ECE381(CAD), Lecture 4:

VHDL Basics

Mehdi Modarressi

Department of Electrical and Computer Engineering,

University of Tehran

Some slides are taken (with modifications) from ECE-448 of GMU

Outline

- An introduction to VHDL
  - Basic structure and syntax
  - Basic modeling techniques

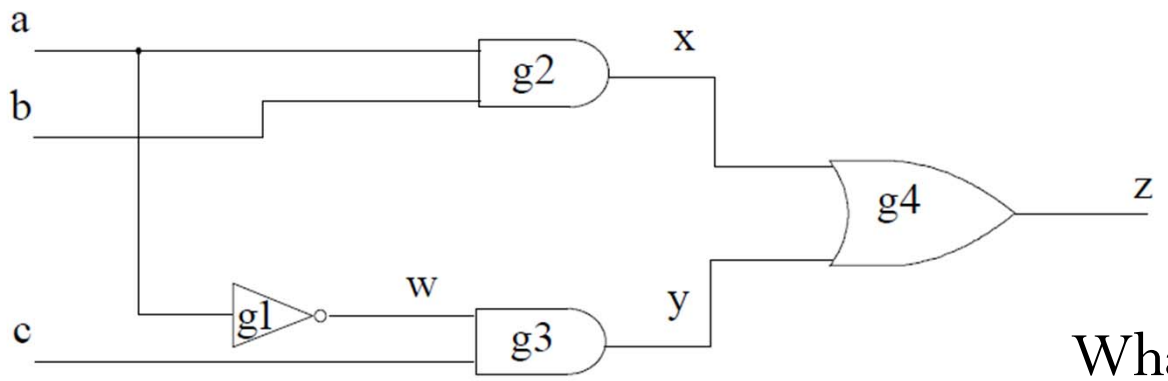
HDL

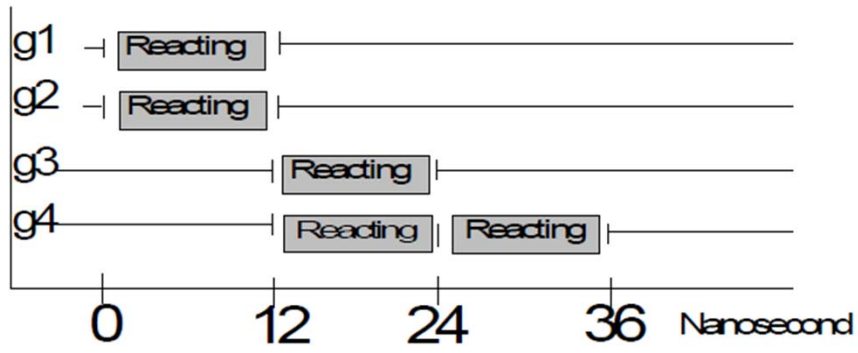
- Hardware description language: a language that ease hardware description
- Several famous HDLs:
  - VHDL
  - Verilog
  - AHDL
  - ….
- Major requirements:
  - Concurrency
  - Time notion
  - Hardware-specific data types

HDL - Concurrency

- Software : Statements are executed sequentially
  - The *von Neumann* model of computation
  - The sequence of statements is significant, since they are executed in that order
  - Java, C++, C, Ada, Pascal, Fortran,…
- Hardware : No sequence, events happen concurrently
  - No Sequence: a software language cannot be used for describing and simulating hardware
  - There is an explicit notion of time

Concurrency





What does happen if ***a*** changes from '1' to '0', when all gates have a delay of 12 ns?

VHDL

- VHDL: VHSIC Hardware Description Language
- VHSIC is an acronym of *Very High Speed Integrated Circuits*
- A Formal Language for Specifying the Behavior and Structure of a Digital Circuit
  - Allows Top-Down design
  - Allows description at different levels

VHDL history

- VHDL originated in the early 1980s
  - The American Department of Defense initiated the development of VHDL in the early 1980s
  - Based on the **Ada** programming language
- VHDL was standardized in 1987 by the IEEE (IEEE std-1076-1987)
- Easy to move VHDL code between different commercial platforms (tools)
- The standard was revised version in several times including revisions in 1993, 2000, 2002, and 2008

→ ***See Chapter 2 of Dr Navabi's book for a comprehensive survey on the HDL background***

VHDL basics

- Some points about the VHDL semantics
- An introduction to ENTITY and ARCHITECTURE in VHDL

Naming and labeling

- VHDL is case insensitive
- General rules of thumb
  1. All names should start with an alphabet character (a-z or A-Z)
  2. Use only alphabet characters (a-z or A-Z) digits (0-9) and underscore (_)
  3. Do not use any punctuation or reserved characters within a name (!, ?, ., &, +, -, etc.)
  4. Do not use two or more consecutive underscore characters (__) within a name (e.g., Sel__A is invalid)
  5. All names and labels in a given entity and architecture must be unique

Free format

- VHDL is a "free format" language

No formatting conventions, such as spacing or indentation imposed by VHDL compilers. Space and carriage return treated the same way.

Example:

```
if (a=b) then
```

or

```
if      (a=b)          then
```

or

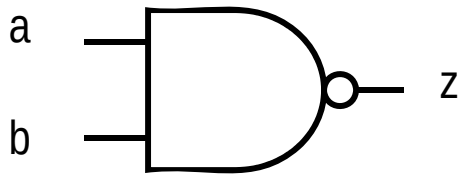
```
if (a =
b) then
```

are all equivalent

Comments

- Comments in VHDL are indicated with a "double dash", i.e., "--"
  - Comment indicator can be placed anywhere in the line
  - Any text that follows in the same line is treated as a comment
  - Carriage return terminates a comment
- No method for commenting a block extending over a couple of lines
- Examples:
  - -- main sub_circuit
  - Data_in <= Data_bus; -- reading data from the input FIFO

VHDL basics- ENTITY



Example: NAND Gate

a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

VHDL basics- ENTITY

- 3 sections of a VHDL code
- ENTITY: most basic building block of a design

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

ENTITY nand_gate IS
    PORT(
        a   : IN STD_LOGIC;
        b   : IN STD_LOGIC;
        z   : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE model OF nand_gate IS
BEGIN
    z <= a NAND b;
END model;
```

LIBRARY DECLARATION

ENTITY

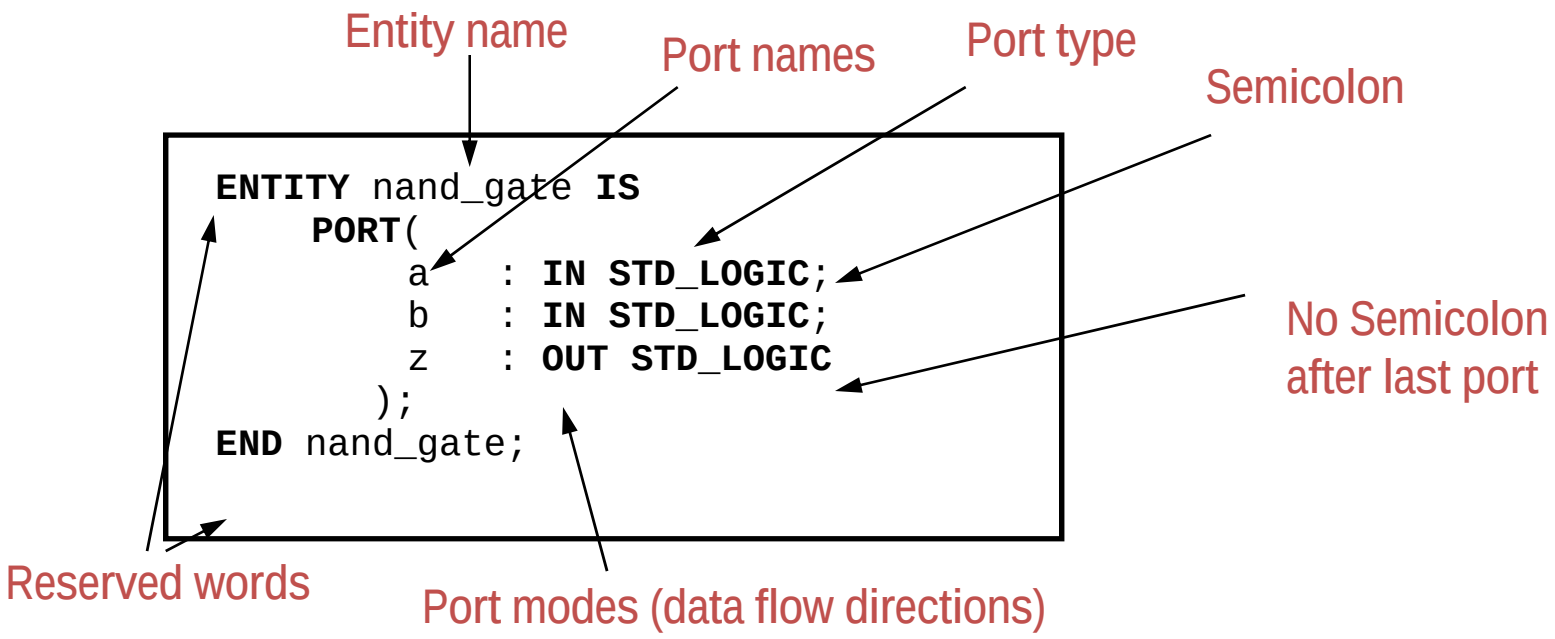
ARCHITECTURE

Entity– simplified syntax

```
ENTITY entity_name IS
   PORT (
      port_name :  port_mode  signal_type;
      port_name :  port_mode  signal_type;
      ……..
      port_name :  port_mode  signal_type);
END entity_name;
```

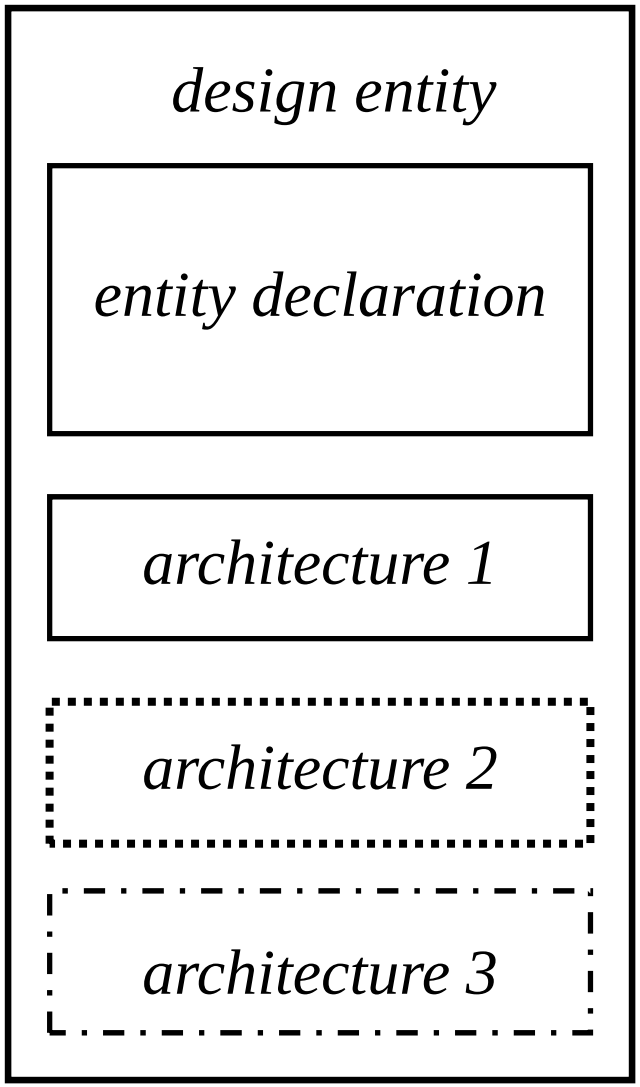
Entity declaration

- Entity Declaration describes the interface of the component
  - Input and output ports.



Entity and Architecture

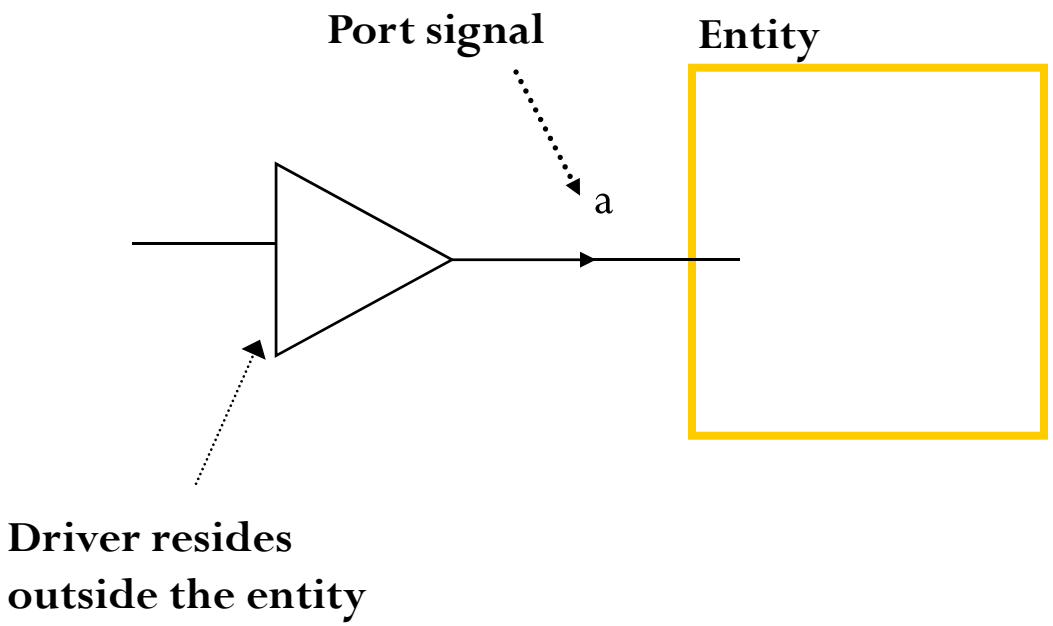
One ENTITY can have many different ARCHITECTUREs



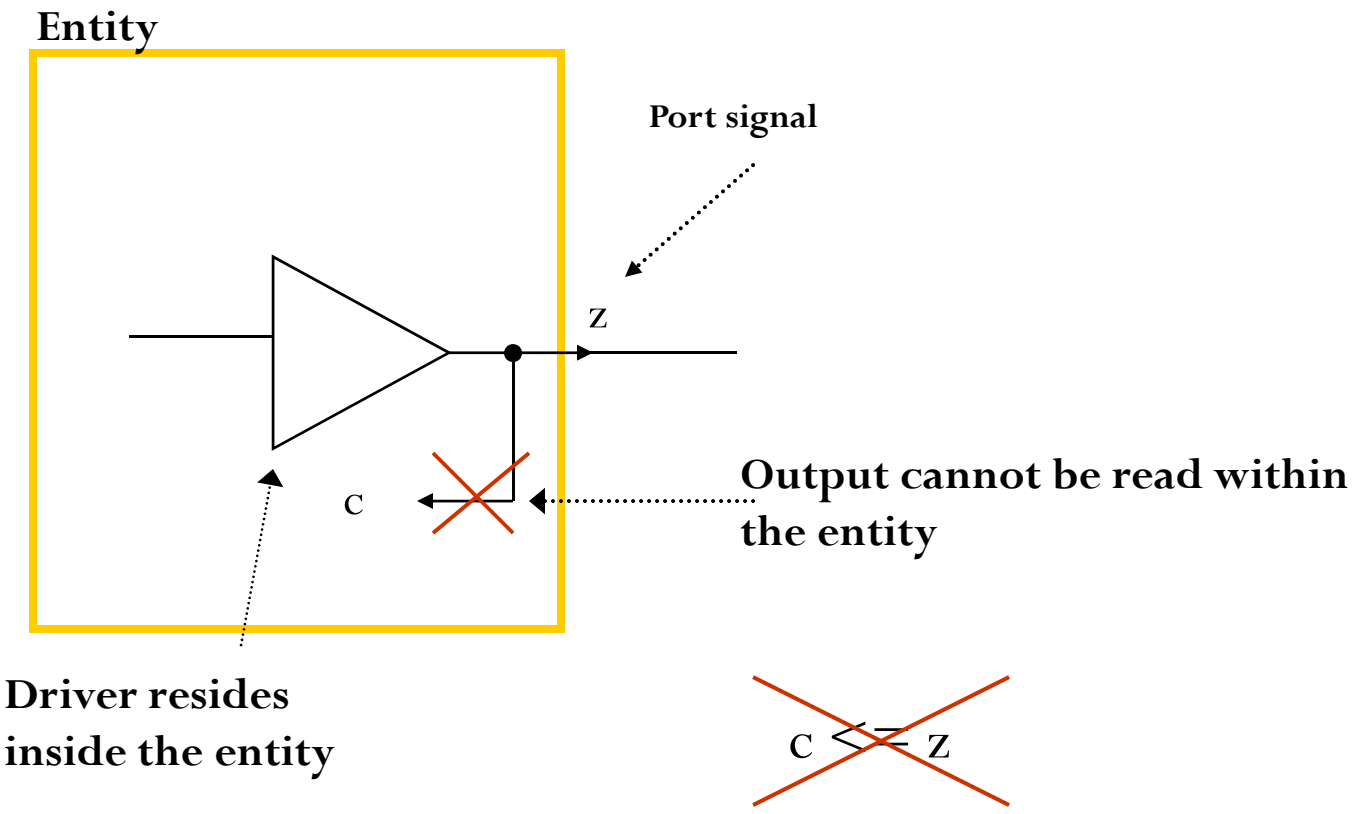
Objects in VHDL

- four object classes:
  - Signal: represent hardware wires and have timing
  - Variable: storage of temp. data, no timing
  - Constant: constant value of a given type
  - File: external storage
- The difference between variables and signals will be introduced in due course!
- Entity *Ports* behave as a signal

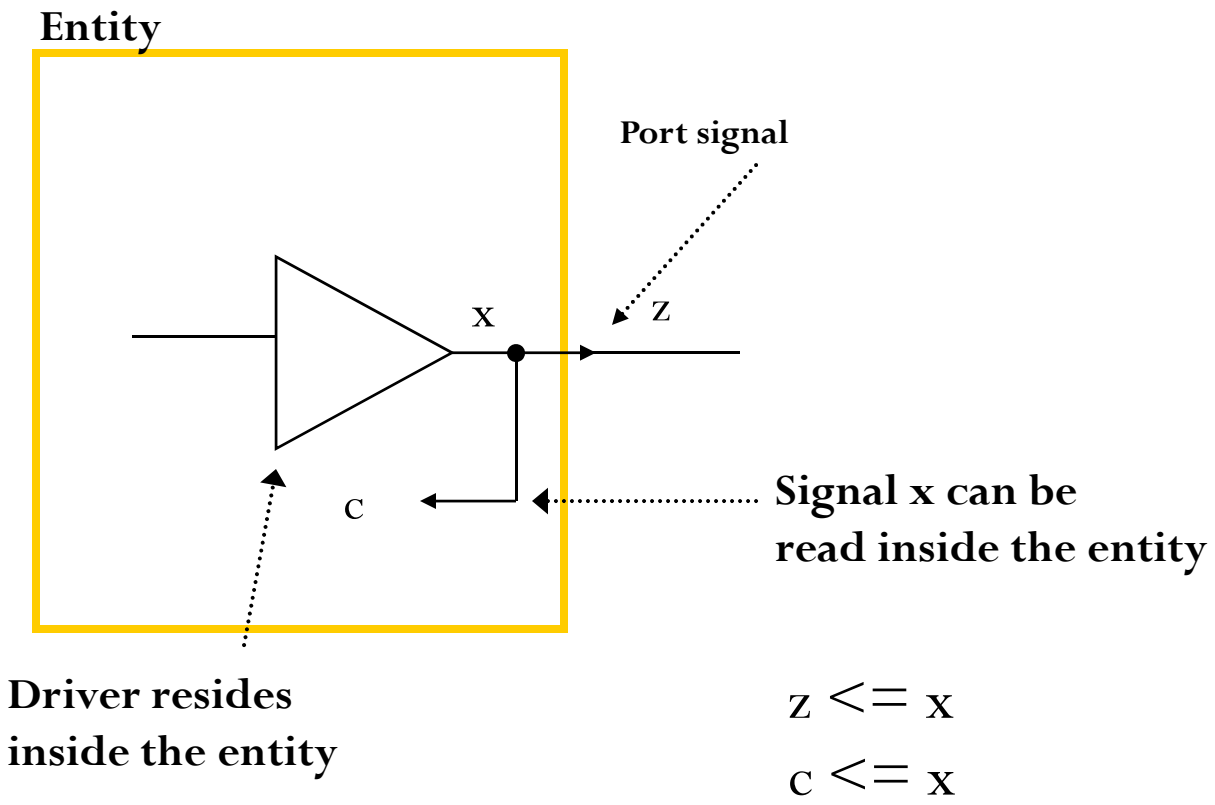
Port mode: IN



Port mode: OUT

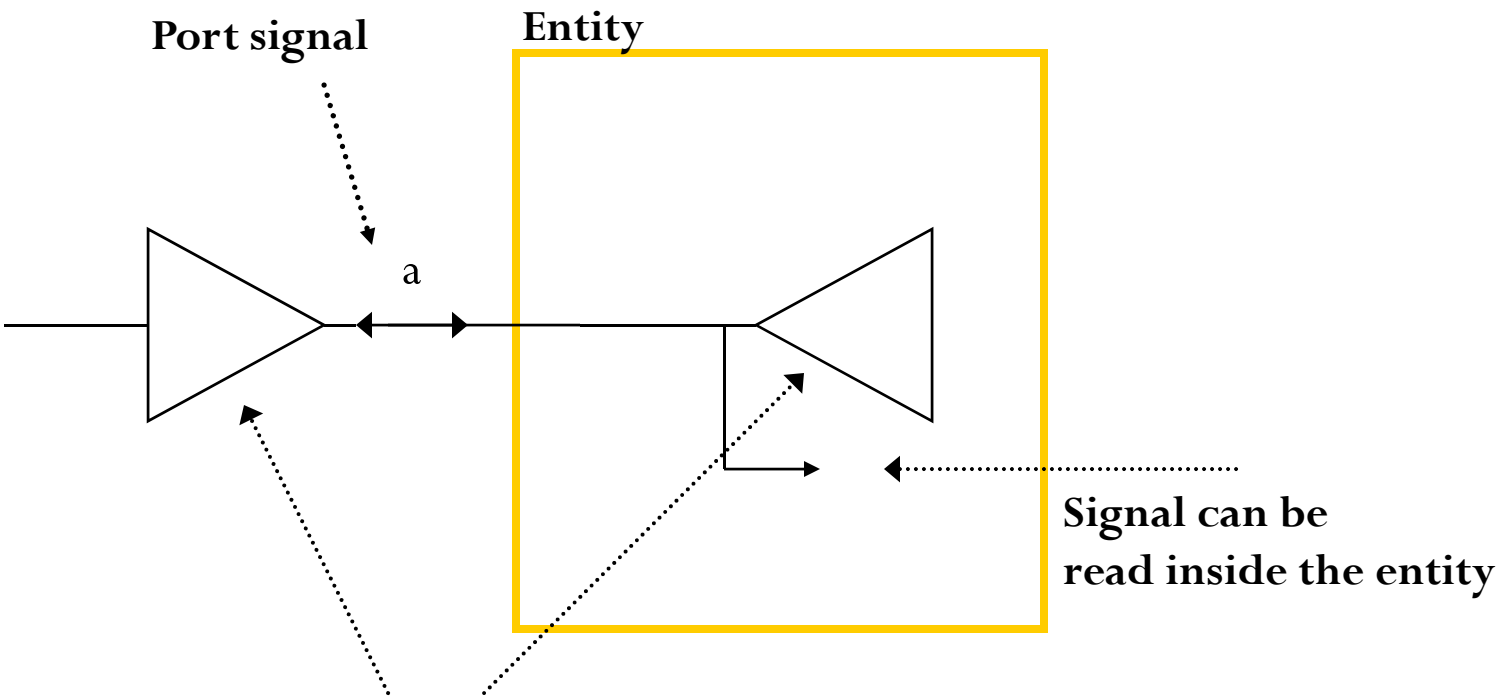


Port mode: BUFFER



- Is output but can be read inside the entity

Port mode: INOUT



Port modes - summary

- Port Mode: describes the direction in which data travels with respect to the component
  - **In**: Data comes in this port and can only be read within the entity. It can appear **only on the right side** of a signal or variable assignment.
  - **Out**: The value of an output port can only be updated within the entity. **It cannot be read**. It can only appear **on the left side** of a signal assignment.
  - **InOut**: The value of a bi-directional port can be read and updated within the entity model. It can appear on **both sides** of a signal assignment.
  - **Buffer:** Used for a signal that is an output from an entity. The value of the signal can be used inside the entity, which means that in an assignment statement the signal can appear on the left and right sides of the <= operator

ARCHITECTURE

- Describes an implementation of a design entity
- Architecture example:

```
ARCHITECTURE model OF nand_gate IS
BEGIN
   z <= a NAND b;
END model;
```

ARCHITECTURE – simplified syntax

```
ARCHITECTURE architecture_name OF entity_name IS
   [ declarations ]
BEGIN
    code
END architecture_name;
```

ENTITY & ARCHITECTURE

nand_gate.vhd

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

ENTITY nand_gate IS
    PORT(
        a   : IN STD_LOGIC;
        b   : IN STD_LOGIC;
        z   : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE dataflow OF nand_gate IS
BEGIN
    z <= a NAND b;
END dataflow;
```

Library declarations

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

ENTITY nand_gate IS
    PORT(
        a   : IN STD_LOGIC;
        b   : IN STD_LOGIC;
        z   : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE model OF nand_gate IS
BEGIN
    z <= a NAND b;
END model;
```

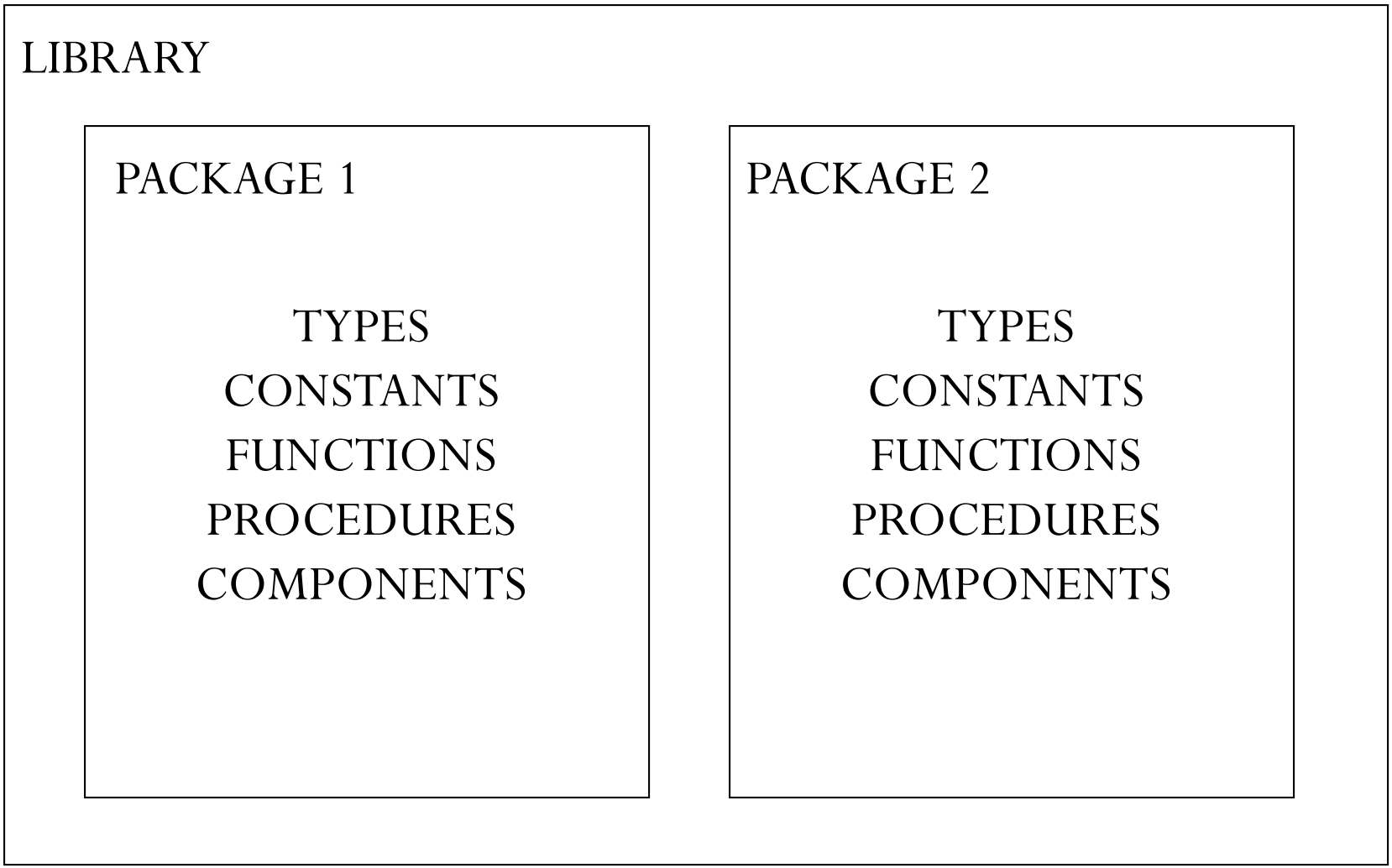
Library declaration

Use all definitions from the package std_logic_1164

Library declarations - syntax

```
LIBRARY  library_name;
USE library_name.package_name.package_parts;
```

Fundamental parts of a library



Libraries

- ieee

  Specifies multi-level logic system, including STD_LOGIC, and STD_LOGIC_VECTOR data types

Need to be explicitly declared

- std

  Specifies pre-defined data types (BIT, BOOLEAN, INTEGER, REAL, SIGNED, UNSIGNED, etc.), arithmetic operations, basic type conversion functions, basic text i/o functions, etc.

- work

  Current designs after compilation

Visible by default