Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт

Лабораторна робота № 3

З дисципліни «Тестування і діагностика програмно-апаратних засобів»

На тему: «Тестування цифрового автомата»

Виконав: ст. гр. КІ-3X

Тупісь М.Н.

Перевірив:

професор кафедри ЕОМ

Глухов В.С.

Львів 2025

Зміст

[Розділ 1. Мета роботи. 3](#_Toc158568300)

[Розділ 2. Теоретичні відомості. 4](#_Toc158568301)

[Розділ 3. Завдання. 5](#_Toc158568302)

[Розділ 4. Хід роботи. 6](#_Toc158568303)

[4.1. Еталонний вузол. 6](#_Toc158568304)

[4.2. Еталонний вузол з наперед заданою помилкою. 6](#_Toc158568305)

[4.3. Вузол порівняння. 7](#_Toc158568306)

[4.4. Схема порівняння. 7](#_Toc158568307)

[4.5. Генератор тестових послідовностей. 8](#_Toc158568308)

[Висновки 10](#_Toc158568309)

[Список використаних джерел 11](#_Toc158568310)

[ДОДАТОК А. Код вузла FSM\_m. 12](#_Toc158568311)

[ДОДАТОК Б. Код вузла FSM\_Error. 14](#_Toc158568312)

[ДОДАТОК В. Код вузла Comp. 16](#_Toc158568313)

[ДОДАТОК Г. Код вузла test\_bench. 17](#_Toc158568314)

Перелік рисунків

[Рис. 4.1. Еталонний ЦА. 6](#_Toc158568315)

[Рис. 4.2. ЦА з помилкою. 7](#_Toc158568316)

[Рис. 4.3. Вузол порівняння. 7](#_Toc158568317)

[Рис. 4.4. Схема. 8](#_Toc158568318)

[Рис. 4.5. Створення TestBench. 8](#_Toc158568319)

# Мета роботи.

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

# Теоретичні відомості.

Цифровий автомат є прикладом пристрою із пам'яттю. Цифровий автомат характеризується набором вхідних сигналів, набором вихідних сигналів, набором внутрішніх станів, початковим станом, правилом формування вихідних сигналів і правилом формування наступного внутрішнього стану. Існують різні способи завдавання автоматів, включаючи

* табличний метод,
* часові діаграми,
* аналітичний метод за допомогою мов опису апаратної частини цифрових вузлів (HDL),
* алгоритм роботи автомата,
* граф автомата та
* опис звичайною мовою.

Цифрові автомати поділяються на

1. Автомати Мура (де вихідні сигнали залежать тільки від стану автомата) та
2. Автомати Мілі (де вихідні сигнали залежать від стану автомата і вхідних сигналів).

Більшість цифрових автоматів є синхронними, тобто зміна стану автомата відбувається фронтом синхроімпульсів. Основний принцип тестування цифрових автоматів полягає у перевірці переходів і формуванні вихідних сигналів відповідно до графа

# Завдання.

Необхідно розробити програму тестування ЦА, описати поведінку еталонного ЦА (створити еталон), за результатами тестування ЦА знайти прояви помилки, визначити тип несправності (неправильне формування вихідних сигналів, неправильний перехід до наступного стану), визначити причину несправності (обрив або закоротка вхідного чи вихідного сигналу). Завдання було взяте з літератури [[](#с)1].

При виконанні лабораторної роботи використано засоби проектування ПЛІС [2].

Варіант -Помилка у стані S[6] ,а саме у сигналі s[2].

# Хід роботи.

## Еталонний вузол.

Опис еталонного цифрового атомату наведено на Рис. 4.1, опис вузла FSM\_m містить ДОДАТОК А.

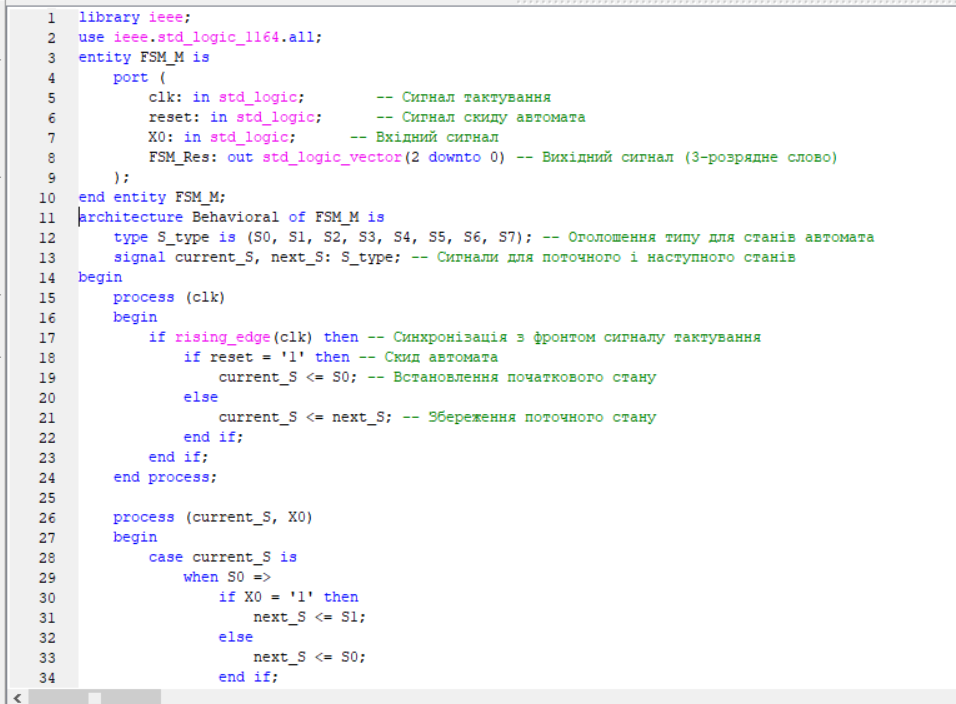


Рис. 4.1. Еталонний ЦА.

## Еталонний вузол з наперед заданою помилкою.

Опис цифрового атомату з помилкою наведено на Рис. 4.2, опис вузла FSM\_Error містить ДОДАТОК Б.

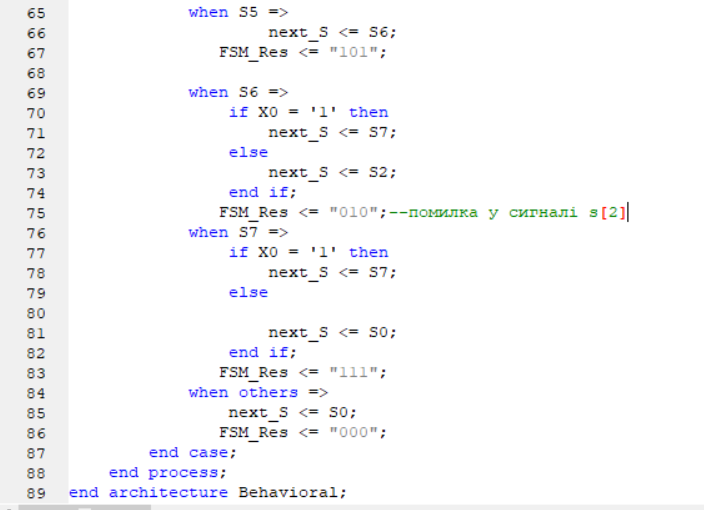


Рис. 4.2. ЦА з помилкою.

## Вузол порівняння.

Опис вузлу порівняння наведено на Рис. 4.3, опис вузла Comparison містить ДОДАТОК В.

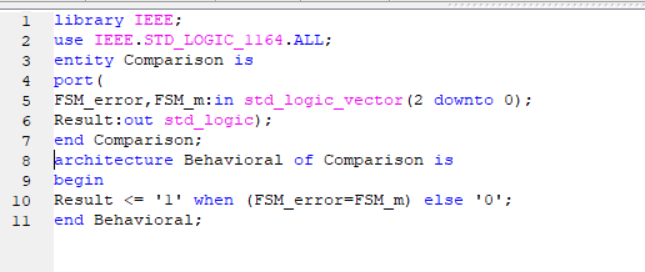


Рис. 4.3. Вузол порівняння.

## Схема порівняння.

Створюю схему порівняння. Опис схеми наведено на Рис. 4.4.

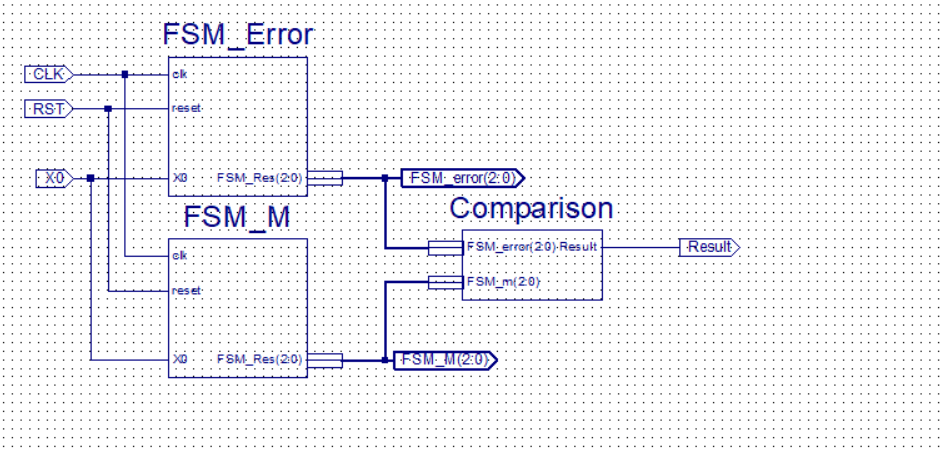


Рис. 4.4. Схема.

## Генератор тестових послідовностей.

Опис створення TestBench наведено на Рис. 4.5, приклади послідовностей сигналів в цьому тестовому стенді служать для симуляції різних сценаріїв роботи автомата. Опис вузла test\_bench містить ДОДАТОК Г.

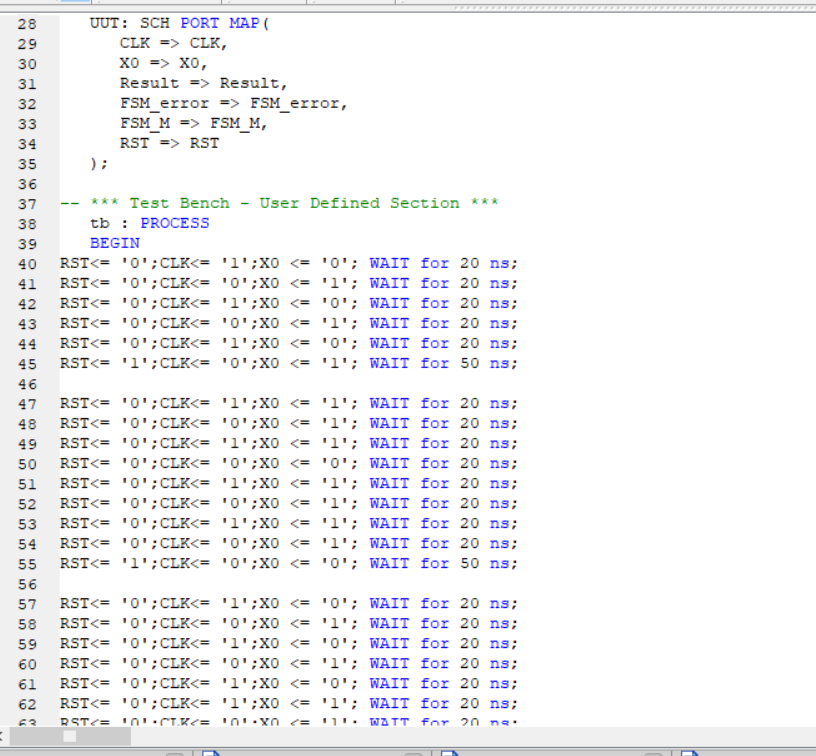


Рис. 4.5. Створення TestBench.

На графіку часової діаграми, зображеному на Рис. 4.6, можна помітити помилку у сигналі result, яка виявляється тоді, коли значення цього сигналу дорівнює нулю. Було виявлено, що помилка виникає в стані S[6]. Визначив цю помилку за допомогою Рис. 4.6. Порівнявши значення fsm\_error (зі значенням 010) і fsm\_m (зі значенням 110 ), було визначено, що помилка відбувається на сигналі s[2].

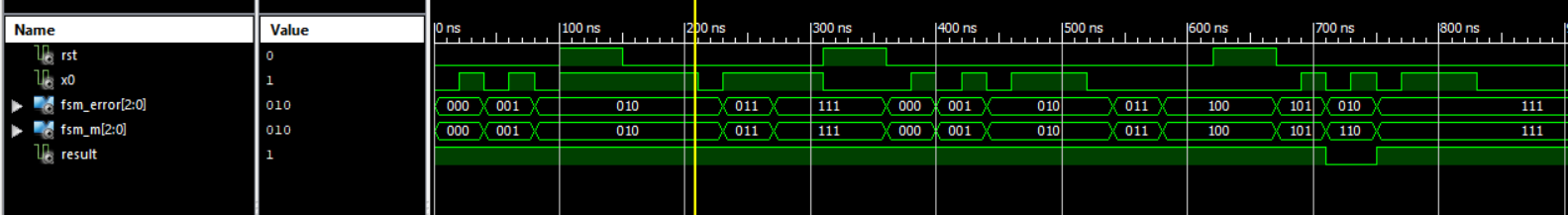


Рис. 4.6 Часова діаграма та симуляція роботи TestBench.

Висновки

Під час даної лабораторної роботи ознайомився із загальною схемою тестування цифрового автомата, розробив програму тестування ЦА. Також мною була виявлена помилка у схемі. Помилка у стані S[6], а саме у сигналі s[2], була виявлена за допомогою симуляції TestBench.

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. – 117 с.
2. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL / И. Е. Тарасов. – М.: Горячая линия – Телеком, 2005. – 252 с .
3. Код вузла FSM\_m.

library ieee;

use ieee.std\_logic\_1164.all;

entity FSM\_M is

port (

clk: in std\_logic;

reset: in std\_logic;

X0: in std\_logic;

FSM\_Res: out std\_logic\_vector(2 downto 0)

);

end entity FSM\_M;

architecture Behavioral of FSM\_M is

type S\_type is (S0, S1, S2, S3, S4, S5, S6, S7);

signal current\_S, next\_S: S\_type;

begin

process (clk)

begin

if rising\_edge(clk) then

if reset = '1' then

current\_S <= S0;

else

current\_S <= next\_S;

end if;

end if;

end process;

process (current\_S, X0)

begin

case current\_S is

when S0 =>

if X0 = '1' then

next\_S <= S1;

else

next\_S <= S0;

end if;

FSM\_Res <= "000";

when S1 =>

if X0 = '1' then

next\_S <= S2;

else

next\_S <= S5;

end if;

FSM\_Res <= "001";

when S2 =>

if X0 = '1' then

next\_S <= S2;

else

next\_S <= S3;

end if;

FSM\_Res <= "010";

when S3 =>

if X0 = '1' then

next\_S <= S7;

else

next\_S <= S4;

end if;

FSM\_Res <= "011";

when S4 =>

next\_S <= S5;

FSM\_Res <= "100";

when S5 =>

next\_S <= S6;

FSM\_Res <= "101";

when S6 =>

if X0 = '1' then

next\_S <= S7;

else

next\_S <= S2;

end if;

FSM\_Res <= "110";

when S7 =>

if X0 = '1' then

next\_S <= S7;

else

next\_S <= S0;

end if;

FSM\_Res <= "111";

when others =>

next\_S <= S0;

FSM\_Res <= "000";

end case;

end process;

end architecture Behavioral;;

1. Код вузла FSM\_Error.

library ieee;

use ieee.std\_logic\_1164.all;

entity FSM\_Error is

port (

clk: in std\_logic;

reset: in std\_logic;

X0: in std\_logic;

FSM\_Res: out std\_logic\_vector(2 downto 0)

);

end entity FSM\_Error;

architecture Behavioral of FSM\_Error is

type S\_type is (S0, S1, S2, S3, S4, S5, S6, S7);

signal current\_S, next\_S: S\_type;

begin

process (clk)

begin

if rising\_edge(clk) then

if reset = '1' then

current\_S <= S0;

else

current\_S <= next\_S;

end if;

end if;

end process;

process (current\_S, X0)

begin

case current\_S is

when S0 =>

if X0 = '1' then

next\_S <= S1;

else

next\_S <= S0;

end if;

FSM\_Res <= "000";

when S1 =>

if X0 = '1' then

next\_S <= S2;

else

next\_S <= S5;

end if;

FSM\_Res <= "001";

when S2 =>

if X0 = '1' then

next\_S <= S2;

else

next\_S <= S3;

end if;

FSM\_Res <= "010";

when S3 =>

if X0 = '1' then

next\_S <= S7;

else

next\_S <= S4;

end if;

FSM\_Res <= "011";

when S4 =>

next\_S <= S5;

FSM\_Res <= "100";

when S5 =>

next\_S <= S6;

FSM\_Res <= "101";

when S6 =>

if X0 = '1' then

next\_S <= S7;

else

next\_S <= S2;

end if;

FSM\_Res <= "010";-- помилка у сигналі s[2]

when S7 =>

if X0 = '1' then

next\_S <= S7;

else

next\_S <= S0;

end if;

FSM\_Res <= "111";

when others =>

next\_S <= S0;

FSM\_Res <= "000";

end case;

end process;

end architecture Behavioral;

1. Код вузла Comp.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Comparison is

port(

FSM\_error,FSM\_m:in std\_logic\_vector(2 downto 0);

Result:out std\_logic);

end Comparison;

architecture Behavioral of Comparison is

begin

Result <= '1' when (FSM\_error=FSM\_m) else '0';

end Behavioral;

1. Код вузла test\_bench.

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

LIBRARY UNISIM;

USE UNISIM.Vcomponents.ALL;

ENTITY SCH\_SCH\_sch\_tb IS

END SCH\_SCH\_sch\_tb;

ARCHITECTURE behavioral OF SCH\_SCH\_sch\_tb IS

COMPONENT SCH

PORT( CLK:INSTD\_LOGIC;

X0:INSTD\_LOGIC;

Result:OUTSTD\_LOGIC;

FSM\_error:OUTSTD\_LOGIC\_VECTOR (2 DOWNTO 0);

FSM\_M:OUTSTD\_LOGIC\_VECTOR (2 DOWNTO 0);

RST:INSTD\_LOGIC);

END COMPONENT;

SIGNAL CLK:STD\_LOGIC;

SIGNAL X0:STD\_LOGIC;

SIGNAL Result:STD\_LOGIC;

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

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

SIGNAL RST:STD\_LOGIC;

BEGIN

UUT: SCH PORT MAP(

CLK => CLK,

X0 => X0,

Result => Result,

FSM\_error => FSM\_error,

FSM\_M => FSM\_M,

RST => RST

);

-- \*\*\* Test Bench - User Defined Section \*\*\*

tb : PROCESS

BEGIN

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '1';CLK<= '0';X0 <= '1'; WAIT for 50 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '1';CLK<= '0';X0 <= '0'; WAIT for 50 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '0'; WAIT for 20 ns;

RST<= '1';CLK<= '0';X0 <= '0'; WAIT for 50 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '0'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '1';X0 <= '1'; WAIT for 20 ns;

RST<= '0';CLK<= '0';X0 <= '0'; WAIT for 20 ns;

WAIT; -- will wait forever

END PROCESS;

-- \*\*\* End Test Bench - User Defined Section \*\*\*

END;