Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 3

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

на тему: «Тестування цифрового автомату»

Варіант 3

Виконав:

ст. гр. КІ-303

Порубайміх О.Є.

Перевірив:

Ст.викл. Хомуляк М.О.

Львів – 2024

Зміст

[Зміст 2](#__RefHeading___Toc55759_1171611487)

[Перелік рисунків 2](#__RefHeading___Toc55761_1171611487)

[РОЗДІЛ 1. Мета 4](#__RefHeading___Toc14349_1995045277)

[РОЗДІЛ 2. Теоретичні відомості 5](#__RefHeading___Toc14355_1995045277)

[РОЗДІЛ 3. Індивідуальне завдання 7](#__RefHeading___Toc14353_1995045277)

[РОЗДІЛ 4. Виконання завдання 8](#__RefHeading___Toc14351_1995045277)

[4.1. Створення проєкту та файлів модулів. 8](#__RefHeading___Toc14357_1995045277)

[4.2. Створення еталонного вузла. 8](#__RefHeading___Toc14369_1995045277)

[4.3. Вузол ЦА з помилкою. 10](#__RefHeading___Toc14367_1995045277)

[4.4. Вузли порівняння. 11](#__RefHeading___Toc14365_1995045277)

[4.5. Генератор тестових послідовностей. 12](#__RefHeading___Toc14363_1995045277)

[4.6. Схема. 15](#__RefHeading___Toc14361_1995045277)

[4.7. Процес тестування 16](#__RefHeading___Toc14359_1995045277)

[Висновок 18](#__RefHeading___Toc14347_1995045277)

[Список використаних джерел 19](#__RefHeading___Toc14345_1995045277)

[Додаток А. Код вузла DD. 20](#__RefHeading___Toc14331_1995045277)

[Додаток Б. Код вузла DD\_Err. 22](#__RefHeading___Toc14343_1995045277)

[Додаток В. Код вузла CMP. 24](#__RefHeading___Toc14341_1995045277)

[Додаток Г. Код вузла CMP\_S. 25](#__RefHeading___Toc14333_1995045277)

[Додаток Д. Код компонента TSG. 26](#__RefHeading___Toc14339_1995045277)

[Додаток Е. Код архітектури компонента TSG для виконання. 27](#__RefHeading___Toc14337_1995045277)

[Додаток Є. Код архітектури компонента TSG для симуляції. 28](#__RefHeading___Toc14335_1995045277)

Перелік рисунків

[Рис. 4.1. Створення проєкту. 8](#%25D0%25A0%25D0%25B8%25D1%2581._4.!0|seq)

[Рис. 4.2. Граф-схема ЦА. 9](#%25D0%25A0%25D0%25B8%25D1%2581._4.!17|se)

[Рис. 4.3. Створення файлу DD.vhd. 9](#%25D0%25A0%25D0%25B8%25D1%2581._4.!1|seq)

[Рис. 4.4. Опис еталонного ЦА. 10](#%25D0%25A0%25D0%25B8%25D1%2581._4.!2|seq)

[Рис. 4.5. Створення файлу DD\_Err.vhd. 10](#%25D0%25A0%25D0%25B8%25D1%2581._4.!3|seq)

[Рис. 4.6. Опис ЦА з помилкою. 11](#%25D0%25A0%25D0%25B8%25D1%2581._4.!4|seq)

[Рис. 4.7. Створення файлу CMP.vhd. 12](#%25D0%25A0%25D0%25B8%25D1%2581._4.!5|seq)

[Рис. 4.8. Опис взула порівняння. 12](#%25D0%25A0%25D0%25B8%25D1%2581._4.!6|seq)

[Рис. 4.9. Створення файлу TSG.vhd. 13](#%25D0%25A0%25D0%25B8%25D1%2581._4.!7|seq)

[Рис. 4.10. Опис компонента TSG.vhd. 13](#%25D0%25A0%25D0%25B8%25D1%2581._4.!8|seq)

[Рис. 4.11. Створення файлу TSG\_IMP.vhd. 14](#%25D0%25A0%25D0%25B8%25D1%2581._4.!9|seq)

[Рис. 4.12. Опис архітектури TSG для виконання. 14](#%25D0%25A0%25D0%25B8%25D1%2581._4.!10|se)

[Рис. 4.13. Створення файлу TSG\_SIM.vhd. 15](#%25D0%25A0%25D0%25B8%25D1%2581._4.!11|se)

[Рис. 4.14. Опис архітектури TSG для симуляції. 15](#%25D0%25A0%25D0%25B8%25D1%2581._4.!12|se)

[Рис. 4.15. Створення файлу схеми. 16](#%25D0%25A0%25D0%25B8%25D1%2581._4.!13|se)

[Рис. 4.16. Схема. 16](#%25D0%25A0%25D0%25B8%25D1%2581._4.!14|se)

[Рис. 4.17. Часова діаграма тестування шини адреси. 17](#%25D0%25A0%25D0%25B8%25D1%2581._4.!15|se)

1. Мета

Ознайомлення із загальною схемою тестування цифрової техніки. Засвоєння методів та засобів тестування цифрових схем з пам’яттю на прикладі цифрового автомата (ЦА).

1. Теоретичні відомості

У світі сучасних технологій, цифрові автомати виступають як невід'ємна складова більшості цифрових систем, відіграючи ключову роль у процесах автоматизації та обробки інформації. Ці пристрої, засновані на виконанні заздалегідь заданих алгоритмів, залежно від вхідних сигналів і поточного стану, вимагають ретельного тестування для забезпечення їх надійності та ефективності.

Властивості та Класифікація.

Цифрові автомати можуть взаємодіяти з навколишнім середовищем через вхідні та вихідні сигнали, мати внутрішні стани, що відображають їх поточний стан, і початковий стан, з якого розпочинається їх робота. Основні правила роботи автоматів визначають логіку генерації вихідних сигналів і переходів між станами. Залежно від цих характеристик, автомати поділяють на два основних типи: автомати Мура, вихідні сигнали яких залежать лише від поточного стану, та автомати Мілі, де враховуються як поточний стан, так і вхідні сигнали.

Методики Тестування.

Для забезпечення бездоганної роботи цифрових автоматів, існує кілька методик тестування:

1. Табличний метод дозволяє перевірити відповідність між сигналами, станами та виходами за допомогою таблиць станів і переходів.
2. Часові діаграми візуалізують динаміку зміни станів і сигналів.
3. Аналітичний метод включає математичний опис поведінки автомата через мови опису апаратури, такі як VHDL або Verilog.
4. Моделювання та графи станів надають візуальне уявлення про роботу автоматів і переходи між станами.

Окрім традиційних методів, застосовуються і сучасні підходи, такі як формальна верифікація для математичного доведення правильності роботи автоматів і динамічна перевірка, яка використовує тестові середовища для імітації реальних умов експлуатації.

Застосування та Важливість.

Цифрові автомати знаходять своє застосування у різноманітних галузях, включаючи автоматизовані системи управління, телекомунікації, електроніку та комп'ютерну техніку. Важливість тестування в цьому контексті не може бути переоцінена, оскільки воно не лише забезпечує надійність цифрових систем, але й сприяє оптимізації їх роботи згідно з конкретними вимогами.

Висновок.

Тестування цифрових автоматів є критично важливим етапом у процесі їх проектування, який вимагає детального аналізу поведінки автоматів і оптимізації їх роботи. Це не лише підвищує ефективність цифрових систем, але й забезпечує їх безперебійну роботу, що є невід'ємною умовою для сучасного технологічного прогресу.

1. Індивідуальне завдання

Змоделювати діаграму станів та протестувати її при всіх можливих комбінаціях.

Варіант № 3: Закоротка вхідного сигналу на живлення +5В.

1. Виконання завдання
   1. Створення проєкту та файлів модулів.

Створюю проєкт (Рис. 4.1) та встановлюю його налаштування згідно методичних вказівок.

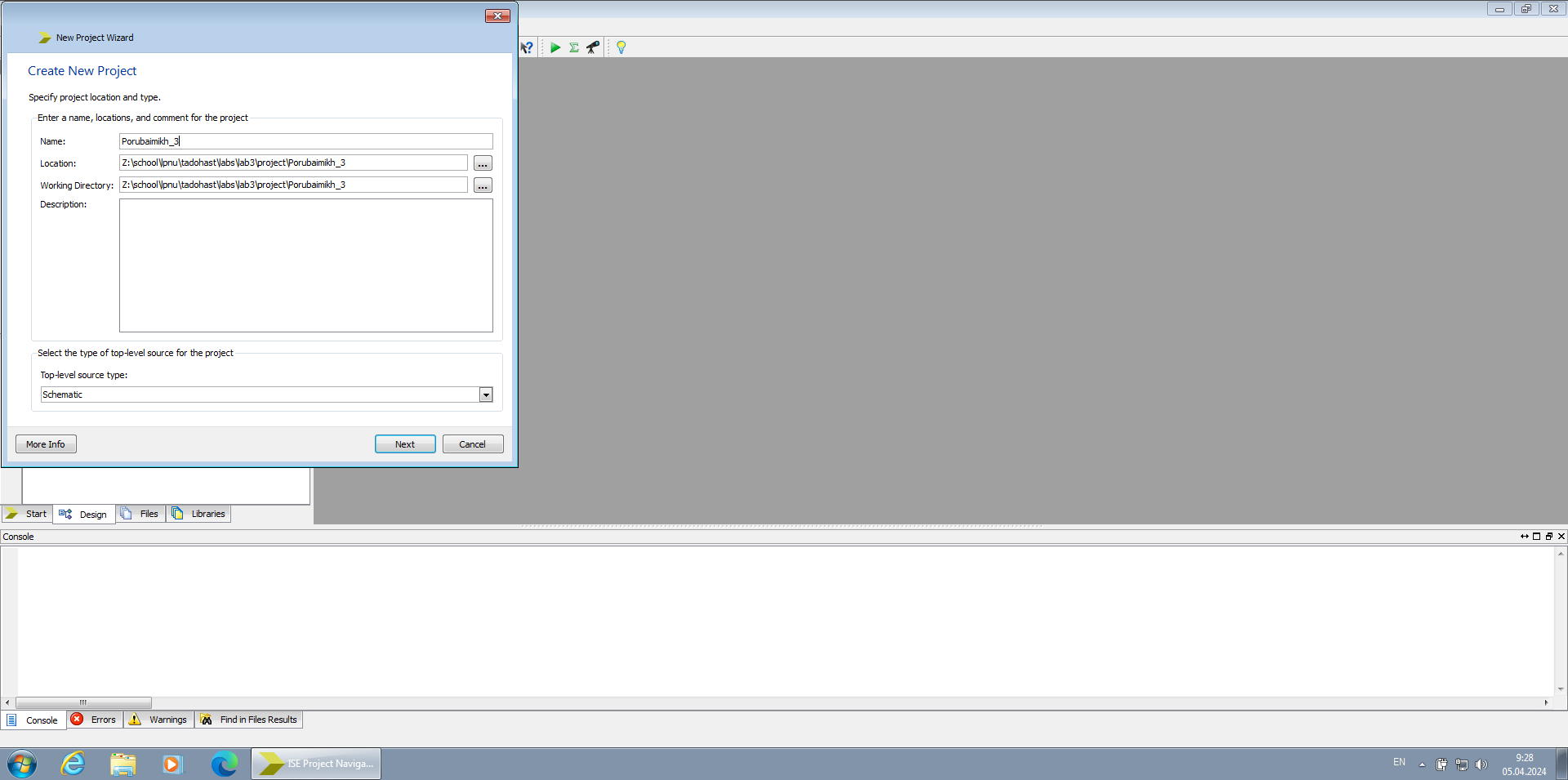


Рис. 4.1. Створення проєкту.

* 1. Створення еталонного вузла.

Процес створення VHDL файлу для опису еталонного ЦА наведено на Рис. 4.3, опис еталонного ЦА наведено на Рис. 4.4, повний опис вузла DD містить ДОДАТОК А. Еталонний ЦА реалізує граф-схему з Рис. 4.2

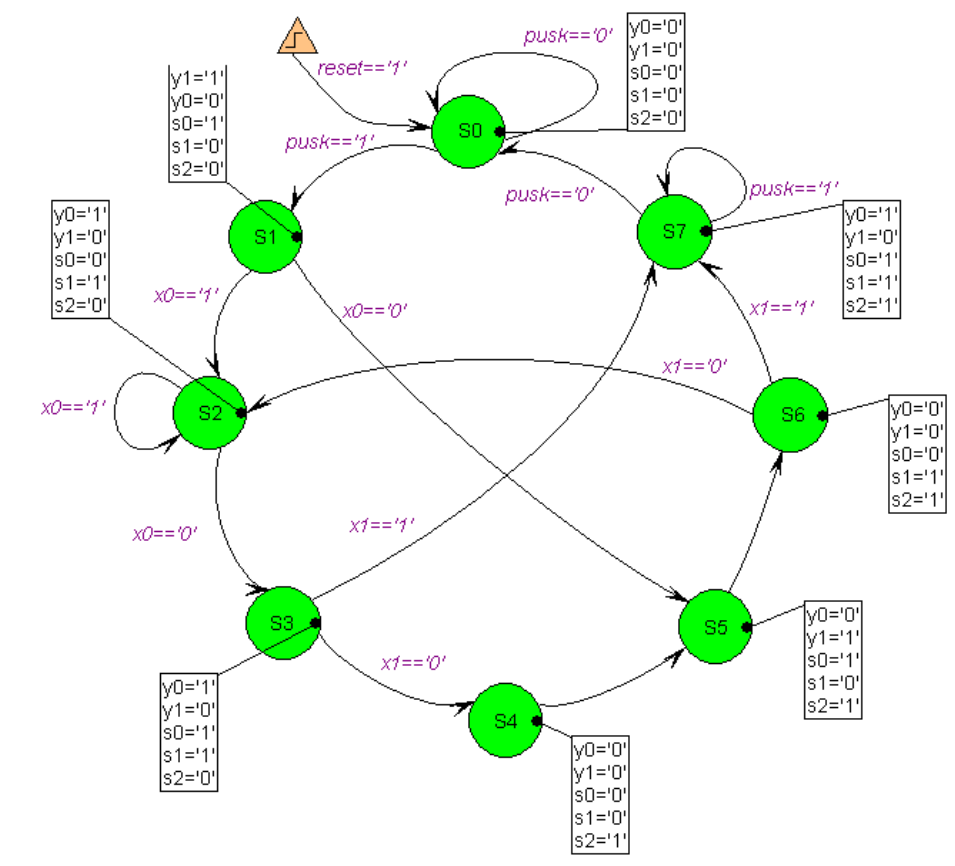


Рис. 4.2. Граф-схема ЦА.

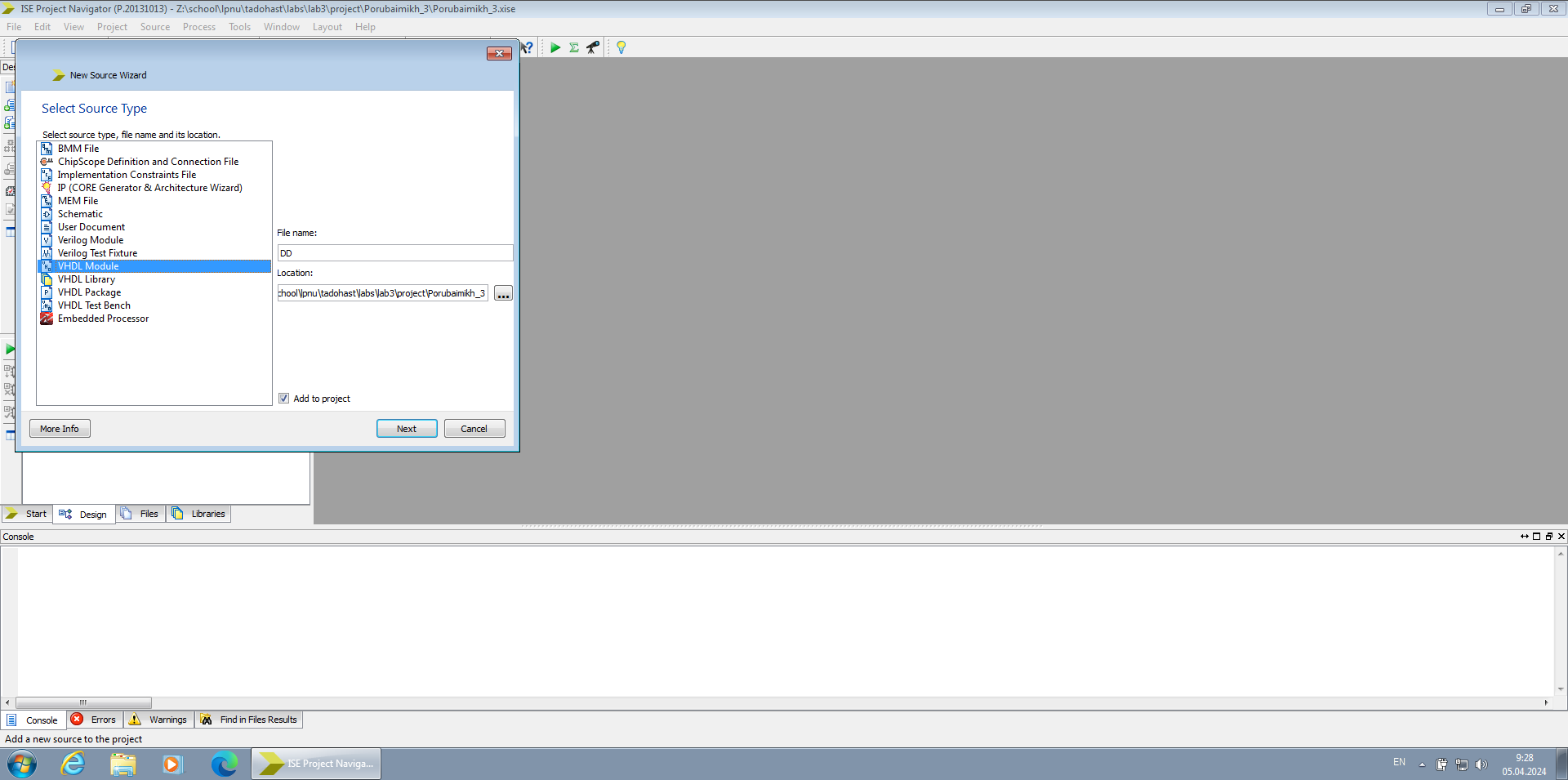


Рис. 4.3. Створення файлу DD.vhd.

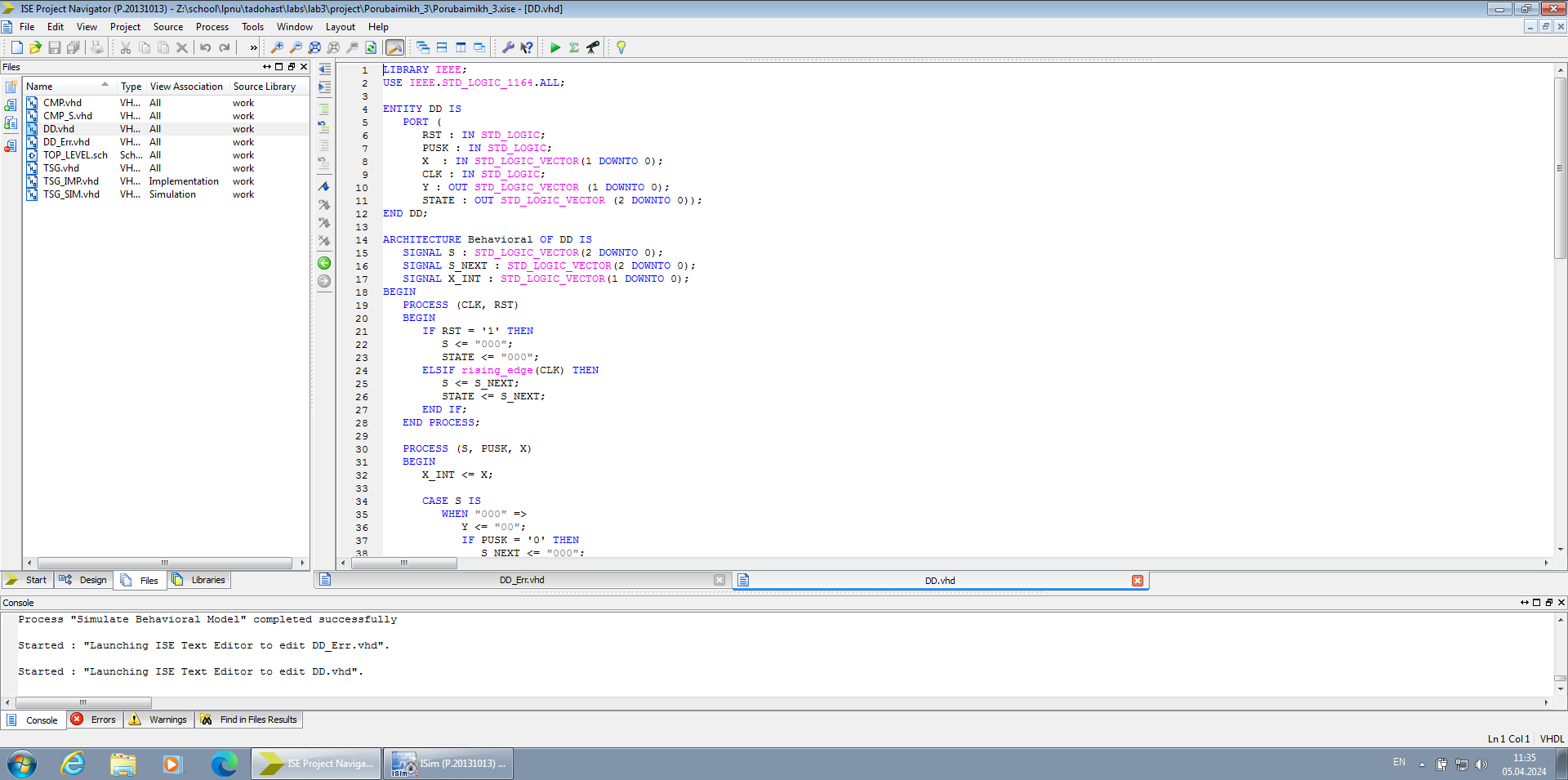


Рис. 4.4. Опис еталонного ЦА.

* 1. Вузол ЦА з помилкою.

Процес створення VHDL файлу для опису вузла ЦА з помилкою наведено на Рис. 4.5. Опис вузла ЦА з помилкою наведено на Рис. 4.6, опис вузла DD\_Err містить ДОДАТОК Б. Несправний ЦА реалізує граф-схему з Рис. 4.2, але має закоротку вхідного сигналу X(0) на живлення +5В.

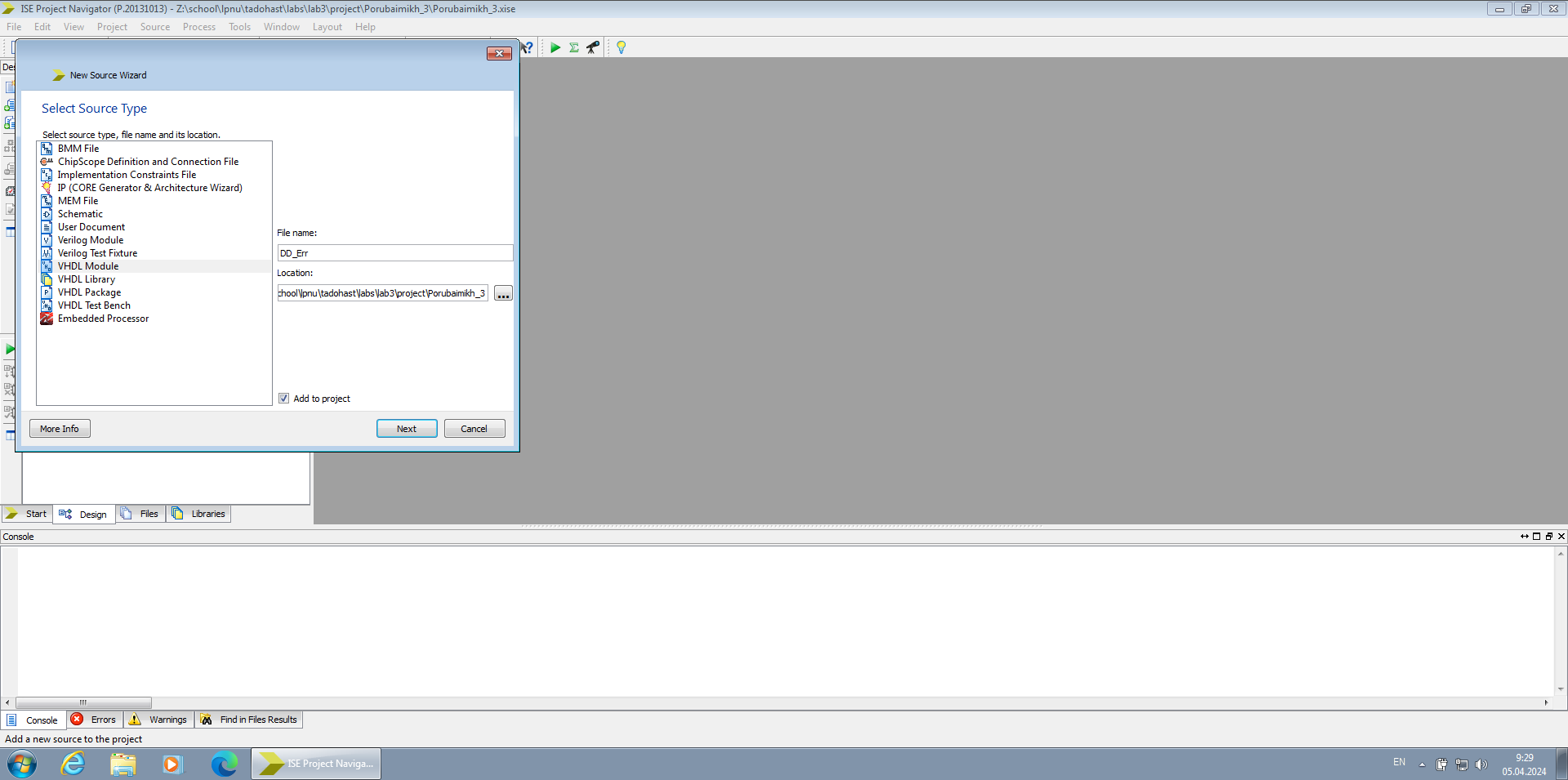


Рис. 4.5. Створення файлу DD\_Err.vhd.

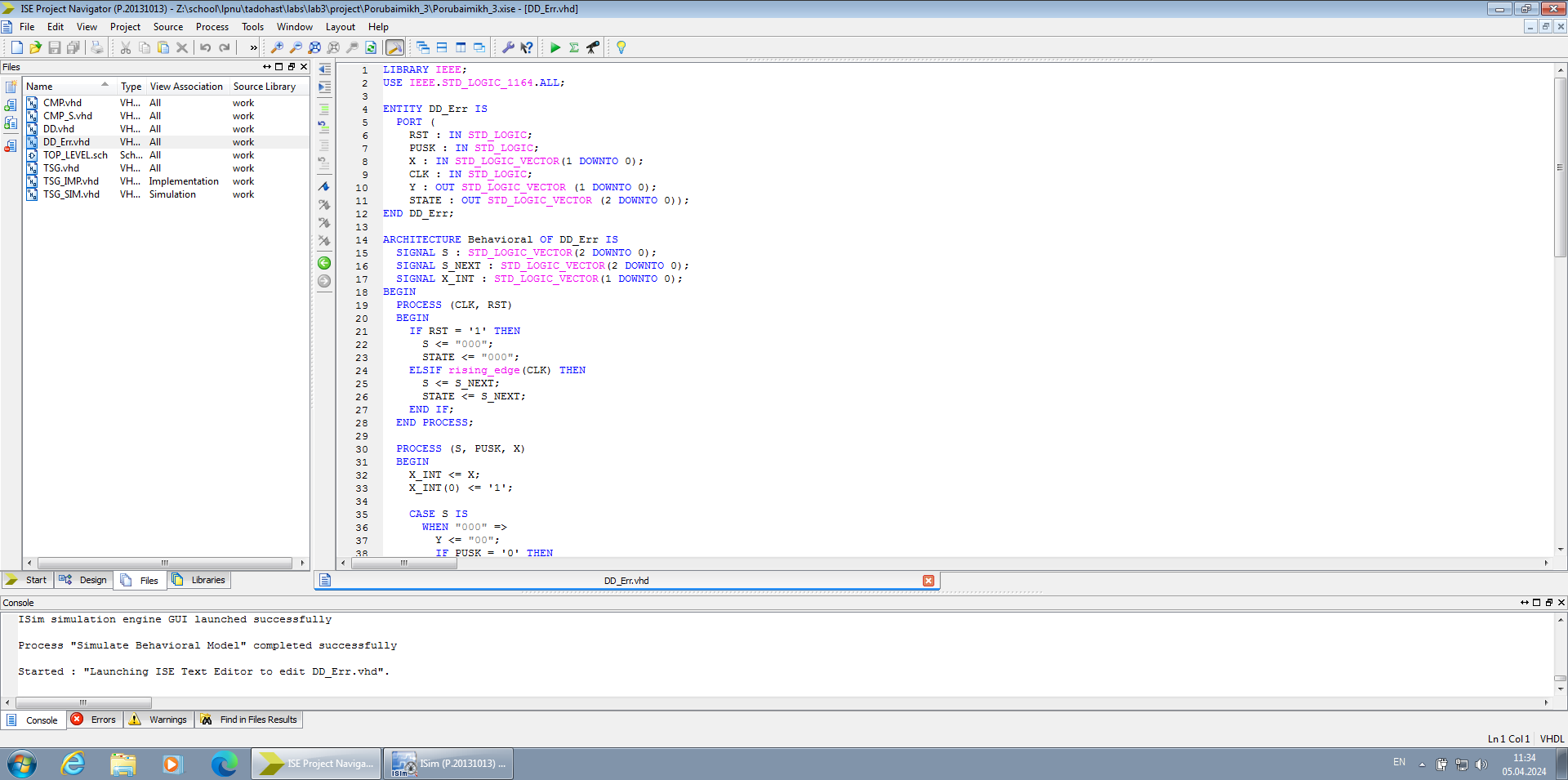


Рис. 4.6. Опис ЦА з помилкою.

* 1. Вузли порівняння.

Процес створення VHDL файлу для опису вузла порівняння виходів Y досліджуваних ЦА наведено на Рис. 4.7. Опис цього вузла порівняння наведено на Рис. 4.8, опис вузла CMP містить ДОДАТОК В. При надходженні сигналів на входи A та B, модуль порівняння робить перевірку на рівність даних сигналів. Якщо значення сигналів співпадає, на виході Error буде 0, інакше – 1.

За таким самим принципом створено вузол порівняння для станів ЦА (CMP\_S). Опис вузла CMP\_S містить додаток Г.

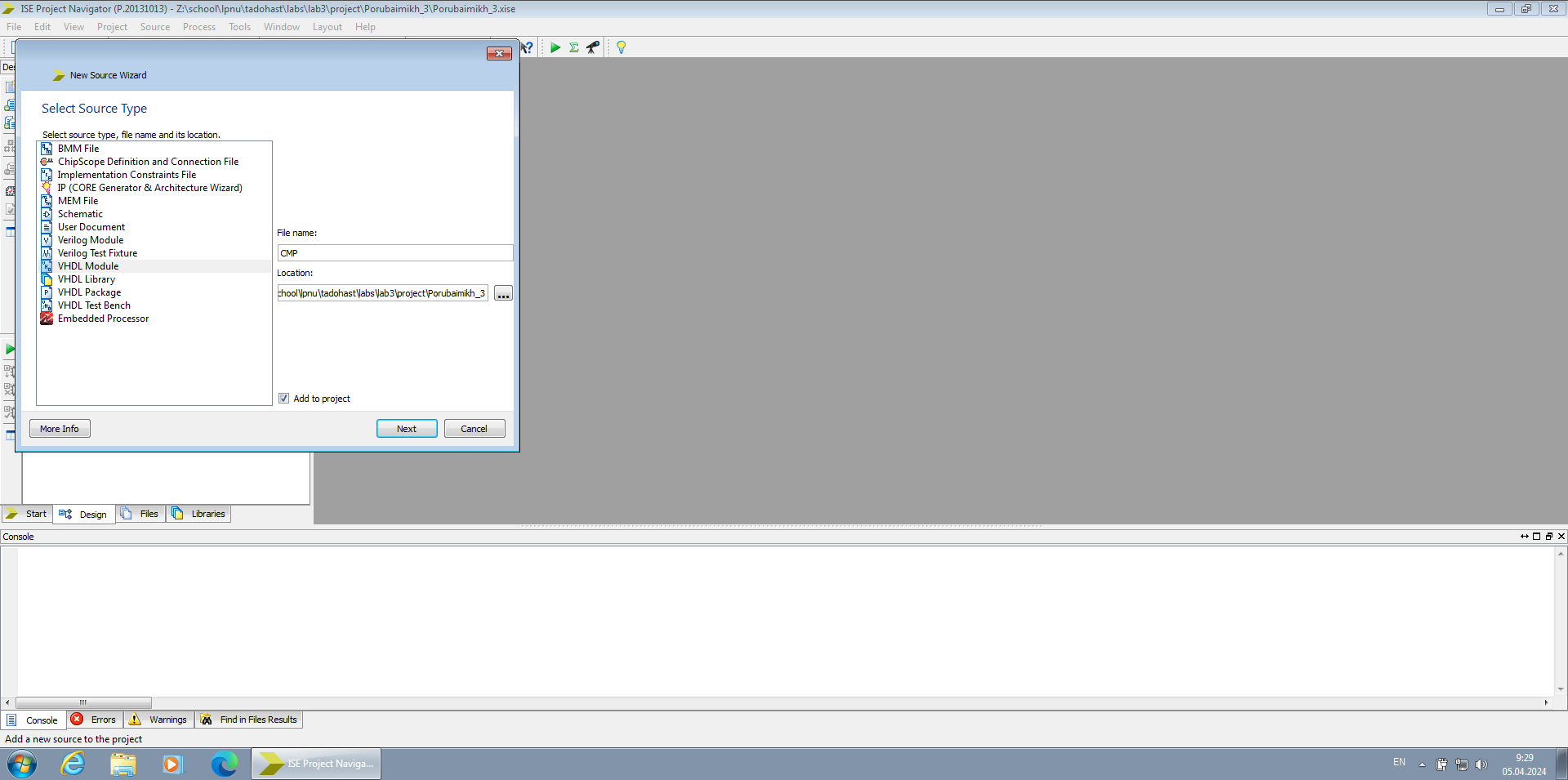


Рис. 4.7. Створення файлу CMP.vhd.

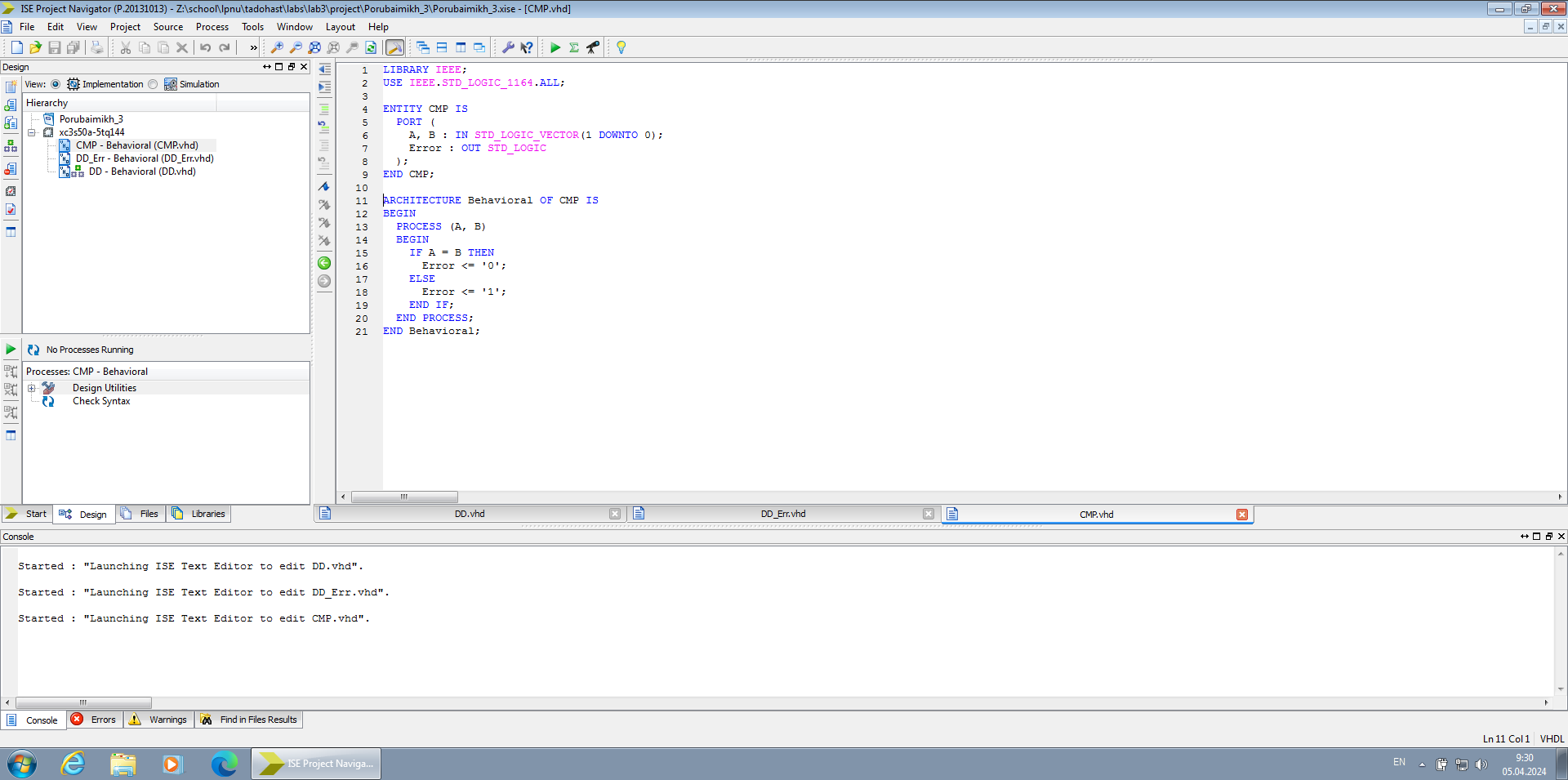


Рис. 4.8. Опис взула порівняння.

* 1. Генератор тестових послідовностей.

Для опису генератора тестових послідовностей, необхідно створити 3 файли:

1. Файл з описом компонента ГТС;
2. Файл з описом архітектури ГТС для виконання;
3. Файл з описом архітектури ГТС для симуляції.

Процес створення файлу з описом компонента ГТС наведено на Рис. 4.9. Опис компонента ГТС наведено на Рис. 4.10. Повний опис компонента ГТС містить ДОДАТОК Д.

Процес створення файлу з описом архітектури ГТС для виконання наведено на Рис. 4.11. Опис архітектури ГТС для виконання наведено на Рис. 4.12. Повний опис архітектури ГТС для виконання містить ДОДАТОК Е.

Процес створення файлу з описом архітектури ГТС для симуляції наведено на Рис. 4.13. Опис архітектури ГТС для симуляції наведено на Рис. 4.14. Повний опис архітектури ГТС для симуляції містить ДОДАТОК Є.

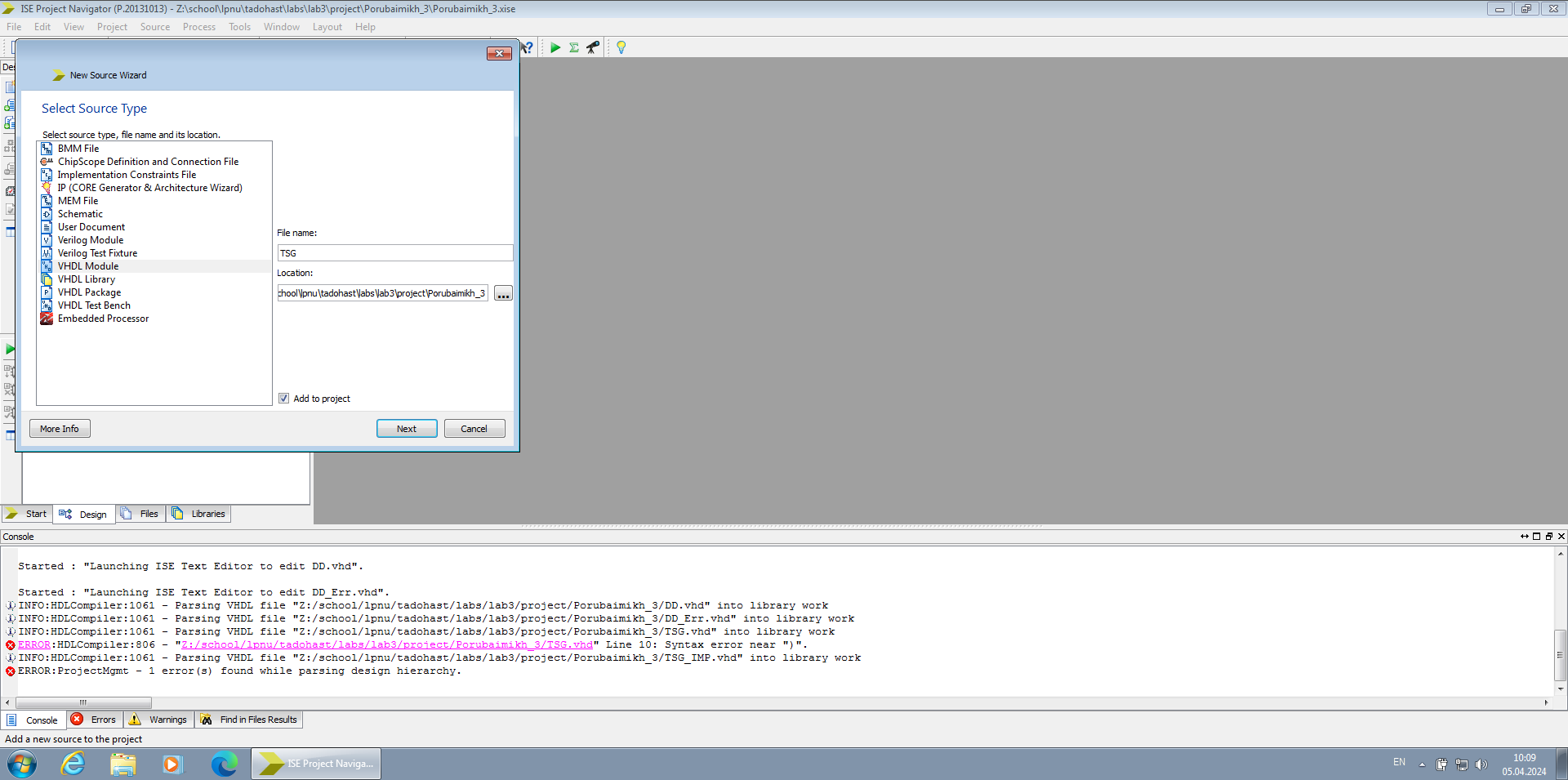


Рис. 4.9. Створення файлу TSG.vhd.

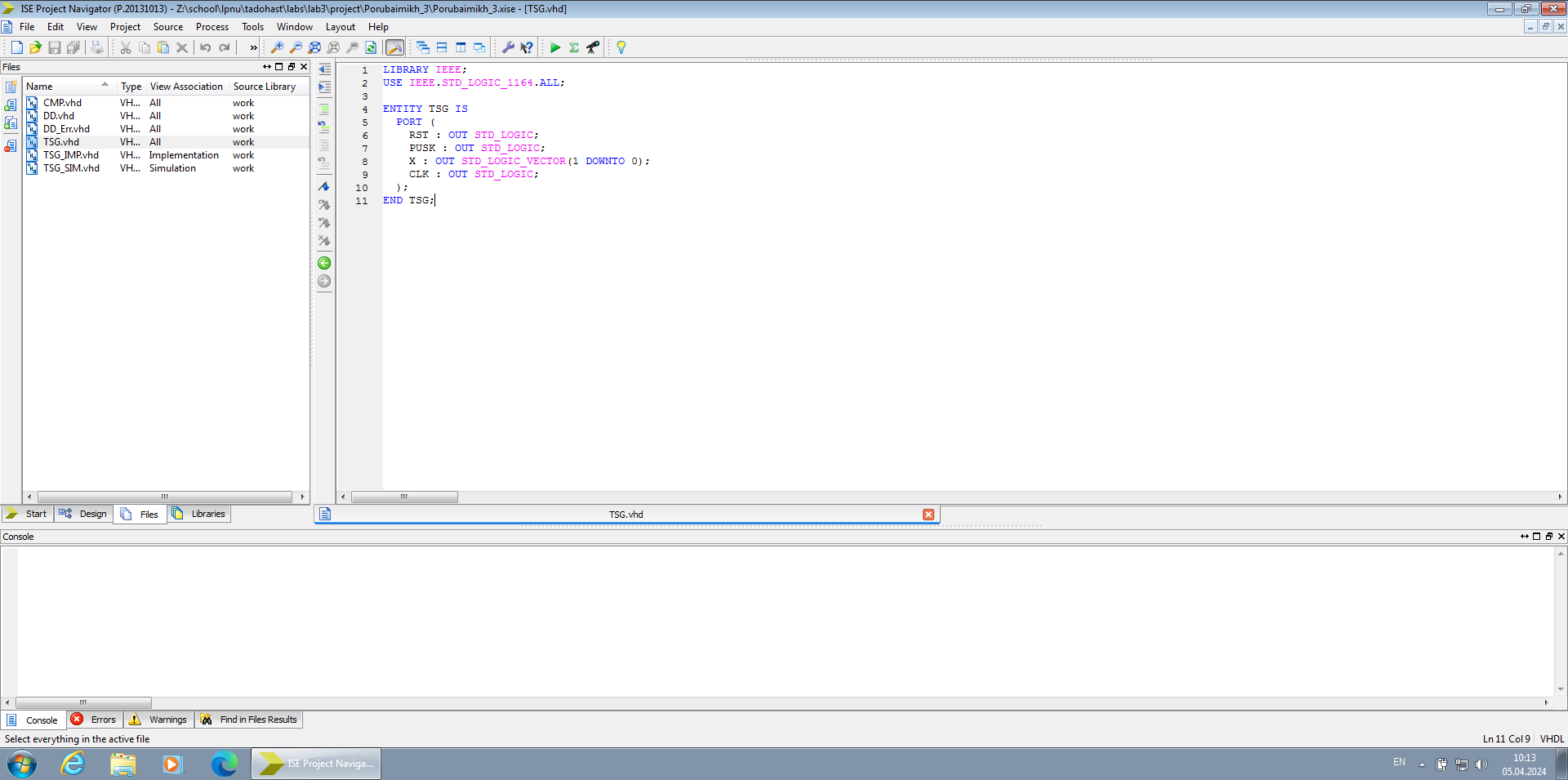


Рис. 4.10. Опис компонента TSG.vhd.

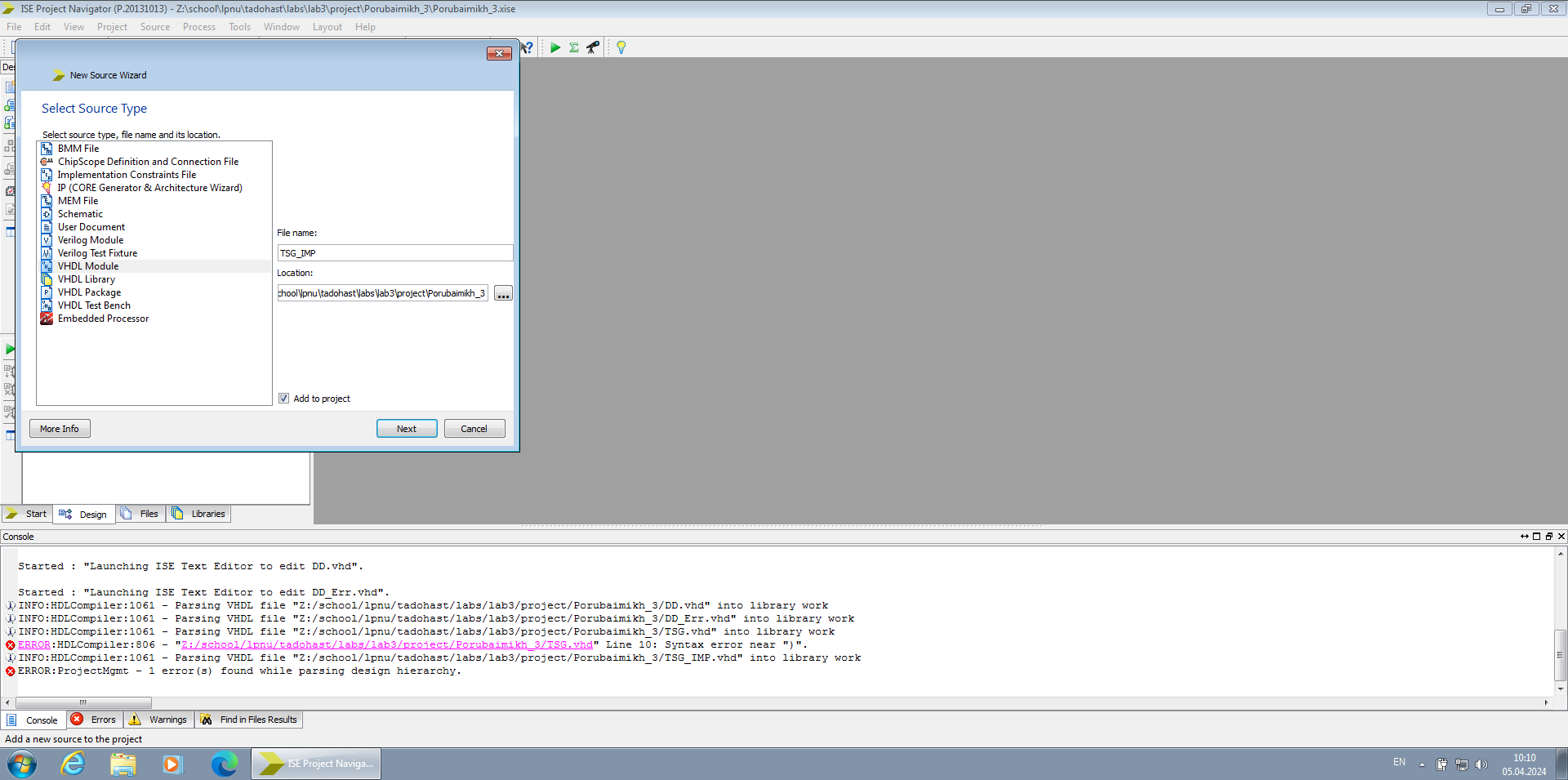


Рис. 4.11. Створення файлу TSG\_IMP.vhd.

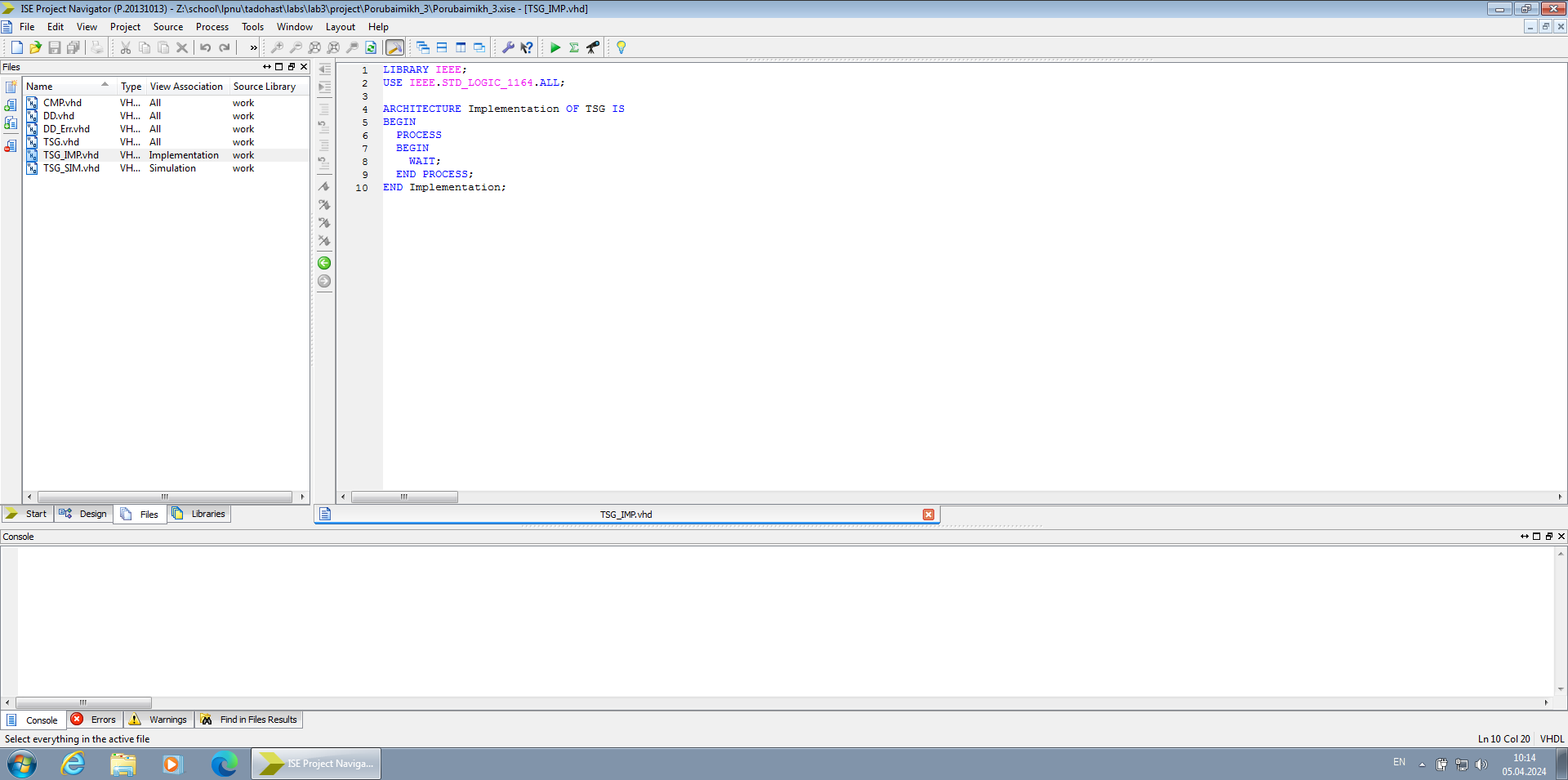


Рис. 4.12. Опис архітектури TSG для виконання.

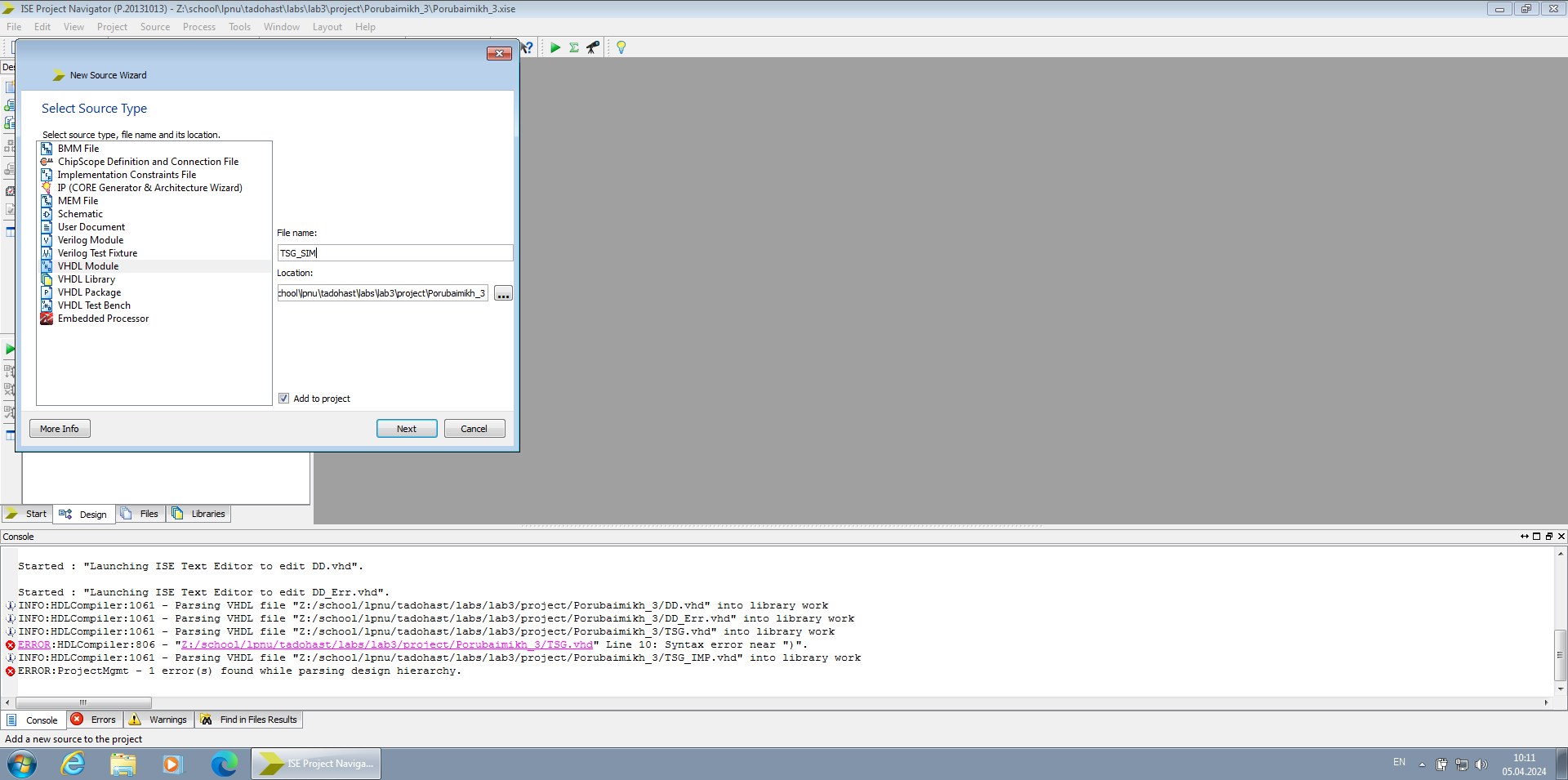


Рис. 4.13. Створення файлу TSG\_SIM.vhd.

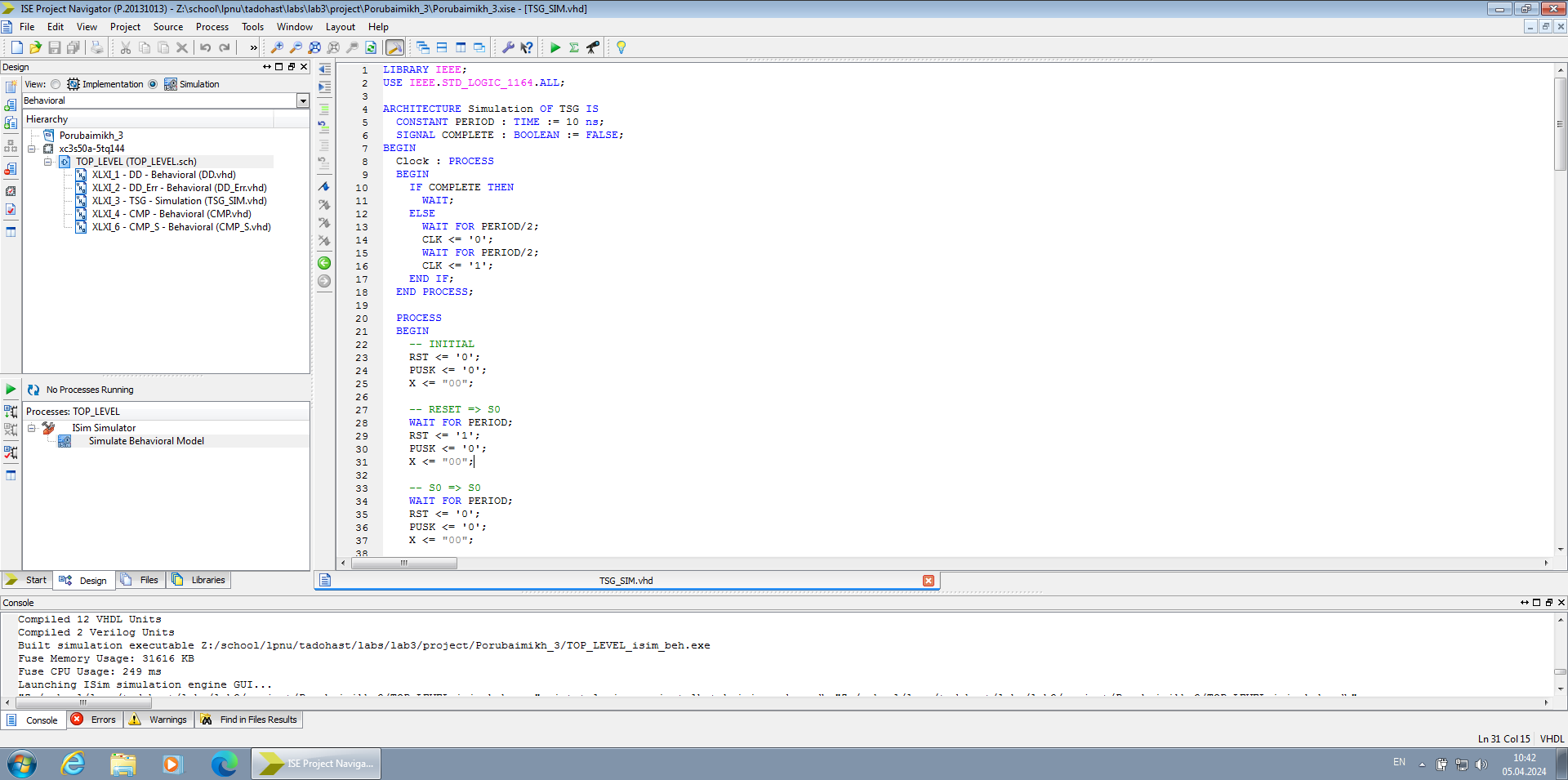


Рис. 4.14. Опис архітектури TSG для симуляції.

* 1. Схема.

Процес створення файлу схеми наведено на Рис. 4.15. Готову схему наведено на рис. 4.16.

Як видно, на схемі присутній генератор тествоих послідовностей, що генераує вхідні значення для схем ЦА. Кожний вихід генератора підключенно до відповідого маркеру виводу для зручного тестування. На схемі присутні дві схеми ЦА: еталонний (DD) і ЦА з закороткою (DD\_Err). Виходи Y кожного з них під’єднано до вузла порівняння CMP, а також до маркерів виводу для зручного тестування. Виходи STATE, відповідно, під’єднано до вузла CMP\_S і маркерів виводу. Вузли порівняння порівнюють два вхідних сигнали, і, якщо вони не співпадають, виводять логічну одиницю на виході Error.

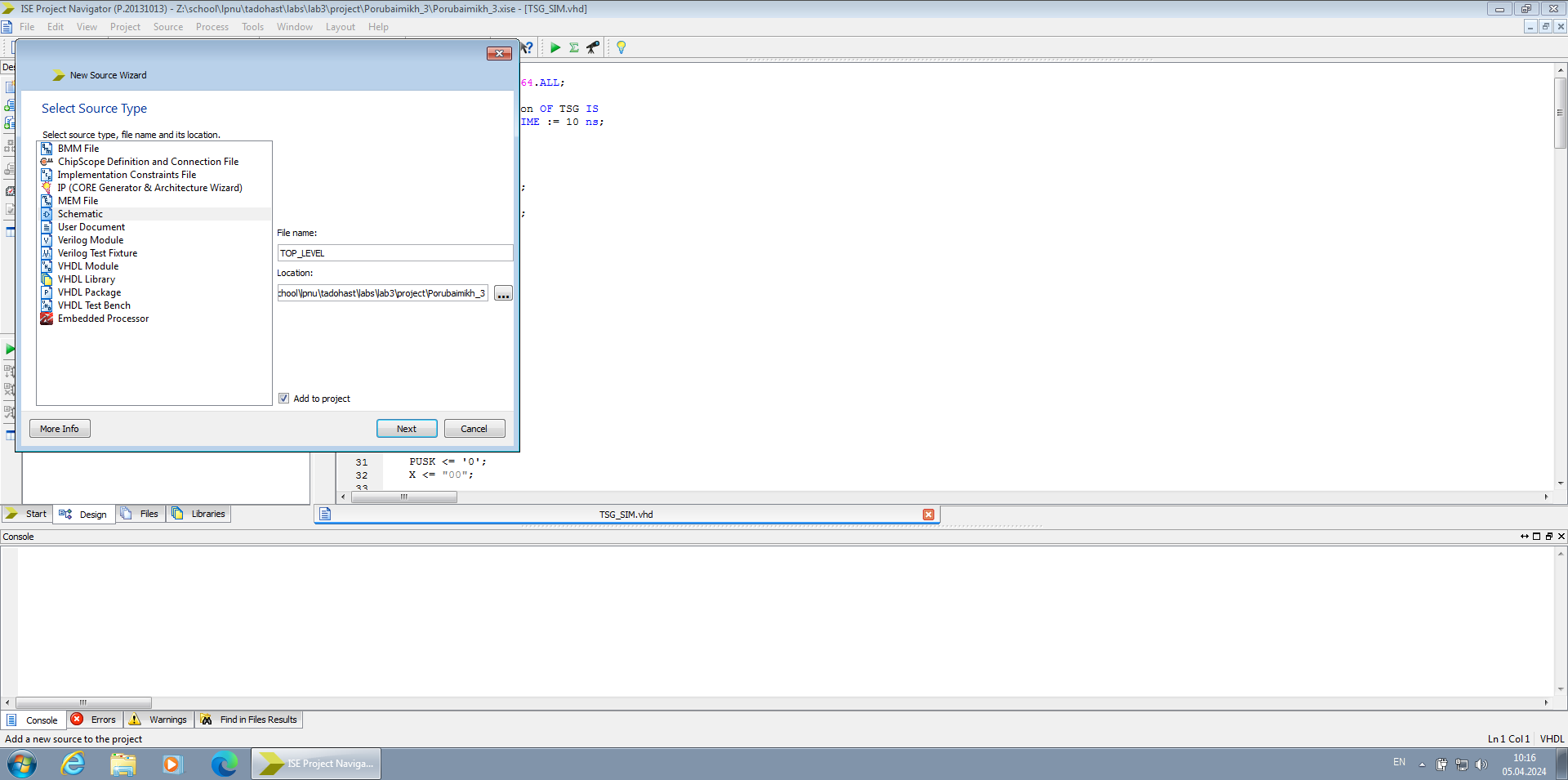


Рис. 4.15. Створення файлу схеми.

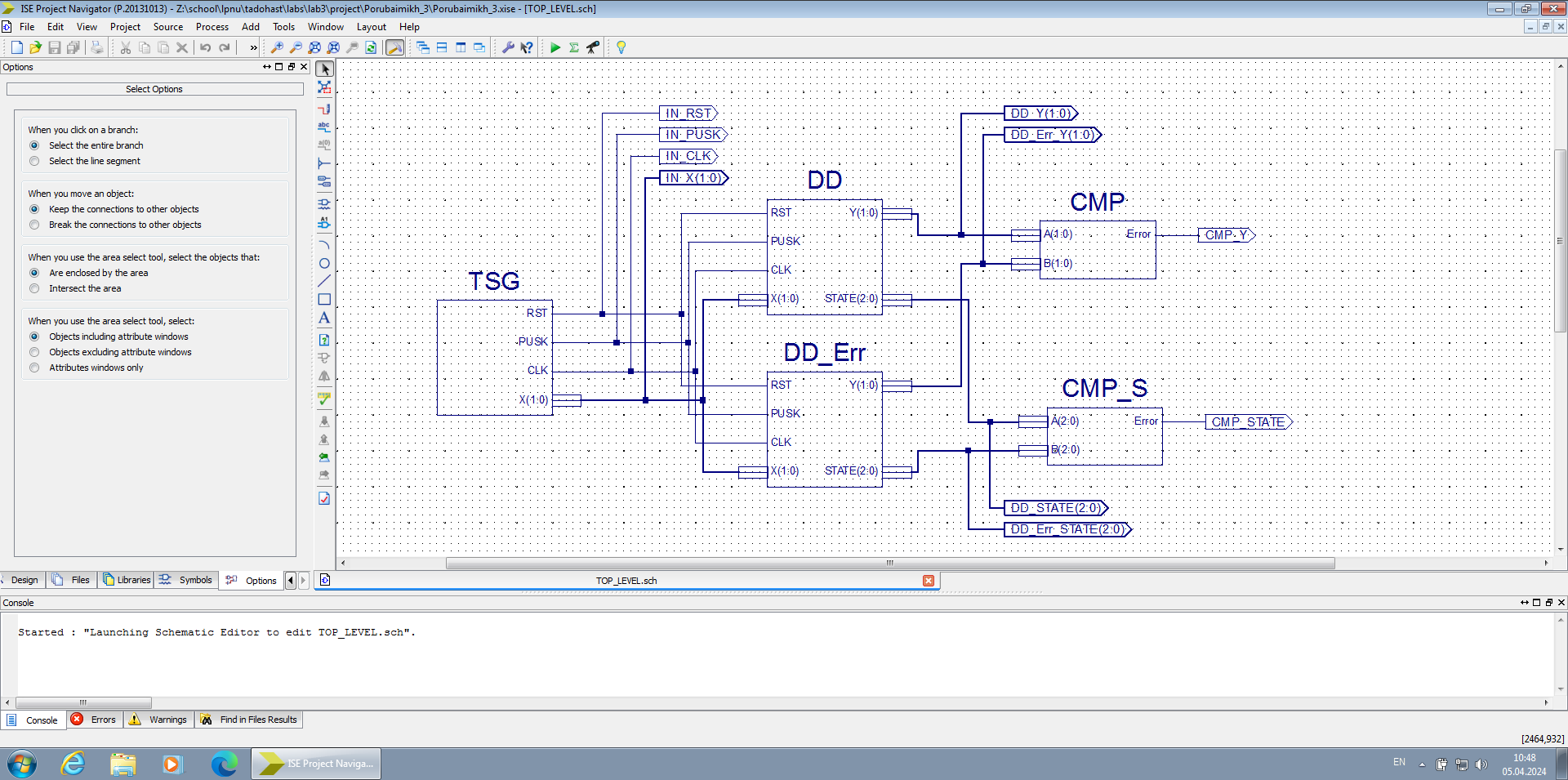


Рис. 4.16. Схема.

* 1. Процес тестування

Для виявлення помилок у схемі DD\_Err, необхідно порівняти його зі справним DD, перебравши усі можливі комбінації. Для цього було створено генератор тестових послідовностей, що перевіряє усі можливі переходи.

Як видно з результату тестування, еталонний ЦА переходить в стан S5 після стану S1 через передачу «1» в X(0). В свою чергу, несправний ЦА переходить у стан S2 через закоротку.

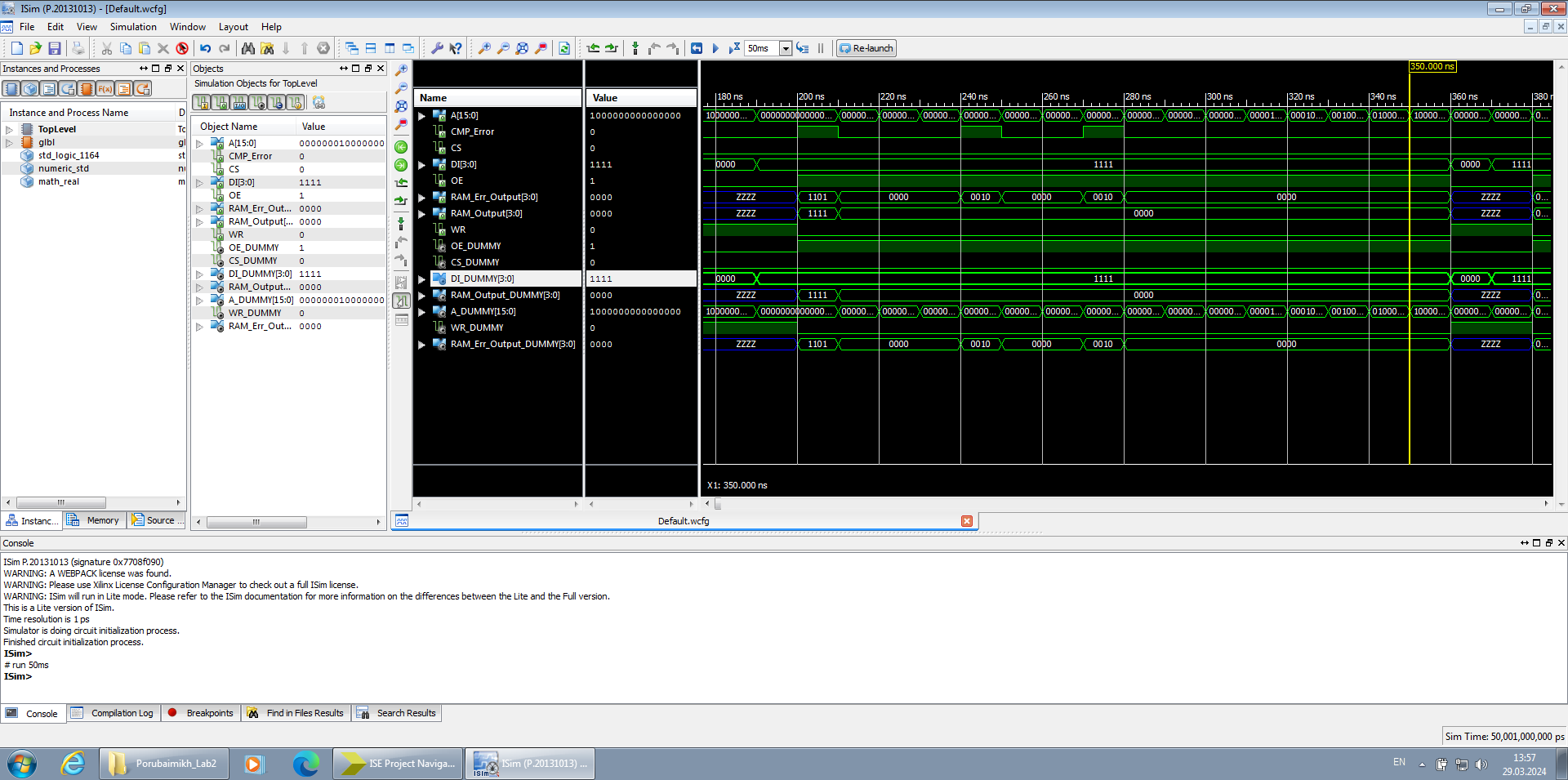


Рис. 4.17. Часова діаграма тестування шини адреси.

Висновок

Я ознайомився із загальною схемою тестування цифрової техніки. Засвоїв методи та засоби тестування цифрових схем з пам’яттю на прикладі цифрового автомата.

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

Додаток А. Код вузла DD.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY DD IS

PORT (

RST : IN STD\_LOGIC;

PUSK : IN STD\_LOGIC;

X : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

CLK : IN STD\_LOGIC;

Y : OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);

STATE : OUT STD\_LOGIC\_VECTOR (2 DOWNTO 0));

END DD;

ARCHITECTURE Behavioral OF DD IS

SIGNAL S : STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL S\_NEXT : STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL X\_INT : STD\_LOGIC\_VECTOR(1 DOWNTO 0);

BEGIN

PROCESS (CLK, RST)

BEGIN

IF RST = '1' THEN

S <= "000";

STATE <= "000";

ELSIF rising\_edge(CLK) THEN

S <= S\_NEXT;

STATE <= S\_NEXT;

END IF;

END PROCESS;

PROCESS (S, PUSK, X)

BEGIN

X\_INT <= X;

CASE S IS

WHEN "000" =>

Y <= "00";

IF PUSK = '0' THEN

S\_NEXT <= "000";

ELSE

S\_NEXT <= "001";

END IF;

WHEN "001" =>

Y <= "10";

IF X\_INT(0) = '0' THEN

S\_NEXT <= "101";

ELSE

S\_NEXT <= "010";

END IF;

WHEN "010" =>

Y <= "01";

IF X\_INT(0) = '0' THEN

S\_NEXT <= "011";

ELSE

S\_NEXT <= "010";

END IF;

WHEN "011" =>

Y <= "01";

IF X\_INT(1) = '0' THEN

S\_NEXT <= "100";

ELSE

S\_NEXT <= "111";

END IF;

WHEN "100" =>

Y <= "00";

S\_NEXT <= "101";

WHEN "101" =>

Y <= "10";

S\_NEXT <= "110";

WHEN "110" =>

Y <= "00";

IF X\_INT(1) = '0' THEN

S\_NEXT <= "010";

ELSE

S\_NEXT <= "111";

END IF;

WHEN "111" =>

Y <= "01";

IF PUSK = '0' THEN

S\_NEXT <= "000";

ELSE

S\_NEXT <= "111";

END IF;

WHEN OTHERS =>

S\_NEXT <= "000";

END CASE;

END PROCESS;

END Behavioral;

Додаток Б. Код вузла DD\_Err.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY DD\_Err IS

PORT (

RST : IN STD\_LOGIC;

PUSK : IN STD\_LOGIC;

X : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

CLK : IN STD\_LOGIC;

Y : OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);

STATE : OUT STD\_LOGIC\_VECTOR (2 DOWNTO 0));

END DD\_Err;

ARCHITECTURE Behavioral OF DD\_Err IS

SIGNAL S : STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL S\_NEXT : STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL X\_INT : STD\_LOGIC\_VECTOR(1 DOWNTO 0);

BEGIN

PROCESS (CLK, RST)

BEGIN

IF RST = '1' THEN

S <= "000";

STATE <= "000";

ELSIF rising\_edge(CLK) THEN

S <= S\_NEXT;

STATE <= S\_NEXT;

END IF;

END PROCESS;

PROCESS (S, PUSK, X)

BEGIN

X\_INT <= X;

X\_INT(0) <= '1';

CASE S IS

WHEN "000" =>

Y <= "00";

IF PUSK = '0' THEN

S\_NEXT <= "000";

ELSE

S\_NEXT <= "001";

END IF;

WHEN "001" =>

Y <= "10";

IF X\_INT(0) = '0' THEN

S\_NEXT <= "101";

ELSE

S\_NEXT <= "010";

END IF;

WHEN "010" =>

Y <= "01";

IF X\_INT(0) = '0' THEN

S\_NEXT <= "011";

ELSE

S\_NEXT <= "010";

END IF;

WHEN "011" =>

Y <= "01";

IF X\_INT(1) = '0' THEN

S\_NEXT <= "100";

ELSE

S\_NEXT <= "111";

END IF;

WHEN "100" =>

Y <= "00";

S\_NEXT <= "101";

WHEN "101" =>

Y <= "10";

S\_NEXT <= "110";

WHEN "110" =>

Y <= "00";

IF X\_INT(1) = '0' THEN

S\_NEXT <= "010";

ELSE

S\_NEXT <= "111";

END IF;

WHEN "111" =>

Y <= "01";

IF PUSK = '0' THEN

S\_NEXT <= "000";

ELSE

S\_NEXT <= "111";

END IF;

WHEN OTHERS =>

S\_NEXT <= "000";

END CASE;

END PROCESS;

END Behavioral;

Додаток В. Код вузла CMP.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY CMP IS

PORT (

A, B : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

Error : OUT STD\_LOGIC

);

END CMP;

ARCHITECTURE Behavioral OF CMP IS

BEGIN

PROCESS (A, B)

BEGIN

IF A = B THEN

Error <= '0';

ELSE

Error <= '1';

END IF;

END PROCESS;

END Behavioral;

Додаток Г. Код вузла CMP\_S.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY CMP\_S IS

PORT (

A, B : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

Error : OUT STD\_LOGIC

);

END CMP\_S;

ARCHITECTURE Behavioral OF CMP\_S IS

BEGIN

PROCESS (A, B)

BEGIN

IF A = B THEN

Error <= '0';

ELSE

Error <= '1';

END IF;

END PROCESS;

END Behavioral;

Додаток Д. Код компонента TSG.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY TSG IS

PORT (

RST : OUT STD\_LOGIC;

PUSK : OUT STD\_LOGIC;

X : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0);

CLK : OUT STD\_LOGIC

);

END TSG;

Додаток Е. Код архітектури компонента TSG для виконання.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ARCHITECTURE Implementation OF TSG IS

BEGIN

PROCESS

BEGIN

WAIT;

END PROCESS;

END Implementation;

Додаток Є. Код архітектури компонента TSG для симуляції.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ARCHITECTURE Simulation OF TSG IS

CONSTANT PERIOD : TIME := 10 ns;

SIGNAL COMPLETE : BOOLEAN := FALSE;

BEGIN

Clock : PROCESS

BEGIN

IF COMPLETE THEN

WAIT;

ELSE

WAIT FOR PERIOD/2;

CLK <= '0';

WAIT FOR PERIOD/2;

CLK <= '1';

END IF;

END PROCESS;

PROCESS

BEGIN

-- INITIAL

RST <= '0';

PUSK <= '0';

X <= "00";

-- RESET => S0

WAIT FOR PERIOD;

RST <= '1';

PUSK <= '0';

X <= "00";

-- S0 => S0

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S0 => S1

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '1';

X <= "00";

-- S1 => S5

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S5 => S6

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S6 => S7

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "10";

-- S7 => S7

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '1';

X <= "10";

-- S7 => S0

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "10";

-- RESET => S0

WAIT FOR PERIOD;

RST <= '1';

PUSK <= '0';

X <= "00";

-- S0 => S1

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '1';

X <= "00";

-- S1 => S2

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "01";

-- S2 => S2

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "01";

-- S2 => S3

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S3 => S7

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "10";

-- S7 => S0

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- RESET => S0

WAIT FOR PERIOD;

RST <= '1';

PUSK <= '0';

X <= "00";

-- S0 => S1

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '1';

X <= "00";

-- S1 => S2

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "01";

-- S2 => S3

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S3 => S4

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S4 => S5

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S5 => S6

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- S6 => S7

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "10";

-- S7 => S0

WAIT FOR PERIOD;

RST <= '0';

PUSK <= '0';

X <= "00";

-- RESET

WAIT FOR PERIOD;

RST <= '1';

PUSK <= '0';

X <= "00";

COMPLETE <= TRUE;

WAIT;

END PROCESS;

END Simulation;