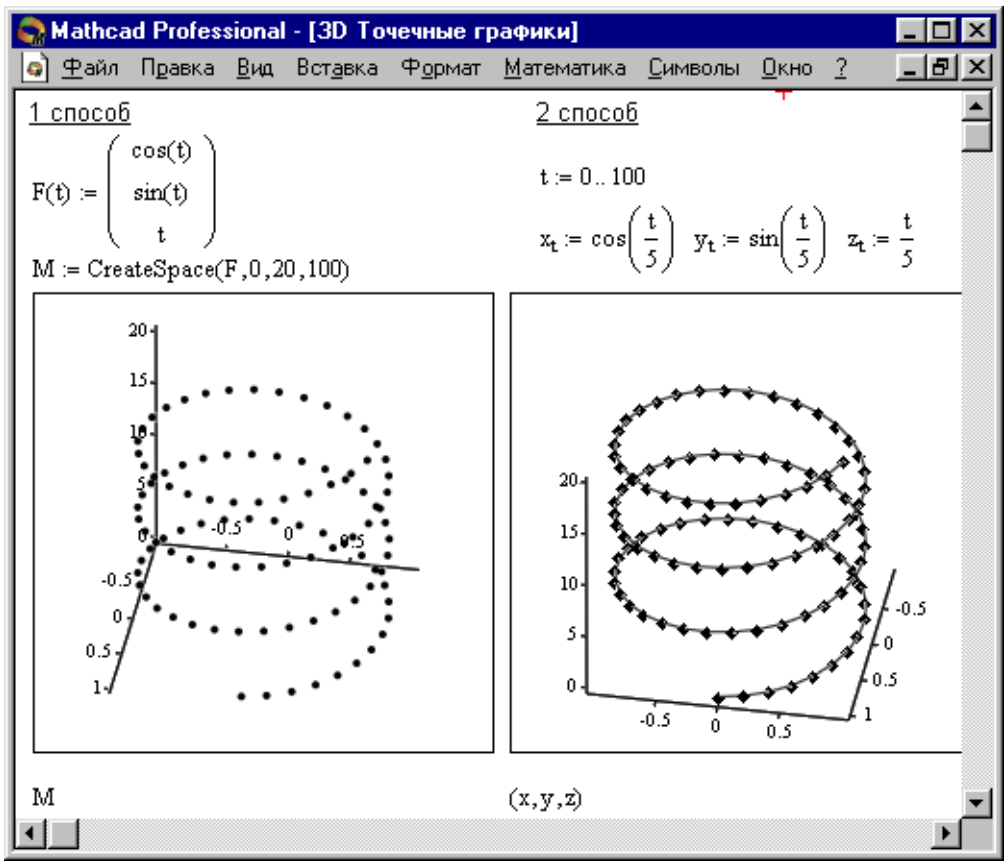


Рис. 7.4 - Побудова 3D Точкових графіків

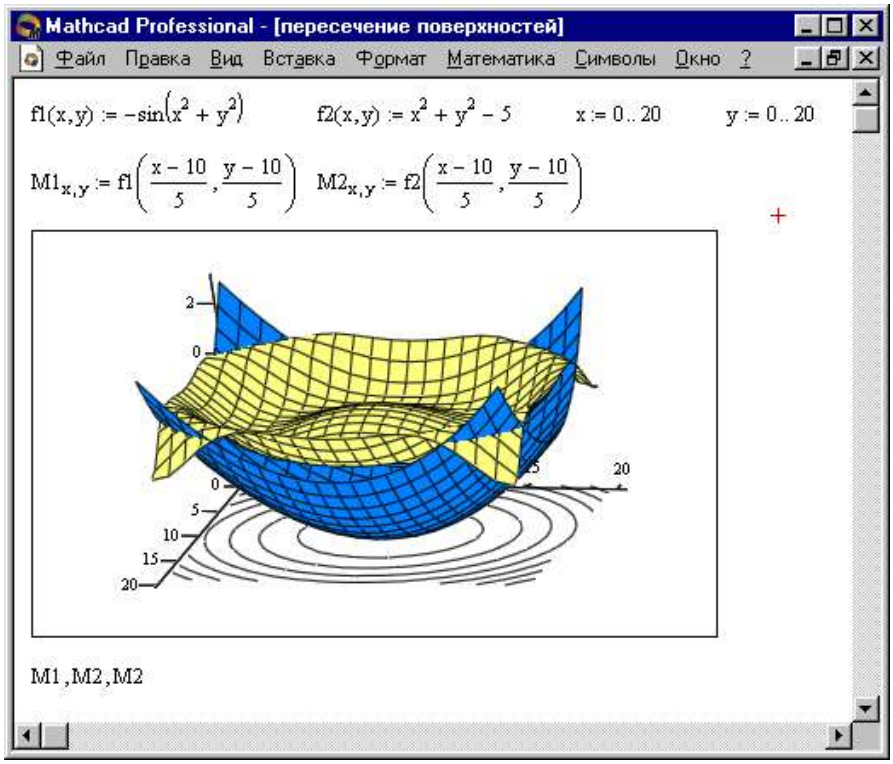


Рис. 7.5 - Побудова двох пересічних поверхонь і одночасно контурного графіка однієї з них

Створення анімаційного кліпу

MathCAD має вбудовану змінну FRAME, чиє єдине призначення - керування *анімаціями*:

- Створіть об'єкт, чий вид залежить від FRAME.
- Переконаєтеся, що встановлено режим автоматичного розрахунку (**Математика ⇒ Автоматичне Обчислення**).
- Виберіть **Вид ⇒ Анімація** для виклику однойменного діалогового вікна.
- Вкладіть в пунктирний прямокутник, що виділяє, частину робочого документа, яку потрібно анімувати.
- Встановіть нижні і верхні границі FRAME (**Від:** і **До:**).
- У поле **Швидкість** введіть значення швидкості відтворення (кадрів/сек).
- Виберіть **Анімація**. Зараз анімація тільки створюється.
- Збережіть анімацію як AVI файл (**Зберегти як**).
- Відтворіть збережену анімацію **Вид ⇒Відтворення.**

Приклад обчислень фунцйії в MathCAD

Цей приклад демонструє обробку виборки малого обсягу. Нехай

$x_0 := 10 \quad x_1 := 10 \quad x_2 := 10 \quad x_3 := 30 \quad x_4 := 20$

$x_5 := 12 \quad x_6 := 10 \quad x_7 := 12 \quad x_8 := 20 \quad x_9 := 10$

$N := \text{length}(x) \quad N = 10$,

де N - обсяг вибірки. За допомогою внутрішньої функції **sort** в масиві Y отримаємо варіаційний ряд для початкової вібірки.

$Y := \text{sort}(x) \quad Y^T =$

	0	1	2	3	4	5
0	10	10	10	10	10	12

Побудуємо статистичний ряд. Виберемо елементи, що не повторюються та запишемо їх в масив X:

$X_0 := 10 \quad X_1 := 12 \quad X_2 := 20 \quad X_3 := 30$

$n := \text{length}(X) \quad n = 4$

n – кількість елементів, що не повторюються.

Обчислимо абсолютні та відностні частоти для всіх елементів X:

$j := 0..n-1 \quad k := 0..N-1$

$m_j := \sum_k \text{if}(x_k = X_j, 1, 0) \qquad p := \frac{m}{N} \qquad \sum_j p_j = 1$

Обчислимо статистичну функцію розподілу та побудуємо її графік:

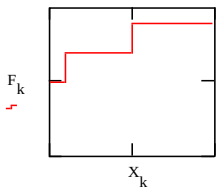
$j := 1..n-1 \qquad F_0 := p_0 \qquad F_j := F_{j-1} + p_j$

$F^T = (0.5 \quad 0.7 \quad 0.9 \quad 1)$

$p^T = (0.5 \quad 0.2 \quad 0.2 \quad 0.1)$

функция распределения

$k := 0..n-1$



Практична робота. Варіаційні та кумулятивні ряди

Тема. Вибіркова сукупність. Варіаційні та кумулятивні ряди. Графіки варіаційних і кумулятивних рядів.

Мета роботи: розглянути, що таке генеральна сукупність, вибірка та варіаційний ряд, кумулятивний ряд. Навчитись складати варіаційні та кумулятивні ряди та будувати їх графіки.

Завдання для самостійної підготовки:

1. Вивчити, що таке математична статистика, статистична і генеральна сукупність, варіація та варіанта.
2. Розібратися для чого використовується вибірка та якого головного правила потрібно дотримуватись у цьому разі.
3. Вивчити методику складання безінтервальних та інтервальних варіаційних і кумулятивних рядів, чим вони відрізняються, коли застосовуються та як зображаються на графіках (варіаційна крива, гістограма, кумулята).

Теоретична частина

Математична статистика як розділ вищої математики, вивчає закономірності властивостей систем із багатьох тіл, об'єктів, суб'єктів, явищ тощо. Аналізує велику кількість дослідних даних, які призначені для даної спеціалізації - біоорганізми, особини, екосистеми тощо.

З метою зменшення матеріальних затрат і часу на дослідження (обстеження) генеральної сукупності (наприклад, фермерських господарств і т.п.) за відповідною ознакою x_i (x_i – екологічні показники, що досліджуються в АПК). З генеральної сукупності N особин вибирають частину n (n<N) репрезентовано, тобто у вибірку попадають особини з різним значенням ознаки.

Складають варіаційний ряд:

$$X_i: X_1, X_2, ..., X_K \qquad (1)$$
$$f_i: f_1, f_2, ..., f_K,$$

де f_i - частота повторюваності ознаки, тобто кількість особин, які мають однакове значення ознак x_i, складають класи. Причому ряд (1) рангують, тобто розміщують за зростаючим порядком

$$X_i : X_1 < X_2, ..., < X_K$$
$$f_i : f_1 < f_2, ..., < f_K \cdot$$

Таким чином, за частотою f_i вибірка ділиться на класи, кількість К яких визначається за формулою Стерджерса (значення k заокруглюється):

$$k = 1 + 3{,}32 \cdot \lg(n) \tag{2}$$

Інтервал $[x_{\min} - x_{\max}]$ значень ознаки ділять на К підінтервалів, довжина яких обчислюється за формулою:

$$\lambda = \frac{x_{\max} - x_{\min}}{k} \cdot \tag{3}$$

Кількість класів збільшується на 1 зміщенням $x_{\min}$ і $x_{\max}$ відповідно вліво і вправо на $\lambda/2$, тобто початкове хп і кінцеве хк значення ознаки буде визначатися так:

$$x_n = x_{\min} - \frac{\lambda}{2} \tag{4}$$

$$x_к = x_{\max} + \frac{\lambda}{2} \tag{5}$$

Визначають початкові xn_i і кінцеві xk_i (i=1,2,…, k+1) значення інтервалів шляхом додавання λ, тобто:

$$xn_1 = xn, \quad xn_i = xn_{(i-1)} + \lambda \quad i = 2,3...k+1$$
$$xk_i = xn_i + (\lambda - 1). \tag{6}$$

Тут кінцеве значення кожного інтервалу зменшують на 1, щоб воно не співпадало з початком кожного наступного інтервалу. Обчислюють середнє арифметичне значення xc_i ознаки кожного інтервалу:

$$x_{c_i} = \frac{x_{n_i} + x_{к_i}}{2}. \tag{7}$$

Відносні частини в долях одиниці або у відсотках обчислюють за формулами:

$$v_i = \frac{f_i}{n}, \tag{8}$$

$$v_i = \frac{f_i}{n} \cdot 100\%. \tag{9}$$

За варіаційним рядом (1) складають кумулятивний (накопичуючий) ряд $(i = 1,2, ..., k+1)$

$$Sf_1 = f_1 \quad (10),$$

$$Sf_i = Sf_{(i-1)} + f_i,\ (i = 2,3,\ldots,k+1), \quad (11)$$

а також відносний кумулятивний ряд:

$$Sv_i = \frac{Sf_i}{n}, \quad (12)$$

$$Sv_i = \frac{Sf_i}{n} \cdot 100\%. \quad (13)$$

За даними розрахунків будують графіки варіаційного (полігон, гістограма) і кумулятивного рядів (кумулята) в декартовій системі координат, відкладаючи на осі абсцис значення x_i, а на осі ординат f_i або v_i - для варіаційного, і Sf_i або Sv_i - для кумулятивного рядів. З графіка кумулятивного ряду (кумуляти) визначають кількість особин Sv_i' у відсотках, значення ознаки яких не перевищує задане x', в межах інтервалу [x', x''] або декількох інтервалів.

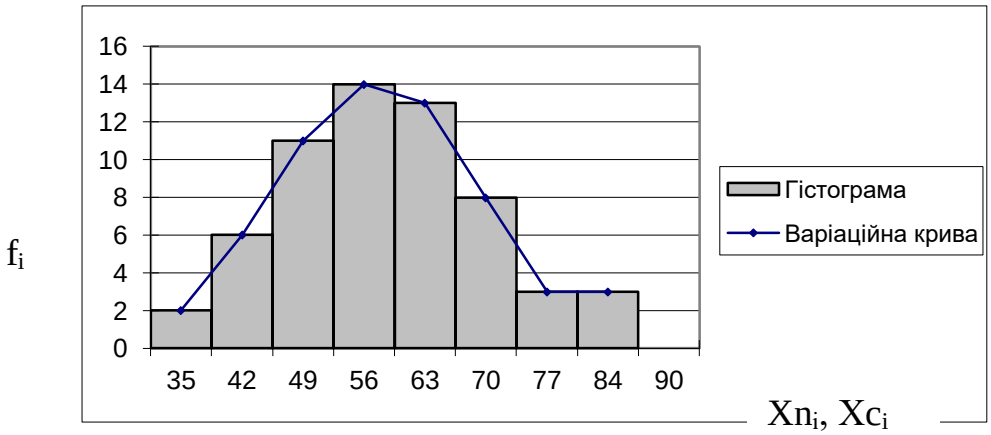


Рис.7.6 - Приклад побудови графічної функції в MathCAD

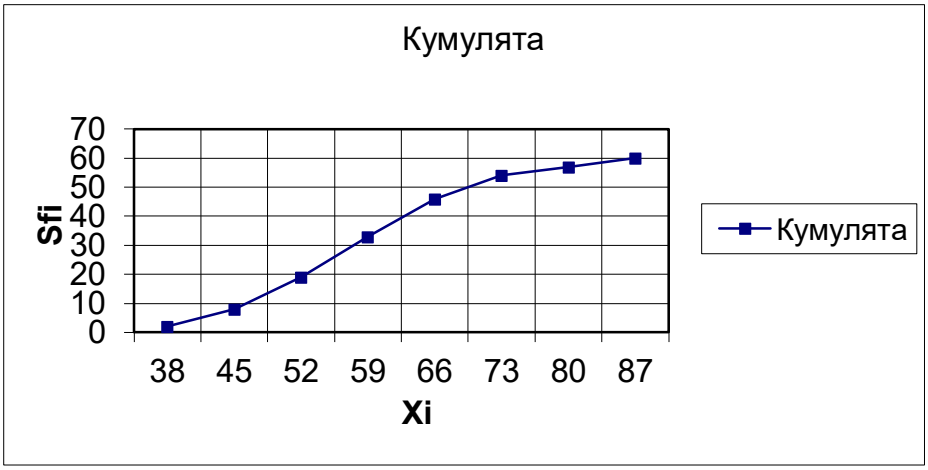


Рис.7.7 - Приклад побудови графічної функції в MathCAD

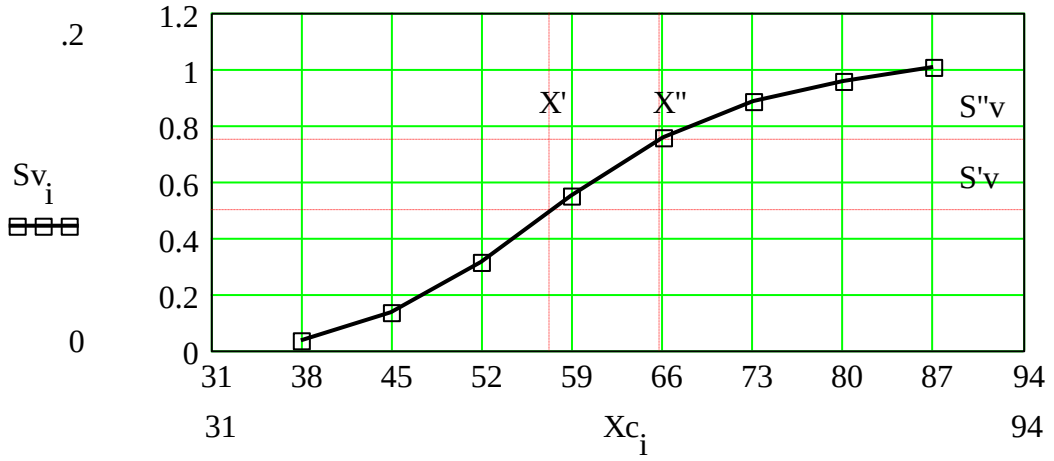


Рис. 7.8 - Приклад побудови графічної функції в MathCAD

Застосування математичного додатка Mathcad *для розв'язання задач математичної статистики.*

Згідно теоретичних відомостей наведених вище, задача складання варіаційного та кумулятивного рядів складається з декількох кроків, а саме:

1) ранжування варіаційного ряду;
2) визначення кількості класів та довжини класового інтервалу;
3) визначення початкових та кінцевих значень для кожного класу;
4) визначення середніх значень класових інтервалів;
5) знаходження частоти попадання значення ознаки;
6) знаходження відносної частоти;
7) знаходження накопиченої частоти;
8) знаходження відносної накопиченої частоти;

Розглянемо реалізацію наведених кроків з використанням математичного додатку Mathcad на конкретному прикладі.

Приклад

Задача. У 62 пробах підземних джерел води вміст хлору (мкг/л) становив: 84,53, 67, 66, 61, 64, 56, 59, 38, 63, 60, 55, 47, 57, 54, 55, 55, 62, 62,60, 60, 54, 61, 60, 68, 68, 44, 80, 82, 76, 71, 69, 69, 50, 86, 86, 76, 74, 72, 76, 53, 40, 43, 74, 64, 62, 49, 68, 78, 57, 49, 64, 69, 44, 87, 52, 56, 66, 63, 54, 71, 74.

Скласти варіаційний та кумулятивний ряди. Побудувати їх графіки.

Розв'язання

Введення даних у середовищі Mathcad для їх статистичної обробки має деякі особливості. Відповідні дані потрібно ввести у вигляді вектора-стовпця або вектора-рядка, що транспонований. Для цього використовуємо меню **Вставка** на панелі інструментів Mathcad.

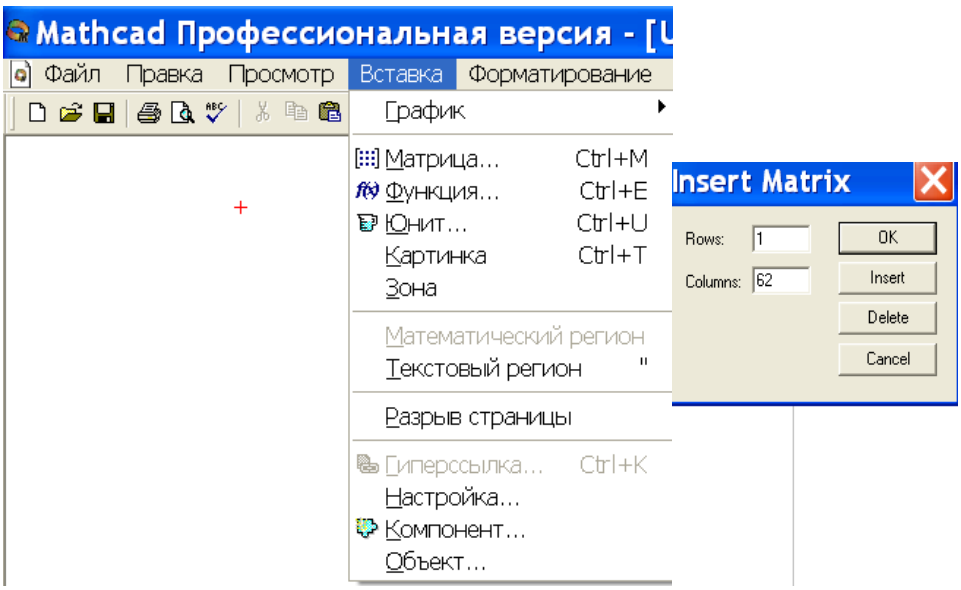


Рис. 7.9 - Меню Вставка на панелі інструментів Mathcad

Проробляють наступні дії:

- обирають місце де буде введено відповідний вектор, наприклад *x*;
- записують символ x та знак присвоювання у вигляді x:=;
- на панелі інструментів у меню **Вставка** вибирають пункт **Матрица**;
- у меню **Матрица** (**Matrix**) обирають кількість рядків та стовпців;
- якщо обирають представлення даних у вигляді вектора-рядка, то кількість рядків дорівнює 1, а стовпців – кількості елементів вибірки;
- якщо обрано представлення у вигляді вектора-стовпця, навпаки;
- у кінці представлення вектора-рядка обов'язково ставлять символ $()^T$, що означає матриця (вектор) транспонована, який викликають натисканням відповідного символу $\mathbf{M}^T$ на панелі **Матрица** (**Matrix**);
- натискають кнопку ОК і отримують відповідний вектор-рядок або вектор-стовпець;
- у отримані комірки введенняять дані.

Ранжування даних виконують за допомогою вбудованих функцій Mathcad. Для того, щоб використати деяку вбудовану функцію виконують наступні дії:

- Обирають місце у документі, куди потрібно вставити функцію;
- Надають ім'я відповідному відсортованому вектору, наприклад X та набирають символ присвоїти (:=);
- Натискають кнопку з надписом ***f(x)*** на стандартній панелі інструментів або у меню **Вставк**а знаходять меню **Функция**;

- У списку **Function Category** (Категория функции) обирають категорію вбудованої функції. В нашому випадку це сортування **Sorting**;
- У списку **Function Name** (Имя функции) обирають функцію **csort** (сортування за елементами стовпця);
- У відповідному діалоговому вікні вказуємо вектор, який потрібно відсортувати x і номер стовпця, що дорівнює 0.
- Даємо команду комп'ютеру застосувати функцію X=.
- Отримаємо відсортовані (записані у порядку зростання дані).

Наступним кроком буде визначення кількості значень.

Застосовують вбудовану функцію **length**. Кількість елементів вибірки позначають буквою ***n***. Виконують наступні дії:

- Обирають місце де буде знаходитись вираз, введенняять *n* присвоїти (*n*:=)
- Натискають кнопку з надписом ***f(x)*** на стандартній панелі інструментів або у меню **Вставка** знаходять меню **Функция;**
- У списку **Function Name** (**Имя функции**) обирають ім'я вбудованої функції **length**;
- У відповідному діалоговому вікні вказуємо вектор, у якому необхідно визначити кількість значень – X;
- Даємо команду комп'ютеру застосувати функцію *n*=.
- Отримаємо кількість значень.

Визначаємо кількість класів за формулою (2).

Пам'ятка Перед тим як обчислити значення деякої величини за формулою у Mathcad, необхідно спочатку присвоїти заданій величині значення у вигляді формули для обчислення, а потім вже обчислити.

Для обчислення кількості класів виконують наступні дії:

- Обирають місце, де буде знаходитись вираз;
- Присвоюють величині к значення за формулою;
- Обчислюють значення k (див. рис.).

$$k := 1 + 3.32 \cdot \log(n)$$

$$k = 6.951$$

Аналогічно за формулою (3) обчислюємо значення довжини класового інтервалу λ.

Збільшуємо кількість класів на 1 і позначаємо остаточну кількість класів символом К.

Округлюємо отриману величину, використовуючи вбудовану функцію **round,** аргументами якої є число, яке потрібно округлити, та кількість знаків після коми для округлення.

$K := k + 1$

$K = 7.951$

$\mathrm{round}(K, 0) = 8$

Визначаємо початкове значення першого класу (xn1)за формулою (4).

Визначаємо кінцеве значення першого класу, враховуючи правило (6).

Отримані значення округлюємо до цілих.

Аналогічно розраховуємо початкові та кінцеві значення наступних семи класів.

$xn1 := xn, \quad xk1 := xn + (\lambda - 1)$

$xn2 := xn1 + \lambda \;\; xk2 := xn2 + (\lambda - 1)$

$xn3 := xn2 + \lambda \;\; xk3 := xn3 + (\lambda - 1)$

$xn4 := xn3 + \lambda \;\; xk4 := xn4 + (\lambda - 1)$

$xn5 := xn4 + \lambda \;\; xk5 := xn5 + (\lambda - 1)$

$xn6 := xn5 + \lambda \;\; xk6 := xn6 + (\lambda - 1)$

$xn7 := xn6 + \lambda \;\; xk7 := xn7 + (\lambda - 1)$

$xn8 := xn7 + \lambda \;\; xk8 := xn8 + \lambda$

Розраховуємо середні значення класових інтервалів за формулою (7).

$$xc1 := \frac{xn1 + xn2}{2} \quad xc5 := \frac{xn5 + xn6}{2}$$

$$xc2 := \frac{xn2 + xn3}{2} \quad xc6 := \frac{xn6 + xn7}{2}$$

$$xc3 := \frac{xn3 + xn4}{2} \quad xc7 := \frac{xn7 + xn8}{2}$$

$$xc4 := \frac{xn4 + xn5}{2} \quad xc8 := \frac{xn8 + xk8}{2}$$

Значення частот знаходимо використовуючи відсортовані дані X, визначаючи кількість ознак, що попадають у відповідний класовий інтервал.

За формулами (8), (10), (11), (12) обчислюємо величини відносних частот, накопичених та відносних накопичених частот.

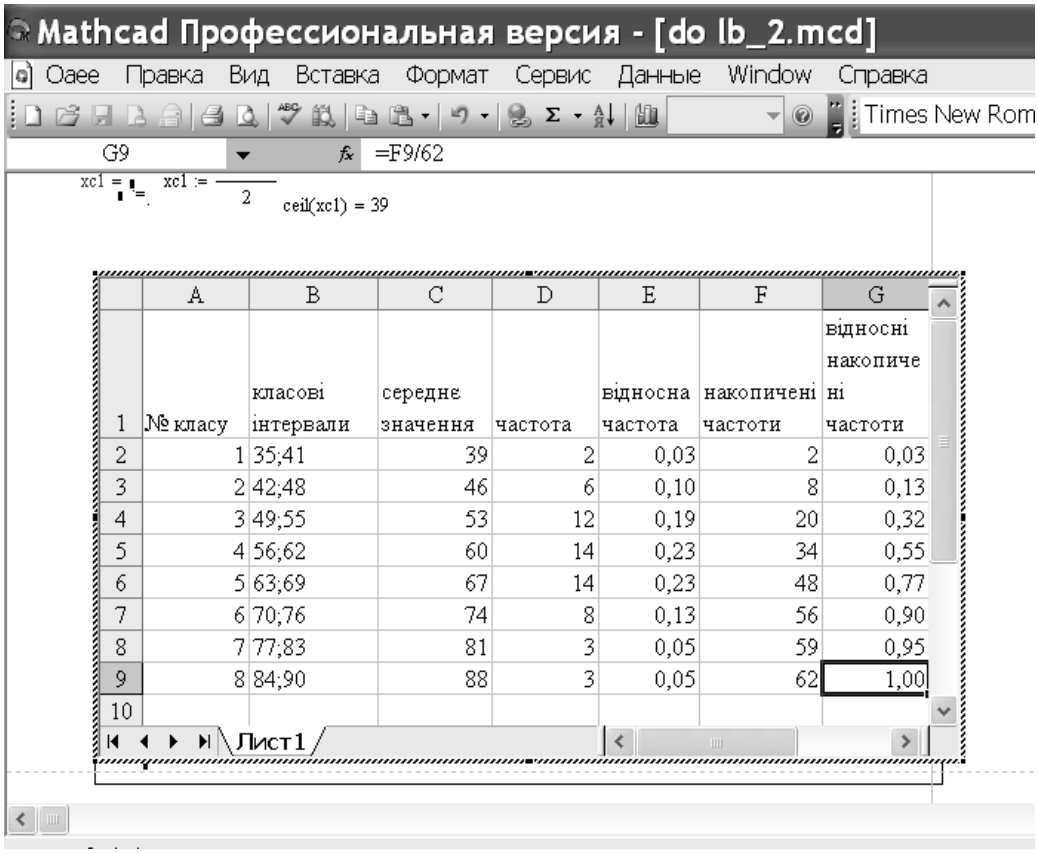
За результатами розрахунків складаємо таблицю, подібну до представленої у лабораторній роботі.

Для того, щоб вставити таблицю у документ Mathcad, необхідно проробити наступні дії.

Обирають місце де буде знаходитись таблиця.

- У меню **Вставк**а на панелі інструментів Mathcad обирають меню **Компонент;**
- У відповідному діалоговому вікні вказують тип таблиці - **Excel**.
- Натискають ОК і отримують таблицю.

В одержаній таблиці робимо наступні стовпці: № класу, класові інтервали, середнє значення класового інтервалу, частоти, відносні частоти, накопичені частоти, відносні накопичені частоти. Стовпці заповнюємо отриманими даними (див. рис. 7.10.)



	A	B	C	D	E	F	G
1	№ класу	класові інтервали	середнє значення	частота	відносна частота	накопичені частоти	відносні накопичені частоти
2	1	35;41	39	2	0,03	2	0,03
3	2	42;48	46	6	0,10	8	0,13
4	3	49;55	53	12	0,19	20	0,32
5	4	56;62	60	14	0,23	34	0,55
6	5	63;69	67	14	0,23	48	0,77
7	6	70;76	74	8	0,13	56	0,90
8	7	77;83	81	3	0,05	59	0,95
9	8	84;90	88	3	0,05	62	1,00
10							

Рис.7.10 - Приклад побудови таблиці в Mathcad

За отриманими даними середнього значення класового інтервалу та частоти будуємо полігон та варіаційну криву.

Перед побудовою графіка введенняимо кількість значень (i), що відповідає кількості класів. Значення цієї величини змінюється в межах від 1 до 8. Введенняимо значення частот та середні значення класових інтервалів.

Для побудови гістограми потрібно:

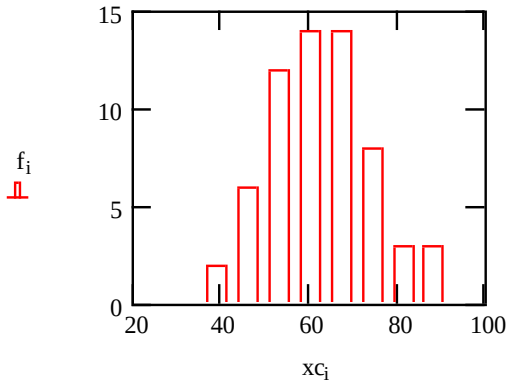
- побудувати графік залежності середнього значення від частоти;
- змінити формат лінії, двічі клацнувши лівою клавішею миші у будь-якій точці графіка;
- відкрити меню форматування графіка;
- на закладці **След**, у стовпчику **Тип** обирають **bar**;
- натискають ОК і отримують гістограму.

Для побудови кумуляти будуємо залежність середнього значення класового інтервалу від накопиченої частоти (див. рис. 7.11.)

$i := 1..8$

$f_i :=$ $xc_i :=$

f_i	xc_i
2	39
6	46
12	53
14	60
14	67
8	74
3	81
3	88



7.11. Приклад побудови графіку залежності в Mathcad

Програма виконання на комп'ютері.

1. Завантажити математичний додаток Mathcad.
2. Виконати наведене нижче завдання згідно зразка.
3. Зберегти створений документ у власній папці.
4. Завершити роботу у Mathcad.

Завдання.

Скласти варіаційний та кумулятивний ряди. Побудувати варіаційну криву, гістограму, кумулятивну криву.

Варіанти завдань.

1 варіант: 58, 55, 56, 56, 63, 63, 61, 61, 55, 62, 61, 69, 69, 45, 81, 83, 85, 54, 68, 67, 62, 57, 70, 70, 51, 87, 77, 75, 73, 77, 54, 54, 41, 44, 75, 65, 63, 44, 69, 79, 58, 50, 60, 39, 64, 61, 56, 48, 65, 65, 70, 45, 88, 53, 57, 67, 64, 55, 72, 48, 77, 72.

2 варіант: 59, 56, 57, 57, 64, 64, 62, 62, 56, 63, 62, 70, 70, 46, 82, 84, 86, 55, 69, 68, 63, 58, 71, 71, 52, 88, 78, 76, 74, 78, 55, 55, 42, 45, 76, 66, 64, 45, 70,

80, 59, 51, 61, 40, 65, 62, 57, 49, 66, 66, 71, 46, 89, 54, 58, 68, 65, 56, 73, 49, 78, 73.

3 варіант: 54, 54, 61, 61, 59, 59, 63, 60, 59, 67, 67, 43, 79, 81, 83, 52, 66, 65, 60, 55,68, 49, 85, 75, 52, 52, 39, 42, 73, 63, 61, 42, 67, 77, 56, 48, 58, 37, 62, 59, 54, 46, 56, 53, 63, 68, 43, 86, 51, 55, 65, 62, 53, 70, 46, 75, 70, 53, 64, 70, 71, 72.

Завдання для самостійної роботи.

Формування знань та вмінь використання математичного програмного середовища "MathCAD" у біометрії і статистичних розрахунках

Теоретична частина

Пакет програм "MathCAD" дає можливість обчислювати числові вирази і значення функцій, які включають дії додавання, віднімання, множення, ділення, піднесення в степінь, логарифмування, диференціювання, інтегрування, знаходження розв'язків алгебраїчних і диференціальних рівнянь та їх систем в аналітичному або числовому вигляді. Такого типу задачі виникають при математичному моделюванні будь-яких процесів, включаючи аграрні взагалі та зооінженерні зокрема.

В основу "MathCAD" покладена традиційна послідовність запису і проведення обчислень за числовими виразами чи формулами.

У середовищі MathCAD є ряд особливостей, які необхідно враховувати при проведенні обчислень:

- робочий документ читається згори вниз, зліва направо.
- символи, які використовуються в розрахунках, крім текстових зон, повинні бути записані при англійській розкладці клавіатури.
- десяткові значення чисел відокремлюються крапкою.
- сувора послідовності розміщення на робочому полі символів та операцій:
  - символи та відповідні їм числові значення, які входять до виразу;
  - область або конкретні значення змінних;
  - вираз у загальному вигляді;
  - результат обчислень;
  - графічна залежність на основі записаного виразу.

При порушенні зазначеного порядку розміщення даних або операцій, червоним кольором виділяються ті символи у виразах, значення яких не визначено і виводиться повідомлення про помилку в тому місці де вона допущена.

Запуск програми MathCAD

- завантажити операційну систему **Windows**;
- клацнути на кнопці “**Пуск**”**;**
- у пункті меню “**Программы”** вибрати підпункт “**MathSoft Apps”**, перейти на каскадне підменю і вибрати пункт “**Mathcad 14**”;
- на екрані з’явиться заставка **MathCAD**, потім вікно **MathCAD.**

Перед початком проведення будь-яких розрахунків необхідно налаштувати в робочому вікні панель "АРИФМЕТИКА**", яка дає можливість виведення на робоче поле знаків математичних дій, символів, операторів, шаблонів графіків тощо. Порядок кодування основних математичних операцій за допомогою клавіатури наведено в додатку 1 Налаштування панелі "Математика"**

- у пункті меню **"Вид"** вибрати підпункт **"Панели инструментов"**;
- перейти на каскадне підменю і відмітити пункт **"Математика"**;
- розкрити необхідні палітри панелі **"Математика"** шляхом одноразового натискання лівої клавіші миші по кнопці відповідної палітри.

Математичне середовище MathCAD використовується для складних математичних обчислень, але і як простий калькулятор, при розрахунках числових виразів.

Обчислення виразу в числовому вигляді

- встановити курсор у будь-яке місце робочого документа;
- з клавіатури або палітри **"Калькулятор"** вивести: число, над яким буде проводитися математична дія, знак дії, число, знак дії і т.д.;
- для отримання результату вивести знак дорівнює.

Приклад:

$$\frac{648+(24.6\cdot 2.73)-\left(\frac{245}{69}\right)\cdot\sqrt{367}}{|174-440|}=2.4333$$

Як правило, в даному математичному середовищі проводять обчислення, використовуючи запис математичних дій не в числових, а в аналітичних виразах, що значно розширює можливості обчислень та скорочує час при розрахунках значень функцій при заданих значеннях аргументу. Для обчислення в такій формі необхідно витримати наведену вище послідовність.

Наприклад, необхідно отримати значення лінійної ($y=a+bx$) та квадратичної ($z=a+bx+cx^2$) функцій для значень змінної x від 1 до 5, та побудувати в одних координатних осях графіки цих функцій. У цьому разі,

функції позначають різними символами. Запис даних та порядок виконання такої побудови в MCAD матиме вигляд:

$x := 1..5 \quad a := 3 \quad b := 5 \quad c := -1$

$y(x) := a + b \cdot x$

$z(x) := a + b \cdot x + c \cdot x^2$

y(x) = z(x) =

y(x)	z(x)
8	7
13	9
18	9
23	7
28	3

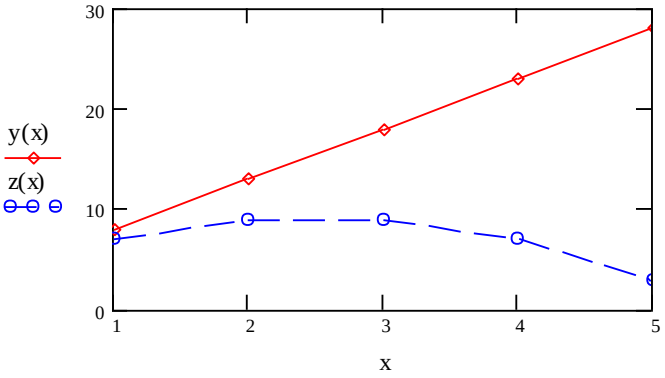


Рис. 7.12 - Графік лінійної та квадратичної функцій в Mathcad

Запис символів та числових значень

- з клавіатури або палітри вивести символ, який входить до виразу і має числове значення;
- поєднанням клавіш "**Shift + :**" або з палітри **Калькулятор** вивести символ "надати значення", який має вигляд (**:=**);
- з клавіатури набрати відповідне числове значення.

Значення змінної, яке входить до формули, може набувати значень у певному діапазоні із зазначеним кроком, або ці значення являють собою набір конкретних чисел.

Задання області змінних

- з клавіатури або палітри вивести символ змінної;
- вивести символ **"надати значення"** (**:=**);
- клавішею "**:**" або з палітри **Калькулятор** вивести шаблон "**диапазон дискретной величины**" **(m ..n)** та записати у вільні зони початкове та кінцеве значення інтервалу;
- для задання кроку змінної, відмінної від одиниці записати, через кому після початкового значення інтервалу наступне значення змінної (наприклад x := 0, 0.5 ..5). В даному випадку крок змінної буде становити 0,5.

Задання конкретних значень змінної

У випадку, коли значення змінної неможливо представити і вигляді діапазону (1...8), а необхідно вказати їх конкретні значення, форма їх запису набуває іншого вигляду. Для цього необхідно:

- з клавіатури вивести літеру “**i**”, з палітри **"Матриця"** вивести символ **“надати значення”**, вивести шаблон “**диапазон дискретной величины**” **(m ..n)** ;
- записати замість початкового значення інтервалу 1, замість кінцевого – число, яке відповідає кількості змінних;
- записати символ змінної, клавішею “[” або з палітри **Калькулятор** створити нижній індекс біля змінної, записати у вільне місце індексу літеру “**i**”;
- вивести символ “**надати значення**”(:=);
- з клавіатури через кому набрати конкретні числові значення змінної (вони будуть оформлені як таблиця).

Запис виразу в загальному вигляді

- записати з клавіатури символ функції;
- у дужках біля символу функції записати символ змінної, якщо вона задана через інтервал значень;
- біля символу функції вивести значок нижнього індексу і записати в індекс літеру “**i**”, якщо змінна задана конкретними значеннями;
- вивести символ “надати значення”;
- записати вираз із символами сталих і змінних у тому вигляді, в якому вони були записані раніше, та встановити між ними знаки відповідних математичних дій**.**

Отримання результату обчислень

- записати символ функції в тому вигляді, як вона була задана;
- вивести знак (=) з клавіатури або з палітри;
- результат обчислень буде оформлений як таблиця.

Побудова графіків двовимірних залежностей

- поєднанням клавіш **“Shift + 2”** або мишкою з палітри “**Графики”** вивести шаблон двомірної системи координат **X-Y График**;
- у вільну зону під віссю абсцис записати символ аргументу;
- у вільну зліва від осі ординат записати символ функції;
- для побудови графіків двох або більше функцій в одній системі координат необхідно записати через кому по осі ординат символи всіх функцій;
- для побудови графіка натиснути клавішу **“Enter”** або клацнути мишкою на вільному місці поза полем графіка.

Форматування графіків двовимірних залежностей

- двічі клацнути мишкою у полі графіка;
- у діалоговому вікні, що з'явиться, в закладці **X-Y Оси (X–Y Axes)** задається тип координатних осей, наявність та щільність координатної сітки тощо;

- у закладці **След (Traces)** задається тип накреслення, колір, товщина будь-якої з ліній графіків функцій, використовуючи списки системи.

Копіювання об'єктів (формул, графіків)

Для зменшення часу запису обчислювального блоку, який містить однотипні формули, графіки чи тексти, доцільно проводити їх копіювання з подальшим незначним форматуванням.

- встановити курсор біля об'єкта, який планується скопіювати;
- виділити об'єкт за допомогою протягування курсору по його діагоналі при натиснутій лівій клавіші миші. Виділений об'єкт обрамляється штриховою лінією;
- скопіювати виділений об'єкт у буфер обміну, клацнувши по кнопці **"Копировать"** на панелі інструментів ();
- перемістити курсор у місце копіювання об'єкта;
- вставити об'єкт з буферу обміну, клацнувши по кнопці **"Вставить"** на панелі інструментів ().

Переміщення об'єктів

Іноді, для забезпечення визначеної послідовності розміщення операцій та символів на робочому полі, або для впорядкування структури записів необхідно перемістити вже створений об'єкт. Для цього:

- виділити об'єкт, клацнувши по ньому. Він буде обрамлений в чорну рамку;
- встановити курсор на лінію рамки, щоб він набув вигляду руки;
- при натиснутій лівій клавіші миші перемістити об'єкт у потрібне місце.

Збереження файлу

- у пункті меню **"Файл" (File)** вибрати підпункт **"Сохранить как" (Save as)** - при первинному збереженні документа або **"Сохранить" (Save)** - при повторному збереженні;
- у діалоговому вікні, що з'явиться, у стрічці **"Сохранить в"** вказати ім'я папки або диску, де буде зберігатися файл, у стрічці **"Имя файла"** записати назву файлу.

Завантаження збереженого файлу

- у пункті меню **"Файл" (File)** вибрати підпункт **"Открыть" (Open)**;
- у діалоговому вікні, що з'явиться, знайти і відкрити потрібну папку, та виділити потрібний файл;
- натиснути кнопку **"Открыть"** в межах діалогового вікна.

Завершення роботи з "MathCAD"

- закрити **"MathCAD"**, клацнувши на відповідній кнопці у правому куті стрічки меню  ;

Вимикання комп'ютера:

- клацнути лівою клавішею на кнопці **"Пуск"**;
- вибрати пункт **"Завершення роботи"**, натиснувши лівою клавішею мишки;
- з'явиться вікно, вибрати **"Вимкнутий комп'ютер"**, натиснути на кнопку **ОК**.

Програма виконання на комп'ютері.

1. Завантажити математичний додаток Mathcad.
2. Виконати наведені нижче завдання.
3. Зберегти створений документ у власній папці.
4. Завершити роботу у Mathcad.

Завдання.

І. Обчислити значення виразу. Номери виразів визначаємо за останньою цифрою власного варіанту. Наприклад, якщо остання цифра вашого варіанту 15, значить, ви обираєте завдання №5 і наступне за ним з непарним номером, а саме №7. Якщо ваш номер 8, то ви обираєте наступне за ним з парним номером - №8 та №0 тощо.

Памятка. А. Проводячи обчислення у середовищі MATHCAD з мішаними дробами (дроби, що мають цілу та дробову частину), необхідно перевести мішаний дріб у неправильний.

Для цього необхідно:

1) знаменник мішаного дробу помножити на його цілу частину і додати до чисельника;

2) отриманий результат записати у чисельнику неправильного дробу;

3) знаменник залишити без зміни.

Наприклад, дріб $2\frac{1}{6}=\frac{2\cdot 6+1}{6}=\frac{13}{6}$.

Б. Дію ділення (:) введенняимо у вигляді (÷) за допомогою сполучення клавіш <Ctrl>+</>, що означає ділення в один рядок.

1. $$\frac{0{,}4+8\left(5-0{,}8\cdot\frac{5}{8}\right)-5:2\frac{1}{2}}{\left(1\frac{7}{8}\cdot 8-\left(8{,}9-2{,}6:\frac{2}{3}\right)\right)\cdot 34\frac{2}{5}}\cdot 90.$$

2. $$\frac{\left(5\frac{4}{45}-4\frac{1}{6}\right):5\frac{8}{15}}{\left(4\frac{2}{3}+0{,}75\right)\cdot 3\frac{9}{13}}\cdot 34\frac{2}{7}+\frac{0{,}3:0{,}01}{70}+\frac{2}{7}.$$

3. $$\frac{\left(1{,}88+2\frac{3}{25}\right)\cdot\frac{3}{16}}{0{,}625-\frac{13}{18}:\frac{26}{9}}+\frac{\left(\frac{0{,}216}{0{,}15}+0{,}56\right):0{,}5}{\left(7{,}7:24\frac{3}{4}+\frac{2}{15}\right)\cdot 4{,}5}.$$

4. $$\left(2:3\frac{1}{5}+\left(3\frac{1}{4}:13\right):\frac{2}{3}+\left(2\frac{5}{18}-\frac{17}{36}\right)\cdot\frac{18}{65}\right)\cdot\frac{1}{3}.$$

II. Побудувати графіки функцій лінійної $y=ax+b$, квадратичної $y=ax^2+bx+c$, гіперболічної $y=\frac{a+x}{b+x}$:

1) для непарних номерів варіантів на проміжку [-1;1] з кроком 0,1;

2) для парних - на проміжку [-2;2] з кроком 0,2.

Коефіцієнти a, b, c отримати з наступної таблиці:

Таблиця 7.1.

	Непарні варіанти	Парні варіанти
a	Номер варіанту	Число, протилежне номеру варіанту
b	Число, протилежне номеру варіанту	Число, обернене номеру варіанту
c	Число, обернене номеру варіанту	Номер варіанту

Відформатувати отримані графіки за наступними вимогами.

Для лінійної залежності:

- Символ – ромби;
- Лінія – пунктирна;
- Колір – синій;
- Тип – лінія:
- Розмір – 2.

Для квадратичної залежності:

- Символ – кружечки;
- Лінія – штрихова;
- Колір – зелений;
- Тип – лінія;
- Розмір – 3.

Для гіперболічної залежності:

- Символ – квадратики;
- Лінія – штрих-пунктирна;
- Колір – коричневий;
- Тип – стовбур;
- Розмір – 4.

III. Побудувати графік функції $y = a\cos bx + c$ для заданих значень змінної:

X	−1	−0,7	−0,6	−0,2	0	2,3	3,4	4,8	5

Відформатувати графік згідно наступних вимог:

– Символ – плюси;

– Лінія – пунктирна;

– Колір – чорний;

– Тип – малюнок

– Розмір – 3.

IV. Розв'язати одну систему лінійних алгебраїчних рівнянь двома способами: методом визначників та з застосуванням обчислювального блоку Given/Find.

1. $\begin{cases} 2x - 3y + z = 7, \\ x + 4y - 2z = -5, \\ 3x - y + 3z = 2. \end{cases}$

2. $\begin{cases} x + 2y + z = 4, \\ 3x - 5y + 3z = 1, \\ 2x + 7y - z = 8. \end{cases}$

3. $\begin{cases} 2x + y = 5, \\ x + 3z = 16, \\ 5y - z = 10. \end{cases}$

4. $\begin{cases} 7x + 2y + 3z = 15, \\ 5x - 3y + 2z = 15, \\ 10x - 11y + 5z = 36. \end{cases}$

Практична робота. Статистичні показники варіації.

Мета роботи: вивчити алгоритм розрахунку показників варіації.

Завдання для самостійної підготовки

Вивчити що таке:

- розмах варіації;
- дисперсія та середнє квадратичне відхилення;
- коефіцієнт варіації;
- нормовані відхилення;

Теоретична частина

З метою аналізу варіаційних рядів, крім середніх величин, введенняять наступні показники:

1) Границі (ліміти): це граничні значення ознаки Xmin (мінімальне) і Xmax (максимальне), які визначають межу зміни варіюючої величини (виробничі показники, параметри характеристики зооінженерії тощо).

2) Розмах варіації - це різниця між граничними значеннями ознаки:

R = Xmax - Xmin . (1)

3) Дисперсія D та середнє квадратичне відхилення σ:

$$D = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} = \frac{\sum_{i=1}^{k} f_i (x_i - \bar{x})^2}{n}, \quad (2)$$

$$\sigma = \sqrt{D} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} = \sqrt{\frac{\sum_{i=1}^{k} f_i (x_i - \bar{x})^2}{n}}, \quad (3)$$

де додавання по *i* в (2), (3) проводиться за номерами водних організмів вибірки і її класів відповідно, n - об'єм вибірки, к - кількість її класів, f_i - частота: кількість особин з однаковим значенням ознаки , $\bar{x}$ - середнє арифметичне значення.

Розкривши квадрат різниці двох чисел, отримаємо:

$$D = \frac{\sum_{i=1}^{n}\left(x_i^2 - 2 \cdot x_i \cdot \bar{x} + \bar{x}^2\right)}{n} = \frac{\sum_{i=1}^{n} x_i^2}{n} - 2 \cdot \bar{x} \cdot \frac{\sum_{i=1}^{n} x_i}{n} + \frac{n \cdot \bar{x}^2}{n} = \quad (4)$$

$$= \overline{x^2} - 2 \cdot \bar{x}^2 + \bar{x}^2 = \overline{x^2} - \bar{x}^2,$$

$$\sigma = \sqrt{\overline{x^2} - \bar{x}^2} \quad , \quad (5)$$

$$\overline{x^2} = \frac{\sum_{i=1}^{n} x_i^2}{n} \quad , \qquad (5')$$

де $\overline{x^2}$ – середнє арифметичне квадрата x_i^2 ознаки, $\bar{x}$ – середнє арифметичне (див. попередню практичну роботу).

Розрахунки за формулами (2), (3) і (4), (5) повинні співпадати, що служить перевіркою вірності цих обчислень.

4) Коефіцієнт варіації - це відносне значення середнього квадратичного відхилення:

$$V = \frac{\sigma}{\bar{x}} \ . \qquad (6)$$

5) Парціальне або нормоване відхилення - це відношення відхилення і-того значення ознаки від його середнього $\left(x_i - \bar{x}\right)$ до середнього квадратичного відхилення σ:

$$t_i = \frac{x_i - \bar{x}}{\sigma} \qquad (7)$$

Застосування математичного додатку Mathcad *для знаходження показників варіації*

Визначення показників варіації за формулами.

Знаходженню показників варіації передує створення відповідного варіаційного ряду за результатами дослідних.

Складемо варіаційний ряд, що складається з x_i та f_i.

- ✓ Для визначення розмаху варіації виконують наступне.
  - Визначають межі зміни ознаки, що варіює.

xmax :=

xmin :=

  - Введенняять формулу для обчислення розмаху варіації

R := xmax – xmin

  - Дають команду комп'ютеру зробити розрахунок

R=

- ✓ Для знаходження дисперсії спочатку визначають середнє арифметичне xc (див. практичну роботу 3).

Після визначення середнього арифметичного визначають дисперсію D.

Формула для обчислення дисперсії та середнього квадратичного відхилення у Mathcad має вигляд

$$D := \frac{\sum_i f_i \cdot (x_i - xc)^2}{n}$$

D=

$\sigma := \sqrt{D}$

$\sigma =$ ∎

Аналогічно за формулами (6) та (7), адаптуючи формули для Mathcad, обчислюємо коефіцієнт варіації та значення парціальних відхилень.

Використання вбудованих функцій Mathcad для обчислення показників варіації

Розрахунок середнього арифметичного з використанням вбудованої функції **mean** описано у лабораторній роботі 3.

Примітка. Введення даних потрібно виконати у вигляді вектора-стовпця.

Для введення даних виконують наступні дії.

- обирають місце де буде введено відповідний вектор, наприклад x;
- записують символ x та знак присвоювання у вигляді x:=;
- на панелі інструментів у меню **Вставка** вибирають пункт **Матрица**;
- у меню **Матрица** (**Matrix**) обирають кількість рядків та стовпців;
- для представлення даних у вигляді вектора-стовпця, обирають кількість рядків, що дорівнює кількості елементів вибірки, а стовпців – 1;
- натискають кнопку ОК і отримують відповідний вектор-стовпець;
- у отримані комірки введенняять дані.

Для розрахунку вибіркової дисперсії використовують вбудовану функцію **var(x)**, для розрахунку середнього квадратичного відхилення – функцію **stdev(x)**, для знаходження максимального значення - **max(x)**, мінімального - **min(x)**.

Використання вбудованих функцій для розрахунку показників варіації аналогічно до використання вбудованих функцій у попередній лабораторній роботі.

Загальний план застосування вбудованих функцій у Mathcad.

1. Обирають місце у документі, куди потрібно вставити функцію;
2. Натискають кнопку з надписом ***f(x)*** на стандартній панелі інструментів або у меню **Вставк**а знаходять меню **Функция**;
3. У списку **Function Name** (Имя функции) обирають ім'я вбудованої функції, під яким вона фігурує у Mathcad.
4. У відповідному діалоговому вікні вказуємо відповідний вектор - x.
5. Даємо команду комп’ютеру застосувати функцію натискаємо =
6. Отримаємо необхідне значення.

Програма виконання на комп’ютері.

1. Завантажити математичний додаток Mathcad.

2. Обчислити значення показників варіації за формулами та з використанням вбудованих функцій Mathcad.

3. Завдання з використанням вбудованих функцій необхідно оформити на окремому листі.

4. Зберегти створений документ у власній папці.

5. Завершити роботу у Mathcad.

Варіанти завдань:

1 вар.	x_i	45	49	67	88	94	96
	f_i	2	3	6	7	5	2
2 вар.	x_i	48	52	70	91	97	99
	f_i	1	3	7	9	5	2
3 вар.	x_i	47	51	69	84	96	98
	f_i	3	5	9	7	4	2

Практична робота. Перевірка H_0-гіпотези за допомогою критеріїв Ст'юдента і Фішера.

Тема: Нульова гіпотеза H_0. Перевірка H_0 за допомогою критеріїв Ст'юдента і Фішера

Мета роботи: вивчити, як перевірити нульову гіпотезу, використовуючи t–критерій Стьюдента та F–критерій Фішера

Завдання для самостійної підготовки

1. Розглянути, що таке нульова гіпотеза та в чому вона полягає.
2. Вивчити, як застосовується t-критерій Стьюдента для перевірки H_0.
3. Вивчити, як застосовується F-критерій Фішера для перевірки H_0.
4. Вивчити, що означає число степенів вільності, та як воно визначається для обох критеріїв.

Теоретична частина

За різницею показників вибірок роблять порівняльні оцінки генеральних показників.

Порівняння проводиться за породами водних організмів або в дослідах впливу будь-якого фактора (наприклад включення в підкормку мікроелементів, вітамінів і т. д..) на показники виробництва у порівнянні з контрольними вибірками, для яких вплив відсутній.

Порівняння базується на H_o-гіпотезі, згідно з якою, різниця між показниками генеральних сукупностей дорівнює 0, а можлива різниця між параметрами вибірок має випадковий характер.

Позначимо через $\tilde{x}_0, \tilde{\sigma}_0$ *і* $\tilde{x}_k, \tilde{\sigma}_k$ показники дослідної і контрольної генеральної сукупностей відповідно.

H_0-гіпотеза задовольняється, якщо $\tilde{x}_0 = \tilde{x}_k, \tilde{\sigma}_0 = \tilde{\sigma}_k$ і для вибіркових показників допускається $\tilde{x}_0 \neq \tilde{x}_k, \tilde{\sigma}_0 \neq \tilde{\sigma}_k$. Коли ці нерівності мають системний характер, H_0-гіпотеза відхиляється.

Для перевірки H_0-гіпотези розроблено спеціальні статистичні критерії з рівнями значимості α=1- P:

α_1=5%, α_2=1%, α_3=0.1%.

Одним із таких критеріїв є t-критерій Стьюдента:

$$t_\phi = \frac{|\overline{x_1} - \overline{x_2}|}{Sd}, \qquad (1)$$

де $$S_d = \sqrt{\frac{(n_1 - 1)\cdot \sigma_1^2 + (n_2 - 1)\cdot \sigma_2^2}{n_1 + n_2 - 2} \times \frac{n_1 + n_2}{n_1 \cdot n_2}} \quad . \qquad (2)$$

σ_1 *і* σ_2- середні квадратичні відхилення для дослідної і контрольної вибірок, n_1 і n_2 - їх об'єми.

Фактичне значення t_ϕ порівнюють з стандартним (табличним) t_{st}. За відповідним рівнем значимості α і степенем вільності $m = m_1 + m_2 = n_1 - 1 + n_2 - 1 = n_1 + n_2 - 2$ з таблиці визначать t_{st}. При $t_\phi < t_{st}$ H_0-гіпотеза задовольняється, а при $t_\phi > t_{st}$ відхиляється. Але для достовірності H_0-гіпотезу провіряють на більшому рівні значимості α. t-критерій застосовується для генеральних сукупностей з нормальним законом розподілу із будь-яким об’ємом вибірки або з ненормальним при n>30.

Для перевірки H_0–гіпотези за значеннями середнього (стандартного) квадратичного відхилення використовують F–критерій Фішера:

$$F_\phi = \frac{\sigma_1^2}{\sigma_2^2} \geq 1,$$

тому, що вибирають $\sigma_1 \geq \sigma_2$.

Значення F_ϕ порівнюють з табличними (стандартним) F_{st} за відомим рівнем значимості α і ступенями вільностей $m_1 = n_1 - 1,\ m_2 = n_2 - 1$ вибірок. Якщо $F_\phi < F_{st}$, то H_0 –гіпотеза задовольняється, а при $F_\phi > F_{st}$ - відхиляється.

Використання математичного додатку Mathcad *для перевірки статистичних гіпотез*

Перевірка нульової гіпотези за критерієм Стьюдента.

1. Знаходимо розрахункове значення t критерію Стьюдента за формулою (1), попередньо зробивши обрахунок величини Sd за формулою (2). Значення n1, n2, σ1, σ2 введенняимо на початку або обчислюємо за формулами.

2. Задаємо рівень значущості α:=0,01

3. Критичне (табличне) значення t критерію Стьюдента знаходимо з використанням вбудованих функцій (див.табл.7.2.)

Таблиця 7.2.

Вбудовані функції що використовуються для знаходження критичних значень за критеріями Стьюдента, Пірсона, Фішера.

Критерій	Категорія функції	Вбудована функція, ім'я функції	Аргументи функції		
Стьюдента	Функция распределения	**qt**	ймовірність $1-\frac{\alpha}{2}$, α – рівень надійності (значущості)	Число степенів свободи n-1	
Фішера	Функция распределения	**qF**	Ймовірність 1-α, α – рівень надійності (значущості)	Число степенів свободи I вибірки n1-1	Число степенів свободи II вибірки n2-1
Пірсона	Функция распределения	**qchisq**	Ймовірність 1-α, α – рівень надійності (значущості)	Число степенів свободи	

4. Алгоритм застосування вбудованої функції

- записують tкрит:=
- натискають кнопку з надписом ***f(x)*** на стандартній панелі інструментів або у меню **Вставк**а знаходять меню **Функция**;
- у меню **Категория Функции** обирають **Все;**
- у списку **Function Name** (Имя функции) обирають ім'я вбудованої функції, під яким вона фігурує у Mathcad. В нашому випадку це **qt** тощо;

- у відповідному діалоговому вікні вказуємо аргументи функції – імовірність ($1-\frac{\alpha}{2}$), число степенів свободи (n1+n2-2).
- даємо команду комп'ютеру застосувати функцію натискаємо =.
- отримаємо відповідне розрахункове значення критерію Стьюдента.

5. Порівнюємо критичне та розрахункове значення t критерію Стьюдента. Для цього введенняимо наступне

$$|t| < tкрит = 0 \ (або \ 1)$$

6. Якщо в результаті порівняння отримаємо 0 (неправда), умова не виконується, значить нульова гіпотеза відхиляється.

7. Якщо в результаті порівняння отримаємо 1 (істина), умова виконується, значить нульова гіпотеза задовольняється.

Перевірка H_0 за допомогою критерію Фішера.

1. Знаходимо розрахункове значення F критерію Фішера за формулою

$$Fрозр := \frac{\sigma 1}{\sigma 2}$$

$$Fрозр =$$

Причому, в чисельнику дробу значення більшої дисперсії.

2. Знаходимо критичне (табличне) значення F критерію Фішера з використанням вбудованої функції **qF.**

Алгоритм застосування вбудованої функції

- записують Fкрит:=
- натискають кнопку з надписом ***f(x)*** на стандартній панелі інструментів або у меню **Вставк**а знаходять меню **Функция**;
- у меню **Категория Функции** обирають **Все;**
- у списку **Function Name** (Имя функции) обирають ім'я вбудованої функції, під яким вона фігурує у Mathcad. В нашому випадку **qF** тощо;
- у відповідному діалоговому вікні вказуємо аргументи функції – імовірність ($1-\alpha$), число степенів свободи більшої вибірки (n1-1), число степенів свободи меншої вибірки (n2-1)
- даємо команду комп'ютеру застосувати функцію натискаємо =.
- отримаємо відповідне розрахункове значення критерію Фішера.

5. Порівнюємо критичне та розрахункове значення F критерію Фішера. Для цього введенняимо наступне

$$|Fрозр| < Fкрит = 0 \ (або \ 1)$$

6. Якщо в результаті порівняння отримаємо 0 (неправда), умова не виконується, значить нульова гіпотеза відхиляється.

7. Якщо в результаті порівняння отримаємо 1 (істина), умова виконується, значить нульова гіпотеза задовольняється.

Програма виконання на комп'ютері.

1. Завантажити математичний додаток Mathcad.
2. Перевірити гіпотезу про рівність середніх значень двох незалежних вибірок за критерієм Стьюдента та зробити аналіз одержаних результатів.
3. Перевірити однорідність дисперсій двох вибірок за критерієм Фішера. Проаналізувати отримані результати.
4. Зберегти створений документ у власній папці.
5. Завершити роботу у Mathcad.

Контрольні запитання

1. Що таке нульова гіпотеза, її суть ?
2. Як розрахувати фактичне значення t-критерію Стьюдента ?
3. Які величини потрібно знати та як їх знайти, щоб визначити стандартне значення t-критерію Стьюдента ?
4. Як розрахувати фактичне значення F- критерію Фішера ?
5. Які величини потрібно знати та як їх знайти, щоб визначити стандартне значення F- критерію Фішера ?

Практична робота. Перевірка H_0 за допомогою критерію Пірсона

Тема: Нульова гіпотеза (H_0). Перевірка H_0 за допомогою критерію Пірсона

Мета роботи: опрацювати методику перевірки нульової гіпотези використовуючи критерій Пірсона.

Завдання для самостійної підготовки

1. Розглянути яка різниця у використанні критерію Пірсона та критеріїв Стьюдента і Фішера.
2. Вивчити, як розраховується фактичне значення критерію .
3. Розібратися, які величини входять до формули фактичного значення критерію .
4. Розглянути, які обмеження має даний метод.

Теоретична частина

Критерій Пірсона застосовується при перевірці гіпотез стосовно законів розподілу. Цим критерієм з'ясовується узгодженість дослідних з теоретичними розподілами, наприклад нормальним, і являє собою степінь розбіжностей між ними відносно частот:

$$x_{\phi}^2 = \sum_{i=1}^{k} \frac{(f_i - \tilde{f}_i)^2}{\tilde{f}_i},$$

де f_i і $\tilde{f}_i$ - дослідні та теоретичні частоти ознаки X, сума ведеться за класами варіаційних рядів.

Значення x_{ϕ}^2 порівнюють з табличними x_{st}^2 при відповідному рівні значимості α і степеня вільності m=k-3.

Якщо $x_{\phi}^2 < x_{st}^2$, то розподіл є нормальним, а при $x_{\phi}^2 > x_{st}^2$ H_0 – гіпотеза не підтверджується (спростовується).

Критерій Пірсона використовують при n>50 і $f_k \geq 5$, у протилежному випадку сусідні класи об'єднують.

Використання математичного додатку Mathcad *для перевірки нульової гіпотези за критерієм Пірсона*

Критерій Пірсона дозволяє перевірити гіпотезу про відповідність розподілу частот нормальному розподілу, наприклад, якщо α=0,01 (1%).

Значення частот f_i, $f1_i$, які потрібно перевірити на відповідність нормальному розподілу візьмемо з попередніх практичних робіт.

Згідно вимог – критерій Пірсона можна застосувати для об'ємів вибірки n>50 і $f_k \geq 5$, у протилежному випадку сусідні класи потрібно об'єднати.

№ класу	класові інтервали	середнє значення	частота	відносна частота	накопичені частоти	відносні накопичені частоти
1	35;41	39	2	0,03	2	0,03
2	42;48	46	6	0,10	8	0,13
3	49;55	53	12	0,19	20	0,32
4	56;62	60	14	0,23	34	0,55
5	63;69	67	14	0,23	48	0,77
6	70;76	74	8	0,13	56	0,90
7	77;83	81	3	0,05	59	0,95
8	84;90	88	3	0,05	62	1,00

Рис. 7.13 - Таблиця рохрахунків для перевірки нульової гіпотези за критерієм Пірсона

Значення f_i і $\tilde{f}_i$ для перших двох і останніх трьох класів об'єднаємо. Складемо допоміжну таблицю.

Таблиця 7.4.

Допоміжна таблиця для розрахунку

Емпіричні частоти	Об'єднані емпіричні частоти	Теоретичні частоти (округлені)	Об'єднані теоретичні частоти

2	8	2	8
6		6	
12	12	11	11
14	14	15	15
14	14	14	14
8	14	9	14
3		4	
3		1	

У Mathcad введенню частот передує задання кількості класів утворених об'єднаних частот. В нашому випадку шляхом поєднання утворилось 5 класів. Але це не єдиний спосіб. Можливий варіант об'єднання першого з другим та двох (не трьох) останніх класів. Тоді б кількість класів дорівнювала не 5, а 6.

$k := 1..5$

$f_k :=$ $f1_k :=$

f_k	$f1_k$
8	8
12	11
14	15
14	~~14~~
14	14

За формулою (1) обчислюємо фактичне значення критерію Пірсона.

$$\chi 2_\phi := \sum_k \frac{(f_k - f1_k)^2}{f1_k}$$

$$\chi 2_\phi =$$

Визначаємо число степенів свободи m=k-3. За значеннями m=2 на рівні значущості α=0,01 знаходимо стандартне (критичне, табличне) значення критерію узгодження Пірсона χ^2_{st} за допомогою вбудованої функції qchisq, аргументами якої є імовірність 1-α і число степенів свободи m.

$$\alpha := 0.01 \quad K := 5 \quad m := K - 3$$

$$\chi 2_{st} := qchisq(1 - \alpha, m)$$

$$\chi 2_{st} = 9.21$$

Порівняння стандартного та фактичного значень критерію Пірсона проводимо у вигляді

$$\chi 2_\phi < \chi 2_{st} = 1$$

Якщо в результаті порівняння отримаємо 1 (істина), значить гіпотеза на даному рівні значущості приймається: розподіл частот відповідає нормальному. Якщо ж отримаємо 0 (неправда) гіпотеза відхиляється: розподіл частот не відповідає нормальному.

Програма виконання на комп'ютері.

1. Завантажити математичний додаток Mathcad.

2. Скопіювати на окремий лист результати знаходження емпіричних частот з лабораторної роботи №2.

3. Скопіювати на цей самий робочий лист результати знаходження теоретичних частот з лабораторної роботи №6.

4. Округлити значення теоретичних частот $f1_i$ за допомогою функції **round**, аргументами якої є значення $f1_i$ та кількість знаків після коми – 0.

5. Об'єднати в класи емпіричні та теоретичні частоти згідно правила, описаного у попередньому розділі.

6. Перевірити гіпотезу про відповідність розподілу частот нормальному розподілу за методикою, описаною вище.

7. Проаналізувати отримані результати.

8. Зберегти створений документ у власній папці.

9. Завершити роботу у Mathcad.

Контрольні запитання

1. Для чого використовується критерій узгодження Пірсона?
2. Як розраховувати фактичне значення критерію χ^2?
3. Чому до формули фактичного значення критерію χ^2 входять лише частоти?
4. Які обмеження має даний метод та як їх уникнути?

Практична робота. Коефіцієнти кореляції лінійної регресії та їх зв'язок з параметрами лінійної залежності

Мета роботи: практично оволодіти поняттями коефіцієнтів кореляції лінійної регресії, через які визначаються параметри лінійної залежності між двома змінними ознаками

Завдання для самостійної підготовки

1.Два типи залежностей: кореляційна та функціональна. Навести приклади з області зооінженерії.

2.Коефіцієнт кореляції r та його властивості.

3.Коефіцієнти лінійної залежності R_{yx} і R_{xy} та їх властивості.

4.Виведення еквівалентних формул σ_x, σ_y, r, R_{yx} і R_{xy}.

5.Рівняння лінійної залежності.

6.Зв'язок параметрів a і b лінійної залежності з коефіцієнтами регресії R_{yx} (R_{xy}) і середніми значеннями $\overline{x}$ і $\overline{y}$

7.Побудова графіків дослідної та теоретичної лінійної залежності.

Теоретична частина

Як відомо, формули коефіцієнтів кореляції r та лінійної залежності R_{xy}, R_{yx} мають вигляд:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{n \cdot \sigma_x \cdot \sigma_y}, \quad (1)$$

$$R_{xy} = r \cdot \frac{\sigma_x}{\sigma_y}, \quad (2)$$

$$R_{yx} = r \cdot \frac{\sigma_y}{\sigma_x}, \quad (3)$$

де

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}, \quad (4)$$

$$\bar{y} = \sum_{i=1}^{n} \frac{y_i}{n} \quad (5)$$

- середні арифметичні значення ознак x і y,

$$\sigma_x = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}}, \quad (6)$$

$$\sigma_y = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \bar{y})^2}{n}} \quad (7)$$

- їх середні квадратичні відхилення.

Підставивши вирази (6), (7) в (1)-(3), отримаємо:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \quad (8)$$

$$R_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \quad (9)$$

З формули (8) (9) і (10) випливає, що коефіцієнти r, R_{xy} і R_{yx} можуть приймати значення від -1 до +1 (знаки + і - вказують на зростання і спадання у від х чи х від у).

Лінійна залежність Y від x має вигляд:

$$Y = ax + b, \quad (11)$$

коефіцієнти a і b якого визначаються за формулами:

$$a = R_{yx}, \tag{12}$$

$$b = \bar{y} - R_{yx} \cdot \bar{x}. \tag{13}$$

Для зворотного зв'язку:

$$X = a'y + b', \tag{14}$$

де

$$a' = R_{xy}, \tag{15}$$

$$b' = \bar{x} - R_{xy} \cdot \bar{y}. \tag{16}$$

Формули (13) і (16) випливають із середніх значень виразів (11) і (14).

Вірогідність математичної моделі знаходиться за критерієм Фішера:

$$F_{\Phi} = \frac{\sigma_2^2}{\sigma_1^2} > F_{st}. \tag{19}$$

Якщо виконується нерівність (24), то гіпотеза, що $a = 0$ відхиляється на довірчому рівні (1-α) (α – ймовірність похибки). Отже, математичну модель можна будувати у вигляді лінійної функції.

Використання математичного додатку Mathcad *для знаходження коефіцієнтів кореляції лінійної регресії*

I. Обчислення коефіцієнтів кореляції лінійної регресії за формулами

Початок обчислень у Mathcad, як відомо, передбачає попереднє присвоювання значень величинам, що будуть фігурувати у залежностях.

1. Присвоюють значення індексу i в межах діапазону від 1 до номера останнього елемента вибірки.

2. Задають (присвоюють) кількість елементів вибірки *n*.

3. Присвоюють значення ознак x_i та y_i.

4. Знаходять за формулами (попередньо їх адаптувавши до Mathcad) (4) і (5) значення середніх для x та y. Позначають ці величини xc та yc.

5. Визначають за формулами (6), (7) величини середніх квадратичних відхилень для x та y, враховуючи попередні позначення середніх. Позначають ці величини σx, σy.

6. Визначають коефіцієнт кореляції r за формулою (8).

7. Знаходять Rxy та Ryx за формулами (2), (3).

8. Присвоюють з наступним обчисленням значення:

коефіцієнтам a, b, використовуючи співвідношення (12), (13);

коефіцієнтам a1, b1, використовуючи співвідношення (15), (16).

9. Присвоюють лінійним залежностям відповідного вигляду

$$Y_i := a \cdot x_i + b$$

$$X_i := a1 \cdot x_i + b1$$

10. Будують графіки дослідних і теоретичних залежностей y_i і $У_i$ від x_i, а також графіки залежностей x_i і X_i від y_i.

11. Форматують графіки згідно наведених малюнків.

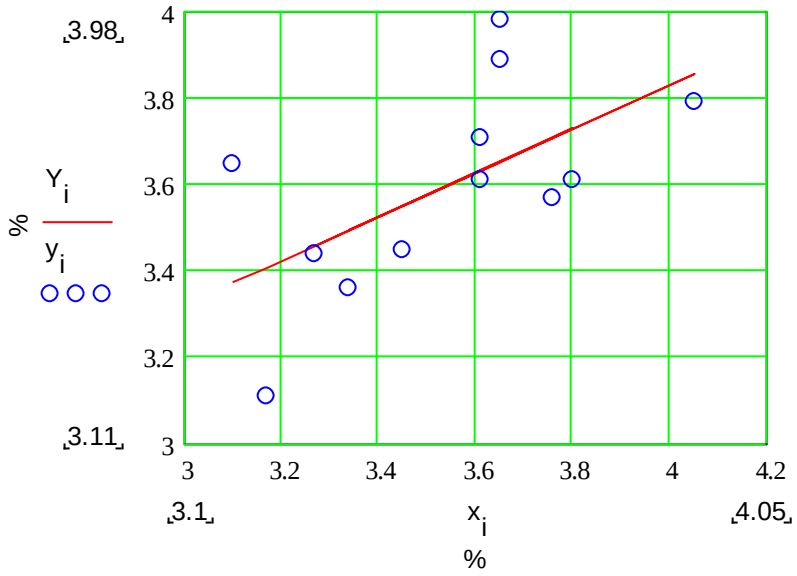


Рис. 7.14 - Графікпорівняння дослідних і теоретичних залежностей

II. Використання вбудованих функцій Mathcad для знаходження коефіцієнтів кореляції лінійної регресії.

1. Задають значення величин x та y в вигляді вектора-стовпця або транспонованого вектора-рядка.

Для цього проробляють наступні дії:

- обирають місце де буде введено відповідний вектор, наприклад *x*;
- записують символ x та знак присвоювання у вигляді x:=;
- на панелі інструментів у меню **Вставка** вибирають пункт **Матрица**;
- у меню **Матрица** (**Matrix**) обирають кількість рядків та стовпців;
- якщо обирають представлення даних у вигляді вектора-рядка, то кількість рядків дорівнює 1, а стовпців – кількості елементів вибірки;
- якщо обрано представлення у вигляді вектора-стовпця, навпаки;
- у кінці представлення вектора-рядка обов'язково ставлять символ $( )^T$, що означає матриця (вектор) транспонована, який викликають натисканням відповідного символу $\mathbf{M}^\mathbf{T}$ на панелі **Матрица** (**Matrix**);
- натискають кнопку ОК і отримують відповідний вектор-рядок або вектор-стовпець;
- у отримані комірки введенняють дані.

2. Визначають за допомогою вбудованих функцій наступні величини, що представлені у таблиці.

Загальний план застосування вбудованих функцій у Mathcad.

- Обирають місце у документі, куди потрібно вставити функцію;
- Натискають кнопку з надписом ***f(x)*** на стандартній панелі інструментів або у меню **Вставк**а знаходять меню **Функция**;
- у меню **Категория Функции** обирають **Все;**
- У списку **Function Name** (Имя функции) обирають ім'я вбудованої функції, під яким вона фігурує у Mathcad.
- У відповідному діалоговому вікні вказуємо відповідний вектор - x .
- Даємо команду комп'ютеру застосувати функцію натискаємо =.
- Отримаємо необхідне значення.

Назва статистичної величини	Загально-прийняте позначення	Відповідна вбудована функція	Рекомендоване позначення у Mathcad
Середнє арифметичне ознаки x	$\bar{x}$	mean(x)	xc
Середнє арифметичне ознаки y	$\bar{y}$	mean(y)	yc
Середнє квадратичне відхилення ознаки x	σ_x	stdev(x)	σx
Середнє квадратичне відхилення ознаки y	σ_y	stdev(y)	σy
Коефіцієнт кореляції	r	corr(x,y)	r
Коефіцієнт лінійної регресії	b	intercept(x,y)	b
Коефіцієнт лінійної регресії	a	slope(x,y)	a
Вектор коефіцієнтів лінійної регресії	(a,b)	line(x,y)	

Програма виконання на комп'ютері.

1. Завантажити математичний додаток Mathcad.

2. Обчислити коефіцієнти кореляції лінійної регресії за формулами, попередньо адаптувавши їх до Mathcad.

3. Побудувати в одній системі координат графіки теоретичних та дослідних залежностей. Від форматувати графіки згідно малюнка.

4. Відкрити новий лист Mathcad. Обчислити коефіцієнти кореляції лінійної регресії, використовуючи вбудовані функції математичного додатка Mathcad.

5. Порівняти результати одержані у пункті 2 та 4.

6. Зберегти створений документ у власній папці.

7. Завершити роботу у Mathcad.

Завдання до практичної роботи

1 варіант.

x_i,%	3,30	3,37	3,96	3,81	3,47	3,81	4,00	3,85	3,65	4,25	3,54	3,85
y_i,%	3,85	3,31	3,77	3,81	3,44	3,91	3,81	4,18	3,65	3,99	3,50	4,09

2 варіант.

x_i, %	3,00	3,07	3,66	3,51	3,17	3,51	3,70	3,55	3,35	3,95	3,24	3,55
y_i,%	3,55	3,01	3,47	3,51	3,34	3,61	3,51	3,88	3,35	3,69	3,20	3,79

3 варіант.

x_i, %	2,90	2,97	3,56	3,41	3,07	3,41	3,60	3,45	3,25	3,85	3,14	3,45
y_i, %	3,45	2,91	3,37	3,41	3,24	3,51	3,41	3,78	3,25	3,59	3,10	3,69

Практична робота. Застосування методу найменших квадратів

Тема: Застосування методу найменших квадратів для обчислення параметрів лінійної залежності Y=ax+b

Мета роботи: опанувати навичками обчислення параметрів лінійної залежності методом найменших квадратів та співставлення із статистичним (дисперсійним) аналізом (див. практичну роботу №11)

Завдання для самостійної підготовки

1.Рівняння лінійної залежності.

2.Обчислення точок перетину графіка прямолінійної залежності з осями координат.

3.Зростаюча та спадна лінійні залежності.

4.Знаки коефіцієнтів *a* і *b* при зростаючій та спадній лінійних залежностях.

5.Сутність методу найменших квадратів.

6.Застосування методу найменших квадратів для моделювання лінійних залежностей між двома ознаками.

Теоретична частина

Для визначення параметрів *a* і *b* лінійної залежності

$$Y = ax + b \tag{1}$$

досліджують вираз *D*:

$$D = \sum_{i=1}^{n} \frac{(y_i - Y_i)^2}{n} = \sum_{i=1}^{n} (\frac{y_i - a \cdot x_i - b)^2}{n}, \tag{2}$$

де Y_i і y_i - теоретичні (обчислені за формулою (1)) і дослідні значення. Вираз (2) являє собою дисперсію відхилень дослідних значень *y* відносно теоретичних

значень *Y*. Коефіцієнти *a* і *b* потрібно підібрати таким чином, щоб вираз *D* (2) мав мінімальне значення. Рівняння прямої (1) отримано в [14] із подібності трикутників, що входить в шкільну програму і тим самим є більш доступним для студентів. З математики відомо, що для цього потрібно прирівняти до нуля частинні похідні:

$$\frac{\partial D}{\partial a} = 0, \quad \frac{\partial D}{\partial b} = 0. \tag{3}$$

Знайшовши похідні, одержимо систему двох рівнянь з двома невідомими *a* і *b*:

$$\begin{cases} b + a \cdot \sum_{i=1}^{n} \frac{x_i}{n} = \sum_{i=1}^{n} \frac{y_i}{n} \\ b \cdot \sum_{i=1}^{n} \frac{x_i}{n} + a \cdot \sum_{i=1}^{n} \frac{x_i}{n} = \sum_{i=1}^{n} x_i \cdot \frac{y_i}{n} \end{cases} \tag{4}$$

або

$$\begin{cases} b + a\bar{x} = \bar{y}, \\ b\bar{x} + a\overline{x^2} = \overline{xy}, \end{cases} \tag{5}$$

де

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \ \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \ \overline{x^2} = \frac{\sum_{i=1}^{n} x_i^2}{n}, \ \overline{xy} = \frac{\sum_{i=1}^{n} x_i y_i}{n}. \tag{6}$$

Система рівнянь (5) називається нормальною і у підручниках записується у вигляді

$$\begin{cases} b \cdot n + a \cdot \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ b \cdot \sum_{i=1}^{n} x_i + a \cdot \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}, \tag{7}$$

яка виводиться з умови мінімуму виразу

$$L = \sum_{i=1}^{n} (y_i - Y_i)^2 \tag{8}$$

або множення системи (4) на n.

Використання системи нормальних рівнянь у вигляді (5), має наочний зміст, оскільки вона виражається через відповідні характеристики варіаційних рядів, а саме: середні арифметичні $\bar{x}$, $\bar{y}$, $\overline{x^2}$ та коефіцієнт коваріації $c = \overline{xy}$.

Розглянемо приклад, наведений в минулій практичній роботі.

Згідно з умовою задачі, $n = 12$.

Знайдемо розв'язки системи (5) методом підстановки. З першого рівняння визначимо *b* і підставимо у друге, після чого визначимо *a*. В результаті одержимо:

$$b = \bar{y} - a\bar{x}\,, \tag{7}$$

$$\bar{x}\cdot\bar{y} - a\bar{x}^2 + a\overline{x^2} = \bar{x}\cdot\bar{y}\,, \tag{8}$$

$$a\left(\overline{x^2} - \bar{x}^2\right) = \overline{xy} - \overline{xy}\,, \tag{9}$$

$$a = \frac{\overline{xy} - \bar{x}\cdot\bar{y}}{\overline{x^2} - \bar{x}^2} = R_{yx}\,, \tag{10}$$

$$b = \bar{y} - \frac{\overline{x\cdot y} - \bar{x}\cdot\bar{y}}{\overline{x^2} - \bar{x}^2}\cdot\bar{x} = \bar{y} - R_{yx}\cdot\bar{x}\,. \tag{11}$$

Таким чином, формула коефіцієнта лінійної регресії R_{yx} (див. вирази (15) попередньої лабораторної роботи 11 і (10)) виводиться методом найменших квадратів.

Використання математичного додатку Mathcad *для знаходження параметрів лінійної залежності методом найменших квадратів*

I. Розв'язання системи рівнянь (7) для визначення параметрів a, b лінійної залежності методом визначників.

Початок обчислень у Mathcad, як відомо, передбачає попереднє присвоювання значень величинам, що будуть фігурувати у залежностях.

1. Присвоюють значення індексу i в межах діапазону від 1 до номера останнього елемента вибірки.

2. Задають (присвоюють) кількість елементів вибірки *n*.

3. Присвоюють значення ознак x_i та y_i.

4. Утворюють матрицю системи Δ, складену з коефіцієнтів біля невідомих a, b.

5. Для задання матриці розміром 2×2 проробляють наступні дії:

- записують символ Δ, яким буде позначено матриця;
- виводять знак "надати значення" або присвоїти, що має вигляд (:=);
- у палітрі **"Матрица"** клацнути по значку $\begin{pmatrix}\cdot&\cdot\\\cdot&\cdot\end{pmatrix}$;
- або сполученням клавіш <**Ctrl**>+<**M**> викликати шаблон матриці;
- у діалоговому вікні, що з'явиться, задають кількість рядків 2 та стовпчиків матриці 2;
- заповнюють виведений шаблон матриці відповідними символами

$$\Delta := \begin{pmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^{\,2} \end{pmatrix}$$

6. Аналогічно складають матриці Δa та Δb, замінюючи перший та другий стовпці головного визначника Δ стовпчиком вільних членів системи (7)

$$\Delta a := \begin{pmatrix} \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i \cdot y_i & \sum_{i=1}^{n} x_i^{\,2} \end{pmatrix} \quad \Delta b := \begin{pmatrix} n & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i \cdot y_i \end{pmatrix}$$

7. Визначають коефіцієнти a, b, попередньо присвоївши їм відповідні значення за формулами $a := \frac{|\Delta a|}{|\Delta|} \quad b := \frac{|\Delta b|}{|\Delta|}$

8. Підставляють значення коефіцієнтів a, b у рівняння прямої $Y_i := a + b \cdot x_i$ та будують графіки емпіричних точок та лінійної залежності.

9. Визначають за допомогою вбудованих функцій наступні величини, що представлені у таблиці.

Загальний план застосування вбудованих функцій у Mathcad.

- Обирають місце у документі, куди потрібно вставити функцію;
- натискають кнопку з надписом f(x) на стандартній панелі інструментів або у меню Вставка знаходять меню Функция;
- у меню Категория Функции обирають Все;
- у списку Function Name (Имя функции) обирають ім'я вбудованої функції, під яким вона фігурує у Mathcad.
- у відповідному діалоговому вікні вказуємо вектор - x .
- надаємо команду комп’ютеру застосувати функцію натискаємо знак =.
- Отримаємо необхідне значення.

Назва статистичної величини	Загально-прийняте позначення	Відповідна вбудована функція	Рекомендоване позначення у Mathcad
Середнє арифметичне ознаки x	$\overline{x}$	mean(x)	xc
Середнє арифметичне ознаки y	$\overline{y}$	mean(y)	yc
Середнє квадратичне відхилення ознаки x	σ_x	stdev(x)	σx
Середнє квадратичне відхилення ознаки y	σ_y	stdev(y)	σy
Коефіцієнт кореляції	r	corr(x,y)	r
Коефіцієнт лінійної регресії	b	intercept(x,y)	b

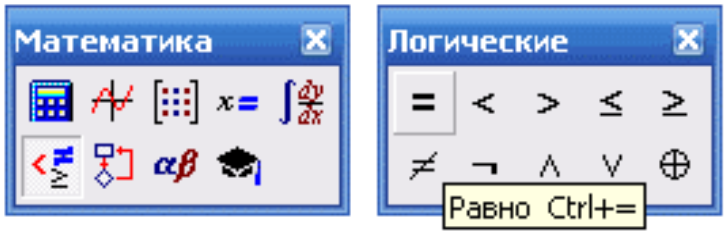
Коефіцієнт лінійної регресії	a	slope(x,y)	a
Вектор коефіцієнтів лінійної регресії	(a,b)	line(x,y)	

II. Розв’язання системи рівнянь (7) для визначення параметрів a, b лінійної залежності з застосуванням обчислювального блоку Given/Find.

1. Проводять підготовчу роботу аналогічно до пункту 1-3 попереднього пункту або копіюють введені у п.1-3 дані на новий лист Mathcad.

2. Потім проробляють наступне:

- невідомим змінним a, b надають (:=) довільного значення, зазвичай це 0 або 1;
- записують ключове слово початку обчислювального блоку **Given** (Дано);
- записують систему рівнянь, але без знака фігурної дужки;
- замість знака дорівнює (=) використовують знак "**булево равенство**", який знаходиться на "**Логические**" у палітрі "**Математика**";



- виводять шаблон матриці в *n* рядків і в один стовпчик, де *n* - число невідомих;
- заповнюють матрицю символами невідомих;
- надають матриці значення у вигляді слова **Find** (Знайти) і в дужках після слова перелічують, через кому, невідомі;
- для знаходження значень невідомих записують їх символи і пишуть знак дорівнює (=).

У **MathCAD** запис розв’язку матиме вигляд:

$a := 1 \qquad b := 1$

Given

$$b \cdot n + a \cdot \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$b \cdot \sum_{i=1}^{n} x_i + a \cdot \sum_{i=1}^{n} x_i^{\,2} = \sum_{i=1}^{n} x_i \cdot y_i$$

$$\begin{pmatrix} a \\ b \end{pmatrix} := \mathrm{Find}(a, b)$$

$a =$

$b =$

Програма виконання на комп’ютері.

1. Завантажити математичний додаток Mathcad.

2. Обчислити параметри a, b лінійної залежності методом визначників, використовуючи завдання до лабораторної роботи №11.

3. Скопіювати дані i, n, x_i та y_i на новий лист Mathcad.

4. Обчислити параметри a, b лінійної залежності з застосуванням обчислювального блоку Given/Find.

5. Порівняти результати одержані у лабораторній роботі №11 та у пункті 2 та 4 лабораторної роботи №12.

6. Зберегти створений документ у власній папці.

7. Завершити роботу у Mathcad.

Контрольні запитання

1.Що таке лінійна залежність і якою формулою вона визначається?

2.Що означають коефіцієнти *a* і *b* лінійної залежності?

3.Які знаки мають *a* і *b* для зростаючої та спадаючої залежностей?

4.У чому полягає суть метода найменших квадратів?

5.З яких умов підбираються *a* і *b* методом найменших квадратів?

6.Вивести систему рівнянь відносно параметрів a і b лінії регресії.

Розділ 8. Системи логічних схем мовою Verilog. Проектування кінцевих автоматів

8.1 Системи логічних схем мовою Verilog

Verilog – мова опису цифрових схем. Базовий тип джерела сигналу в Verilog – це дріт, *wire*. Таким чином, якщо у вас є арифметичний або логічний вираз, ви можете асоціювати результат виразу з іменованим дротом і пізніше використовувати його в інших виразах. Це трохи схоже на змінні, тільки їх (як дроти в схемі) не можна перекомутувати. Значення дроту (*wire*) - це функція того, що приєднане до нього.

Розглянемо на прикладі декларації однобітового дроту в «програмному середовищі» Verilog:

wire a;

Ви можете йому призначити інший сигнал, скажемо сигнал "*b*", от так:

wire b;
assign a = b;

Або ви можете визначити сигнал і зробити призначення йому одночасно в одному виразі:

wire a = b;

У вас можуть бути дроти передаючі декілька біт:

wire [3:0] c; //це чотири дроти

Дроти, які передають декілька біт інформації називають – «шина», іноді «вектор». Посилання на них робляться так само:

wire [3:0] d;
assign з = d; // «підключення» однієї шини до іншої

Кількість дротів в шині визначається будь-якими двома цілими числами розділеними двокрапкою усередині квадратних дужок.

wire [11:4] e; // восьми бітова шина
wire [0:255] f; //256-ти бітова шина

З шини можна вибрати деякі потрібні біти і призначити іншому дроту:

wire g;
assign g = f[2]; //призначити сигналу "g" другий біт шини
//"f"

Крім того, вибираний з шини біт може бути призначено для змінної:

wire [7:0] h;
wire i = f[h]; //призначити сигналу "i" біт номер "h"
//з шини "f"

Ви можете вибрати з сигнальної шини деякий діапазон біт і призначити іншій шині з тією ж кількістю біт:

wire [3:0] j = e[7:4];

Так само, в більшості діалектів Verilog, ви можете визначити масиви сигнальних шин:

wire [7:0] k [0:19]; //масив з двадцяти 8-ми бітових шин

Ще існує інший тип джерела сигналу - регістр: *reg*. Його використовують при поведінковому (*behavioral*) описі схеми. Якщо регістру постійно привласнюється значення комбінаторної (логічної) функції, то він поводиться точно як дріт (*wire*). Якщо ж регістру привласнюється значення в синхронній логіці, наприклад по фронту сигналу тактової частоти, то йому, відповідатиме фізичний **D** - тригер або група **D** - тригерів.

Регістри описуються так само як і дроти:

reg [3:0] m;

reg [0:100] n;

Вони можуть використовуватися так само, як і дроти в правій частині виразів, як операнди:

wire [1:0] p = m[2:1];

Ви можете визначити масив регістрів, які звичайно називають «пам'ять» (RAM):

reg [7:0] q [0:15]; //пам'ять з 16 слів, кожне по 8 біт

Ще один тип джерела сигналу – це *integer*. Він схожий на регістр *reg*, але завжди є 32-х бітовим знаковим типом даних. Наприклад, оголосимо:

integer loop_count;

Verilog дозволяє групувати логіку в блоки. Кожний блок логіки називається «модулем» (*module*). Модулі мають входи і виходи, які поводяться як сигнали *wire*.

При описі модуля спершу перераховують його порти (входи і виходи):

module my_module_name (port_a, port_b, w, y, z);

А потім описують напрям сигналів:

input port_a;
output [6:0] port_b;
input [0:4] w;
inout y; // двонаправлений сигнал,
//звичайно використовується

//тільки для зовнішніх контактів мікросхем

Вихід модуля може бути відразу задекларований як регістр *reg*, а не як дріт *wire*:

```
output [3:0] z;
reg [3:0] z;
```

Ще простіше можна відразу в описі модуля вказати тип і напрям сигналів:

```
module my_module
(
  input wire port_a
  output wire [6:0]port_b
  input wire [0:4]w
  inout wire y
  output reg [3:0]z
);
```

Тепер можна використовувати вхідні сигнали, як дроти *wire*:

```
wire r = w[1];
```

Тепер можна робити постійні призначення виходам, як функції від входів:

```
assign port_b = h[6:0];
```

В кінці опису логіки кожного модуля пишемо слово *endmodule*.

```
module my_module_name (input wire a, input wire b, output wire c);
assign c = a & b;
endmodule
```

Останній тип джерела сигналу, про який ми розглянемо – це постійні сигнали або просто числа:

```
wire [12:0] s = 12; //32-х бітове десяткове число
wire [12:0] z = 13'd12; //13-ти бітове десяткове число
wire [3:0] t  = 4'b0101; //4-х бітове двійкове число
wire [3:0] q  = 8'hA5;   //8-ми бітове шістнадцятирічне число A5
wire [63:0] u = 64'hdeadbeefcafebabe;  //64-х бітове шістнадцятирічне число
```

Якщо точно не визначити розмір числа, то воно приймається за умовчанням 32-х розрядним.

Числа можуть використовуватися у арифметичних і логічних виразах. Наприклад, можна додати 1 до вектора “*aa*”:

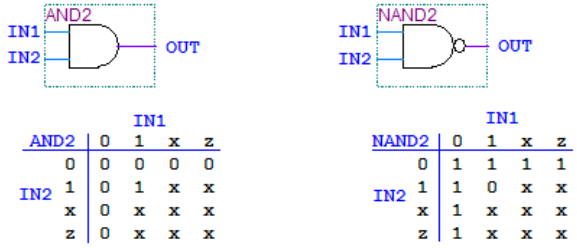
wire [3:0] aa;
wire [3:0] bb;
assign bb = aa + 1;

В проекті, особливо складному, буває багато модулів, з'єднаних між собою. Зазначимо, що в проекті завжди є один модуль самого верхнього рівня (*top level*). Він складається з декількох інших модулів. Ті у свою чергу можуть містити ще модулі і так далі. Не обов'язково, щоб всі модулі були написані на одному язиці опису апаратури. Зовсім навпаки. Досить зручно мати модуль самого верхнього рівня виконаним у вигляді схеми, що складається з модулів більш низького рівня. Ці модулі можуть бути написані різними людьми, на різних язиках (Verilog, VHDL, AHDL, і навіть виконані у вигляді схеми). Насправді – це все справа смаку і можливостей компілятора (синтезатора), а так само вимог замовника.

Розглядатимемо проект з погляду програмування на Verilog.

Отже, усередині тіла будь-якого модуля, можна оголошувати екземпляри інших модулів і потім з'єднати їх один з одним дротами.

Припустимо у нас є наступні примітивні модулі, див. рис. 8.1



AND2	IN1=0	IN1=1	IN1=x	IN1=z
IN2=0	0	0	0	0
IN2=1	0	1	x	x
IN2=x	0	x	x	x
IN2=z	0	x	x	x

NAND2	IN1=0	IN1=1	IN1=x	IN1=z
IN2=0	1	1	1	1
IN2=1	1	0	x	x
IN2=x	1	x	x	x
IN2=z	1	x	x	x

Рис. 8.1. Примітивні модулі «І» та «І-НЕ» у Verilog

Ліворуч зображено логічний елемент **І**. Праворуч зображено логічний елемент **І-НЕ**. Ці елементи так само вказані в таблиці істинності для повноти картини. Ці логічні модулі можна б було описати на Verilog таким чином:

module AND2(output, input IN1, input IN2);
assign = IN1 & IN2;
endmodule

module NAND2(output, input IN1, input IN2);
assign = ~(IN1 & IN2);
endmodule

Проте, ось такий опис їхньої робити не потрібен. Це базові елементи (*gates*) і вони є в стандартних бібліотеках синтезатора.

Ось ще пари важливих логічних елементів, див. рис. 8.2.

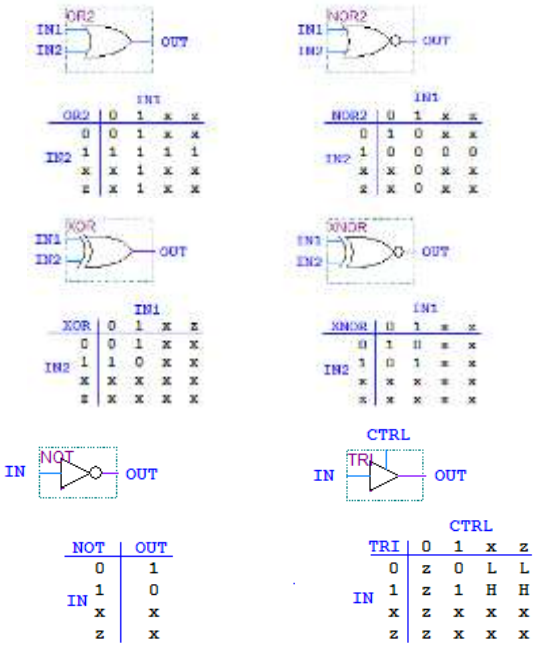


Рис. 8.2. Примітивні модулі «АБО», «АБО-НЕ», та ін. у Verilog

Використовуємо базові елементи в модулі більш високого рівня, з попередньої лабораторної роботи. Зробимо одно бітовий суматор, див. рис. 8.3.

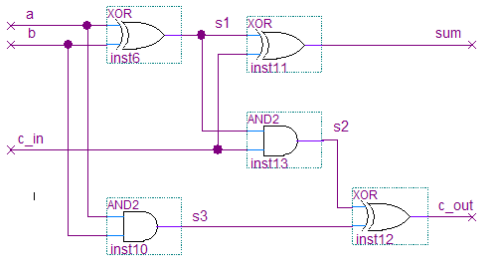


Рис. 8.3. Схема суматора

Цей суматор складає два однобітові числа *a* і *b*. При виконанні складання однобітових чисел може трапитися «переповнювання», тобто результат вже буде двохбітовим (1+1=2 або в двійковому вигляді 1'b1+1'b1=2'b10). Тому у нас є вихідний сигнал перенесення *c_out*. Додатковий вхідний сигнал *c_in* служить для прийому сигналу перенесення від суматорів молодших розрядів, при побудові багатобітових суматорів.

Подивимося, як описати цю схему на Verilog, встановлюючи в тілі модуля екземпляри інших модулів. Цей опис на рівні елементів (*gate-level modelling*). Ми встановимо в наш модуль 3 екземпляри модуля XOR і два екземпляри модуля *AND2*.

```
module adder1(output sum, output c_out, input a, input b, input c_in);
wire s1,s2,s3;

XOR  my_1_xor(  .OUT (s1),  .IN1 (a), .IN2 (b)  );
```

AND2 my_1_and2(.OUT (s3), .IN1 (a), .IN2 (b));
XOR my_2_xor(.OUT (sum), .IN1 (s1), .IN2 (c_in));
AND2 my_2_and2(.OUT (s2), .IN1 (s1), .IN2 (c_in));
XOR my_3_xor(.OUT (c_out), .IN1 (s2), .IN2 (s3));
endmodule

Порядок опису екземпляра модуля такий:
Пишемо назву модуля, екземпляр якої нам потрібен.
Пишемо назву цього екземпляра модуля (за бажанням).
Описуємо підключення сигналів: крапка і потім ім'я сигналу модуля, потім в дужках ім'я дроту, який сюди підключений.

Подальша робота в середовищі Quartus II починається з створенням проекту. Необхідно створити окремий каталог для зберігання файлів проекту. Для створення файлу, який міститиме програму на схему пристрою (після створення проекту) слід виконати команду **New** меню **File**. У діалоговим вікні на вкладці **Devise Design File** слід вибрати тип файлу - **Verilog HDL File** і натиснути **OK,** у вікні, див. рис. 8.4. набираємо код модуля *module adder1*.

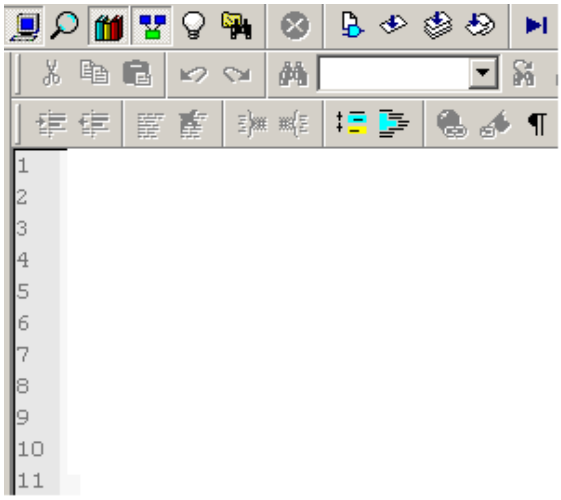


Рис. 8.4. Вікно редактора

Виберіть пункт меню File→Save, вказавши ім'я файлу SM.

Перевіримо роботу суматора у симуляторі.

У вікні **Project Navigator** виберіть закладку **Files** і пункт контекстного меню (права кнопка миші) для файлу *SM.v* **Set as Top-Level Entity** (якщо вікно Project Navigator закрито, то виберіть пункт меню **View→Utility Windows> Project Navigator**).

Виконайте компіляцію проекту, для цього виберіть **Processing→Start Compilation**.

Дочекайтеся повідомлення про успішну компіляцію або повідомлення про помилки, у разі появи помилок відкоректуйте HDL опис і повторіть компіляцію.

Для перевірки пристрою у симуляторі необхідно задати вхідні дії і вказати, які вихідні сигнали контролюватимуться. Для цього виберіть пункт меню

File→New. У вікні, що з'явилося, виберіть **Vector Waveform File** і натискуйте кнопку **OK**.

Додайте необхідні сигнали для побудови тимчасових діаграм. Це робиться за допомогою діалогу **Insert Node or Bus**, який з'являється після подвійного клацання мишею в будь-якому рядку тимчасової діаграми. Далі натискуйте кнопку **Node Finder**. У вікні **Node Finder Node Found** відобразяться назви всіх портів модуля *SM.v*, скопіюйте їх в список **Selected Nodes** і натисніть кнопку **OK**. Потім натисніть кнопку **OK** в діалозі **Insert Node or Bus**.

Встановіть час закінчення симуляції 30 мкс за допомогою команди **Edit→End Time**, крок сітки 1 мкс (за допомогою команди **Edit→Grid Size)**. Щоб відредагувати тимчасову діаграму необхідного вузла, виділіть частину діаграми (всю діаграму вузла можна виділити, клацнувши мишею в полі **Value** потрібного вузла). Далі за допомогою панелі інструментів (розташованої зліва) в редакторі тимчасових діаграм, сформуйте необхідні тимчасові діаграми. Збережіть файл вхідних дій з ім'ям - *SM.vwf* командою **File→>Save**.

Прив'яжіть створений файл вхідних дій до модуля *SM.v*. Для цього у вікні **Project Navigator** виберіть закладку **Hierachy** і пункт контекстного меню для файлу *SM.v Settings*.

У вікні, що з'явилося, виберіть категорію **Simulator Settings**, вкажіть *SM.vwf* в полі **Simulation Input** і натисніть кнопку **OK**.

Запустіть симулятор командою **Processing→Start Simulation**.

Проаналізуйте одержані тимчасові діаграми, у разі потреби відкоректуйте HDL-опис і файл вхідних дій та повторіть компіляцію проекту і моделювання.

Більш детально приклади симуляції проекту розглянуто у Додатку А.

8.2.

Кінцеві автомати широко використовуються в різних цифрових прикладних системах і пристроях, особливо в контролерах. Вихід автомата Мура є функцією тільки поточного стану, вихід автомата Милі - функція як поточного стану, так і початкової зовнішньої дії.

Звичайно кінцевий автомат складається з трьох основних частин:

Регістр поточного стану. Цей регістр є набором тактованих D- тригерів, що синхронізуються одним синхросигналом. Цей набір використовується для зберігання коду поточного стану автомата. Для автомата з n станами потрібен $\mathrm{Log2}(n)$ тригерів.

Логіка переходів. Кінцевий автомат може знаходитися в кожний конкретний момент часу тільки в одному стані. Кожний тактовий імпульс викликає перехід автомата з одного стану в інший. Правила переходу визначаються комбінаційною схемою, званою логікою переходів. Наступний стан визначається як функція поточного стану і вхідної дії.

Логіка формування виходу. Вихід цифрового автомата звичайно визначається як функція поточного стану і початкової установки (у разі автомата Милі). Формування вихідного сигналу автомата визначається за допомогою логіки формування виходу.

Як приклад розглянемо проектування найпростішого синхронного автомата, який формує два імпульси **Out1** і **Out2**, що не перекриваються, у відповідь на появу сигналу **Run** на вході автомата (рис. 7.1 а). Повністю синхронний кінцевий автомат використовує регістри для фіксації всіх вихідних сигналів управління і станів, а також для асинхронних вхідних сигналів. Слід помітити, що синхронні кінцеві автомати по швидкодії поступаються асинхронним.

Мітка, розташована в кожному колі вище за лінію, - це ім'я стану, а влучні нижче за лінії - це вихідні сигнали, які видаються, коли даний стан активний. «дуги», які повертаються в той же самий стан, - це переходи, які працюють за умовчанням. Ці дуги матимуть істинні значення тільки у разі, коли не буде істинних значень інших умов переходів. Кожна умова переходу із стану в стан має відповідну логічну умову, яка повинна виконуватися, щоб кінцевий автомат міг перейти в наступний стан.

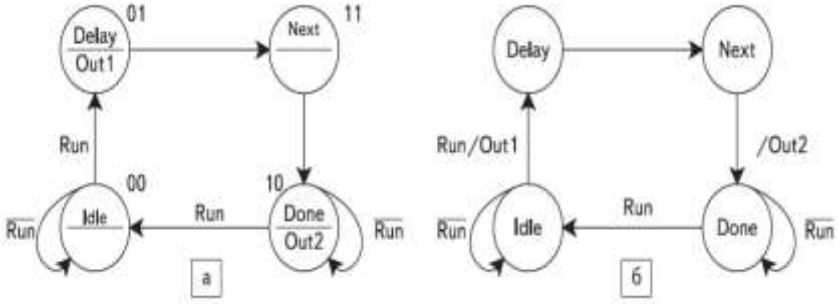


Рис. 7.1. Граф - автомата

Автомат приймає чотири стани: **Idle, Delay, Next, Done** (рис. 7.1 а). Скористаємося методом двійкового кодування станів, який забезпечує високий ступінь кодування послідовності станів. Для кодування станів буде потрібно два тригери. Запишемо булеві логічні рівняння:

$$Idle = \overline{S0} * \overline{S1};$$
$$Delay = \overline{S1} * S0;$$
$$Next = S1 * S0;$$
$$Done = S1 * \overline{S0};$$
$$S0 := (Idle, Run) + Delay;$$
$$S1 := Delay + Next + (Done, \overline{Run});$$
$$Out1 := Idle, Run;$$
$$Out2 := Next.$$

Символ = позначає комбінаційну схему, відповідальну за перехід по станах, а символ := позначає вихід тригера, необхідний для зберігання коду поточного стану автомата і вихідних сигналів. Схема, побудована по булевих рівняннях в САПР ПЛИС Quartus II компанії Altera, показана на рис. 7.2. На рис. 7.3 показані тимчасові діаграми роботи автомата. Рівняння для вихідного

сигналу **Out1** є функцією як стану, так і вхідного сигналу **Run.** Кінцевий автомат з таким видом стробує виходів називається автоматом Милі. Рівняння для вихідного сигналу **Out2** записується як функція тільки стану автомата, що відповідає структурі автомата Мура.

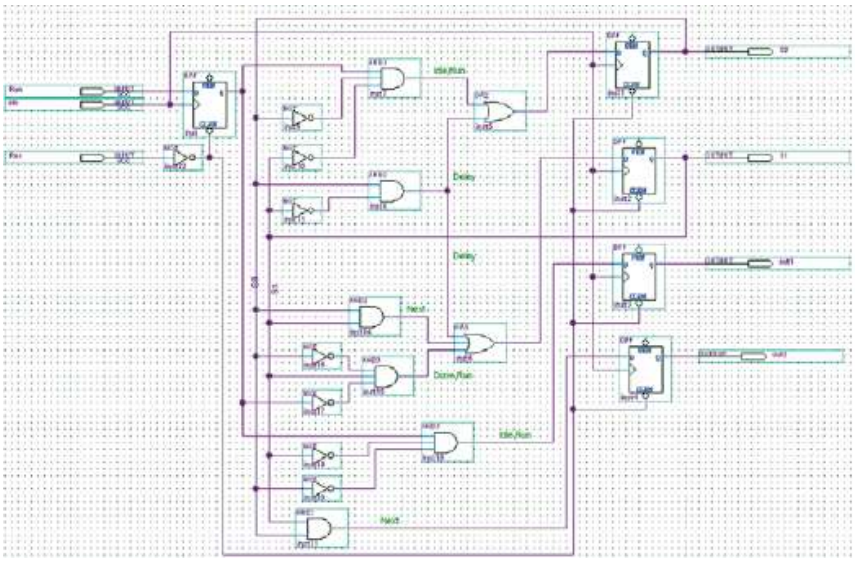


Рис. 7.2. Схема автомата Мура

Для автоматів Милі зручно представляти діаграми в іншому вигляді (рис. 7.1 б).

Метод **one hot encoding** (**OHE** - кодування з одним активним станом; інакше - унітарне кодування) одержав таку назву тому, що в кожний конкретний момент часу активним **(hot)** може бути тільки один тригер стану.

Використовування методу **OHE** для ПЛИС по архітектурі ПКВМ (програмовані користувачем вентильні матриці, зарубіжна абревіатура - FPGA) було запропоновано компанією High-Gate Design. Побудова кінцевого автомата з використанням методу OHE здійснюється по наступній методиці - спочатку для відображення кожного стану автомата виділяється індивідуальний тригер, а потім організовується схема, що дозволяє в кожний конкретний момент часу тільки одному стану бути активним.

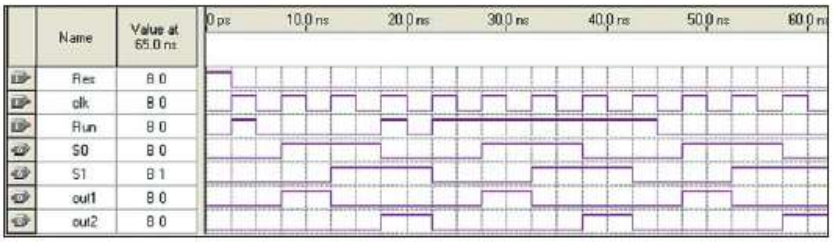


Рис. 7.3. Тимчасові діаграми роботи автомата

Аби описати синтез і моделювання цифрового автоматів роглянемо їх роботу на дискретних компонентах. Спочатку необхідно припустити, що є два логічні сигнали – *Y* і *Z*. Цифровий автомат, що розглядається має виділяти 1

повний імпульс із послідовності цифрових сигналів *G* після надходження керівного сигналу *Y* і другу повну паузу – після приходу управляючого сигналу *Z*. В [1] детально висловлена теорія цифрових синхронних і асинхронних автоматів і методи їх синтезу на основі RS-тригерів і мультиплексорів. Отже за теорією маємо синтезувати заданий автомат як асинхронний автомат Мура на основі асинхронних RS-тригерів. Автомати Мура описуються функціями переходів і виходів

$$a_{t+1} = f(a_1, x_1),\ \ z_1 = \varphi(a_1),$$

де a_1 и z_1 - стан автомата та його вихідний сигнал у момент часу t відповідно.

Користуючись отриманою формулою, можемо зазначити, що новий стан автомата однозначно визначається кожним попереднім станом і вхідним сигналом, при цьому в цей момент часу стан автомата однозначно визначає його вихідний сигнал. Отже, вихідний сигнал автомата Мура визначається тільки станом автомата. А напрям сигналу не залежить від вхідних сигналів. В іншому випадку їх зміни без зміни стану автомата, а вихідний сигнал не змінюється. Тому стани автомата (вихідні сигнали) можна зазначити у вершинах графа переходів.

Граф переходів пристрою, який був розроблений як автомат Мура (див. визначення вище). Визначимо з використовування коду Грея вершини графа (стійкі стани автомата). Отже початковий стан автомата матиме значення - 000.

За умов можливості виокремлення повного імпульсу є присутність керуючого сигналу *Y* і паузи в послідовності, а умовою виділення повної паузи – наявність сигналу *Z* й імпульсу в послідовності.

Визначені комбінації сигналів впорядковують два шляхи переходу автомата з початкового стану 000 у стан 001 або 100. Ці шляхи показано на рис. 7.4 відповідними стрілками. Для завершення дії керуючих сигналів необхідно передбачити повернення автомата у вихідний стан. Керуючою комбінацією для цього переходу слугує ($Y \bullet Z$). Охопимо замкнутою лінією всі стани на графі переходів, у яких значення однієї і тієї ж змінної дорівнює одиниці при цьому стан одного елемента пам'яті.

Замкнуті криві показано суцільною ($Q_i=1$), штриховою ($Q_2=1$) і пунктирною лініями ($Q_3=1$) відповідно. Вхід автомата в подібну замкнуту область станів і вихід з неї вимагають певних сигналів перемикання тригерів. Покажчиками, що входять в область, позначають сигнали установки (які привласнюють змінній одиничне значення), а покажчиками, що виходять з області – сигнали скидання, ті, що набувають нульове значення. Для кожного тригера складається два вирази у формі суми кон'юнкцій (ДНФ): одне для сигналу статкування (Prebar - S), а інше – для сигналу збросу (ClrBar - R). Кожна кон'юнкція повинна містити вхідні змінні (сигнали, якими здійснюється перехід)

і вторинні змінні, пов'язані з цим перехідом, при цьому не змінюють своє значення.

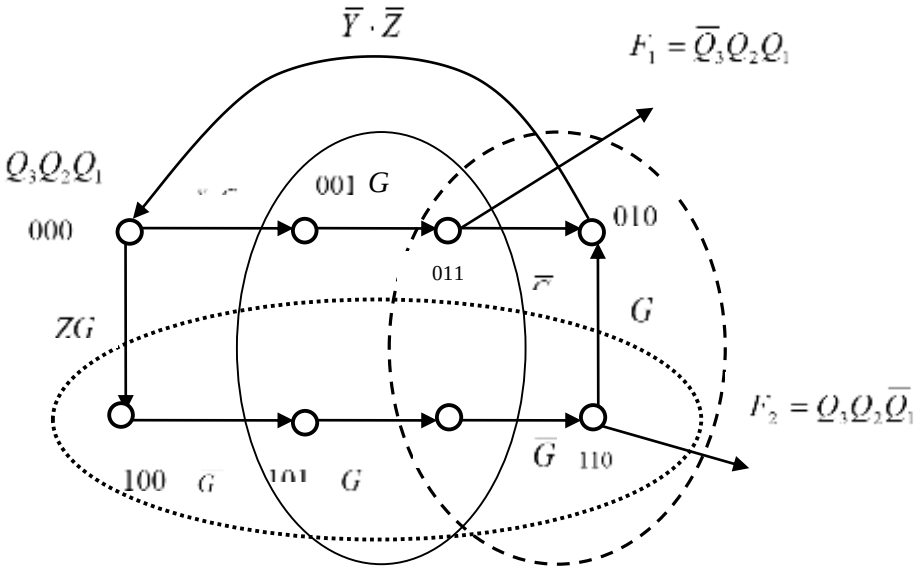


Рис. 7.4. Граф переходів асинхронного автомата Мура

Як вторинні змінні постають двійкові розряди коду стану в коді Грея. Включення у вирази вторинних змінних гарантує виконання переходів у правильній послідовності.

$S_1=\overline{Q}_3\overline{Q}_2YG\vee Q_3\overline{Q}_2G$,

$R_1=\overline{Q}_3Q_2\overline{G}\vee Q_3Q_2\overline{G}=Q_2\overline{G}$;

$S_2=Q_1G$,

$R_2=\overline{Q}_3\overline{Q}_1\overline{YZ}$;

$S_3=\overline{Q}_2\overline{Q}_1ZG$,

$R_3=Q_2\overline{Q}_1G$;

$F_1=\overline{Q}_3Q_2Q_1$,

$F_2=Q_3Q_2\overline{Q}_1$,

$F_3=\overline{Q}_3Q_2Q_1\vee Q_3Q_2\overline{Q}_1$.

Після чого складається схема на базі асинхронних **RS**-тригерів, що реалізовує одержані логічні вирази. Для моделювання за допомогою програми MicroCap-8 як асинхронні **RS**-тригери використовуються **JK**- або **D**-тригери з асинхронними входами статкування (PREBAR) і збросу (CLRBAR). При цьому входи синхронізації й інформаційні входи не задіються. Схема для моделювання наведена на рис. 7.5. Цифрові мікросхеми, що необхідно вибрати з розділу **Component>Digital Library>...** У цьому прикладі для реалізації автомата використана серія ІМС - ТТЛ - 74NNN. Зазначимо, що при складанні цифрових пристроїв і оптимізації зображення схеми зручно використовувати такі інструменти як „гумова лінія" і „ортогональний провідник".

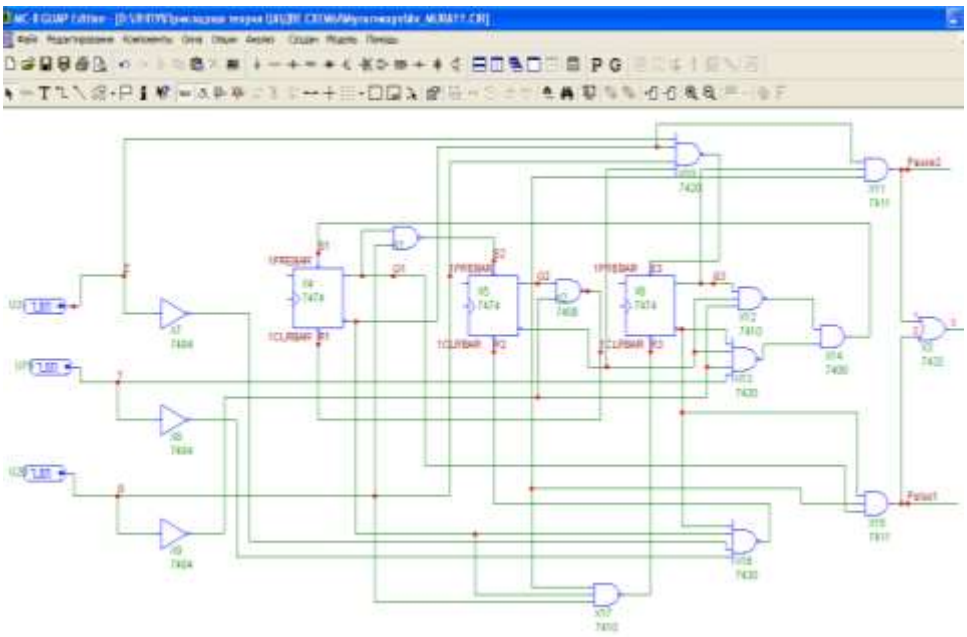


Рис. 7.5. Схема цифрового автомата у вікнах схемного редактора MicroCap 8

Питання для самоперевірки

1. Етапи проектування кінцевого автомату?
2. Як проводиться підключення файлу симуляції до проекту?

Розділ 9. Теоретичне підґрунтя бінарної логіки комп'ютерних систем

Задачі проектування комп'ютерних систем (КС) охоплюють велику кількість питань пов'язаних з технологічною, алгоритмічною та схемотехнічною сферами, які є тісно пов'язані одна з іншою та впливають одна на одну. Однак, найбільшим фактором, який впливає на можливість їх реалізації та застосування є, мабуть, прикладний аспект – яку задачу повинна вирішувати кожна конкретна комп'ютерна система. Але усе різноманіття сучасних комп'ютерних систем у своїй основі використовує математичний апарат бінарної логіки. Слід, також, зазначити, що пов'язані з бінарною логікою питання є більш загальними та фундаментальними у площині формального представлення механізмів програмної обробки даних, втілюючи механізми теорії множин та операційного обчислення.

9.1. Твердження та сполучники у бінарної логіки

Математична логіка - це основа, на якій спираються докази та аргументи. **Пропозиції** - це твердження, що використовуються в математичній логіці, які є істинними, або хибними, але не є обома, і ми можемо однозначно визначити, чи є судження правдивим чи хибним.

Пропозиція (**твердження**) класифікується як декларативний вираз, до якого належить лише одне з визначених значень - може бути істинним (true - T) чи помилковим (false - F).

Так само, як ми формуємо нові речення із наявних речень, використовуючи такі додаткові слова як "і", "але", "якщо", можна отримувати нові пропозиції з поданих пропозицій, використовуючи "сполучники". Але нове речення, отримане із поданих пропозицій із використанням сполучників, буде пропозицією лише тоді, коли воно набуватиме значення істинності - або T, або F (але не обидва). Значення правдивості нового речення залежить від використовуваних (логічних) сполучників та значення істинності поданих пропозицій.

Є п'ять основних сполучників, які використовують для формування логічних математичних виразів:

1) Заперечення (НІ)
2) Сполучення (ТА)
3) Диз'юнкція (АБО)
4) Наслідки (ЯКЩО … ТОДІ …)
5) Якщо і тільки Якщо

Заперечення (NOT)

Якщо P - пропозиція, то заперечення P або NOT P (читається як «не P») - це пропозиція (позначається як ¬ **P**), значення істинності якої **T**, якщо **P** має значення **F**, а значення істинності **F**, якщо **P** має значення істинності **T**.

Зазвичай значення істинності пропозиції, визначені за допомогою сполучника, перераховуються в таблиці, що називається таблицею істинності для цього сполучника (табл. 9.1)

Таблиця 9.1

Таблиця істинності ЗАПЕРЕЧЕННЯ

P	¬ P
T	F
F	T

Сполучення (кон'юнкція) AND

Якщо P і Q є двома пропозиціями, то сполучення P і Q (читається як "P та Q") є пропозицією (позначається P ∧ Q), значення істинності як наведено в таблиці 9.2.

Таблиця 9.2

Таблиця істинності СПОЛУЧЕННЯ

P	Q	P ∧ Q
F	F	F
F	T	F
T	F	F
T	T	T

Диз'юнкція (OR)

Якщо P і Q є двома пропозиціями, то диз'юнкція P і Q (читається як "P або Q") є пропозицією (позначається P ∨ Q), яка набуває значення істинності як наведено в таблиці 9.3.

Таблиця 9.3

Таблиця істинності сполучення ДИЗ'ЮНКЦІЯ

P	Q	P ∨ Q
F	F	F
F	T	T
T	F	T
T	T	T

Слід зазначити, що P v Q є істинним, якщо P є істинним, або Q є істинним, або обом відповідає правда. Таке АБО відоме як включне (інклюзивне) АБО, тобто P є істинним, або Q є істинним, або обидва є істинними. Тут ми визначили АБО в інклюзивному сенсі. Визначимо інший сполучник, який називається виключним (ексклюзивним, **XOR**) АБО - або P є істинним, або Q є істинним, але не те і інше, тобто, де АБО використовується в ексклюзивному значенні.

Розглянемо приклад. Якщо P являє собою вираз «Ця книга хороша», а Q означає «Ця книга дешева», запишемо у символічній формі такі речення:

1) Ця книга хороша і дешева.

2) Ця книга не є доброю, але дешевою.
3) Ця книга дорога, але гарна.
4) Ця книга не є ні гарною, ні дешевою.
5) Ця книга є або хорошою, або дешевою.
Рішення
1) P ∧ Q
2) ¬ P ∧ Q
3) ¬ Q ∧ P
4) ¬ P ∧ ¬ Q
5) P ∨ Q

Примітка: таблиці істинності для P ∧ Q і Q ∧ P збігаються. Отже, P ∧ Q і Q ∧ P еквівалентні. Але в природних мовах цього не потрібно. Наприклад, два речення, а саме: "Я пішов на залізничний вокзал і сів на поїзд", і "Я сів на поїзд і поїхав на залізничний вокзал" мають різний зміст. Вочевидь, що ми не можемо написати друге речення замість першого речення.

Наслідок (Імплікація)

Якщо P і Q є двома пропозиціями, тоді вираз "IF P THEN Q" - це пропозиція (позначається P => Q), яка приймає значення істинності наведені в таблиці 9.4. Також, читається P => Q як "P означає Q".

Таблиця 9.4

Таблиця істинності ІМПЛІКАЦІЇ

P	Q	P => Q
F	F	T
F	T	T
T	F	F
T	T	T

Можна відзначити, що P => Q приймає значення істини F лише у тому випадку, якщо P має значення істини T, а Q має значення істини F. У всіх інших випадках P => Q приймає значення істинності T. У випадку звичайних мов, це співвідноситься із значенням істинності речення "ЯКЩО P ТОДІ Q" лише тоді, коли P є істинним. Коли P неправдиво, значення істинності 'IF P THEN Q' є не співставним. Але у випадку математичної логіки потрібно обов'язково вказати значення істинності P => Q у всіх випадках. Тож, значення істинності P => Q завжди визначається як T, коли P має значення істини F (незалежно від значення істинності Q). Таке тлумачення можна співставити з виразом «Ми можемо довести що завгодно, якщо почати з помилкового припущення».

Пропозицію імплікації використовуємо P => Q, коли хочемо транслювати щось із наступних умов: «P тільки якщо Q», «P є достатньою умовою для Q», «Q є необхідною умовою P», «Q випливає з P», «Q, коли P».

Поєднувач «Якщо і тільки якщо»

Якщо P і Q - два твердження, тоді вираз «P тоді і тільки тоді, коли Q» - це твердження (позначається P <=> Q), значення істинності якого T, коли значення

істинності P і Q однакові, а значення істинності F, коли твердження відрізняються. Значення істинності P <=> Q наведено в таблиці 9.5.

Розглянемо пропозиції P ∧ Q і Q ∧ P. Таблиці істинності цих двох пропозицій ідентичні незалежно від будь-яких пропозицій замість P і будь-яких пропозицій замість Q.

Отже, можна розробити концепції (правила) пропозиції- змінної (відповіднр до пропозицій) та добре сформованої формули (відповідно пропозиціям, що поєднуються сполучниками).

Таблиця 9.5

Таблиця істинності P <=> Q

P	Q	P <=> Q
F	F	T
F	T	F
T	F	F
T	T	T

Запропонована змінна – символ, що представляє будь-яку пропозицію. Зауважимо, що зазвичай дійсна змінна представлена символом *x*. Це означає, що *x* не є реальним числом, але може приймати реальне значення. Аналогічно, змінна пропозиція не є пропозицією, але може бути замінена якоюсь пропозицією.

Зазвичай математичний об'єкт може бути визначеним у термінах умов або властивостей, що задовольняють математичному об'єкту. Ще один спосіб визначення математичного об’єкта - це рекурсія. Спочатку деякі об'єкти оголошуються відповідно до визначення. Вказується процес, за допомогою якого можна побудувати більше об'єктів. Цей спосіб визначення математичного об'єкта називається рекурсивним визначенням. Це, наприклад, відповідає у мовою програмування функції, яка викликає себе.

Добре сформована формула (ДСФ) визначається рекурсивно за умов:

1) Якщо P – пропозиція-змінна, то це ДСФ.

2) Якщо α - ДСФ, тоді ¬ α- ДСФ.

3) Якщо α і β є добре сформованими формулами, тоді (α ∨ β), (α ∧ β), (α => β) і (α <=> β) являють собою ДСФ.

4) Рядок символів являє собою ДСФ, якщо і тільки якщо він отриманий скінченною кількістю застосувань правил 1 - 3.

Зауважимо, що:

(1) ДСФ не є пропозицією, але якщо підставити пропозицію замість пропозиції-змінної, отримаємо пропозицію. Наприклад:

1) ¬ (P ∨ Q) ∧ (¬ Q ∧ R) => Q є ДСФ.

2) (¬ P ∧ Q) <=> Q є ДСФ.

Також, можна відкинути дужки, коли немає двозначності. Наприклад, із пропозицій ми можемо видалити найбільш віддалені дужки. Ми також можемо вказати ієрархію сполучників для уникнення дужок.

Для зручності, посилатимемося на ДСФ як на формулу.

Таблиці істинності

Якщо замінити пропозиції-змінні у формулі реальними пропозиціями, ми отримаємо пропозицію, що включає сполучники. Таблиця, що дає значення істинності такої пропозиції, отриманої шляхом заміни пропозицій-змінних довільними пропозиціями, називається таблицею істинності.

Якщо похідна пропозиція включає n констант-пропозицій, отримаємо 2^n можливих комбінацій значень пропозицій істини, що замінюють змінні.

Наприклад, треба визначити таблицю істинності для пропозиції:

$$\propto = (P \vee Q) \wedge (P => Q) \wedge (Q => P) \qquad (9.1)$$

Таблиця 9.6

Таблиця істинності

P	Q	$(P \vee Q)$	$(P => Q)$	$(P \vee Q) \wedge (P => Q)$	$(Q => P)$	$\propto$
T	T	T	T	T	T	T
T	F	T	F	F	T	F
F	T	T	T	T	F	F
F	F	F	T	F	T	F

Деякі формули мають значення істинності T для всіх можливих сполучень істинних значень змінних-пропозицій. Наприклад. P ∨ ¬P має значення істини T незалежно від значення істинності P. Такі формули називаються тавтологіями.

Тавтологія або загально правдива формула - це ДСФ, значення істинності якої T для всіх можливих значень істинності змінних-пропозицій.

Коли не можливо одразу з'ясувати, чи дана формула є тавтологією, потрібно побудувати таблицю істинності та перевірити, що значення істинності є T для всіх можливих комбінацій значень істинності пропозицій-змінних, що наведені у заданій формулі.

Протиріччя (суперечності або абсурд) - це ДСФ, значення істинності якого F для всіх можливих значень істинності запропонованих змінних.

Наприклад, P ∧ ¬P, або (P ∧ Q) ∧ ¬ Q - протиріччя.

Зауважимо, що: α - це протиріччя, якщо і тільки якщо ¬ α - тавтологія.

Еквівалентні ДСФ

Дві ДСФ α і β від пропозицій-змінних P_1, P_2, … P_n "еквівалентні (або логічно еквівалентні), якщо α <=> β є тавтологією. Коли α і β еквівалентні, пишемо α ≡ β.

Зауважимо, що ДСФ α і β еквівалентні, якщо таблиці правдивості для α і β однакові.

Наприклад, $(P \wedge Q) \equiv (Q \wedge P)$, або $(Q \wedge Q) \equiv Q$.

Важливо, також, відзначити різницю між $\alpha <=> \beta$ та $\alpha \equiv \beta$.

$\alpha <=> \beta$ - це формула, тоді як $\alpha \equiv \beta$ не є формулою, а є визначенням відношення між α і β.

Деякі еквівалентності корисні для виведення інших співвідношень та спрощення пропозицій. Перелік таких ідентичностей наведено у таблиці 9.7.

Таблиця 9.7

Таблиця еквівалентностей

Назва	Еквівалентні вирази
закони ідемпотентності	$x \wedge x = x;$ $x \vee x = x$
комутативність	$x \wedge y = y \wedge x;\ x \vee y = y \vee x$
асоціативність	$x \wedge (y \wedge z) = (x \wedge y) \wedge z = x \wedge y \wedge z;$ $x \vee (y \vee z) = (x \vee y) \vee z = x \vee y \vee z$
закони дистрибутивності	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
закон поглинання	$(x \wedge y) \vee x = x$
закони де Моргана	$\overline{x \vee y} = \bar{x} \wedge \bar{y};\ \overline{x \wedge y} = \bar{x} \vee \bar{y}$
закон подвійного заперечення	$\bar{\bar{x}} = x$
закон протиріччя	$x \wedge \bar{x} = 0$
константні еквівалентності	$x \wedge 1 = x;\ x \wedge 0 = 0;\ x \vee 1 = x;\ x \vee 0 = x;$ $\bar{0} = 1; \bar{1} = 0$
імплікації і еквівалентності	$x \to y = \bar{x} \vee y$ $x \to y \equiv \neg y \ \to \ \neg x$

Нормальна форма ДСФ

Як видно, різні ДСФ існують для двох запропонована змінних, званих P і Q. Також відомо, що дві такі формули є рівнозначними, якщо і тільки якщо вони мають однакову таблицю істинності. Кількість таблиць істинності для формул в P і Q дорівнює 2^4. Оскільки можливі комбінації значень істинності P і Q є TT, TF, FT, FF, таблиця істинності будь-якої формули від P і Q має чотири рядки. Отже, кількість різних таблиць істинності становить 2^4. Таким чином, існує лише 16 чітких (нееквівалентних) формул, і будь-яка формула від P і Q еквівалентна одній з цих 16 формул.

Визначимо метод зменшення заданої формули до еквівалентного формули, яка зветься "нормальною формою". Також, використовуємо "суму" для диз'юнкції, "продукт" для сполучення і "літерал" для кожного P або для ¬ P, де P - будь-яка запропонована змінна.

Визначимо, що елементарний добуток - це добуток літералів. Елементарна сума - це сума літералів. Наприклад, P ∧ ¬ Q, ¬ P ∧ ¬ Q, P ∧ Q, ¬ P ∧ Q - елементарні добутки. І P ∨ ¬ Q, P ∨ ¬ R - елементарні суми.

Формула знаходиться в диз'юнктивному нормальному вигляді (диз'юнктивній нормальній формі - ДНФ), якщо це сума елементарних добутків.

Наприклад, P ∨ (Q ∧ R) і P ∨ (¬ Q ∧ R) знаходяться в диз'юнктивній нормальній формі, в P ∧ (Q ∨ R) не знаходиться в нормальній диз'юнктивній формі.

Для отримання ДНФ треба виконати наступну послідовність дій:

Крок 1 - усунути <=> та => за допомогою логічних ідентичностей.

Крок 2 - використати закони ДеМоргана для усунення заперечення (інверсії) перед сумами або множенням. Отримана формула містить інверсії лише перед запропонована змінними, тобто вона включає суму, добуток і літерали.

Крок 3 - застосувати закони розподілу кілька разів, щоб усунути добуток сум. Отримана формула буде сумою добутків літералів, тобто сумою елементарних добутків.

Для однієї і тієї ж формули ми можемо отримати різні диз'юнктивні нормальні форми. Наприклад, $(P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R)$ і $P \wedge Q$ - диз'юнктивні нормальні форми $P \wedge Q$.

Введемо ще одну нормальну форму, що називається основною диз'юнктивною формою або канонічною формою суми продукцій у визначенні, що наведено далі. Перевагами побудови основної диз'юнктивної нормальної форми є:

1) Для даної формули її основна диз'юнктивна нормальна форма унікальна.

2) Дві формули є рівнозначними лише тоді, коли їх основні диз'юнктивні нормальні форми збігаються.

9.2. Методика побудови канонічної диз’юнктивної нормальної форми для заданої формули

Для побудови канонічної диз’юнктивної нормальної форми потрібно виконати наступні дії:

Крок 1. Отримате нормальну диз’юнктивної форму.

Крок 2. Скоротити елементарні продукти, що завдають суперечність (наприклад, P ∧ ¬P),

Крок 3. Якщо P_i і $\neg P_i$ відсутні в елементарному добутку α, замінити α на $(\alpha \wedge P_i) \vee (\alpha \wedge \neg P_i)$

Крок 4 Повторити крок 3, поки всі елементарні продукції не зведуться до сум.

Щоб уникнути повторення виразів, застосовують закони еквівалентності.

Наприклад, визначити основну ДНФ для формули:

$$\alpha = P \vee (\neg P \wedge \neg Q \wedge R)$$

Тут α вже в нормальній диз'юнктивній формі. Суперечностей немає. Таким чином, потрібно ввести відсутні змінні (крок 3).

$$\begin{aligned} P &\equiv (P \wedge Q) \vee (P \wedge \neg Q) \\ &\equiv ((P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R)) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge \neg R) \\ &\equiv ((P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R)) \vee ((P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge \neg R)) \end{aligned}$$

Таким чином, канонічна форма має вигляд:

$$(P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (P \wedge \neg Q \wedge R)$$
$$\vee (P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R)$$

Визначимо мінітерм від n пропозиційних змінних як добуток, який містить усі n змінні у вигляді літералу. Кожен такий мінітерм, крім того, може бути представлений добутком бінарних еквівалентів для кожного з літералів, а канонічна ДНФ у вигляді сум бінарних строк.

У двох змінних P і Q мінтермами є 00, 01, 10 і 11, кожна ДСФ еквівалентна своїй канонічній диз'юнктивній нормальній формі. Кожна головна ДНФ відповідає минтермам в ньому, і отже, до підмножини {00, 01, 10, 11}. Оскільки кількість підмножин становить 2^4, то кількість чітких формул дорівнює 16.

Таблиця істинності та основна диз'юнктивна нормальна форма тісно пов'язані. Кожен мінітерм відповідає певному значенню істинностей змінних, що дають значення істинності T визначеній похідній функції. Наприклад, $P \wedge Q \wedge \neg R$ відповідає значенням T, T, F до P, Q і R відповідно. Отже, якщо дана таблиця істинності, тоді мінтерми - це ті, які відповідають елементам, що надають значення істинності T похідній функції.

Можна сформувати "двійника" диз'юнктивної нормальної форми, який називають кон'юнктивною нормальною формою.

Формула є у кон'юнктивній нормальній формі, якщо вона є добутком елементарних сум. Якщо α знаходиться в нормальній диз'юнктній формі, то ¬α знаходиться в кон'юнктивній нормальній формі. (Це можна побачити, застосувавши закони ДеМоргана.) Отже, щоб отримати кон'юнктивну нормальну форму α, побудуємо її диз'юнктивну нормальну форму і використовуємо заперечення.

Визначимо як макситерм від n пропозиційних змінних P_1, P_2, … , P_n вираз виду Q_l V Q_2 V ... V Q_n, де кожен Q_n або P_n, або $\neg P_n$.

Формула α - це основна кон'юнктивна нормальна форма, якщо α - добуток макситермів. Для отримання основної кон'юнктивної нормальної форми α побудується основна диз'юнктивна нормальна форма для ¬α та застосувється заперечення.

У логічному міркуванні певна кількість пропозицій приймається істинною, і виходячи з цього припущення виводяться інші пропозиції. Розглянемо деякі важливі правила логічного міркування (правила виводу). Пропозиції, які вважаються істинними, називаються гіпотезами або передумовами. Пропозиція, отримана за допомогою правил виводу, називається висновком. Тож, мається на увазі процес приходу до висновку, а не саме отримання висновку.

9.3. Предикатні обчислення

Розглянемо дві пропозиції: «Рам - студент», і «Сем - студент». Щодо пропозицій, між ними немає ніякого відношення, але ми знаємо, що вони мають щось спільне. І Рам, і Сем наділяються властивістю бути студентом. Ми можемо замінити пропозиції одним твердженням «х є студентом». Замінивши х на Ram або Sam (або будь-яке інше ім'я), ми отримуємо багато пропозицій. Загальна ознака, виражена «є студентом», називається предикатом. У предикатному обчисленні ми маємо справу з реченнями, що включають предикати.

Вирази за участю предикатів використовують у математиці та мовах програмування. Наприклад. «2x + 3y = 4», «IF (D. GE. 0.0) GO TO 20» - це твердження з математики та FORTRAN, відповідно із залученням предикатів. Деякі логічні виведення можливі лише шляхом «розділення» предикатів.

Частина декларативного речення, що описує властивості предмета чи відношення між предметами, називається предикатом. Наприклад, "є студентом" - це предикат.

Речення, які включають предикат, що описує властивість об'єктів, позначаються P(x), де P позначає предикат, а x - змінна, що позначає будь-який об'єкт. Наприклад, P (x) може позначати "x - учень". У цьому реченні x є змінною, а P позначає предикат «є студентом».

Речення «х є батьком у» також включає предикат «є батьком». Тут предикат описує стосунки між двома особами. Ми можемо записати це речення як F (x, y), Аналогічно, 2x + 3y = 4z можна описати S (x, y, z).

Хоча P (x), що включає предикат, схожий на пропозицію, він не є пропозицією. Оскільки P (x) включає змінну x, ми не можемо призначити значення істини P (x). Однак, якщо замінити х окремим об'єктом, отримаємо пропозицію. Наприклад, якщо замінити х на Ram на P (x), отримаємо пропозицію "Ram - студент". (Ми можемо позначити це судження P (Ram).)

Якщо замінити х на "Кішка", то також отримаємо судження (значення істинності - F). Аналогічно, S (2, 0, 1) є пропозицією 2x2 + 3x0 = 4x1 (значення істинності якого T). Також S (1, 1, 1) є пропозицією 2x1 + 3x1 = 4x1 (значення істинності якого F).

Наступні визначення стосується можливих "значень", які можна присвоїти змінним.

Для розповідного речення за участю предиката, множина виразів, або просто множина, є безліч всіх можливих значень, які можуть бути віднесені до змінних.

Наприклад, множина міркування для P (x): «х студент», можна розглядати як сукупність всіх людських імен. Множина міркувань для E(n): «n парне ціле число», може бути прийнято як набір із цілих чисел (або набір дійсних чисел).

Надаючи значення змінним, ми можемо отримати пропозиції з декларативних тверджень, що включають предикати. Деяким реченням, що містять змінні, також можна присвоїти значення істинності. Наприклад, розглянемо "Існує х такий, що. $X^2 = 5$", і "Для всіх X,. $(X)^2 = (-X)^2$,. В обох цих реченнях можна присвоїти значення істини (T в обох випадках). "Існує" та "Для всіх" кількісно оцінюють змінні.

Фраза «для всіх» (позначається ∀) називається універсальним кількісним показником. Використовуючи цей символ, ми можемо записати "Для всіх x, $x^2=(-x)^2$, як ∀x Q(x), де Q(x) $x^2 = (-x)^2$.

Словосполучення "існує" (позначається ∃) називається екзистенціальним квантором. Речення «Існує x таке, що $x^2 = 5$» можна записати як ∃x R(x), де R(x) є $x^2 = 5$.

P(x) у ∀x P(x) або в ∃x P(x) називається областю кількісного показника ∀ або ∃.

Символ ∀ можна читати як "для кожного", "для будь-якого". Символ ∃ можна читати як «для деяких», «як мінімум для одного». Коли ми використовуємо квантори, ми повинні визначати границі виразів. Якщо змінено границі, значення істинності може змінитися. Наприклад, розглянемо ∃x R (x), де R(x) $x^2 = 5$. Якщо множина огляду - це сукупність усіх цілих чисел, тоді ∃x R(x) хибний вираз. Якщо область дії - це множина всіх реальних чисел, тоді ∃x R(x) вірно.

Логічні сполучники, включаючи предикати, можуть використовуватися для декларативних речень. Наступний приклад ілюструє використання сполучників.

Висловіть у символічній формі такі речення за участю предикатів:

1. Усі учні розумні.
2. Деякі студенти не мають успіху.
3. Кожен розумний студент успішний.
4. Деякі успішні студенти не розумні.
5. Деякі студенти розумні та успішні.

Наявність кванторів обумовлює те, що потрібно конкретизувати множину аргументів. Ми можемо визначити множину аргументів як набір усіх учнів.

Нехай C(x) позначає "x розумний".

Нехай S(x) позначає «x є успішним».

Тоді речення 1 можна записати як "∀x C(x). Речення 2-5 можна записати, відповідно, як:

2) ∃x (¬S(x))

3) ∀x (C(x) =>S(x))

4) ∃x (S(x) ∧ ¬ C(x))

5) ∃x (C(x) ∧ ¬S(x))

Для обчислення предикатів добре сформована формула - це рядок змінних, таких як сполучники $x_1,x_2, \ldots x_n$, круглі дужки та квантори, визначені рекурсивно за такими правилами:

1) P($x_1,x_2, \ldots x_n$) – ДСФ, де P - предикат, що включає n змінних $x_1,x_2, \ldots x_n$.

2) Якщо α є ДСФ, тоді ¬α – також ДСФ.

3) Якщо α і β - ДСФ, тоді α ∧ β, α ∨ β, α <=> β, α => β також є ДСФ.

4) Якщо α є ДСФ, а x - будь-яка змінна, тоді) ∀x (α), ∃x (a) - ДСФ.

5) Рядок – це ДСФ, якщо і тільки якщо вона отримана за допомогою обмеженої кількості застосувань правил 1 - 5.

Тож, судження можна розглядати як пропозицію за участю предикатів з 0 змінними. Отже, логічні вирази є ДСФ предикатного обчислення за правилом 1.

Нехай α і β - дві формули предикатів у змінних $x_1, x_2, \ldots x_n$, а U – множина аргументів для α і β. Тоді α і β еквівалентні один одному над U, якщо для кожного можливого значення кожної змінної в α і β результуючі твердження мають однакові значення істинності.

Ми можемо написати α = β над U. Кажуть, що α і β еквівалентні один одному, якщо α ≡ β над U для кожного значення із множини U.

Зауваження У предикатних формулах предикатні змінні можуть бути кванторами або ні. Ми можемо класифікувати предикатні змінні у формулі предиката залежно від того, чи вони кількісно визначені, чи ні. Це призводить до наступних визначень.

Якщо формула виду ∀x P(x) або в ∃x P(x) використана як частина предиката формули α, то така частина називається x-обмеженою (границею) частиною α, і оточення x називаються граничним обмеженням x. Входження x є вільним, якщо воно не є гранично обмеженим. Змінна предиката в α є вільною, якщо його розташування є вільним у будь-якій частині α.

Наприклад, $\propto = \left(\exists x_1\, P(x_1, x_2)\right) \wedge \left(\forall x_2\, Q(x_2, x_3)\right)$, наприклад, область x_1 в $\exists x_1\, P(x_1, x_2)$ є обмеженою і x_2 не обмеженою. У $\forall x_2\, Q(x_2, x_3)$ входження x_2 є обмеженим, а x_3 вільне.

Зауважимо, що у кількісно визначній частині формули предиката, такі як ∀x P(x) або в ∃x P(x), є логічними пропозиціями. Ми можемо вводити значення області дії лише вільним змінним в предикатній формулі α.

Предикатна формула справедлива, якщо для всіх можливих значень з будь-якої області для вільних змінних отримані пропозиції мають значення істинності T (істина, правда).

Предикатна формула здійсненна, якщо для деякого присвоєння значень змінних предиката пропозиція має значення істини T.

Предикатна формула нездійсненна, якщо для всіх можливих присвоєнь значень з будь-якої множини предикативних змінних отримані пропозиції мають значення істинності F.

Зауважимо, що дійсні формули предикатів відповідають тавтологіям серед формул пропозицій, а незадовільні формули предикатів відповідають протиріччям.

Зазначимо, що формули пропозицій є, також, формулами предикатів; формули предикатів (де всі кількісні змінні визначені) є формулами пропозицій. Тому всі правила логічного виводу (перетворень) для формул пропозицій також застосовні для предикаційного обчислення.

Множина та підмножина

Множина - це чітко визначений набір елементів, наприклад, набір усіх студентів освітнього закладу. Так само, колекція всіх книг у бібліотеці коледжу - це також множина. Окремі об'єкти називаються членами або елементами набору.

Для позначення множин використовують великі літери A, B, C, ... Маленькі літери a, b, c, ... використовуються для позначення елементів набору. Коли a - елемент множини A, це позначають як $a \in A$. Якщо a не є елементом A, позначають $a \notin A$.

Множини можуть бути представлені кількома шляхами:

- Перерахувавши його елементи. Всі елементи набору (без повторення) записують, вклавши їх в дужки. Можна записати елементи в будь-якому порядку. Наприклад, набір усіх натуральних чисел, що ділиться на 15 і менше 100, може бути визначений як {15, 30, 45, 60, 75, 90}.
- Описавши властивості елементів набору. Наприклад, попередня множина {15, 30, 45, 60, 75, 90} може бути описана як: {n | n є цілим числом, яке ділиться на 15 і менше 100}. Опис властивості називається предикатом. У цьому випадку кажуть, що набір вказано неявно.
- За допомогою рекурсії. Елементи множини визначають за допомогою правила обчислення елементів. Наприклад, набір усіх натуральних чисел, що залишають залишок 1, коли ділиться на 3, можна описати як $\{a_n \mid a_0 = 1, a_{n+1} = a_n + 3\}$.
- Коли правило обчислення зрозуміло з контексту, задають набір кількома початковими елементами. Попередній набір можна записати у вигляді {1, 4, 7, 10, ...}. Чотири наведені елементи припускають, що обчислювальне правило: $a_{n+1} = a_n + 3$.

Кажуть, що множина A є підмножиною B (записується як $A \subseteq B$), якщо кожен елемент A також є елементом B.

Дві множини A і B рівні (пишемо A = B), якщо їх члени однакові. На практиці довести, що A = B, означає що доводимо $A \subseteq B$ та $B \subseteq$ A.

Набір без елемента називається порожнім набором, також називається нульовим набором або порожнім набором, і позначається $\emptyset$.

Операції з множинами

Визначимо деякі операції з множинами:

- об'єднання $A \cup B = \{x \mid x \in A$ або $x \in B\}$
- перетинання $A \cap B = \{x \mid x \in A$ та $x \in B\}$
- доповнення $A - B = \{x \mid x \in A$ та $x \notin B\}$

Множина всіх підмножин множини A називається множиною-ступенем A. Його потужність – кількість елементів,- визначається як $2^{|A|}$.

Якщо A і B - це два набори. Тоді A x B (декартовий добуток) визначається як {(a, b) | a ∈ *A* і b ∈ *B* }, (a, b) називається впорядкованою парою і відрізняється від (b, a).

Бінарна [1]операція * на множині S - це правило, яке призначає до кожної впорядкованої пари (a, b) елементів із S, унікальний елемент, позначений «a * b». Додавання, наприклад, є бінарною операцією на множині Z всіх цілих чисел. Поєднання є бінарною операцією на 2^A де A - це будь-який не порожній набір. Для бінарних операцій справедливі п’ять постулатів, наведених далі:

1) Замкнутість. Якщо a і b знаходяться в S, то a * b ∈ S.

2) Асоціативність. Якщо a, b і c знаходяться в S, то (a * b) * c = a * (b * c).

3) Елемент ідентичності. Існує унікальний елемент, званий елемент ідентичності (або нейтральний, або одиниця групи) e ∈ S такий, що для будь-якого елемента x ∈ S, x * e = e * x = x.

4) Зворотність. Для кожного елемента x в S існує унікальний елемент x' в S, такий, що x * x' = x' * x = e. Елемент x' називається зворотнім до x.

5) Комутативність. Якщо a, b ∈ S. то a * b = b * a.

Можна зазначити, що двійкова операція може не задовольнити жодному із перерахованих вище п'яти постулатів. Наприклад, S = {1. 2, 3, 4, ...} і двійкова операція буде відніманням, тобто a * b = a - b. Постулат про замкнутість не виконується, оскільки 2 - 3 = -1 ∉ S. Також, (2 - 3) - 4 ≠ 2 - (3 - 4), і тому асоціативність не виконується. Оскільки ми не можемо знайти натуральне число таке, що x - e = e - x = x, постулати 3 і 4 не задовольняються. Очевидно, a - b ≠ b – a і тому комутативність не задовольняється.

Взагалі, інтерес полягає в множинах з бінарними операціями, що задовольняють зазначеним аксіомам.

У теорії множин виділяють кілька типів множин, які класифікають за наступними правилами:

1) Множина S з бінарною операцією * називається напівгрупою, якщо задоволені постулати 1 і 2.

2) Множина S з бінарною операцією * називається моноїдом, якщо задоволені постулати 1-3.

3) Множина S з бінарною операцією * називається групою, якщо постулати 1-4 задоволені.

4) Напівгрупу (моноїдну чи групову) називають комутативною чи абелєвою напівгрупою (моноїдом чи групою), якщо задоволений постулат 5.

На рисунку 1 наведено співвідношення між напівгрупами, моноїдами, групами тощо, де вказано на постулати, які утворюють співвідношення між множинами.

Ми інтерпретуємо рисунок 1 наступним чином: моноїд, що задовольняє постулату 4 - це група. Група, що задовольняє постулату 5 - абелева група тощо.

Нижче наведемо кілька прикладів множин з однією бінарною операцією:

---

[1] У цьому випадку визначення «бінарна операція» вказує на кількість аргументів (операндів).

- Z (множина цілих чисел) з операцією додавання - абелева група.

- Z з операцією множення - абелевий моноїд. (Це не група, оскільки вона не відповідає постулату 4.)

- {1, 2. 3, ...} з додаванням є комутативною напівгрупою, але не моноїд. (Елемент ідентичності може бути лише 0, але 0 не встановлено.)

- Множина-ступінь 2^A де $A \neq 0$ з об'єднанням, є комутативним моноїдом. (Елемент ідентичності дорівнює 0.)

- Набір усіх матриць 2 x 2 з операцією множення є моноїдним, але не абелевим моноїдом.

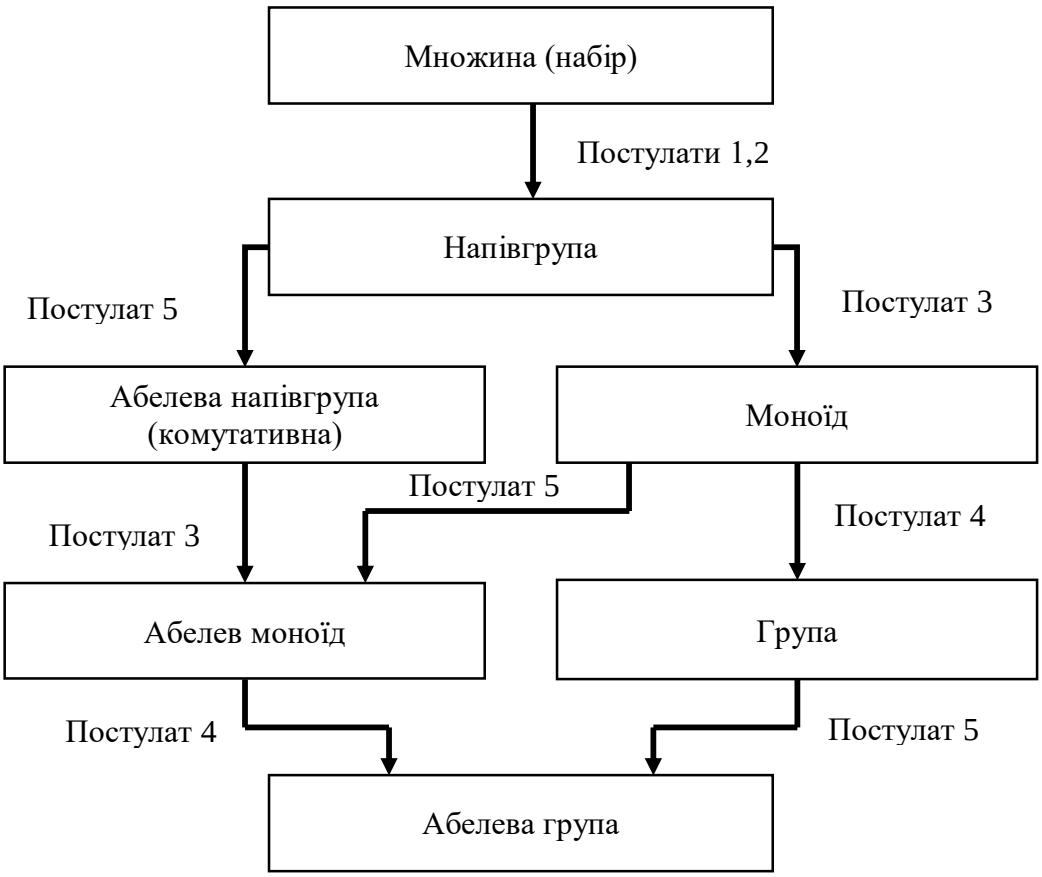


Рис. 9.1 – Співвідношення множин з однією бінарною операцією.

Іноді оперують з множинами з двома визначеними на них бінарними операціями, наприклад, у випадку чисел ми маємо додавання та множення. Нехай S - множина з двома двійковими операціями * і o. Нижче розглянемо 11 постулатів наступним чином:

- Постулати 1 – 5 ідентичні розглянутим вище (операція « * »).

- Постулати 6 – 8 відповідають постулатам 1 – 5, але відносно операції « o ».

- Постулат 9 – Якщо множина S по операції « * » задовольняє постулатам 1-5, то для кожного $x \in S$, з $x \neq e$, існує унікальний елемент $x' \in S$, такий, що x' O x = x O x' = e', де e' - елемент ідентичності, відповідний до операції «O».

- Постулат 11: Поширення. Для a, b, c в S: $a \circ (b * c) = (a \circ b) * (a \circ c)$.

Набір з однією або кількома двійковими операціями називається алгебраїчною системою. Наприклад, групи, моноїди, напівгрупи - це алгебраїчні системи з однією бінарною операцією.

Відношення

Поняття відношення є базовим поняттям як в інформатиці, так і в реальному житті. Ця концепція виникає, коли ми розглядаємо пару об'єктів і порівнюємо один з іншим. Наприклад, "бути батьком" дає відношення між двома особами. Ми можемо виразити відношення впорядкованими парами, наприклад, "a є батьком b", або може бути представлений упорядкованою парою (a, b).

Під час виконання програми проводиться порівняння, і на основі результату виконуються різні завдання. Таким чином, в інформатиці поняття взаємозв'язку виникає так само, як і у випадку зі структурами даних.

Відношення R у множині S - це сукупність упорядкованих пар елементів у S (тобто підмножина S x S). Коли (x, y) знаходяться у відношенні R, це позначають «x R y». Коли (x, y) не є у відношенні R, пишуть «x R' y».

Наприклад, відношення R в Z (множина цілих чисел) може бути визначене xRy, якщо x>y.

Властивості відносин визначаються наступними положеннями:

- Відношення R у S є рефлексивним, якщо xRx для кожного x ∈ S.
- Ставлення До в S є симетричним, якщо для x, y ∈ S, yRx всякий раз, коли xRy.
- Відношення R в S є транзитивним, якщо для x, y ∈ S, xRz всякий раз, коли xRy і yRz.

Зауважимо, що відношення, наведене в прикладі вище, не є ні рефлексивним, ні симетричним, але є транзитивним.

Так, наприклад, відношення R в {1, 2, 3, 4, 5, 6} задається через {(1, 2), (2, 3), (3,4), (4, 4), (4, 5)}. Це відношення не є рефлексивним оскільки 1R'1. Воно не симетричне оскільки 2R3 але 3R'2. Воно також не є транзитивним, оскільки 1R2 і 2R3, але 1R'3.

Інший приклад - визначимо відношення R у {1, 2, ... 10} за допомогою aRb, якщо a ділить b. R є рефлексивним та транзитивним, але не симетричним, оскільки 3R6, але 6R'3.

Відношення R у множині S називається відношенням еквівалентності, якщо воно рефлексивне, симетричне і перехідне. Ми можемо визначити відношення еквівалентності R на будь-якому множині S, визначивши aRb, якщо a = b. Очевидно, що a = a для кожного a, тому R є рефлексивним. Якщо a = b, то b = a. Отже, R симетричне. Також, якщо a = b і b = c, тоді a = c. Отже R є транзитивним.

Розглянемо наступний приклад. Визначимо відношення R на множині всіх осіб у Делі за допомогою aRb, якщо особи a і b мають однакову дату народження. Тоді R - відношення еквівалентності.

Відповідно до будь-якого дня року, ми можемо пов’язати набір усіх осіб, народжених у цей день. Таким чином, поділ усіх людей у Делі можна розділити на 366 підмножин. У кожній з 366 підмножин будь-які два елементи пов'язані між собою. Це призводить до ще однієї властивості відносин еквівалентності.

Нехай R - відношення еквівалентності на множині S. Нехай a ∈ S. Тоді C_a визначається як $\{b \in S \mid aRb\}$. C_a називають класом еквівалентності, що містить a, а множину можливих C_a називають класами еквівалентності.

Для еквівалентних відносин справедлива наступна теорема. Будь-яке відношення еквівалентності R на множині S поділяє S на непересічні класи еквівалентності.

Доказ цієї теореми полягає у наступному. Нехай $\bigcup_{a\in S} C_a$ позначає об'єднання різних класів еквівалентності. Ми повинні довести, що:

1) $S = \bigcup_{a\in S} C_a$
2) $C_a \cap C_b = \emptyset$, якщо $C_a \neq C_b$

Положимо, що $s \in S$. Тоді, $s \in C_s$, оскільки sRs, а R є рефлексивним. Але, $C_s \subseteq C_a$. Таким чином, $S \subseteq C_a$. За визначенням C_a, $C_a \subseteq S$, для кожного a в S. Таким чином, $\bigcup_{a\in S} C_a \subseteq S$. Отже, перше положення доведено.

Зазначимо, що відношення еквівалентності обумовлює те що при aRb, C_a=C_b.

Оскільки aRb, за умови симетрії отримуємо bRa. Припустимо, $d \in C_a$. За визначенням C_a, маємо aRd. Оскільки, bRa та aRd, та враховуючи транзитивність R, отримуємо bRd. Це обумовлює, що $d \in C_b$, що вказує на $C_a \subseteq C_b$. Таким же шляхом можна довести, що $C_b \subseteq C_a$. Відповідно, теорему доказано.

Питання для самостійної роботи

1.1 Які з наведених речень є пропозиціями?
(a) трикутник має три сторони.
(b) 11111 - просте число.
(c) Кожна собака - тварина.
(г) Рам побіг додому.
(e) парне число - це просте число.
(f) 10 - корінь рівняння x2 - 1002x + 10000 = 0
(g) Іди додому і відпочивай.

1.2 Висловіть наступне речення у символічній формі: Для будь-яких двох чисел a і b має місце лише одне з наступних: a <b, a = h і a> h.

1.3 Таблиця істинності сполучника під назвою «виключне або» наведена в наступній таблиці.

P	Q	$P \bar{\vee} Q$
T	T	F
T	F	T
F	T	T
F	F	F

Наведіть приклад речення загальної мови, в якому використовується «виключне або». Покажіть, що цей сполучник є асоціативним, комутативним і дистрибутивним.

1.4 Знайдіть два сполучники, за допомогою яких можна описати будь-який інший сполучник.

1.5 Побудуйте таблицю істинності для наступних пропозицій:

$$(P \vee Q) \Rightarrow ((P \vee R) \Rightarrow (R \vee Q))$$
$$(P \vee (Q \Rightarrow R)) \Leftrightarrow ((P \vee \neg R) \Rightarrow Q)$$

1.6 Доведіть наступні еквівалентності:

$$(\neg P \Rightarrow (\neg P \Rightarrow (\neg P \wedge Q))) \equiv P \vee Q$$
$$P \equiv (P \vee Q) \wedge (P \vee \neg Q)$$
$$\neg (P \Leftrightarrow Q) \equiv (P \wedge \neg Q) \vee (\neg P \wedge Q)$$

1.7 Доведіть, що наступне твердження є тавтологією:

$$P \Rightarrow (Q \Rightarrow (R \Rightarrow (\neg P \Rightarrow (\neg Q \Rightarrow \neg R))))$$

1.8 Чи є наступний вираз тавтологією, суперечністю чи ні тим ні іншим:

$$(P \Rightarrow \neg P) \Rightarrow \neg P$$

1.9 Чи є тавтологією наступний вираз:

$$(P \wedge (P \Rightarrow \neg Q)) \vee (Q \Rightarrow \neg Q) \Rightarrow \neg Q$$

1.10 Визначте основну диз’юнктивну нормальну форму для наступних пропозицій:

$$P \Rightarrow (P \Rightarrow Q \wedge (\neg (\neg Q \vee \neg P)))$$
$$(Q \wedge \neg R \wedge \neg S) \vee (R \wedge S).$$

1.11 Скоротіть формулу, для якої основна ДНФ представлена як 110 ∨ 100 ∨ 010 ∨ 000.

1.12 Перевірте справедливість наступного логічного виводу:
- Якщо Рам розумний, то Прем добре поводиться.
- Якщо Джо хороший, то Сем поганий, а Прем не дуже поводиться.
- Якщо Лал здобув освіту, то Джо добрий або Рам розумний.
Отже, якщо Лал здобув освіту, а Прем не дуже поводиться, то Сем поганий.

1.13 Компанія звернулась із заявами до кандидатів і встановила такі умови:
1) Заявник повинен бути випускником.
2) Якщо він знає Java, він повинен знати C ++.
3) Якщо він знає Visual Basic, він повинен знати Java.
4) Заявник повинен знати Visual Basic.
Чи можете ви спростити вищезазначені умови?

1.14 Побудувавши відповідну множину аргументів, покажіть що наступне твердження є невірним:

$$\exists x\,(P(x) \Rightarrow Q(x)) \Leftrightarrow (\exists x\,P(x) \Rightarrow \exists x\,Q(x))$$

1.15 Покажіть, що наступний логічний вивід дійсний:
- Усі чоловіки смертні.
- Сократ - це людина.
Тож Сократ смертний.

1.16 Чи правдиве таке твердження - якщо філософи не грошові, а деякі люди, що не розумні, то є люди, які не є ні філософами, ні розумними?

1.17 Перевірте правильність наступних міркувань:
- Жодна людина, крім неосвічених, не пишається своїм багатством.
- Деякі люди, які пишаються своїм багатством, не допомагають іншим.
Тому деякі неосвічені особи не можуть допомогти іншим.

1.18 Запишіть у символічній формі такі речення:
а) Ця книга цікава, але вправи важкі.
б) Ця книга цікава, але тема є складною.
в) Ця книга не цікава. вправи важкі, але тема не є складною.
г) Якщо ця книга цікава, а вправи не складні, то тема не є складною.
д) Ця книга цікава, що означає, що тема не є складною, і навпаки.
е) Тема не складна, але ця книга цікава, а вправи - важкі.
ж) Тема не складна, але вправи важкі.
з) Будь-яка книга цікава, або тема складна.

1.19 Якщо A = {a, b} і B = {b, c}, знайдіть:
1) (A ∪ B) *
2) (A ∩ B) *
3) A * ∪ B *
4) A * ∩ B *
5) (A - B) *

1.20 Нехай S = {a, b} *. Для x, y ∈ S визначте x O y = xy, тобто x O .y отримано шляхом склеювання x і y.
1) Чи замкнуте S під a?
2) Чи O асоціативна?
3) Чи є елемент ідентичності відносно операції O?
4) Чи O комутативна?

1.21 Покажіть, що такі відносини є відношеннями еквівалентності:
1) На множині S. aRb, якщо a = b.
2) На множині всіх прямих у площині l1 R l2, якщо l1 паралельна l2.
3) На N = {0,1,2, ...}. mRn, якщо m відрізняється від n кратно з 3.

Розділ 10. Технології проектування комп'ютерних систем. Інформаційний (програмний) рівень

Інформація в сучасному світі перетворилася в один з найбільш важливих ресурсів, а інформаційний рівень як складова в комп'ютерних системах (ІРКС) став необхідним інструментом практично у всіх сферах діяльності.

Різноманітність завдань, що вирішуються за допомогою інформаційного рівня комп'ютерних систем, призвело до появи безлічі різнотипних систем, що відрізняються принципами побудови і закладеними в них правилами обробки інформації.

Інформаційні системи можна класифікувати по цілому ряду різних ознак. В основу даної класифікації покладено найбільш істотні ознаки, що визначають функціональні можливості і особливості побудови сучасних систем. Залежно від обсягу вирішуваних завдань, використовуваних технічних засобів, організації функціонування, інформаційні системи діляться на ряд груп (класів) (рис. 1.0).

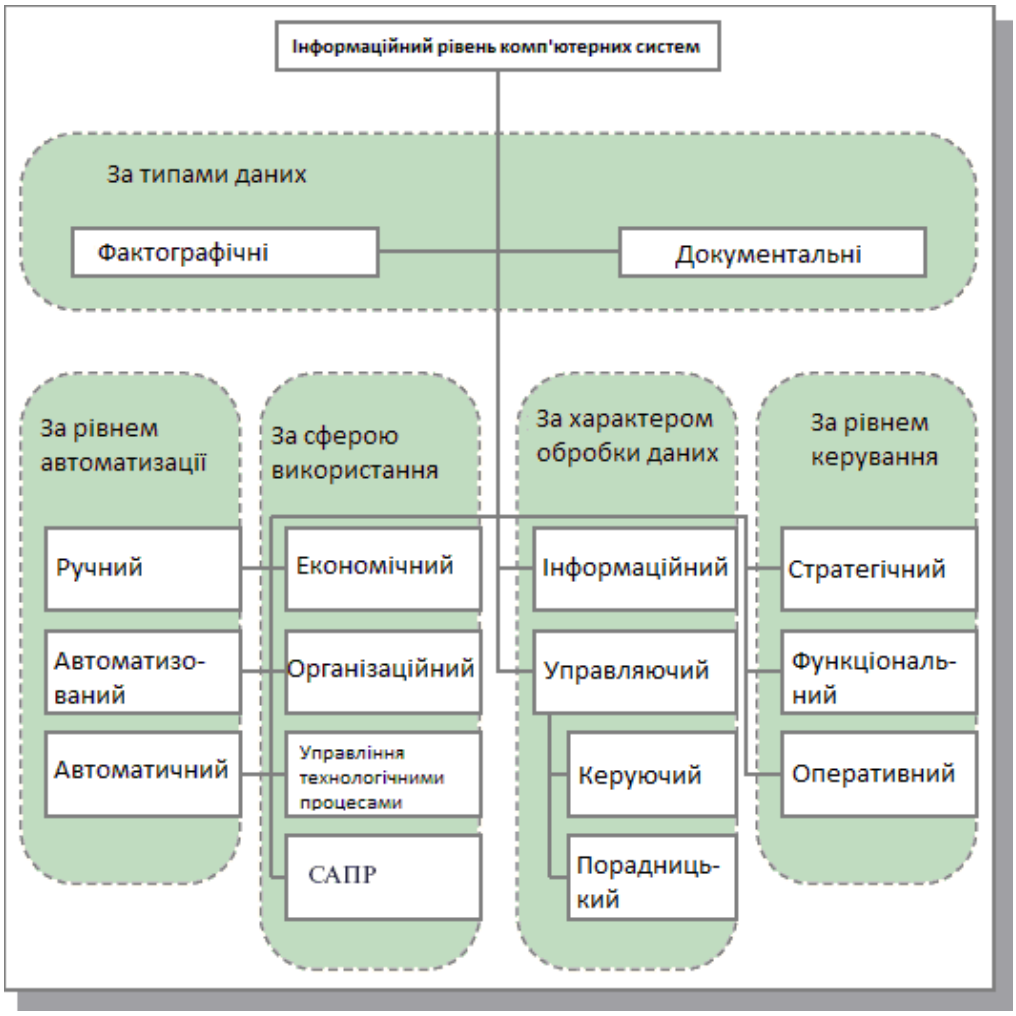


Рис. 10.1. Класифікація інформаційного рівня комп'ютерних систем

За типами даних, що зберігаються, ІРКС ділиться на фактографічні та документальні. Фактографічні системи призначені для зберігання і обробки структурованих даних у вигляді чисел і текстів. Над такими даними можна виконувати різні операції. У документальних системах інформація представлена у вигляді

документів, що складаються з найменувань, описів, рефератів і текстів. Пошук по неструктурованих даних здійснюється з використанням семантичних ознак. Відібрані документи надаються користувачеві, а обробка даних в таких системах практично не проводиться.

За рівнем автоматизації інформаційних процесів в системі управління фірмою, підприємством, організацією тощо, ІРКС ділиться на ручний, автоматичний і автоматизований.

Ручний ІРКС характеризується відсутністю або недостатнім в наявності сучасних технічних засобів обробки інформації та виконанням всіх операцій людиною. В автоматичному ІРКС всі операції з обробки інформації виконуються без участі людини. Автоматизований ІРКС припускає участь в процесі обробки інформації і людини, і технічних засобів, причому головна роль у виконанні рутинних операцій обробки даних відводиться комп'ютерній та іншій техніці. Саме цей клас систем відповідає сучасному уявленню поняттю «інформаційний рівень комп'ютерних систем» або «інформаційна система».

Залежно від характеру обробки даних ІРКС ділиться на інформаційний та управляючий.

Інформаційний ІРКС робить введення, систематизацію, зберігання, видачу інформації за запитом користувача без складних перетворень даних, наприклад, система бібліотечного обслуговування, резервування і продажу квитків на транспорті, бронювання місць в готелях та ін.

Управляючий ІРКС здійснює, крім того, операції переробки інформації за певним алгоритмом управління. За характером використання вихідної інформації такі системи прийнято ділити на керуючі і порадницькі.

Результуюча інформація керуючого ІРКС безпосередньо трансформується в прийняті людиною рішення. Для цих систем характерні завдання розрахункового характеру і обробка великих обсягів даних, наприклад, система планування виробництва або замовлень, бухгалтерського обліку тощо.

Порадницький ІРКС формує інформацію, яка приймається людиною до відома і враховується при формуванні управлінських рішень, а не ініціює конкретні дії. Ці системи імітують інтелектуальні процеси обробки знань, а не даних, наприклад, експертні системи.

Залежно від **сфери використання** розрізняють такі підкласи ІРКС.

Інформаційні системи організаційного управління – призначені для автоматизації функцій управлінського персоналу як промислових підприємств, так і непромислових об'єктів (готелів, банків, магазинів та ін.).

Основними функціями подібних систем є: оперативний контроль і регулювання, оперативний облік і аналіз, перспективне і оперативне планування, бухгалтерський облік, управління збутом, постачанням і інші економічні і організаційні завдання.

ІРКС управління технологічними процесами (ТП) – служить для автоматизації функцій виробничого персоналу по контролю і управлінню виробничими операціями. У таких системах зазвичай передбачається наявність розвинених засобів вимірювання параметрів технологічних процесів (температури, тиску, хімічного складу та ін.), процедур контролю допустимості значень параметрів і регулювання технологічних процесів.

ІРКС автоматизованого проектування (САПР) – призначений для автоматизації функцій інженерів-проектувальників, конструкторів, архітекторів, дизайнерів при створенні нової техніки або технології. Основними функціями подібних систем є: інженерні розрахунки, створення графічної документації (креслень, схем, планів), створення проектної документації, моделювання проектованих об'єктів.

Інтегрований (корпоративний) ІРКС – використовується для автоматизації всіх функцій фірми і охоплюють весь цикл робіт від планування діяльності до збуту продукції. Вони включають в себе ряд модулів (підсистем), що працюють в єдиному інформаційному просторі і виконують функції підтримки відповідних напрямів діяльності.

Існує класифікація ІРКС в залежності від рівня управління, на якому система використовується.

Інформаційний рівень комп'ютерної системи оперативного рівня – підтримує виконавців, обробляючи дані про угоди і події (рахунки, накладні, зарплата, кредити, потік сировини і матеріалів). ІРКС оперативного рівня є сполучною ланкою між фірмою і зовнішнім середовищем.

Завдання, цілі, джерела інформації та алгоритми обробки на оперативному рівні заздалегідь визначені і у високому ступені структуровані.

Інформаційний рівень комп'ютерної системи фахівців – підтримує роботу з даними і знаннями, підвищує продуктивність роботи інженерів і проектувальників. Завдання подібного ІРКС – інтеграція нових відомостей в організацію і допомогу в обробці паперових документів.

Інформаційний рівень комп'ютерної системи рівня менеджменту – використовуються працівниками середньої управлінської ланки для моніторингу, контролю, прийняття рішень і адміністрування. Основні функції ІРКС в цьому рівні:

- порівняння поточних показників з минулими;
- складання періодичних звітів за певний час, а не видача звітів щодо поточних подій, як на оперативному рівні;
- забезпечення доступу до архівної інформації та ін.

Стратегічний рівень ІРКС – комп'ютерна інформаційна система, що забезпечує підтримку прийняття рішень по реалізації стратегічних перспективних цілей розвитку організації.

ІРКС стратегічного рівня допомагає вищій ланці управлінців вирішувати неструктуровані завдання, здійснювати довгострокове планування. Основне завдання – порівняння змін, що відбуваються в зовнішньому оточенні з існуючим потенціалом фірми. Стратегічний рівень покликаний створити загальне середовище комп'ютерної телекомунікаційної підтримки рішень в несподівано виникаючих ситуаціях. Використовуючи найдосконаліші програми, ці системи здатні в будь-який момент надати інформацію з багатьох джерел. Деякі стратегічні системи мають обмежені аналітичними можливостями.

В технологіях проектування комп'ютерних систем є інформаційна (програмна) складова і є методи, орієнтовані на створення проектних основ, моделей та алгоритмів, призначених для розробки інформаційних систем та програмних комплексів, що дозволяє провести аналіз і використати різні методи збору та узагальнення інформації, що передують розробці проекту. Використовуючи різні види нотацій для моделювання

діяльності організації, можливо побудувати різні комп'ютерні системи. Необхідність контролювати процес створення інформаційного рівня комп'ютерної системи, гарантувати досягнення цілей розробки і дотримання різних обмежень (бюджетних, тимчасових і ін.) привело до широкого використання в цій сфері методів і засобів програмної інженерії: структурного аналізу, об'єктно-орієнтованого моделювання, CASE-систем.

Моделювання бізнес-процесів

Моделювання бізнес-процесів (Business process modeling – BPM) – формалізований, виконаний за певними правилами опис послідовності дій фахівців у формі логічних блок-схем, що визначають вибір подальших дій, виходячи з ситуативного факту. Наприклад: «якщо всі документи по лабораторним аналізам для формування сертифікату якості продукції є в наявності, то формуємо цей документ. Якщо немає, то вживаємо заходів для отримання документів, яких не вистачає». У моделі інформаційного рівня певні послідовності окремих дій об'єднуються у відповідні процедури і сценарії бізнес-процесів. Описується взаємодія фахівців різних підрозділів в рамках одного бізнес-процесу.

Моделювання бізнес-процесів – це процесові графічні відображення діяльності підприємства (фірми, установи, організації тощо) з тим, щоб в подальшому інформацію про ці процеси можна було проаналізувати і вдосконалювати.

Як правило метою моделювання бізнес-процесів є:

1. Документування бізнесу компанії.
   1.1. Для отримання знання про діяльність компанії (підприємства).
   1.2. Формування карти підрозділів.
   1.3. Переведення діяльності підприємства (фірми тощо) в інші місця.
   1.4. Для задоволення потреб бізнес-партнерів або об'єднань (наприклад, з метою сертифікації).
   1.5. Для підвищення кваліфікації, навчання співробітників (передачі та обміну знань).
   1.6. Для впровадження проектів (наприклад, підтримки системи менеджменту якості) та екологічного менеджменту.
2. Підготовка бізнес-процесів (який зазвичай починається з аналізу фактичного стану)
   2.1. З метою впровадження нових організаційних структур.
   2.2. Впровадження аутсорсингу, віддаленої роботи або робота з філіалами.
3. Підготовка і впровадження автоматизації IT-підтримки бізнес-систем.
4. Визначення якісних і кількісних показників процесу.
5. Бенчмаркінг.
6. Знаходження найкращої практики діяльності (функціонування компанії).
7. Організаційні зміни.
   7.1. При підготовці до продажу бізнесу.
   7.2. При підготовці до інтеграції компаній або їх частин.
   7.3. Введення або зміна IT-систем та/або організаційних структур.
8. Участь у конкурсах (наприклад, Європейський фонд управління якістю).
9. Вдосконалення внутрішніх процесів.

Для опису та моделювання бізнес-процесів у BPM-системах використовують наступні мови і стандарти (нотації):

Business Process Model and Notation (BPMN) – візуальна нотація моделювання бізнес-процесів. Діаграми бізнес-процесів – основа BPMN. Вони будуються приблизно на тих же принципах, що і традиційні блок-схеми. У процесі виконання модель бізнес-процесу в нотації BPMN транслюється в опис процесу на BPEL, який потім завантажується в движок BPM-системи. Діаграми бізнес-процесів – основа BPMN, будується приблизно на тих же принципах, що і традиційні блок-схеми.

Business Process Execution Language (BPEL) – XML-мова виконання бізнес-процесів. Описує бізнес-процес як пов’язану послідовність веб-сервісів.

XML Process Definition Language (XPDL) – формат обміну даними між BPM-системами.

XPDL запропонований як стандарт для імпорту/експорту описів бізнес-процесів.

IDEF0 – методологія функціонального моделювання і графічного опису процесів, призначена для формалізації і опису бізнес-процесів. (Business Process Modeling, стандарт США). Моделі в нотації IDEF0 призначені для високорівневого опису бізнесу компанії. Їх основна перевага полягає в можливості описувати керування процесами організації. Особливістю IDEF0 є її акцент на ієрархічне представлення об'єктів, що значно полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання в часі (WorkFlow). Як правило, моделювання методологією IDEF0 є першим етапом вивчення будь-якої системи.

IDEF3 (англ. Integrated DEFinition for Process Description Capture Method) – методологія опису потоків робіт (Work Flow Modelling). Призначена для опису робочих процесів або, іншими словами, потоків робіт, коли процеси виконуються в певній послідовності, а також описати об'єкти, що спільно беруть участь в одному процесі. Стандарт IDEF3 близький до алгоритмічних методів побудови схем процесів і стандартних засобів створення блок-схем.

Data Flow Diagramming (DFD) – призначені для опису потоків даних. Вони дозволяють відобразити послідовність робіт, виконуваних по ходу процесу, і потоки інформації, що циркулюють між цими роботами. Крім того, нотація DFD надає можливість описувати потоки документів (документообіг) і матеріальних ресурсів (наприклад, рух матеріалів від однієї роботи до іншої).

Предметні області кожного з стандартів приблизно схожі, а цілі і завдання кожного з них, гранично різні: BPMN – графічна інтерпретація моделі, XPDL – семантика її зберігання і проміжна ланка між іншими стандартами, а BPEL – це рівень високорівневої мови опису взаємодії процесів.

Нотація IDEF0 (Integration Definition for Function Modeling) затверджена як стандарт в Україні, США та інших країнах, в тому числі в МВФ, і успішно експлуатується в багатьох проектах, пов’язаних з описом діяльності підприємств.

1. Короткий опис нотації IDEF0.

В основі методології лежать чотири основні поняття.

Першим з них є поняття функціонального блоку (Activity Box). Графічно функціональний блок зображується у вигляді прямокутника (див. рис. 1.1) і представляє собою деяку конкретну функцію в рамках системи, що розглядається. За вимогами

стандарту назва кожного функціонального блоку має бути сформульована дієслівним способом (наприклад, «аналізувати документи», а не «аналіз документів»).

Кожна з чотирьох сторін функціонального блоку має своє певне призначення (роль), при цьому: верхня сторона блоку має значення «Control» (Управління); ліва сторона має значення «Input» (Вхід); права сторона має значення «Output» (Вихід); нижня сторона має значення «Mechanism» (Механізм).

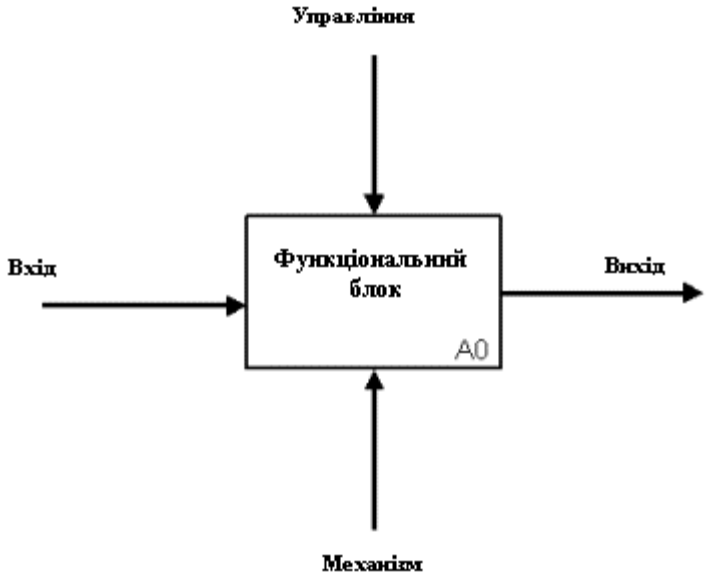


Рис. 10.1. Функціональний блок

Кожен функціональний блок в рамках єдиної системи, що розглядається, повинен мати свій унікальний ідентифікаційний номер.

Далі в методології IDEF0 використовується поняття інтерфейсної дуги (Arrow). Також інтерфейсні дуги можуть називатись потоками або стрілками. Інтерфейсна дуга відображає елемент системи, який обробляється функціональним блоком або надає інший вплив на функцію, що відображається даним функціональним блоком.

Графічним позначенням інтерфейсної дуги є однонаправлена стрілка. Кожна інтерфейсна дуга повинна мати своє унікальну назву (Arrow Label). Згідно вимогам стандарту, найменування повинно бути обігом іменника.

За допомогою інтерфейсних дуг відображають різні об’єкти, які в тій чи іншій мірі визначають процеси, що відбуваються в системі. Такими об’єктами можуть бути елементи реального світу (обладнання, тварини, морозиво, автомобілі, співробітники та ін.) або потоки даних та інформації (документи, дані, інструкції, звіти тощо).

В залежності від того, до якої з сторін функціонального блоку підходить дана інтерфейсна дуга, вона може мати назву «вхідна», «вихідна» або «керуюча». До того ж, «джерелом» (початком) і «приймачем» (кінцем) кожної функціональної дуги можуть бути тільки функціональні блоки, при цьому «джерелом» може бути тільки вихідна сторона функціонального блоку, а «приймачем» будь-яка з трьох, що залишилися.

Слід звернути увагу на те, що будь-який функціональний блок за рекомендацією стандарту повинен мати щонайменше одну керуючу інтерфейсну дугу і одну вихідну. Це є зрозумілим – кожен процес має відбуватися за певними правилам, що

відображується керуючої дугою, і повинен видавати певний результат, який відображується вихідною дугою, інакше його розгляд не має ніякого сенсу.

Обов'язкова наявність «керуючих інтерфейсних дуг» є одним з головних відмінностей стандарту IDEF0 від інших нотацій класів DFD (Data Flow Diagram) і WFD (Work Flow Diagram).

Третім основним поняттям стандарту IDEF0 є декомпозиція (Decomposition). Принцип декомпозиції використовується при розбитті складного процесу на складові його функції. При цьому рівень деталізації процесу, що описується (кількість ітерацій декомпозиції) визначається безпосередньо розробником моделі.

Декомпозиція дозволяє поступово і структуровано представляти модель системи у вигляді ієрархічної структури окремих діаграм, що робить її менш перевантаженою й легкою для засвоєння.

Модель IDEF0 завжди починається з представлення системи як єдиного цілого – одного функціонального блоку з інтерфейсними дугами, що тягнуться за межі даної області. Така діаграма з одним функціональним блоком має назву контекстна діаграма, і позначається ідентифікатором «A-0».

У пояснювальному тексті до контекстної діаграмі повинна бути зазначена мета (Purpose) побудови діаграми у вигляді короткої анотації і зафіксована точка зору (Viewpoint) розробника. Визначення та формалізація мети розробки IDEF0 – моделі є вкрай важливим моментом. Фактично мета визначає відповідні області в системі, що досліджується, на яких необхідно фокусуватися першочергово. Так, наприклад, якщо моделюється діяльність підприємства з метою створення в подальшому на базі цієї моделі комп'ютерної системи, то ця модель буде істотно відрізнятися від тієї, яку б створювали для того ж самого підприємства, але вже з метою оптимізації логістичних зв'язків. Точка зору визначає головний напрямок розвитку моделі і рівень необхідної деталізації. Чітке фіксування точки зору дозволяє розвантажити модель, відмовившись від детального опису і дослідження окремих елементів, які не є необхідними, виходячи з обраної точки зору на систему. Наприклад, функціональні моделі одного і того ж підприємства з точок зору головного інженера та генерального директора будуть істотно відрізнятися за спрямованістю їх деталізації. Це пов'язано з тим, що в кінцевому підсумку, генерального директора не цікавлять аспекти обробки сировини на виробничих ділянках, а головному інженеру ні до чого деталізовані схеми фінансових потоків. Таким чином, правильний вибір точки зору істотно скорочує тимчасові витрати на створення кінцевої моделі.

В процесі декомпозиції, функціональний блок, який в контекстній діаграмі відображає систему як єдине ціле, піддається деталізації в іншій діаграмі. Отримана діаграма другого рівня містить функціональні блоки, які відображають головні підфункції функціонального блоку контекстної діаграми і називається дочірня діаграма (Child diagram) по відношенню до нього (кожен з функціональних блоків, що належить дочірній діаграмі, відповідно, називається дочірнім блоком – Child Box). В свою чергу, функціональний блок – предок, називається батьківським блоком по відношенню до дочірньої діаграми (Parent Box), а діаграма, до якої він належить – батьківською діаграмою (Parent Diagram). Кожна з підфункцій дочірньої діаграми може бути далі деталізована шляхом аналогічної декомпозиції відповідного їй функціонального блоку. Важливо звернути увагу, що в кожній декомпозиції функціонального блоку всі

інтерфейсні дуги, що входять в цей блок, або виходять з нього фіксуються на дочірній діаграмі. Цим досягається структурна цілісність IDEF0–моделі. Принцип декомпозиції наочно представлений на рис. 1.2. Також необхідно звернути увагу на взаємозв'язок позначень функціональних блоків і діаграм – кожен блок має свій унікальний порядковий номер на діаграмі (цифра у правому нижньому куті прямокутника), а позначення під правим кутом вказує на номер дочірньої для цього блоку діаграми. Відсутність цього позначення говорить про те, що декомпозиції для даного блоку не існує.

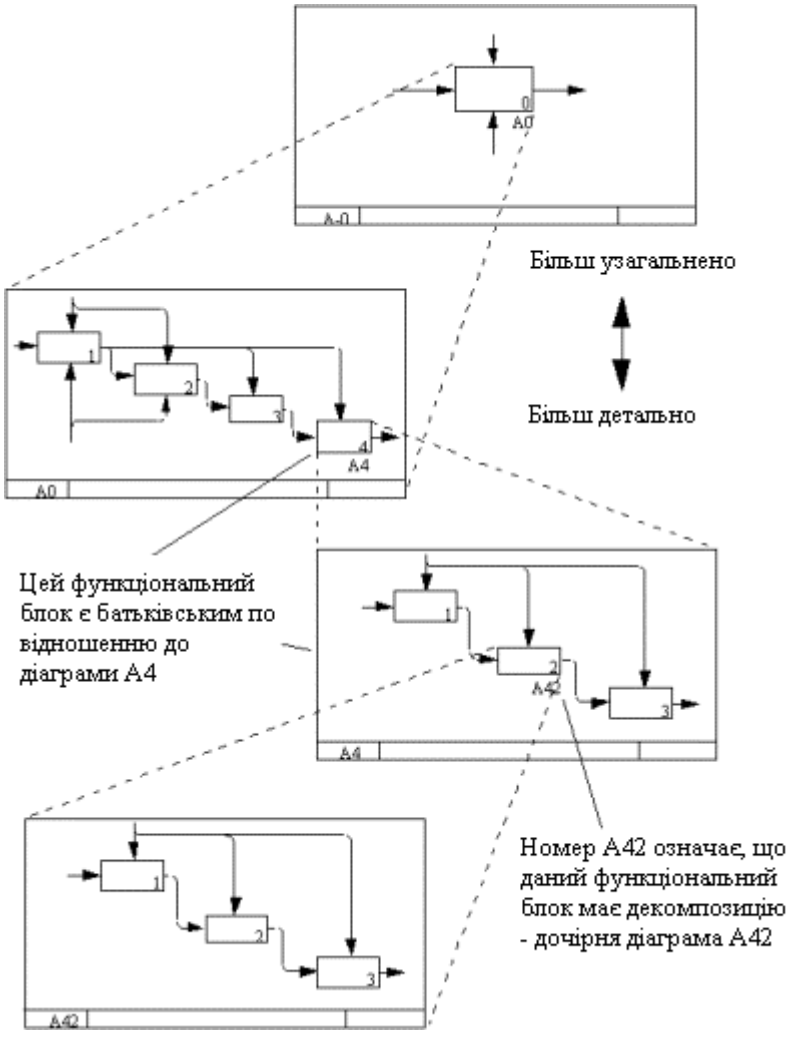


Рис. 10.2. Декомпозиція функціональних блоків

Часто бувають випадки, коли не має необхідності окремі інтерфейсні дуги продовжувати розглядати в дочірніх діаграмах глибше якогось певного рівня в ієрархії, або навпаки – окремі дуги не мають практичного сенсу вище якогось рівня. З іншого боку, виникає необхідність позбутися від певних «концептуальних» інтерфейсних дуг і не деталізувати їх глибше деякого рівня. Для вирішення подібних завдань в нотації IDEF0 передбачено поняття «тунелювання». Позначається «тунель» (Arrow Tunnel) у вигляді двох круглих дужок навколо початку інтерфейсної дуги і це означає, що ця дуга не була успадкована від функціонального батьківського блоку і з'явилася (з «тунелю») тільки на цій діаграмі. В свою чергу, таке ж позначення навколо кінця (стрілки) інтерфейсної дуги в безпосередній близькості від блоку – приймача означає той факт,

що в дочірній по відношенню до цього блоку діаграмі ця дуга відображатися і розглядатися не буде. Найчастіше буває, що окремі об'єкти та відповідні їм інтерфейсні дуги не розглядаються на деяких проміжних рівнях ієрархії – в такій ситуації, вони спочатку «занурюються в тунель», а потім, за необхідністю «повертаються з тунелю».

Останнім з понять IDEF0 є глосарій (Glossary). Для кожного з елементів IDEF0: діаграм, функціональних блоків, інтерфейсних дуг існуючий стандарт нотації передбачає створення та підтримку набору відповідних термінів, визначень, ключових слів, оповідних викладів тощо, які характеризують об’єкт, відображений даним елементом. Такий набір називається глосарієм і є описом суті цього елемента. Наприклад, для керуючої інтерфейсної дуги «розпорядження про виплату» глосарій може містити перелік полів відповідно дузі документа, необхідний набір віз і т.д. Глосарій гармонійно доповнює наочну графічну мову, забезпечуючи діаграми необхідною додатковою інформацією.

Для прикладу розглянемо бізнес-процес пивоварного виробництва.

На першому етапі дослідження будується загальна контекстна діаграма діяльності пивоварного заводу, яка зображена на рис. 1.3.

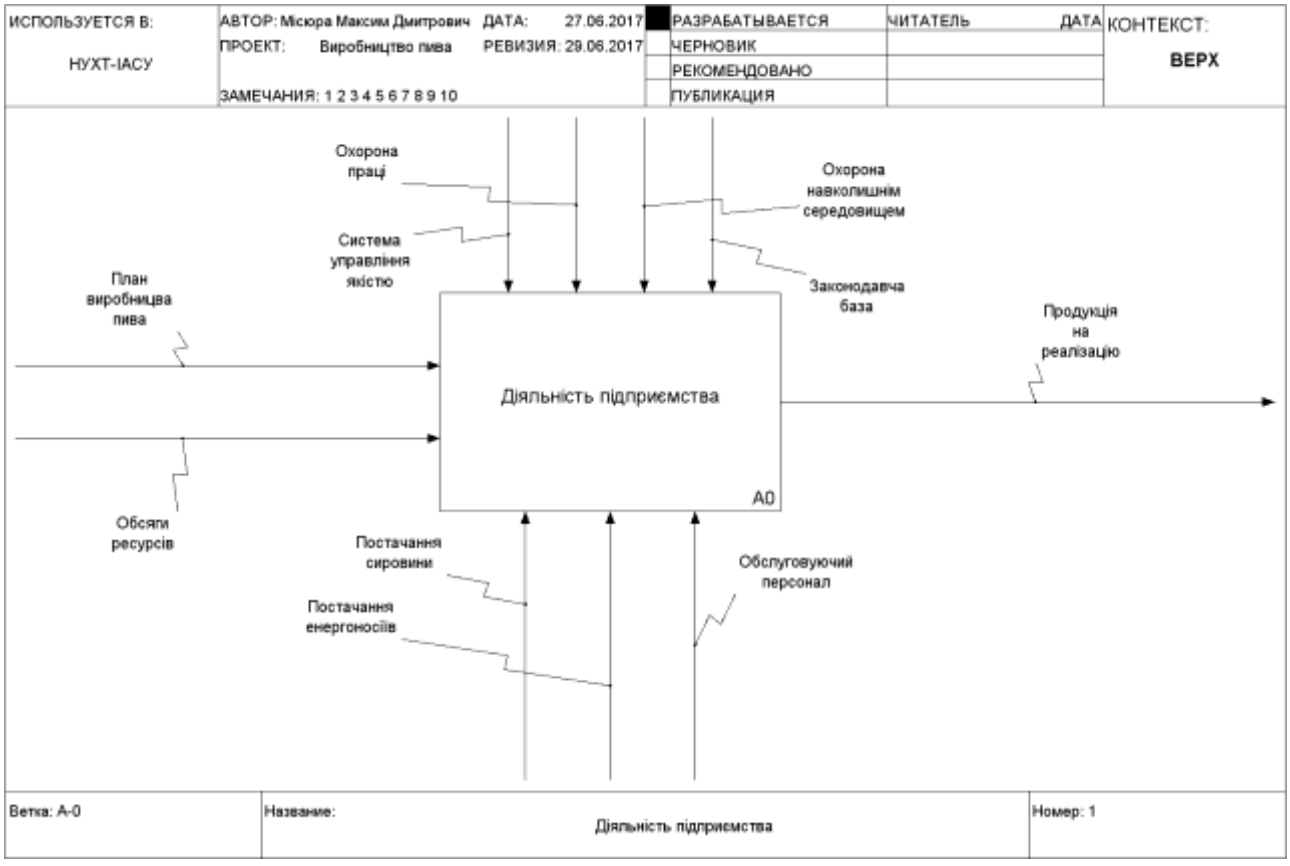


Рис. 10.3. Контекстна діаграма за методологією IDEF0

На вхід контекстної діаграми моделювання бізнес-процесів технології виробництва пива подається інформація про планові завдання, такі як план виробництва пива, обсяги ресурсів, які встановлюються керівництвом підприємства. До механізмів впливу і ресурсів можна віднести: постачання сировини (солод, скляна тара, заготовки ПЕТ-тари тощо), постачання енергоресурсів, що необхідні для виробництва пива;

обслуговуючий персонал. Керуючий вплив, наприклад, може містити у собі: нормативно-правову базу (юридичну базу); систему управління якістю; систему управління охороною праці та навколишнім середовищем. На виході моделі відображаються бізнес-процеси зберігання та відвантаження готової продукції (реалізація пива).

Для детального аналізу та моделювання бізнес-процесів, виконуємо декомпозицію виробництва пива. На рис. 1.4 представлена дочірня діаграма деталізації діяльності пивоварного підприємства. Модель містить в собі наступні бізнес-процеси: «Постачання сировини», «Виробництво пива», «Зберігання, відвантаження пива». Перш за все визначаються нормативи витрат згідно плану виробництва із врахуванням обсягу різних матеріальних і нематеріальних ресурсів. Ці процеси контролюються поставкою сировини та обслуговуючим персоналом та системою управління якістю (наприклад, ISO 9000:2015 та ISO 9001:2015) з врахуванням законодавчої бази (наприклад, ДСТУ 3139:2015 «Пивоваріння. терміни та визначення понять» та ДСТУ 3888:2015 «Пиво. Загальні технічні умови»). Наступним етапом описується бізнес-процес безпосереднього процесу виробництва пива. Даний етап контролюється підсистемою постачання відповідних ресурсів на виробництво, обслуговуючим персоналом. Це враховується відповідними системами якістю виробництва, вимогами до охорони навколишнього середовища та охорони праці, різними законами та підзаконними актами України та світу. Останній етап описує (визначає, моделює) бізнес-процеси зберігання готової продукції та відвантаження покупцеві товарну продукцію (пиво). Такого роду моделювання можна віднести до типу «AS-IS» («Як є»). Якщо проектується нова модель бізнес-процесів або необхідно вирішити задачі оптимізації, то рекомендується використовувати модель «AS-TO-BE» («Як повинно бути»).

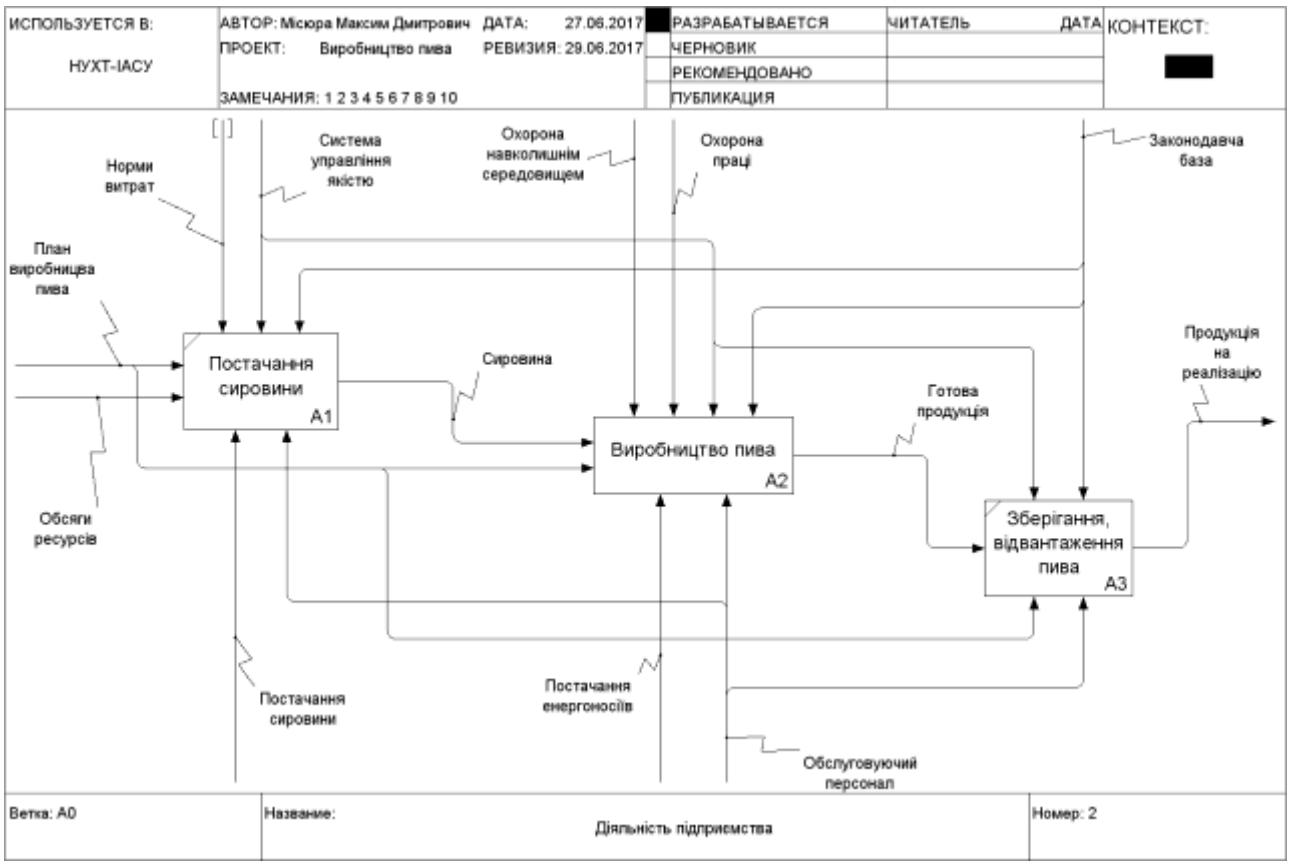


Рис. 10.4. Дочірня діаграма за методологією IDEF0

Деталізована модель бізнес-процесів виробництва пива представлена на рис. 1.5. На першому етапі буде розглядатись підготовка до виробництва, що включає інформацію про робоче обладнання та обсяги (план) виробництва із врахуванням технологічного регламенту, наявності енергоресурсів тощо. Контроль за виконанням на даному етапі буде покладено на обслуговуючий персонал. Наступний (другий) етап це безпосередньо технологічний процес виробництва пива (позначено як «Виробництво пива»), який може передбачати різні ситуації: процеси протікають згідно з технологічним регламентом та інших факторів, на виході отримуємо готову продукцію належної якості; виникають проблеми технічного характеру, що пов’язано в аварією обладнання (поломкою обладнання); проблеми, що викликані порушенням технологічного процесу (регламенту технологічного процесу).

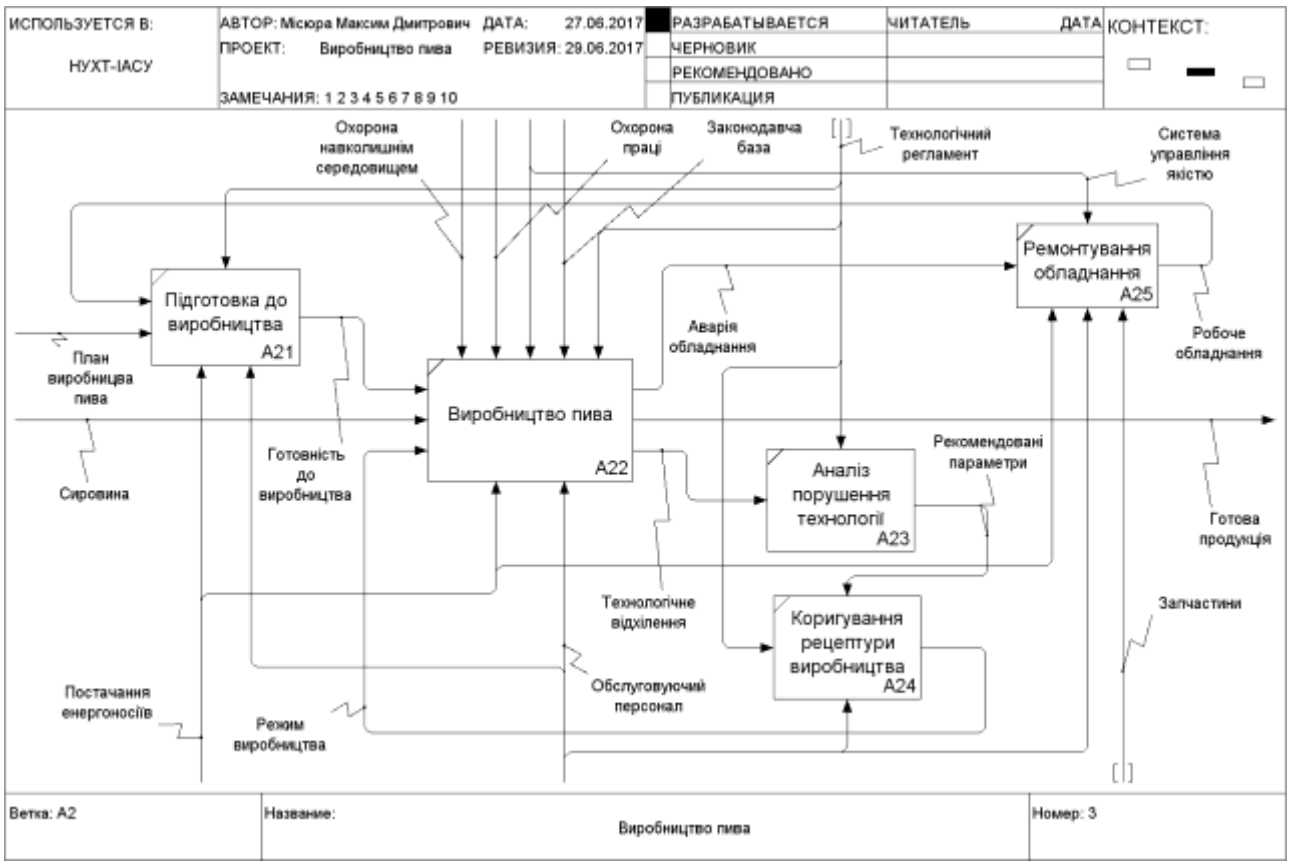


Рис. 10.5. Моделювання бізнес-процесів виробництва пива

2.Короткий опис нотації BPMN

Процес моделювання в нотації BPMN здійснюється за допомогою діаграм з невеликим числом графічних елементів. Це допомагає користувачам (проектантам) швидко розуміти логіку процесу моделювання.

Виділяють чотири основні категорії елементів:

1. *Об'єкти потоку управління:* події, дії і логічні оператори;
2. *З'єднуючі об'єкти:* потік управління, потік повідомлень і асоціації;
3. *Ролі:* пули і доріжки;
4. *Артефакти:* дані, групи і текстові анотації.

За допомогою елементів наведених категорій дослідники мають можливість будувати найпростіші діаграми бізнес-процесів. Для підвищення наочності побудови моделі специфікація дозволяє створювати нові типи об'єктів потоку управління і артефактів.

Об'єкти потоку управління

Об'єкти потоку управління поділяються на три основних типи: події (events), дії (activities) і логічні оператори (gateways).

Події

Зображуються у вигляді кола і означають будь яку подію в світі. Події ініціюють дії або є їх результатами.

Згідно розташуванню, в процесі події можуть бути класифіковані на: початкові (англ. Start), проміжні (intermediate) і завершальні (end).

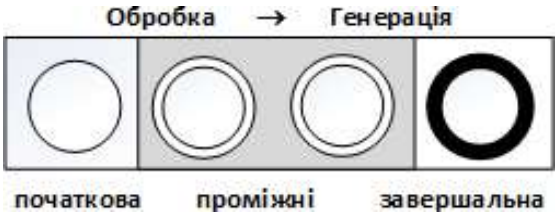
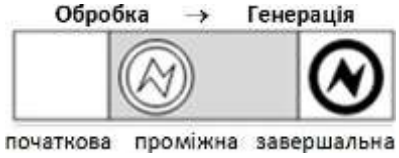
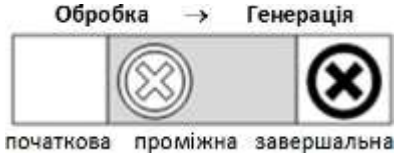
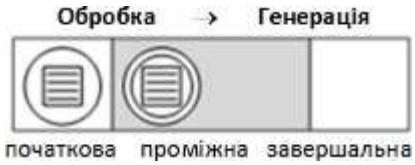
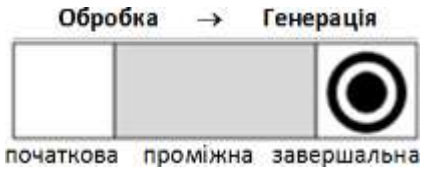


Рис. 2.1 Позначення подій

В BPMN розрізняють події обробки і генерації. Нижче представлена категоризація подій за типами (табл. 2.1.).

Таблиця 2.1. Категоризація подій за типами

Позначення	Опис
	**Прості події (plain events)** це нетипізовані події, які використовуються, найчастіше, для того, щоб показати початок або закінчення процесу.
	**Події-повідомлення (message events)** в ході виконання процесу показують отримання і відправку повідомлень.
	**Події-таймери (timer events)** Моделюють події, що регулярно відбуваються по часу. Також дозволяють моделювати моменти часу, періоди і тайм-аути.
	**Події-помилки (error events)** Помилки можуть мати різні типи. Дозволяють змоделювати генерацію і обробку помилок в процесі.
	**Події-відміни (cancel events)** Ініціюють чи реагують на відміну транзакцій.
	**Події-компенсації (compensation events)** Ініціюють компенсацію чи виконують дію по компенсації.

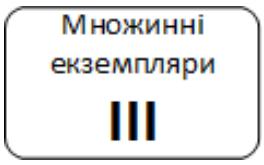
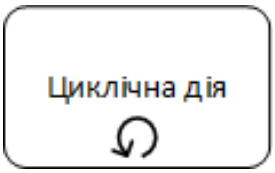
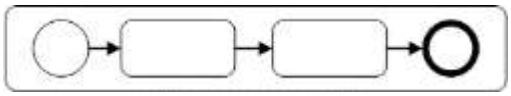
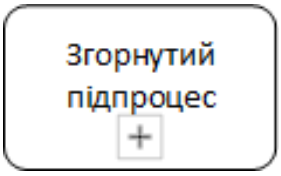
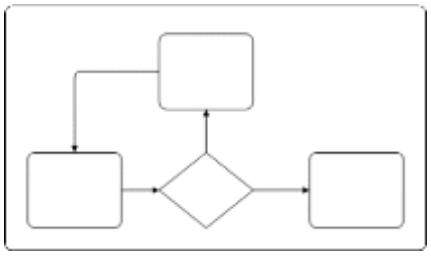
Позначення	Опис
	**Події-умови (conditional events)** Дозволяють інтегрувати бізнес правила в процес.
	**Події-сигнали (signal events)** Розсилають і приймають сигнали між декількома процесами. Один сигнал може оброблятись декількома адресатами. Таким чином, події-сигнали дозволяють реалізувати широкомовну розсилку повідомлень.
	**Складні Події (multiple events)** Моделює генерацію і моделювання однієї події з множини.
	**Події-посилання (link events)** Використовуються як міжсторінкових з'єднання. Пара відповідних посилань еквівалентна потоку управління.
	**Події-зупинки (terminate events)** Ініціюють негайне завершення всього бізнес-процесу (на свій діаграмі).

Дії

Зображуються прямокутниками із закругленими кутами. Серед дій розрізняють завдання та підпроцеси. Графічне зображення згорнутого підпроцесу містить знак плюс біля нижньої межі прямокутника (табл. 2.2.).

Таблиця 2.2. Дії BPMN

Позначення	Опис
	**Завдання (task)** – це одиниця роботи, елементарна дія в процесі.

Позначення	Опис
	**Множинні екземпляри (multiple instances)** - показують, що одна дія виконується багаторазово, по одному разу для кожного об'єкта. Наприклад, для кожного об'єкта в замовленні клієнта виконується один екземпляр дії. Екземпляри дії можуть виконуватися паралельно чи послідовно.
	**Циклічна дія (loop activity)** - виконується, поки умова циклу вірна. Умова циклу може перевірятися до чи після виконання дії.
	**Розгорнутий підпроцес (expanded subprocess)** є складним процесом і містить в собі власну діаграму бізнес-процесів.
	**Згорнутий підпроцес (collapsed subprocess)** також є складовою дією, але приховує деталі реалізації процесу.
	**Ad-hoc-підпроцес (ad-hoc subprocess)** містить завдання. Завдання виконуються до тих пір, поки не виконана умова завершення підпроцесу.

Логічні оператори

Зображуються ромбами і представляють точки прийняття рішень в процесі. За допомогою логічних операторів організовується розгалуження і синхронізація потоків управління в моделі процесу (табл. 2.3.).

Таблиця 2.3. Логічні оператори

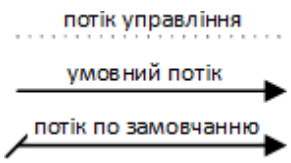
Позначення	Опис
	**Оператор виключне «або», керований даними (англ. Data-based exclusive gateway).** Якщо оператор використовується для розгалуження, то потік управління спрямовується лише по одній вихідній гілці. Якщо оператор використовується для синхронізації, то він очікує завершення виконання однієї вхідної гілки і активує вихідний потік.

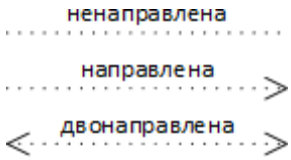
Позначення	Опис
	**Оператор виключне «або», що управляється подіями (event-based exclusive gateway)** Направляє потік управління лише по тій вихідній гілці, на якій першою відбулась подія. Після оператора даного типу можуть іти лише події чи дії-обробники повідомлень.
	**Оператор включне «або» (inclusive gateway)** Активує одну або більше вихідних гілок, в разі, коли здійснюється розгалуження. Якщо оператор використовується для синхронізації, то він очікує завершення виконання однієї вхідної гілки і активує вихідний потік.
	**Оператор «і» (parallel gateway)**, використовується для розгалуження. Розділяє один потік управління на кілька паралельних. При цьому всі вихідні гілки активуються одночасно. Якщо оператор використовується для синхронізації, то він очікує завершення виконання всіх вхідних гілок і лише потім активує вихідний потік.
	**Складний оператор (complex gateway)** Має кілька умов, в залежності від виконання яких активуються вихідні гілки. Оператор ускладнює розуміння діаграми, так як умови, що визначають семантику оператора, графічно не виражені на діаграмі. Внаслідок цього використання оператора небажане.

З'єднувальні об'єкти

Об'єкти потоку управління пов'язані один з одним з'єднуючими об'єктами. Існує три види з'єднуючих об'єктів: потоки управління, потоки повідомлень і асоціації (табл. 2.4.).

Таблиця 2.4. З'єднувальні об'єкти

Позначення	Опис
	**Потік управління** Зображується суцільною лінією, що закінчується зафарбованою стрілкою. Потік управління задає порядок виконання дій. Якщо лінія потоку управління перекреслена діагональної рисою з боку вузла, з якого вона виходить, то вона позначає потік, що виконується за замовчуванням.

Позначення	Опис
	Потік сповіщень Зображується штриховий лінією, що закінчується відкритою стрілкою. Потік повідомлень показує, якими повідомленнями обмінюються учасники.
	**Асоціації** Зображуються пунктирною лінією, що закінчується стрілкою. Асоціації використовуються для асоціювання артефактів, даних або текстових анотацій з об'єктами потоку управління.

Ролі

Ролі - візуальний механізм організації різних дій в категорії з подібною функціональністю. Існує два типи ролей (рис. 2.2.).

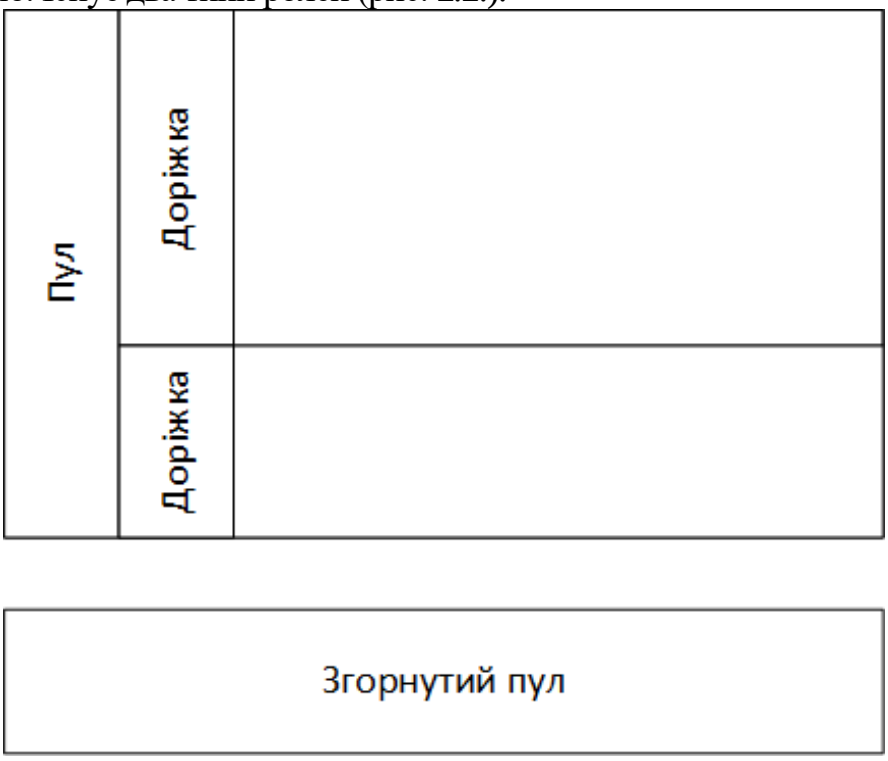


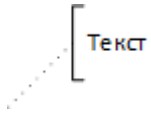
Рис. 2.2. Ролі та пули

- **Пули -** зображуються прямокутником, який містить кілька об'єктів потоку управління, що з'єднують об'єкти і артефакти.
- **Доріжки -** являють собою частину пулу. Доріжки дозволяють організувати об'єкти потоку управління, що зв'язують об'єкти і артефакти.

Артефакти

Артефакти дозволяють розробникам в діаграмі відображати додаткову інформацію. Це робить діаграму більш наочною, зрозумілою і насиченою. Існують три види зумовлених артефактів (табл.2.5.)

Таблиця 2.5. Артефакти

Позначення	Опис
	**Дані** - показують читачеві, які дані необхідні діям для виконання і які дані дії виробляють.
	**Група** - зображується прямокутником із закругленими кутами, межа якого – штрихова лінія. Група дозволяє об'єднати різні дії, але не впливає на потік управління в діаграмі.
	**Текстові анотації -** використовуються для уточнення значення елементів діаграми і підвищення її інформативності.

Засоби сповіщення в BPMN

Основним типом бізнес-комунікацій є повідомлення, надіслані та отримані від одного учасника до іншого в рамках процесу. Повідомлення являє собою вміст (інформацію) діалогу між двома учасниками.

Рис 2.3 – З'єднувальні об'єкти

Сам же діалог (процес обміну інформацією) між двома учасниками відображається у вигляді Потоку повідомлень. Потік повідомлень використовується для відображення того порядку, в якому відбувається обмін повідомленнями (інформацією, даними) між двома зацікавленими сторонами, готовими як відсилати, так і отримувати ці повідомлення. У специфікації BPMN зацікавлені сторони представлені пулами (бізнес-об'єкти або ділової ролі).

Графічний елемент Потоку повідомлень являє собою стрілку з вільним кінцем, виконаної у вигляді пунктирною чорною лінії (пунктир дозволяє легко відрізнити Потік повідомлень від Потоку операцій, виконаного стрілкою у вигляді безперервної чорної лінії).

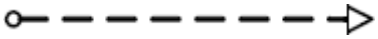


Рис. 2.4 - Графічне представлення Потоку повідомлень

Потік повідомлень може з'єднувати лише два різних пулу (учасників процесу) між собою або елементи, розташовані всередині цих пулів. Однак, Потік повідомлень не може поєднувати два елементи, розташовані всередині одного і того ж пулу.

Наприклад, в процесі «Прийом замовлення» взаємодія (обмін інформацією) між Клієнтом і Менеджером компанії може відображатися наступною схемою:

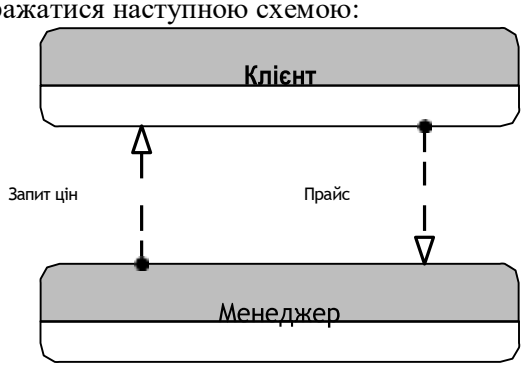


Рис 2.5 - Потік повідомлень між учасниками процесу

Стрілка використовується для відображення взаємодії між процесами - для зв'язку елементів потоку зі згорнутими пулами. При необхідності Потік може бути

іменованим. Потік повідомлень не відображає хід виконання процесу, а показує передачу повідомлень або об'єктів з одного процесу в інший процес або зовнішнє посилання.

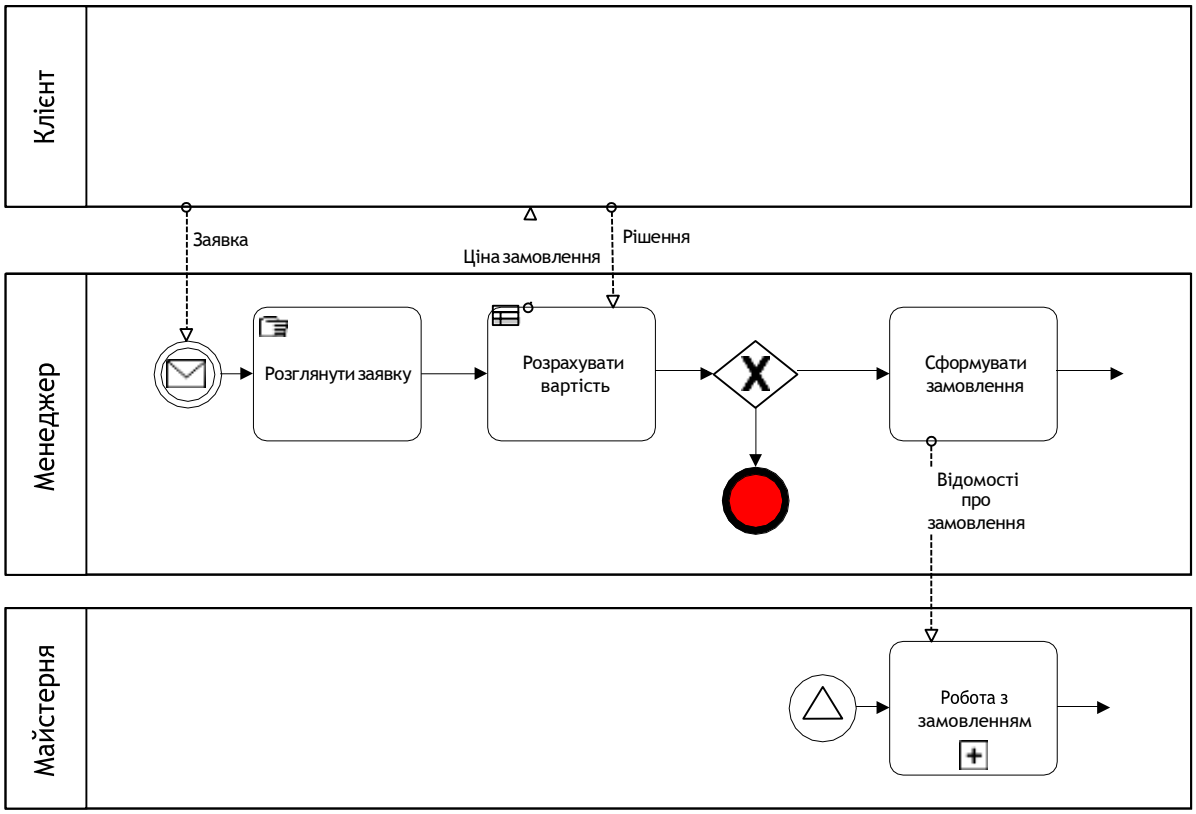


Рис. 2.6 - Приклади використання Потоку Повідомлень

Якщо Повідомлення відображає зміст діалогу (взаємодії) між учасниками процесу, то безпосередньо сам діалог здійснюється через дії (завдання) – Відправку і Отримання повідомлень.

Отримання повідомлень являє собою просте завдання, суть якого полягає в отриманні повідомлення, яке повинно надійти від зовнішнього учасника процесу (що має відношення до даного бізнес-процесу). Завдання вважається виконаним у разі, якщо повідомлення було отримано хоча б один раз.

Графічно Отримання повідомлень відображається у вигляді прямокутника з закругленими кутами (встановлене в BPMN відображення графічного елементу Завдання) і відрізняється від інших типів Завдань наявністю маркера у вигляді конвертика без заливки.

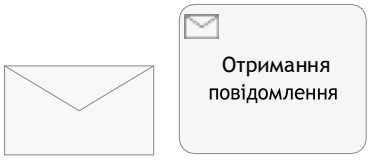


Рис. 2.7 - Графічний елемент Отримання Повідомлення

Відправлення повідомлень являє собою Завдання, суть якої полягає у відправці повідомлення зовнішньому учаснику процесу (яке має відношення до даного бізнес-процесу). Завдання вважається виконаним у разі, якщо повідомлення було відправлено хоча б один раз.

Графічно Відправлення повідомлень відображається у вигляді прямокутника з закругленими кутами (встановлене в BPMN відображення графічного елементу Завдання) і відрізняється від інших типів Завдань наявністю маркера у вигляді конвертика з темної заливкою:

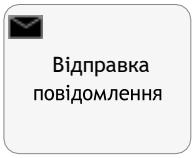


Рис. 2.8 - Графічний елемент Відправка Повідомлення

BPMN виділяє кілька елементів, призначених для зберігання і передачі компонентів в ході виконання процесу: Об'єкти даних і сховище даних. Зазвичай такі елементи відносять до «пов'язаним з компонентами».

Графічне представлення елемента Об'єкт даних має вигляд листа документа з загнутим кутом.

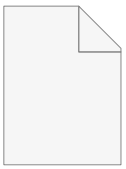


Рис. 2.9 - Графічне зображення об'єкта даних

В нотації BPMN 2.0 (остання версія нотації) вводиться нове поняття Сховище даних для моделювання постійної пам'яті. Даний об'єкт використовується процесом для запису і вилучення даних як, наприклад, бази даних. Збережена інформація буде дійсна навіть після завершення виконання примірника процесу.

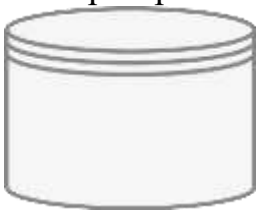


Рис. 2.10 - Графічний елемент Сховище даних

Приклад створення бізнес-процесу в нотації BPMN наведено на рис. 2.11

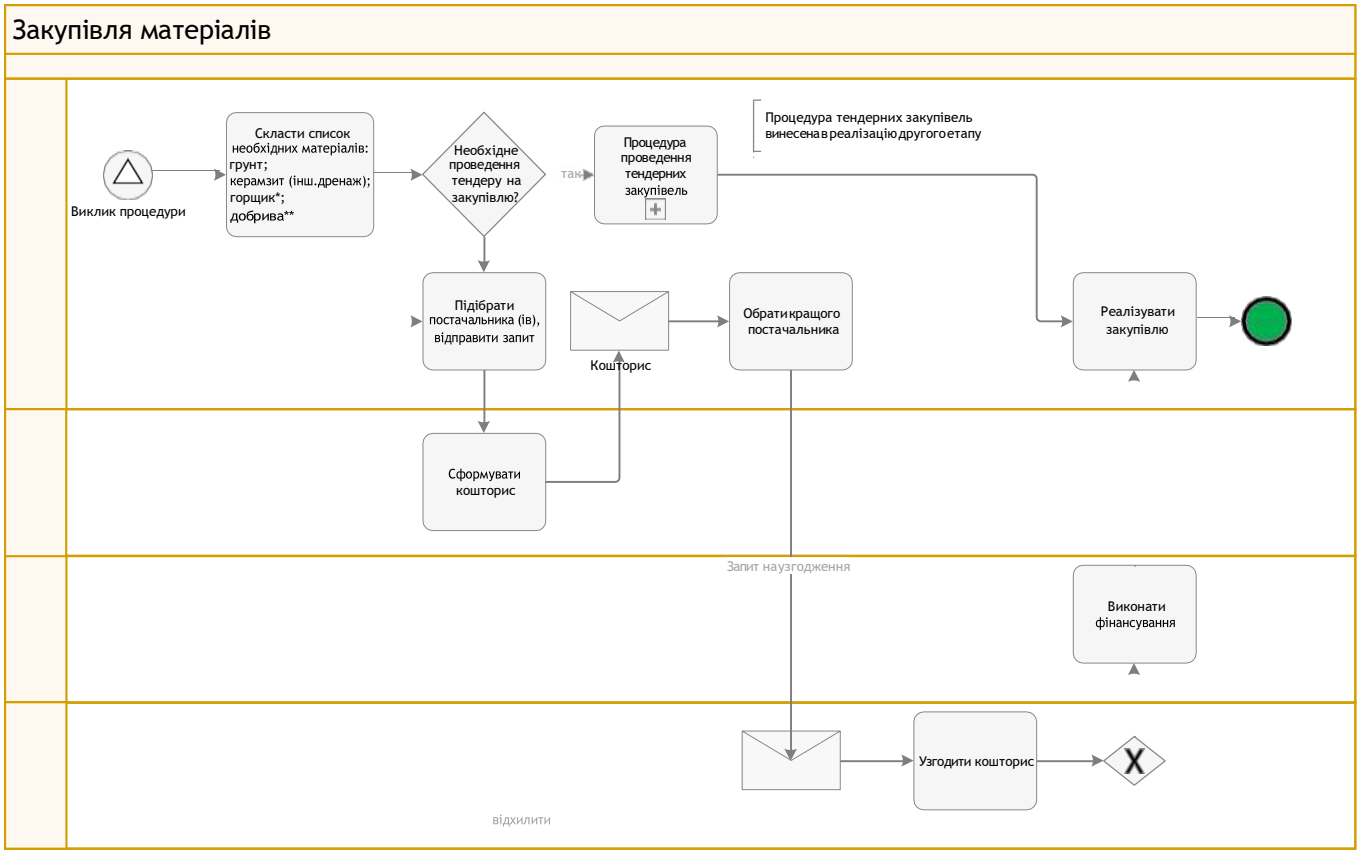


Рис. 2.11 Приклад бізнес-процесу «Закупівля матеріалу»

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Антонов А.П. Язык описания цифровых устройств Altera HDL. Практический курс. – М.: Издательское предприятие «РадиоСофт». 2002
2. Армстронг Дж.Р. Моделирование цифровых систем на языке VHDL. Пер. с англ. – М.: Мир. 1992. – 175 с.
3. Балик Н.Р. Бази даних MySQL. – Тернопіль: Навчальна книга – Богдан, 2010.- 158 с.
4. Белошапка В К Информационное моделирование в примерах и задачах. - Омск: Из-во ОГПИ, 1992.
5. Бройдо В.Л. Вычислительные системы, сети и телекоммуникации. [Текст] / В.Л. Бройдо. – СПб. : Питер, 2003. – 688 с.
6. Гмурман В.Е. Теория вероятностей и математическая статистика. М.: Высшая школа, 1978. –360с.
7. Губанов Д.А., Стешенко В.Б., Храпов В.Ю., Шипулин С.Н. Перспективы реализации алгоритмов цифровой фильтрации на основе ПЛИС фирмы ALTERA // Chip News, № 9–10, 1997, с. 26–33.
8. Іващук В.В. Автоматизація бізнес-процесів. Курс лекцій для студентів спеціальності Автоматизоване управління технологічними процесами та Комп’ютерно-інтегровані технологічні процеси та виробництва денної та заочної форм навчання. [Текст] / В.В. Іващук. – К. : НУХТ, 2007. – 76 с.
9. Інтегровані системи управління [Текст] : метод. рекомендації до практ. занять для студ. освіт. ступ. "Бакалавр" спец. 151 "Автоматизація та комп'ютерно-інтегровані технології" ден. та заоч. форм навч. / уклад. : В. Г. Трегуб, М. Д. Місюра ; Нац. ун-т харч. технол. — Київ : НУХТ, 2017. — 31 с. Режим доступу: http://library.nuft.edu.ua/ebook/file/100.72.pdf
10. Лахно В. А. Прикладна теорія цифрових автоматів: метод. вказівки до лабораторних робот для студ. спец. 7.091501 “Комп’ютерні системи та мережі”; Держ. закл. “Луган. нац. Ун-т імені Тараса Шевченка”.- Луганськ: Вид-во ДЗ «ЛНУ імені Тараса Шевченка», 2008. – 68 с.
11. Стешенко В.Б. ПЛИС фирмы “ALTERA”: элементная база, система проектирования и язы-ки описания аппаратуры. – М.: Издательский дом “Додэка-XXI”, 2002. – 576 с.
12. Тарасенко Р.О., Лисенко В.П., Касаткін Д.Ю. Інформаційні технології в системах якості, стандартизації та сертифікації. Київ, НАУ, 2002. – 82 с.
13. Точчи Рональд. Цифровые системы. Теория и практика/ Точчи Рональд, Дж., Уидмер, Нил, С.; Пер. с англ. - М.:Вильямс, 2004.-1024. с.

14. Трегуб В.Г. Основи комп'ютерно-інтегрованого керування [Текст]: Навчальний посібник / В.Г. Трегуб. – К. : НУХТ, 2005. – 191 с.
15. Andrews S., Tsochantaridis I., Hofmann T. Support vector machines for multiple-instance learning. In Advances in Neural Information Processing Systems (NIPS), volume 15, MIT Press, 2003, pages 561–568.
16. Angluin D. Queries and concept learning. Machine Learning, 2:319–342, 1988.
17. Angluin D. Queries revisited. In Proceedings of the International Conference on Algorithmic Learning Theory, pages 12–31. Springer-Verlag, 2001.
18. Balcan M.F., Beygelzimer A., Langford J. Agnostic active learning. In Proceedings of the International Conference on Machine Learning (ICML), pages 65–72. ACM Press, 2006.
19. Baldridge J. and Osborne M.. Active learning and the total cost of annotation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9–16. ACL Press, 2004.
20. Chapman B.L. Enhancing Interactivity and Productivity Through Object-Oriented Authoring: An Instructional Designer's Perspective // Journal of Interactive Instruction Development, 1994, №7(2), pp.3-11.
21. Craven M., Andrzejewski D., Zhu X. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In Proceedings of the International Conference on Machine Learning (ICML), pages 25–32. ACM Press, 2009.
22. Dougiamas Martin. Computer Science и Education диссертация (Ph.D.) "The use of Open Source software to support a social constructionist epistemology of teaching and learning within Internet-based communities of reflective inquiry" http://moodle.udec.ntu-kpi.kiev.ua/moodle/mod/ resource/view.php?inpopup=true&id=1124
23. Fishwick P.A. Computer Simulation: Grouth through Extension. - 1994. - http://www.cis.ufl.edu/7Efishwick/paper/paper.html
24. Flexible Distant Learning // Communicstion and Information Technologies (CIT) Course. Applied Module for Teachers. Chapter 1. - 1999. - http://dlab.kiev.ua/cit/ap_ch1/c1l12_1.htm.
25. Mamitsuka H. and Abe Query learning strategies using boosting and bagging. In Proceedings of the International Conference on Machine Learning (ICML), Morgan Kaufmann, 1998. pages 1–9.
26. Nyberg E., Arora S., Ros´e C.P.. Estimating annotation cost for active learning in a multi-annotator environment. In Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing, pages 18–26. ACL Press, 2009.
27. Palmer A., Baldridge J. How well does active learning actually work? Timebased evaluation of cost-reduction strategies for language documentation. In

Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 296–305. ACL Press, 2009.

28. SCORM. Shareable Content Object Reference Model. 2d Edition. – Advanced Distributed Learning, 2004. 96 p.
29. Skinner B.F. The science of learning and art of teaching. // Harward Education Review, Spring, 24, 1954. p. 86-97.)
30. W.Lee, D.Owens Multimedia-Based Instructional Design: Computer-Based Training, Web-Based Training, and Distance Learning. Pfeiffer, 2000.
31. Wortman J., Balcan M.F., Hanneke S. The true sample complexity of active learning. In Proceedings of the Conference on Learning Theory (COLT), pages 45–56. Springer, 2008.

ДОДАТКИ

Додаток А.

Приклади 1.

Мовна конструкція для синтезування простої логіки з однобітових входами. Тут і далі текст на Verilog наводиться моноширінним шрифтом.

```
module logic_operators
(Input a, b, c, d, // оголошення вхідних портів або входів
output y); // оголошення вихідних портів або виходів
wire e; // оголошення внутрішньої ланцюга,
//необхідної для роботи
// схеми і не є вхідним / вихідним портом
assign y = (a & b) | e;
assign e = (c | d);
endmodule
```

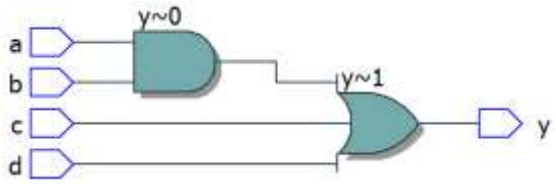


Рис. 1. Реалізація прикладу 1

Приклади 2.

Реалізація логічних регістрів

```
module register_inference
(Input d, clk, clear, preset, load,
  output reg q1, q2, q3, q4, q5);

 // register with active-low clock
always @ (negedge clk) q1 = d;
      // register with active-high clock and asynchronous clear always @ (posedge clk,
posedge clear)
       if (clear) q2 = 0;
       else q2 = d;
     // register with active-high clock and asynchronous preset always @ (posedge clk,
posedge preset)
       if (preset) q3 = 1;
       else q3 = d;
// register with active-high clock and synchronous load
always @ (posedge clk)
       if (load) q4 = d;
       else q4 = q4;
// register with active-low clock and asynchronous clear and
```

```
preset
always @ (negedge clk, posedge clear, posedge preset)
        if (clear) q5 = 0;
        else if (preset) q5 = 1;
             else q5 = d;
endmodule
```

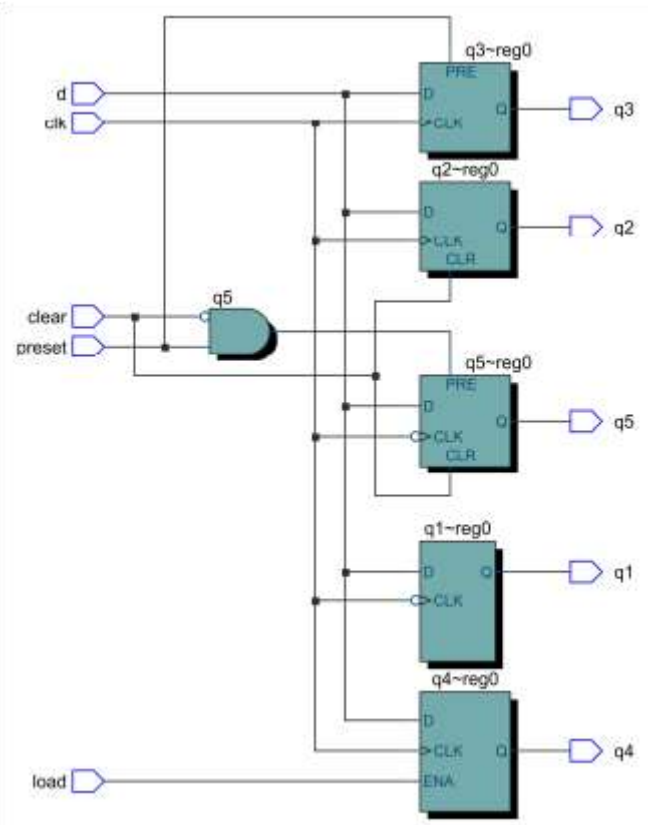


Рис. 2. Реалізація прикладу 2

Приклад 3.
Verilog реалізація логічних лічильників

```
module counters
(Input [7: 0] d,
  input clk, enable, clear, load, up_down,
  output reg [7: 0] qa, qb, qc, qd, qe, qf);

    integer direction;
                        // An enable counter
```

```
always @ (posedge clk)
          if (enable) qa = qa + 1;

                    // A synchronous load counter
always @ (posedge clk)
    begin
        if (load) qb = d;
        else qb = qb + 1;
    end
                    // A synchronous clear counter
always @ (posedge clk)
      begin
         if (clear) qc = 0;
         else qc = qc + 1;
      end
                     // An up / down counter
always @ (posedge clk)
      begin if (up_down) direction = 1;
           else direction = -1;
           qd = qd + direction;
      end
                   // A synchronous load clear counter
always @ (posedge clk)
      begin

         if (clear) qe = 0;
         else if (load) qe = d;
              else qe = qe + 1;
      end
            // A synchronous load enable up / down counter always @ (posedge clk)
     begin
        if (up_down) direction = 1;
        else direction = -1;

        if (load) qf = d;
        else if (enable)
            qf = qf + direction;
     end
endmodule
```

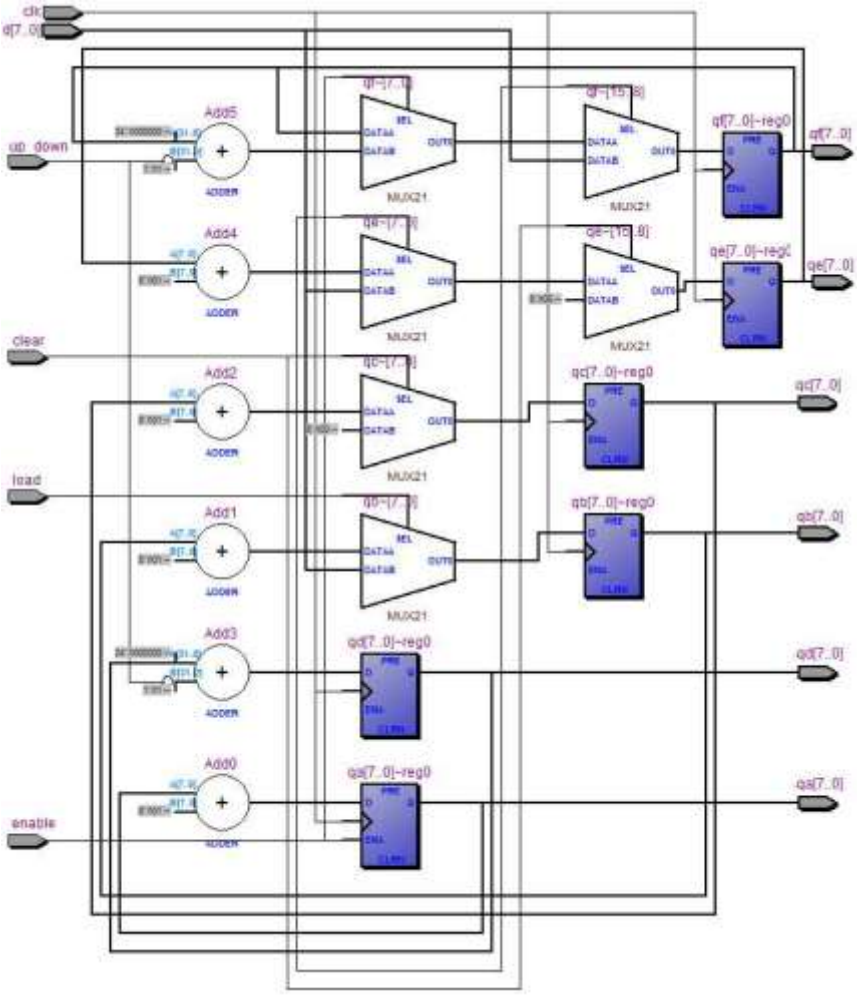


Рис. 3. Реалізація прикладу 3

Приклад 4.
Verilog реалізація подільника частоти або (non-free-running) лічильника з модулем рахунку 12.

```
module divider11
(Input clk, reset,
   output reg clkdiv11);
   reg [3: 0] cnt;
   reg n;

always @ (posedge clk, posedge reset)
//
     begin
        if (reset)
             begin
              cnt = 0; n = 0;
```

```
            end
        else cnt = cnt + 1;

        if (cnt == 11) n = 1;
        else n = 0;

        if (n == 1) cnt = 0;
      end

always @ (n) clkdiv11 = n;
// еквівалентно assign clkdiv11 = n;
endmodule
```

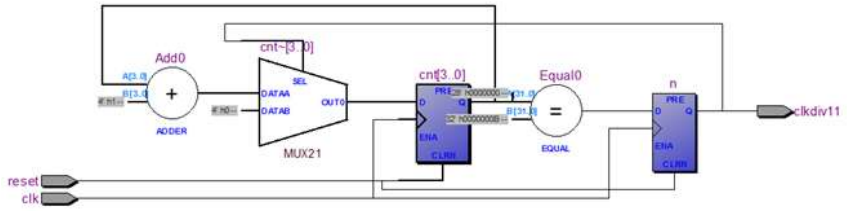


Рис. 4. Реалізація прикладу 4

Приклад 5.

Verilog-опис таймера або віднімає лічильника з модулем рахунку, код якого міститься на вхідний шині data.

```
module timer
(Input clk, load, enable,
   input [15: 0] data,
   output reg timeout);
  reg [15: 0] cnt;
always @ (posedge clk)
        begin
          if (load &! enable) cnt = data;
          else if (! load & enable) cnt = cnt - 1;
               else cnt = cnt;

         if (cnt == 0) timeout = 1;
         else timeout = 0;
        end
endmodule
```

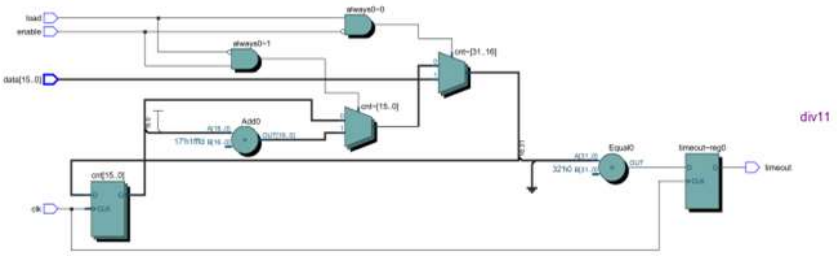


Рис. 5. Реалізація прикладу 5

Приклад 6.

Verilog-опис одновібратора.

```
module monostable_multivibrator
(Input clk, reset,
output reg reset_short);
always @ (posedge clk)
   if (reset_short == 1) reset_short <= 0; else if (reset == 1) reset_short <= 1; else
reset_short <= 0;
endmodule
```

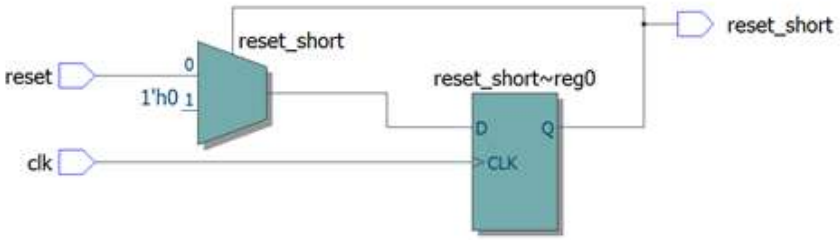


Рис. 6. Реалізація прикладу 6

Приклад 7. FSM-машина типу Moore

```
module moore1 (d, a, clk)
(Output reg d,
  input a,
  input clk,
  input e);

    reg b; // вихід логіки, яка визначає наступний стан reg c; // поточний стан
```

```
      // [..] позначають розмір вихідний шини
function [..] next_state_logic;
    input a, c;
  begin
      next_state_logic = a + c;
  end
endfunction

function [..] output_logic;
  input c, e;
  begin
      output_logic = c + e;
  end
endfunction

// генерація наступного стану
always @ (a or c) begin
                b = next_state_logic (a, c);
               end
// system output
always @ (c or e) begin
            d = output_logic (c, e);
          end
// state transition
always @ (posedge clk) begin
                     c = b;
                   end
endmodule
```

Більш компактне опис цієї архітектури могло бути написано таким чином:

```
Verilog-опис FSM-машини типу Moore
module moore2
(Output reg d,
 input a,
 input clk,
 input e);
   reg c; // поточний стан

      // [..] позначають розмір вихідний шини
function [..] next_state_logic;
   input a, c;
 begin
      next_state_logic = a + c;
```

```
 end
endfunction

function [..] output_logic;
 input c, e;
 begin
      output_logic = c + e;
 end
endfunction

// system output
always @ (c) begin
            d = output_logic (c);
          end
// state transition
always @ (posedge clk) begin
                    c = next_state_logic (a, c); end
endmodule // moore2
```

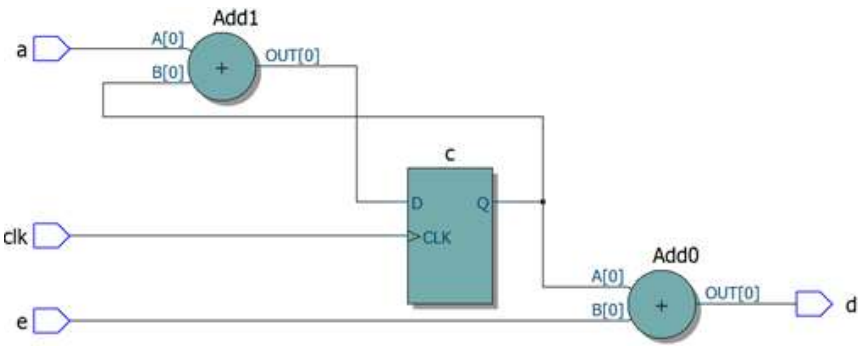


Рис. 7. Реалізація прикладу 7

Результат синтезування схеми цих двох кодів буде однаковий.

Приклад 8. Реалізація дешифратора адреси

```
module address_decoder
(Input [9: 0] address,
  output reg select0, select1, select2);
  always @ (address)
    begin // first segment
      if (address> = 0 && address <= 255) select0 = 1; else select0 = 0; // second
segment
```

```
    if (address> = 256 && address <= 511) select1 = 1; else select1 = 0; // third
segment
    if (address> = 512) select2 = 1;
    else select2 = 0;
   end
endmodule
```

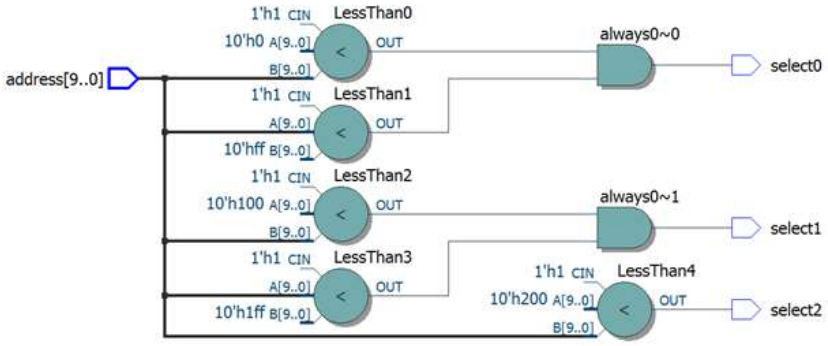


Рис. 8. Реалізація прикладу 8

Приклад 9. Реалізація простого АЛУ

```
module alu
# (Parameter addab = 4'b0000, inca = 4'b0001, incb = 4'b0010, andab = 4'b0011, orab
= 4'b0100, nega = 4'b0101, shal = 4'b0110, shar = 4'b0111 ,
            passa = 4'b1000, passb = 4'b1001)
(Input [7: 0] a, b,
   input [3: 0] opsel,
   output reg [7: 0] f);
always @ (a or b or opsel)
   begin
      case (opsel)
          addab: f = a + b;
           inca: f = a + 1;
           incb: f = b + 1;
          andab: f = a & b;
           orab: f = a | b;
           nega: f =! a;
           shal: f = a << 1; shar: f = a >> 1; passa: f = a;
          passb: f = b;
        default: f = 8'bX;
      endcase
   end
endmodule
```

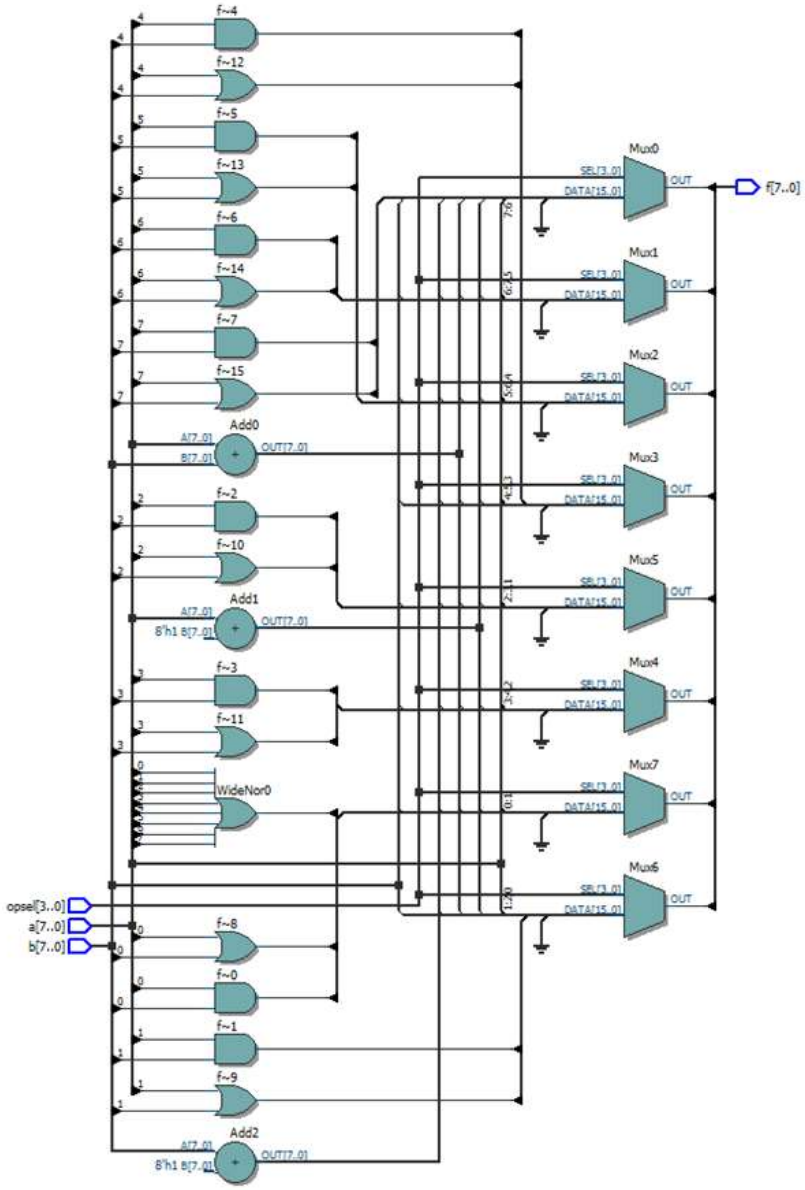


Рис. 9. Реалізація прикладу 9

Приклад 10. Лінія затримки 8 × 64 з відводами для зовнішнього використання на 16, 32, 48 і 64 осередках

```
module shift_8x64_taps (
  input clk, shift,
  input [7: 0] sr_in,
  output [7: 0] sr_tap_one, sr_tap_two, sr_tap_three, sr_out);

    reg [7: 0] sr [63: 0];
    integer n;

always @ (posedge clk)
    begin
```

```
    if (shift == 1'b1)
    begin
      for (n = 63; n> 0; n = n-1)
      begin
        sr [n] <= sr [n-1];
      end
      sr [0] <= sr_in;
    end
  end
        // виходи вхідний послідовності sr_in
assign sr_tap_one = sr [15]; // з затримкою на 16 тактів assign sr_tap_two = sr [31];
// з затримкою на 32 такту assign sr_tap_three = sr [47]; // з затримкою на 48
тактів assign sr_out = sr [63]; // з затримкою на 64 такту

endmodule
```

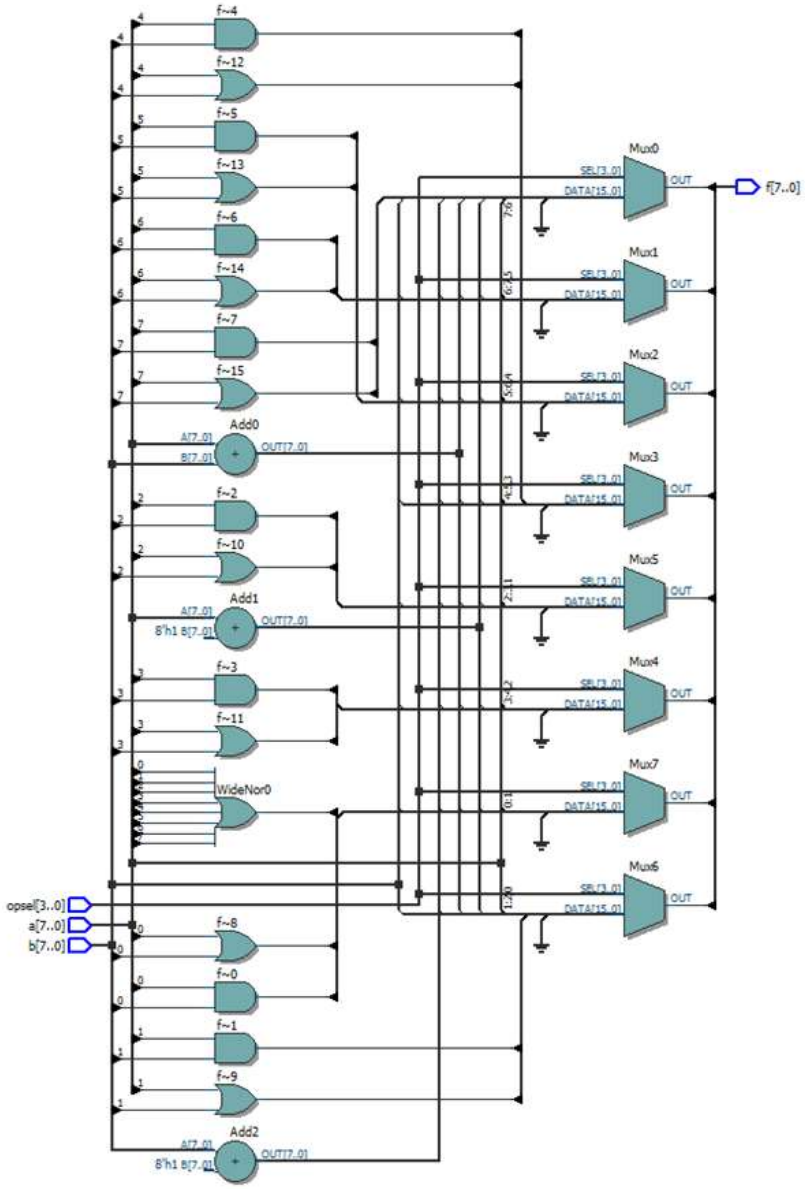
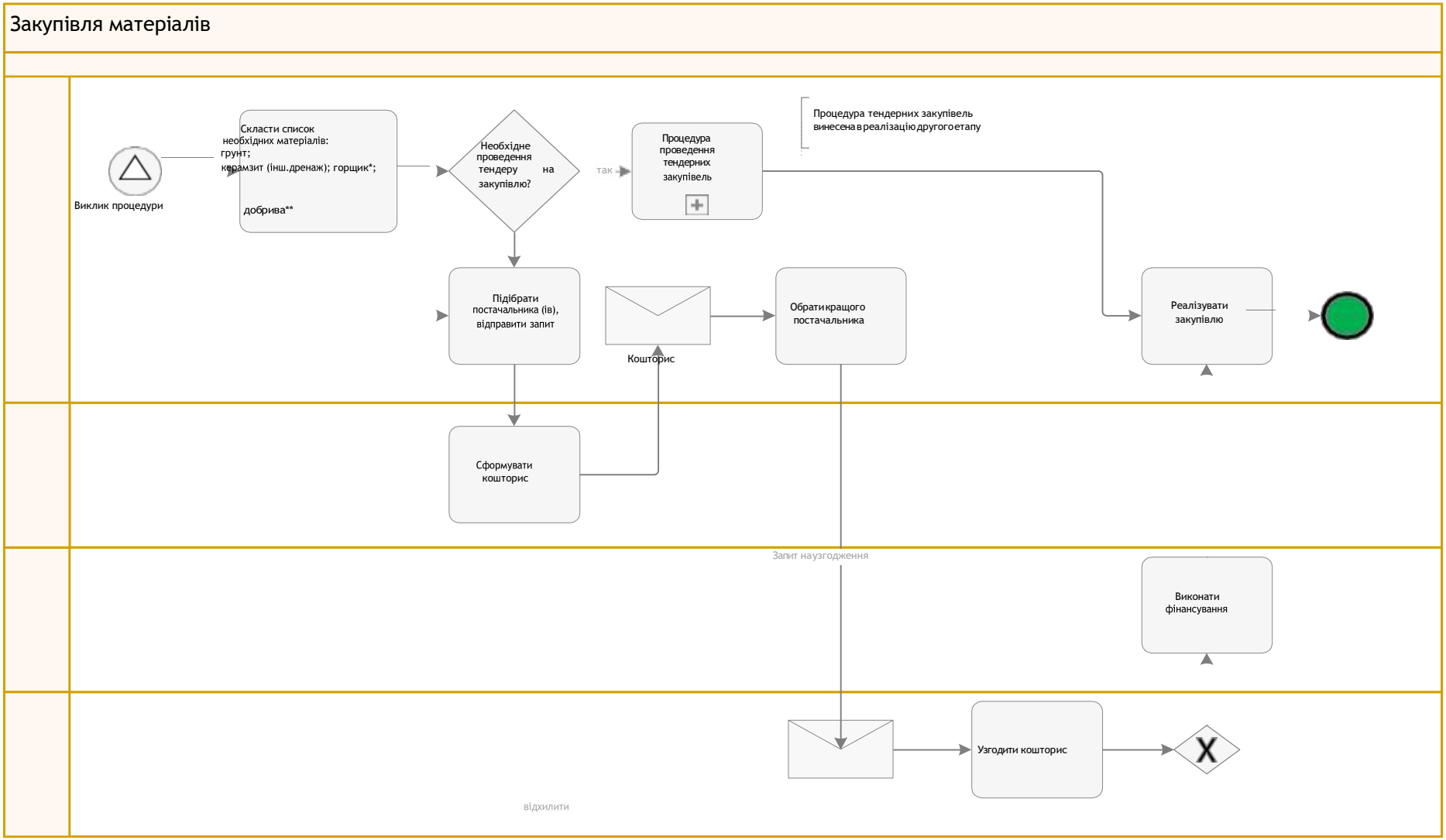


Рис. 10. Реалізація прикладу 10

Додаток Б

Приклад бізнес-процесу «Закупівля матеріалу»



Додаток В

Підключення стенду DE0 до напруги живлення та початок виконання лабораторної роботи відбувається тільки з дозволу викладача.

Під час підготовки до лабораторної роботи студент повинен вивчити устрій, структурну організацію та органи керування лабораторним стендом DE0, засвоїти принцип функціонування САПР QUARTUS II та процедуру завантаження логічних схем до стенду DE0 і перевірки функціонування пристроїв, що досліджуються.

В ході виконання лабораторної роботи необхідно створити проекти для плати «Суматор», за допомогою яких виконується дослідження однобітних суматорів. В якості вхідних даних буде використовуватися два однобітних операнди та вхідний перенос до суматора. На платі стенду DE0 для підключення вхідних операндів використовується три кнопки (*a, b, e*). Відображення результату (*S, E*) арифметичної операції додавання здійснюється за допомогою світлодіодів, які входять до складу блока індикації стенда DE0.

Відповідно до заданого варіанту завдань необхідно виконати синтез вказаного пристрою в заданому базисі логічних елементів. Варіанти завдань приведені в таблицы:

Варіанти елементного базису

Номер варіанта	Базис ЛЕ	Номер варіанта	Базис ЛЕ
1	2І-НІ	19	4І-НІ
2	2АБО-НІ	20	3І, 2АБО-НІ
3	3І-НІ	21	2І-НІ, 2АБО-НІ
4	2І, 2АБО-НІ	22	3І-НІ, 2АБО-НІ
5	2І, 3І-НІ	23	3І, 2І-Н
6	2АБО, 2І-НІ	24	2АБО, 3І-НІ
7	3АБО-НІ	25	4АБО-НІ
8	3АБО, 2І-НІ	26	3АБО, 3І-НІ
9	2XOR, 2І-НІ	27	2XNOR, 2І-НІ
10	2XOR, 2АБО-НІ	28	2XNOR, 2АБО-НІ
11	2XOR, 2І, 2АБО-НІ	29	2XNOR, 2І, 2АБО-НІ
12	2XOR, 2АБО, 2І-НІ	30	2XNOR, 2АБО, 2І-НІ
13	булевий	31	4І-НІ, 2АБО-НІ
14	3АБО-НІ, 3І-НІ	32	3АБО-НІ, 2І-НІ
15	2І-2І-2І-3АБО-НІ	33	2І-2І-2І-3І-4АБО-НІ
16	2І-2І-2АБО-НІ	34	3І-3І-2АБО-НІ
17	2І-2І-2АБО-НІ, 3І	35	2І-2І-2АБО-НІ, 2АБО
18	2XOR, 2І-2І-2АБО-НІ	36	2АБО, 2АБО-НІ

Якщо в якості базису задано кілька логічних функцій, то під час реалізації схеми необхідно використовувати всі логічні функції з заданого базису.

Номер варіанту завдання видається викладачем. Якщо номер заданого варіанту (**V**), більше за максимальний номер з відповідної таблиці варіантів, то номер завдання необхідно визначити за формулою (**V mod n) + 1**, де **n** – кількість варіантів в таблиці; **mod** – функція визначення залишку під час ділення чисел.

Після процедури синтезу виконати моделювання функціонування заданих пристроїв в середовищі САПР QUARTUS II та пересвідчитись, що реакція пристроїв на вхідні сигнали відповідає таблиці істинності.

Хід виконання роботи

Кожний дослід лабораторної роботи виконується в наступній послідовності:

- з дозволу викладача виконати завантаження схеми пристрою до лабораторного стенда DE0;
- задаючи на вхід пристрою, що досліджується, вхідні сигнали відповідно до таблиці істинності, зафіксувати за допомогою блоку індикації стенду реакцію пристрою на кожну комбінацію вхідних сигналів.

Послідовність дослідів під час виконання лабораторної роботи приведена нижче:

1. Виконати дослідження статичного режиму роботи напівсуматора в заданому базисі логічних елементів.
2. Виконати дослідження статичного режиму роботи напівіднімача в заданому базисі логічних елементів.
3. Виконати дослідження статичного режиму роботи канонічного суматора в заданому базисі логічних елементів.
4. Виконати дослідження статичного режиму роботи канонічного суматора на базі дешифратора 3→8.
5. Виконати дослідження статичного режиму роботи канонічного суматора на базі мультиплексора 8→1.
6. Виконати дослідження статичного режиму роботи канонічного суматора на базі мультиплексора 4→1.
7. Виконати дослідження статичного режиму роботи мінімального суматора з трактом розповсюдження переносу ($e \to \overline{E}$) в заданому базисі логічних елементів.
8. Виконати дослідження статичного режиму роботи мінімального суматора з трактом розповсюдження переносу $(\overline{e} \to E)$ в заданому базисі логічних елементів.

Зміст звіту

1. Титульний аркуш.
2. Завдання.
3. Синтез напівсуматора та перетворення логічних виразів до заданого базису.
4. Синтез напіввіднімача та перетворення логічних виразів до заданого базису.
5. Синтез канонічного суматора в заданому базисі, мінімізація і перетворення логічних виразів до заданого базису.

6. Синтез канонічного суматора на базі дешифратора 3→8. Використовувати результати мінімізації з попередніх дослідів.
7. Синтез канонічного суматора на базі мультиплексора 8→1. Використовувати результати мінімізації з попередніх дослідів.
8. Синтез канонічного суматора на базі мультиплексора 4→1. Використовувати результати мінімізації з попередніх дослідів.
9. Синтез мінімального суматора з трактом розповсюдження переносу ($e \to \overline{E}$) в заданому базисі логічних елементів. Використовувати результати мінімізації з попередніх дослідів.
10. Синтез мінімального суматора з трактом розповсюдження переносу ($\overline{e} \to E$) в заданому базисі логічних елементів. Використовувати результати мінімізації з попередніх дослідів.
11. Виводи за результатами проведених дослідів.

Питання для самоперевірки

9. Назвіть етапи проектування комбінаційної схеми із застосуванням ПЛІС?
10. Як проводиться підключення комбінаційної схеми до зовнішніх контактів стенду?
11. Яким чином визначити реакцію пристрою на вхідні сигнали?
12. Як задавати вхідні сигнали на вхід пристрою?
13. Для чого використовується блок індикації стенду?
14. Для чого використовується задавальна частина стенду?
15. Як виконати моделювання логічної схеми в середовищі системи MicroCap?
16. Як виконати моделювання логічної схеми в середовищі САПР QUARTUS II?
17. Дайте визначення напівсуматора.
18. Приведіть таблицю істинності напівсуматора.
19. Приведіть логічні вирази, що характеризують функціонування напівсуматорів.
20. Наведіть формули напівсуси і вихідного переносу в булевому базисі.
21. Для чого використовуються напівсуматори?
22. За допомогою яких типових пристроїв можна виконувати мікрооперація «інкремент»?
23. Наведіть ДДНФ напівсуми.
24. Наведіть ДКНФ напівсуми.
25. Наведіть формули напівсуси і вихідного переносу в базисі 2І-НІ.
26. Наведіть формули напівсуси і вихідного переносу в базисі 2АБО-НІ.
27. Наведіть формули напівсуси в базисі 2XNOR.
28. Наведіть формули напівсуси в базисі 2XOR.
29. Дайте визначення напівнімача.

30. Приведіть таблицю істинності напівіднімача.
31. Приведіть логічні вирази, що характеризують функціонування напівіднімачів.
32. Наведіть формули напіврізниці і вихідної позики в булевому базисі.
33. Для чого використовуються напівіднімачі?
34. За допомогою яких типових пристроїв можна виконувати мікрооперація «декремент»?
35. Наведіть ДДНФ напіврізниці.
36. Наведіть ДКНФ напіврізниці.
37. Наведіть формули напіврізниці і вихідної позики в базисі 2І-НІ.
38. Наведіть формули напіврізниці і вихідної позики в базисі 2АБО-НІ.
39. Наведіть формули напіврізниці в базисі 2XNOR.
40. Наведіть формули напіврізниці в базисі 2XOR.
41. Побудуйте логічні схеми напівсуматора на базі дешифратора 2→4.
42. Побудуйте логічні схеми напівсуматора на базі мультиплексора 4→1.
43. Побудуйте логічні схеми напівсуматора на базі мультиплексора 2→1.
44. Побудуйте логічні схеми напіввіднімача на базі дешифратора 2→4.
45. Побудуйте логічні схеми напіввіднімача на базі мультиплексора 4→1.
46. Побудуйте логічні схеми напіввіднімача на базі мультиплексора 2→1.
47. Виконати синтез логічної схеми канонічного суматора в базисі 3І-НІ та 2І-НІ, з використанням ДДНФ.
48. Виконати синтез логічної схеми канонічного суматора в базисі 3І-НІ та 2І-НІ, з використанням ДКНФ.
49. Виконати синтез логічної схеми канонічного суматора в базисі 3АБО-НІ та 2АБО-НІ, з використанням ДДНФ.
50. Виконати синтез логічної схеми канонічного суматора в базисі 3АБО-НІ та 2АБО-НІ, з використанням ДКНФ.
51. Побудуйте канонічний суматор на базі дешифраторів 2→4.
52. Побудуйте канонічний суматор на базі дешифратора 3→8.
53. Побудуйте канонічний суматор на базі мультиплексорів 8→1.
54. Побудуйте канонічний суматор на базі мультиплексорів 4→1.
55. Побудуйте канонічний суматор на базі мультиплексорів 2→1, використовуючи каскадне з'єднання цих мультиплексорів.
56. Побудуйте канонічний суматор на базі дешифратора і мультиплексорів 2→1.
57. Виконайте порівняння апаратних витрат на реалізацію канонічного суматора в різних базисах.
58. Доведіть коректність логічних виразів:

$$\overline{S} = \overline{a}be + a\overline{b}e + ab\overline{e} + \overline{a}\overline{b}\overline{e};$$

$\overline{E} = \bar{a}\bar{b}\bar{e} + \bar{a}b\bar{e} + \bar{a}\bar{b}e + \bar{a}be$;

$\overline{S} = \overline{\bar{a}\bar{b}\bar{e} + \bar{a}b\bar{e} + a\bar{b}\bar{e} + abe}$;

$\overline{E} = \overline{\bar{a}be + a\bar{b}e + ab\bar{e} + abe}$;

$E = \bar{a}be + a\bar{b}e + ab\bar{e} + abe$.

59. Доведіть коректність логічних виразів:

$S = \overline{\bar{a}be + a\bar{b}e + ab\bar{e} + \bar{a}\bar{b}\bar{e}}$;

$E = \overline{\bar{a}\bar{b}\bar{e} + \bar{a}b\bar{e} + a\bar{b}\bar{e} + \bar{a}\bar{b}\bar{e}}$;

$S = \bar{a}\bar{b}e + \bar{a}b\bar{e} + a\bar{b}\bar{e} + abe$;

60. Дайте визначення однобітного суматора.
61. Для чого використовуються суматори?
62. Поясніть, як створити таблицю істинності однорозрядного двійкового суматора?
63. Поясніть, в чому полягає властивість самоподвійності логічних функцій.
64. Доведіть самоподвійність функції суми.
65. Доведіть самоподвійність функції переносу.
66. Яка операція виконується за допомогою суматорів?
67. Визначити реакцію суматора при використанні інверсних значень всіх вхідних змінних.
68. Для чого викорсовується індикатор інверсії на вході УГП суматора?
69. За якою формулою визначається результат додавання в поточному біті суматора?
70. Приведіть УГП однобітного суматора з прямими виводами доданків і суми та інверсними виводами переносів.
71. Доведіть коректність логічних виразів:

$\overline{a \oplus b} = \bar{a}\bar{b} + ab;$ $\quad \overline{\bar{a} \oplus \bar{b}} = \overline{a \oplus b};$

$a \oplus \bar{b} = \overline{a \oplus b};$ $\quad 1 \oplus a = \bar{a};$ $\quad a \oplus \bar{a} = 1;$

$a \oplus b = (a + b)\overline{ab};$ $\quad a \oplus b = \overline{(a + b)} \oplus \overline{ab};$

72. Виконати реалізацію логічної функції $S = a \oplus b \oplus e$ в базисі 3І- НІ.
73. Виконати реалізацію логічної функції $S = (\overline{a \oplus b}) \oplus \bar{e}$ в базисі І-АБО-НІ.
74. Виконати порівняння апаратних витрат на побудову мінімальних суматорів в різних базисах.
75. Визначити логічний вираз функції переносу мінімального суматора з використанням ДНФ.
76. Визначити логічний вираз функції переносу мінімального суматора з використанням КНФ.
77. Привести карту Карно та виконати мінімізацію функції переносу мінімального суматора.

78. Приведіть і поясніть формулу для реалізації переносу мінімального суматора з трактом $\bar{e} \rightarrow E$.
79. Приведіть і поясніть формулу для реалізації переносу мінімального суматора з трактом $e \rightarrow \overline{E}$.

Додаток Г

Розробка та дослідження багаторозрядних ариффметичних пристроїв в середовищі QUARTUS II

Мета роботи: отримання і узагальнення теоретичних знань, придбання вмінь і навичок під час вирішення завдань проектування та аналізу характеристк багаторозрядних двійкових суматорів:

- *вивчення і засвоєння принципів функціонування та синтезу багаторозрядних двійкових суматорів;*
- *вивчення і засвоєння принципів структурної організації багаторозрядних двійкових суматорів з метою прискореня виконання арифметичної операції додавання;*
- *закріплення практичних навичок під час виконання процедури синтезу логічних схем багаторозрядних двійкрвих суматорів;*
- *використання САПР QUARTUS II та дослідження функціонування двійкових багаторозрядних суматорів за допомогою лабораторного стенда DE0 для визначення статичних і динамічних характеристик досліджуваних арифметичних пристроїв;*
- *придбання практичних навичок з визначення та вимірювання динамічних параметрів багаторозрядних двійкових суматорів.*

Стислі теоретичні відомості

Багаторозрядні суматори використовуються для виконання арифметичних операцій і входять до складу арифметико-логічних пристроїв (АЛП) комп'ютерних систем. При цьому мікрооперація додавання є найбільш поширеною серед арифметичних операцій і використовується як складова частина при виконанні більш складних арифметичних операцій (множення, ділення, тощо). Необхідно також відмітити, що мікрооперація додавання є найбільш довгою за тривалістю серед мікрооперацій, що виконуються процесором комп'ютерних систем. В зв'язку з цим розвитку методів прискорення виконання мікрооперації додавання і, таким чином, підвищенню швидкодії арифметичних пристроїв в теорії проектування комп'ютерних систем приділяється дуже велика увага.

Швидкодія двійкових суматорів залежить від способу організації переносу між розрядами суматора. Серед багаторозрядних суматорів з різними способами організації переносу можна виділити:

- багаторозрядні арифметичні пристрої з послідовним трактом розповсюдження переносу між розрядами;
- багаторозрядні суматори з паралельним трактом розповсюдження переносу між розрядами;

- багаторозрядні суматори з груповим переносом з використанням послідовного переносу між групами (послідовні групові суматори);
- багаторозрядні суматори з груповим переносом з використанням паралельного переносу між групами.

Розглянемо реалізацію багаторозрядного напівсуматора. Такі пристрої будуються на базі одноразрядних напівсуматорів і забезпечують апаратну реалізацію мікрооперації «інкремент».

Найпростішим багаторозрядним напівсуматором є напівсуматор з послідовним трактом розповсюдження переносу між розрядами. Для отримання логічної схеми такого напівсуматора необхідно використовувати послідовне з'єднання одноразрядних напівсуматорів. Для цього треба з'єднати вивід вихідного переносу *E* поточного розряду з виводом вхідного переносу *e* наступного розряду напівсуматора. Логічна схема n-розрядного напівсуматора з послідовним переносом приведена на рис..1.

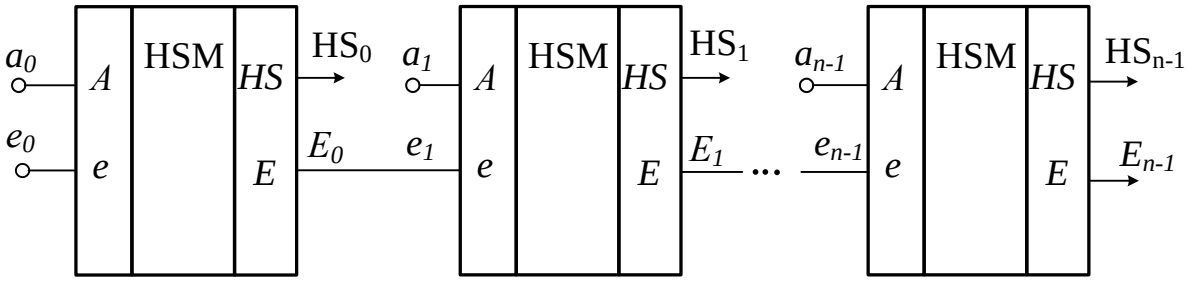


Рисунок .1 – Функціональна схема n-розрядного напівсуматора з послідовним трактом розповсюдження переносу

На ис.1.1 використовуються такі позначення:

- $A = a_{n-1}, \ldots, a_0$ – *n*-розрядний операнд *A*, де a_i – значення *i*-того біту операнду;
- e_0 – сигнал вхідного переносу до нульового розряду напівсуматора;
- $HS_{n-1}, \ldots, HS_0$ – *n*-розрядна напівсума;
- E_{n-1} – сигнал вихідного переносу з *n-1* розряду напівсуматора.

Очевидно, що для виконання мікрооперації «інкремент» сигнал вхідного переносу повинен приймати одиничне значення ($e_0 = 1$). При нульовому значенні цього сигналу напівсуматор формує на своїх виходах значення вхідного операнду. Сигнал вихідного переносу E_{n-1} може використовуватися для фіксації переповнення вмісту напівсуматора або для нарощування розрядності цього пристрою.

Приведена на рис..1 схема реалізує мікрооперацію «інкремент», тобто на виході пристрою формується код $A+e_0$.

На рис..2 приведена логічна схема для моделювання трьохбітного напівсуматора з послідовним трактом розповсюдження переносу в середовищі

MicroCap. Результати моделювання статичного режиму функціонування цього пристрою приведено на рис..3.

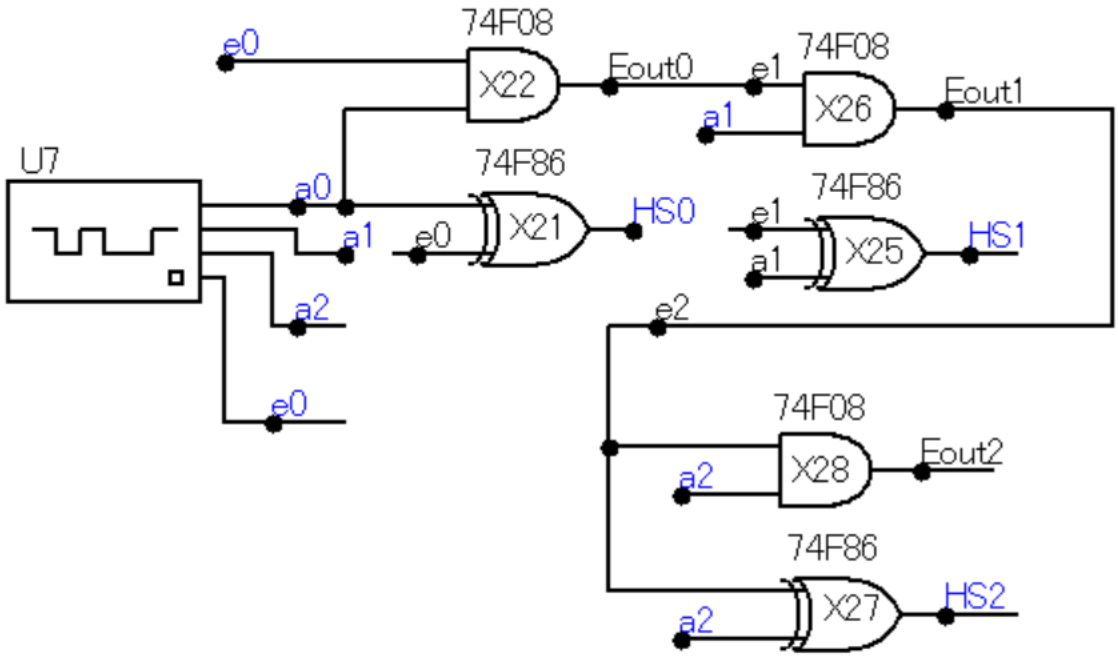


Рисунок .2 – Схема для моделювання 3-бітного напівсуматора з послідовним переносом в середовищі MicroCap

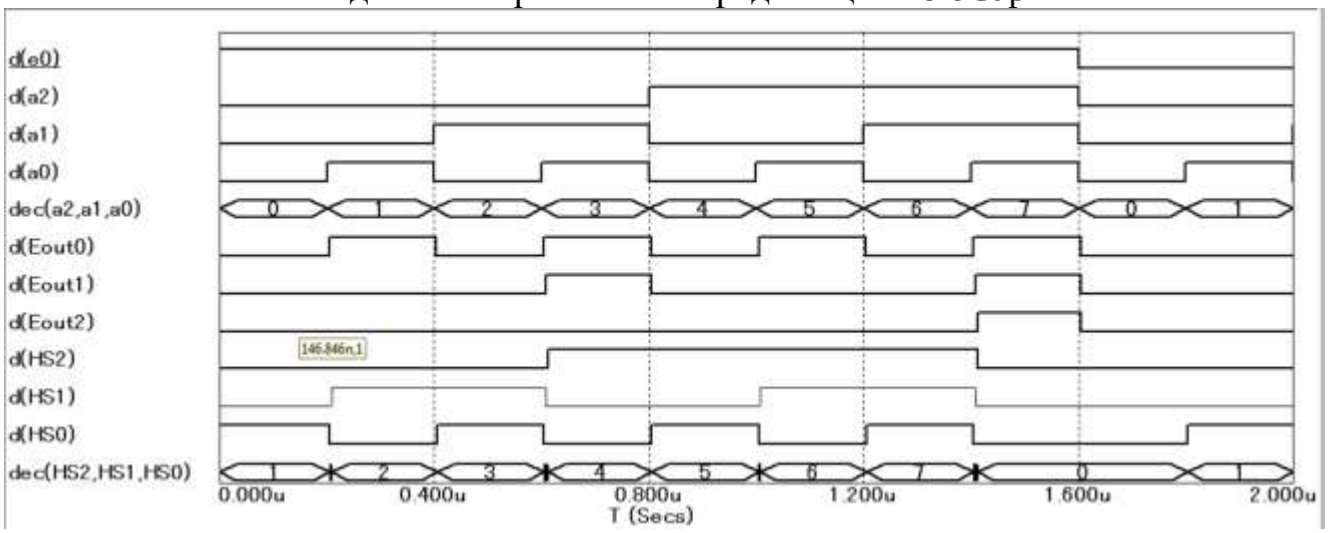


Рисунок .3 – Результати моделювання статичного режиму функціонування 3-бітного напівсуматора з послідовним переносом в середовищі MicroCap

В загальному випадку швидкодія арифметичних пристроїв з послідовним трактом розповсюдження переносу залежить від значень вхідних операндів. В найгіршому випадку з точки зору швидкодії цей параметр має найменше значення, якщо перенос розповсюджується через всі розряди пристрою. При цьому швидкодія напівсуматора характеризується часом спрацьовування

пристрою t_{HSM}, тобто часом розповсюдження переносу через всі розряди напівсуматор. Таким чином, час спрацьовування напівсуматора може бути визначений за формулою $t_{HSM} = (n\text{-}1)\cdot t_E + t_{HS}$, де n – кількість розрядів напівсуматора; t_E – час формування вихідного переносу; t_{HS} – час формування напівсуми. Очевидно, що при $t_E = t_{HS} = t$ час спрацьовування напівсуматора визначається за виразом $n \cdot t$. При цьому необхідно відмітити, що функціонування пристрою характеризується найменшою швидкодією, якщо значення операнду дорівнює $A = 2^{n-1}$ (наприклад, для трьохрозрдного напівсуматора двійковий код *111* на входах пристрою приводить до розповсюдження переносу через всі розряди)

Результати моделювання динамічного режиму функціонування трьохрозрядного пристрою в середовищі MicroCap приведено на рис..4.

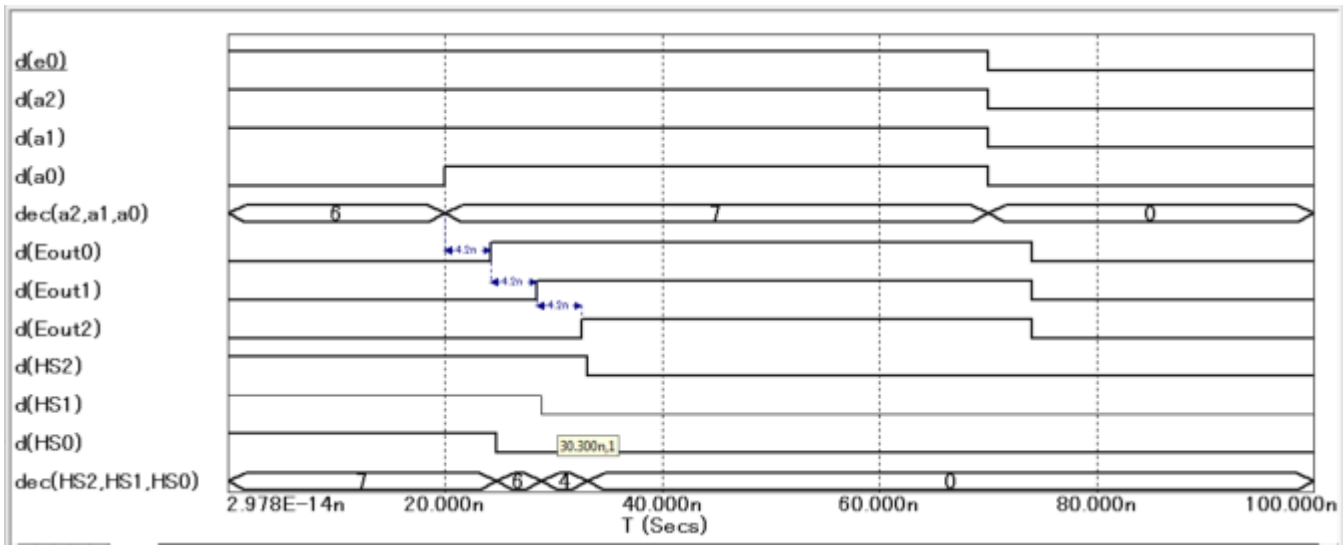


Рисунок .4 – Результати моделювання днамічного режиму функціонування 3-бітного напівсуматора з послідовним переносом в середовищі MicroCap

На рис..4 показано спрацьовування напівсуматора при виконанні інкременту коду *111*. Як було відзначено вище при цьому вхідному коді напівсуматор характеризується найнижчою швидкодією. На рисунку можна спостерігати поступове спрацьовування сигналів переносу $E_{out0}(E_0)$, $E_{out1}(E_1)$, $E_{out2}(E_2)$ через *4,2нс* (на рисунку показано виносками). Таким же чином спостерігається поступове формування сигналів напівсуми.

Аналогічно виконується побудова напіввіднімачів.

Розглянемо принципи структурної побудови багаторозрядних двійкових суматорів, які використовуються для організації виконання операцій додавання-віднімання в комп'ютерних системах. Очевидно, що до складу багаторозрядних суматорів входять одноразрядні суматори. Найпростішими такими пристроями

є суматори з послідовним трактом розповсюдження переносу між розрядами. Функціональна схема такого суматора приведена на рис..5.

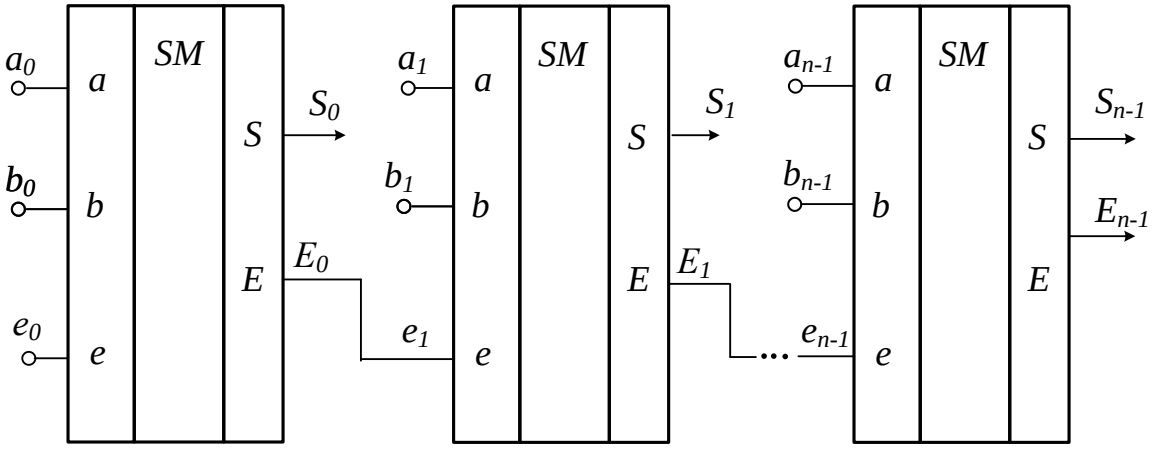


Рисунок .5 – Функціональна схема n-розрядного суматора з послідовним трактом розповсюдження переносу

На ис.1.5 використовуються такі позначення:

- $A = a_{n-1}, \ldots, a_0$, $B = b_{n-1}, \ldots, b_0$ – n-розрядні операнди (доданки) A і B, де a_i, b_i – значення i-того біту відповідно операндів A і B;
- e_0 – сигнал вхідного переносу до нульового розряду суматора;
- $S_{n-1}, \ldots, S_0$ – n-розрядна сума;
- E_{n-1} – сигнал вихідного переносу з $n-1$ розряду суматора.

Сигнали вхідного і вихідного переносів e_0, E_{n-1} можуть використовуватися для нарощування розрядності цього пристрою. Крім того, вихідний перенос E_{n-1} формує ознаку переповнення результату операції додавання при використанні прямих кодів операндів. Сигнал вихідного переносу переключається до одиничного значення ($E_{n-1} = 1$), якщо $A + B + e_0 \geq 2^n$.

Приведена на рис..5 функціональна схема реалізує мікрооперацію додавання, тобто на виході пристрою формується код $A+B+e_0$.

Під час реалізації багаторозрядних суматорів з послідовним переносом широко використовується властивість самоподвійності логічних функцій суми і переносу. Для цього в якості однорозрядних суматорів використовуються суматори з тратами $(e \to \overline{E})$ і $(\overline{e} \to E)$. В таких багаторозрядних суматорах вихідний сигнал переносу тракту $(e \to \overline{E})$ безпосередньо використовується як вхідний переносу в старший розряд з трактом $(\overline{e} \to E)$ або, навпаки, вихідний перенос розряду з трактом $(\overline{e} \to E)$ є вхідним переносом для розряду з трактом $(e \to \overline{E})$. Отже, в кожному розряді такого багаторозрядного суматора формується взаємна інверсія сигналів переносу, тобто в суматорі використовується

почергова інверсія сигналу переносу ($e_0 \rightarrow \overline{e_1} \rightarrow e_2 \rightarrow \overline{e_3}$ і т.д. або навпаки $\overline{e_0} \rightarrow e_1 \rightarrow \overline{e_2} \rightarrow e_3$).

Функціональна схема чотирьохрозрядного фрагменту багаторозрядного суматора з послідовним трактом розповсюдження переносу, реалізованого за принципом почергової інверсії сигналів переносу, приведена на рис..6.

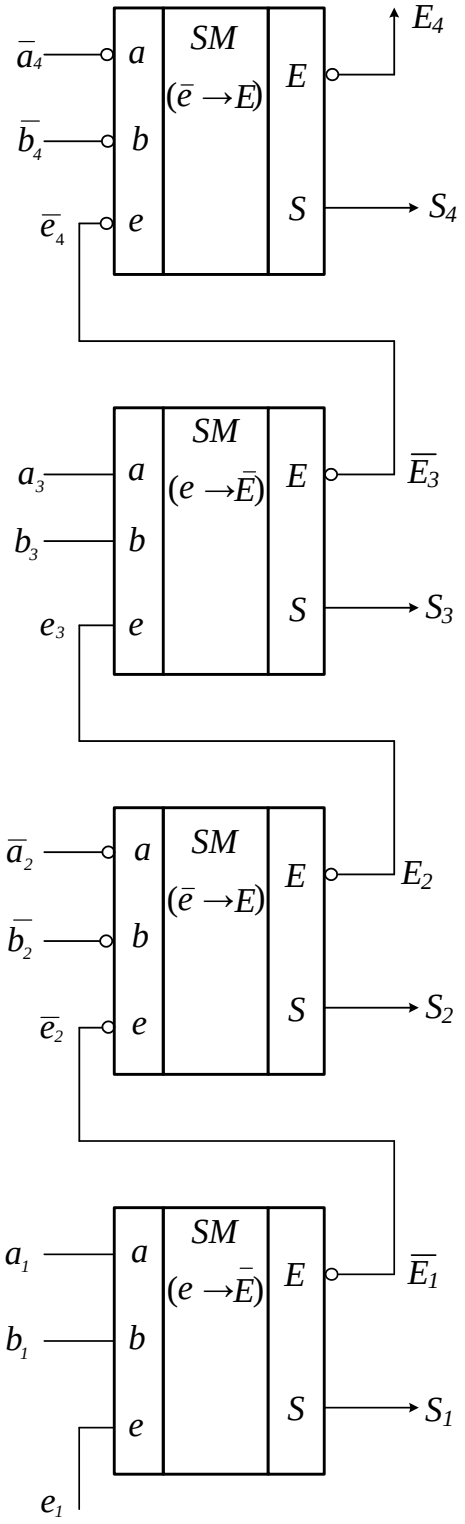


Рисунок .6 – Функціональна схема чотирьохрозрядного фрагменту багаторозрядного суматора з почерговою інверсією сигналів переносу

Логічна схема такого суматора для моделювання в середовищі системи схемотехнічного проектування MicroCap приведена на рис..7. Результати моделювання статичного режиму функціонування у вигляді часових діаграм в двійковому і десятковому форматах в середовищі системи MicroCap приведені на рис .8.

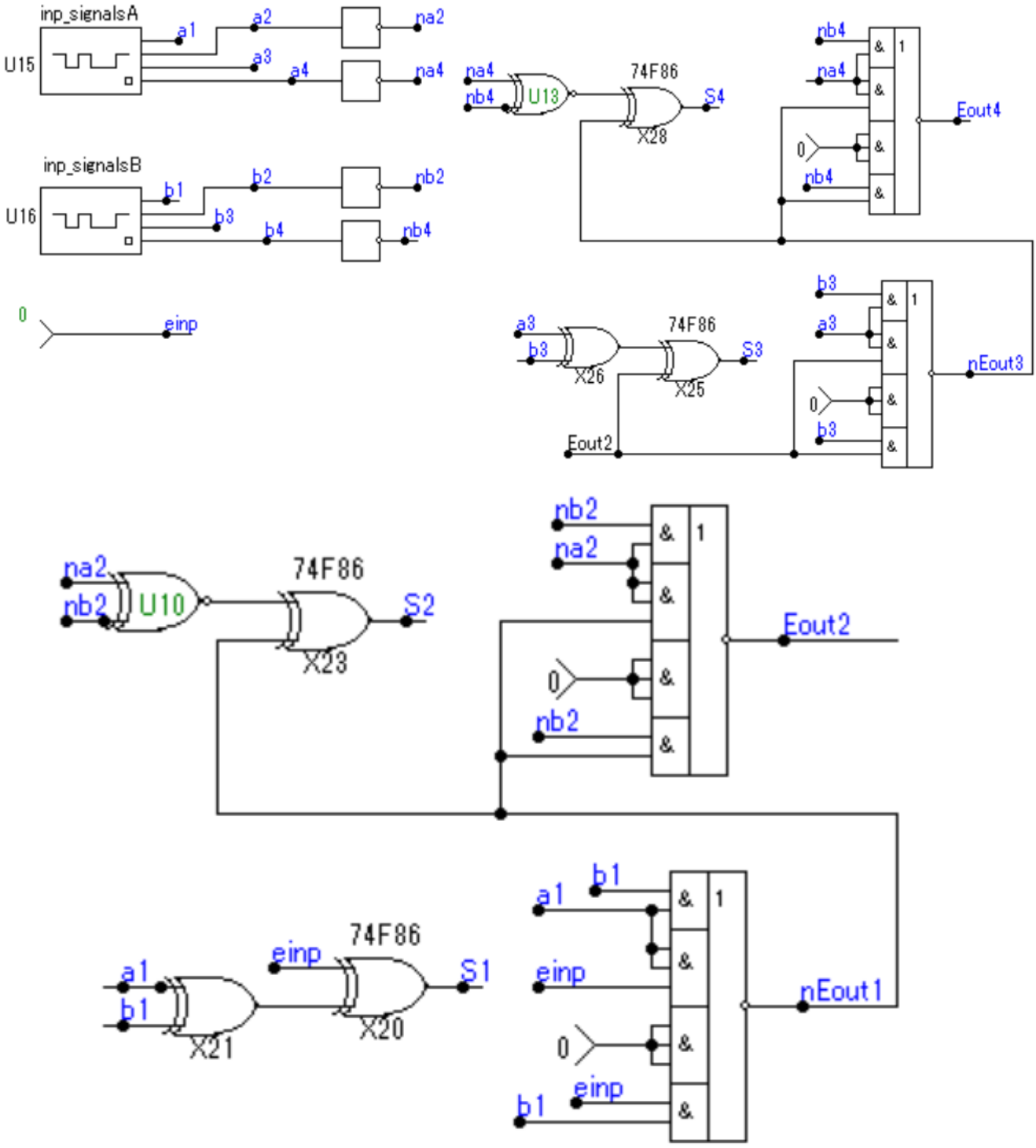


Рисунок .7 – Схема моделювання чотирьохбітного фрагменту суматора з почерговою інверсією

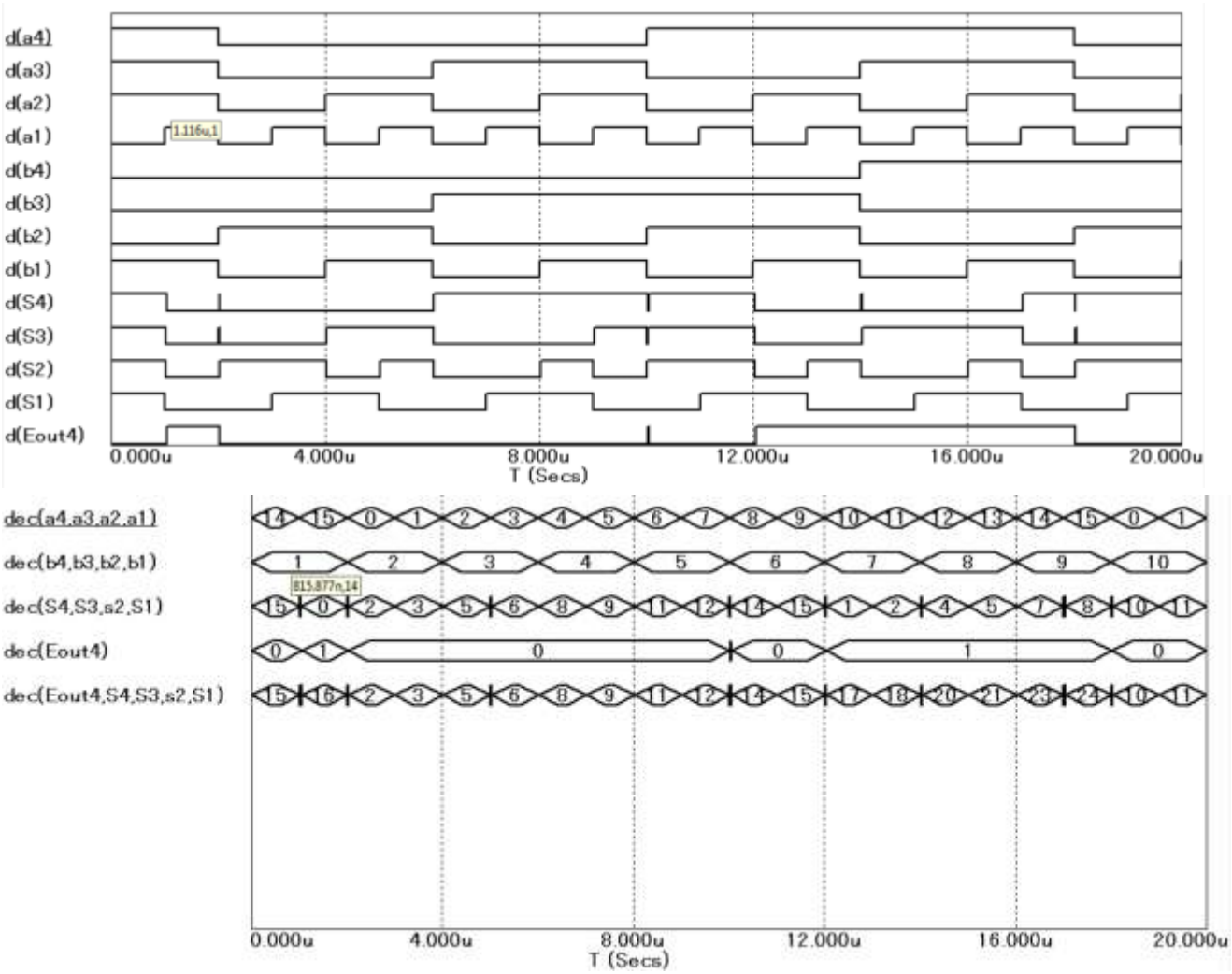


Рисунок .8 – Часові діаграми чотирьохрозрядного фрагмента багаторозрядного суматора з почерговою інверсією сигналів переносу

По аналогії з напівсуматорами швидкодія двійкових суматорів з послідовним трактом розповсюдження переносу залежить від значень вхідних операндів *A, B, e_0.* В найгіршому випадку час спрацьовування суматора t_{SM} буде найбільшим при умові, що перенос розповсюджується через всі розряди пристрою. При цьому час спрацьовування суматора може бути визначений таким чином: $t_{SM} = (n-1)\cdot t_E + t_S$, де n – кількість розрядів суматора; t_E – час формування вихідного переносу; t_S – час формування суми. На відміну від напівсуматорів вхідних кодів, при яких суматор має найменшу швидкодію, достатньо багато (наведемо приклади деяких таких кодів: *A = 1111, B = 0001, $e_0 = 0$; A = 1101, B = 0011, $e_0 = 0$; A = 1001, B = 0110, $e_0 = 1$*).

Результати моделювання динамічного режиму функціонування в середовищі системи MicroCap приведені на рис..9. Часові діаграми на цьому рисунку демонструють розповсюдження переносу через всі розряди суматора при значення операндів *A = 1111, B = 0001, $e_0 = 0$.* На часовій діаграмі розповсюдження сигналу переносу викликає поступове спрацьовування розрядів

суматора. Час формування сигналу вихідного переносу показний виносками і складає 49 нс.

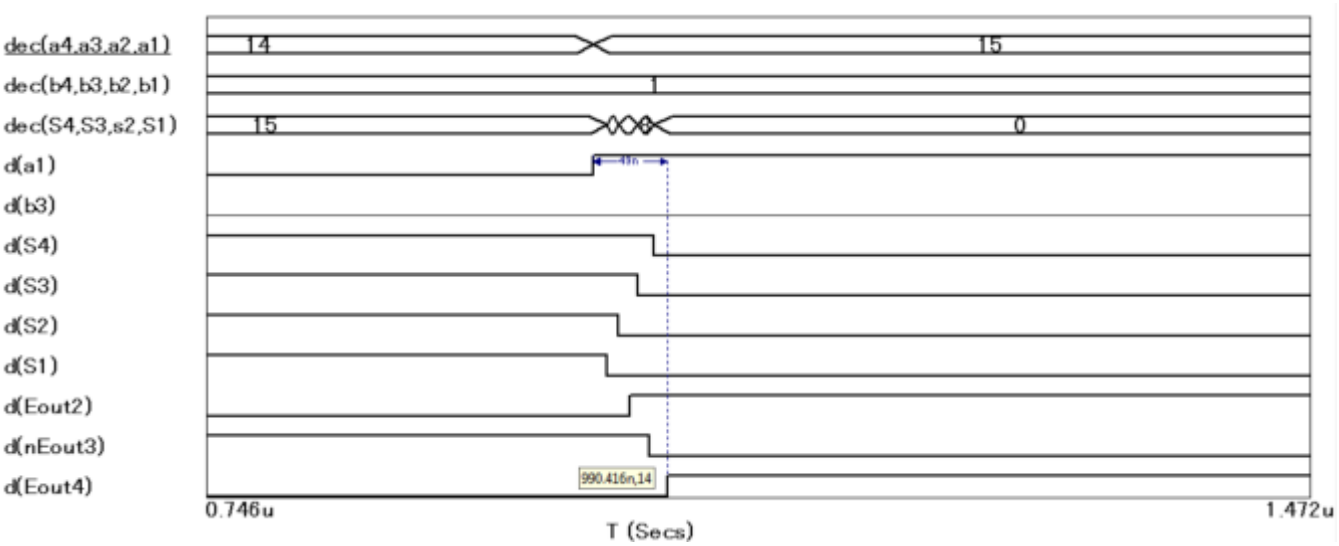


Рисунок .9 – Часова діаграма розповсюдження переносу
в 4-розрядному суматорі

Поведінкова модель чотирьохрозрядного суматора з послідовним переносом з використанням почергової інверсії переносу приведена нижче:

```
module ProjectNULES (
 a,b,einp, S, Eout4, Eout2
);
        input [3:0] a;
        input [3:0] b;
        input einp;

        output [3:0] S;
        output Eout2;
        output Eout4;

        wire nEout1;
        wire nEout3;

        wire [3:0] na;
        wire [3:0] nb;

        assign na = ~a;
        assign nb = ~b;
        assign S[0] = (a[0] ^ b[0]) ^ einp;
        assign nEout1 = ~((b[0] && a[0]) | (a[0] && einp) | (einp && b[0]));
```

```
        assign S[1] = (na[1] ^ nb[1]) ^ nEout1;
        assign Eout2 = ~((nb[1] && na[1]) | (na[1] && nEout1) | (nEout1 &&
nb[1]));
        assign S[2] = (a[2] ^ b[2]) ^ Eout2;
        assign nEout3 = ~((b[2] && a[2]) | (a[2] && Eout2) | (Eout2 && b[2]));
        assign S[3] = (na[3] ^ nb[3]) ^ nEout3;
        assign Eout4 = ~((nb[3] && na[3]) | (na[3] && nEout3) | (nEout3 &&
nb[3]));
        endmodule
```

На рис..10 приведені результаті моделювання такого суматора в середовищі САПР QUARTUS (сигнали S_3, S_1 реалізують інверсні значення суми в цих розрядах).

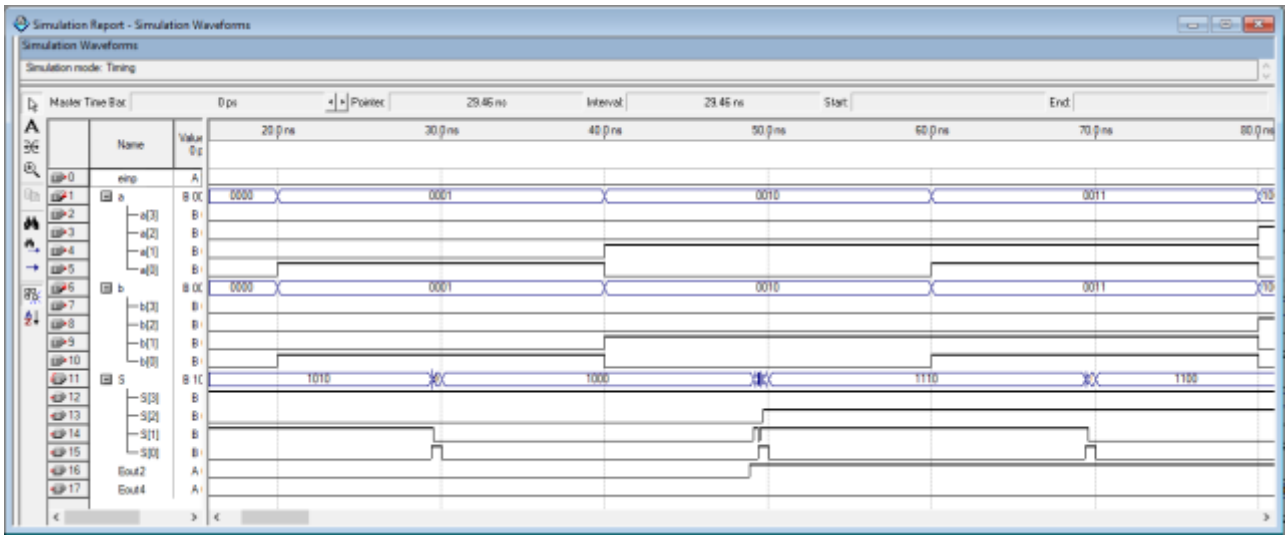


Рисунок .10 – Часова діаграма функціонування 4-розрядного суматора в середовищі САПР QUARTUS

Таким чином, аналізуючи структурну організацію і результати моделювання багаторозрядних двійкових суматорів з послідовним трактом розповсюдження переносу, можна відзначити, що з моменту надходження операндів A, B і e_0 на входи такого суматора відбувається переключення сигналів на виході пристрою впродовж певного часу, що визначається розповсюдженням переносу через розряди суматора. Як вже відмічалося, величина тривалості перехідного процесу в суматорі визначається часом затримки сигналів в кожному логічному елементі пристрою і кількості каскадів, що спрацювали, в колі розповсюдження переносу, тобто час виконання мікрооперації додавання в суматорі з послідовним трактом розповсюдження переносу прямо пропорційний кількості розрядів пристрою .

Мікрооперації додавання-віднімання є одними з найчастіше використовуваними процесором під час функціонування комп'ютерних систем. В той же час тривалість виконання цих мікрооперацій є найдовшою при використанні арифметичних пристроїв з послідовним переносом. У зв'язку з цим розробці методів підвищення швидкодії арифметичних пристроїв приділяється велика увага.

Одним зі способів підвищення швидкодії арифметичних пристроїв є використання додаткових вузлів, які забезпечують формування паралельних кіл для розповсюдження переносу в суматорі, тобто всі вхідні сигнали переносу в кожному розряді формуються одночасно незалежно від значень переносу з сусідніх розрядів.

Вхідним переносом в молодший розряд є зовнішній сигнал переносу e_0 (e_{BX}). В наступному розряді перенос формується за звичайним мінімальним виразом за допомогою підготовчих функцій g_0 і p_0 цього розряду: $E_0(e_1) = g_0 + p_0 e_0$, де g_i, p_i – відповідно функції генерації і розповсюдження переносу в *i*-тому розряді.

Значення вихідного переносу в наступному розряді паралельного суматора формується таким чином:

$$E_1(e_2) = g_1 + p_1 e_1 = g_1 p_1 + p_1(g_0 + p_0 e_0) = g_1 + p_1 g_0 + p_1 p_0 e_0;$$

Аналогічно можна отримати остаточні формул для сигналів переносу з другого і третього розрядів суматора:

$$E_2(e_3) = g_2 + p_2 e_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 e_0;$$

$$E_3(e_4) = g_3 + p_3 e_2 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 e_0.$$

На рис..11 приведена логічна схема 4-розрядного фрагменту суматора (розряди 1-4) з вузлом формування паралельного переносу, який побудований відповідно до зазначених вище формул.

Час виконання мікрооперації додавання в суматорі з паралельним переносом визначається однаковою затримкою для всіх розрядів, тобто швидкодія суматора не залежить від кількості розрядів пристрою.

З зазначених вище логічних виразів для формування паралельного переносу втікає, що апаратні витрати на реалізацію суматора з паралельним переносом значно перевищують апаратні витрати на побудування аналогічного суматорі з послідовним трактом розповсюдження переносу. Ці витрати дуже швидко зростають при збільшенні кількості розрядів пристрою. У зв'язку з цим схемна реалізація паралельного переносу в суматорах як правило обмежується чотирма розрядами.

Фрагмент схеми для моделювання 4-бітного суматора з паралельним переносом в середовищі MicroCap приведена на рис..12. Результати

моделювання статичного режиму функціонування цього суматора приведені на рис..13.

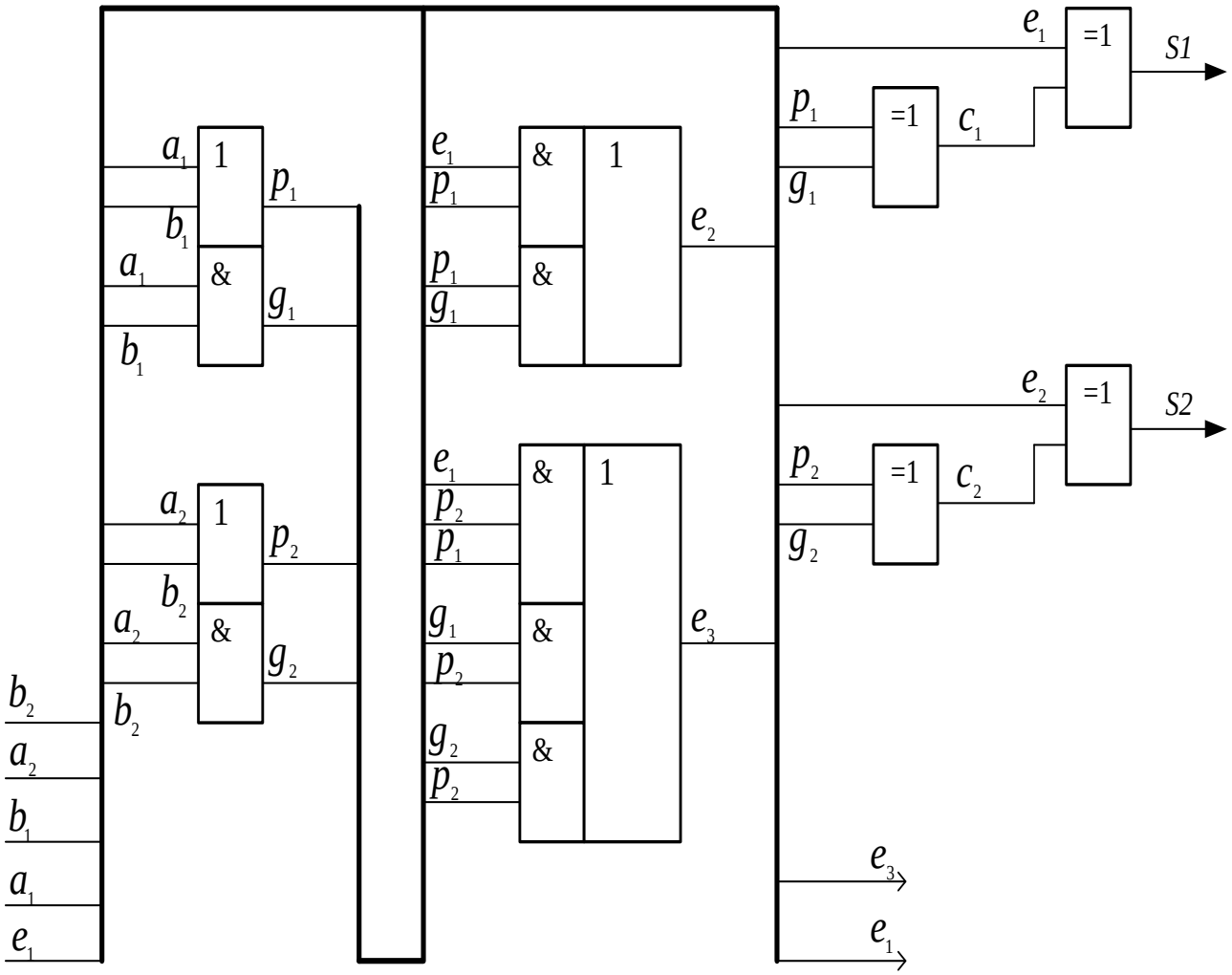


Рисунок .11,а – Схема формування сигналів S_2, S_1, e_3, e_2 в 4-розрядному суматорі з паралельним переносом

На рис..14 приведені результати моделювання динамічного режиму функціонування 4-бітного двійкового суматора з паралельним переносом в середовищі MicroCap у вигляді часової діаграми з урахуванням затримок логічних елементів. На часовій діаграмі можна спостерігати паралельне формування міжрозрядних переносів.

Нижче приведена поведінкова модель на мові Verilog 4-бітного суматора з паралельним переносом:

```
module ProjectNULES (
 a, b, einp1, Eout, S
);
        input [4:1] a;
        input [4:1] b;
```

```
input einp1;

output [4:1] S;
output [4:1] Eout;

wire [4:1] g;
wire [4:1] p;

assign S[1] = (a[1] ^ b[1]) ^ einp1;
assign g[1] = b[1] & a[1];
assign p[1] = b[1] | a[1];
assign Eout[1] = (g[1] & p[1]) | (p[1] & einp1);

assign S[2] = (a[2] ^ b[2]) ^ Eout[1];
assign g[2] = b[2] & a[2];
assign p[2] = b[2] | a[2];
assign Eout[2] = (p[2] & g[2]) | (g[1] & p[2]) | (p[2] & p[1] & Eout[1]);

assign S[3] = (a[3] ^ b[3]) ^ Eout[2];
assign g[3] = b[3] & a[3];
assign p[3] = b[3] | a[3];
assign Eout[3] = (g[3] & p[2] & g[1]) | (g[3] & p[3]) | (p[3] & g[2]) |
(p[3] & p[2] & p[1] & Eout[2]);

assign S[4] = (a[4] ^ b[4]) ^ Eout[3];
assign g[4] = b[4] & a[4];
assign p[4] = b[4] | a[4];
assign Eout[4] = (g[2] & p[3] & p[4]) | (p[4] & g[3]) | (g[4] & p [4]) |
(g[1] & p[2] & p[3] & p[4]) | (p[4] & p[3] & p[2] & p[1] & Eout[3]);

endmodule
```

Результати моделювання в середовищі САПР QUARTUS 4-бітного суматора з паралельним переносом приведені на рис..15.

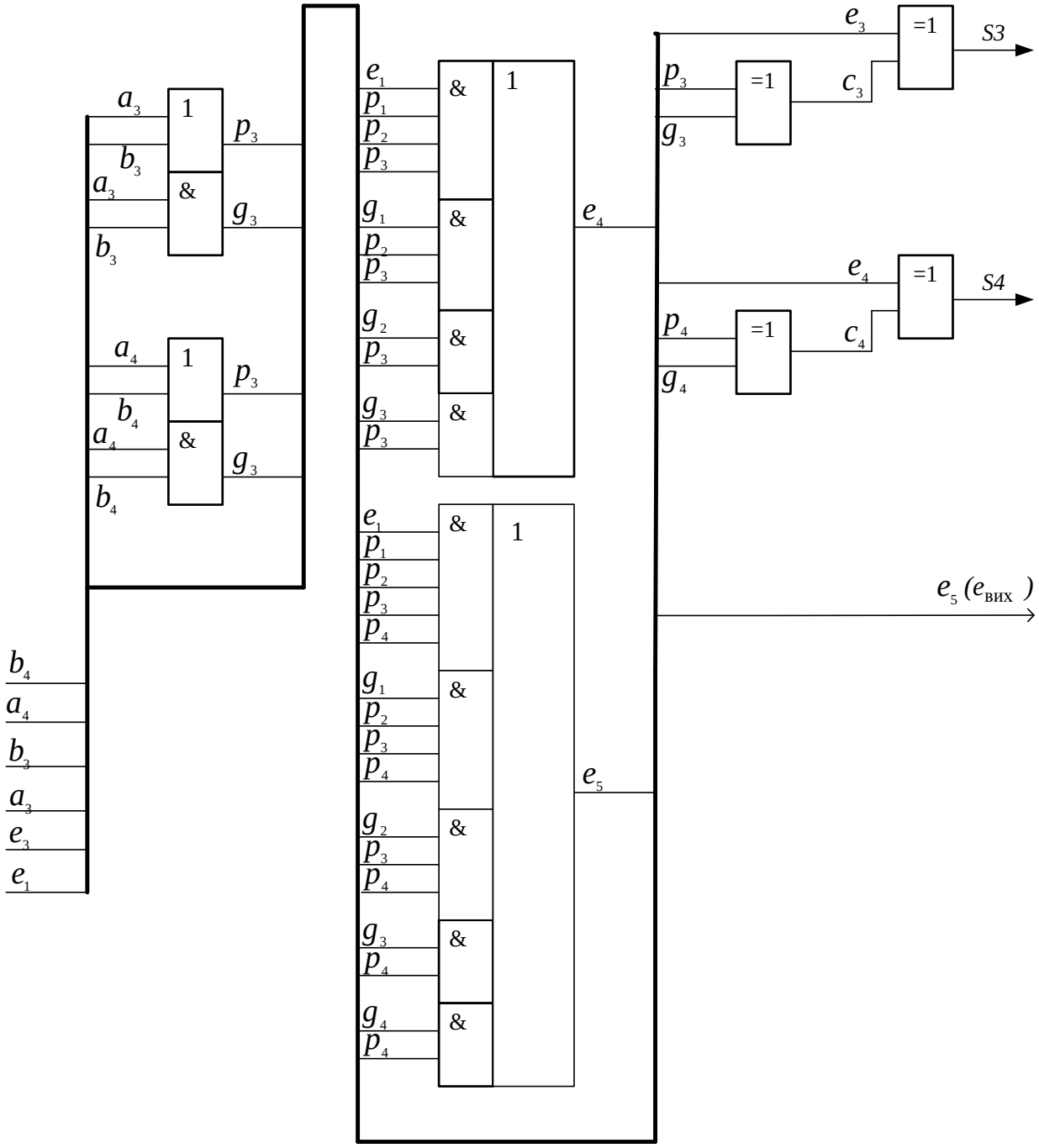


Рисунок .11,б – Схема формування сигналів S_4, S_3, e_5, e_4
4-розрядного суматора з паралельним переносом

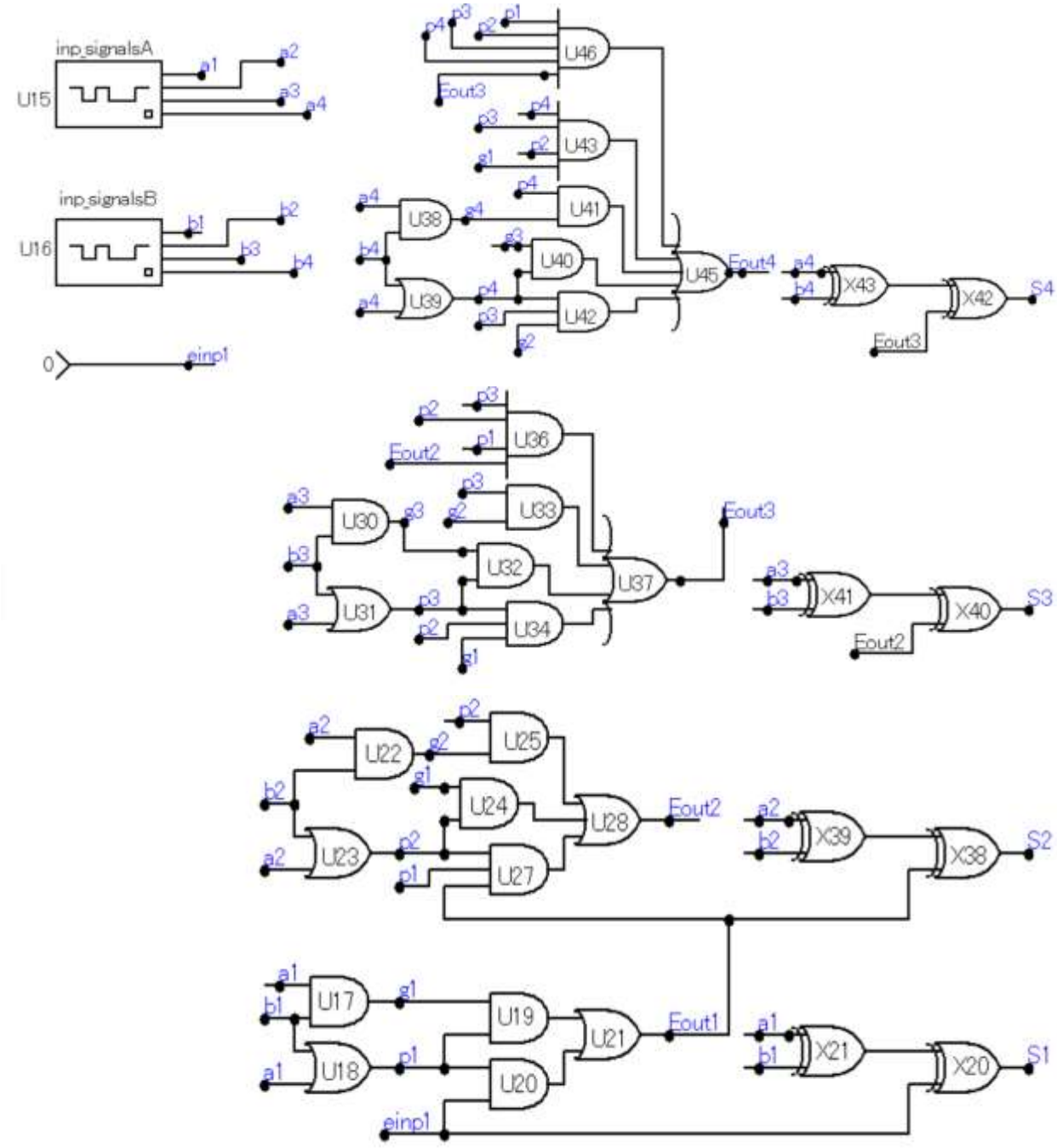


Рисунок .12 – Схема моделювання в середовищі MicroCap
4-розрядного суматора з паралельним переносом

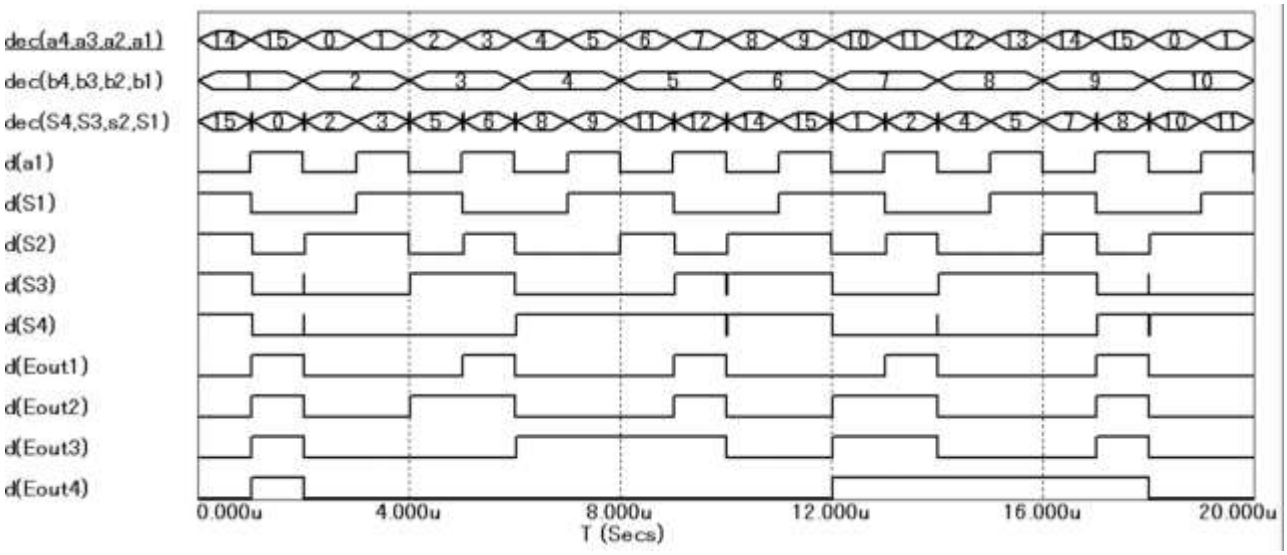


Рисунок .13 – Результати моделювання статичного режиму функціонування 4-рирозрядного суматораа з паралельним переносом

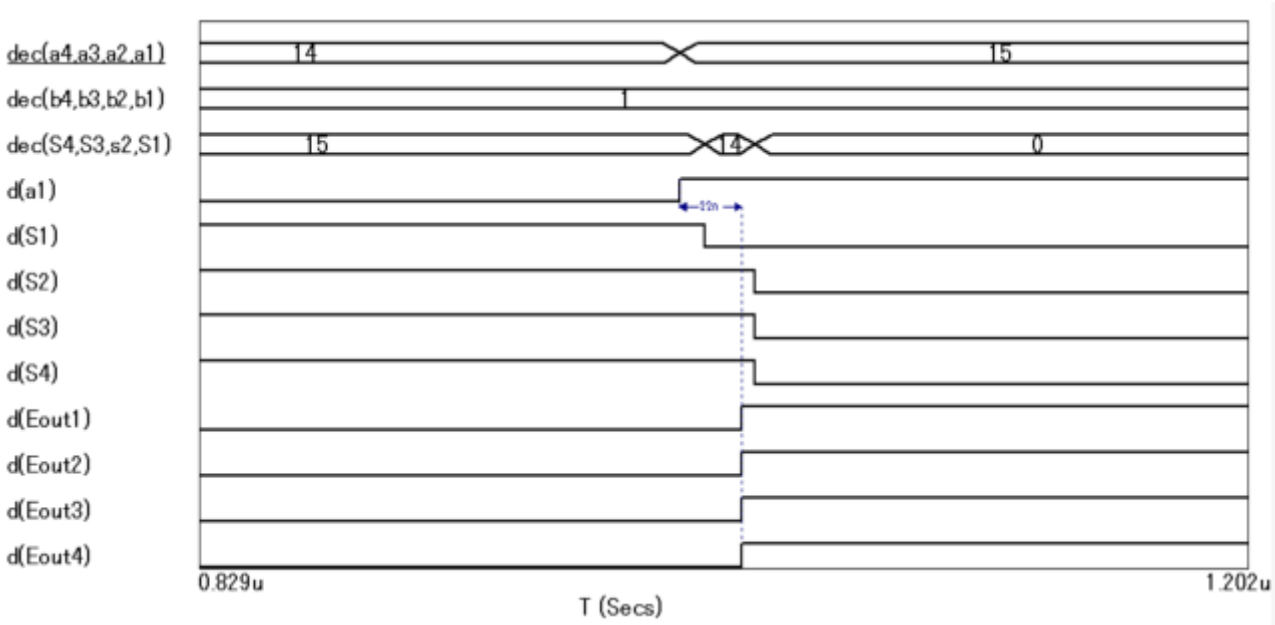


Рисунок .14 – Часова діаграма динамічного режиму спрацьовування суматора з паралельним переносом

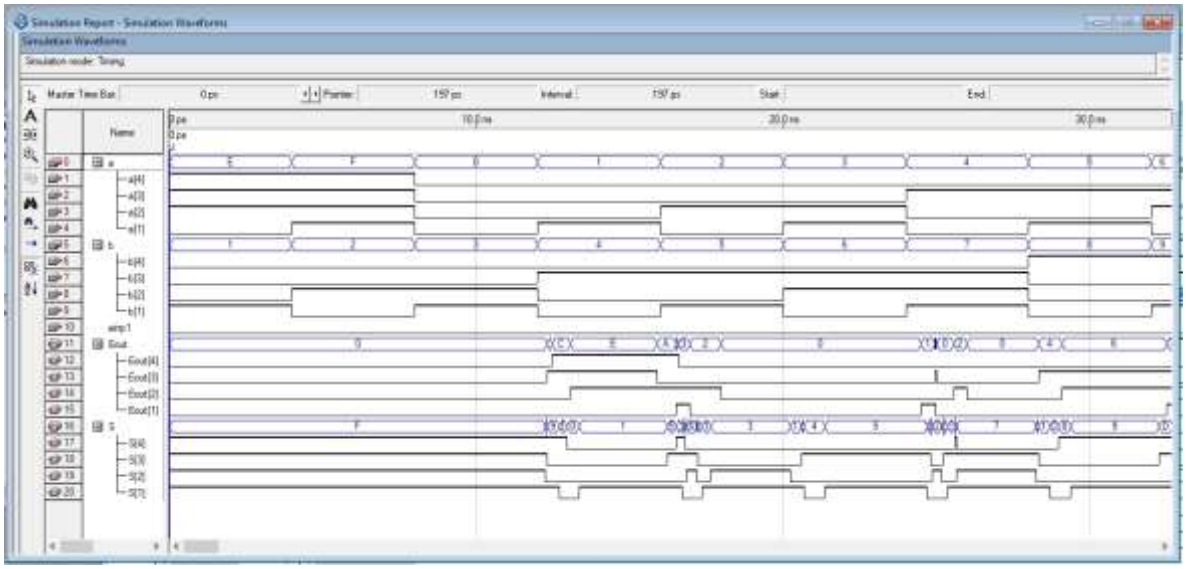


Рисунок .15 – Результати моделювання в середовищі САПР QUARTUS II 4-бітного суматора з паралельним переносом

Підготовка до лабораторної роботи

Під час підготовки до лабораторної роботи студент повинен знати устрій, структурну організацію та органи керування лабораторним стендом DE0, засвоїти принцип функціонування САПР QUARTUS II та процедуру завантаження логічних схем до стенду DE0 і перевірки функціонування пристроїв, що досліджуються.

В ході виконання лабораторної роботи необхідно створити проекти для плати «Суматор», за допомогою яких виконується дослідження багаторозрядних суматорів з послідовним і паралельним переносом. В якості вхідних даних буде використовуватися два операнди та вхідний перенос до суматора. На платі стенду DE0 для підключення вхідних операндів використовується відпрвідні кнопки. Відображення результату арифметичної операції додавання здійснюється за допомогою світлодіодів, які входять до складу блока індикації стенда DE0.

Відповідно до завдання необхідно виконати синтез пристрою із заданою структурною організацією.

Номер варіанту завдання видається викладачем.

Після процедури синтезу виконати моделювання функціонування заданих пристроїв в середовищі САПР QUARTUS II та пересвідчитись, що реакція пристроїв на вхідні сигнали відповідає таблиці істинності.

Додаток Д

Підключення стенду DE0 до напруги живлення та початок виконання лабораторної роботи відбувається тільки з дозволу викладача. Підсумки виконання кожного лабораторного експерименту необхідно обов'язково надавати викладачу для перевірки результатів досліду та їх фіксації.

Хід виконання роботи

Кожний дослід лабораторної роботи виконується в наступній послідовності:

- з дозволу викладача виконати завантаження схеми пристрою до лабораторного стенда DE0;
- задаючи на вхід пристрою, що досліджується, вхідні сигнали відповідно до таблиці істинності, зафіксувати за допомогою блоку індикації стенду реакцію пристрою на кожну комбінацію вхідних сигналів.

Послідовність дослідів під час виконання лабораторної роботи приведена нижче:

1. Виконати дослідження статичного режиму роботи чотирьохрозрядного напівсуматора з послідовним трактом розповсюдження переносу.

1. Виконати дослідження статичного режиму роботи чотирьохрозрядного напівсуматора з послідовним трактом розповсюдження переносу.

2. Виконати дослідження статичного режиму роботи чотирьохрозрядного напіввіднімача з послідовним трактом розповсюдження переносу..

3. Виконати дослідження статичного режиму роботи чотирьохрозрядного суматора з послідовним трактом розповсюдження переносу.

4. Виконати дослідження статичного режиму роботи чотирьохрозрядного суматора з паралельним переносом.

Зміст звіту

1. Титульний аркуш.
2. Завдання.
3. Синтез чотирьохрозрядного напівсуматора з послідовним трактом розповсюдження переносу.
4. Синтез чотирьохрозрядного напіввіднімача з послідовним трактом розповсюдження переносу.
5. Синтез чотирьохрозрядного суматора з послідовним трактом розповсюдження переносу.
6. Синтез чотирьохрозрядного суматора з паралельним переносом.
7. Виводи за результатами проведених дослідів.

Питання для самоперевірки

1. Як використовується властивість самоподвійності функцій *S* і *E* суматора при реалізації багаторозрядних суматорів?
2. Як реалізувати напівсуматор з послідовним трактом розповсюдження переносу?
3. Як реалізувати напіввіднімач з послідовним трактом розповсюдження позики?
4. Яку мікрооперацію виконує напівсуматор при $e_0=1$?
5. Яку мікрооперацію виконує напівсуматор при $m_0=1$?
6. Яку мікрооперацію виконує напівсуматор при $e_0=0$?
7. Яку мікрооперацію виконує напівсуматор при $m_0=0$?
8. Чи є функція напівсуми самоподвійною? Обґрунтуйте відповідь.
9. Чи є функція напіврізниці самоподвійною? Обґрунтуйте відповідь.
10. Чи є функція вихідного переносу напівсуматора самоподвійною? Обґрунтуйте відповідь.
11. Чи є функція вихідної позики напіввіднімача самоподвійною? Обґрунтуйте відповідь.
12. Як визначити час спрацьовування *n*-розрядного напівсуматора?
13. Як визначити час спрацьовування *n*-розрядного напіввіднімача?
14. З якою метою використовується вихідний перенос зі старшого розряду напівсуматора?
15. З якою метою використовується вихідна позика зі старшого розряду напіввіднімача?
16. Як спростити логічну схему *n*-розрядного напівсуматора, якщо вхідний перенос до наймолодшого розряду завжди дорівнює одиниці, а вихідний перенос з найстаршого розряду не використовується?
17. Як спростити логічну схему *n*-розрядного напіввіднімача, якщо вхідна позика до наймолодшого розряду завжди дорівнює одиниці, а вихідна позика з найстаршого розряду не використовується?
18. Побудуйте арифметичний пристрій, який за керуючим сигналом u_1 виконує мікрооперацію «інкремент», а за сигналом u_2 – мікрооперацію «декремент».
19. Побудуйте 4-розрядний напівсуматор в базисі І-НІ.
20. Побудуйте 4-розрядний напівсуматор р в базисі АБО-НІ.
21. Побудуйте 4-розрядний напіввіднімач в базисі І-НІ.
22. Побудуйте 4-розрядний напіввіднімач в базисі АБО-НІ.
23. Як побудувати *n*-розрядний суматор з послідовним трактом розповсюдження переносу?

24. Як побудувати *n*-розрядний віднімач з послідовним трактом розповсюдження позики?
25. Побудуйте 4-розрядний суматор з послідовним трактом розповсюдження переносу в базисі І-АБО-НІ.
26. Побудуйте 4-розрядний віднімач з послідовним трактом розповсюдження позики в базисі І-АБО-НІ.
27. Поясніть термін «почергова інверсія переносу».
28. Поясніть термін «почергова інверсія позики».
29. Для чого в багаторозрядних суматорах використовуються тракти ($\bar{e} \rightarrow E$) і ($e \rightarrow \overline{E}$)?
30. Для чого в багаторозрядних віднімачах використовуються тракти ($\bar{m} \rightarrow M$) і ($m \rightarrow \overline{M}$)?
31. Розрахуйте значення сигналів суми $S_3\ S_2\ S_1\ S_0$ і переносу $E_3\ \overline{E}_2\ E_1\ \overline{E}_0$ на виходах 4-розрядного суматора при додаванні двійкових операндів:

$a_3\ a_2\ a_1\ a_0 = 0111$, $b_3\ b_2\ b_1\ b_0 = 0111$, $e_1 = 0$;
$a_3\ a_2\ a_1\ a_0 = 1001$, $b_3\ b_2\ b_1\ b_0 = 0100$, $e_1 = 1$;
$a_3\ a_2\ a_1\ a_0 = 1010$, $b_3\ b_2\ b_1\ b_0 = 1110$, $e_1 = 1$;
$a_3\ a_2\ a_1\ a_0 = 0110$, $b_3\ b_2\ b_1\ b_0 = 1001$, $e_1 = 1$;
$a_3\ a_2\ a_1\ a_0 = 0111$, $b_3\ b_2\ b_1\ b_0 = 1001$, $e_1 = 0$;
$a_3\ a_2\ a_1\ a_0 = 1101$, $b_3\ b_2\ b_1\ b_0 = 1011$, $e_1 = 0$.

32. Якими параметрами характеризується швидкодія багаторозрядних суматорів?
33. Приведіть двійкові коди операнду $a_3\ a_2\ a_1\ a_0$ і e_0, при яких 4-розрядний напівсуматор буде мати найменшу швидкодію.
34. Приведіть двійкові коди операнду $a_3\ a_2\ a_1\ a_0$ і m_0, при яких 4-розрядний напіввіднімач буде мати найменшу швидкодію.
35. Приведіть значення двійкових кодів операндів $a_3\ a_2\ a_1\ a_0$, $b_3\ b_2\ b_1\ b_0$, і e_0 при яких 4-розрядний суматор буде мати найбільшу і найменшу швидкодію.
36. Як визначити час спрацьовування багаторозрядного двійкового суматора?
37. Порівняйте швидкодію суматорів з використанням і без використання почергової інверсії переносів.
38. Визначити значення сигналів на виводах 4-розрядного суматора при додаванні чисел $A = 0111$ і $B = 1010$ та $e_0 = 0$.
39. Визначити значення функцій генерації і розповсюдження переносу при додаванні чисел $A = 0101$ і $B = 0110$ та $e_0 = 0$.

40. Приведіть логічну схему 4-розрядного суматора з почерговою інверсією сигналів переносу ($e_0 \to \overline{e_1} \to e_2 \to \overline{e_3}$) на базіі композиційних мінімальних суматорів.
41. Приведіть логічну схему 4-розрядного суматора з почерговою інверсією сигналів переносу ($\overline{e_0} \to e_1 \to \overline{e_2} \to e_3$) на базіі композиційних мінімальних суматорів.

Тести для самоперевірки

В завданнях, що приведені нижче, вказати одну правильну відповідь:

1. Почергова інверсія сигналів переносу в багаторозрядних двійкових суматорах використовується для:
   A) збільшення завадостійкості; *B) збільшення швидкодії;*
   C) збільшення коефіцієнту розгалуження;
   D) збільшення ціни за Квайном; *E) немає правильної відповіді.*
2. Використання почергової інверсії сигналів позики в багаторозрядних віднімачах призводить до:
   A) організації позитивної логіки кодування сигналів;
   B) зменшення часу спрацьовування віднімача;
   C) отримання мінімальної функції для формування сигналу різниці;
   D) збільшення часу затримки формування позики;
   E) немає правильної відповіді.
3. Визначте значення $E_3 \overline{E}_2 E_1 \overline{E}_0$ на виходах 4-розрядного суматора при додаванні операндів: $a_3\, a_2\, a_1\, a_0$ = 0110, $b_3\, b_2\, b_1\, b_0$ = 1001, e_0 = 1;
   A) 0000; *B) 1111;* *C) 0101;*
   D) 1010; *E) немає правильної відповіді.*
4. Вкажіть коди операндів $a_3\, a_2\, a_1\, a_0$ і $b_3\, b_2\, b_1\, b_0$, при яких 4-розрядний суматор має найменшу швидкодію при e_0 = 0.
   A) 0110 і 1101; *B) 1111 і 1111;* *C) 0101 і 1011;*
   D) 1010 і 0101; *E) немає правильної відповіді.*
5. Вкажіть коди операндів $a_3\, a_2\, a_1\, a_0$ і $b_3\, b_2\, b_1\, b_0$, при яких 4-розрядний суматор має найбільшу швидкодію при e_0 = 1.
   A) 0110 і 1100; *B) 1110 і 1111;* *C) 0101 і 1010;*
   D) 1010 і 0100; *E) немає правильної відповіді.*
6. Час формування переносу в кожному розряді складає 20нс, а час формування суми – 9нс. Визначити час спрацьовування 4-бітного суматора при операндах 1101 і 0011, e_0 = 0.
   A) 9нс; *B) 29нс;* *C) 49нс;* *D) 69нс;* *E) 89нс;*
   F) 80нс *G) немає правильної відповіді.*

7. Вкажіть тип переносу, при якому багаторозрядний суматор має найменшу швидкодію.

   A) обхідний; B) паралельний; C) груповий*; D) послідовний;*

   E) немає правильної відповіді.

8. Визначити вираз, який використовується для формування сигналу переносу E_3 при використанні паралельного тракту переносу в суматорі:

   A) $p_3 + p_3g_2 + p_3g_2g_1 + p_3g_2g_1g_0 + p_3g_2g_1g_0e_0$*;*

   B) $g_3 + g_3p_2 + g_3g_2p_1 + g_3g_2g_1p_0 + g_3p_2p_1p_0e_0$*;*

   C) $g_3 + p_3g_2 + g_3p_2g_1 + g_3p_2g_1p_0 + g_3g_2g_1g_0e_0$*;*

   D) $g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0e_0$*;*

   E) немає правильної відповіді..

9. Визначити вираз, якмй використовується для організації паралельного переносу в суматорах:

   A) $ab\bar{e} + a\bar{b}\,e + \bar{a}\,be + abe$*;* *B)* $ab + ae + be$*;*

   C) $g + pe$*;* *D)* $ab + (a \oplus b)e$*;* *E)* $\bar{g} + \bar{p}\,e$*;*

   F) немає правильної відповіді.

Додаток Г

Підключення стенду DE0 до напруги живлення та початок виконання лабораторної роботи відбувається тільки з дозволу викладача. Підсумки виконання кожного лабораторного експерименту необхідно обов'язково надавати викладачу для перевірки результатів досліду та їх фіксації.

Під час підготовки до лабораторної роботи студент повинен вивчити устрій, структурну організацію та органи керування лабораторним стендом DE0, засвоїти принцип функціонування САПР QUARTUS II та процедуру завантаження логічних схем до стенду DE0 і перевірки функціонування пристроїв, що досліджуються.

В ході виконання лабораторної роботи необхідно створити проекти для плати «Суматор», за допомогою яких виконується дослідження однобітних суматорів. В якості вхідних даних буде використовуватися два однобітних операнди та вхідний перенос до суматора. На платі стенду DE0 для підключення вхідних операндів використовується три кнопки (*a, b, e*). Відображення результату (*S, E*) арифметичної операції додавання здійснюється за допомогою світлодіодів, які входять до складу блока індикації стенда DE0.

Відповідно до заданого варіанту завдань необхідно виконати синтез вказаного пристрою в заданому базисі логічних елементів. Варіанти завдань приведені в табл.5.4.

Варіанти елементного базису Таблиця 5.4

Номер варіанта	Базис ЛЕ	Номер варіанта	Базис ЛЕ
1	2I-HI	19	4I-HI
2	2АБО-HI	20	3I, 2АБО-HI
3	3I-HI	21	2I-HI, 2АБО-HI
4	2I, 2АБО-HI	22	3I-HI, 2АБО-HI
5	2I, 3I-HI	23	3I, 2I-H
6	2АБО, 2I-HI	24	2АБО, 3I-HI
7	3АБО-HI	25	4АБО-HI
8	3АБО, 2I-HI	26	3АБО, 3I-HI
9	2XOR, 2I-HI	27	2XNOR, 2I-HI
10	2XOR, 2АБО-HI	28	2XNOR, 2АБО-HI
11	2XOR, 2I, 2АБО-HI	29	2XNOR, 2I, 2АБО-HI
12	2XOR, 2АБО, 2I-HI	30	2XNOR, 2АБО, 2I-HI
13	булевий	31	4I-HI, 2АБО-HI
14	3АБО-HI, 3I-HI	32	3АБО-HI, 2I-HI
15	2I-2I-2I-3АБО-HI	33	2I-2I-2I-3I-4АБО-HI

16	2І-2І-2АБО-НІ	34	3І-3І-2АБО-НІ
17	2І-2І-2АБО-НІ, 3І	35	2І-2І-2АБО-НІ, 2АБО
18	2XOR, 2І-2І-2АБО-НІ	36	2АБО, 2АБО-НІ

Якщо в якості базису задано кілька логічних функцій, то під час реалізації схеми необхідно використовувати всі логічні функції з заданого базису.

Номер варіанту завдання видається викладачем. Якщо номер заданого варіанту (**V**), більше за максимальний номер з відповідної таблиці варіантів, то номер завдання необхідно визначити за формулою (**V mod n) + 1**, де **n** – кількість варіантів в таблиці; **mod** – функція визначення залишку під час ділення чисел.

Після процедури синтезу виконати моделювання функціонування заданих пристроїв в середовищі САПР QUARTUS II та пересвідчитись, що реакція пристроїв на вхідні сигнали відповідає таблиці істинності.

Хід виконання роботи

Кожний дослід лабораторної роботи виконується в наступній послідовності:

- з дозволу викладача виконати завантаження схеми пристрою до лабораторного стенда DE0;
- задаючи на вхід пристрою, що досліджується, вхідні сигнали відповідно до таблиці істинності, зафіксувати за допомогою блоку індикації стенду реакцію пристрою на кожну комбінацію вхідних сигналів.

Послідовність дослідів під час виконання лабораторної роботи приведена нижче:

1. Виконати дослідження статичного режиму роботи напівсуматора в заданому базисі логічних елементів.
2. Виконати дослідження статичного режиму роботи напівіднімача в заданому базисі логічних елементів.
3. Виконати дослідження статичного режиму роботи канонічного суматора в заданому базисі логічних елементів.
4. Виконати дослідження статичного режиму роботи канонічного суматора на базі дешифратора 3→8.
5. Виконати дослідження статичного режиму роботи канонічного суматора на базі мультиплексора 8→1.
6. Виконати дослідження статичного режиму роботи канонічного суматора на базі мультиплексора 4→1.
7. Виконати дослідження статичного режиму роботи мінімального суматора з трактом розповсюдження переносу ($e \to \overline{E}$) в заданому базисі логічних елементів.
8. Виконати дослідження статичного режиму роботи мінімального суматора з трактом розповсюдження переносу ($\overline{e} \to E$) в заданому базисі логічних елементів.

Зміст звіту

1. Титульний аркуш.
2. Завдання.
3. Синтез напівсуматора та перетворення логічних виразів до заданого базису.
4. Синтез напіввіднімача та перетворення логічних виразів до заданого базису.
5. Синтез канонічного суматора в заданому базисі, мінімізація і перетворення логічних виразів до заданого базису.
6. Синтез канонічного суматора на базі дешифратора 3→8. Використовувати результати мінімізації з попередніх дослідів.
7. Синтез канонічного суматора на базі мультиплексора 8→1. Використовувати результати мінімізації з попередніх дослідів.
8. Синтез канонічного суматора на базі мультиплексора 4→1. Використовувати результати мінімізації з попередніх дослідів.
9. Синтез мінімального суматора з трактом розповсюдження переносу ($e \to \overline{E}$) в заданому базисі логічних елементів. Використовувати результати мінімізації з попередніх дослідів.
10. Синтез мінімального суматора з трактом розповсюдження переносу ($\overline{e} \to E$) в заданому базисі логічних елементів. Використовувати результати мінімізації з попередніх дослідів.
11. Виводи за результатами проведених дослідів.

Питання для самоперевірки

80. Назвіть етапи проектування комбінаційної схеми із застосуванням ПЛІС?
81. Як проводиться підключення комбінаційної схеми до зовнішніх контактів стенду?
82. Яким чином визначити реакцію пристрою на вхідні сигнали?
83. Як задавати вхідні сигнали на вхід пристрою?
84. Для чого використовується блок індикації стенду?
85. Для чого використовується задавальна частина стенду?
86. Як виконати моделювання логічної схеми в середовищі системи MicroCap?
87. Як виконати моделювання логічної схеми в середовищі САПР QUARTUS II?
88. Дайте визначення напівсуматора.
89. Приведіть таблицю істинності напівсуматора.
90. Приведіть логічні вирази, що характеризують функціонування напівсуматорів.
91. Наведіть формули напівсуси і вихідного переносу в булевому базисі.

92. Для чого використовуються напівсуматори?
93. За допомогою яких типових пристроїв можна виконувати мікрооперація «інкремент»?
94. Наведіть ДДНФ напівсуми.
95. Наведіть ДКНФ напівсуми.
96. Наведіть формули напівсуси і вихідного переносу в базисі 2І-НІ.
97. Наведіть формули напівсуси і вихідного переносу в базисі 2АБО-НІ.
98. Наведіть формули напівсуси в базисі 2XNOR.
99. Наведіть формули напівсуси в базисі 2XOR.
100. Дайте визначення напівіднімача.
101. Приведіть таблицю істинності напівіднімача.
102. Приведіть логічні вирази, що характеризують функціонування напівіднімачів.
103. Наведіть формули напіврізниці і вихідної позики в булевому базисі.
104. Для чого використовуються напівіднімачі?
105. За допомогою яких типових пристроїв можна виконувати мікрооперація «декремент»?
106. Наведіть ДДНФ напіврізниці.
107. Наведіть ДКНФ напіврізниці.
108. Наведіть формули напіврізниці і вихідної позики в базисі 2І-НІ.
109. Наведіть формули напіврізниці і вихідної позики в базисі 2АБО-НІ.
110. Наведіть формули напіврізниці в базисі 2XNOR.
111. Наведіть формули напіврізниці в базисі 2XOR.
112. Побудуйте логічні схеми напівсуматора на базі дешифратора 2→4.
113. Побудуйте логічні схеми напівсуматора на базі мультиплексора 4→1.
114. Побудуйте логічні схеми напівсуматора на базі мультиплексора 2→1.
115. Побудуйте логічні схеми напіввіднімача на базі дешифратора 2→4.
116. Побудуйте логічні схеми напіввіднімача на базі мультиплексора 4→1.
117. Побудуйте логічні схеми напіввіднімача на базі мультиплексора 2→1.
118. Виконати синтез логічної схеми канонічного суматора в базисі 3І-НІ та 2І-НІ, з використанням ДДНФ.
119. Виконати синтез логічної схеми канонічного суматора в базисі 3І-НІ та 2І-НІ, з використанням ДКНФ.
120. Виконати синтез логічної схеми канонічного суматора в базисі 3АБО-НІ та 2АБО-НІ, з використанням ДДНФ.
121. Виконати синтез логічної схеми канонічного суматора в базисі 3АБО-НІ та 2АБО-НІ, з використанням ДКНФ.
122. Побудуйте канонічний суматор на базі дешифраторів 2→4.
123. Побудуйте канонічний суматор на базі дешифратора 3→8.

124. Побудуйте канонічний суматор на базі мультиплексорів 8→1.
125. Побудуйте канонічний суматор на базі мультиплексорів 4→1.
126. Побудуйте канонічний суматор на базі мультиплексорів 2→1, використовуючи каскадне з'єднання цих мультиплексорів.
127. Побудуйте канонічний суматор на базі дешифратора і мультиплексорів 2→1.
128. Виконайте порівняння апаратних витрат на реалізацію канонічного суматора в різних базисах.
129. Доведіть коректність логічних виразів:

$\overline{S} = \overline{a}be + a\overline{b}e + ab\overline{e} + \overline{a}\overline{b}\overline{e}$;

$\overline{E} = a\overline{b}\overline{e} + \overline{a}b\overline{e} + \overline{a}\overline{b}e + \overline{a}\overline{b}\overline{e}$;

$\overline{S} = \overline{\overline{a}be + \overline{a}b\overline{e} + a\overline{b}\overline{e} + abe}$;

$\overline{E} = \overline{\overline{a}be + a\overline{b}e + ab\overline{e} + abe}$;

$E = \overline{a}be + a\overline{b}e + ab\overline{e} + abe$.

130. Доведіть коректність логічних виразів:

$S = \overline{\overline{a}be + a\overline{b}e + ab\overline{e} + \overline{a}\overline{b}\overline{e}}$;

$E = \overline{a\overline{b}\overline{e} + \overline{a}b\overline{e} + \overline{a}\overline{b}e + \overline{a}\overline{b}\overline{e}}$;

$S = \overline{a}\overline{b}e + \overline{a}b\overline{e} + a\overline{b}\overline{e} + abe$;

131. Дайте визначення однобітного суматора.
132. Для чого використовуються суматори?
133. Поясніть, як створити таблицю істинності однорозрядного двійкового суматора?
134. Поясніть, в чому полягає властивість самоподвійності логічних функцій.
135. Доведіть самоподвійність функції суми.
136. Доведіть самоподвійність функції переносу.
137. Яка операція виконується за допомогою суматорів?
138. Визначити реакцію суматора при використанні інверсних значень всіх вхідних змінних.
139. Для чого викорсовується індикатор інверсії на вході УГП суматора?
140. За якою формулою визначається результат додавання в поточному біті суматора?
141. Приведіть УГП однобітного суматора з прямими виводами доданків і суми та інверсними виводами переносів.
142. Доведіть коректність логічних виразів:

$$\overline{a \oplus b} = \overline{a}\overline{b} + ab; \qquad \overline{\overline{a} \oplus \overline{b}} = \overline{a \oplus b};$$

$$a \oplus \overline{b} = \overline{a \oplus b}; \qquad 1 \oplus a = \overline{a}; \qquad a \oplus \overline{a} = 1;$$

$$a \oplus b = (a + b)\overline{ab}; \quad a \oplus b = \overline{(a + b)} \oplus \overline{ab};$$

143. Виконати реалізацію логічної функції $S = a \oplus b \oplus e$ в базисі 3І- НІ.
144. Виконати реалізацію логічної функції $S = (\overline{a \oplus b}) \oplus \bar{e}$ в базисі І-АБО-НІ.
145. Виконати порівняння апаратних витрат на побудову мінімальних суматорів в різних базисах.
146. Визначити логічний вираз функції переносу мінімального суматора з використанням ДНФ.
147. Визначити логічний вираз функції переносу мінімального суматора з використанням КНФ.
148. Привести карту Карно та виконати мінімізацію функції переносу мінімального суматора.
149. Приведіть і поясніть формулу для реалізації переносу мінімального суматора з трактом $\bar{e} \rightarrow E$.
150. Приведіть і поясніть формулу для реалізації переносу мінімального суматора з трактом $e \rightarrow \overline{E}$.

Тести для самоперевірки

В завданнях, що приведені нижче, вказати одну правильну відповідь:

10. Визначити значення виразу $\bar{a} \ \& \ a$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

11. Визначити значення виразу $1 \oplus a$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

12. Визначити значення виразу $\bar{a} \vee a$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

13. Визначити значення виразу $\bar{a} \oplus a \oplus a$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

14. Визначити значення виразу $\bar{a} \oplus 1 \oplus a$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

15. Визначити значення виразу $\bar{a} \oplus a \oplus \bar{a}$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

16. Визначити значення виразу $\bar{a} \oplus 1 \oplus \bar{a}$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

17. Визначити значення виразу $0 \oplus a \oplus 1 \oplus a$

A) 1; B) $\bar{a}$*; C) 0; D) a; E) немає правильної відповіді.*

18. Виконати спрощення логічного виразу $y = a \vee \bar{a} d \vee \bar{a} c$

A) $\bar{a} \vee d \vee c$*; B)* $\bar{a}$*; C)* $a \vee d \vee c$ *; D)* $\bar{a} d \vee \bar{a} c$*;*

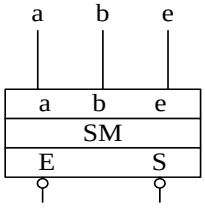
E) немає правильної відповіді.

19. Визначити значення суми $11_2 + 10_8 + 11_{16}$ в двійковій системі числення (індекс позначає основу системи числення, наприклад, 10_8 позначає, що число записано у вісімковій системі числення).

A) 10000; B) 11000; C) 11100; D) 11110;

E) немає правильної відповіді.

20. Задано УГП двійкового суматора. Визначити формулу для виходу $\overline{E}$.

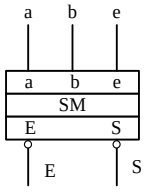


A) $ab \vee ae \vee be$; *B)* $(a \vee b)(a \vee e)(b \vee e)$;

C) $\bar{a}\,\bar{b} \vee b\bar{e} \vee a\bar{e}$; *D)* $\bar{a}\,\bar{b} \vee \bar{a}\,\bar{e} \vee \bar{b}\,\bar{e}$;

E) немає правильної відповіді.

21. Задано УГП двійкового суматора. Визначити формулу для виходу $\overline{S}$.



A) $a\overline{E} \vee b\overline{E} \vee e\overline{E} \vee \bar{a}\,\bar{b}\,\bar{e}$; *B)* $\bar{a}\,be \vee a\bar{b}\,e \vee ab\bar{e} \vee \bar{a}\,\bar{b}\,\bar{e}$;

C) $a\overline{E} \vee b\overline{E} \vee e\overline{E} \vee abe$; *D)* $(a \vee b \vee E)\,\bar{e} \vee \bar{a}\,\bar{b}\,\bar{e}$;

E) немає правильної відповіді.

22. Вкажіть таблицю істинності для виходу суми *S* однорозрядного суматора

a	*b*	*e*	***A)*** *S*	***B)*** *S*	***C)*** *S*	***D)*** *S*
0	*0*	*0*	*1*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*	*1*	*1*
0	*1*	*0*	*0*	*0*	*1*	*1*
0	*1*	*1*	*1*	*1*	*0*	*0*
1	*0*	*0*	*0*	*0*	*1*	*1*
1	*0*	*1*	*1*	*1*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*	*0*
1	*1*	*1*	*0*	*1*	*1*	*0*

E) немає правильної відповіді.

23. Вкажіть таблицю істинності для виходу переносу *E* однорозрядного суматора

a	*b*	*e*	***A)***	*E*	***B)***	*E*	***C)***	*E*	***D)***	*E*
0	*0*	*0*		*1*		*0*		*0*		*1*
0	*0*	*1*		*0*		*0*		*1*		*1*
0	*1*	*0*		*0*		*0*		*1*		*1*
0	*1*	*1*		*1*		*1*		*0*		*0*
1	*0*	*0*		*0*		*0*		*1*		*1*
1	*0*	*1*		*1*		*1*		*0*		*0*
1	*1*	*0*		*1*		*1*		*0*		*0*
1	*1*	*1*		*0*		*1*		*1*		*0*

E) немає правильної відповіді.

24. Вкажіть таблицю істинності для виходу суми $\overline{S}$ однорозрядного суматора

a	*b*	*e*	***A)***	$\overline{S}$	***B)***	$\overline{S}$	***C)***	$\overline{S}$	***D)***	$\overline{S}$
0	*0*	*0*		*1*		*0*		*0*		*1*
0	*0*	*1*		*0*		*0*		*1*		*1*
0	*1*	*0*		*0*		*0*		*1*		*1*
0	*1*	*1*		*1*		*1*		*0*		*0*
1	*0*	*0*		*0*		*0*		*1*		*1*
1	*0*	*1*		*1*		*1*		*0*		*0*
1	*1*	*0*		*1*		*1*		*0*		*0*
1	*1*	*1*		*0*		*1*		*1*		*0*

E) немає правильної відповіді.

25. Вкажіть таблицю істинності для виходу переносу $\overline{E}$ однорозрядного суматора

a	*b*	*e*	***A)***	$\overline{E}$	***B)***	$\overline{E}$	***C)***	$\overline{E}$	***D)***	$\overline{E}$
0	*0*	*0*		*1*		*0*		*0*		*1*
0	*0*	*1*		*0*		*0*		*1*		*1*
0	*1*	*0*		*0*		*0*		*1*		*1*
0	*1*	*1*		*1*		*1*		*0*		*0*
1	*0*	*0*		*0*		*0*		*1*		*1*
1	*0*	*1*		*1*		*1*		*0*		*0*
1	*1*	*0*		*1*		*1*		*0*		*0*
1	*1*	*1*		*0*		*1*		*1*		*0*

E) немає правильної відповіді.

26. Яку арифметичну операцію можна виконати однобітним суматором (*a*, *b* – доданки; *e* – вхідний перенос)?

A) a + e; B) a + b - e; C) a + b; D) a + b + e;

E) немає правильної відповіді.

27. Яка властивість логічної функції суми двійкового суматора визначається за виразом:

$$f_s(a, b, e) = \overline{f_s(\bar{a},\bar{b},\bar{e})}$$

A) асоціативність; *B) дистрибутивність;*

C) самоподвійність; *D) комутативність;*

E) немає правильної відповіді.

28. Яка властивість логічної функції переносу двійкового суматора визначається за виразом:

$$f_E(a, b, e) = \overline{f_E(\bar{a},\bar{b},\bar{e})}$$

A) асоціативність; *B) дистрибутивність;*

C) самоподвійність; *D) комутативність;*

E) немає правильної відповіді.

29. Визначити вираз, за яким формується сигнал *S* канонічного суматора:

A) $\bar{a}\bar{b}e \vee \bar{a}b\bar{e} \vee a\bar{b}\bar{e} \vee abe;$ *B)* $ab\bar{e} \vee a\bar{b}e \vee \bar{a}be \vee abe;$

C) $\overline{\bar{a}\bar{b}e \vee \bar{a}b\bar{e} \vee a\bar{b}\bar{e} \vee abe};$ *D)* $\overline{ab\bar{e} \vee a\bar{b}e \vee \bar{a}be \vee abe};$

E) немає правильної відповіді.

Навчальне видання

Лахно Валерій Анатолійович
Гусєв Борис Семенович
Смолій Віктор Вікторович
Місюра Максим Дмитрович
Касаткін Дмитро Юрійович

Технології проектування комп'ютерних систем

Навчальний посібник

Підп. до друку 26.11.2019.
Формат 60x84 1/16.
Папір офсет. Друк офсет.
Ум. друк. арк. 11.50
Наклад 100 прим.

Видавництво та друк: «КОМПРИНТ»
73033, м. Київ а/с 15
e-mail komprint@ukr.net
Свід. ХС №2 від 16.08.2000 р.