Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 1

з дисципліни: «Тестування та діагностика програмно-апаратних засобів»

на тему: «Тестування арифметико-логічного пристрою»

Варіант 25

Виконав:

ст. гр. КІ-303

*Порубайміх О.Є.*

Перевірив:

старший викладач

Хомуляк М.О.

Львів – 2024

Зміст

[Зміст 2](#__RefHeading___Toc55759_1171611487)

[Перелік рисунків 2](#__RefHeading___Toc55761_1171611487)

[РОЗДІЛ 1. Мета 3](#__RefHeading___Toc4821_1098515561)

[РОЗДІЛ 2. Теоретичні відомості 4](#__RefHeading___Toc4823_1098515561)

[РОЗДІЛ 3. Завдання 6](#__RefHeading___Toc4825_1098515561)

[РОЗДІЛ 4. Хід роботи 7](#__RefHeading___Toc4827_1098515561)

[1. Створення проєкту та файлів модулів. 7](#__RefHeading___Toc4829_1098515561)

[2. Еталонний вузол. 9](#__RefHeading___Toc4831_1098515561)

[3. Вузол з помилкою. 10](#__RefHeading___Toc55763_1171611487)

[4. Вузол порівняння. 10](#__RefHeading___Toc55765_1171611487)

[5. Створення схеми. 11](#__RefHeading___Toc4833_1098515561)

[6. Створення генератору тестових послідовностей. 12](#__RefHeading___Toc4835_1098515561)

[7. Виявлення помилок 13](#__RefHeading___Toc4837_1098515561)

[Висновок 15](#__RefHeading___Toc4839_1098515561)

[Список використаних джерел 16](#__RefHeading___Toc4841_1098515561)

[Додаток А. Код вузла ALU. 17](#__RefHeading___Toc4843_1098515561)

[Додаток Б. Код вузла ALU\_Err. 18](#__RefHeading___Toc55767_1171611487)

[Додаток В. Код вузла CMP. 19](#__RefHeading___Toc55769_1171611487)

[Додаток Г. КОД генератора тестових послідовностей. 20](#__RefHeading___Toc55771_1171611487)

Перелік рисунків

[Рис. 4.1. Створення проєкту. 8](#%2525D0%2525A0%2525D0%2525B8%2525D1%2525)

[Рис. 4.2. Створення файлу схеми. 8](#%2525D0%2525A0%2525D0%2525B8%2525D1%2521)

[Рис. 4.3. Створення файлу коду мовою VHDL для опису справного вузла. 9](#%2525D0%2525A0%2525D0%2525B8%2525D1%2522)

[Рис. 4.4. Створення VHDL файлу для опису вузла із помилкою. 9](#%2525D0%2525A0%2525D0%2525B8%2525D1%2523)

[Рис. 4.5. Створення VHDL файлу для опису модулю порівняння. 10](#%2525D0%2525A0%2525D0%2525B8%2525D1%2524)

[Рис. 4.6. Опис еталонного АЛП 10](#%2525D0%2525A0%2525D0%2525B8%2525D1%2526)

[Рис. 4.7. Опис АЛП з помилкою 11](#%2525D0%2525A0%2525D0%2525B8%2525D1%2527)

[Рис. 4.8. Вузол порівняння 12](#%2525D0%2525A0%2525D0%2525B8%2525D1%2528)

[Рис. 4.9. Схема порівняння виходів АЛП 13](#%2525D0%2525A0%2525D0%2525B8%2525D1%2529)

[Рис. 4.10. Створення генератора тестових послідовностей 14](#%2525D0%2525A0%2525D0%2525B8%2525D1%252a)

[Рис. 4.11. Реалізація генератора тестових послідовностей 14](#%2525D0%2525A0%2525D0%2525B8%2525D1%252b)

[Рис. 4.12. Часова діаграма результату симуляції. 15](#%2525D0%2525A0%2525D0%2525B8%2525D1%252c)

1. Мета

Ознайомлення із загальною схемою тестування цифрової техніки. Засвоєння методів та засобів тестування комбінаційних схем на прикладі арифметико-логічного пристрою (АЛП).

1. Теоретичні відомості

Суть тестування полягає у проведенні порівняльного аналізу роботи досліджуваного пристрою з еталонним зразком. Для цього використовується генератор тестових послідовностей (ГТП), який подає тестові дані на обидва пристрої. У випадку виявлення розбіжностей між результатами роботи формується сигнал "Помилка".

Еталон-модель використовується для налагодження цифрових пристроїв. Загальна схема тестування включає генератор еталонних послідовностей (ГЕП) та генератор масочних послідовностей (ГМП), які дозволяють проводити тестування лише за наявності еталонних значень.

Складність генераторів та схеми порівняння може варіюватися від простих кнопок до спеціалізованих комп'ютерів, а сама схема порівняння може бути реалізована як візуальна, так і комп'ютеризована.

Рекомендована послідовність перевірки включає в себе візуальний огляд для виявлення пошкоджень, перевірку на обриви та закорочення, подачу напруги живлення та проведення тестування різних параметрів, таких як напруга живлення, технологічного тренування, частоти генераторів, загальна перевірка за тестами, перевірку шин для з'єднання з зовнішнім середовищем, а також перевірку вузлів в нормальних та граничних умовах.

В рамках тестування цифрових пристроїв використовуються різноманітні види тестових послідовностей, серед яких важливі такі, як біжучий 0, біжуча 1, хвиля 0, хвиля 1 та шаховий код.

* Біжучий 0: Це послідовність, в якій всі біти мають значення "0", але вона поступово змінюється на "1" для перевірки чутливості пристрою до зміни вхідних даних.
* Біжуча 1: У цій послідовності всі біти мають значення "1", але, подібно до біжучого 0, вона поступово переходить до послідовності з усіма бітами "0", для виявлення проблем при зміні вхідних сигналів.
* Хвиля 0: Це послідовність, в якій значення бітів поступово змінюється від "0" до "1", а потім повертається до "0", створюючи хвилястий шаблон. Вона допомагає виявити проблеми зі стабільністю та переходними процесами в пристрої.
* Хвиля 1: Аналогічно хвилі 0, але в цьому випадку значення бітів поступово змінюється від "1" до "0" та назад. Також слугує для виявлення аномалій у переходних процесах.
* Шаховий код: Це спеціальний тип послідовності, в якій біти альтернативно мають значення "0" і "1", утворюючи шахову дошку. Цей шаблон допомагає виявити проблеми зі змінними шаблонами даних, такими як взаємодія між сусідніми бітами та узорами помилок.

Важливими аспектами є врахування індивідуальних особливостей та призначення пристрою, використання спеціалізованих методів для пристроїв з високою швидкістю передачі даних, ведення документації тестування та регулярне оновлення методик тестування та налагодження.

Тестування та налагодження цифрових пристроїв є важливими процесами, які забезпечують коректну роботу пристроїв, допомагають виявити та виправити несправності, гарантують належну якість та надійність продукції, а також підтримують високу якість та конкурентоспроможність продукції.

Текст описує загальні принципи тестування та налагодження, проте детальні методи та інструменти можуть відрізнятися залежно від конкретного типу пристрою.

Ці процеси корисні для розробників та виробників цифрових пристроїв, фахівців з тестування та налагодження, а також для користувачів, які бажають зрозуміти принципи роботи цифрових пристроїв.

1. Завдання

Змоделювати арифметико-логічний пристрій та протестувати його при різних помилках.

Варіант № 25: Логічний зсув ліворуч B.

1. Хід роботи
   1. Створення проєкту та файлів модулів.

Створюю проєкт (Рис. 4.1) та встановлюю його налаштування згідно методичних вказівок. Створюю файл схеми (Рис. 4.2), файл справного модуля (Рис. 4.3), файл модуля з помилкою (Рис. 4.4) та файл модуля порівняння (Рис. 4.5).

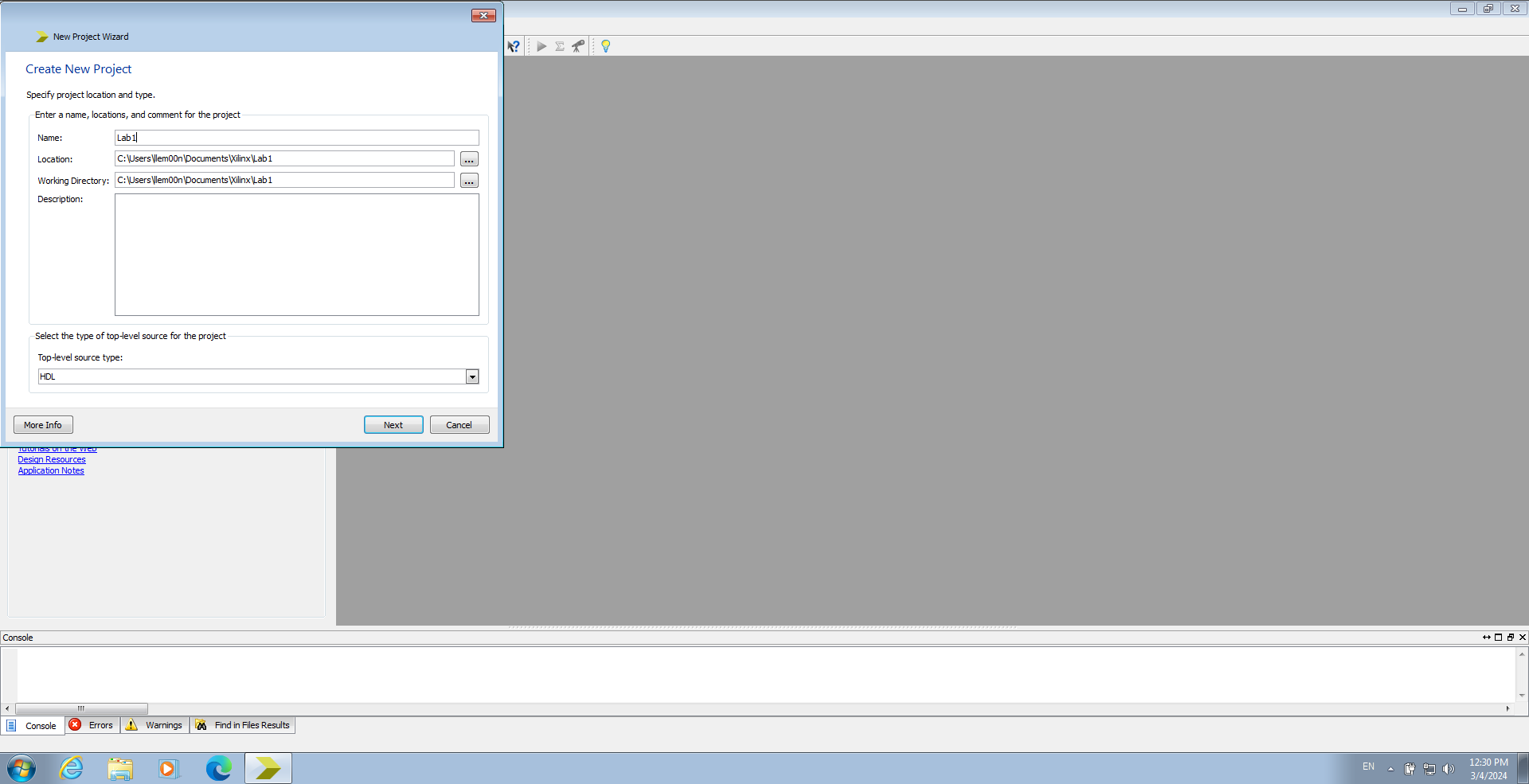


Рис. 1. Створення проєкту.

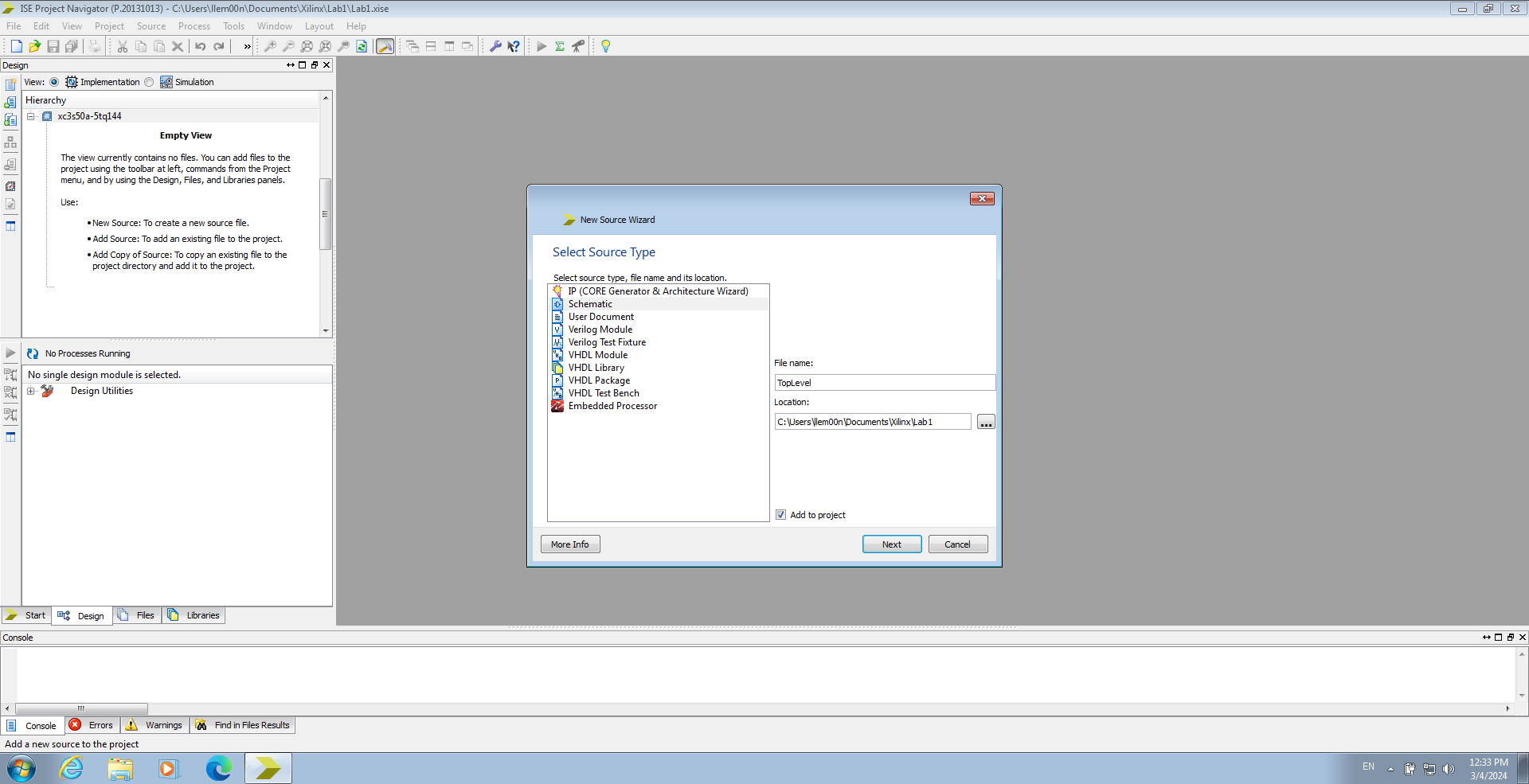


Рис. 2. Створення файлу схеми.

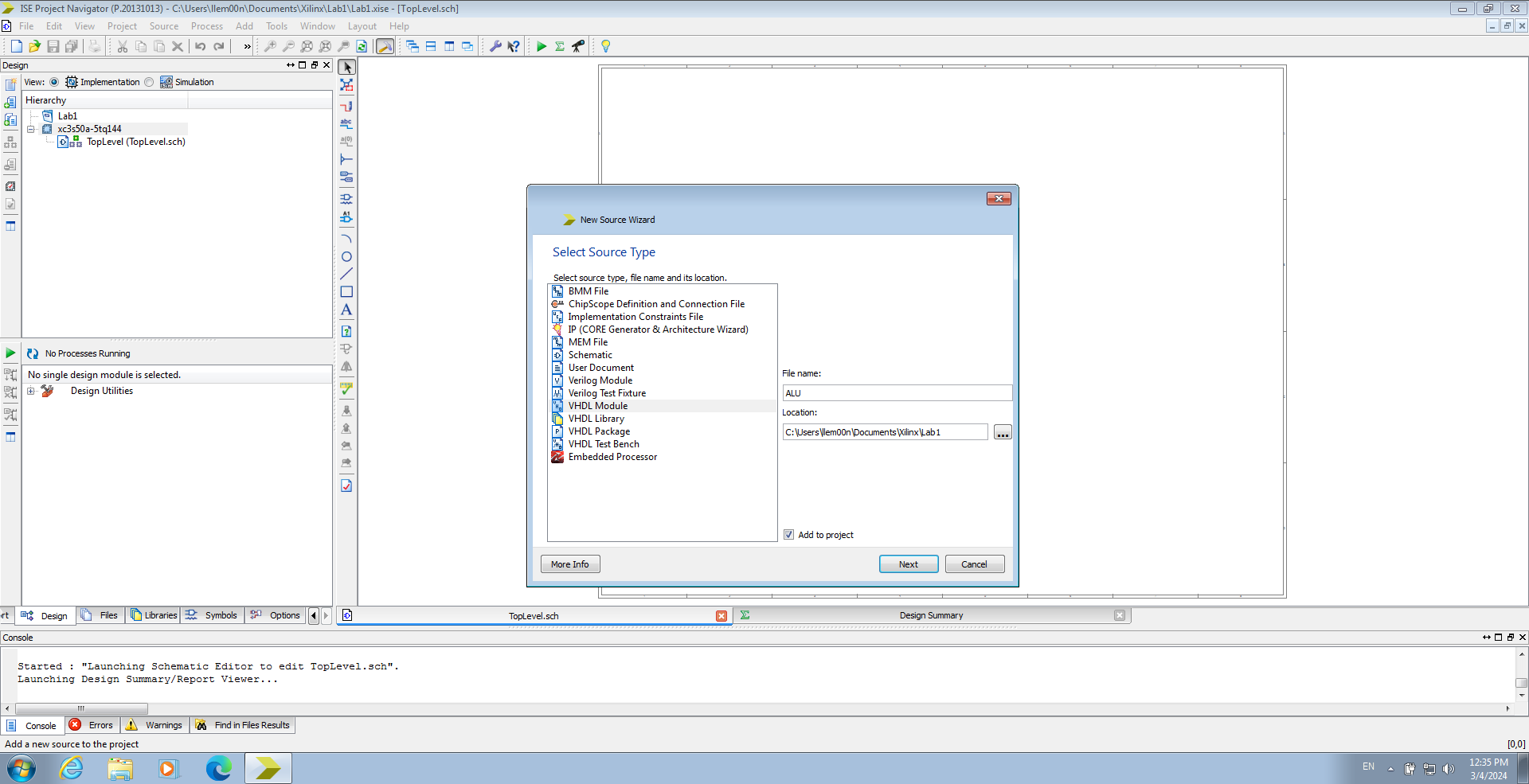


Рис. 3. Створення файлу коду мовою VHDL для опису справного вузла.

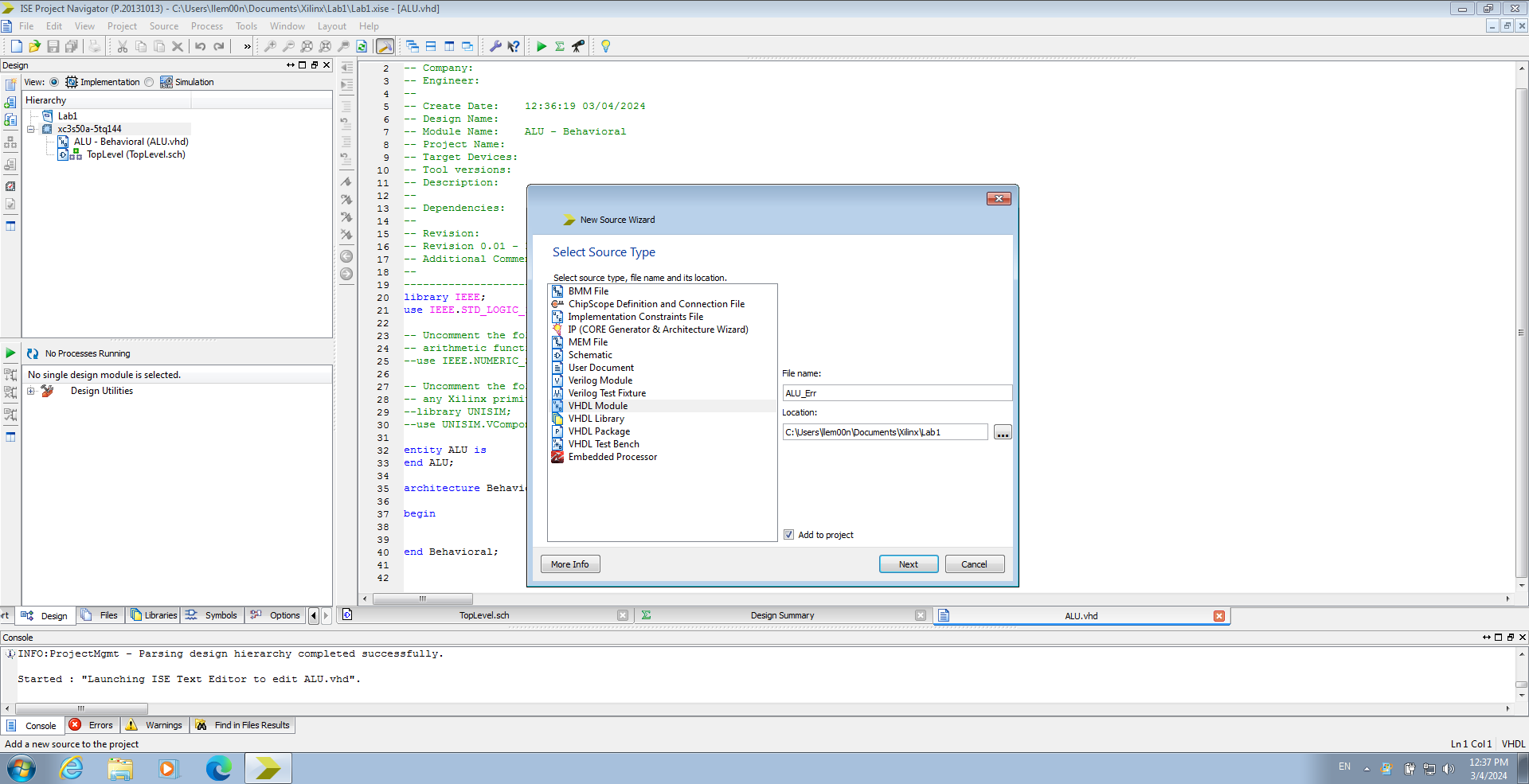


Рис. 4. Створення VHDL файлу для опису вузла із помилкою.

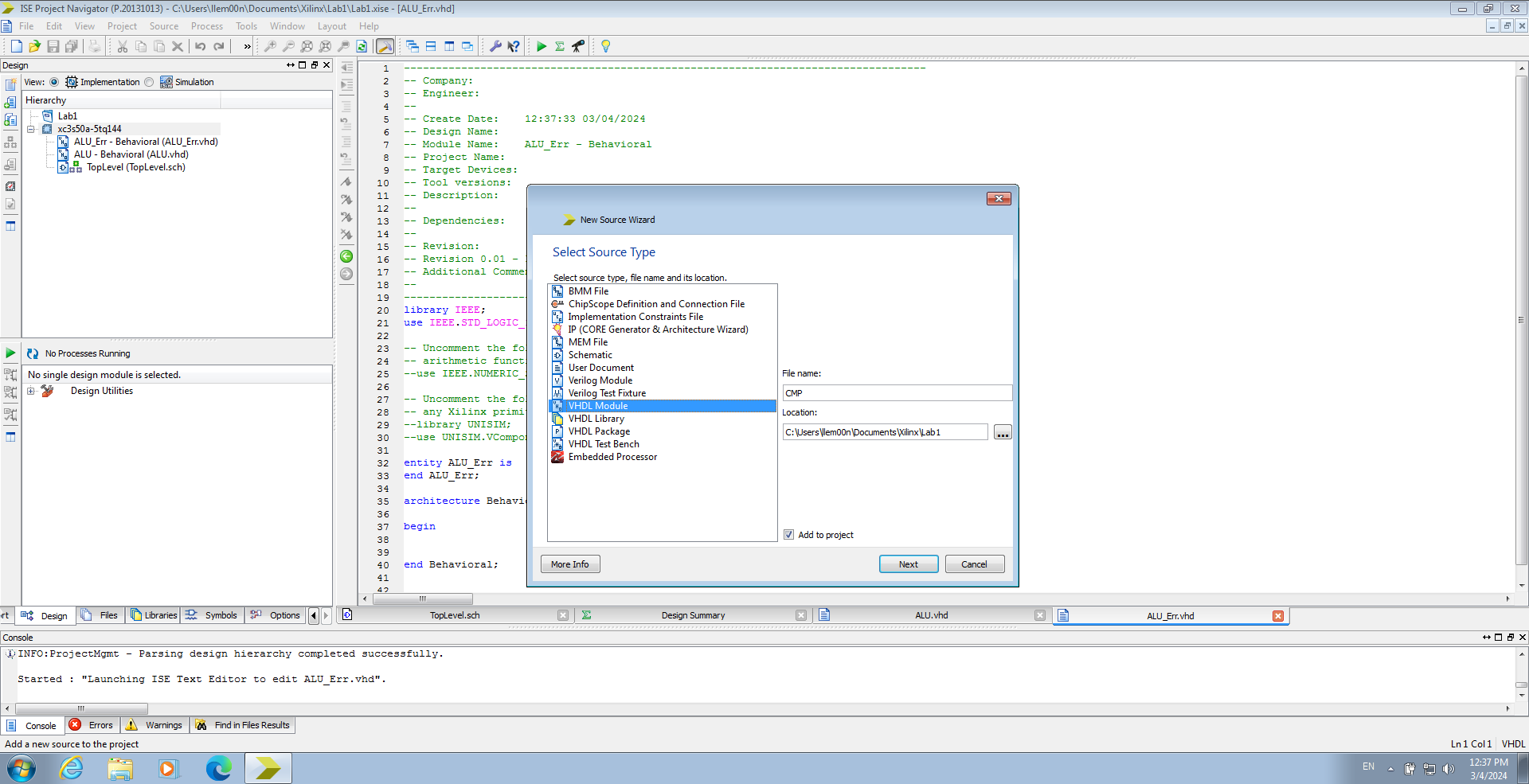


Рис. 5. Створення VHDL файлу для опису модулю порівняння.

* 1. Еталонний вузол.

Опис еталонного АЛП наведено на Рис. 4.5, опис вузла ALU містить ДОДАТОК А. Справний АЛП, при надходженні вхідного сингналу на вхід Input, робить побітовий зсув ліворуч і видає результат на вихід Output.

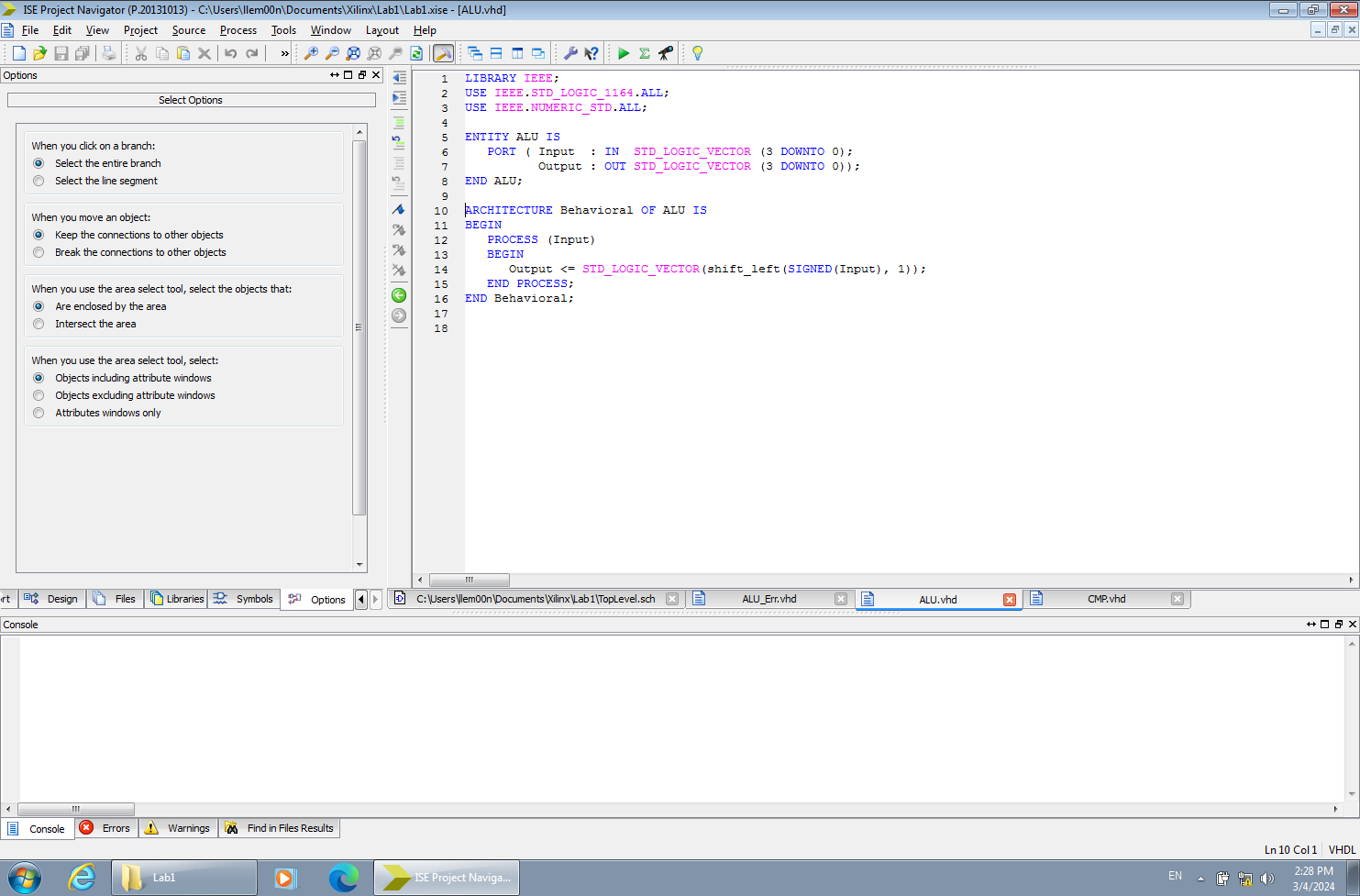


Рис. 6. Опис еталонного АЛП.

* 1. Вузол з помилкою.

Опис вузла з помилкою наведено на Рис. 4.7, опис вузла ALU\_Err містить ДОДАТОК Б. Несправний АЛП, при надходженні вхідного сингналу на вхід Input, робить побітовий зсув ліворуч, але симулює закорочення контактів Input(0) та Input(1), і видає результат на вихід Output.

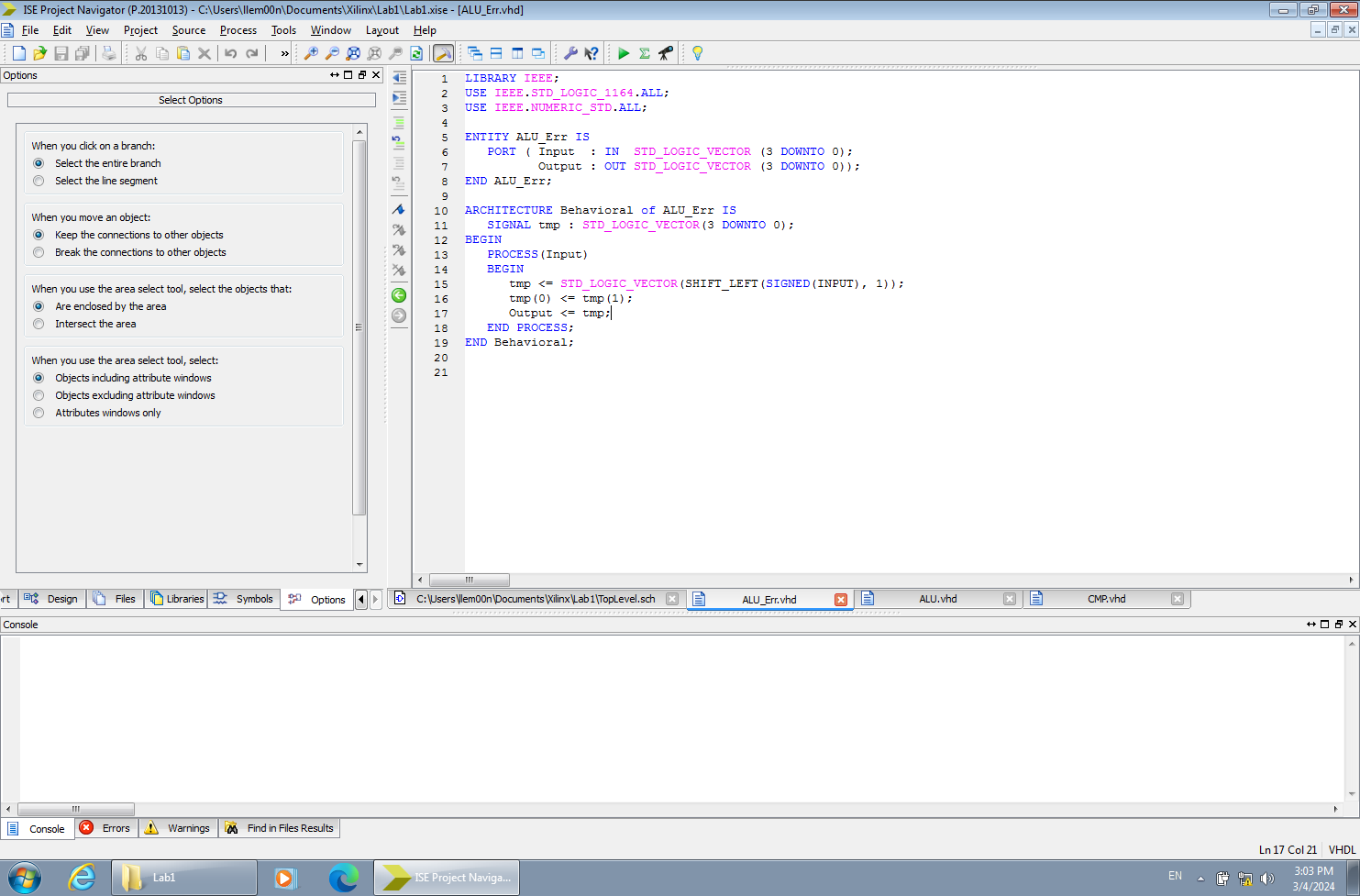


Рис. 7. Опис АЛП з помилкою.

* 1. Вузол порівняння.

Опис вузла порівняння наведено на Рис. 4.8, опис вузла CMP містить ДОДАТОК В. При надходженні сигналів на входи InpA та InpB, модуль порівняння робить перевірку на рівність даних сигналів. Якщо значення сигналів співпадає, на виході Err буде 0, інакше – 1.

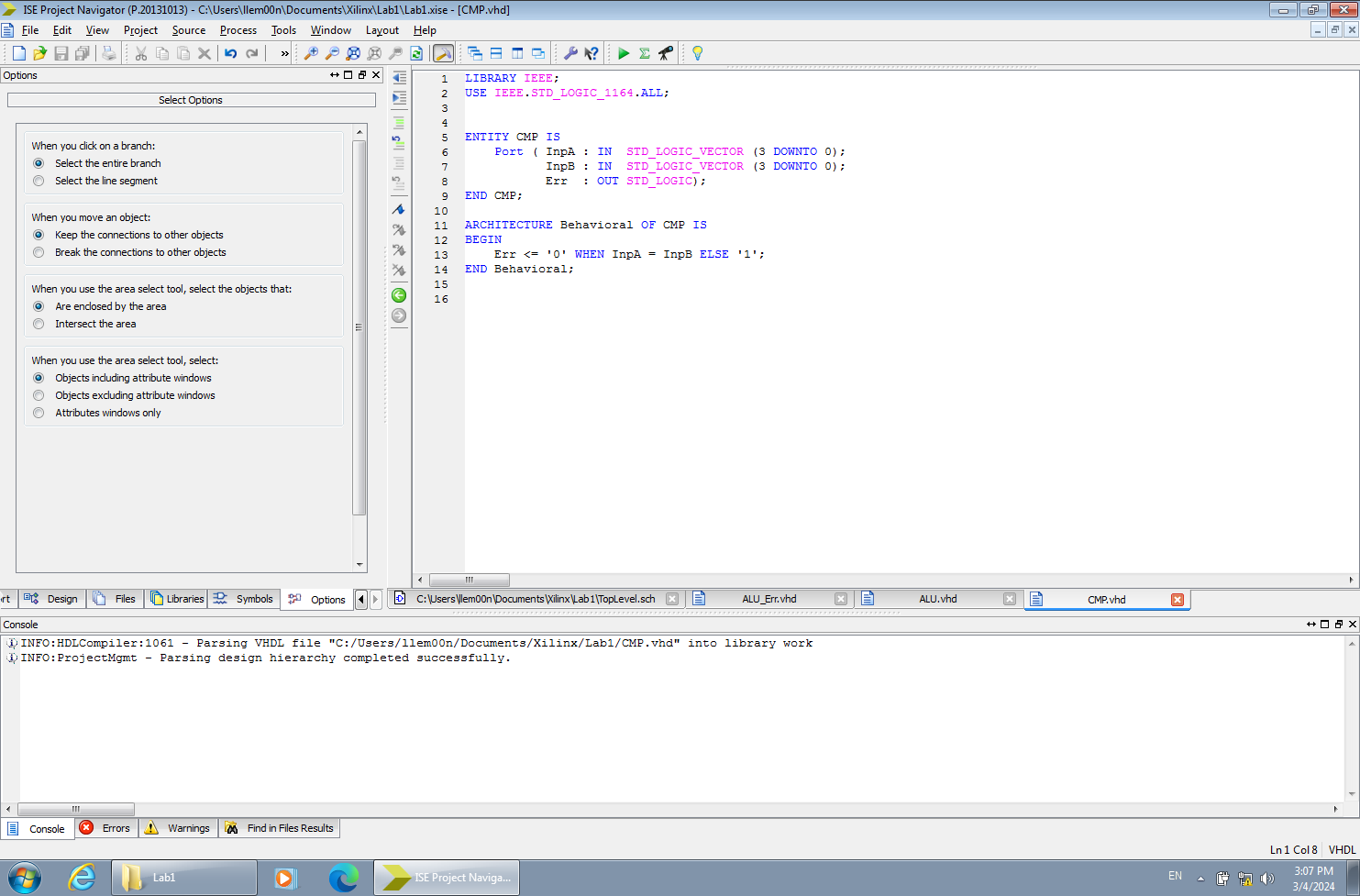


Рис. 8. Вузол порівняння.

* 1. Створення схеми.

Реалізацію схеми порівняння справного АЛП з АЛП з помилкою за допомогою вузла порівняння наведено на Рис. 4.9. Вхід схеми Input під’єднано до входів еталонного та несправного АЛП. Вихід справного АЛП позначено маркером ALU\_Output. Вихід несправного АЛП позначеного маркером ALU\_Err\_Output. Виходи справного та несправного АЛП під’єднано до входів InpA та InpB вузла порівняння відповідно. Результат порівняння позначено вихідним маркером CMP\_Output.

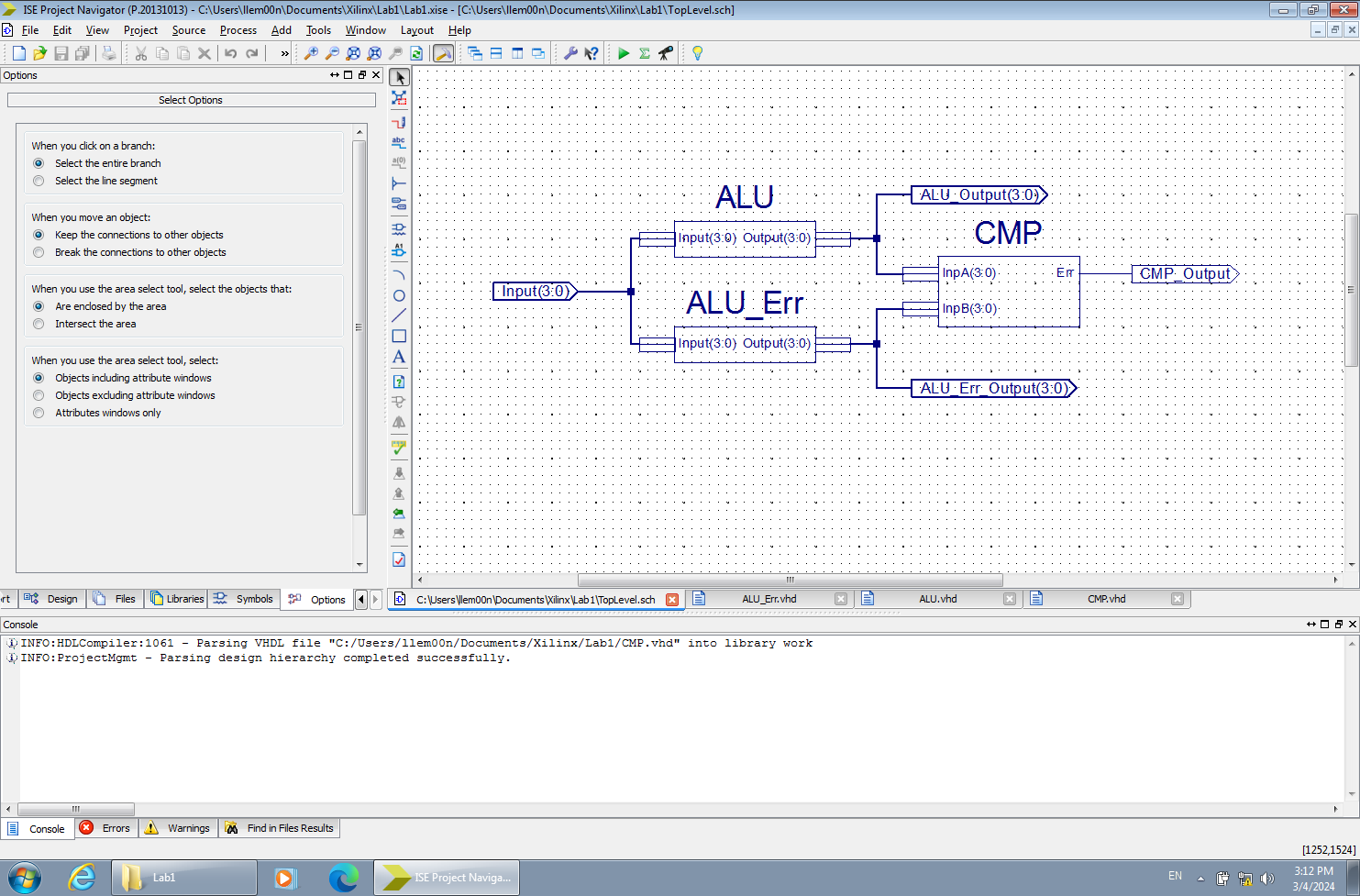


Рис. 9. Схема порівняння виходів АЛП.

* 1. Створення генератору тестових послідовностей.

В генераторі послідовності я реалізував наступні тестові послідовності: біжуча 1, біжучий 0, хвиля 1, хвиля 0, шаховий код. Повний опис генератора послідовності описано в додатку Г.

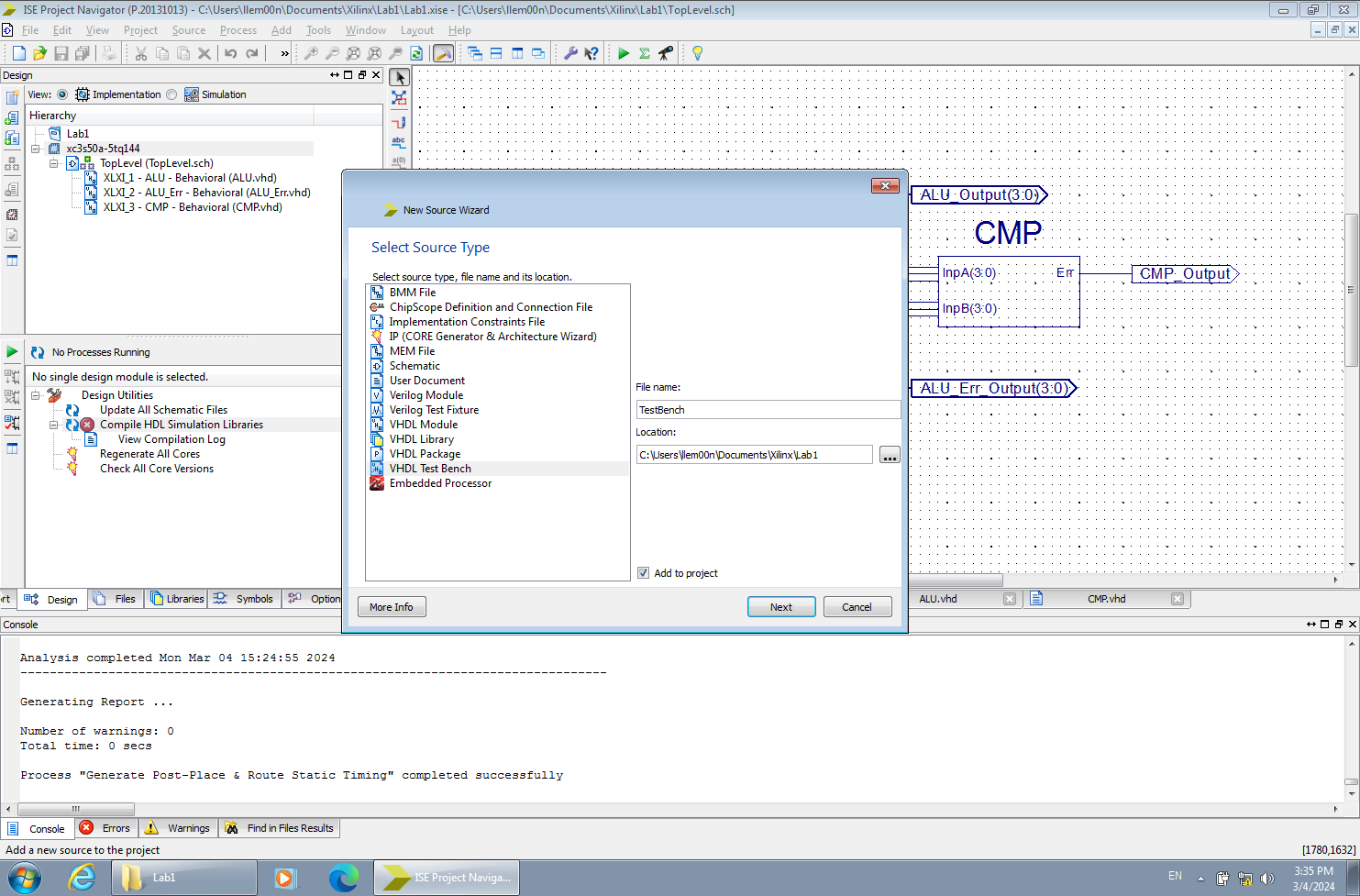


Рис. 10. Створення генератора тестових послідовностей.

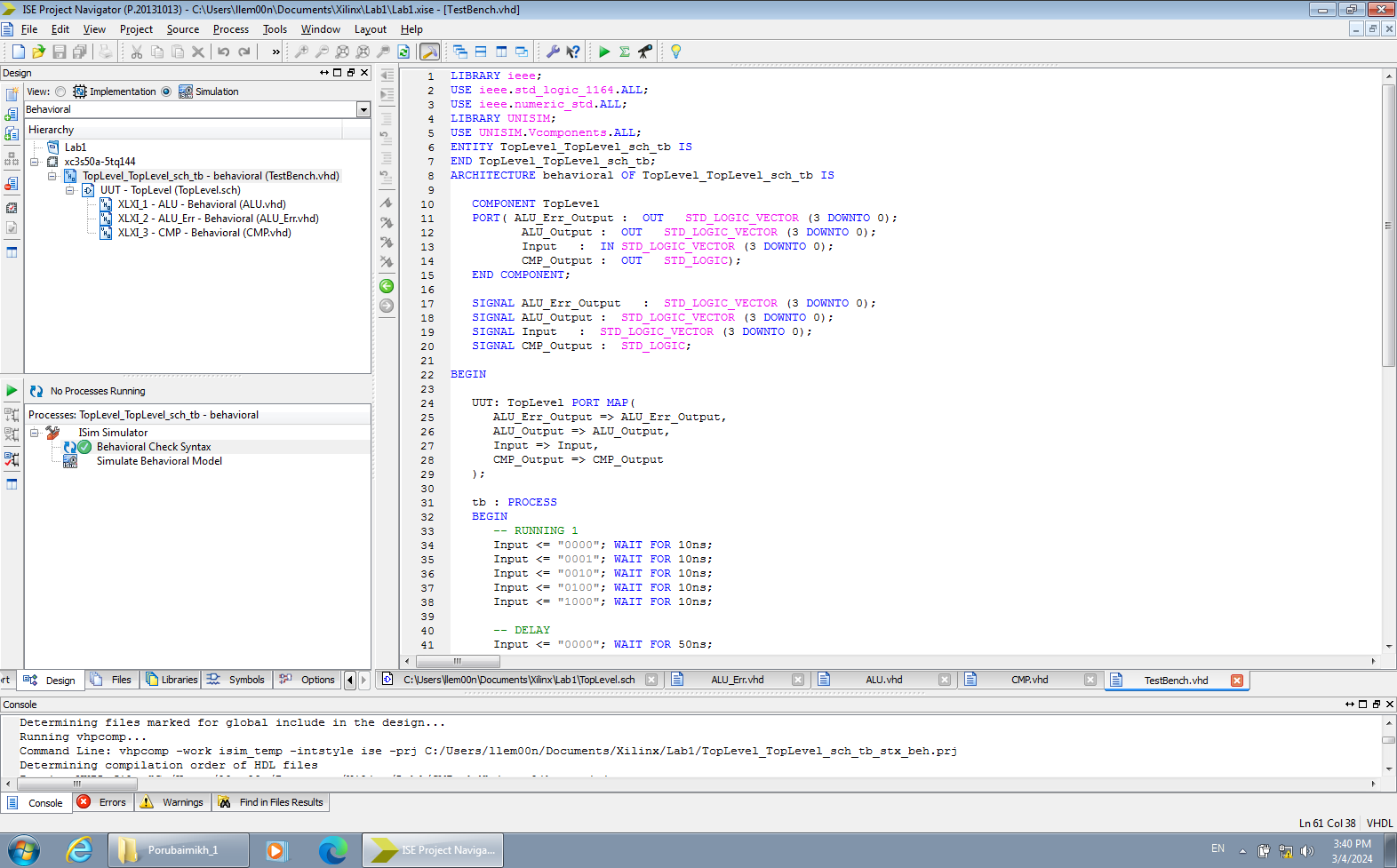


Рис. 11. Реалізація генератора тестових послідовностей.

* 1. Виявлення помилок

Під час симуляції було виявлено помилку при певних вхідних даних.

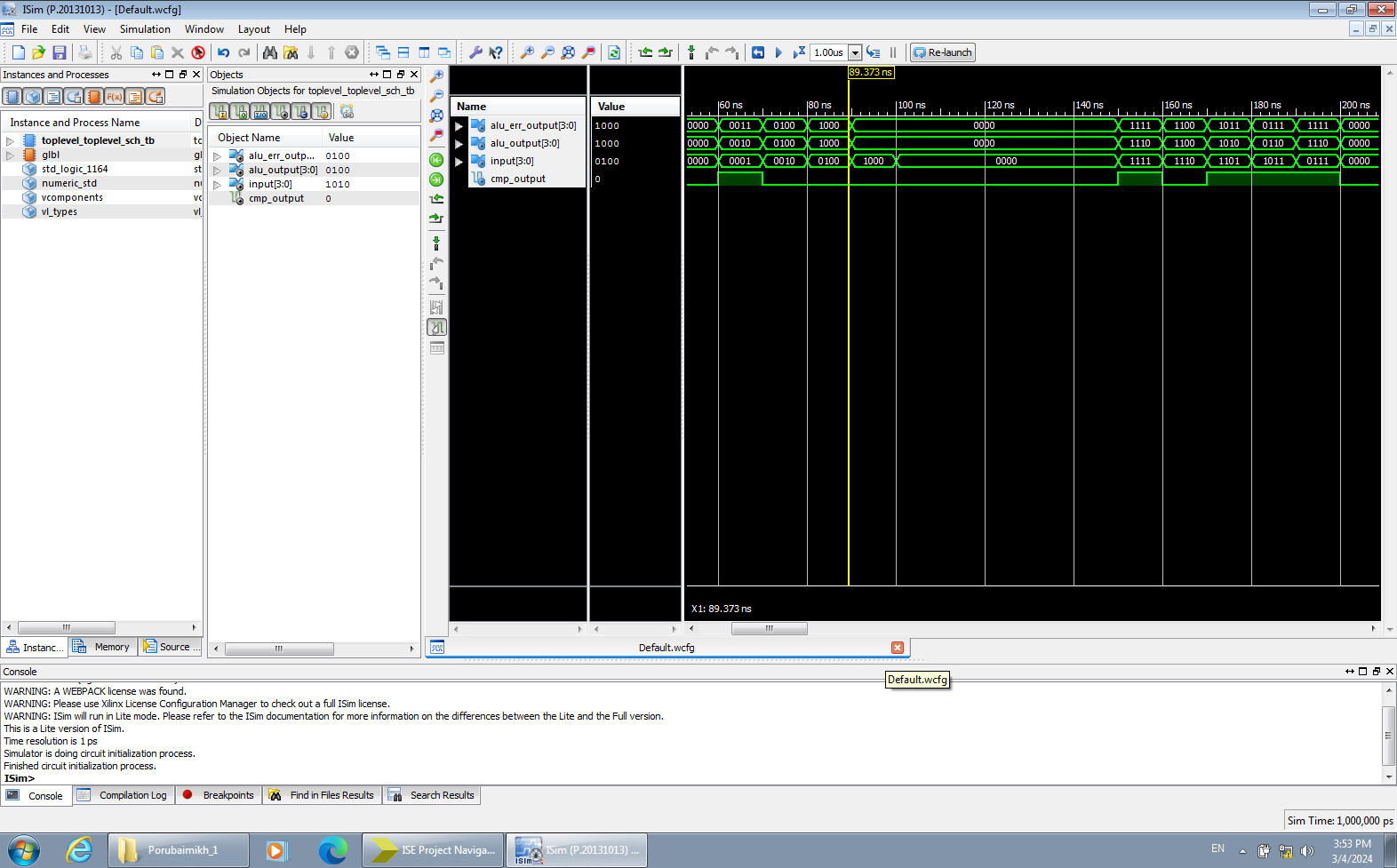


Рис. 12. Часова діаграма результату симуляції.

Висновок

Я ознайомився із загальною схемою тестування цифрової техніки. Засвоїв методи та засоби тестування комбінаційних схем на прикладі арифметично-логічного пристрою. Використавши тестові послідовності біжуча 1, біжучий 0, хвиля 1, хвиля 0 і шаховий код, знайшов помилку в модулі ALU\_Err.

Список використаних джерел

1. Тестування і діагностика програмно-апаратних засобів : лабораторний практикум для студентів спеціальності 123 “Комп’ютерна інженерія” / В. С. Глухов, М. О. Хомуляк, Г. В. Бойко, І. М. Жолубак. – Львів : Видавництво Національного університету “Львівська політехніка”, 2021. – 120 с
2. ISE In-Depth Tutorial <https://docs.xilinx.com/v/u/en-US/ise_tutorial_ug695> 25.10.2023
3. Код вузла ALU.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

ENTITY ALU IS

PORT ( Input : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Output : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0));

END ALU;

ARCHITECTURE Behavioral OF ALU IS

SIGNAL tmp : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

BEGIN

PROCESS (Input, tmp)

BEGIN

tmp <= STD\_LOGIC\_VECTOR(shift\_left(SIGNED(Input), 1));

Output <= tmp;

END PROCESS;

END Behavioral;

1. Код вузла ALU\_Err.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

ENTITY ALU\_Err IS

PORT ( Input : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Output : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0));

END ALU\_Err;

ARCHITECTURE Behavioral of ALU\_Err IS

SIGNAL tmp : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

BEGIN

PROCESS(Input, tmp)

BEGIN

tmp <= STD\_LOGIC\_VECTOR(SHIFT\_LEFT(SIGNED(INPUT), 1));

tmp(0) <= tmp(1);

Output <= tmp;

END PROCESS;

END Behavioral;

1. Код вузла CMP.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY CMP IS

Port ( InpA : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

InpB : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Err : OUT STD\_LOGIC);

END CMP;

ARCHITECTURE Behavioral OF CMP IS

BEGIN

Err <= '0' WHEN InpA = InpB ELSE '1';

END Behavioral;

1. КОД генератора тестових послідовностей.

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

LIBRARY UNISIM;

USE UNISIM.Vcomponents.ALL;

ENTITY TopLevel\_TopLevel\_sch\_tb IS

END TopLevel\_TopLevel\_sch\_tb;

ARCHITECTURE behavioral OF TopLevel\_TopLevel\_sch\_tb IS

COMPONENT TopLevel

PORT( ALU\_Err\_Output : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0);

ALU\_Output : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Input : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

CMP\_Output : OUT STD\_LOGIC);

END COMPONENT;

SIGNAL ALU\_Err\_Output : STD\_LOGIC\_VECTOR (3 DOWNTO 0);

SIGNAL ALU\_Output : STD\_LOGIC\_VECTOR (3 DOWNTO 0);

SIGNAL Input : STD\_LOGIC\_VECTOR (3 DOWNTO 0);

SIGNAL CMP\_Output : STD\_LOGIC;

BEGIN

UUT: TopLevel PORT MAP(

ALU\_Err\_Output => ALU\_Err\_Output,

ALU\_Output => ALU\_Output,

Input => Input,

CMP\_Output => CMP\_Output

);

tb : PROCESS

BEGIN

-- DELAY

Input <= "0000"; WAIT FOR 50ns;

-- RUNNING 1

Input <= "0000"; WAIT FOR 10ns;

Input <= "0001"; WAIT FOR 10ns;

Input <= "0010"; WAIT FOR 10ns;

Input <= "0100"; WAIT FOR 10ns;

Input <= "1000"; WAIT FOR 10ns;

-- DELAY

Input <= "0000"; WAIT FOR 50ns;

-- RUNNING 0

Input <= "1111"; WAIT FOR 10ns;

Input <= "1110"; WAIT FOR 10ns;

Input <= "1101"; WAIT FOR 10ns;

Input <= "1011"; WAIT FOR 10ns;

Input <= "0111"; WAIT FOR 10ns;

-- DELAY

Input <= "0000"; WAIT FOR 50ns;

-- WAVE 1

Input <= "0000"; WAIT FOR 10ns;

Input <= "0001"; WAIT FOR 10ns;

Input <= "0011"; WAIT FOR 10ns;

Input <= "0111"; WAIT FOR 10ns;

Input <= "1111"; WAIT FOR 10ns;

-- DELAY

Input <= "0000"; WAIT FOR 50ns;

-- WAVE 2

Input <= "1111"; WAIT FOR 10ns;

Input <= "1110"; WAIT FOR 10ns;

Input <= "1100"; WAIT FOR 10ns;

Input <= "1000"; WAIT FOR 10ns;

Input <= "0000"; WAIT FOR 10ns;

-- DELAY

Input <= "0000"; WAIT FOR 50ns;

-- CHESS BOARD

Input <= "0101"; WAIT FOR 10ns;

Input <= "1010"; WAIT FOR 10ns;

Input <= "0101"; WAIT FOR 10ns;

Input <= "1010"; WAIT FOR 10ns;

-- WAIT FOREVER

WAIT;

END PROCESS;

END;