

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Запорізький національний технічний університет

## **МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт з дисципліни  
"Комп'ютерна логіка"  
для студентів спеціальності 123 "Комп'ютерна інженерія"  
всіх форм навчання  
Частина 1

2018

Методичні вказівки до виконання лабораторних робіт з дисципліни "Комп'ютерна логіка" для студентів спеціальності 123 "Комп'ютерна інженерія" всіх форм навчання, частина 1/ Укл. С.С. Грушко, М.П. Проскурін - Запоріжжя: ЗНТУ, 2018. – 42 с.

Укладачі: С.С. Грушко, асистент  
М.П. Проскурін, к.т.н., доцент

Рецензент: Зеленцова І.Я., к.т.н., доцент

Відповідальний  
за випуск: С.С. Грушко, асистент

Затверджено  
на засіданні кафедри  
“Комп'ютерні системи  
та мережі”  
Протокол № 9  
від “17” травня 2018.

Рекомендовано до видання  
НМК факультету комп'ютерних  
наук і технологій  
Протокол № 10  
від “28” травня 2018.

## ЗМІСТ

|                                                             |    |
|-------------------------------------------------------------|----|
| 1 ЛАБОРАТОРНА РОБОТА №1 .....                               | 4  |
| ВИВЧЕННЯ СХЕМ БУЛЕВИХ ФУНКЦІЙ ДВОХ ЗМІННИХ .....            | 4  |
| 1.1 Основні положення.....                                  | 4  |
| 1.2 Варіанти індивідуальних завдань .....                   | 7  |
| 1.3 Порядок виконання роботи.....                           | 7  |
| 1.4 Приклад виконання роботи.....                           | 8  |
| 1.5 Зміст звіту.....                                        | 11 |
| 1.6 Контрольні запитання .....                              | 11 |
| 2 ЛАБОРАТОРНА РОБОТА №2 .....                               | 12 |
| ОБОЛОНКА ПРОЕКТУВАННЯ VHDL. ОСОБЛИВОСТІ                     |    |
| СТВОРЕННЯ СХЕМ ПРОЕКТІВ .....                               | 12 |
| 2.1 Основні положення: мова VHDL і оболонка Active HDL..... | 12 |
| 2.2 Варіанти індивідуальних завдань .....                   | 14 |
| 2.3 Порядок виконання роботи.....                           | 14 |
| 2.3.1 Створення проекту в оболонці Active HDL. ....         | 15 |
| 2.3.2 Покрокова робота з меню .....                         | 15 |
| 2.3.3 Додавання сигналів до проекту.....                    | 16 |
| 2.3.4 Додавання логічної схеми .....                        | 19 |
| 2.3.5 Функціональне моделювання .....                       | 21 |
| 2.4 Етапи виконання роботи .....                            | 25 |
| 2.5 Зміст звіту.....                                        | 25 |
| 2.6 Контрольні запитання .....                              | 26 |
| 3 ЛАБОРАТОРНА РОБОТА №3 .....                               | 27 |
| КОМБІНАЦІЙНІ СХЕМИ .....                                    | 27 |
| 3.1 Основні положення.....                                  | 27 |
| 3.2 Варіанти індивідуальних завдань .....                   | 38 |
| 3.3 Порядок виконання роботи.....                           | 40 |
| 3.4 Зміст звіту.....                                        | 41 |
| 3.5 Контрольні запитання .....                              | 41 |
| ЛІТЕРАТУРА .....                                            | 42 |

## 1 ЛАБОРАТОРНА РОБОТА №1 ВИВЧЕННЯ СХЕМ БУЛЕВИХ ФУНКЦІЙ ДВОХ ЗМІННИХ

**Метою лабораторної роботи №1** є вивчення булевих функцій однієї або двох змінних, створення **математичних моделей функцій алгебри логіки** (ФАЛ), їх мінімізація та ознайомлення з імітаційним або симуляційним моделюванням в середовищі **Electronics Workbench** (EWB, демоверсія 5.04 і вище) [1] або **MultiSim** (MS, демоверсія 7.0 і вище) [2], отримання навичок роботи з ними.

### 1.1 Основні положення

**Математичний апарат**, який описує дії дискретних пристроїв, базується на алгебрі логіки, названої по імені автора - англійського математика Дж.Буля (1815–1864), - булевою алгеброю. Першим знайшов практичне застосування **функцій алгебри логіки** (ФАЛ) американський вчений К. Шеннон у 1938 р. (дослідження електричних кіл з контактними вимикачами). Для формального опису цифрових автоматів (ЦА) використовується апарат алгебри логіки. Логічною (булевою) змінною називається величина, яка може приймати тільки два значення «0» і «1». Сукупність різних значень вхідних  $X$  (або вихідних  $Y$ ) змінних називаються **набором кодів** (або алфавітом ЦА).

Основним **предметом булевої алгебри** є висловлювання – просте твердження, про яке можна стверджувати: воно істинне («1») або хибне («0»). Вхідні змінні позначають буквами, наприклад  $x_1, x_2, \dots, x_m$ , вони з'єднуються між собою **логічними операціями** і формують **кортежі змінних у вигляді мінтермів, макстермів**, ін. -- підфункції, сукупність яких є математичною моделлю у вигляді ФАЛ.

**Завданням ФАЛ** є аналіз, синтез математичної моделі і структурне моделювання будь-яких дискретних систем. **Логічна операція** – це чітко визначена дія над одним (декількома) аргументами вхідного слова  $X$  (з розрядністю  $x_1, x_2, \dots, x_m$ ), яка створює результат цифрового автомату (ЦА) - вихідне слово  $Y$  (з розрядністю  $y_1, y_2, \dots, y_n$ ). У булевій операції вхідний операнд  $X$  і результат  $Y$  можуть набувати значення «1» або «0». Булеві функції можуть залежати від однієї, двох і в цілому  $n$  – вхідних змінних. Булева функція  $n$ -аргументів може мати до  $M = 2^n$  наборів вхідних

значень. Оскільки функції приймають тільки два значення, загальне число булевих функцій  $n$  аргументів дорівнює  $2^n$ . Основними **булевими функціями** (розгорнутий базис Буля) є логічні функції: **заперечення** – операція  $\neg$  (інверсія), **диз'юнкція** – операція  $\vee$  (логічне додавання) і **кон'юнкція** – операція  $\wedge$  (логічне множення). Їх комбінації формують **мінімальні базиси**: схеми  $\neg\vee$ -НІ (базис Пірса) і  $\neg\wedge$ -НІ (базис Шеффера), де  $n$  – кількість вхідних змінних  $X$  (аргументів). Також важливу роль відіграють функції: **виключне  $\neg\vee$**  (сума mod2) і еквіваленція (інверсія функції виключне  $\neg\vee$ ), які входять у базис Жегалкіна. Функції імплікації зустрічаються рідко, бо використовуються лише для двох аргументів.

Таблиця 1.1 – Назви логічних функцій ( $Y_1$ – $Y_{16}$ )

| $X_1$<br>$X_2$ | <b>0 0 1 1</b><br><b>0 1 0 1</b> | <b>Позначення</b>     | <b>Назва</b>                             |
|----------------|----------------------------------|-----------------------|------------------------------------------|
| $Y_1$          | 0 0 0 0                          | 0                     | Константа 0                              |
| $Y_2$          | 0 0 0 1                          | $x_1 x_2$             | Кон'юнкція, логічне множення, І          |
| $Y_3$          | 0 0 1 0                          | $x_1 \leftarrow x_2$  | Заперечення імплікації (заборона $x_2$ ) |
| $Y_4$          | 0 0 1 1                          | $x_1$                 | Повторення першого аргументу             |
| $Y_5$          | 0 1 0 0                          | $x_2 \leftarrow x_1$  | Заперечення оберненої імплікації         |
| $Y_6$          | 0 1 0 1                          | $x_2$                 | Повторення другого аргументу             |
| $Y_7$          | 0 1 1 0                          | $x_1 \oplus x_2$      | Сума за модулем 2, виключне АБО          |
| $Y_8$          | 0 1 1 1                          | $x_1 + x_2$           | Диз'юнкція, логічне додавання, АБО       |
| $Y_9$          | 1 0 0 0                          | $x_1 \downarrow x_2$  | Стрілка Пірса                            |
| $Y_{10}$       | 1 0 0 1                          | $x_1 \sim x_2$        | Еквіваленція                             |
| $Y_{11}$       | 1 0 1 0                          | $\bar{x}_2$           | Заперечення другого аргументу            |
| $Y_{12}$       | 1 0 1 1                          | $x_2 \rightarrow x_1$ | Обернена імплікація                      |
| $Y_{13}$       | 1 1 0 0                          | $\bar{x}_1$           | Заперечення першого аргументу            |
| $Y_{14}$       | 1 1 0 1                          | $x_1 \rightarrow x_2$ | Імплікація                               |
| $Y_{15}$       | 1 1 1 0                          | $x_2 / x_1$           | Штрих Шеффера                            |
| $Y_{16}$       | 1 1 1 1                          | 1                     | Константа 1                              |

Для спрощення логічних функцій (ЛФ) використовують основні закони алгебри логіки та правила їх перетворень, що наведені для функцій лАБО та лІ (табл. 1.2 та 1.3 відповідно).

Таблиця 1.2 – Основні закони алгебри логіки

| Закон                            | Логічного додавання (лАБО)                                                                   | Логічного множення (лІ)                                                                      |
|----------------------------------|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Комута-<br>тивності              | $x_1 + x_2 = x_2 + x_1$                                                                      | $x_1 \cdot x_2 = x_2 \cdot x_1$                                                              |
| Асоціа-<br>тивності              | $(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$                                                      | $(x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3)$                                      |
| Дистрибу-<br>тивності            | $(x_1 + x_2) \cdot x_3 = x_1 \cdot x_3 + x_2 \cdot x_3$                                      | $x_1 \cdot x_2 + x_3 = (x_1 + x_3) \cdot (x_2 + x_3)$                                        |
| Ідемпотен-<br>ції                | $x + x = x$                                                                                  | $x \cdot x = x$                                                                              |
| Закон де<br>Моргана              | $\overline{x_1 + x_2 + \dots + x_n} = \bar{x}_1 \cdot \bar{x}_2 \cdot \dots \cdot \bar{x}_n$ | $\overline{x_1 \cdot x_2 \cdot \dots \cdot x_n} = \bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n$ |
| Наслідок<br>закону де<br>Моргана | $x_1 + x_2 + \dots + x_n = \overline{\bar{x}_1 \cdot \bar{x}_2 \cdot \dots \cdot \bar{x}_n}$ | $x_1 \cdot x_2 \cdot \dots \cdot x_n = \overline{\bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n}$ |

Таблиця 1.3 – Правила перетворень ЛФ

| Правило                   | Диз'юнкція                                                                    | Кон'юнкція                                                                                        |
|---------------------------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Інверсії                  | $0 = \bar{1}$                                                                 | $1 = \bar{0}$                                                                                     |
| Незмінності               | $x + 0 = x$                                                                   | $x \cdot 1 = x$                                                                                   |
| Універсальності           | $x + 1 = 1$                                                                   | $x \cdot 0 = 0$                                                                                   |
| Повторення                | $x + x = x$                                                                   | $x \cdot x = x$                                                                                   |
| Доповнення                | $x + \bar{x} = 1$                                                             | $x \cdot \bar{x} = 0$                                                                             |
| Поглинання                | $x + xy = x$                                                                  | $x(x+y) = x$                                                                                      |
| Подвійного<br>заперечення | $\overline{\bar{x}} = x$                                                      |                                                                                                   |
| Склеювання                | $xy + x\bar{y} = x$<br>$x + \bar{x}y = x + y$<br>$\bar{x} + xy = \bar{x} + y$ | $(x+y)(x+\bar{y}) = x$<br>$(x+y)(x+z) = x+yz$<br>$x(\bar{x}+y) = xy$<br>$\bar{x}(x+y) = \bar{x}y$ |

## **1.2 Варіанти індивідуальних завдань**

Індивідуальне завдання студента включає синтез ФАЛ і моделювання роботи схеми двох логічних функцій з табл. 1.1, що задані викладачем.

Якщо не отримано інших вказівок, то одна функція після спрощення подається у базисі Пірса, інша – у базисі Шеффера.

До кожної ФАЛ будуються 2 схеми. Одна з використанням віртуальних елементів, що мають спрощені умовні графічні позначення, наступна – з використанням аналогів реальних мікросхем.

При правильно виконаному завданні обидві схеми повинні працювати ідентично.

## **1.3 Порядок виконання роботи**

1.3.1 Перш ніж приступити до завдання по створенню, реалізації, моделюванню і дослідженню схем булевих функцій двох змінних на ПК доцільно ознайомитися з вступним описом користувача програмного забезпечення EWB або MS.

1.3.2 Записати таблицю істинності (ТІ) заданої функції і згідно неї записати ДДНФ і ДКНФ функції.

1.3.3 Провести спрощення (мінімізацію) обох рівнянь, користуючись правилами, що наведені в табл. 1.2 і 1.3. Кінцевий вираз обох рівнянь повинен співпадати, це свідчить про правильність спрощення.

1.3.4 Подати спрощенні ФАЛ у заданому базисі (Пірса, Шеффера або ін.), для чого за необхідністю можна скористатись законом де Моргана.

1.3.5 За отриманим рівнянням, що є математичною моделлю (виразом) ФАЛ побудувати у середовищі EWB або MS логічну схему функції з віртуальних логічних елементів. Перевірити правильність роботи схеми підключивши до неї інструменти формування та візуалізації сигналів.

1.3.6 Побудувати аналогічну схему, використовуючи аналоги з бібліотеки моделей (БМ) EWB або MS реальних схем. Впевнитися в тому, що обидві схеми працюють ідентично.

### 1.4 Приклад виконання роботи

Запишемо ТІ, задану викладачем у такому вигляді:

| $x_1$ | $x_2$ | $y$ |
|-------|-------|-----|
| 0     | 0     | 1   |
| 0     | 1     | 0   |
| 1     | 0     | 0   |
| 1     | 1     | 1   |

За ТІ запишемо досконалу диз'юнктивну та досконалу кон'юнктивну форми функції та проведемо їх мінімізацію:

$$\text{СДНФ } y = \overline{x_1} \cdot \overline{x_2} + x_1 x_2$$

$$\text{СКНФ } y = (x_1 + x_2)(\overline{x_1} + \overline{x_2}) = x_1 x_2 + \overline{x_1} \cdot \overline{x_2}$$

Перетворимо мінімізоване рівняння у вигляд, що відповідає базису Пірса або базису Шеффера:

$$y = \overline{\overline{x_1 + x_2 + x_1 + x_2}} \quad (\text{базис Пірса}),$$

$$y = \overline{\overline{x_1 x_2} \cdot \overline{x_1} \cdot \overline{x_2}} \quad (\text{базис Шеффера}).$$

За отриманим рівнянням побудуємо логічну схему та схему із використанням аналогів реальних мікросхем.

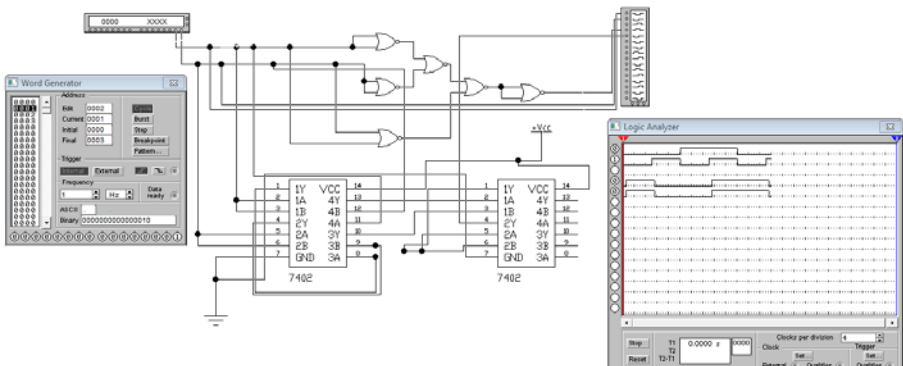


Рисунок 1.1 – Схема ФАЛ у базисі Пірса

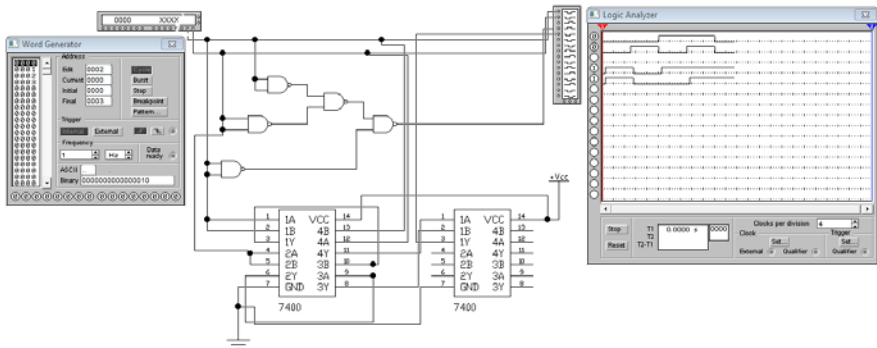


Рисунок 1.2 – Схема ФАЛ у базісі Шеффера

Зверніть увагу, що схеми можна будувати, використовуючи один прилад формування сигналу, та один прилад візуалізації сигналу.

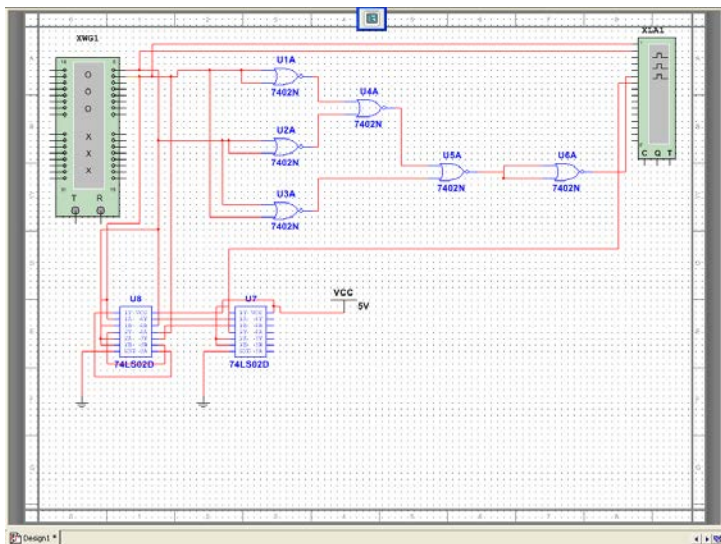


Рисунок 1.3 – Приклад побудови схеми ФАЛ у базісі Пірса в програмному середовищі Multisim.

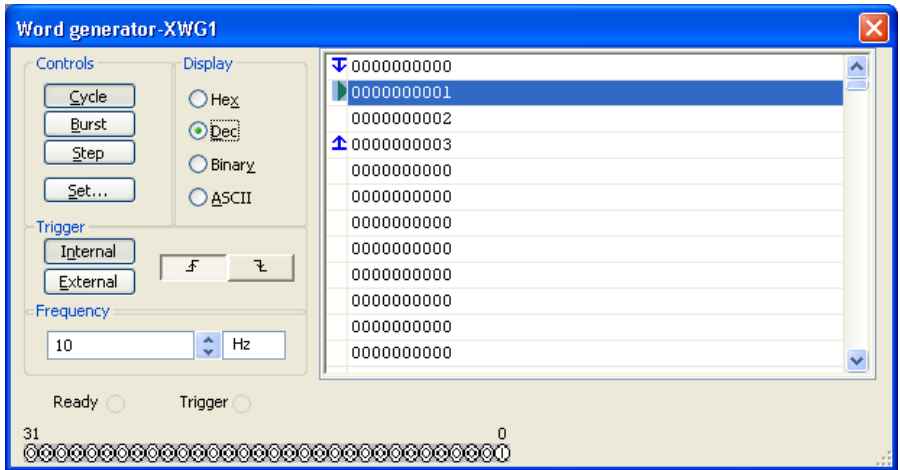


Рисунок 1.4 – Налаштування генератора слів

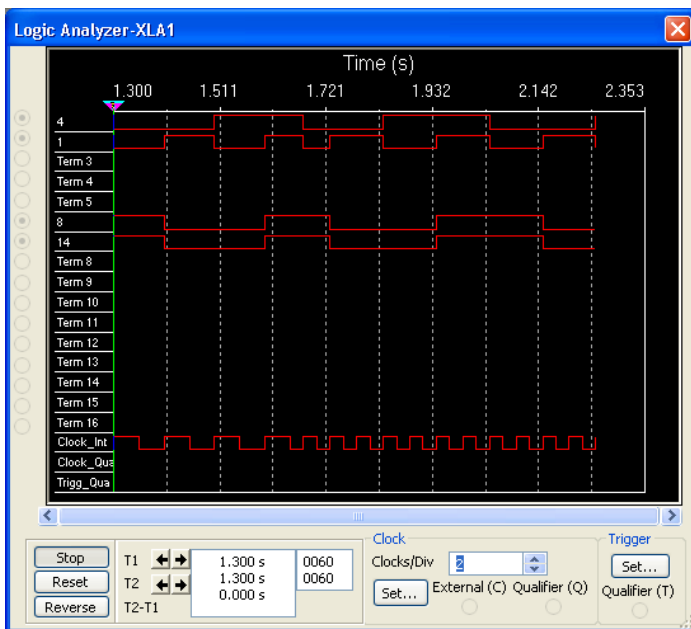


Рисунок 1.5 – Графік, отриманий в логічному аналізаторі

### 1.5 Зміст звіту

1.5.1 Короткий теоретичний огляд по елементарним двовходовим логічним схемам, їх ТІ.

1.5.2 Етапи аналізу, синтезу і мінімізації отриманих ДНФ (КНФ) до мінімізованих ДНФ і КНФ з поясненнями.

1.5.3 Роздруківки графічних матеріалів, що переконливо і повно ілюструють функціонування схеми відповідно до ТІ.

1.5.4 Визначення типу використаних мікросхем, їх призначення, особливості з погляду виконуваних функцій, назви серій цифрових інтегральних схем, типу логіки (ЕЗЛ, Т<sup>2</sup>Л, Т<sup>2</sup>ЛШ, К-МДН, І<sup>2</sup>Л, ін.).

1.5.5 Виводи за отриманими даними з аналізом функціональних, часових характеристик, а також типу логіки (струми споживання схеми, напруга живлення, ін.).

При утрудненнях у виконанні ЛР скористатися літературою [1-8].

### 1.6 Контрольні запитання

1.6.1 Дайте визначення логічних функцій: інверсія, диз'юнкція, кон'юнкція, штрих Шеффера, стрілка Пірса, еквівалентність, константа «1» («0»), імплікація, сума за модулем 2, ін.

1.6.2 Зобразіть умовні графічні позначення логічних елементів відомих типів.

1.6.3 Запишіть ТІ заданих логічних елементів.

1.6.4 Визначте вихідний стан логічних елементів при заданих вхідних сигналах та навести ТІ.

1.6.5 Визначте якому логічному елементу належить ТІ.

1.6.6 Які логічні елементи можна використати для отримання функції інверсії?

1.6.7 Що таке ДДНФ і ДКНФ, для чого вони використовуються та чим відрізняються від ДНФ і КНФ?

## 2 ЛАБОРАТОРНА РОБОТА №2

### ОБОЛОНКА ПРОЕКТУВАННЯ VHDL. ОСОБЛИВОСТІ СТВОРЕННЯ СХЕМ ПРОЕКТІВ

**Метою лабораторної роботи** є ознайомлення з мовою **VHDL** і оболонкою **Active HDL** (інструмент опису та моделювання цифрових пристроїв для автоматизованого проектування) на прикладі створення булевих функцій двох змінних (див. ЛР №1) та отримання первинних навичок роботи з ними.

#### 2.1 Основні положення: мова VHDL і оболонка Active HDL

У сучасних комплексах автоматизованого проектування цифрових систем у вигляді великих інтегральних схем (BIC), надвеликих (NBIC) або багаторівневих програмувальних логічних IC (ПЛИС) і замовних кристалів, широке застосування одержала мова **VHDL** (Very high speed integrated circuits **H**ardware **D**escription Language) для формального опису синтезованих проектів.

Мова VHDL є вхідною мовою для багатьох сучасних систем автоматизованого проектування (САПр), вона призначена для опису процесу моделювання цифрових систем і використовується на алгоритмічному і логічному етапах проектування ЦА. Мова VHDL дозволяє описувати поведінку, тобто алгоритми функціонування цифрових схем, систем на їх основі, проводити ієрархічне, структурне моделювання, описувати асинхронні і синхронні паралельні процеси, регулярні структури і ін. [15-23]. Стандарти мови:

- VHDL-87 (ANSI/IEEE Std 1076-1987), розроблений у США;
- VHDL-93 (ANSI/IEEE Std 1076-1993), орієнтований на проектування цифрових систем;
- VHDL-AMS (ANSI/IEEE Std 1076.1-1999), орієнтований на цифрові і аналогові системи.

Попередньо VHDL розроблялася як мова моделювання. Однак проблема синтезу логічних схем стала центральною і тому був розроблений ряд систем автоматичного синтезу схем за описами мовою VHDL. Однією з найбільш популярних є система Leonardo (автоматичний синтез логічних схем).

Алгоритмічне і логічне проектування ЦА з використанням мови VHDL включає наступні етапи:

- складання і моделювання алгоритмічного VHDL-опису проекту майбутньої цифрової системи;
- автоматичний синтез логічної схеми в заданій бібліотеці елементарних і цільових функцій;
- повторне моделювання логічної схеми з метою верифікації (підтвердження правильності) поводження отриманої логічної схеми вузла в статичному і/або динамічних режимах.

VHDL підтримує три різних стилі опису апаратних архітектур:

- структурний опис (structural description), у якому архітектура представляється у вигляді ієрархії зв'язаних компонентів;
- потоковий опис (data-flow description), у якому архітектура представляється у вигляді множини паралельних реєстрових операцій, кожна з яких управляється вентильними сигналами;
- поведінковий опис (behavioral description), у якому перетворення описується послідовними програмними пропозиціями (схожі на ін. існуючі в будь-якій мові програмування високого рівня).

Всі ці стилі можуть спільно використатися в одній архітектурі. Active HDL [15 - 23] являє собою середовище проектування, розробки і створення моделей цифрових схем, пристроїв на основі ядра логічного моделювання, що підтримує текстове і графічне введення проектів (на мовах VHDL, Verilog, EDIF, Systemc, Systemverilog, їх комбінаціях). Система проектування Active HDL є провідним середовищем для створення проектів і їх моделювання для всіх сімейств програмувальних інтегральних схем ПЛІС, забезпечуючи гнучкість підходу і надаючи розвинені функції підтримки найбільш складних сучасних проектів. Середовище проектування Active HDL дає розробникам незалежність у використанні інших засобів проектування з одного повністю інтегрованого оточення. Система Active HDL структурно побудована навколо свого блоку керування маршрутом проектування, який дозволяє розробникам легко одержувати доступ до засобів проектування і бібліотекам, які вони використовують у проекті. Active HDL різних версій поставляється з бібліотеками моделей виробників ПЛІС, які вже прокомпільовані і готові до використання в проектах. Блок керування проектом Active HDL дозволяє виконувати всі модифікації і операції над проектом з єдиного оточення; така інтеграція дає моніторинг процесу проектування і зберігає час, усуваючи необхідність у запуску безлічі віконних інтерфейсів і процесів. Інтеграція додатків забезпечує

контроль над проектом від специфікації на всьому шляху до його фізичної реалізації. Active HDL має інтерфейси з усіма провідними програмними продуктами, надаючи розробникам вільний вибір у використанні засобів проектування, що найбільше відповідають вимогам конкретного проекту. Постачальник Active HDL - компанія Aldec, яка заснована в 1984р. і займається розробкою програмного і апаратного забезпечення при автоматизації проектування мікроелектронних пристроїв.

## 2.2 Варіанти індивідуальних завдань

Індивідуальне завдання студенту задається на початку ЛР викладачем. Запропоновані для синтезу функціональні вузли (табл. 1.1 ЛР №1) належать до елементарних КС з двома входами  $X = \{x_1, x_2\}$  і одним виходом  $Y$ , кожний з яких описується своєю булевою функцією у вигляді ДДНФ (ДКНФ).

Оптимальна схема може бути отримана лише при мінімізації булевої функції.

Мінімізовану функцію необхідно реалізувати у вигляді VHDL-коду та за допомогою логічної схеми. Провести верифікацію – підтвердити моделюванням вірність функціонування схем згідно ТІ.

## 2.3 Порядок виконання роботи

Перш ніж приступити до завдання по створенню, реалізації, моделюванню і дослідженню схем булевих функцій двох змінних на екрані ПК за допомогою мови опису ЦА VHDL і оболонки **Active HDL (демоверсія)**, доцільно виконати наступне:

- провести ознайомлення з мовою VHDL та її особливостями [13 - 22];
- ознайомиться із вступним описом користувача програми VHDL в оболонці Active HDL [13, 14];
- виконати необхідні етапи синтезу схем індивідуального завдання, з огляду на заданий логічний базис. Якщо базис не заданий, то проєктант вільний у виборі елементів базису;
- виконати показові приклади моделювання схем у програмі VHDL на основі обраних цифрових IC, що є в каталозі Work\_VHDL.

### 2.3.1 Створення проекту в оболонці Active HDL.

Після запуску програми Active-HDL з'явиться вікно САПР. У головному меню програми оберіть команду **File**. Можна або відкрити один з існуючих робочих просторів (workspace), або створити новий. Для створення нового робочого простору оберіть **New > Workspace**. З'явиться вікно майстра **New Workspace wizard** (рис. 2.1). Введіть ім'я **робочого простору** (наприклад, **comp**), в якому буде знаходитись новий **проект** – поведінкова модель зовнішнього модуля пам'яті. Активізуйте опцію **Add New Design to Workspace** (додати новий проект до робочого простору). Натисніть **OK**.

### 2.3.2 Покрокова робота з меню

Якщо робочий простір існує, то в меню **File** оберіть команду **Open New Workspace / Design Explorer**. З'явиться вікно майстра нового проекту **New Design Wizard**, в якому визначається спосіб створення ресурсів проекту. Виберіть **Create an empty design** (створити порожній проект) і натисніть **Next**.

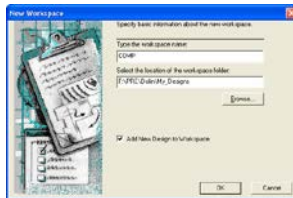


Рисунок 2.1 – Майстер New Workspace

У наступному вікні **New Design Wizard** вказується інформація про інструменти синтезу і реалізації проекту, а також цільове сімейство мікросхем програмованої логіки. Перевірте, що обрана мова опису моделі є **VHDL**. Решту налаштувань залиште без змін (**none**). Натисніть **Next**.

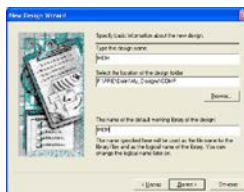


Рисунок 2.2 – Введення назви проекту

У наступному вікні **New Design Wizard** (рис. 2.2) треба вказати назву проекту (**MEM**) і встановити його місцезнаходження. Майстер створить робочу бібліотеку з тією ж назвою. Ця бібліотека міститиме всі компільовані компоненти VHDL-проекту. Натисніть **Next**.

В останньому вікні **New Design Wizard** виводиться інформація про налаштування, що були зроблені за допомогою майстра. Підготовчий етап проектування закінчений. Натисніть **Finish**.

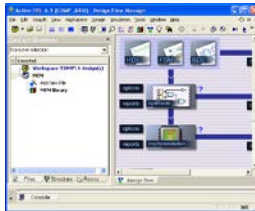


Рисунок 2.3— Вікно Design Browser

З'явиться вікно **Design Browser**, в якому відображається ієрархічна, деревоподібна структура порожнього проекту **MEM**. Якщо натиснути на піктограмі **+** перед назвою проекту, то з'явиться піктограма **Add New File** (рис. 2.3).

### 2.3.3 Додавання сигналів до проекту

Перейдемо до розробки файлів VHDL-проекту. Двічі натисніть на піктограмі **Add New File** у вікні **Design Browser**. У вікні **Add New File** оберіть сторінку **Wizards** і в ній оберіть **VHDL Source Code Wizard** (рис. 2.4). Натисніть **OK**.



Рисунок 2.4 – Запуск майстра VHDL Source Code Wizard

З'явиться вікно майстра **New Source File Wizard**. Переконайтеся, що настройка **Add generated file to design** (додати створюваний файл до поточного проекту) активна і натисніть **Next**. У наступному вікні (рис. 2.5) треба вказати ім'я створюваного файлу, *сутності (entity)* і *архітектури (architecture)* моделі. Введіть **MEM** в полі імені файлу і натисніть **Next**.



Рисунок 2.5 – Вибір імені файлу

Щоб додати новий *порт (сигнал)* в модель, натисніть кнопку **New**, надрукуйте ім'я сигналу в полі імені і оберіть напрям його розповсюдження (**in** – для входів, **out** – для виходів, **inout** – для двонаправлених портів). Якщо порт багаторозрядний (з'єднаний із шиною), треба вказати його розрядність за допомогою операцій **Array Indexes**, як показано на рис. 2.6.

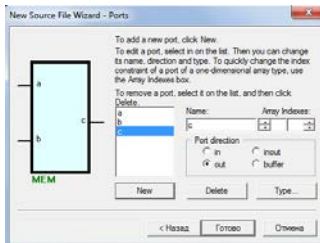


Рисунок 2.6 – Створення портів

Кнопка **Type** дає змогу змінювати тип сигналу (залиште типи **STD\_LOGIC** і **STD\_LOGIC\_VECTOR**). За допомогою кнопки **Delete** можна видаляти помилковий запис. Кожен блок повинен мати входні (**IN**), та вихідні (**OUT**) порти. В даному випадку блок *сутності*

(**entity**) у вигляді MEM може мати два вхідних (**a**, **b**) і один вихідний (**c**) порти. Така структура портів блоку співпадає з логічними функціями такими як: **2AND**; **2OR**; **2XOR**; **2AND-NOT**; **2OR NOT**, **ін.** Для функції **NOT** – дещо інша структура, вона має лише один вхідний і один вихідний порти (рис. 2.7).

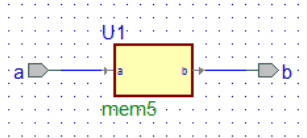


Рисунок 2.7 – Схема для сигнала NOT

Після натиску кнопки **Finish** створюється шаблон файлу опису моделі мовою VHDL (рис. 2.8).

Шаблон містить оголошення бібліотеки підпрограм багаторівневої логіки **USE IEEE.std\_logic\_1164.ALL**, опис портів і порожню область опису архітектури. Встановить курсор у розділі опису архітектури (у рядку *-- enter your statements here --*) і наберіть текст **опису архітектури (architecture)**, наприклад: `c <= not(a) and b`.

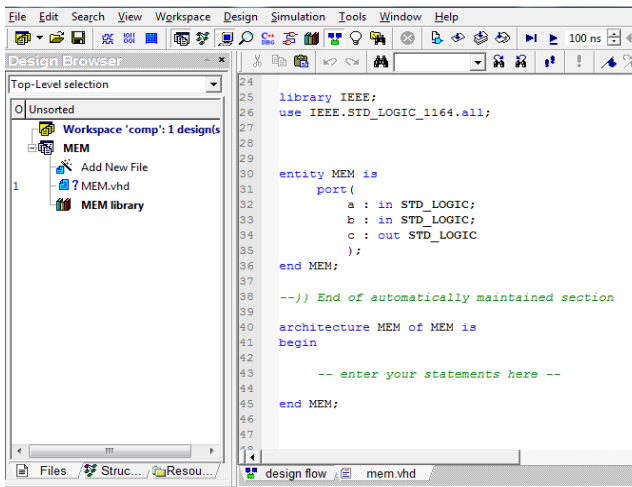


Рисунок 2.8 – Шаблон опису моделі (**entity & architecture**) в текстовому редакторі

Програма мовою VHDL створена і готова до компіляції. Щоб запустити компіляцію файла (перевірка вірності того, що зроблено), натисніть правою кнопкою миші на його імені у вікні **Design Browser**, і оберіть команду **Compile** контекстного меню. Зелений значок поряд з ім'ям файла і відповідне повідомлення у вікні **Console** сигналізують про те, що компіляція виконана успішно. Компоненти моделі, що успішно відкомпільовано, автоматично додаються до робочої бібліотеки проекту.

#### 2.3.4 Додавання логічної схеми

Логічна схема є наочним документом і порядок її складання є корисним для навчання. Для створення логічної схеми двічі натисніть на піктограмі **Add New File** у вікні **Design Browser**. У новому вікні (рис. 2.9) оберіть спосіб опису моделі – **Block Diagram Wizard**. Модель буде представлена у вигляді логічної схеми. Натисніть **OK**.

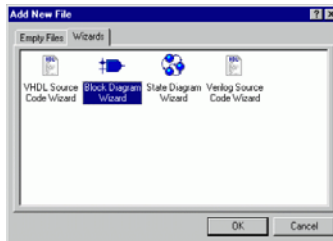


Рисунок 2.9 – Запуск майстра Block Diagram Wizard

Переконайтеся, що у вікні **New Source File Wizard** активізована настройка (*Add generated file to design*) і натисніть **Next**.

Логічна схема, яку створить майстер, може бути перетворена в один з трьох форматів опису: EDIF, VHDL і Verilog. Оберіть мову **VHDL** і натисніть **Next**.

У наступному вікні вкажіть ім'я файла **top** натисніть **Next**.

На наступному кроці треба задати порти схеми, користуючись кнопкою **New**. Логічна схема має два вхідних порти: а - IN STD\_LOGIC і b - IN STD\_LOGIC, і один вихідний порт с - OUT STD\_LOGIC. Введіть їх і натисніть **Finish**.

У вікні **Design Browser** з'явиться новий файл – **top.bde**. Шаблон нової логічної схеми буде автоматично завантажений у вікно редактора BDE (рис. 2.10). Він містить три порти: **a**, **b**, **c**.

Інструментальна панель редактора BDE містить кнопку  **Symbol toolbox**, яка дозволяє поміщати у логічну схему раніше згенеровані моделі – *блоки*. Натисніть на кнопці **Symbol toolbox**. Зверніть увагу, що у вікні, яке з'явилося (рис. 2.10 а), вже є блок **mem**.

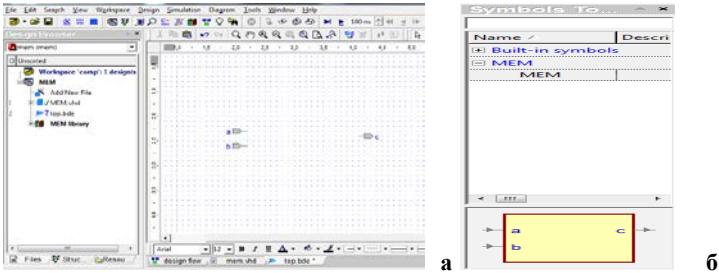


Рисунок 2.10 – Шаблон логічної схеми - а); вікно Symbol Toolbox – б)

Перетягніть блок **mem** на робоче поле редактора BDE, як показано на рис. 2.10 б. Закрийте вікно **Symbol Toolbox**.

Виконаємо з'єднання компонентів логічної схеми. Зв'язки (провідники) прокладаються за допомогою інструменту  **Wire** редактора BDE. Для виконання з'єднання треба активізувати кнопку **Wire** панелі інструментів, натиснути на контакт-джерело, перемістити курсор і натиснути на контакт-одержувач. До логічної схеми буде додано з'єднання. Щоб видалити помилково виконане з'єднання, треба використовувати інструмент **Select**. За його допомогою виділіть компонент логічної схеми і, натиснувши кнопку **DELETE** клавіатури, видаліть його. Зробіть з'єднання, як показано на рис. 2.11 а, б.



Рисунок 2.11 – Компоненти логічної схеми: розміщення – а); з'єднання – б)

Виконаємо компіляцію файлу логічної схеми **top.bde**. У вікні **Design Browser** натисніть на імені файлу і оберіть команду меню

**Compile All with File Reorder** (компіляція всього проекту з перебудовою файлу). Якщо трансляція пройшла успішно, біля імені **top.bde** з'явиться зелена піктограма.

### 2.3.5 Функціональне моделювання

На початку функціонального моделювання треба визначити *об'єкт верхнього рівня* (Top-Level) – пару описів **сутності (entity)** і **архітектури (architecture)**, відносно якої це моделювання буде виконуватися. У даному випадку моделюється поведінка всієї мікропроцесорної системи, яка відповідає об'єкту **mem (mem)**. У вікні **Design Browser** розкрийте зміст компоненту **mem**, натиснувши символ  перед його ім'ям. Оберіть пару **mem (mem)** і натисніть **правою** кнопкою миші. У контекстному меню оберіть команду **Set as Top-Level** (встановити в якості файлу верхнього рівня).

Виконайте ініціалізацію моделювання: оберіть команду **Simulation→Initialize Simulation** головного меню САПР. Після ініціалізації стане активною сторінка **Structure-** вікно **Design Browser**.

Результати моделювання можуть бути збережені у «рідному» форматі часових діаграм САПР Activee-HDL – файлі, що має розширення **.awf**. Для цього треба вибрати команду **File → Save** головного меню. Також є можливість експортувати результати моделювання у файл *VHDL-процесу*, який має розширення **.vhs**. Ці файли можна використовувати для створення *випробувального стенду* VHDL (test bench) – це програма мовою VHDL, в якій описані *тестові послідовності* (закон зміни вхідних сигналів, наприклад згідно ТП). У САПР Activee-HDL тестові дані для моделювання можна або читати з текстового файлу, або експортувати з файлу *VHDL-процесу*. Фактично **випробувальний стенд** – це файл мовою VHDL, який додається до створеного проекту і займає найвищий ступінь його ієрархії. За рахунок цього поведінка усіх компонентів VHDL-проекту підпорядковується тестовій послідовності, яку **він** створює. За його допомогою можна виконувати аналіз сигналів на виході і його використання дозволяє поліпшити якісні показники функціонального моделювання за рахунок автоматизації. Методика забезпечує повнішу перевірку проекту з мінімальною взаємодією із користувачем і виключає з результатів помилки, пов'язані з «людським фактором». Вихідний VHDL-проект, роботу якого перевіряють за допомогою випробувального стенда, зветься **UUT – Unit Under Test** (пристрій, що тестується).

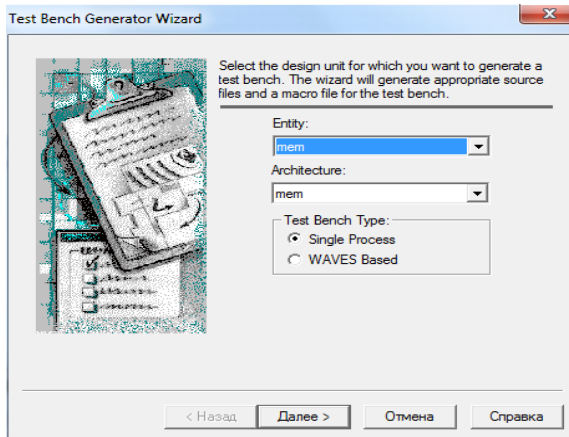


Рисунок 2.12 – Вибір типу випробувального стенда

У вікні **Design Browser** виділіть пару **mem (mem)** верхнього рівня VHDL-проекту. Натисніть на ній правою кнопкою миші і у контекстному меню оберіть команду **Generate Testbench** згенерувати випробувальний стенд).

З'явиться перше вікно майстра **Testbench Generation Wizard** (рис. 2.12). У даному випадку тип випробувального стенда повинен бути *Single Process* (окремий процес) - натисніть **Next** і перейдіть до наступного вікна.

Наступне вікно майстра залишаємо без змін. Для продовження роботи натисніть **Next..**

У наступному вікні майстра можна задати імена **entity**, **architecture** і VHDL-файлу випробувального стенда. Тут же задається назва директорії для його зберігання. Залиште настройки, що використовуються за умовчанням і, натиснувши **Next**, перейдіть до наступного вікна.

Останнє вікно майстра **Testbench Generation Wizard** містить інформацію про результати своєї роботи. До проекту будуть додані два файли, що створені автоматично:

- **mem\_TB.vhd** – файл випробувального стенду;
- **mem\_TB\_runtest.do** – командний файл для запуску теста.

Тут можна вказати файл конфігурації для часового моделювання.

Оскільки нашою задачею є не часове, а функціональне моделювання (перевірка згідно ТІ), то настройку *Generate* залиште невідбраною (рис. 2.13). У кінці роботи з майстром **Testbench Generation Wizard** натисніть на кнопці **Finish**. Дерево проекту, що відображається на сторінці **Files** вікна **Design Browser**, з'явиться директорія **Testbench**, що містить файли випробувального стенда.

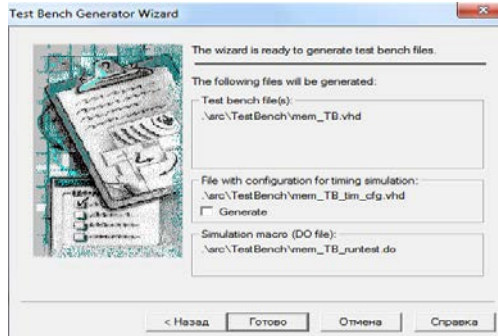
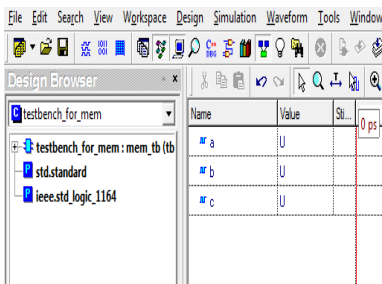


Рисунок 2.13 – Вікно Testbench Generation Wizard

Якщо у вікні **Design Browser** обрати файл **top\_TB\_runtest.do** і, натиснувши на ньому правою кнопкою, вибрати у контекстному меню команду **Execute**, то буде запущений процес трансляції і моделювання тестової послідовності випробувального стенда. При цьому відкриється вікно часових діаграм - Waveform editor (рис. 2.14).



а



б

Рисунок 2.14 – Waveform editor (Таблиці можливих часових діаграм а, б)

Оберіть сигнал **а** і, натиснувши на ньому **правою** кнопкою, у

контекстному меню оберіть команду **Stimulators**. З'явиться вікно **Stimulators**, де можна задавати тестові послідовності для обраних сигналів (рис. 2.15).

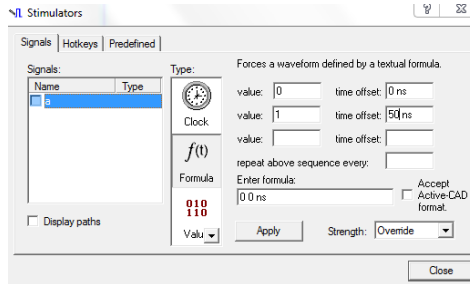


Рисунок 2.15 – Настроювання стимулятора для сигналу **a**

В полі *Type* вікна **Stimulators** для сигналу **reset** оберіть тип стимулятора – **Formula**. У відповідне вікно введіть формулу **0 0 ns, 1 50 ns**. Цей запис означає, що у час моделювання **0 ns** сигнал **a** має рівень логічний «0». В момент часу **50 ns** він змінюється на логічну «1». Для підтвердження внесених змін натисніть **Apply**. Таку саму послідовність дій виконайте для сигналу **b**.

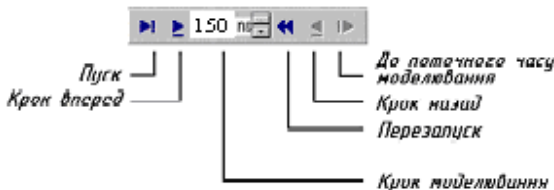


Рисунок 2.16 – Кнопки управління моделюванням

Для виконання моделювання в САПР ActiveHDL використовуються п'ять кнопок, розташованих на панелі інструментів (рис. 2.16). Зверніть увагу на комбінований список поряд з кнопкою  **Run for** - це є час одного кроку моделювання в наносекундах (нс).

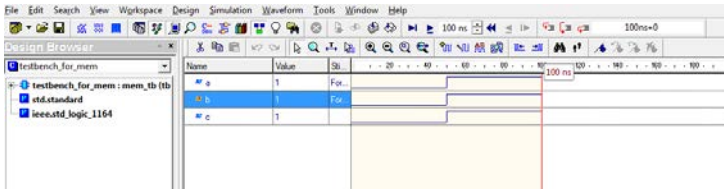


Рисунок 2.17 – Результат автоматичного тестування для двох груп значень входів (input) **a**, **b** (00, 11) і одного виходу (output) **c** (0, 1).

Натиснувши кнопку  **Run for**, виконайте моделювання при заданих настройках вхідних сигналів протягом 100 наносекунд. Результати будуть відображені у вікні **Waveform Editor** (рис. 2.17). Перевірте вірність функціонування логічної схеми згідно з ТІ і збережіть файл часової діаграми за допомогою команди **File > Save**. Для перебору усіх варіантів наборів змінних (00, 01, 10, 11) треба підібрати послідовність 0, 1 (тривалість і момент змін сигналів **a**, **b**).

## 2.4 Етапи виконання роботи

2.4.1 Перш ніж приступити до завдання по створенню, реалізації, моделюванню й дослідженню схем булевих функцій двох змінних на екрані ПК за допомогою оболонки **Active HDL (демоверсія)**, доцільно ознайомиться із вступним описом користувача оболонки Active HDL.

2.4.2 Виконати необхідні етапи синтезу схем індивідуального завдання (по аналогії з ЛР № 1), отримати мінімізовану форму заданих функцій, подати їх за допомогою текстового опису VHDL та логічних схем.

2.4.3 Реалізувати задані функції за допомогою мови VHDL, провести верифікацію проекту.

2.4.4 Реалізувати задані функції за допомогою схемотехнічного редактора, провести верифікацію проекту. При утрудненнях у виконанні ЛР скористатися літературою [9-11].

## 2.5 Зміст звіту

2.5.1 Короткий теоретичний огляд мови VHDL і оболонки Active HDL.

2.5.2 Етап аналізу, синтезу і мінімізації отриманих ДНФ (КНФ) з поясненнями.

2.5.3 Текст програми на VHDL та скрін реалізації схеми у схемотехнічному редакторі.

2.5.4 Файл(и) моделювання розробленої схеми із необхідними графічними матеріалами, що повно ілюструють функціонування логічних схем згідно до ТП.

2.5.5 Виводи за отриманими даними з аналізом функціональних, часових характеристик, типу логіки.

## 2.6 Контрольні запитання

2.6.1 Дайте визначення відомих логічних функцій, їх назву і зобразіть їх умовні графічні позначення.

2.6.2 Надайте та поясніть три варіанти опису ФАЛ, їх особливості.

2.6.3 Зробіть характеристику мови **VHDL** і її основних положень.

2.6.4 Кратко опишіть оболонку Active HDL: основне призначення, характеристики, особливості і відмінність від оболонки EWB або MS.

2.6.5 Назвіть порядок дій при створенні тестового стенду.

2.6.6 Назвіть способи завдання сигналів у тестовому стенді.

2.6.7 Поясніть, як проводиться верифікація проекту.

### 3 ЛАБОРАТОРНА РОБОТА №3 КОМБІНАЦІЙНІ СХЕМИ

**Метою лабораторної роботи** є засвоєння основних етапів синтезу й створення ряду типових комбінаційних схем (КС) у вигляді розробки й моделювання простих вузлів обчислювальних пристроїв (ОП).

#### 3.1 Основні положення

Електронні схеми в обчислювальній техніці за принципом функціонування прийнято поділяти на два типи – комбінаційні й послідовні (цифрові автомати).

**Комбінаційною** називають схему з  $m$  входами й  $n$  виходами, для якої сукупність вихідних сигналів – вектор  $Y = y_1 \dots y_n$  у цей момент часу повністю визначається сукупністю вхідних сигналів – вектором  $X = x_1 \dots x_m$  для цього ж моменту часу  $t_i$  і не залежить від вхідних сигналів, що діють у попередні моменти часу  $t_{i-1}$ .

Розглянемо основні комбінаційні схеми, що використовуються у комп'ютерній техніці.

**Дешифратором** (синонім - декодер, DC) називають функціональний пристрій комп'ютера, призначений для перетворювання кожної комбінації вхідного двійкового входу в управляючий сигнал тільки на одному із своїх виходів.

Вихід дешифратора, на якому з'являється управляючий сигнал, називають активним. Якщо значення сигналу на активному виході дорівнює «1», то на всіх інших пасивних виходах дешифратора сигнали дорівнюють «0».

Активний вихід дешифратора в інтегральному виконанні часто відображається значенням логічного «0», а на решті пасивних виходах він дорівнює «1».

Дешифратори класифікують за основними і другорядними ознаками:

- структурною організацією – одноступеневі й багаступеневі;
- форматом вхідного коду – двійкові, двійково-десяткові, ін.;
- розрядністю коду, який дешифрується – 2, 3, 4, ...,  $n$ ;
- кількістю використаних із  $m$  виходів – повні й неповні дешифратори;

- формою подачі вхідного коду – з однофазним, парофазними входами;

- видом вхідних стробуючих сигналів (прямий «1», інверсний «0», ін.);

- типом використаної логіки – базис Буля, Шефера, Пірса;

В комп'ютерах дешифратори використовують для виконання таких операцій:

а) дешифрації коду операції, записаної в регістр команд процесора, що забезпечує вибір необхідної мікропрограми;

б) перетворення коду адреси операнду команди в управляючі сигнали вибору комірки пам'яті в процесі запису або читання інформації;

в) заперечення/дозвіл візуалізації на зовнішніх пристроях;

г) реалізації логічних операцій, побудови мультиплексорів і демультимплексорів.

Умовне графічне позначення та алгоритм роботи дешифратора подані рис. 3.1 та таблицею істинності (табл. 3.1), яка приведена нижче.

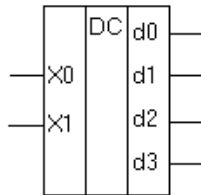


Рисунок 3.1 – Умовне графічне позначення (УГП) дешифратора

Таблиця 3.1 - Таблиця істинності дешифратора

| $x_0$ | $x_1$ | $d_0$ | $d_1$ | $d_2$ | $d_3$ |
|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 1     | 0     | 0     | 0     |
| 0     | 1     | 0     | 1     | 0     | 0     |
| 1     | 0     | 0     | 0     | 1     | 0     |
| 1     | 1     | 0     | 0     | 0     | 1     |

**Шифратором** (синонім - кодер, CD) називають функціональний пристрій комп'ютера, призначений для перетворення вхідного  $m$ -розрядного унітарного позиційного коду у вихідний  $n$ -розрядний двійковий позиційний код.

Двійкові шифратори виконують функцію, обернену функції дешифратора. При активізації одного із входів шифратора на його виходах формується код, який відображає номер активного входу. Повний двійковий шифратор має  $m=2^n$  входів і  $n$ -виходів.

В комп'ютерах шифратори використовують для виконання наступних функцій:

- а) перетворення унітарного вхідного коду у вихідних двійковий позиційний код;
- б) вводу десяткових даних з клавіатури комп'ютера;
- в) показу старшої одиниці в слові;
- г) передачі між різними пристроями при обмеженні на кількість ліній зв'язку.

Умовне графічне позначення (УПГ) та алгоритм роботи шифратора подані рис. 3.2 та таблицею істинності (табл. 3.2), яка приведена нижче.

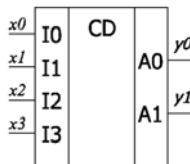


Рисунок 3.2 – УПГ шифратора

Таблиця 3.2 - Таблиця істинності (ТІ) шифратора

| $x_0$ | $x_1$ | $x_2$ | $x_3$ | $y_1$ | $y_0$ |
|-------|-------|-------|-------|-------|-------|
| 1     | 0     | 0     | 0     | 0     | 0     |
| 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | 0     | 1     | 0     |
| 0     | 0     | 0     | 1     | 1     | 1     |

Функції DC, CD взаємно пов'язані своєю зворотністю (див.ТІ 3.1, 3.2).

**Мультиплексором** (рис. 3.3, позначається MS, MX) називають функціональний пристрій комп'ютера, призначений для послідовної комутації інформації від одного із  $n$  входів на його спільний вихід.

Входи мультиплексора поділяють на інформаційні й керуючі (адресні). Конкретний вхід мультиплексора, що підключається до його виходу, визначається адресним кодом  $A_0, A_1, \dots, A_{n-1}$ . Зв'язок між числом інформаційних  $n$  і адресних  $m$  ходів впливає із співвідношення  $n=2^m$ .

Мультиплексори в комп'ютерах використовують для виконання наступних операцій:

- а) комутації як окремих ліній, так і шин передачі інформації;
- б) перетворення паралельного коду в послідовний;
- в) реалізації різних логічних функцій;
- г) формування функцій збудження тригерів ЦА з можливістю їх розширення до кількох  $(1, 2 \dots m)$  алгоритмів роботи відповідного ЦА.

Алгоритм роботи мультиплексора описується таблицею істинності (табл. 3.3), яка приведена нижче.

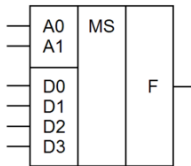


Рисунок 3.3 - УГП мультиплексора

Таблиця 3.3 - Таблиця істинності мультиплексора

| $A_0$ | $A_1$ | $D_0$ | $D_1$ | $D_2$ | $D_3$ | F   |
|-------|-------|-------|-------|-------|-------|-----|
| 0     | 0     | 0/1   | x     | x     | x     | 0/1 |
| 0     | 1     | x     | 0/1   | x     | x     | 0/1 |
| 1     | 0     | x     | x     | 0/1   | x     | 0/1 |
| 1     | 1     | x     | x     | x     | 0/1   | 0/1 |

**Демультимплексором** (рис. 3.4, позначається DMS, DMX) називають функціональний пристрій комп'ютера, призначений для комутації сигналу з одного інформаційного входу D на один із  $n$  інформаційних виходів.

Виходи демультиплексора, також, як і в мультиплексорів, поділяють на інформаційні та керуючі. Номер виходу, на який у кожний момент часу передається значення вхідного сигналу, визначається адресним кодом  $A_0, A_1, \dots, A_{n-1}$ . Адресні входи  $m$  та інформаційні входи  $n$  пов'язані співвідношенням  $n=2^m$ .

Демультиплексор використовують для виконання таких операцій:

- а) комутації як окремих ліній, так і шин передачі інформації в комп'ютерах;
- б) перетворення послідовного коду в паралельний;
- в) реалізації логічних функцій.

Алгоритм роботи демультиплексора описується таблицею істинності (табл. 3.4), яка приведена нижче.

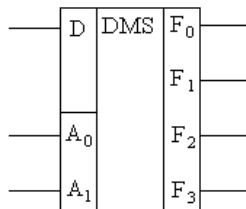


Рисунок 3.4 – УГП демультиплексора

Таблиця 3.4 - Таблиця істинності демультиплексора

| D   | A <sub>1</sub> | A <sub>0</sub> | F <sub>3</sub> | F <sub>2</sub> | F <sub>1</sub> | F <sub>0</sub> |
|-----|----------------|----------------|----------------|----------------|----------------|----------------|
| 0/1 | 0              | 0              | 0              | 0              | 0              | 0/1            |
| 0/1 | 0              | 1              | 0              | 0              | 0/1            | 0              |
| 0/1 | 1              | 0              | 0              | 0/1            | 0              | 0              |
| 0/1 | 1              | 1              | 0/1            | 0              | 0              | 0              |

Функції MS, DMS взаємно пов'язані своєю зворотністю (див.ТІ 3.3, 3.4).

**Суматором** називають комбінаційний логічний пристрій комп'ютера, призначений для виконання операцій арифметичного додавання чисел (операндів X і Y), поданих у вигляді двійкових кодів.

Операція віднімання в суматорі замінюється операцією додавання двійкових чисел у додатковому або оберненому коді. Операція множення і ділення зводиться до багатократного додавання і зсуву. Тому суматор є важливою частиною арифметично-логічного пристрою (АЛП) процесора і комп'ютера. Функцію суматора позначають буквами SM або  $\Sigma$ .

**Повним однорозрядним суматором (ПОС)** називають логічну схему, яка виконує додавання значень  $i$ -х розрядів  $x_i$  і  $y_i$  двійкових чисел (операндів X і Y) з урахуванням переносу  $P_i$  із молодшого сусіднього розряду і дає на виходах функцій результат  $S_i$  і перенос  $P_{i+1}$  в старший сусідній розряд (рис.3.5). На основі одно розрядних схем додавання на три входи і два виходи будуються багато розрядні суматори будь-якого типу. Суматор на  $n$  розрядів складається з окремих схем, які називають повний однорозрядний суматор. Ці схеми виконують усі дії з додавання (множення, віднімання, ділення, їх комбінації) значень однойменних розрядів двійкових чисел.

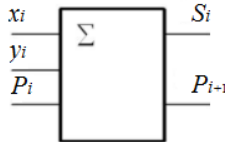


Рисунок 3.5 – УГП однорозрядного суматора

Алгоритм роботи однорозрядного суматора відображається таблицею істинності (табл. 3.6), яка приведена нижче.

Таблиця 3.5 - Таблиця істинності повного однорозрядного суматора (ПОС)

| $x_i$ | $y_i$ | $P_i$ | $S_i$ | $P_{i+1}$ |
|-------|-------|-------|-------|-----------|
| 0     | 0     | 0     | 0     | 0         |
| 0     | 0     | 1     | 1     | 0         |
| 0     | 1     | 0     | 1     | 0         |
| 1     | 0     | 0     | 1     | 0         |
| 0     | 1     | 1     | 0     | 1         |
| 1     | 0     | 1     | 0     | 1         |
| 1     | 1     | 0     | 0     | 1         |
| 1     | 1     | 1     | 1     | 1         |

**Напівсуматором** називають логічну схему, яка виконує додавання значень  $i$ -х розрядів  $x_i$  і  $y_i$  двійкових чисел (операндів  $X$  і  $Y$ ) при  $P_i = 0$  та реалізує на виході значення результату  $S_i$  і перенос в старший (сусідній) розряд  $P_{i+1}$ .

$$M_i = \overline{x_i} \cdot y_i \vee x_i \cdot \overline{y_i} = x_i \oplus y_i; \quad R = x_i \cdot y_i$$

Таким чином, напівсуматор виконує частину завдання підсумовування в  $i$ -му розряді, оскільки не враховує перенос ( $P_i = 0$ ) із сусіднього молодшого розряду і може бути створений на основі схеми ПОС (рис.3.5) при подаванні на вхід  $P_{i+1}$  значення 0 (низький рівень –«земля»).

Алгоритм роботи напівсуматора суматора відображається таблицею істинності (табл. 3.7), яка приведена нижче.

Таблиця 3.6 – Таблиця істинності напівсуматора

| $x_i$ | $y_i$ | $S_i$ | $P_{i+1}$ |
|-------|-------|-------|-----------|
| 0     | 0     | 0     | 0         |
| 0     | 1     | 1     | 0         |
| 1     | 0     | 1     | 0         |
| 1     | 1     | 0     | 1         |

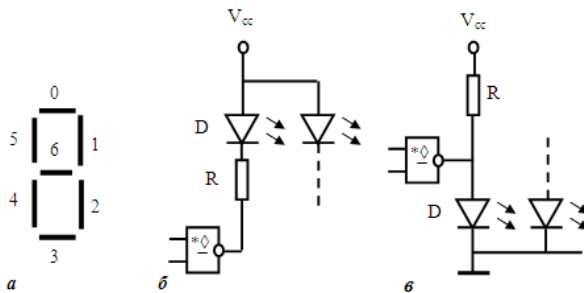
При синтезі комбінаційних схем (КС) виділяють такі етапи:

- формування логічних умов роботи розглянутої схеми або вузла шляхом складання ТІ (з урахуванням обраного логічного базису) для кожного з виходів комбінаційної схеми;
- одержання за ТІ або (на підставі використання методів мінімізації) мінімальних нормальних форм у мінімальній диз'юнктивній або кон'юнктивній формі (МДНФ або МКНФ);
- перетворення МДНФ або МКНФ до виду, що відповідає обраному (або заданому) набору логічних елементів (логічному базису) і побудова схеми КС.

**Сегментний індикатор** – індикатор, елементи відображення якого є сегментами, згрупованими в одне або кілька знакомісць (розрядів).

Сегмент – це елемент відображення інформації, контур якого являє собою прямі та (або) криві лінії, крапки, тире, ін. (наприклад, світлодіод, СД). Форма і положення сегментів на індикаторі розробляється спеціально для передачі певного набору символів або знаків. Символи на таких індикаторах формуються сукупністю кількох сегментів, рис.3.6 а. Семисегментний індикатор має сім сегментів для індикації цифри (у вісьмисегментного додається крапка/кома, дев'ятисегментного додано ще +/-, і так далі).

Семисегментні індикатори на СД, випускаються у двох різновидах за схемним рішенням: із загальним анодом (рис. 3.6 б) або загальним катодом (рис. 3.6 в). Для керування сегментами індикатора за схемою рис. 3.6 б є зручними логічні елементи з виходом типу «загальний колектор». У цьому випадку підключається зовнішній резистор, опір  $R$  якого задає струм світлодіода (СД) із урахуванням характеристик застосовуваних СД (сегмент світиться, коли вихідний транзистор керуючого елемента відмикається – рис. 3.6 б). На рис. 3.6 в, сегмент у вигляді СД світиться, коли вихідний транзистор керуючого елемента замикається. Перевагу мають схеми індикаторів із загальним анодом (рис. 3.6 б) через відсутність споживання, коли вони не світяться.



а) індикатор б) схема з загальним анодом в) схема з загальним катодом  
Рисунок 3.6 – Семисегментний індикатор (а) і схеми керування сегментом із загальним анодом (б) і загальним катодом (в).

Перетворювач коду 8-4-2-1 в 7-розрядний код семисегментного індикатора для подання 16-их чисел 0, ..., 9, A, b, C, d, E, F працює відповідно до його ТІ (див.таблицю 3.7).

Таблиця 3.7 – ТІ перетворювача коду 8-4-2-1 в 7-розрядний код семисегментного індикатора

| Код 16-й | Код 8-4-2-1 | 7-сегментний вихід |
|----------|-------------|--------------------|
|          |             | 6 5 4 3 2 1 0      |
| 0        | 0000        | 0 1 1 1 1 1 1      |
| 1        | 0001        | 0 0 0 0 1 1 0      |
| 2        | 0010        | 1 0 1 1 0 1 1      |
| 3        | 0011        | 1 0 0 1 1 1 1      |
| 4        | 0100        | 1 1 0 0 1 1 0      |
| 5        | 0101        | 1 1 0 1 1 0 1      |
| 6        | 0110        | 1 1 1 1 1 0 1      |
| 7        | 0111        | 0 0 0 0 1 1 1      |
| 8        | 1000        | 1 1 1 1 1 1 1      |
| 9        | 1001        | 1 1 0 1 1 1 1      |
| A        | 1010        | 1 1 1 0 1 1 1      |
| B        | 1011        | 1 1 1 1 1 0 0      |
| C        | 1100        | 0 1 1 1 0 0 1      |
| D        | 1101        | 1 0 1 1 1 1 0      |
| E        | 1110        | 1 1 1 1 0 0 1      |
| F        | 1111        | 1 1 1 0 0 0 1      |

**Перетворювач куду 8-4-2-1в код Грея і навпаки.** У деяких випадках необхідний числовий код, побудований таким чином, щоб при переході від одного числа до наступного змінювався б *завжди тільки один двійковий розряд*. Цією властивістю володіє код Грея, що використовується для розмітки координат кліток карт Карно, у перетворювачах кутового положення вала в цифровий код і в ін. випадках., де умовою його застосування є зміна *одного розряду* при збільшенні (зменшенні) коду на одиницю від вихідного (див. табл. 3.8). Код Грея не є позиційним, тобто ваги його розрядів не визначаються займаними ними місцями: це потребує в цифрових пристроях зворотнього перетворення коду Грея в код 8-4-2-1. Відповідність між 4-розрядним двійковим кодом 8-4-2-1 і кодом Грея відображена в табл. 3.8.

**Схема контролю парності.** Складність цифрових пристроїв (ЦП), ЦА визначає важливість моніторингу переданої/одержуваної цифрової інформації (авіаційно-космічні пристрої, військові системи, керування потужними енергоустановками, цифрові потоки Internet, телеметрія, наземні й супутникові канали зв'язку, дані банківських

операцій, медичних приладів, ін.) з метою запобігання, контролю і діагностики, виправлення помилок.

Таблиця 3.8 – Відповідність між 4-розрядним двійковим і кодом Грея.

| Двійковий код |            |            |            | Код Грея  |           |           |           |
|---------------|------------|------------|------------|-----------|-----------|-----------|-----------|
| $\alpha_1$    | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ |
| 0             | 0          | 0          | 0          | 0         | 0         | 0         | 0         |
| 0             | 0          | 0          | 1          | 0         | 0         | 0         | 1         |
| 0             | 0          | 1          | 0          | 0         | 0         | 1         | 1         |
| 0             | 0          | 1          | 1          | 0         | 0         | 1         | 0         |
| 0             | 1          | 0          | 0          | 0         | 1         | 1         | 0         |
| 0             | 1          | 0          | 1          | 0         | 1         | 1         | 1         |
| 0             | 1          | 1          | 0          | 0         | 1         | 0         | 1         |
| 0             | 1          | 1          | 1          | 0         | 1         | 0         | 0         |
| 1             | 0          | 0          | 0          | 1         | 1         | 0         | 0         |
| 1             | 0          | 0          | 1          | 1         | 1         | 0         | 1         |
| 1             | 0          | 1          | 0          | 1         | 1         | 1         | 1         |
| 1             | 0          | 1          | 1          | 1         | 1         | 1         | 0         |
| 1             | 1          | 0          | 0          | 1         | 0         | 1         | 0         |
| 1             | 1          | 0          | 1          | 1         | 0         | 1         | 1         |
| 1             | 1          | 1          | 0          | 1         | 0         | 0         | 1         |
| 1             | 1          | 1          | 1          | 1         | 0         | 0         | 0         |

Додавання функцій моніторингу завжди пов'язане з надмірністю – збільшенням кількості пристроїв та часу обробки для одержання інформації більш високої якості.

Завдання виявлення помилок вирішуються різними схемотехнічними способами. Цифровий пристрій за заданим алгоритмом забезпечує, наприклад, повторну передачу/прийом інформації (при виявленні помилки при передаванні/прийманні ЦП посилає запит на передачу/прийом блоку, у якому вона була). До схем контролю відносять: мажоритарні елементи,  $\sum \bmod 2$  і схеми з використанням кодів Хемінга.

ЦП на передачі/прийомі автоматично діагностує і (у деяких випадках) виправляє помилки. Одним з перевірених способів є  $n$ -кратне резервування каналу зв'язку ( $n$  – непарне число: 3, 5, 7, ...) з генерацією сигналу мажоритарного голосування (тобто дані вважаються безпомилковими з тих  $i$ -каналів з  $n$ , по яких відсутній сигнал помилки; при цьому  $i \geq (n+1)/2$ ).

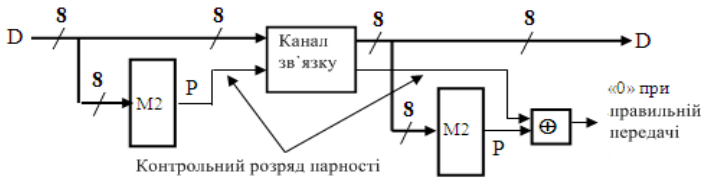


Рисунок 3.7 – Схема передачі даних і перевірки на парність на приймаючій стороні

Найпростіший спосіб контролю (виявлення) непарної кількості помилок при передачі/прийомі даних полягає в передачі розряду перевірки на парність  $P_{\text{пар.}}$  (непарність  $P_{\text{непар.}}$ ). На рис. 3.7 наведена спрощена схема контролю на парність 8-ми розрядного слова. У контрольний розряд (КР) парності (непарності) записується «0» («1»), якщо число одиниць в інформаційному слові парне (непарне) і навпаки, «1» («0») – у випадку непарного (парного) числа одиниць (якщо всі розряди  $D_i$  приймають значення «0», то в КР записується «0»). При цьому загальна кількість переданих одиниць по каналу зв'язку, включаючи КР, завжди парне. На приймаючій стороні КР обчислюється тим же способом для розрядів  $D_i$  і порівнюється з переданим КР. При розбіжності фіксується помилка передачі. Контроль на схемах  $\sum \text{mod} 2$  ефективний у ЦП, де ймовірність одиничної помилки набагато більша, ніж ймовірність подвійної. Виправити помилку при цій системі контролю неможливо, але підхід використання схем  $\sum \text{mod} 2$  може бути розвинуто для створення спеціальних кодів (лінійні, циклічні, ін.) з можливістю контролю і виправлення 1, 2, ...,  $m$  помилок в кодах.

**Мажоритарний елемент** (мажоритарний клапан, перемикач за більшістю однакових даних, ін.) – комбінаційна схема з класу порогових, частіше з непарним числом входів і одним (або кількома додатковими) вихідним сигналом, значення якого збігається зі значенням на більшості входів. Таким чином, елемент працює за «принципом більшості»: якщо на більшості входів буде сигнал «1», то і на виході схеми встановиться сигнал «1»; і навпаки, якщо на більшості входів буде сигнал «0», то і на виході встановиться «0». Мажоритарні пристрої звичайно мають непарну кількість входів (3, 5, 7...), використовуються в цифрових високонадійних системах і

пристроях, наприклад, в системах прийому/передавання, резервування, в перешкодостійких телекомунікаційних системах, ін.

Таблиця 3.9 – Таблиця істинності триканального мажоритарного елементу

| F <sub>1</sub> | F <sub>2</sub> | F <sub>3</sub> | F |
|----------------|----------------|----------------|---|
| 0              | 0              | 0              | 0 |
| 0              | 0              | 1              | 0 |
| 0              | 1              | 0              | 0 |
| 0              | 1              | 1              | 1 |
| 1              | 0              | 0              | 0 |
| 1              | 0              | 1              | 1 |
| 1              | 1              | 0              | 1 |
| 1              | 1              | 1              | 1 |

### 3.2 Варіанти індивідуальних завдань

**Варіант 1, 15.** Розробити та провести моделювання роботи схеми шифратора 8→3.

**Варіант 2, 16.** Розробити та провести моделювання роботи схеми двійкового дешифратора 3→8.

**Варіант 3, 17.** Розробити та провести моделювання роботи схеми повного однорозрядного суматора і напівсуматора..

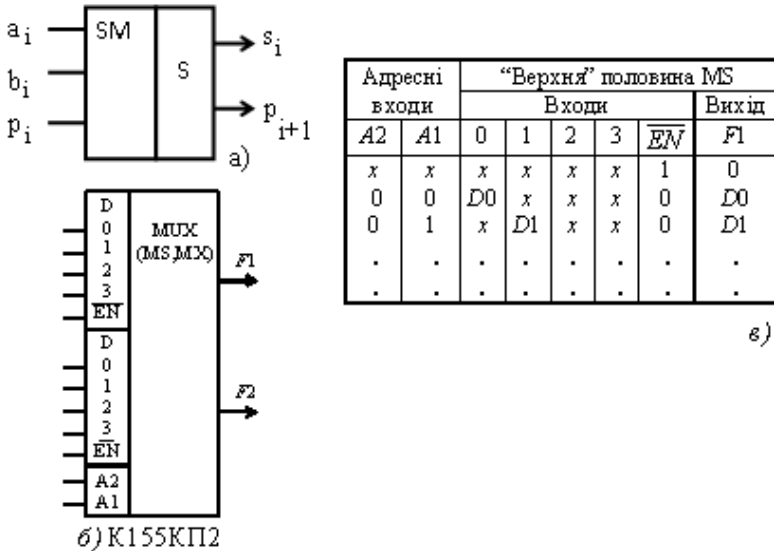
**Варіант 4, 18.** Розробити та провести моделювання роботи схеми мультиплексора 8→1.

**Варіант 5, 19.** Розробити та провести моделювання роботи схеми демультиплексора 1→8.

**Варіант 6, 20.** Розробити та провести моделювання роботи схеми повного однорозрядного суматора на двох мультиплексорах 8→1.

**Варіант 6, 20\*.** Розробити та провести моделювання роботи схеми повного однорозрядного суматора на здвоєному мультиплексорі 4→1.

*Пояснення:* здвоєний мультиплексор включає два мультиплексора, що керуються входом дозволу EN (тобто у визначений час може працювати два або один з мультиплексорів). Умовні графічні позначення суматора і здвоєного мультиплексора 4→1, а також ТІ мультиплексора із входом дозволу подані на рис. 3.8.



а) однорозрядний суматор б) 74153(К155КП2) в) таблиця істинності  
Рисунок 3.8 – Однорозрядний суматор, ПІ однієї половини здвоєного мультиплексора 74153(К155КП2), зображення схеми 74153(К155КП2).

**Варіант 7, 21.** Розробити та провести моделювання роботи схеми чотирьохрозрядного перетворювача прямого коду у обернений. Старший (п'ятий) розряд – знаковий. Необхідно пам'ятати, що позитивні числа мають однакове подання в обох кодах.

*Пояснення:* Інвертуванням цифрових розрядів  $a_i$  керує знаковий розряд  $a_5$  на основі використання логічної операції  $\sum \text{mod} 2$ :  $a_5 \oplus a_i$ ,  $i=1, 2, 3, 4$  для 4-х розрядного операнду А. При цьому необхідно реалізувати у звіті основний варіант рішення схеми й указати інші можливі варіанти.

**Варіант 8, 22.** Розробити та провести моделювання роботи схеми чотирьохрозрядного перетворювача прямого коду у доповняльний (через обернений). Старший (п'ятий) розряд – знаковий. Необхідно пам'ятати, що позитивні числа мають однакове подання в обох цих кодах.

*Пояснення:* див. варіант 7, 21.

Додавання одиниці можливо виконати із використанням 4-розрядного суматора на ІС 74283, 7483N (К155ИМ3) або іншим способом.

**Варіант 9, 23.** Розробити та провести моделювання роботи схеми перетворювача коду 8-4-2-1 у 7-розрядний код семисегментного індикатора.

**Варіант 10, 24.** Розробити та провести моделювання роботи схеми перетворювача 4-розрядного двійкового коду типу 8-4-2-1 у код Грея. Логічний базис - Буля з використанням елементу  $\sum mod 2$ . Використовуючи стандартну методику синтезу з мінімізацією на картах Карно, можна вирішити пряме завдання: перетворення коду 8-4-2-1 ( $a_1, a_2, a_3, a_4$ ) у код Грея ( $\beta_1, \beta_2, \beta_3, \beta_4$ ) - табл.3.8.

**Варіант 11, 25.** Розробити та провести моделювання роботи схеми перетворювача 4-розрядного двійкового коду Грея у код 8-4-2-1. Логічний базис - Буля з використанням елементу  $\sum mod 2$ . Використовуючи стандартну методику синтезу з мінімізацією на картах Карно, можна вирішити пряме завдання: перетворення коду Грея ( $\beta_1, \beta_2, \beta_3, \beta_4$ ) у код 8-4-2-1 ( $a_1, a_2, a_3, a_4$ ) – табл.3.8.

**Варіант 12, 26.** Розробити та провести моделювання роботи схеми пристрою контролю парності (непарності) при передачі даних. Відмінністю варіанту від схеми рис. 3.7 є 4-х розрядне слово і інверсія контрольного розряду парності на передавальній і приймальній стороні

**Варіант 13, 27.** Розробити схему триканального мажоритарного елементу (пристрою) голосування з видачею номеру відмовившого каналу ( в двійковому коді), див. табл. 3.7.

### 3.3 Порядок виконання роботи

3.3.1 Записати ТІ згідно завдання.

3.3.2 Скласти та мінімізувати (аналітичним методом, картами Карно або інш.) отримані логічні функції.

3.3.3 За отриманими логічними функціями побудувати у середовищі моделювання електронних схем (EWB або Multisim) логічну схему, забезпечивши відповідну комутацію входів, виходів, підключення живлення, кнопок і тумблерів, віртуальних вимірювальних приладів (двухканального осцилографа, амперметра, вольтметра, генератора слів, логічного аналізатора, ін. при необхідності).

3.3.4 Отримати діаграми функціонування (часові діаграми), що підтверджують ТІ.

3.3.5 Знайти у бібліотеці середовища моделювання електронних схем (EWB або Multisim) мікросхему, що відповідає розроблюваній. Перевірити коректність її функціонування за аналогією з п. 2.3.4. При утрудненнях у виконанні ЛР скористатися літературою [1-8].

### **3.4 Зміст звіту**

3.4.1 Короткий інформаційно-аналітичний огляд суті завдання.

3.4.2 Ті пристрою за відповідним завданням, його ДНФ (КНФ), мінімізація або спрощення складених функцій.

3.4.3 Скрін логічної схеми, побудованої у середовищі моделювання електронних схем (EWB або Multisim).

3.4.4 Скрін часової діаграми моделювання роботи пристрою.

3.4.5 Скрини логічної схеми і часових діаграм моделювання роботи мікросхем з бібліотеки моделей (БМ) (EWB [1] або MS [2].

3.4.4 Характеристика, призначення входів і виходів, умовне позначення обраної ІС (позначення входів, виходів, цокльовка виводів); назва, тип логіки, напруга живлення й аналоги (наприклад: схеми пАБО-НІ, пІ-НІ реалізовані наборами ІС типу 74..., 741..., 742..., ін. варіанти).

3.4.5 Висновки до роботи.

### **3.5 Контрольні запитання**

3.5.1 Дайте визначення терміну комбінаційна схема (КС).

3.5.2 Дайте визначення терміну КС дешифратор, та наведіть відповідну таблицю істинності, УГП.

3.5.3 Дайте визначення терміну КС шифратор, та наведіть відповідну таблицю істинності, УГП.

3.5.4 Дайте визначення терміну КС мультиплексор, та наведіть відповідну таблицю істинності, УГП.

3.5.5 Дайте визначення терміну КС демультимплексор, та наведіть відповідну таблицю істинності, УГП.

3.5.6 Дайте визначення терміну КС суматор/напівсуматор, та наведіть відповідну таблицю істинності, УГП.

3.5.7 Як записується ДДНФ (ДКНФ) і що вона позначає?

3.5.8 Переведіть задане рівняння в базис пІ-НІ, пАБО-НІ.

3.5.9 Яка максимальна кількість інформаційних входів у мультиплексора з двома (трьома, чотирма) адресними входами?

3.5.10 Скільки суматорів необхідно для операції додавання 2-х, 4-х, 8-ми розрядних чисел в прямому, додатковому кодах.

3.5.11 Принцип роботи семисегментного індикатора.

3.5.12 Перетворювачі кодів. Визначення кодів 8-4-2-1, Грея, ін.. Особливості коду Грея, принцип побудови логічної схеми.

3.5.13 Схеми контролю даних, сутність функції і які підходи використовуються для їх реалізації.

3.5.14 Мажоритарні елементи. Визначення і області застосування.

## ЛІТЕРАТУРА

1. Карлашук В.И. Электронная лаборатория на IBM PC. Программа Electronics Workbench и ее применение / В.И. Карлашук - М.: Солон-Р. 2000, 506с.
2. Хернitter М. Multisim 7: Современная система компьютерного моделирования и анализа схем электронных устройств / М. Хернitter - М: «ДМК-пресс», 2003.- 488с.
3. Коштов В.В. Основы прикладной теории цифровых автоматов (учебное пособие) / В.В. Коштов, К.К. Кипиани— М: Наука, 1999. –155 с.
4. Самофалов К.Г., Росанкевич А.М, Валуйский В.Н., Каневский Ю.С., Пиневич М.М. Прикладная теория цифровых автоматов / К.Г Самофалов, А.М Росанкевич., В.Н., Валуйский, др. – К: Вища школа. Головное изд-во, 1987. – 375 с.
5. Савельев А. Я. Прикладная теория цифровых автоматов / А. Я. Савельев. – М. : Высшая школа, 1987. – 272 с.
6. Бабич Н.П. Компьютерная схемотехника. Учебное пособие / Н.П. Бабич, И.А. Жуков - К.: "МК-Пресс", 2004. - 576 с.
7. Угрюмов Е. Цифровая схемотехника / Е. Угрюмов. –СПб: БХВ – Петербург, 2001. – 528с.
8. Буняк А. Електроніка та мікросхемотехніка /А. Буняк – Київ-Тернопіль:СМП «Астон», 2001.– 382 с.
9. Бибило П.Н. Основы языка VHDL. Второе издание/ П.Н. Бибило - М.: СОЛОН-Р, 2002 - 224с.
10. Бибило П.Н. Синтез логических схем с использованием языка VHDL / П.Н. Бибило - М.: СОЛОН-Р, 2002 - 384с.
11. Бибило П.Н. Системы проектирования интегральных схем на основе языка VHDL П.Н. / Бибило - М.: СОЛОН-Пресс, 2005 – 384 с.
12. Афанасьев В.А. Компьютерная догика. Лабораторные работы / В.А. Афанасьев, 2002 [Электронный ресурс]: [www.yroka.net.ru](http://www.yroka.net.ru)