

**Eastern Mediterranean University**  
**Computer Engineering Department**  
**CMSE 222 Introduction to Computer Organization**

**Lab. 5**

**Single Clock Data Path for 16-bit R-type Instructions in  
ALTERA QUARTUS VHDL Environment**

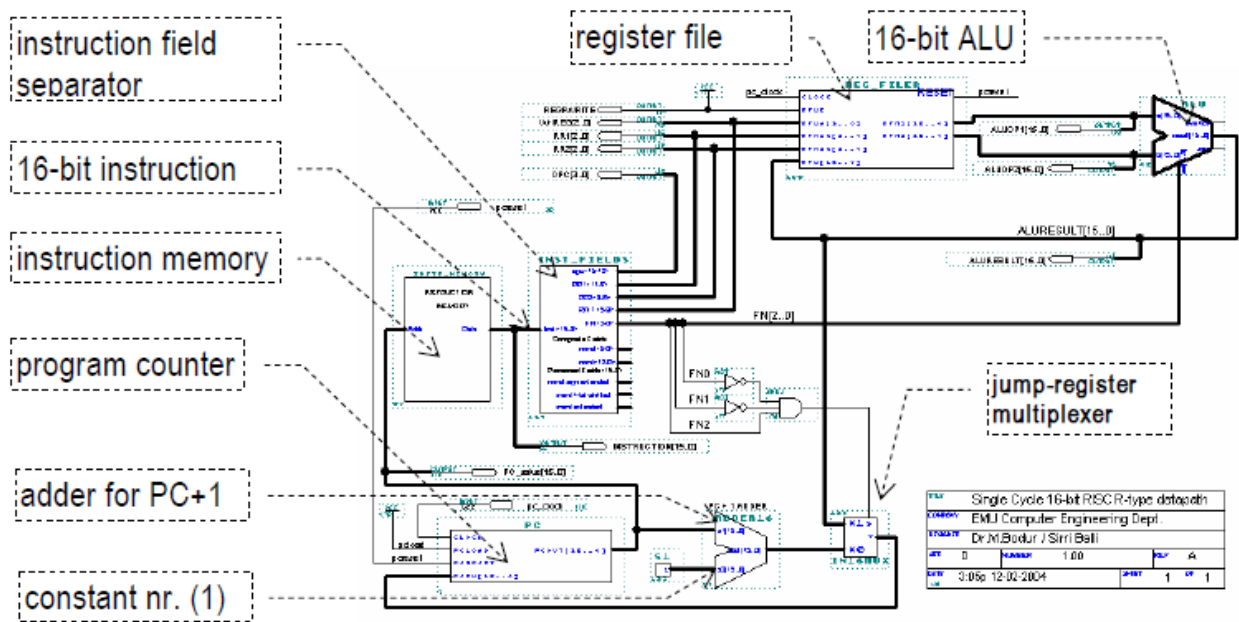
**1. Objective:**

To familiarize with the Single-Clock VHDL implementation for a set of 16-bit R-type instructions, and to measure the response time of several building components in a RISC datapath.

**1. Introduction**

You have already worked to construct a simple 16-bit instruction set for your project/homework. A typical simple 16-bit instruction set and a corresponding R-type datapath is provided in this experiment. You will be asked to take several measurements that can lead you to a conclusion about the typical speed constraints, and possible improvements of the similar circuits.

In this experiment, we will focus only on the R-type instructions, so that we can observe typical properties of some of the basic building blocks such as the Program-Counter, the Instruction-Memory, the Register-File, and the Adder for updating the Program-Counter.



The complete set of the 16-bit instructions are summarized in the following tables.

Opcode	Read1 reg	Read2 reg	Write reg	Function code
4-bits	3-bits	3-bits	3-bits	3-bits

**Table 1-2 Function Codes for the R-type Instructions**

FNCODE	.00	.01	.11	.10
0..	And	Or	Xor	Add
1..	Jr	ShrA	Slt	Sub

Table 1-3 Representation of R-type Instructions with opcodes

instruction	Opcode	Read-reg1	Read-reg2	Write-reg	Fn code
And	0000	XXX	XXX	XXX	000
Or	0000	XXX	XXX	XXX	001
Add	0000	XXX	XXX	XXX	010
Xor	0000	XXX	XXX	XXX	011
Jump-Register	0000	XXX	000	XXX	100
Shift-right-Arith	0000	XXX	XXX	XXX	101
Subtraction	0000	XXX	XXX	XXX	110
Set less than	0000	XXX	XXX	XXX	111

## 2. QUARTUS and VHDL Code

In this section you should learn how to implement a VHDL code for single cycle data path. Before we go any further, let's define some of the terms that we use throughout the book.

**Entity:** All designs are expressed in terms of entities. An entity is the most basic building block in a design. The uppermost level of the design is the top-level entity. If the design is hierarchical, then the top-level description will have lower-level descriptions contained in it. These lower-level descriptions will be lower-level entities contained in the top-level entity description.

**Architecture:** All entities that can be simulated have an architecture description. The architecture describes the behavior of the entity. A single entity can have multiple architectures. One architecture might be behavioral while another might be a structural description of the design.

**Configuration:** A configuration statement is used to bind a component instance to an entity-architecture pair. A configuration can be considered like a parts list for a

design. It describes which behavior to use for each entity, much like a parts list describes which part to use for each part in the design.

**Package:** A package is a collection of commonly used data types and subprograms used in a design. Think of a package as a toolbox that contains tools used to build designs.

**Driver:** This is a source on a signal. If a signal is driven by two sources, then when both sources are active, the signal will have two drivers.

**Bus:** The term “bus” usually brings to mind a group of signals or a particular method of communication used in the design of hardware. In VHDL, a bus is a special kind of signal that may have its drivers turned off.

**Attribute:** An attribute is data that are attached to VHDL objects or predefined data about VHDL objects. Examples are the current drive capability of a buffer or the maximum operating temperature of the device.

**Generic:** A generic is VHDL’s term for a parameter that passes information to an entity. For instance, if an entity is a gate level model with a rise and a fall delay, values for the rise and fall delays could be passed into the entity with generics.

**Process:** A process is the basic unit of execution in VHDL. All operations that are performed in a simulation of a

Thus, let us get our feet wet and look at VHDL structure briefly as follow:

```

ENTITY entity_name IS
    GENERIC (
        generic_1_name      : generic_1_type;
        generic_2_name      : generic_2_type;
        generic_n_name      : generic_n_type
    );
    PORT (
        port_1_name      : port_1_dir port_1_type;
        port_2_name      : port_2_dir port_2_type;
        port_n_name      : port_n_dir port_n_type
    );
END entity_name;

```

```

IN      Input port
OUT     Output port
INOUT   Bidirectional port
BUFFER  Buffered output port

```

```

ARCHITECTURE architecture_name OF entity_name IS
BEGIN
    -- Insert VHDL statements to assign outputs to
    -- each of the output signals defined in the
    -- entity declaration.
END architecture_name;

```

This is a simple example of VHDL code that just work as buffer and send input to output after 10 ns:

- Consider the following VHDL code:

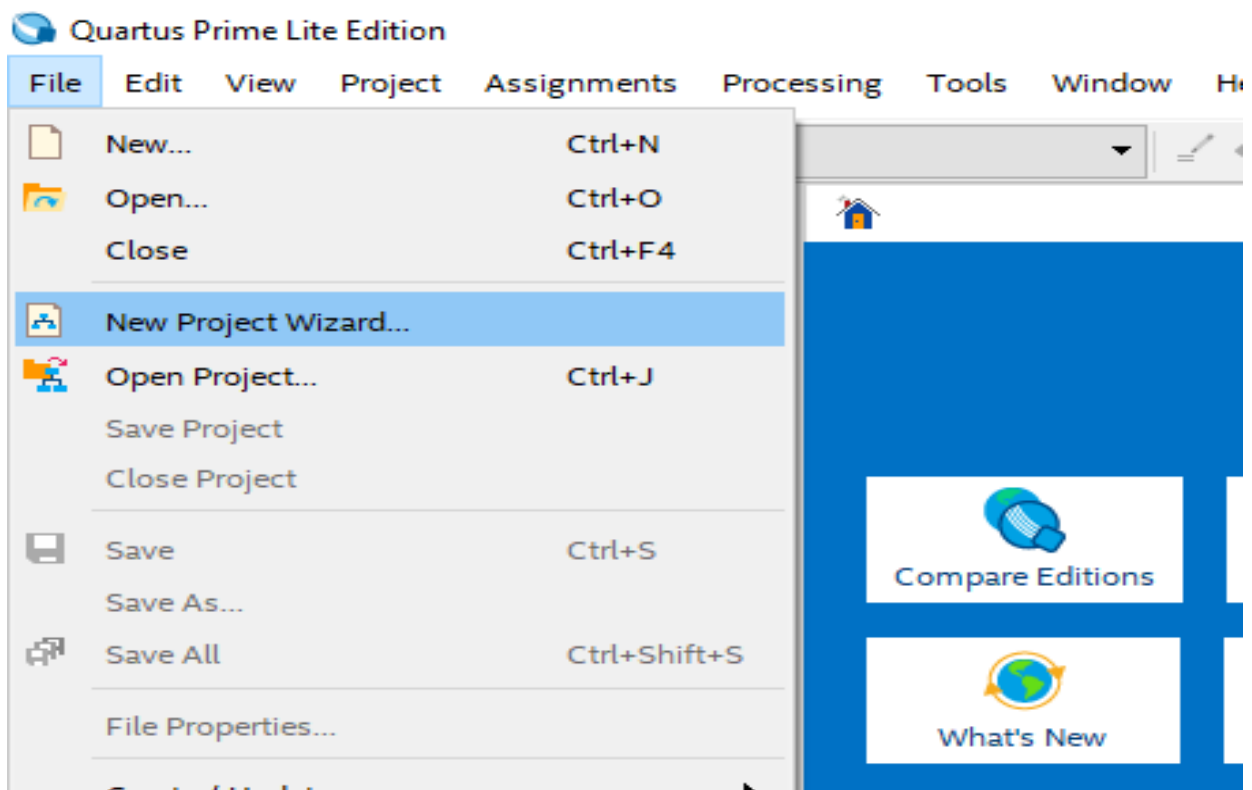
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY simple_buffer IS
    PORT (   din      : IN      std_logic;
           dout      : OUT     std_logic );
END simple_buffer;

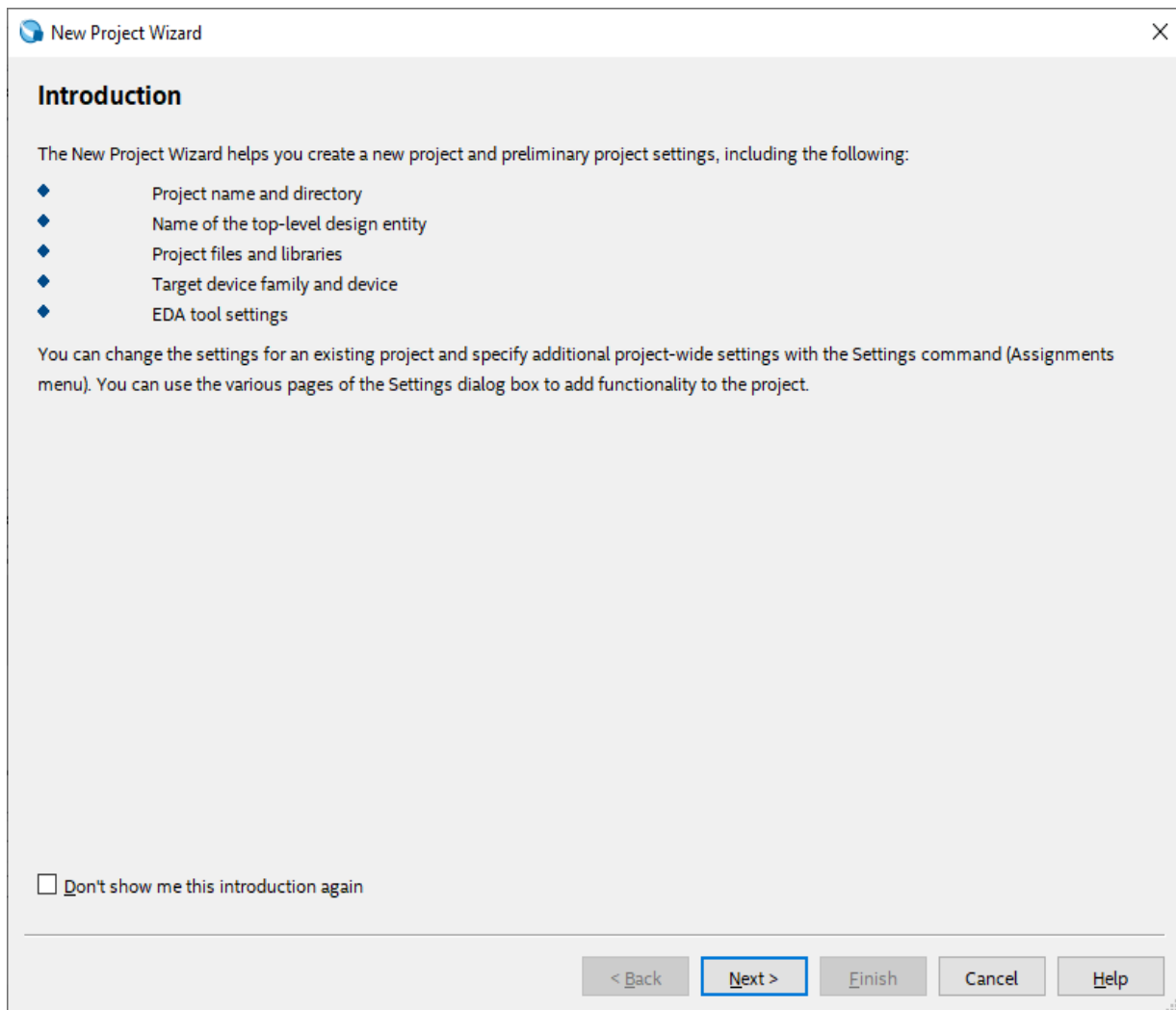
ARCHITECTURE behavioural1 OF simple_buffer IS
BEGIN
    dout <= din AFTER 10 ns;
END behavioural1;
```

### First simple program using VHDL:

Step 1. Create a Project: Every new design you make in the Quartus program must be under a project name. After opening the program, to create a project, click on:  
[File] -> [New Project Wizard]



After this stage you will see the following project builder screen.

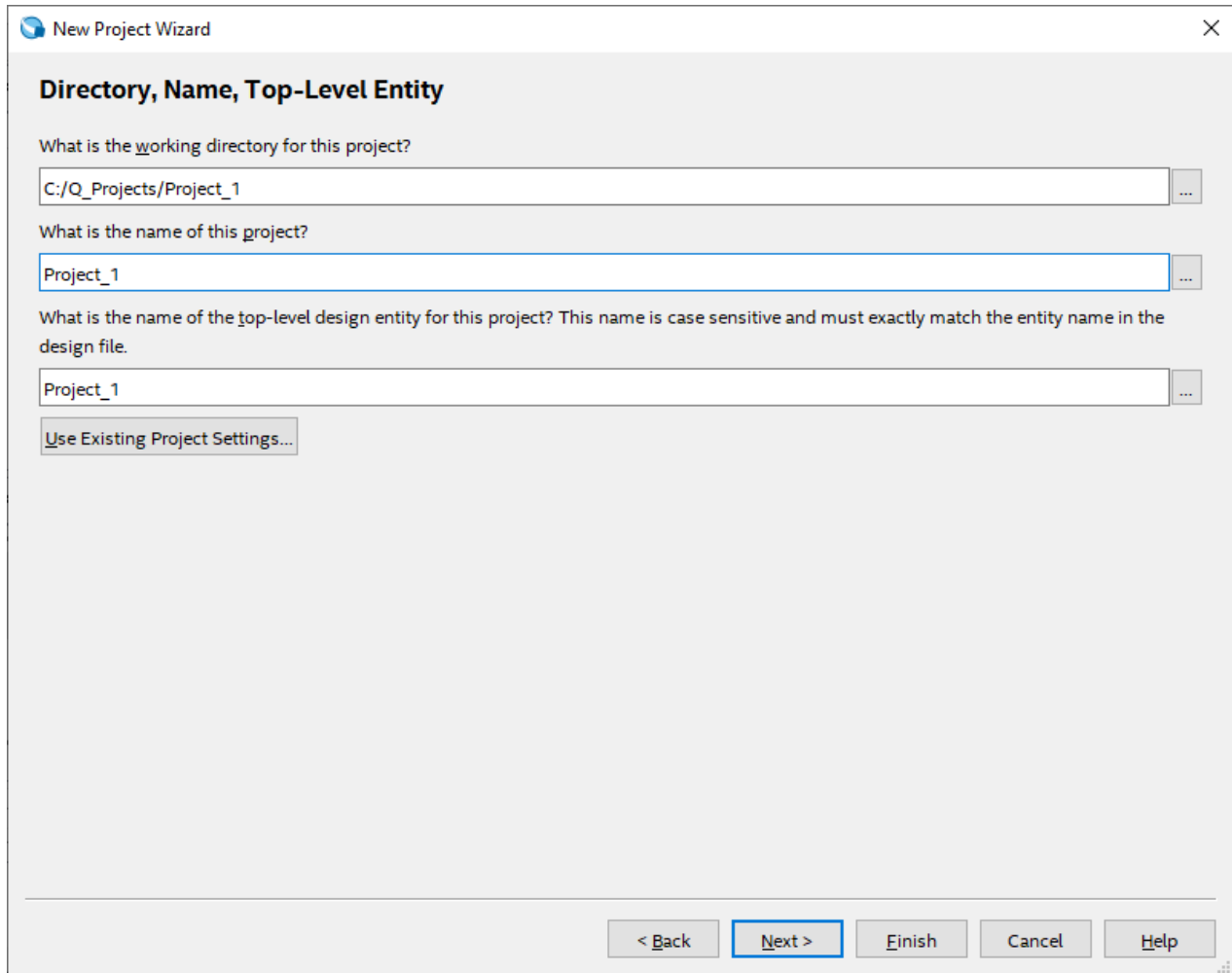


This screen is the home screen of the project builder. Here you will see which stages of your project are formed and what you can organize.

Here, press [Next >] button to continue.

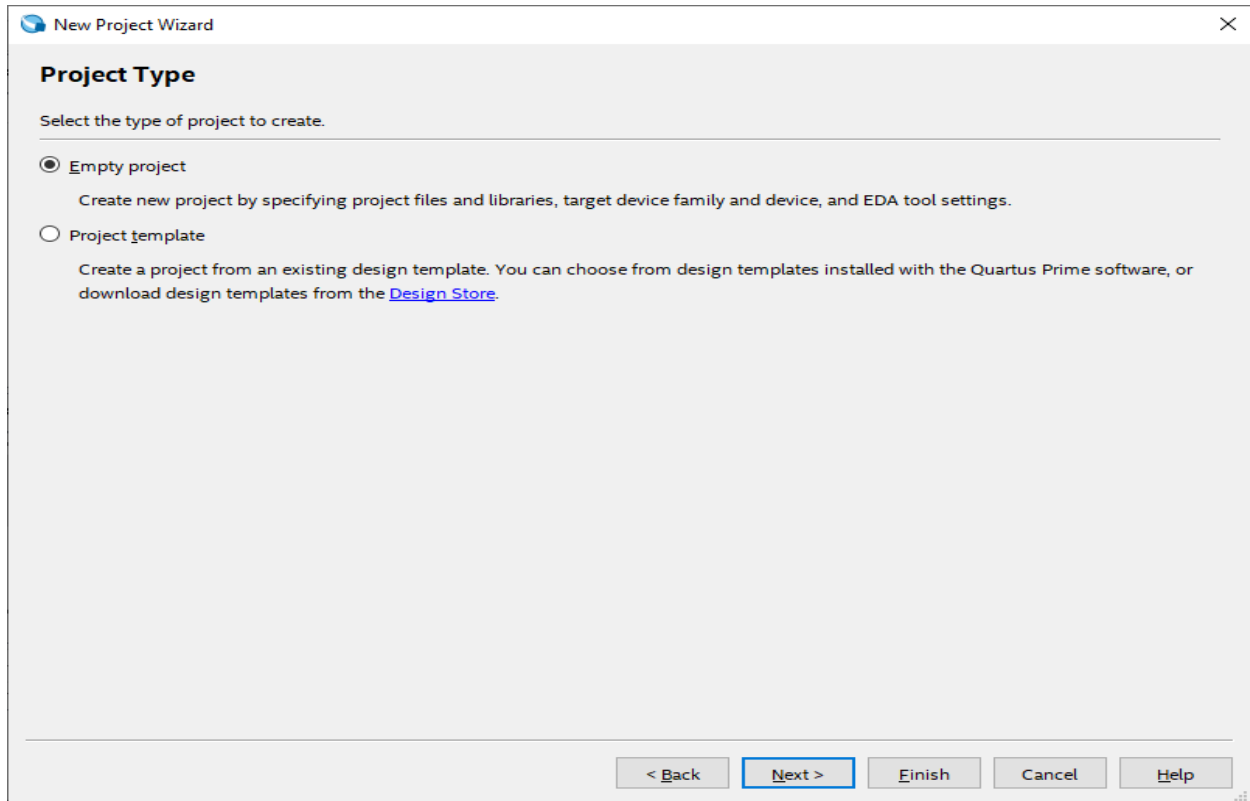
On the new screen, you will see the input sections that ask us to enter the directory and the name of the related project. Please fill in the appropriate information here.

(Project names for the Quartus program is very important.)

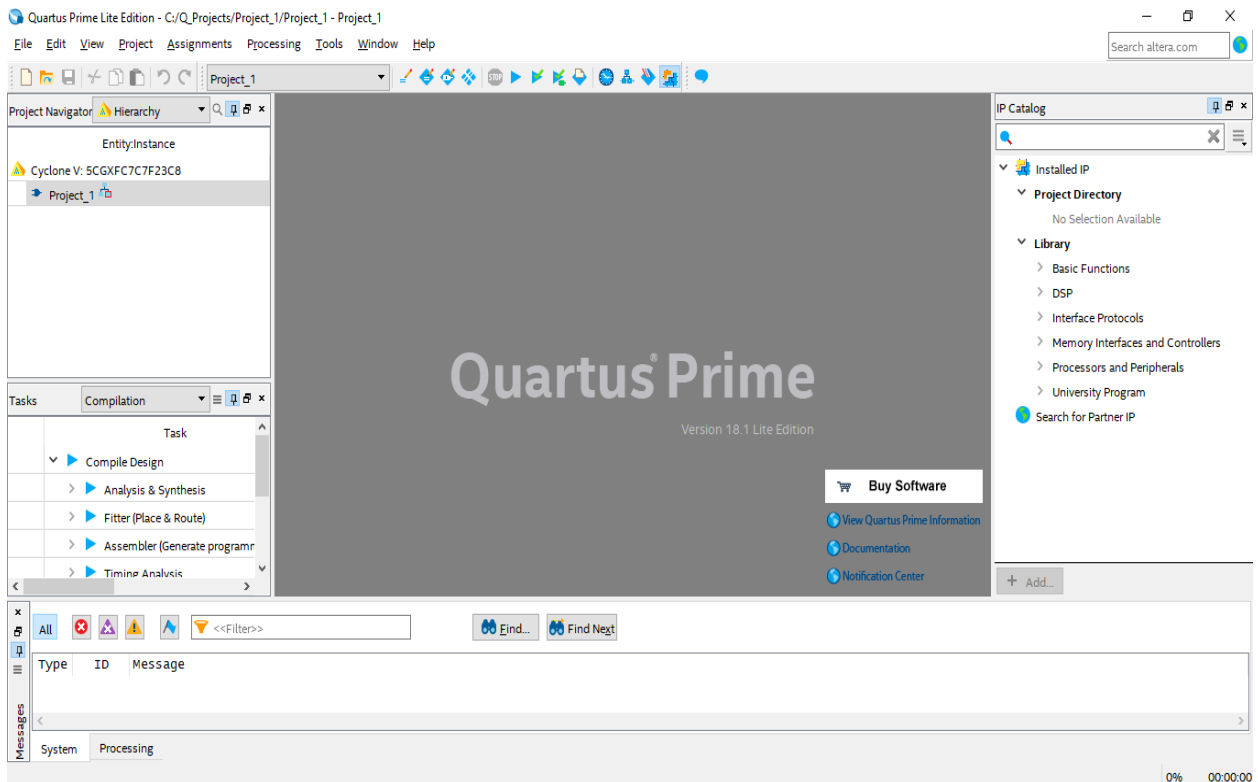


The project type display gives you additional setting options for what hardware the project will be compiled and implemented. Since, we'll simulate our projects on the program and we won't run on any real hardware, we'll choose the [Empty Project] option here, and then click the [Finish] button to create our project.

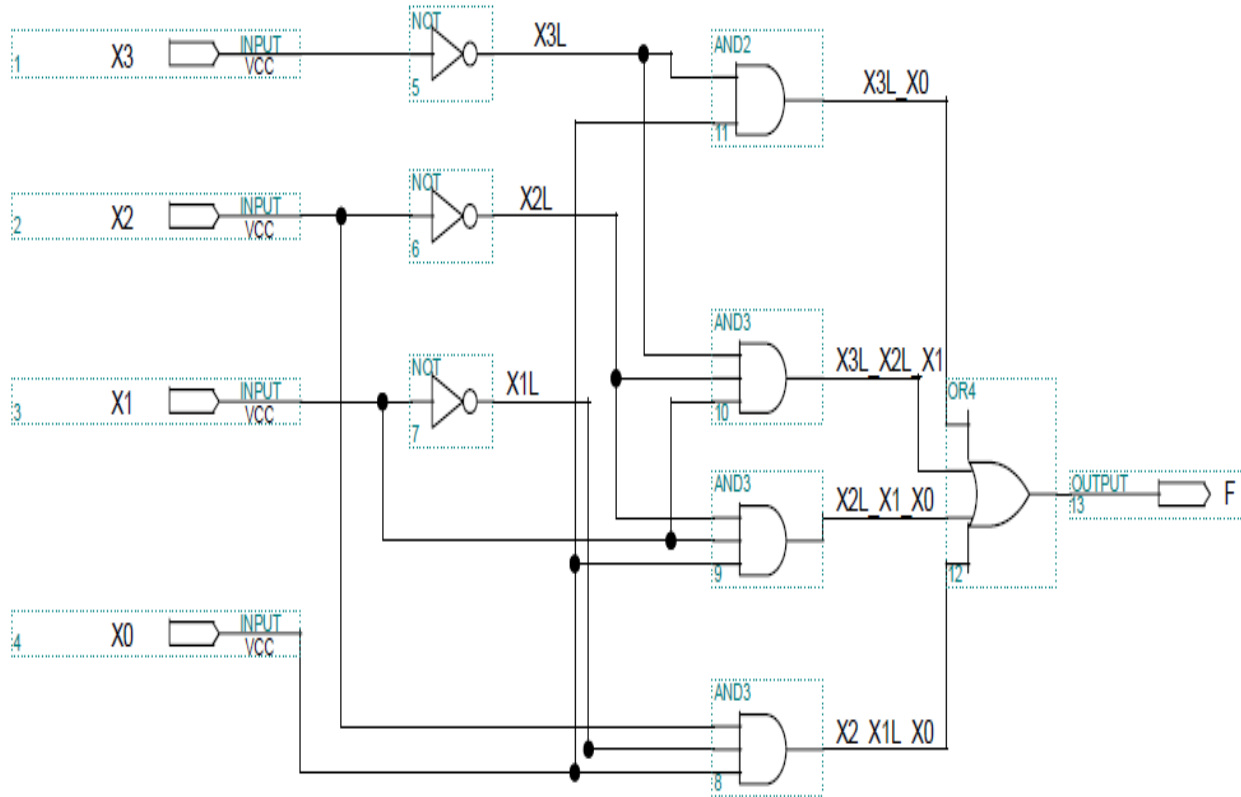




Once your project is created, you will see a screen like this:



**Step2:** Look at the program using VHDL code, here we have a practical circuit, implement it in Quartus (destine / compile/ simulate/show waveform), you can review pervious lab experiment and follow the steps.



**Step3:** In order to create a VHDL file follow the steps:

**File→New→VHDL File**

This is the VHDL code of above program:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY lab6 IS
PORT (x0, x1, x2,X3 : IN STD_LOGIC;
F : OUT STD_LOGIC);
END lab6;

ARCHITECTURE Arch_NelistStruct of lab6 is
SIGNAL X3L, X2L, X1L, X3L_X0, X3L_X2L_X1, X2L_X1_X0, X2_X1L_X0:STD_LOGIC;
BEGIN
X3L <= not X3;
X2L <= not X2;
X1L <= not X1;
X3L_X0 <= X3L and X0;
X3L_X2L_X1 <= X3L and X2L and X1;
X2L_X1_X0 <= X2L and X1 and X0;
X2_X1L_X0 <= X2 and X1 and X0;
F <= x3L_X0 or X3L_X2L_X1 or X2L_X1_X0 or x2_X1L_X0;
END Arch_NelistStruct;
```

**Note:** Trace the program line by line, now you are ready to learn single cycle data path using VHDL code.

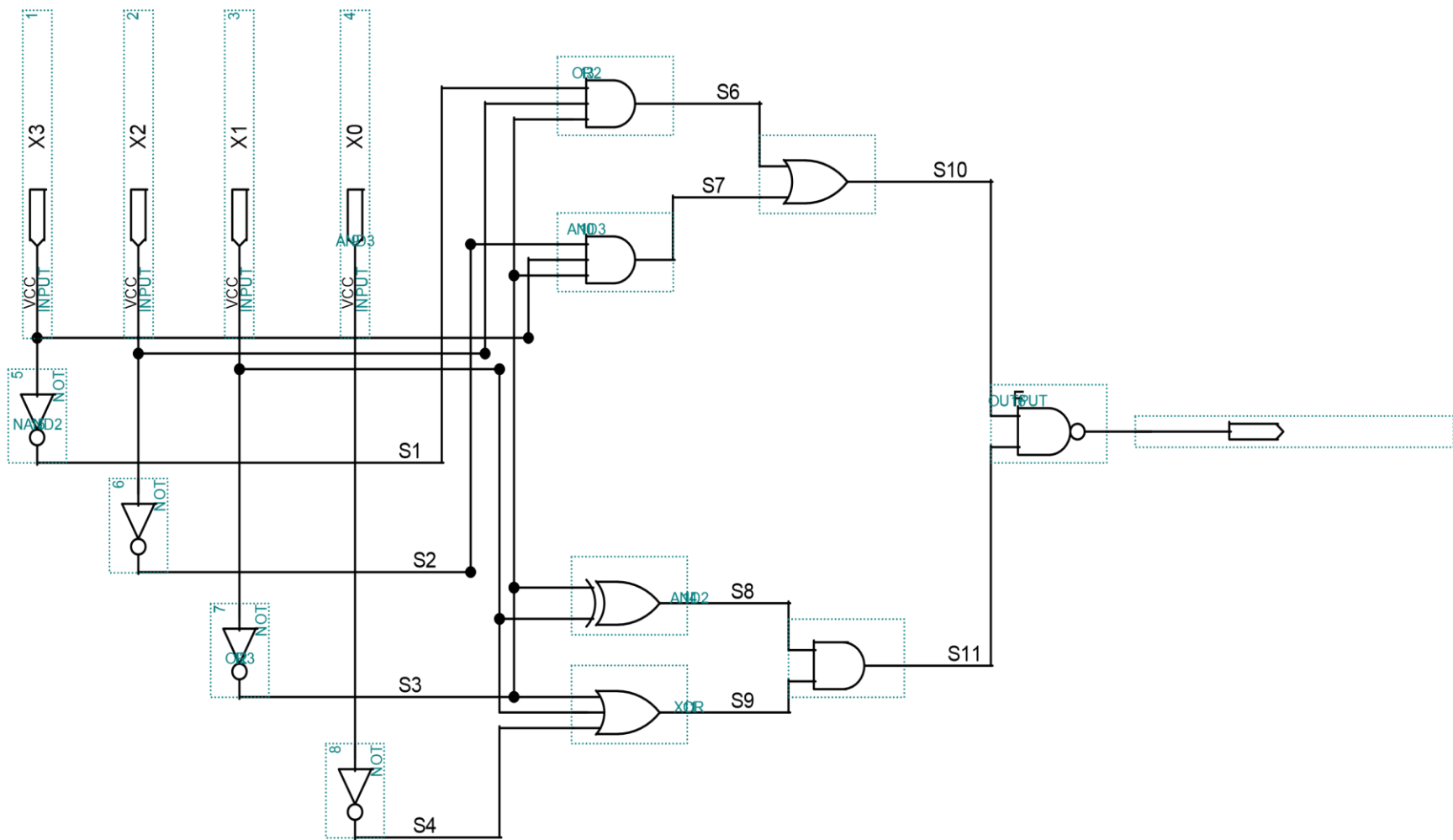
**EXPERIMENT VI:** *Basic VHDL programming of Combinational Circuits : Structural programming through NETLSTS*

**Objectives:** The main objective of this experiment is to introduce the basics of structural programming of combinational circuits using VHDL hardware description language and let the students perform implementations using Quarus II system.

**Preliminary Work:**

*1. Given the following combinational circuit,*

- A. Write down the Boolean expressions for each of the netlist signals from S1 to S11.*
- B. Write down the Boolean expression for the output function F, in terms of the netlist signals.*
- C. Write down a VHDL code describing this circuit using the netlist structural approach (Look at the example).*
- D. Draw the Truth Table for the given circuit.*
- E. Compile and simulate your design using Quartus II software.*
- F. Compare your simulation results with the truth table of the circuit.*



**NOTES:**

1. You should come to the laboratory with well-prepared preliminary work.
2. Your preliminary work must be prepared by yourself, it is encouraged to exchange ideas with your friends, but the final work done must be of your own.
3. Lab duration is 2 hours and it is not possible to extend it. Hence, you should finish experimental work and prepare the experimental report in 2 hours; no delay to the next day or to the next week is possible.

**Good Luck**

Dr. Adnan Acan

Dr. Muhammed Salamah

Dr. Omar Ramadan