Swami Vivekananda Institute of Science and Technology



Project   
 on  
 **Digital IC Tester Using VHDL**

Submitted By:

Subhajit Biswas : 44

Sk. Md. Tazeebur Rahman : 53

Shreya Das : 54

Shibam Puitandy : 56

Sangeeta Sarkar : 62

Rohit Roy : 66 Guided By: Dr.Chittajit Sarkar

**DIGITAL IC TESTER USING VHDL**

**Thesis submitted in partial fulfillment for the degree of Bachelor of Technology in Electronics and Communication Engineering of Maulana Abul Kalam Azad University of Technology**

****

***By***

**Subhajit Biswas Roll No.: 44**

**Sk. Md. Tahzeebur Rahman Roll No.: 53**

**Shreya Das Roll No.: 54**

**Shibam Puitandy Roll No.: 56**

**Sangeeta Sarkar Roll No.: 62**

**Rohit Roy Roll No.: 66**

## *Under the supervision of*

**Dr. Chittajit Sarkar**

(Head Of the Department, Dept of ECE, SVIST)

**Department of Electronics & Communication Engineering**

**Swami Vivekananda Institute of Science and**

**Technology**

****

*P.S. Sonarpur, Kolkata-700145*

*Certificate of Recommendation*

This is to certify that the project entitled ***“Digital IC Tester Using VHDL ”*** submitted by **Subhajit Biswas,Sk. Md. Tahzeebur Rahman,** **Shreya Das, Shibam Puitandy, Sangeeta Sarkar & Rohit Roy** is absolutely based upon their own work under the supervision of Dr. Chittajit Sarkar (H.O.D., Dept of ECE, SVIST) and that neither their thesis has been submitted for any degree/diploma or any other academic award anywhere before.

## *………………………………*

## Dr. Chittajit Sarkar

(H.O.D, Dept. of ECE)

........................................

Dr. Chittajit Sarkar

**Swami Vivekananda Institute of Science & Technology**

***Affiliated to***

***Maulana Abul Kalam Azad University of Technology***

***Certificate of Approval***

The foregoing thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned don’t necessarily endorse or approve any statement made opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which is submitted it.

Signature of the Examiners*:*

*1…………………………………………….*

*2…………………………………………….*

*3…………………………………………….*

***\*Only in the case the thesis is approved***

## *Acknowledgement*

It is a great pleasure to express my deepest gratitude and indebtedness to my (internal / external) guide(s), Dr. Chittajit Sarkar., Department of Electronics and Communication Engineering, Swami Vivekananda Institute of Science & Technology, Kolkata for his supervision, constant help and encouragement throughout the entire period.

We are very much grateful to Dr. Chittajit Sarkar(Head of the Department), Department of Electronics and Communication Engineering, Swami Vivekananda Institute of Science & Technology for necessary suggestions and helpful scientific discussions to improve the quality this thesis.

We are also thankful to the all faculty members and technical staff of the department for their help whenever we needed it.

# Subhajit BiswasSk. Md. Tahzeebur Rahman

###### Shreya DasShibam Puitandy

Sangeeta Sarkar Rohit Roy

Department of Electronics & Communication Engineering,

**Swami Vivekananda Institute of Science & Technology**

**ABSTRACT:**

IC tester is a microcontroller based circuitry that tests weather Digital the IC is in good working condition or bad condition. In industries, testing of the product is a major and expensive and time consuming process. Before making the whole system work, testing is mandatorily performed to avoid errors and undesired results. Similarly, in educational institutions, while performing practical it is necessary to check the ICs whether it is good or bad before performing experiments. Many a small faults at IC level makes system perform inaccurately and produce wrong outputs. The proposed system gives a cheap, small, portable and easy to handle IC tester that tests the ICs belonging to basic gate circuitary such as OR Gate, NAND Gate, AND Gate, NOR Gate, NOT Gate.

**INTRODUCTION:**

The IC tester simply determines which are workable gates and which are faulty. The main purpose of the project is to develop a digital IC tester that is very less expensive and handy than that of what are available in markets. The aim is to check the ICs in very due course of time and display results of ICs being good or faulty immediately. The necessary input signal conditions are applied to the inputs of the gate through FPGA and output of each gate is monitored and compared with the truth table, and depending on that comparison IC is tested whether it is good or faulty. The basic function of digital IC tester is to test the logic functioning of the ICs as described in the truth table/function table. The truth tables are stored in database while coding of the FPGA. The test displays the good ICs and faulty ICs on LCD. The test is being accomplished with the ICs belonging to the basic logic gate IC series.

**PROBLEM STATEMENT:**

In general it is the most important requirement of testing the ICs before using them for IC based work. Faulty ICs gives us results that are inappropriate to work and unexpected. These wrong outputs results leads in wrong evaluation and analysis. Time and resources are very important in this developing era. And one cannot afford wasting so much of time and resources in finding silly faults caused by faulty ICs. Thus there is a need to develop a low cost and easy maintainable, user friendly digital IC tester.

**DIGITAL IC TESTER:**

An Integrated Circuit tester (IC tester) is used to test Integrated Circuits (ICs). We can easily test any digital IC using this kind of an IC tester. For testing an IC, we need to use different hardware circuits for different ICs; like we need a particular kind of tester for testing a logic gate and another for testing flip flops or shift registers which involves more complication and time involved will also be more. So here’s an IC tester to overcome this problem. Unlike other IC testers, this is more reliable and easier since we don’t need to rig up different kind of circuits for different kind of ICs, each time we need to test them.

**IMPLEMENTATION OF DIGITAL IC TESTER USING FPGA AND VHDL:**

At first the Keypad and the ZIF socket are connected to unused pins of FPGA, which can be configured using VHDL or Verilog code. Then entered IC number is communicated to FPGA chip. For a particular IC number, corresponding hex code is provided. Then suitable subroutine is executed.

Result is displayed on the LCD according to the condition of the entered IC. Thus an IC is tested.

Unlike the IC testers available in the market today which are usually expensive, this IC tester is affordable and user-friendly. This IC tester is constructed using SPARTAN 3A along with a keyboard and a display unit. It can test digital ICs having a maximum of 14 pins. Since it is programmable, any number of ICs can be tested within the constraint of the memory available. This IC tester can be used to test a wide variety of ICs which includes simple logic gates and also sequential and combinational ICs like flip-flops, counters, shift registers etc. It is portable and easy to use.

**LIST OF ICs:**

|  |  |  |
| --- | --- | --- |
| Sl.No. | IC Number | IC Name |
| 1. | **74LS32** | **Two input one output OR gate.** |
| 2. | **74HC08** | **Two input one output AND gate.** |
| 3. | **74HC00** | **Two input one output NAND gate.** |
| 4. | **74LS02** | **Two input one output NOR gate.** |
| 5. | **74LS04** | **One input one output NOT gate.** |
| 6. | **74HCT86** | **Two input one output XOR gate.** |
| 7. |  | **Two input one output XNOR gate.** |

Table1:List of ICs

**BLOCK DIAGRAM OF THE SYSTEM:**

LCD

FPGA Chip

FPGA CHIP

fig:1-Block diagram of the system

**VHDL:**

VHDL Stands for VHSIC Hardware Description Language and VHSIC stands for Very High Speed Integrated Circuit. It is a concurrent language so different processes are executed simultaneously. It describes the behavior of digital circuit designs. The entity section declares I/O ports of the circuit. The architecture portion describes the circuit’s be havior, before which Standardized design libraries are included.

**Standard Libraries:**

* Include library ieee; before entity declaration.
* ieee.std\_logic\_1164 : a standard for designers to use in describing interconnection data types used in VHDL modelling.
* ieee.std\_logic\_arith : a set of arithmetic, conversion, comparison functions for signed, unsigned, std\_ulogic, std\_logic, std\_logic\_vector.
* ieee.std\_logic\_unsigned : a set of unsigned arithmetic, conversion, and comparison functions for std\_logic\_vector.

**Data Types:**

* Enumerated Types:
  + BIT ('0','1')
  + BOOLEAN (false, true)
  + STD\_LOGIC ('U','X','0','1','Z','W','L','H','-') where:

'U' means uninitialized

'X' means unknown

'0' means low

'1' means high

'Z' means high impedance

'W' means weak unknown

'L' means weak low

'H' means weak high

'-' means don't care

For XST synthesis, the '0' and 'L' values are treated identically, as are '1' and 'H'. The 'X', and '-' values are treated as don't care. The 'U' and 'W' values are not accepted by XST. The 'Z' value is treated as high impedance.

* + User defined enumerated type:

type COLOR is (RED,GREEN,YELLOW);

The following types are VHDL predefined types:

* BIT
* BOOLEAN
* BIT\_VECTOR
* INTEGER

The following types are declared in the STD\_LOGIC\_1164 IEEE package.

* STD\_LOGIC
* STD\_LOGIC\_VECTOR

This package is compiled in the IEEE library. In order to use one of these types, the following two lines must be added to the VHDL specification:

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

**Entity and Architecture:**

An entity specifies the interface between the specified design (formally called a design entity) and the environment in which it operates. On the other hand, an architecture is a description of the inner design operation and it must be assigned to an entity. The architecture can be assigned to one entity only but one entity may be assigned to a number of architectures.

Example of Entity :

Entity test is

Port( A,B,C,D: in std\_logic;

E: out std\_logic);

End test;

Example of Architecture:

ARCHITECTURE BEHAVIOR OF TEST IS

SIGNAL X,Y : STD\_LOGIC;

BEGIN

X <= A AND B;

Y <= C AND D;

E <= X OR Y; end BEHAVIOR;

**If elsif else Statement:**

The if statement is a statement that depending on the value of one or more corresponding conditions, selects for execution one or none of the enclosed sequences of statements.

## Simplified Syntax

**if** condition**then**

  sequential\_statements

**end if**;

**if** condition**then**

  sequential\_statements

**else**

  sequential\_statements

**end if**;

**if** condition**then**

  sequential\_statements

**elsif** condition**then**

    sequential\_statements**else** sequential\_statement **end if**;

**FPGA:**

A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an Application-Specific Integrated Circuit (ASIC).FPGAs contain an array of programmable logic blocks, and a hierarchy of "reconfigurable interconnects" that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Many FPGAs can be reprogrammed to implement different logic functions allowing flexible reconfigurable computing as performed in computer software.

Contemporary field-programmable gate arrays (FPGAs) have large resources of logic gates and RAM blocks to implement complex digital computations. As FPGA designs employ very fast I/O rates and bidirectional data buses, it becomes a challenge to verify correct timing of valid data within setup time and hold time.

Floor planning enables resource allocation within FPGAs to meet these time constraints. FPGAs can be used to implement any logical function that an ASIC can perform. The ability to update the functionality after shipping, partial re-configuration of a portion of the designand the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.

**TYPICAL FPGA ARCHITECTURE:**

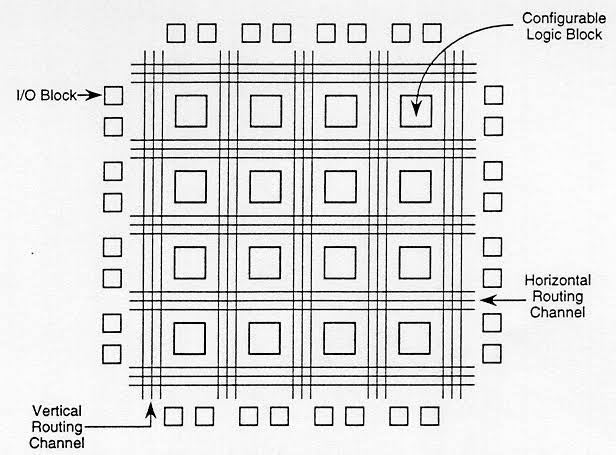


fig:2-Typical FPGA Architecture

**FPGA vs CPLD:**

CPLD (complex programmable logic device) is a programmable logic device that is made up of several simple PLDs (SPLDs) with a programmable switching matrix in between the logic blocks. It has the architectural features of both PAL and FPGA but is less complex than FPGA. Macro cell is the building block of the CPLD, which contains logic-implementing disjunctive normal-form expressions and more specialized logic operations.

The primary differences between CPLD and FPGA are architectural. A CPLD has a restrictive structure which results in less flexibility. The FPGA architecture is dominated by interconnects, which makes them not only far more flexible but also far more complex to design.

One major difference between the FPGA and CPLD architectures is that FPGAs are based on look-up tables while CPLDs form the logic functions with sea-of-gates.

**FPGA vs Microcontroller:**

A microcontroller is an integrated circuit chip that is often part of an embedded system. It is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. A microcontroller differs from a microprocessor, which is a general-purpose chip that is used to create a multi-function computer or device and requires multiple chips to handle various tasks.

A microcontroller is a computing system. It has lots of hierarchical rules and commands over its input and output. It has its own processing unit. On the contrary, FPGA is not a computing system.

FPGAs are used for simpler operations, but they have higher processing speeds than microcontrollers. The important points mentioned above are summarized in the table.

**SPARTAN 3A FPGA KIT**

****

fig.3-SPARTAN 3A KIT

**LCD:**

LCD (Liquid Crystal Display) is a type of flat panel display which uses liquid crystals in its primary form of operation. LEDs have a large and varying set of use cases for consumers and businesses, as they can be commonly found in smart phones, televisions, computer monitors and instrument panels.

* Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
* Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
* Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
* Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
* Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
* Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
* Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
* Pin15 (+ve pin of the LED): This pin is connected to +5V.
* Pin 16 (-ve pin of the LED): This pin is connected to GND.

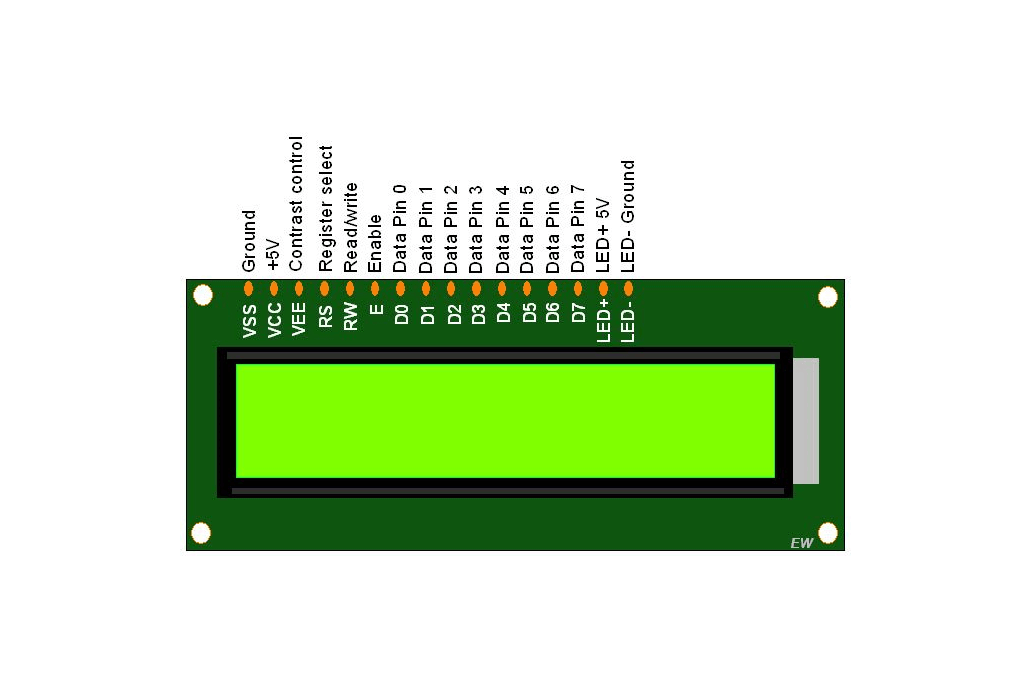


fig.4: LCD Display

**KEYPAD:**

The hex keypad is a peripheral that connects to the DE2 through JP1 or JP2 via a 40-pin ribbon cable. It has 16 buttons in a 4 by 4 grid, labeled with the hexadecimal digits 0 to F. An example of this can been seen in Figure 1, below. Internally, the structure of the hex keypad is very simple. Wires run in vertical columns (we call them C0 to C3) and in horizontal rows (called R0 to R3). These 8 wires are available externally, and will be connected to the lower 8 bits of the port. Each key on the keypad is essentially a switch that connects a row wire to a column wire. When a key is pressed, it makes an electrical connection between the row and column. The internal structure of the hex keypad is shown in Figure 2. The specific mapping of hex keypad wires (C0 to C3 and R0 to R3) to pins.

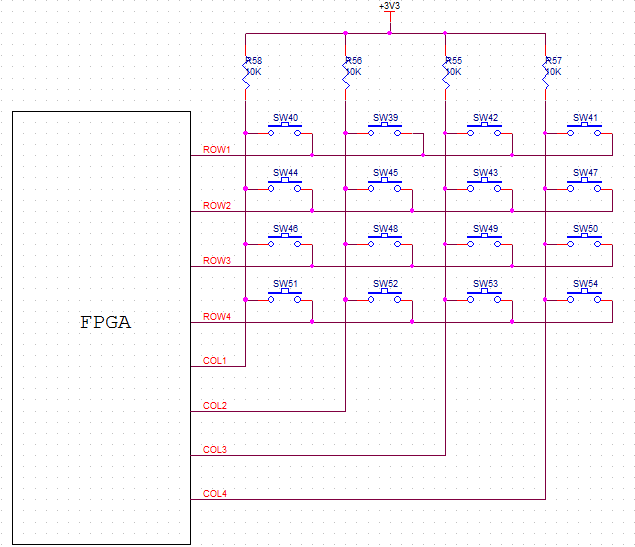
****

fig.5: keypad interfacing with FPGA

**ZIF Socket:**

ZIF(Zero Insertion Force) is a type of IC base which requires very little force for insertion. It fits up to 40 pin IC chips, in 0.3”-0.6” width. It's body size: 0.9”x 2.6” x 0.46”.

****

fig.6-ZIF Socket

**Interface Of ZIF Socket:**

fig.7-Interfacing of ZIF Socket with FPGA

**FLOW CHART:**

If IC is ok then output is '1'.

If IC is faulty then output is '0'.

**ALGORITHM:**

1.Begin.

2.Reset the circuit.

3.Insert a IC in the zif socket.

4.Display “Enter the IC number".

5.Give input values for input1 to input13

6.Enter the select lines inputs for the inserted IC.

7.Relevant block of the program will execute.

8.If IC is ok then output is 1, if IC is faulty then output is 0.

**ICs ACCORDING TO THE SELECT LINES:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sel1 | sel2 | sel3 | GATE | IC No. |
| 0 | 0 | 0 | AND | 74HC08 |
| 0 | 0 | 1 | OR | 74LS32 |
| 0 | 1 | 0 | NAND | 74HC00 |
| 0 | 1 | 1 | NOR | 74LS02 |
| 1 | 0 | 0 | XOR | 74HCT86 |
| 1 | 0 | 1 | XNOR | SN74LS266 |
| 1 | 1 | 0 | NOT | 74LS04 |

Table 2:ICs according to the select lines

**VHDL PROGRAM:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity main1 is

port ( input1 : in STD\_LOGIC;

input2 : in STD\_LOGIC;

input3 : in STD\_LOGIC;

input4 : in STD\_LOGIC;

input5 : in STD\_LOGIC;

input6 : in STD\_LOGIC;

input7 : in STD\_LOGIC;

input8 : in STD\_LOGIC;

input9 : in STD\_LOGIC;

input10 : in STD\_LOGIC;

input11 : in STD\_LOGIC;

input12 : in STD\_LOGIC;

input13 : in STD\_LOGIC;

sel1 : in STD\_LOGIC;

sel2 : in STD\_LOGIC;

sel3 : in STD\_LOGIC;

output : out STD\_LOGIC);

end main1;

architecture Behavioral of main1 is

begin

process ( sel1, sel2, sel3, input1, input2, input3,input4, input5, input6, input7,

input8, input9, input10, input11, input12, input13)

begin

if sel1= '0' and sel2= '0' and sel3= '0' then

output<= ( input3 xnor ( input1 and input2 )) and

( input11 xnor (input13 and input12 )) ;

elseif sel1= '0' and sel2= '0' and sel3= '1' then

output<= ( input3 xnor ( input1 or input2 )) and

( input11 xnor ( input13 or input12 ));

elseif sel1= '0' and sel2= '1' and sel3= '0' then

output<= ( input3 xnor ( input1 nand input2 )) and

( input 11 xnor ( input13 nand input12 ));

elseif sel1= '0' and sel2= '1' and sel3= '1' then

output<= ( input4 xnor ( input5 nor input6 ));

elseif sel1= '1' and sel2= '0' and sel3= '0' then

output<= ( input3 xnor ( input1 xor input2 )) and

( input11 xnor ( input13 xor input12 ));

elseif sel1= '1' and sel2= '0' and sel3= '1' then

output<= ( input3 xnor ( input1 xnor input input2 )) and

( input11 xnor ( input13 xnor input12 ));

elseif sel1= '1' and sel2= '1' and sel3= '0' then

output<= ( input8 xnor ( not input9 ));

end if;

end process;

end Behavioral;

# TESTBENCH PROGRAM:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY main1test is

END main1 test;

ARCHITECTURE behavior OF main1test IS

COMPONENT main1

PORT (

input1 : IN std\_logic ;

input2 : IN std\_logic;

input3 : IN std\_logic;

input4 : IN std\_logic;

input5 : IN std\_logic;

input6 : IN std\_logic;

input7 : IN std\_logic;

input8 : IN std\_logic;

input9 : IN std\_logic;

input10 : IN std\_logic;

input11 : IN std\_logic;

input12 : IN std\_logic;

input13 : IN std\_logic;

sel1 : IN std\_logic;

sel2 : IN std\_logic;

sel3 : IN std\_logic;

output : OUT std\_logic

);

END COMPONENT;

signal input1 : std\_logic : = '0' ;

signal input2 : std\_logic : = '0' ;

signal input3 : std\_logic : = '0' ;

signal input4 : std\_logic : = '0' ;

signal input5 : std\_logic : = '0' ;

signal input6 : std\_logic : = '0' ;

signal input7 : std\_logic : = '0' ;

signal input8 : std\_logic : = '0' ;

signal input9 : std\_logic : = '0' ;

signal input10 : std\_logic : = '0' ;

signal input11 : std\_logic : = '0' ;

signal input12 : std\_logic : = '0' ;

signal input13 : std\_logic : = '0' ;

signal sel1 : std\_logic : = '0' ;

signal sel2 : std\_logic : = '0' ;

signal sel3 : std\_logic : = '0' ;

signal output : std\_logic ;

BEGIN

uut : main1 PORT MAP (

input1 => input1,

input2 => input2,

input3 => input3,

input4 => input4,

input5 => input5,

input6 => input6,

input7 => input7,

input8 => input8,

input9 => input9,

input10 => input10,

input11 => input11,

input12 => input12,

input13 => input13,

sel1 => sel1,

sel2 => sel2,

sel3 => sel3,

output1 => output;

) ;

slim\_proc : process

begin

wait for 100 ns;

sel1 <= '1' ;

sel2 <= '0' ;

sel1 <= '0' ;

input1 <= '1' ;

input2 <= '1' ;

input3 <= '1' ;

input6 <= '1' ;

input5 <= '0' ;

input4 <= '1' ;

input9 <= '0' ;

input8 <= '1' ;

input12 <= '0' ;

input13 <= '0' ;

input11 <= '1' ;

wait;

end process;

END;

**SIMULATION PICTURE:**

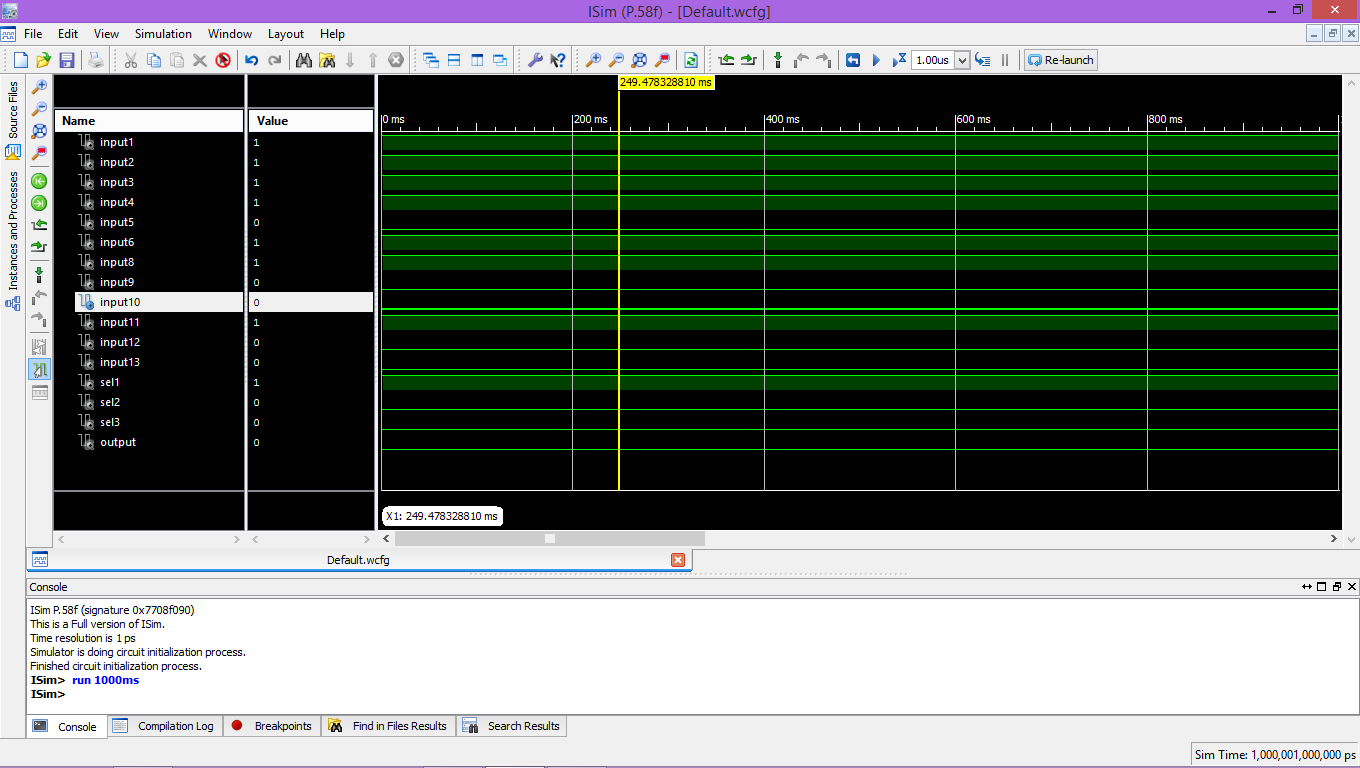


fig8: VHDL program simulation

**PROGRAM ANALYSIS:**

1. Input1 to input13 are given previously.
2. To check a particular IC, particular inputs of select lines(sel1, sel2, sel3) are entered.
3. When select lines are 000, IC7408 (AND gate) is checked.

If the result of (input1 and input2) matches the input3, we will get the result as 1; otherwise the result is 0. Because the XNOR gate gives the output 1 when the inputs are same and gives 0 for different values of inputs.

Similarly, when the result of (input13 and input12) matches the input11, the result will be 1; otherwise 0.

Now, if the final result from input1, input2, input3 and the final result from input11, input12, input13 are same then we'll get the output as 1; otherwise we'll get 0. Because AND gate gives 1 as output for same inputs and 0 for different inputs.

1. When select lines are 001, IC7432 (OR gate) is checked.

For select lines equal to 010, IC7400 (NAND gate) is checked.

For select lines equal to 011, IC7402 (NOR gate) is checked.

For select lines equal to 100, XOR gate is checked.

For select lines equal to 101, XNOR gate is checked.

For select lines equal to 110, IC7404 (NOT gate) is checked.

**DRAWBACKS:**

Generally, IC7408 contains four AND gates, IC7432 contains four OR gates, IC7400 contains four NAND gates, IC7402 contains four NOR gates, IC7404 contains six NOT gates. But as the program is getting complicated, we've coded to check two gates in IC7408, two gates in IC7432, two gates in IC7400, one gate in IC7402 and one gate in IC7404.

**REFERENCES:**

* **VLSI DESIGN By Debaprasad Das, OXFORD University Press.**
* [**https://www.xilinx.com/support/documentation/user\_guides/ug331.pdf**](https://www.xilinx.com/support/documentation/user_guides/ug331.pdf)
* [**https://www.xilinx.com/support/documentation/boards\_and\_kits/ug334.pdf**](https://www.xilinx.com/support/documentation/boards_and_kits/ug334.pdf)
* [**https://electrofriends.com/projects/microcontrollers/digital-ic-tester/**](https://electrofriends.com/projects/microcontrollers/digital-ic-tester/)
* [**http://wiki.sunfounder.cc/index.php?title=4X4\_Matrix\_**Keypad**\_Module**](http://wiki.sunfounder.cc/index.php?title=4X4_Matrix_Keypad_Module)