

# Designing with VHDL and FPGA

**Instructor:**

**Dr. Ahmad El-Banna**

LAB# 1  
FALL 2016



( 1 )

# Agenda

Course Instructor

Course Contents

Course References

Overview of Digital Design

Intro. to VHDL language and FPGA technology

IDE tool Overview

# Course Instructor

- **Dr. Ahmad EL-Banna**

- B.Sc. in Telecommunications and Electronics, Fac. of Eng. at Shoubra, Benha Univ. 2005.
- 9-month Diploma in Embedded Systems, ITI, 2008.
- M.Sc. in Telecommunications and Electronics, Fac. of Eng. at Shoubra, Benha Univ. 2011.
- PhD. in Telecommunications and Electronics, E\_JUST Univ., 2014.
- Visiting Researcher , Wireless Communications Lab, Osaka University, 2013-2014.
- Find more at
  - [www.bu.edu.eg/staff/ahmad.elbanna](http://www.bu.edu.eg/staff/ahmad.elbanna)

# Your turn !

- About You
  - Graduation
    - Year
    - Univ.
    - ...

# Course Contents

- Introduction to VHDL language and FPGA technology
- Overview of digital design paradigms
- VHDL language constructs
- IDE tool Overview
- Data flow and Behavioral Implementation
- Statements (if, when-else, case, ...etc)
- Sequential Statements, Process and Variables
- Test benches
- Structural Implementation
- Designing various projects

# Course References

- **RTL Hardware Design Using VHDL**, P. Chu, 2006.
- **The VHDL Cookbook**, Peter J. Ashenden, 1<sup>st</sup> edition, 1990.
- **VHDL Tutorial: Learn by Example** by Weijun Zhang
  - <http://esd.cs.ucr.edu/labs/tutorial/>

# OVERVIEW OF DIGITAL DESIGN

# Many Design Tasks

- System specification (functionality and requirements)
- Hardware/software trade-offs
- Architecture selection and exploration
- Analysis and simulation
- Synthesis and optimization
- Implementation
- Testing and design for testability
- Verification and validation (V-cycle!)
- Design management: cooperation between tools, design flow, etc.



# Design Objectives

- **Unit cost:** the cost of manufacturing each copy of the system, excluding NRE cost.
- **NRE cost** (Non-Recurring Engineering cost): The one-time cost of designing the system.
- **Size:** the physical space required by the system.
- **Performance:** the execution time or throughput .
- **Power:** the amount of power consumed by the system.
- **Testability:** the easiness of testing the system to make sure that it works correctly.
- **Flexibility:** the ability to change the functionality of the system without incurring heavy NRE cost.
- **Correctness, safety, ...** etc.

# The Main Challenges

- System complexity
- Increasing functionality and diversity
- Increasing performance
- Stringent design requirements
- Low cost and low power
- Dependability: reliability, safety and security
- Testability and flexibility
- Technology challenges for cost-efficient implementation
- Deep submicron effects (e.g., cross talk and soft errors)

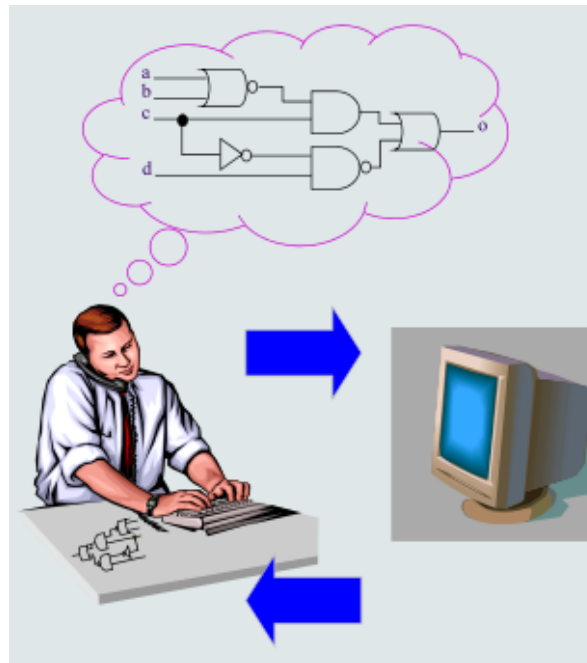
## **Possible Solutions:**

- Powerful design methodology and CAD tools.
- Advanced architecture (modularity).
- Extensive design reuse.

# DESIGN PARADIGMS

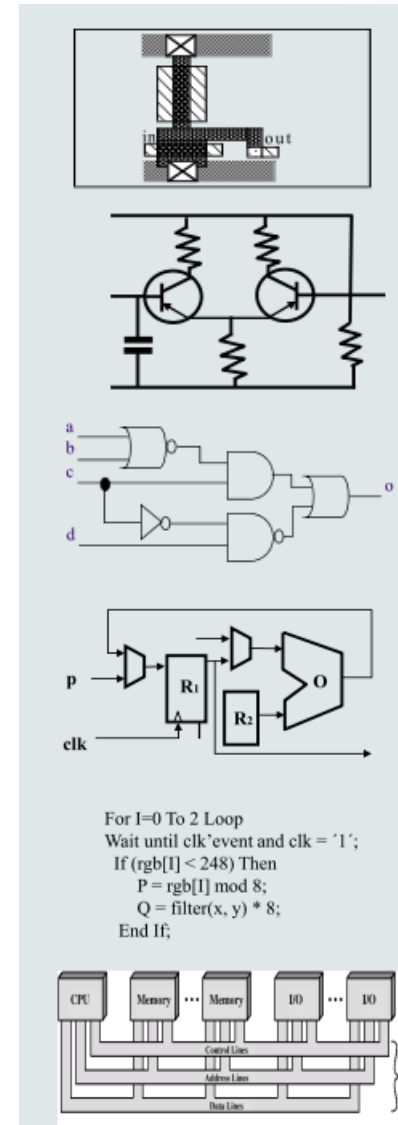
# Capture and Simulate

- The detailed design is captured in a **model**.
- The model is simulated.
- The results are used to guide the improvement of the design.
- All design decisions are made by the designers.



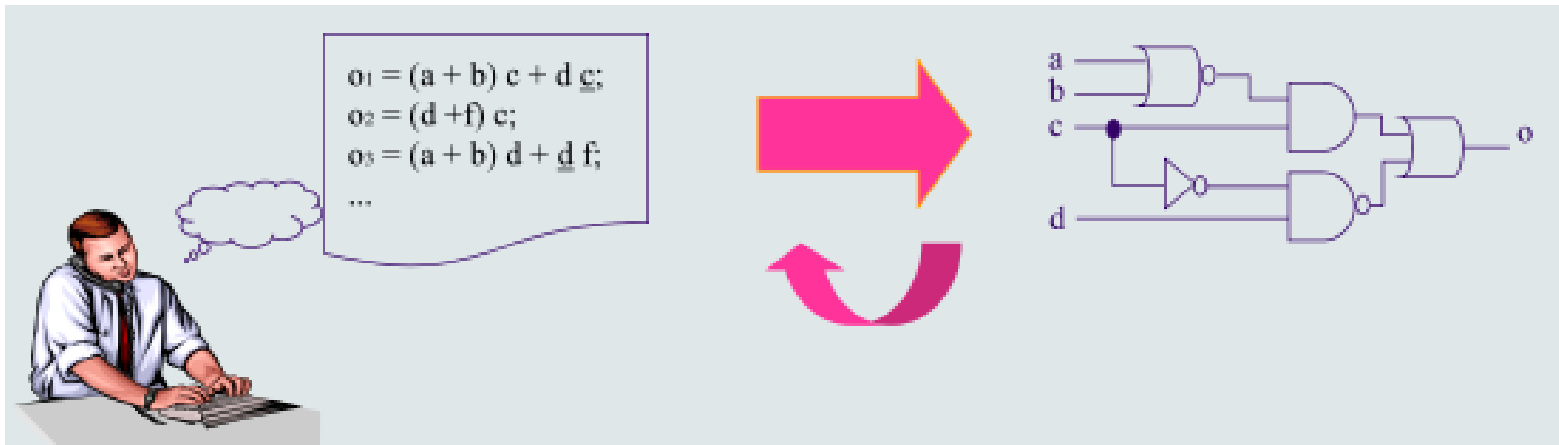
# Abstraction Hierarchy

- **Layout/silicon level :**
  - The physical layout of the integrated circuits is described.
- **Circuit level :**
  - The detailed circuits of transistors, resistors, and capacitors are described.
- **Logic (gate) level :**
  - The design is given as gates and their interconnections.
- **Register-transfer level (RTL) :**
  - Operations are described as transfers of values between registers.
- **Algorithmic level :**
  - A system is described as a set of usually concurrent algorithms.
- **System level :**
  - A system is described as a set of processors and communication channels.



# Describe and Synthesize Paradigm

- Description of a design in terms of **behavioral** specification.
- Refinement of the design towards an implementation by adding **structural** details.
- Evaluation of the design in terms of a cost function and the design is optimized w.r.t. the cost function.



# Other Paradigms

- Y-Chart
  - Behavioral, structure & physical domain
- Core-based Design
  - Reuse of IP blocks e.g. CPU, DSP,...
- Platform-based Design
  - Customized embedded processors or software

# INTRODUCTION TO FPGA

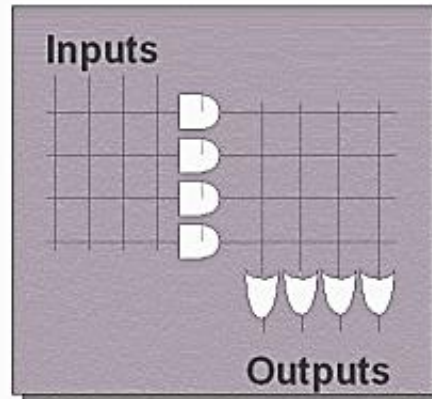


# History of Programmable Logic

## PLA

- Two programmable planes
- Any Combination of ANDs / ORs
- Sharing of AND terms across multiple OR's
- Highest logic density available to user
- High Fuse count, Slower than PALs
- Programmable Logic Array - PLA

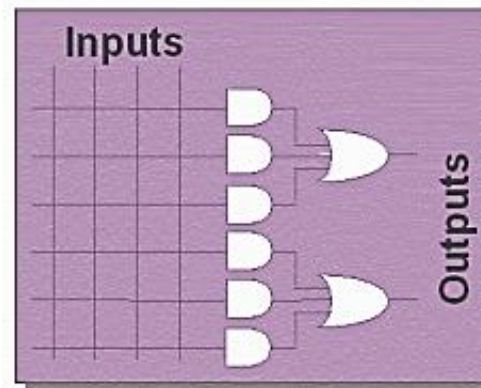
Signetics - Cline - 1975



## PAL

- One programmable plane - AND / Fixed OR
- Finite combination of ANDs / ORs
- Medium logic density available to user
- Lower Fuse count, Faster than PLAs (at this time fabricated on a 10um process)
- Programmable Array Logic - PAL

MMI - Birkner - 1978

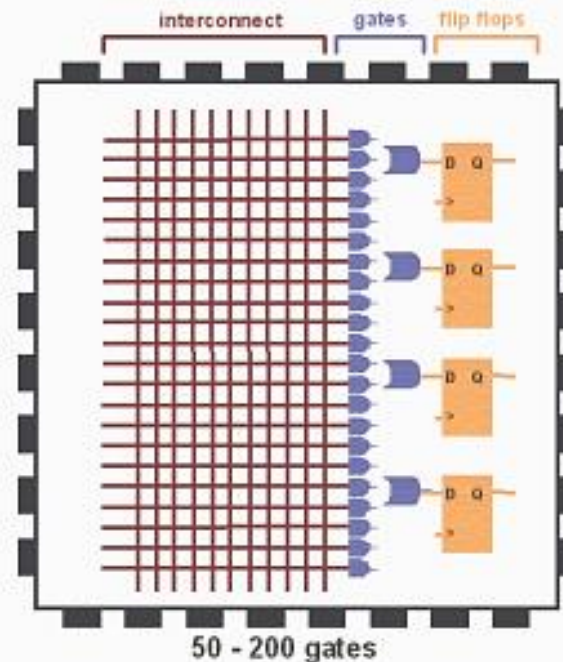


**SPLD:**  
Simple Programmable  
Logic Devices

# History of Programmable Logic

## CPLD Architecture

- Central, Global Interconnect
- Simple, Deterministic Timing
- Easily routed
- PLD Tools add only interconnect
- Wide, fast complex gating



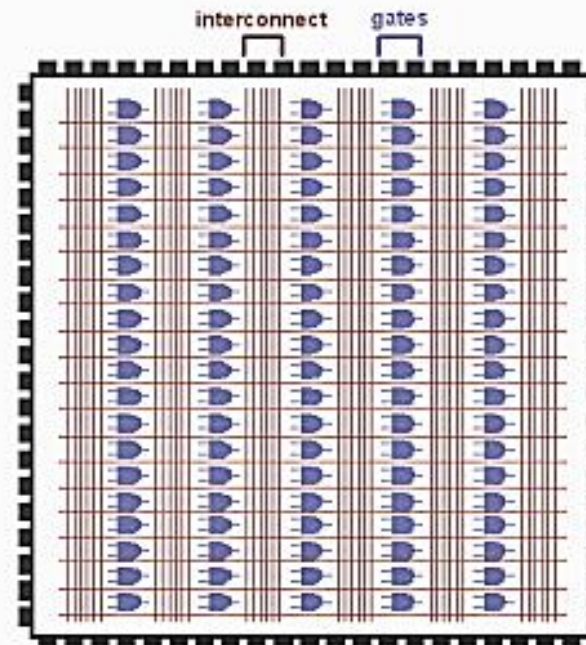
Complex Programmable Logic Devices (CPLDs)

# History of Programmable Logic

## FPGA Architecture

- Channel Based Routing
- Post Layout Timing
- Tools More Complex than CPLDs
- Fine Grained
- Fast register pipelining

"What if we could develop the equivalent of a circuit board full of standard logic parts (like TTL & PAL devices) on a single high density programmable logic chip?"



1,000,000+ gates

Xilinx - Freeman - 1985

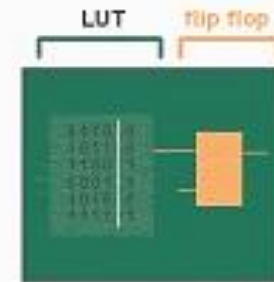
FPGA - Field Programmable Gate Array

# History of Programmable Logic

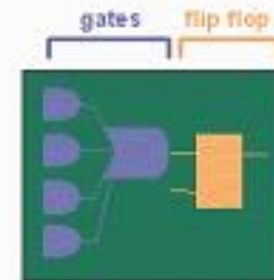
## FPGA - Field Programmable Gate Array

### 2 types of FPGAs

- Reprogrammable (SRAM-based)
  - SRAM determines interconnect
  - SRAM defines logic in Look Up Table (LUT)
- One-time Programmable (OTP)
  - Interconnect is anti-fuse
  - Logic is traditional gates



SRAM logic cell



OTP logic cell



# History of Programmable Logic

## Basic Logic Definitions

### Standard Discrete Logic

- Fixed function devices which can be connected together to implement a system

### PLD (CPLDs & FPGAs)

- Devices which can be re-programmed to implement any function within the device's resources

### Gate Array (GA)

- Blocks of gates that are customized at the fab by adding layers of metal interconnects

### Standard Cell (ASIC)

- ICs designed with cell libraries. All mask layers customized at the fab

# Why FPGAs?

- Ideal for customized designs
  - Product differentiation in a fast-changing market
- Offer the advantages of high integration
  - High complexity, density, reliability
  - Low cost, power consumption, small physical size
- Avoid the problems of ASICs
  - high NRE cost, long delay in design and testing
  - increasingly demanding electrical issues

Fast Time-to-Market,  
fast response to market changes

# FPGA Advantages

- Very fast custom logic
  - massively parallel operation
- Faster than microcontrollers and microprocessors
  - much faster than DSP engines
- More flexible than dedicated chipsets
  - allows unlimited product differentiation
- More affordable and less risky than ASICs
  - no NRE, minimum order size, or inventory risk
- Reprogrammable at any time
  - in design, in manufacturing, after installation

# VHDL BASICS



# VHDL

- Hardware description languages (HDL)
  - Language to describe hardware
  - Two popular languages
    - **VHDL: Very High Speed Integrated Circuits Hardware Description Language**
      - Developed by US DOD from 1983
      - IEEE Standard 1076-1987/1993/200x
      - Based on the ADA language
    - **Verilog**
      - IEEE Standard 1364-1995/2001/2005
      - Based on the C language
- **Applications of HDL:**
  - Model and document digital systems
    - Different levels of abstraction
      - Behavioral, structural, etc.
  - Verify design
  - Synthesize circuits
    - Convert from higher abstraction levels to lower abstraction levels

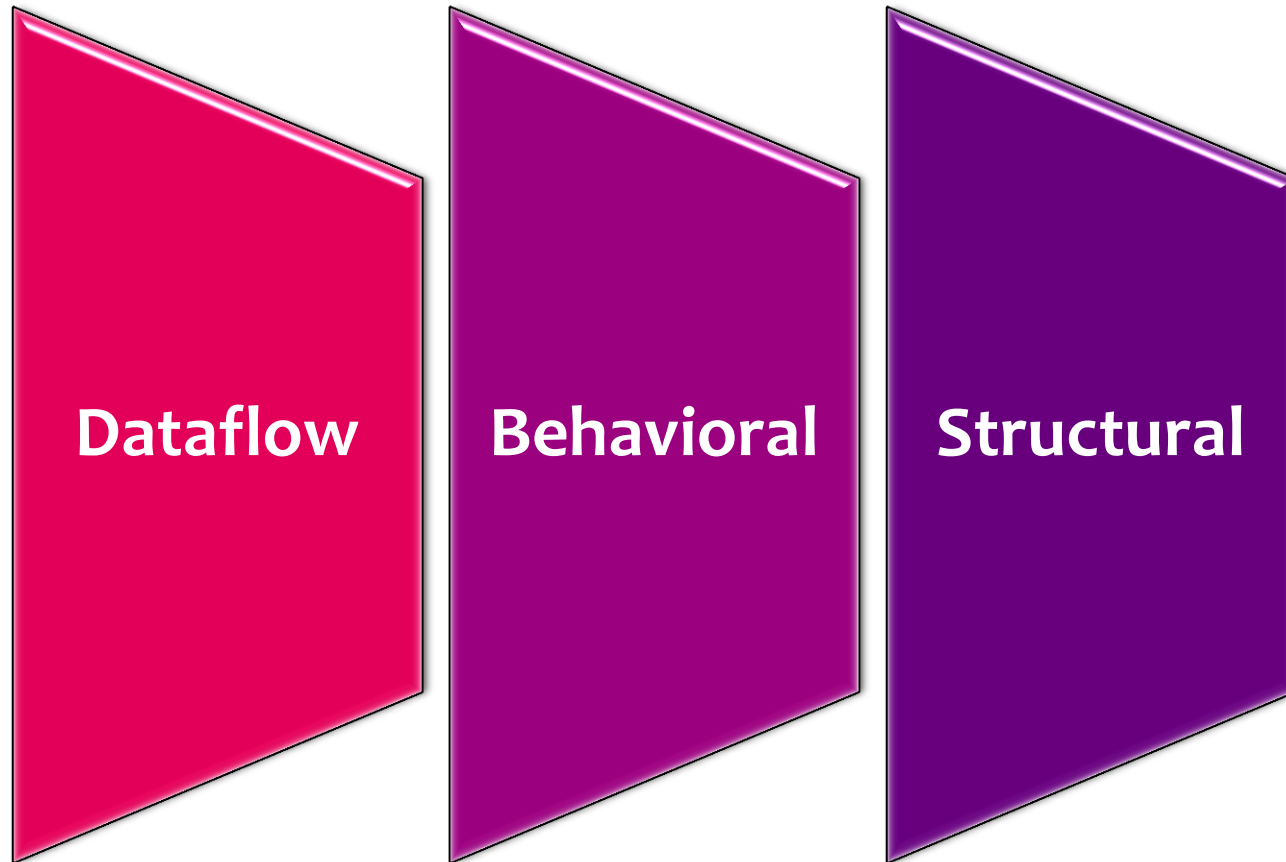
# Modeling Digital Systems

- VHDL is for coding models of a digital system...
- Reasons for modeling
  - requirements specification
  - documentation
  - testing using simulation
  - formal verification
  - synthesis
- Goal
  - most 'reliable' design process, with minimum cost and time
  - avoid design errors!

# Basic VHDL Concepts

- Main Terms
  - Interfaces -- i.e. ports
  - Behavior
  - Structure
  - Test Benches
  - Analysis, simulation
  - Synthesis
- VHDL is a programming language that allows one to model and develop complex digital systems in a dynamic environment.
- Object Oriented methodology -- modules can be used and reused.
- Allows you to designate in/out ports (bits) and specify behavior or response of the system.
- But VHDL is NOT C ...  
There are some similarities, as with any programming language, but syntax and logic are quite different.

# 3 ways to DO IT -- the VHDL way



# Modeling the Dataflow way

- uses statements that defines the actual flow of data.....

such as,

$x \leq y$       -- this is NOT less than equal to  
                  -- told you its not C

this assigns the boolean signal x to the value of boolean signal y...

i.e.  $x = y$

***this will occur whenever y changes....***

# Jumping right in to a Model

- lets look at an **and gate** model -- doing it the dataflow way.....  
ignore the extra junk for now –

entity and\_gate is

```
    port ( a,b: in  bit;  
          c: out bit);
```

end and\_gate;

architecture dataflow of and\_gate is

begin

```
    c<= a and b;
```

end dataflow;

# A FIRST LOOK ON THE IDE

# ISE Design Suite <sup>®</sup> 14.2

The screenshot displays the ISE Design Suite 14.2 interface. The main window is titled "gates\_tst Project Status (08/04/2016 - 14:21:34)". The interface is divided into several panes:

- Design Overview:** A tree view showing the project structure, including Summary, IOB Properties, Module Level Utilization, Timing Constraints, Pinout Report, Clock Report, Static Timing, Errors and Warnings, Parser Messages, Synthesis Messages, and Translation Messages.
- Design Properties:** A section for configuring the design summary, with options like "Enable Message Filtering" and "Optional Design Summary Contents" (Show Clock Report, Show Failing Constraints, Show Warnings).
- Project Status Table:** A table providing key project information:
 

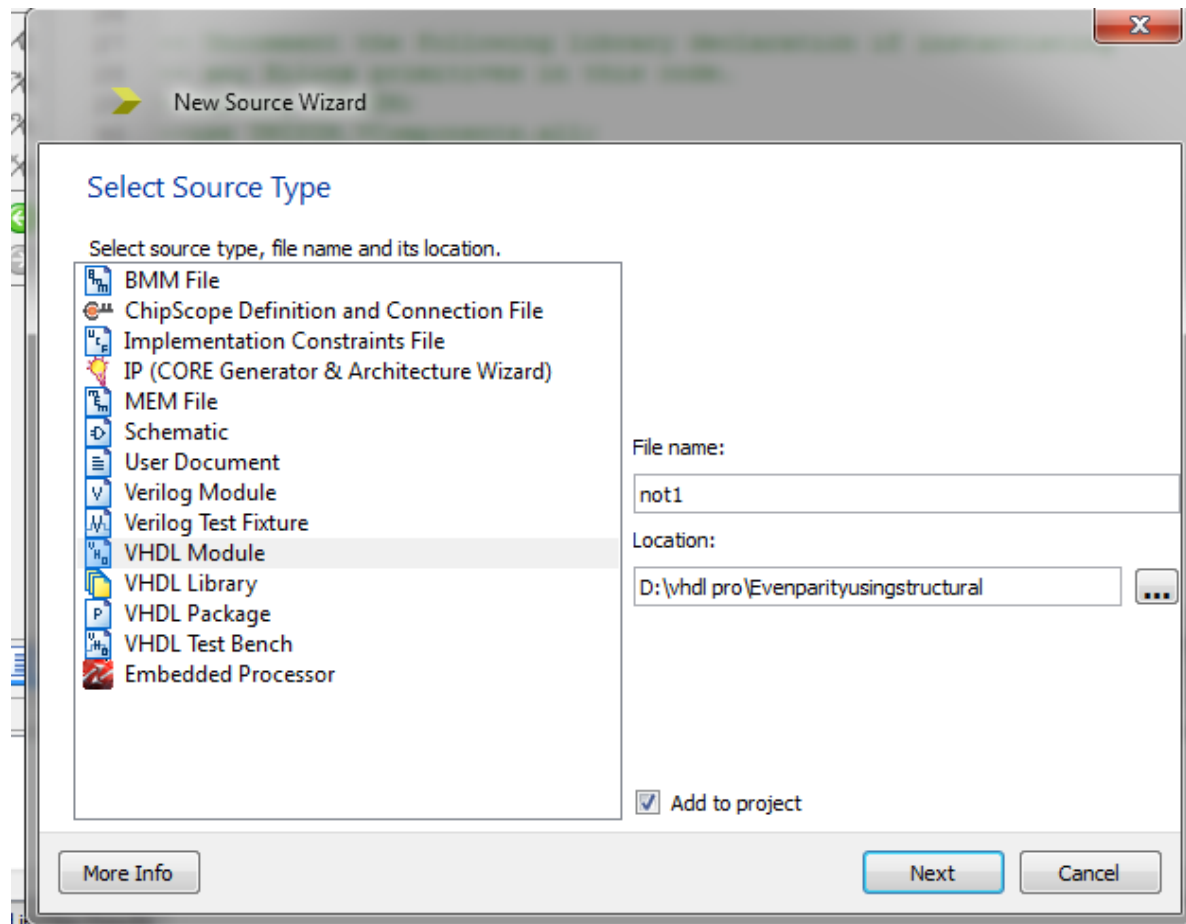
gates_tst Project Status (08/04/2016 - 14:21:34)			
<b>Project File:</b>	tst_prj.xise	<b>Parser Errors:</b>	No Errors
<b>Module Name:</b>	gates_tst	<b>Implementation State:</b>	Placed and Routed
<b>Target Device:</b>	xc7a100t-3csg324	<b>Errors:</b>	No Errors
<b>Product Version:</b>	ISE 14.2	<b>Warnings:</b>	<a href="#">7 Warnings (7 new)</a>
<b>Design Goal:</b>	Balanced	<b>Routing Results:</b>	<a href="#">All Signals Completely Routed</a>
<b>Design Strategy:</b>	<a href="#">Xilinx Default (unlocked)</a>	<b>Timing Constraints:</b>	
<b>Environment:</b>	<a href="#">System Settings</a>	<b>Final Timing Score:</b>	0 ( <a href="#">Timing Report</a> )
- Console:** Shows the following messages:
 

```
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/Users/A3B2/tst_prj/gates_tst.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
```



# Build a vhdl module

## step1: Construct a new source



# Build a vhdl module

## step2: Define the entity (i/o)

New Source Wizard

Define Module

Specify ports for module.

Entity name

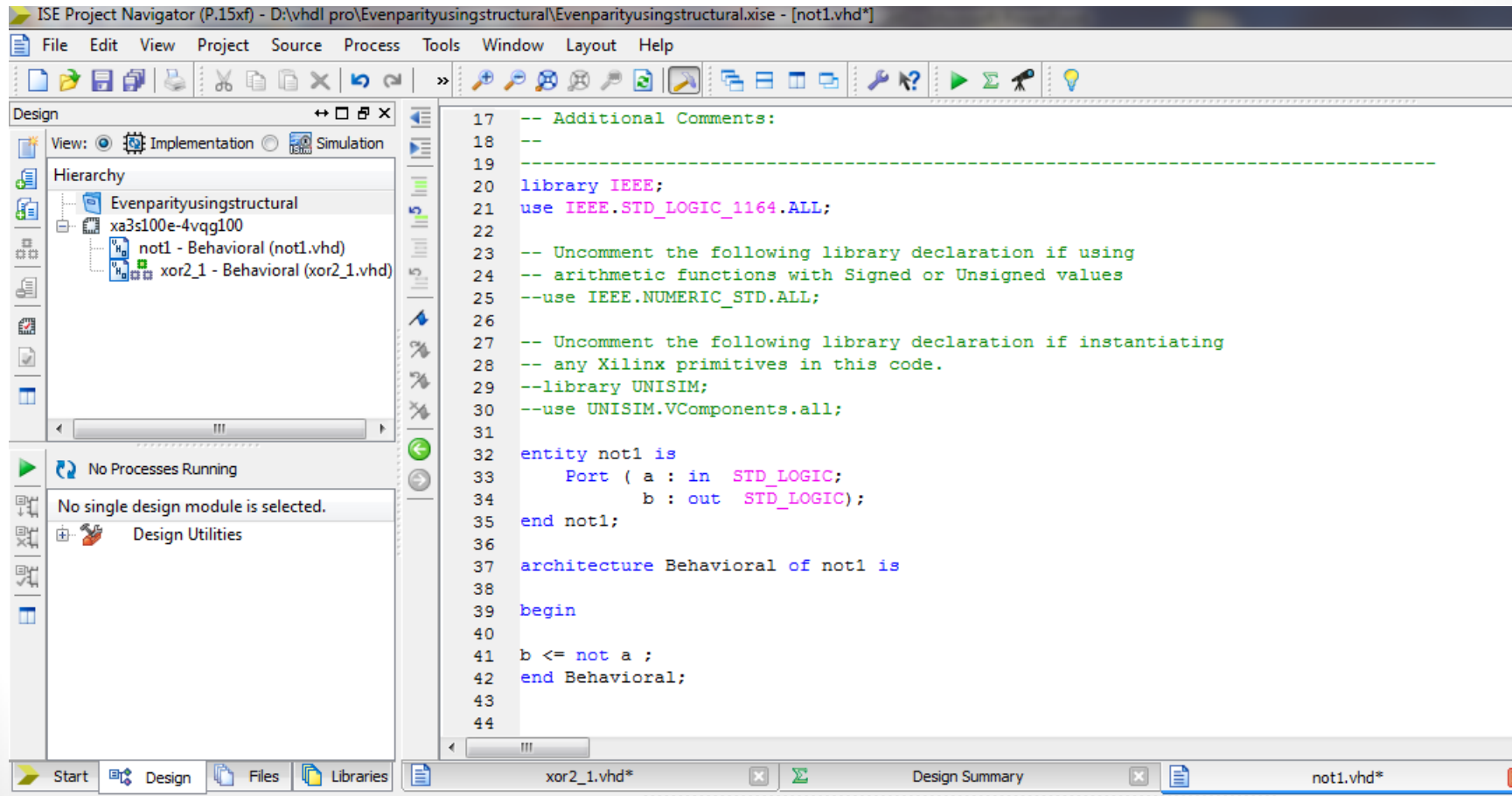
Architecture name

Port Name	Direction	Bus	MSB	LSB
a	in	<input type="checkbox"/>		
b	out	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		
	in	<input type="checkbox"/>		

More Info Next Cancel

# Build a vhdl module

## step3: Define the architecture



The screenshot shows the ISE Project Navigator interface. The main window displays the VHDL code for a module named 'not1'. The code is as follows:

```
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity not1 is
33     Port ( a : in  STD_LOGIC;
34           b : out STD_LOGIC);
35 end not1;
36
37 architecture Behavioral of not1 is
38
39 begin
40
41 b <= not a ;
42 end Behavioral;
43
44
```

The interface also shows a Hierarchy view on the left with the following structure:

- Evenparityusingstructural
  - xa3s100e-4vqg100
    - not1 - Behavioral (not1.vhd)
    - xor2\_1 - Behavioral (xor2\_1.vhd)

The bottom status bar shows the current file is 'not1.vhd\*' and the Design Summary window is open.

- For more details, refer to:
  - VHDL Tutorial: Learn by Example by Weijun Zhang
    - <http://esd.cs.ucr.edu/labs/tutorial/>
  - “**Introduction to VHDL**” presentation by Dr. Adnan Shaout, *The University of Michigan-Dearborn*
  - **The VHDL Cookbook**, Peter J. Ashenden, 1<sup>st</sup> edition, 1990.
- For inquiries, send to:
  - [ahmad.elbanna@feng.bu.edu.eg](mailto:ahmad.elbanna@feng.bu.edu.eg)