# Lab # 01;

## "Introduction to Verilog (HDL)"

## Lab Objectives:

The following are the Lab objectives

- Introduction to HDL
- Introduction to Verilog
- Installation of XILINX ISE Design Suite
- Terminologies in Verilog
- Basic project in ISE Design Suite

## Introduction:

### HDL (Hardware Descriptive Language):

Hardware Description Language (HDL) is a programming language that is used to describe the structure, behavior and timing of electronic circuits, and most commonly, digital logic circuits. HDLs are used for designing processors, motherboards, CPUs and various other Digital circuits. In addition to their use in circuit design, HDLs serve the purpose of simulating the circuit and verifying its response. Many HDLs are available, but the most popular HDLs so far are Verilog and VHDL.

### Verilog:

Verilog stands for verification logic. It is used to model and stimulate the Digital circuits Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs).

### Abstraction Levels in Verilog:

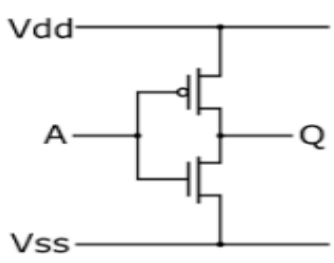
Following are the four different levels of abstraction which can be described by four different coding styles of Verilog language

1. Behavioral or Algorithmic level
2. Dataflow level
3. Gate level or Structural level
4. Switch level

**Switch Level:** Module is implemented in terms of switches, we can represent the entire circuit as a CMOS circuit.

Following sample is used for syntax demonstration:

```
module inverter(Q,A);

input A;
output Q;
supply1 vdd;
supply0 vss;
pmos p(Q,vdd,A);
nmos n(Q,vss,A);

endmodule
```



**Gate level:** Module is implemented in terms of gates, which is the lowest level of abstraction. Basic logic gates are available as predefined primitives. In the digital library of Verilog, these logic gates are already saved and can be used directly like and, or, xor, nand, nor, etc.

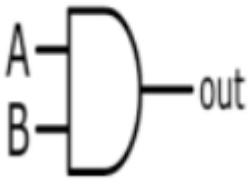
Following sample is used for syntax demonstration:

```
Primitive_name instance_name (output, inputs)
```

Primitive_name is logic gate name and instance_name is optional.

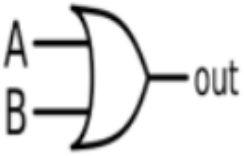
```
module and_gate_circuit (
  input a,
  input b,
  output out
);

  // Instantiate an AND gate primitive
  and and_gate (
    out,
    a,
    b
  );

endmodule
```

1. and G1(out,A,B);




2. nand G2(Y,A,B);


3. or G3(out,A,B);




**Data flow level:** Whenever RTL (Register transfer logic) is heard then the data flow level comes into action. At this level, the module is designed by specifying the data flow, like how the data is flowing in the circuit based on that the equation should be written. Signals are assigned by the data manipulating equations, design is implemented using continuous assignment i.e. assign statement is used in this level, the assignments present in it are concurrent in nature.

```
// Dataflow Verilog code for c = a + b

module add_two_bits (
  input a,
  input b,
  output c
);

assign c = a + b;

endmodule
```

**Behavioral level:** This is the highest level of abstraction provided by HDL, this level describes the behavior of the system. Different elements like functions, tasks, and blocks can be used, two important constructors are initial and always

The “always” keyword in Verilog is used to describe a procedural block that is executed repeatedly. The always block is triggered by any change to the signals in its sensitivity list.

always @(*): This line declares an always block. An always block is a procedural block that is executed repeatedly. The always block is triggered by any change to the signals in its sensitivity list.

```
// Behavioral Verilog code for c = a + b

module add_two_bits (
  input a,
  input b,
  output c
);

//The always block ensures that the sum is updated whenever either of
the inputs a or b changes.

always @(*) begin
  c <= a + b;
end

endmodule
```

**Common Terminologies:**

- FPGA: A field-programmable gate array is a chip that can be programmed to implement a digital circuit.
- Module: A module is a collection of Verilog code that describes a digital circuit.
- Signal: A signal is a variable that represents the state of a digital circuit.
- Port: A port is a signal that is used to connect a module to other modules or to the outside world.
- Always block: An always block is a procedural block that is executed repeatedly.
- Sensitivity list: The sensitivity list of an always block specifies the signals that trigger the always block to be executed.
- Primitive: A primitive is a pre-defined logic gate or other circuit element that can be used to implement the behavior of a digital circuit.
- Assign statement: An assign statement assigns a value to a signal.
- Reg: A reg is a signal that can change its value over time.
- Wire: A wire is a signal that cannot change its value over time.
- Synthesis: The process of converting a Verilog model into a gate level or switch level model.

## XILINX Design Suite:

The Xilinx Design Suite is a suite of software tools that can be used to design, implement, and verify digital circuits using Verilog. The Xilinx Design Suite includes a number of tools, including:

- Vivado IDE: A graphical user interface (GUI) that can be used to create and edit Verilog code.
- Synthesis tools: Tools that can be used to convert Verilog code into a gate level or switch level model.
- Place and route tools: Tools that can be used to place and route the gates in the gate level or switch level model onto an FPGA.

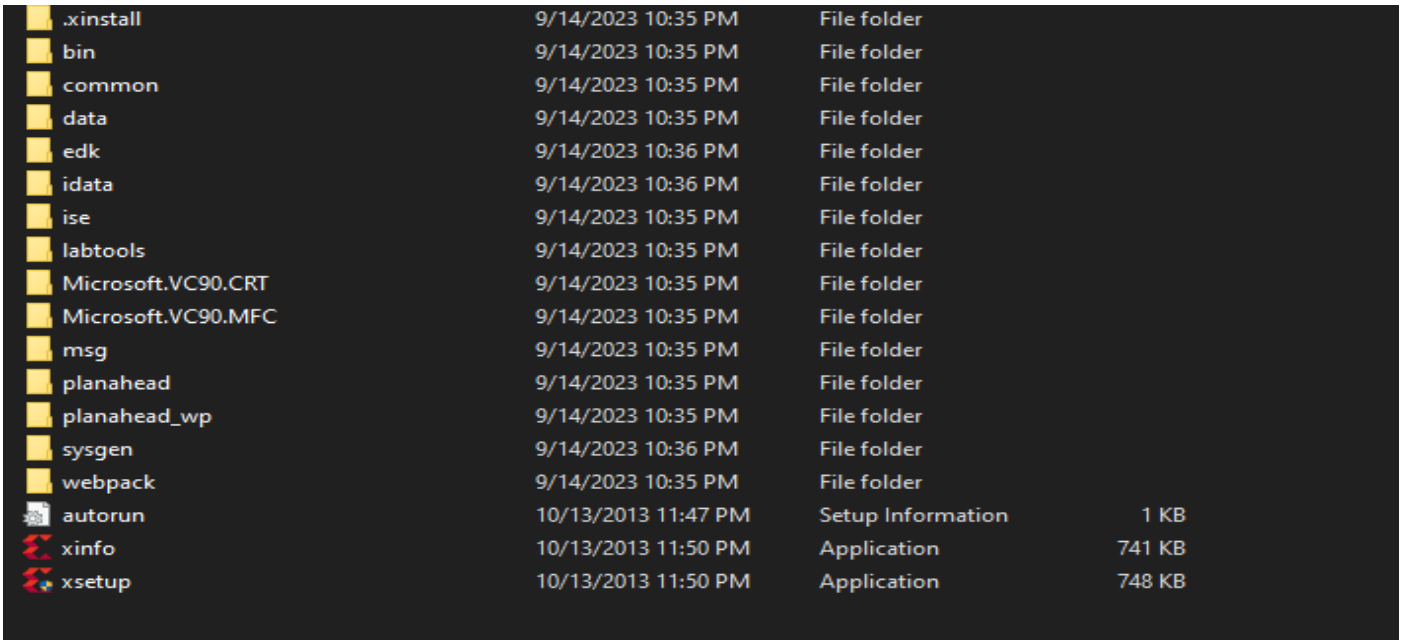
- Simulation tools: Tools that can be used to simulate the behavior of the digital circuit before it is implemented on an FPGA.

The Xilinx Design Suite is a powerful tool that can be used to design and implement a wide variety of digital circuits. It is used by engineers and hobbyists all over the world to create everything from simple logic gates to complex FPGA designs.

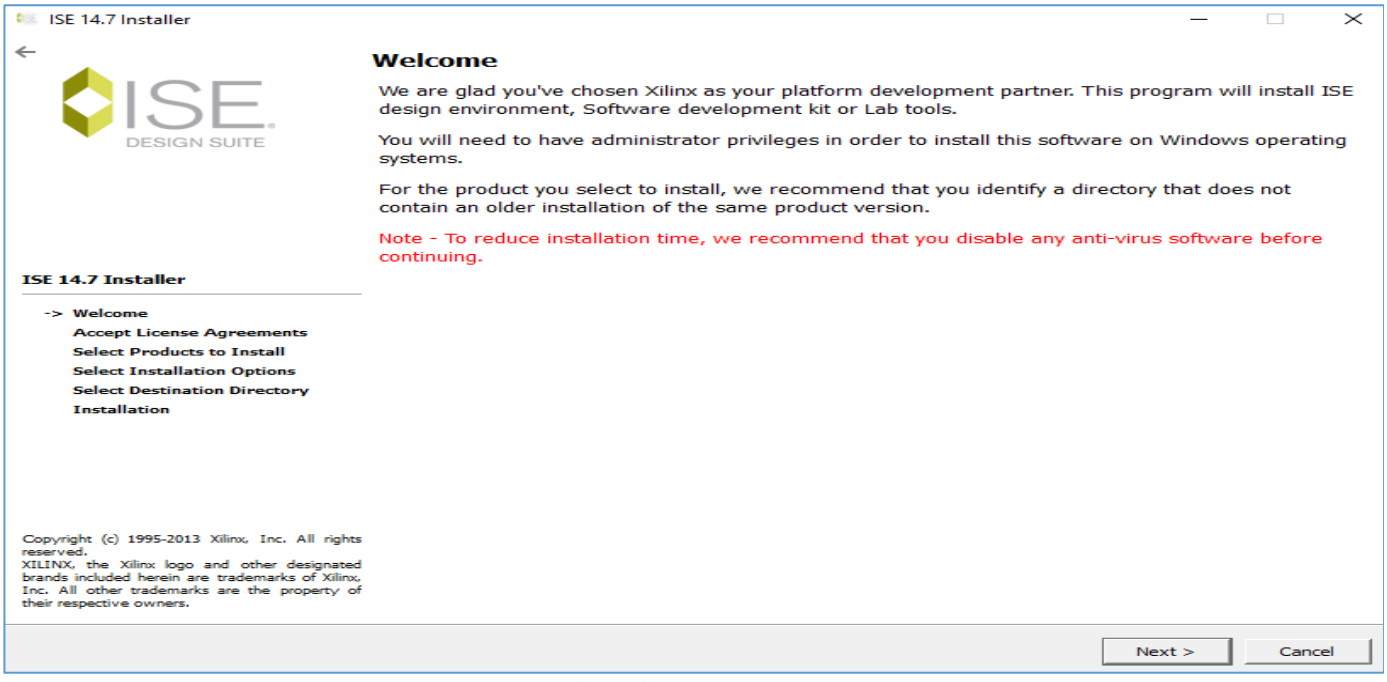
We will be using ISE Project navigator, The ISE Project Navigator manages and processes your design through several steps in the ISE design flow. These steps are Design Entry, Synthesis, Implementation, Simulation/Verification, and Device Configuration.

## Installation Process of XILINX Design Suite :

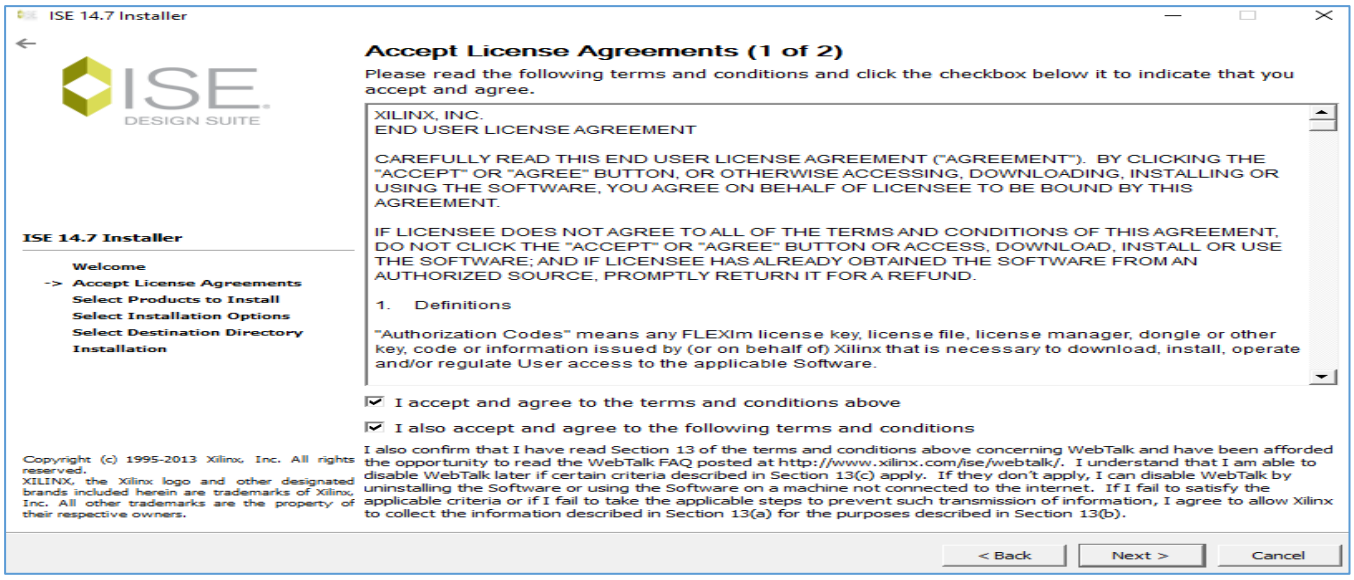
Download the XILINX 14.7 Setup from the internet , once you have downloaded it you will have the following



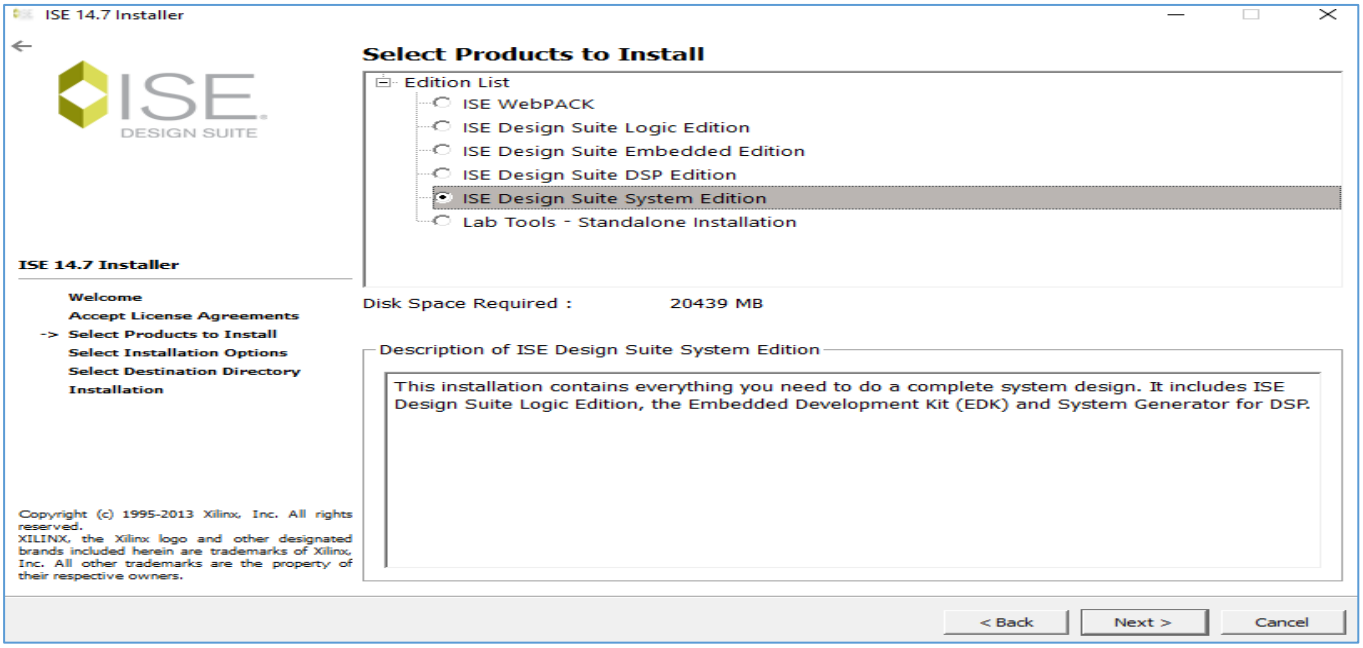

Select the xsetup file and run it as administrator



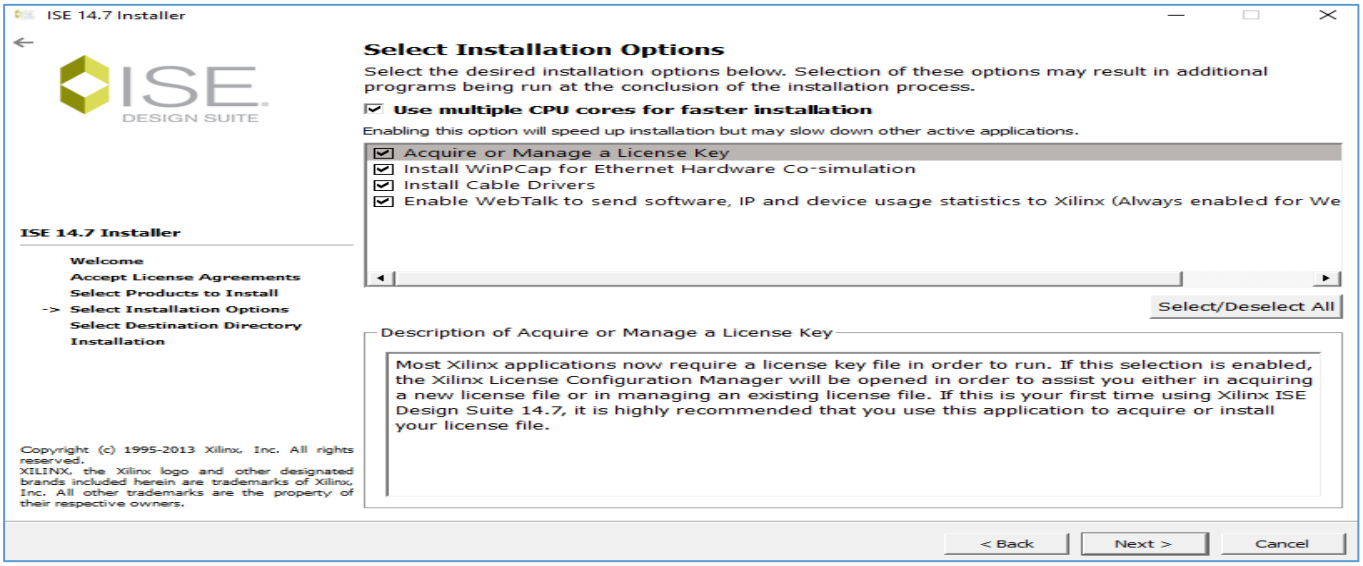

Select the next option



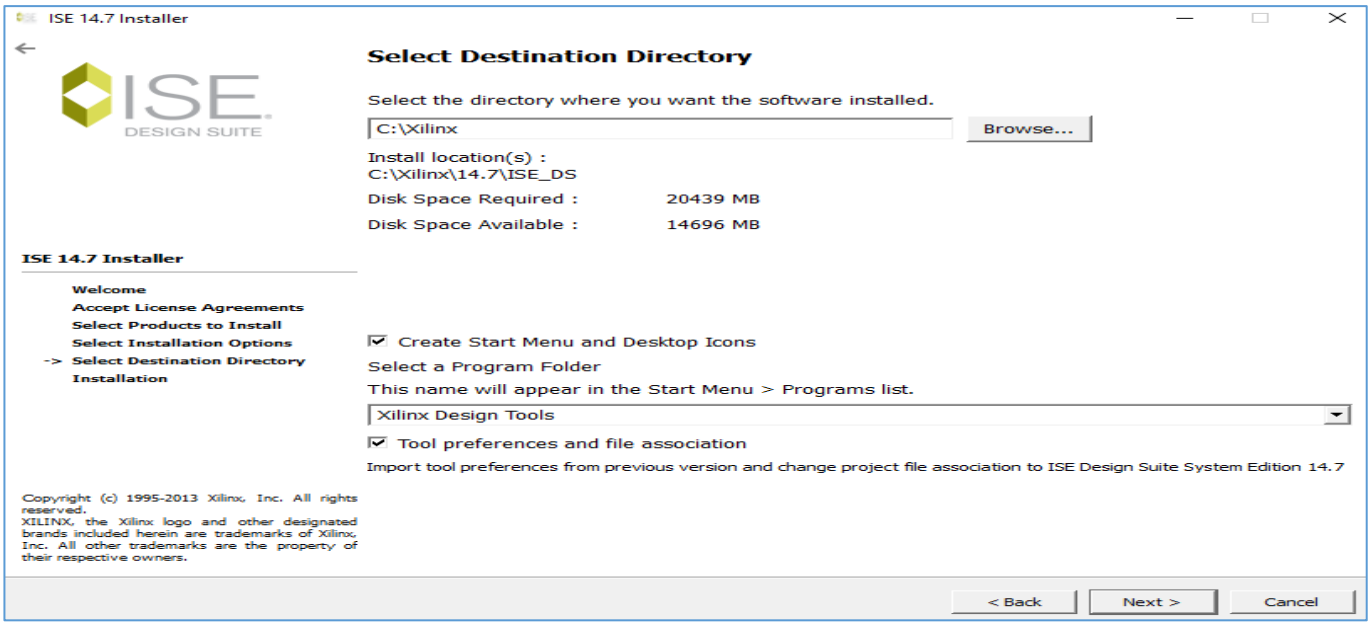

Accept license agreement and select Next



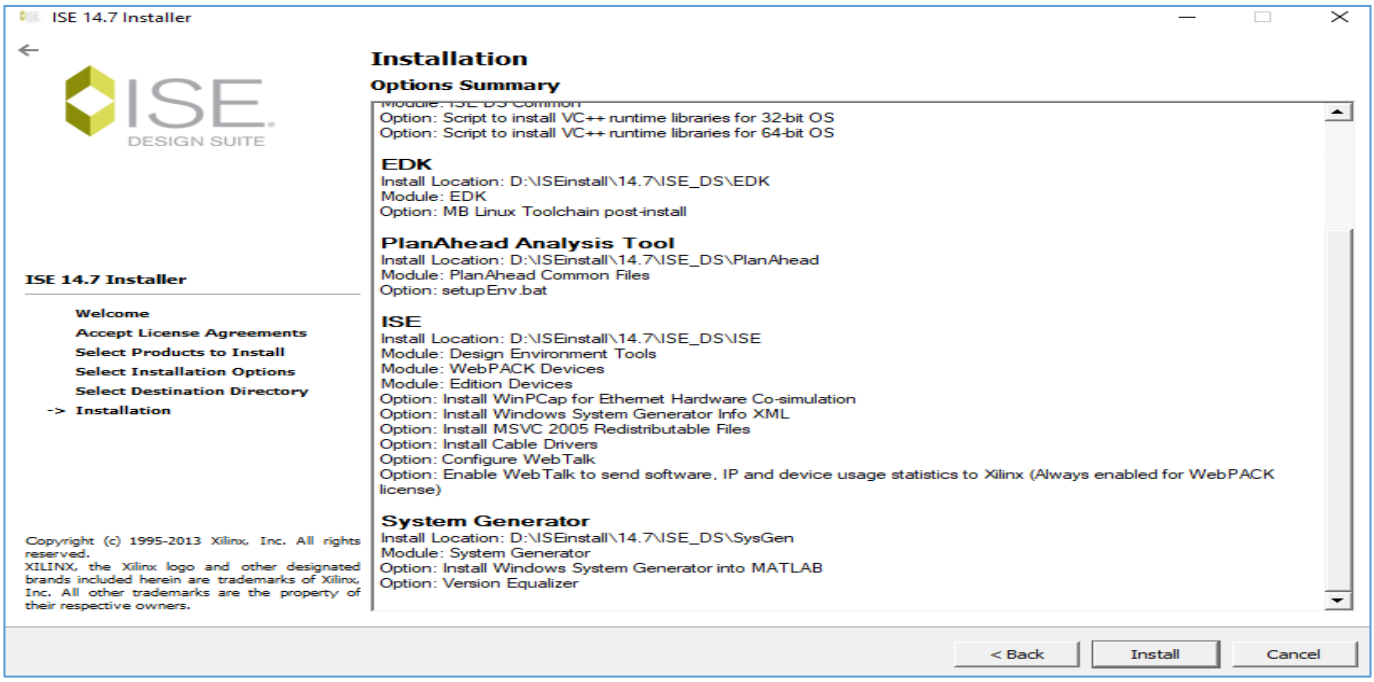

Select the ISE Design Suite System Edition , press Next



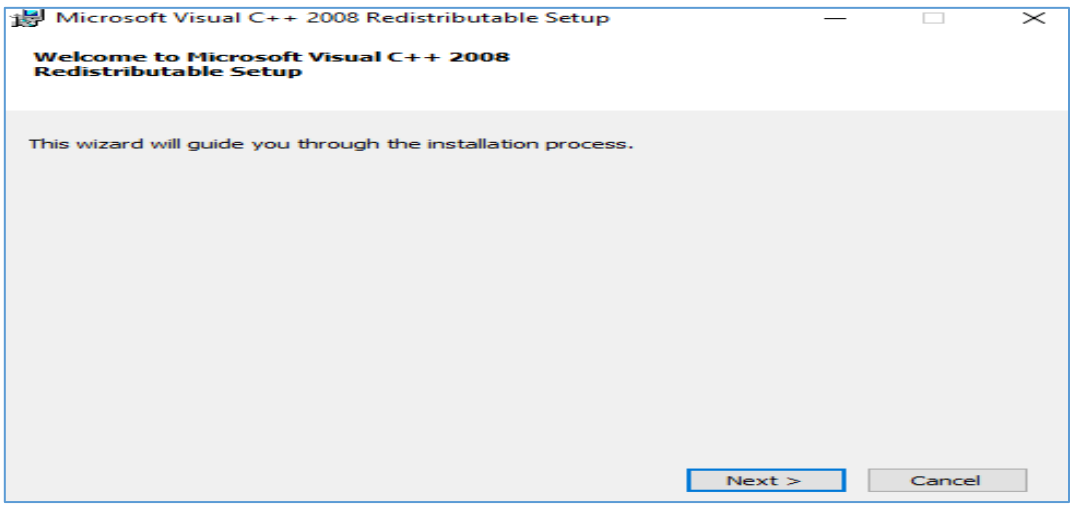

Select the Next option



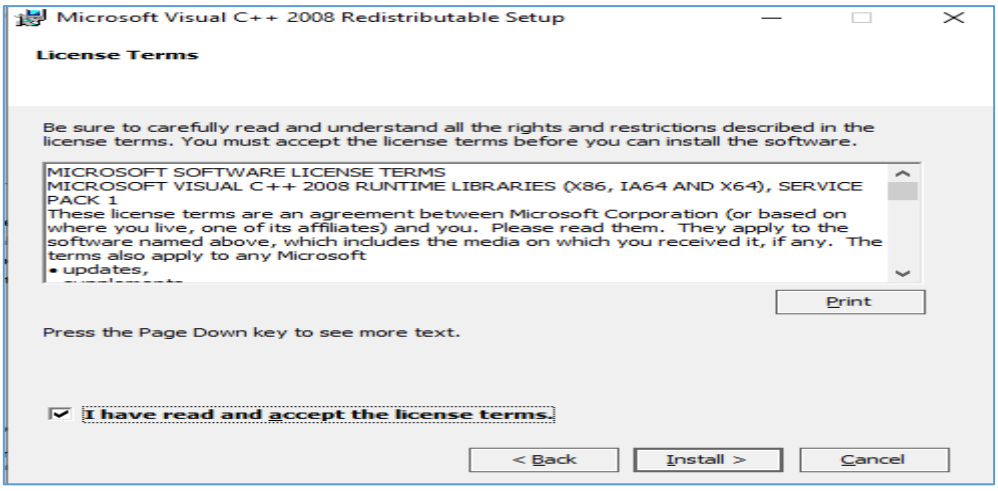

If you want to change the installation location you can use brows option , press Next



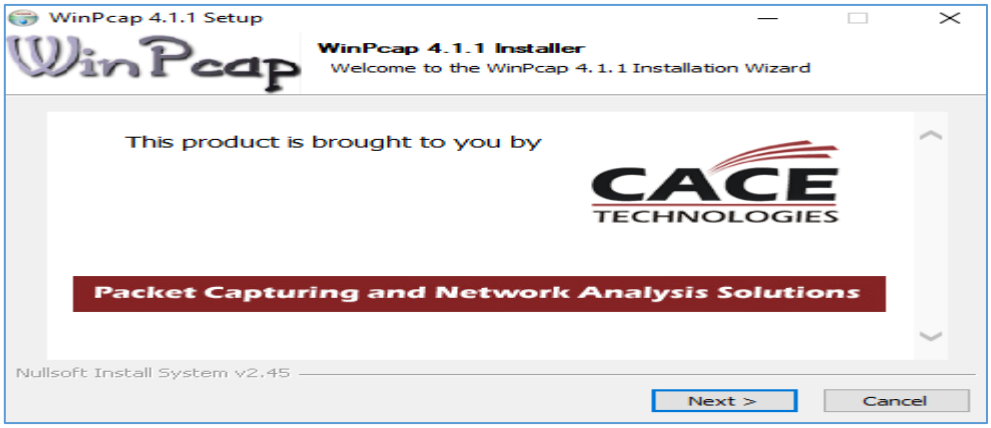

Press Install button , installation will begin



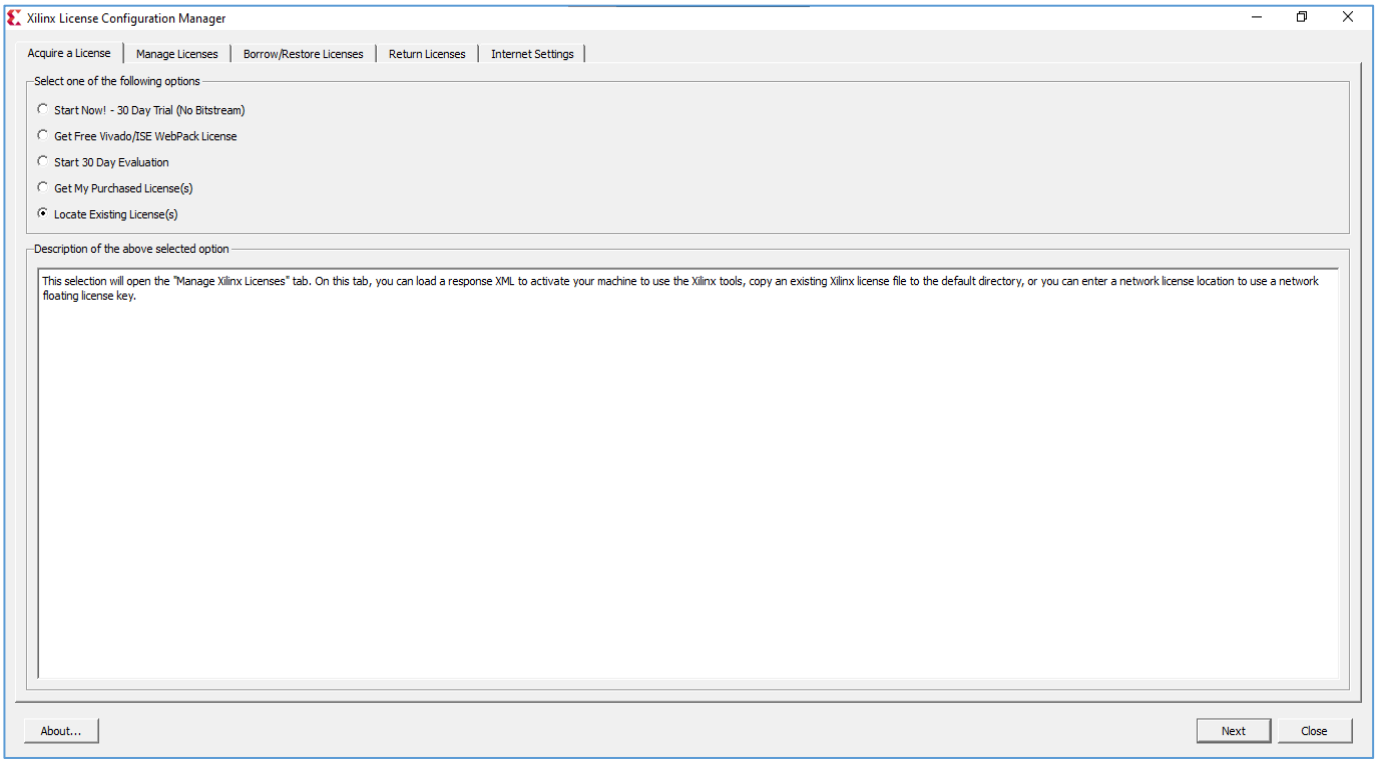

Press the Next button



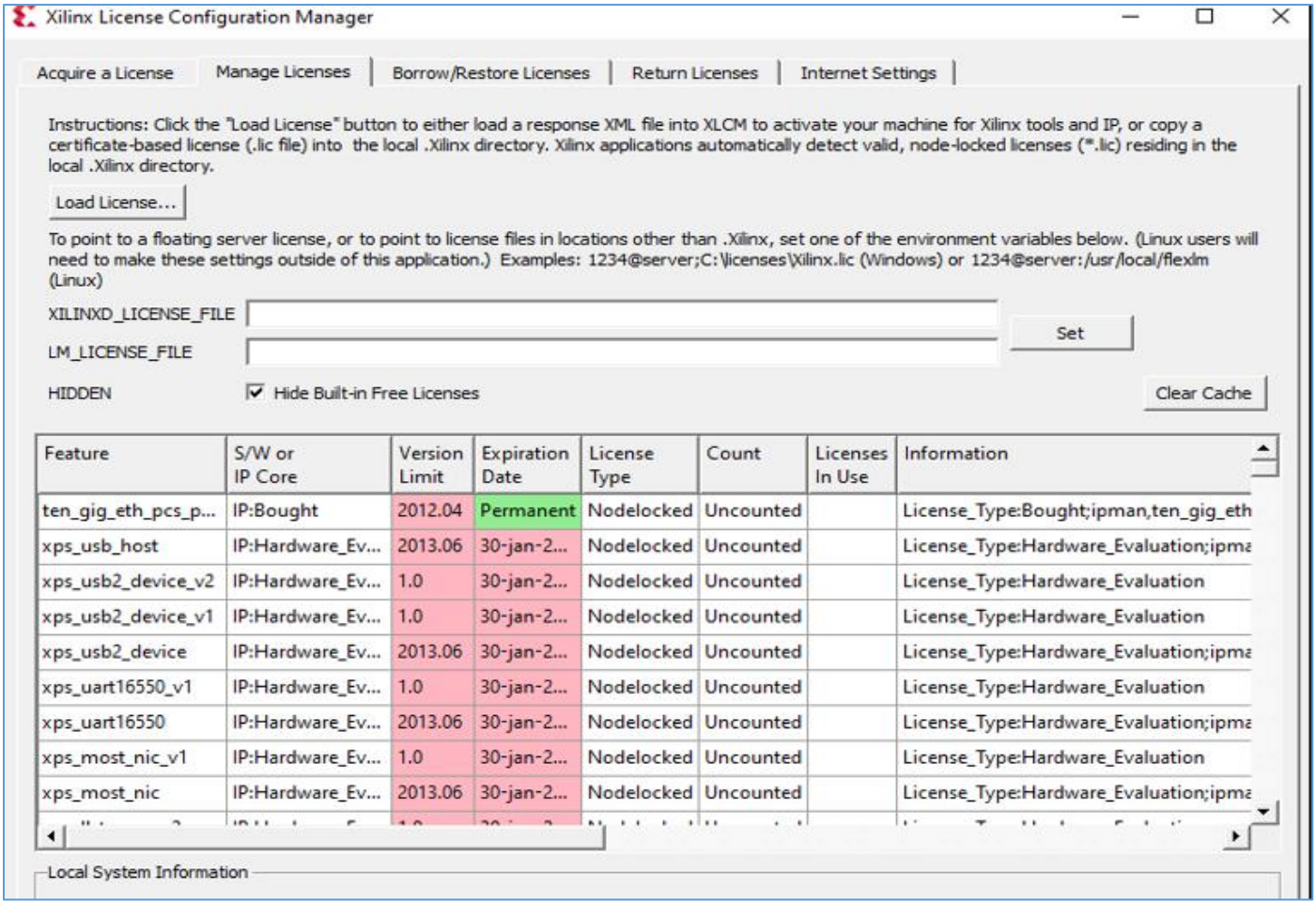

Accept and Install , installation will begin



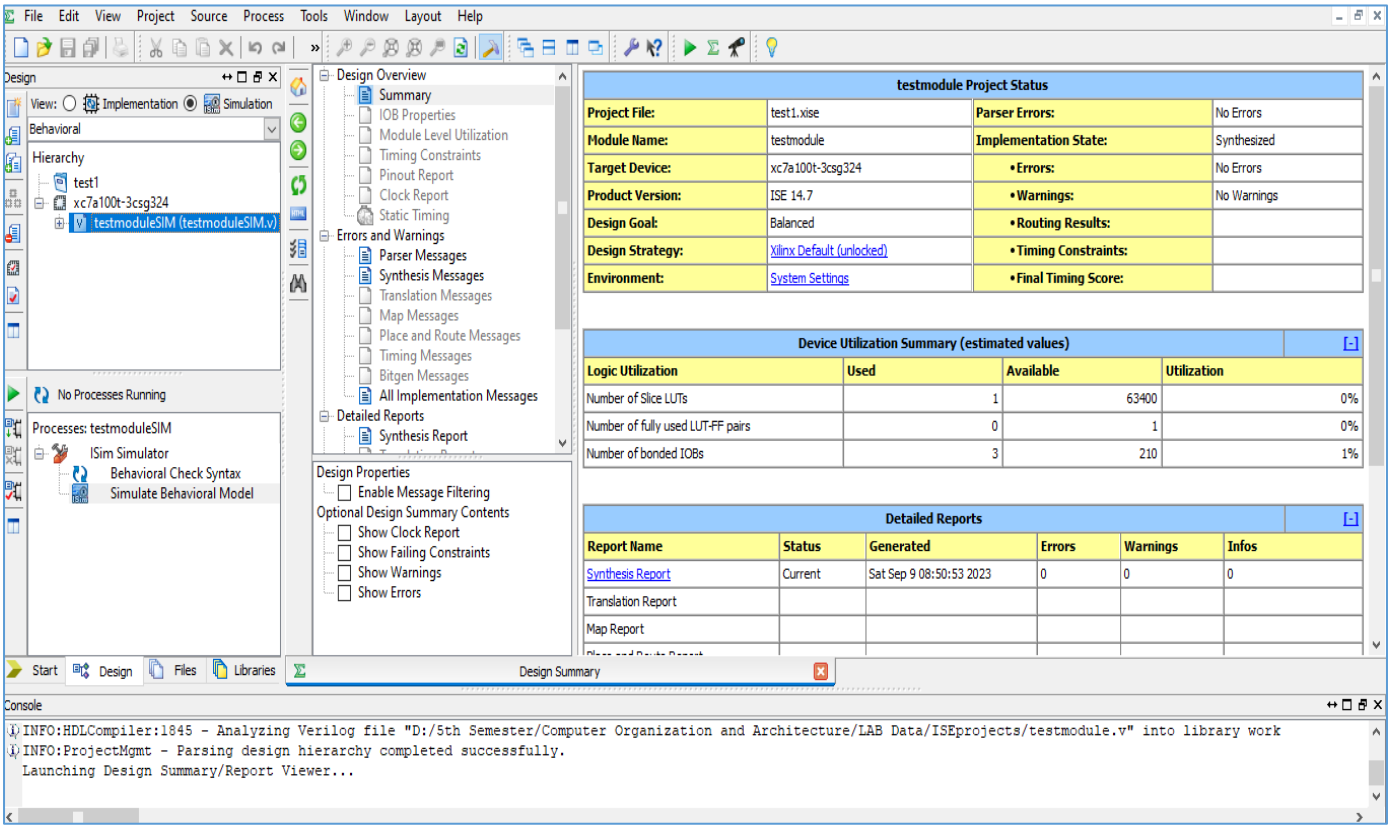

Press the Next button , installation will begin




Select the get my purchased license in order to add the license that you or your organization has purchased

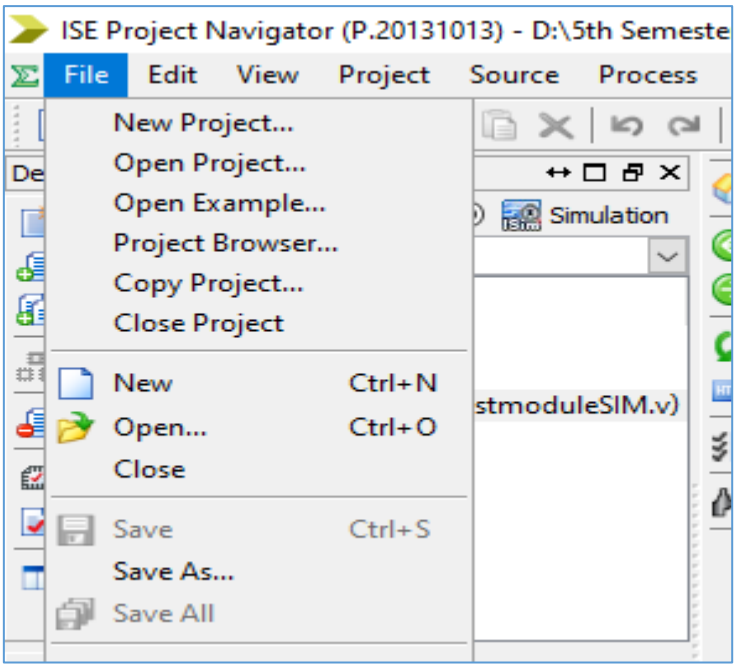



First locate the address of your license and then copy its address then ,In the blank section in front of XLINXD_LICENSE_FILE , paste the address of your license and press the Set button

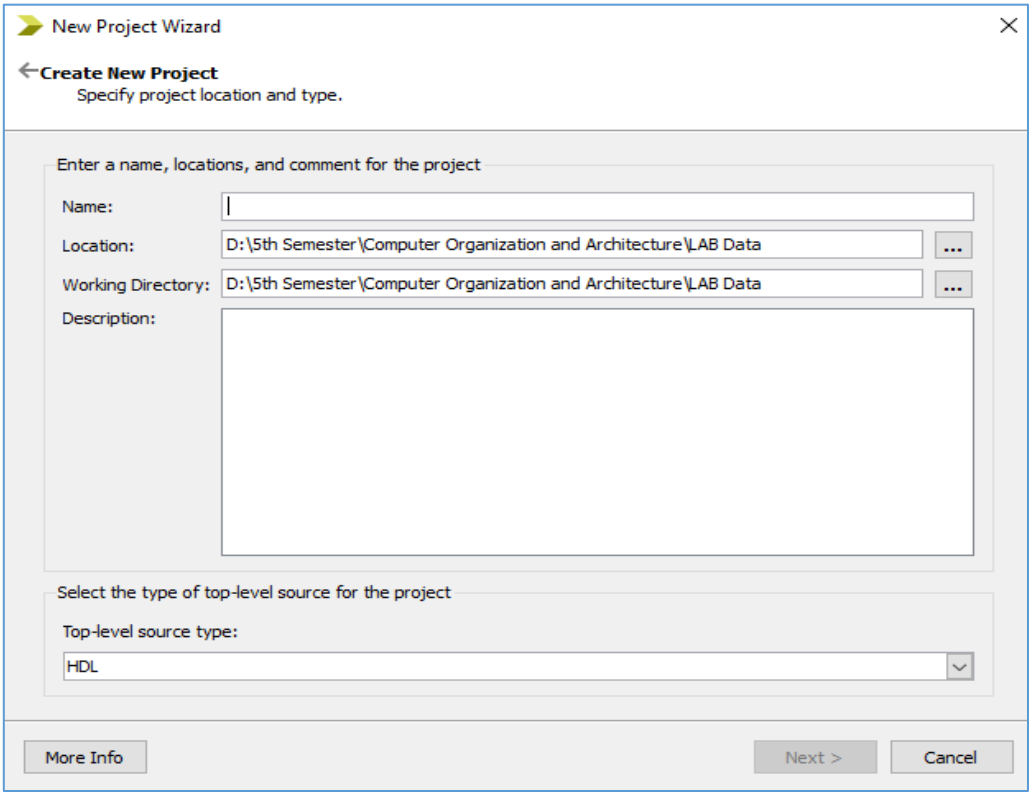



As you open the ISE Design Suite the first appearance should look like this as above

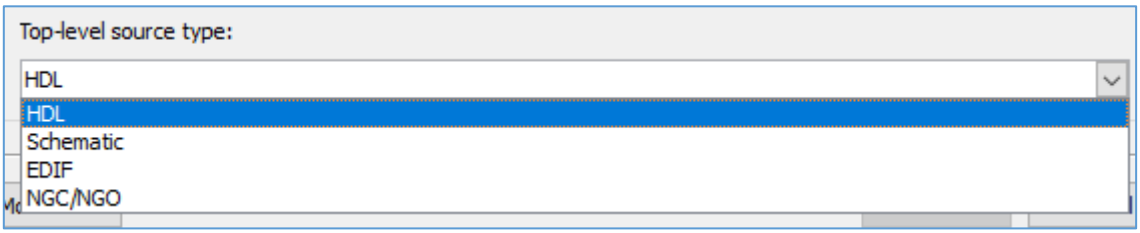
**Creating our first project ISE Project Navigator :**



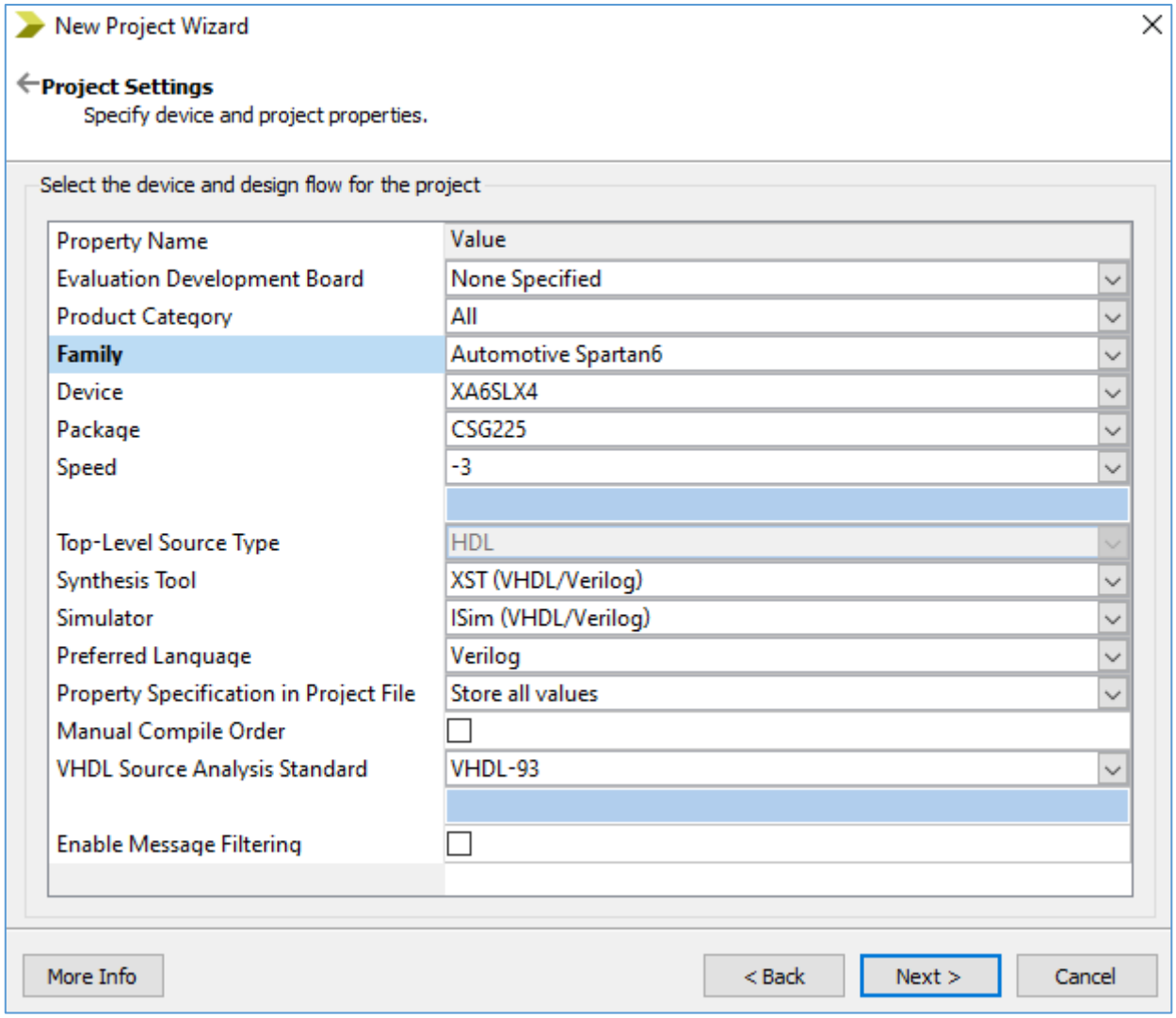

Click the File option in top left corner of the window, and then select the new project option



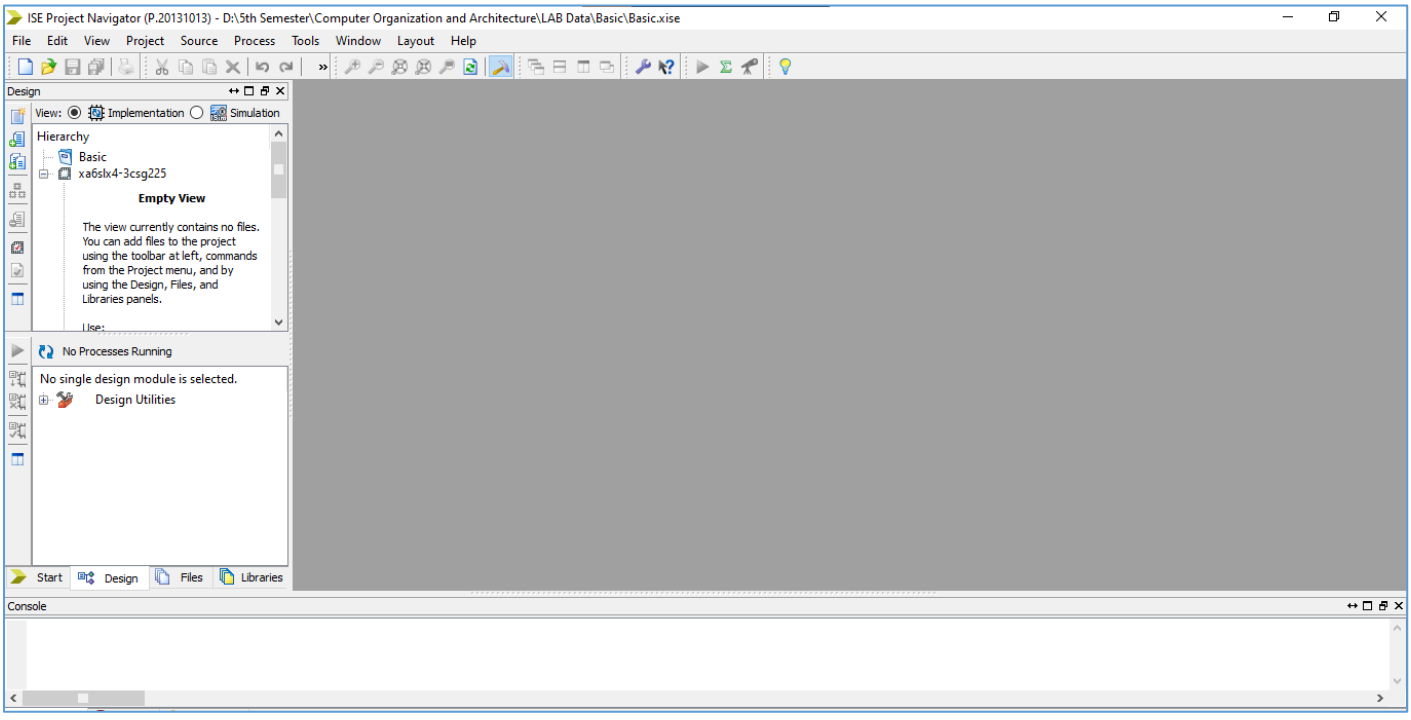

Give the name to your project, save it wherever you want to save




There will be multiple source types, we should start with HDL (Hardware Descriptive Language), then click next



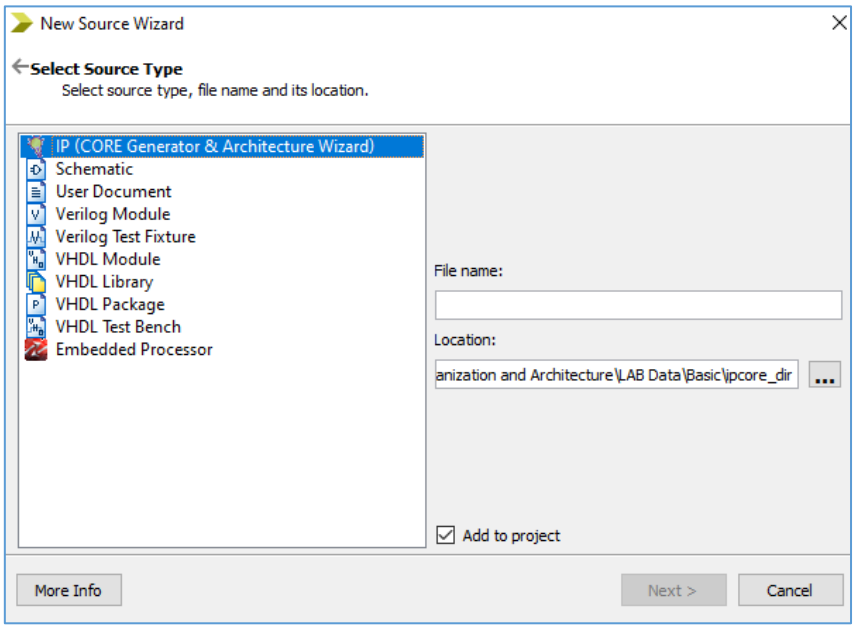

Select the device info according to you LAB availability, for now let's just start with these configuration for Basic project



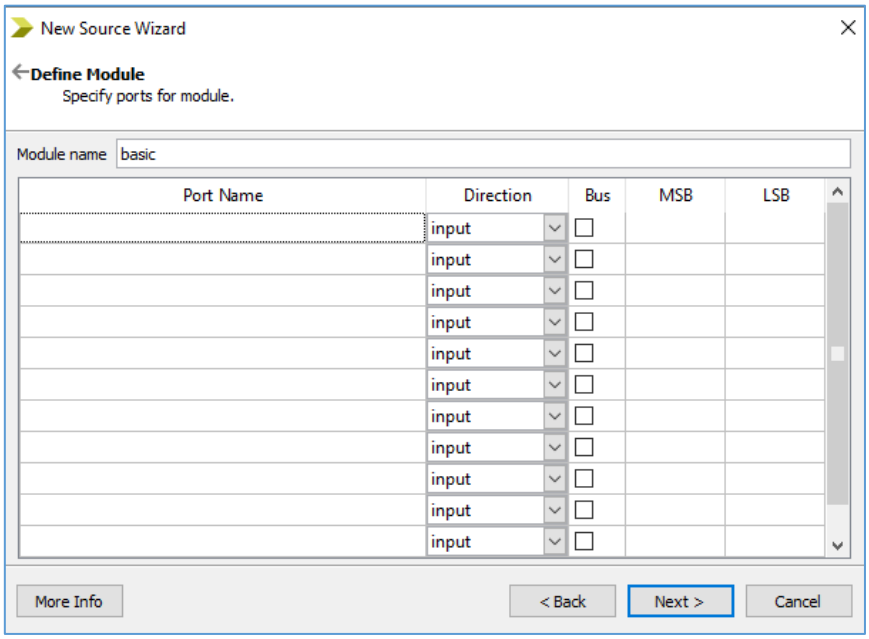

By clicking next button, we have successfully created our project now it's time to add some source files into it so we can implement our basic example

- Design Window: The Design Window shows the schematic or behavioral model of your digital circuit. You can use the Design Window to edit your circuit and to view the results of simulation and synthesis.
- Process Running Window: The Process Running Window shows the progress of the current build or synthesis process. It also shows any errors or warnings that occur during the build or synthesis process.
- Console Window: The Console Window shows the output of the Xilinx ISE tools. This includes messages about the build or synthesis process, as well as information about the resources that are used by your circuit.

Select the option of project from above task bar and add a New source, you will see then the following options




There are different options depends on your needs select the one source type as for this example we want to code in Verilog, so we will select Verilog module, give it the name of your choice

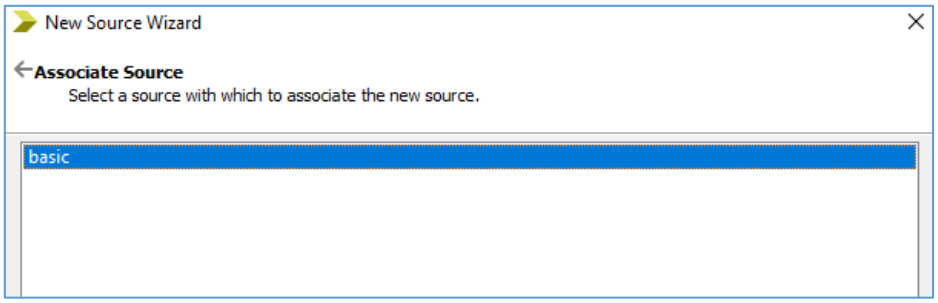



You can select input, output ports and their names also their bit size here and also in code as well, so it's up to you both approaches can be followed in this example we will select it within the code, select next button and then finish

```
1  `timescale 1ns / 1ps
2  //comment
3  module basic(variable1,variable2,variable3);
4      input variable1;
5      input variable2;
6      output variable3;
7      assign variable3 = variable1 + variable2;
8
9  endmodule
10
```

- Line 1 code: This line sets the timescale for the simulation. The timescale specifies how many nanoseconds are represented by each time unit in the simulation. In this case, each time unit represents 1 nanosecond.
- Line 2 code: it shows how we can add comments into our code
- Line 3 code: This line declares a new module called basic. The module has three ports: variable1, variable2, and variable3. The variable1 and variable2 ports are inputs, and the variable3 port is an output.
- Line 4, 5, 6 codes: this line declares the three ports for the basic module. The variable1 and variable2 ports are declared as input port, the variable3 port is declared as an output port.
- Line 7 code: This line assigns the sum of variable1 and variable2 to the variable3 port. The assign statement is used to assign values to signals.
- Line 9 code: This line terminates the basic module.

Now, Select the Project Tab again and add new source, “Verilog Test Fixture” A Verilog text fixture in sources is a Verilog file that contains a test bench for a Verilog module. The test bench can be used to verify the functionality of the module.

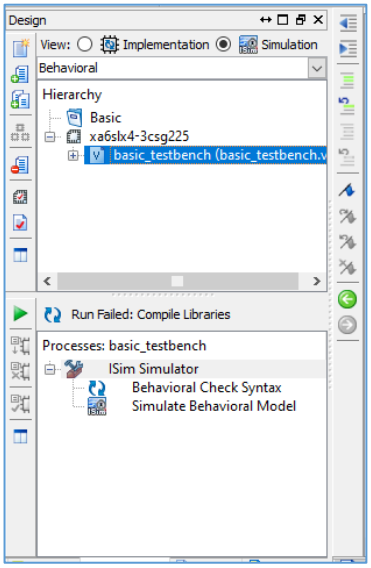



Include any file of your project that you want to test, we will add the basic module which was created in the previous step , select Next and finish in this software the code for test bench is automatically created

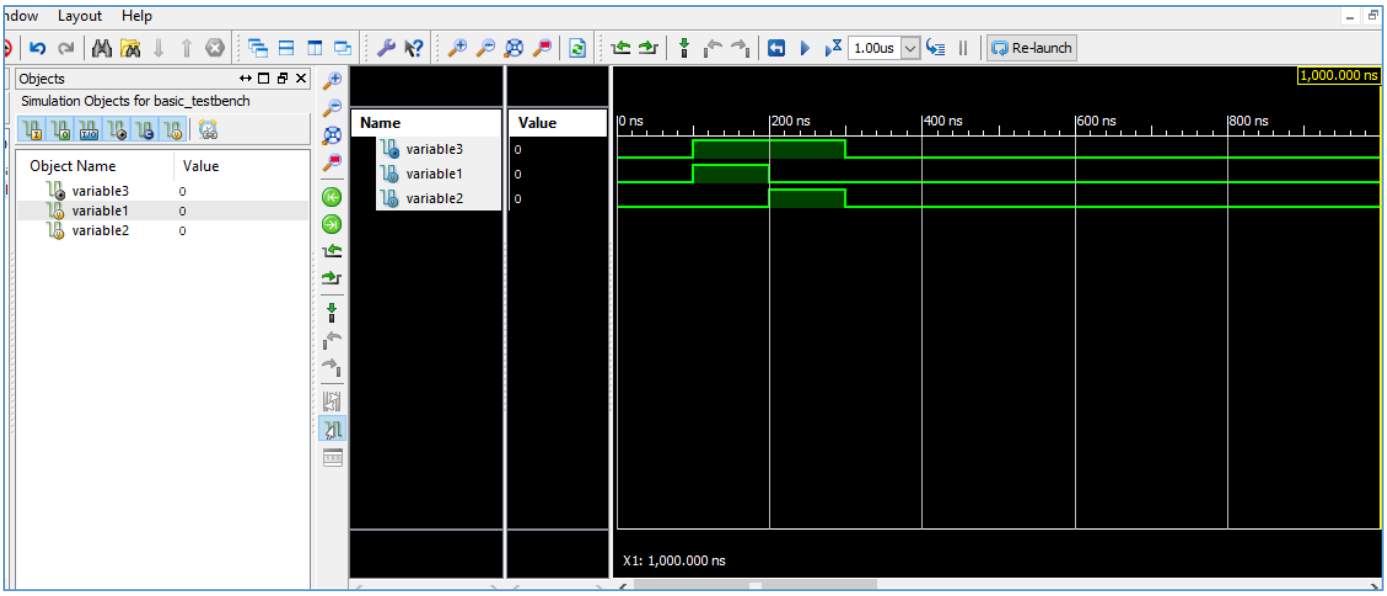
```
1  `timescale 1ns / 1ps
2  module basic_testbench;
3      // Inputs
4      reg variable1;
5      reg variable2;
6
7      // Outputs
8      wire variable3;
9
10     // Instantiate the Unit Under Test (UUT)
11     basic uut (
12         .variable1(variable1),
13         .variable2(variable2),
14         .variable3(variable3)
15     );
16
```

```
16
17      initial begin
18          // Initialize Inputs
19          variable1 = 0;
20          variable2 = 0;
21
22          // Wait 100 ns for global reset to finish
23          #100;
24          // Add stimulus here
25      end
26  endmodule
```

This is the code for test bench you can add or adjust the delays, which are represented by #100 in this code, you can also change the values for each variable after some delays to test the functionality of our system , we will re adjust the values such that variable1 = 1 for first interval then variable2 = 1 for second , for rest of the time both 0




From the design window(switch to simulation mode) select the test bench file and then in running process window select “behavioral check syntax” and then select the Simulate Behavioral Model




This is the simulation of our code which shows the functionality , for first 100ns both values were 0 so sum = 0, then var1 = 1 , var2 = 0 for next 100ns sum = 1, and then var1 = 0 and var2 = 1 for next 100 ns so sum = 0 , then for rest of the time both values were 0 so sum = 0 .

This was a basic code , test bench and simulation for adding 1 bit number