Mehran University of Engineering & Technology Jamshoro

**Institute of Information and Communication Technologies**

**Lab # 01**

**Introduction of FPGA and Familiar with NEXYS2 Board, VHDL & Xilinx ISE**

|  |
| --- |
| Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I.D No\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Score\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date\_\_\_\_\_\_\_\_\_\_ |

**FPGA:**

A **field-programmable gate array** (**FPGA**) is an [integrated circuit](http://en.wikipedia.org/wiki/Integrated_circuit) designed to be configured by a customer or a designer after manufacturing. The FPGA configuration is generally specified using a [hardware description language](http://en.wikipedia.org/wiki/Hardware_description_language) (HDL). FPGAs have large resources of [logic gates](http://en.wikipedia.org/wiki/Logic_gate) and RAM blocks to implement complex digital computations. As FPGA designs employ very fast I/Os and bidirectional data buses it becomes a challenge to verify correct timing of valid data within setup time and hold time. FPGAs contain [programmable logic](http://en.wikipedia.org/wiki/Programmable_logic_device) components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together". Somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex [combinational functions](http://en.wikipedia.org/wiki/Combinational_logic), or merely simple [logic gates](http://en.wikipedia.org/wiki/Logic_gate) like [AND](http://en.wikipedia.org/wiki/AND_gate), [XOR](http://en.wikipedia.org/wiki/XOR_gate). In most FPGAs, the logic blocks also include memory elements, which may be simple [flip-flops](http://en.wikipedia.org/wiki/Flip-flop_%28electronics%29) or more complete blocks of memory. Some FPGAs have analog features in addition to digital functions.

**Nexys2 Board:**

The Nexys2 circuit board as Shown in Figur-1and 2 is a complete, ready-to-use circuit development platform based on a Xilinx Spartan 3E FPGA. Its onboard high-speed USB2 port, 16Mbytes of RAM and ROM, and several I/O devices and ports make it an ideal platform for digital systems of all kinds, including embedded processor systems based on Xilinx‟s . The USB2 port provides board power and a programming interface, so the Nexys2 board can be used with a notebook computer to create a truly portable design station.

* 500K-gate Xilinx Spartan 3E FPGA
* USB2-based FPGA configuration and high-speed data transfers (using the free Adept Suite Software)
* USB-powered (batteries and/or wall-plug can also be used)
* 16MB of Micron PSDRAM &16MB of Intel StrataFlash ROM
* Xilinx Platform Flash for nonvolatile FPGA configurations
* Efficient switch-mode power supplies (good for battery
* powered applications)
* 50MHz oscillator plus socket for second oscillator
* 60 FPGA I/O’s routed to expansion connectors (one highspeed
* Hirose FX2 connector and four 6-pin headers)
* 8 LEDs, 4-digit 7-seg display, 4 buttons, 8 slide switches



 Figure-1 Nexys2 Board Block Diagram



Figure-2 Represents Nexys2 Board

**VHSIC Hardware Description Language (VHDL):**

The “V” in VHDL is short of yet another acronym: VHSIC or Very High-Speed Integrated Circuit. The HDL stands for Hardware Description Language. VHDL is a programming language that allows one to model and develop complex digital systems in a dynamic environment. VHDL is intended for circuit *synthesis* as well as circuit simulation.

A fundamental motivation to use VHDL (or its competitor, Verilog) is that VHDL is a standard, technology/vendor independent language, and is therefore portable and reusable.

Once the VHDL code has been written, it can be used either to implement the circuit in a programmable device (from Altera, Xilinx, Atmel, etc.) or can be submitted to a foundry for fabrication of an ASIC chip.

Currently, many complex commercial chips (microcontrollers, for example) are designed using such an approach.

**Design Flow:**



Figure-3 Design Flow Block Diagram

We start the design by writing the VHDL code, which is saved in a file with the extension *.vhd* and the same name as its ENTITY’s name.

**Step-1**: The synthesis process is compilation. Compilation is the conversion of the high-level VHDL language, which describes the circuit at the Register Transfer Level (RTL), into a netlist at the gate level.

**Step-2:** Optimization, which is performed on the gate-level netlist for speed or for area. At this stage, the design can be simulated. Finally, a place and-route (fitter) software will generate the physical layout for a PLD/FPGA chip or will generate the masks for an ASIC.

**Step-3:** place and-route (fitter) software will generate the physical layout for a PLD/FPGA chip or will generate the masks for an ASIC.

There are several EDA (Electronic Design Automation) tools available for circuit synthesis, implementation, and simulation using VHDL. Some tools (place and route, for example) are offered as part of a vendor’s design suite (e.g., Altera’s Quartus II, which allows the synthesis of VHDL code onto Altera’s CPLD/FPGA chips, or Xilinx’s ISE suite, for Xilinx’s CPLD/FPGA chips).

**VHDL Code:**

A VHDL code consists of ENTITY and ARCHITECTURE.

**ENTITY** is description of I/O pins (ports) of the circuit.

**ARCHITECTURE** describes how the circuit should function.

**Example of Full Adder:**

 







**Circuit Simulation:**

Following figure contains the simulation results from the circuit synthesized with the VHDL code, which implements the full-adder unit.+



**Xilinx ISE Design Software:**

**Introduction:**

The ISE® software controls all aspects of the design flow. Through the Project Navigator interface, you can access all of the design entry and design implementation tools. You can also access the files and documents associated with your project.

**Project Navigator Interface:**

By default, the Project Navigator interface is divided into four panel sub windows, as seen in Figure 1.3. On the top left are the Start, Design, Files, and Libraries panels, which include display and access to the source files in the project as well as access to running processes for the currently selected source. The Start panel provides quick access to opening projects as well as frequently access reference material, documentation and tutorials. At the bottom of the Project Navigator are the Console, Errors, and Warnings panels, which display status messages, errors, and warnings. To the right is a Multi document interface (MDI) window referred to as the Workspace. The Workspace enables you to view design reports, text files, schematics, and simulation waveforms. Each window can be resized, undocked from Project Navigator, moved to a new location within the main Project Navigator window, tiled, layered, or closed.



Figure-4 shows the Project Navigator interface

For the simulation, ISE incorporates a Xilinx version of ISIM Simulator. This powerful Simulator is capable of simulating functional VHDL before synthesis, or Simulating after the Implementation process for timing verification.

**Related Exercises:**

1. Write down VHDL code of the following expressions

X = AB + A’C + AD

Y = (A+B+D’) (A+B+C’)

1. Develop VHDL Code for following logic.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | X |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

1. Attach the output waveform of the related exercise question # 2.

**Review Questions:**

1. What is VHDL? Describe the advantages of VHDL code.

­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. List some EDA (Electronic Design Automation) tools available for circuit synthesis, implementation, and simulation using VHDL.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Define the Synthesis. Write down the steps for synthesize the design.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What is Implementation?

­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_