Digital Circuits

ECS 371

Dr. Prapun Suksompong

prapun@siit.tu.ac.th

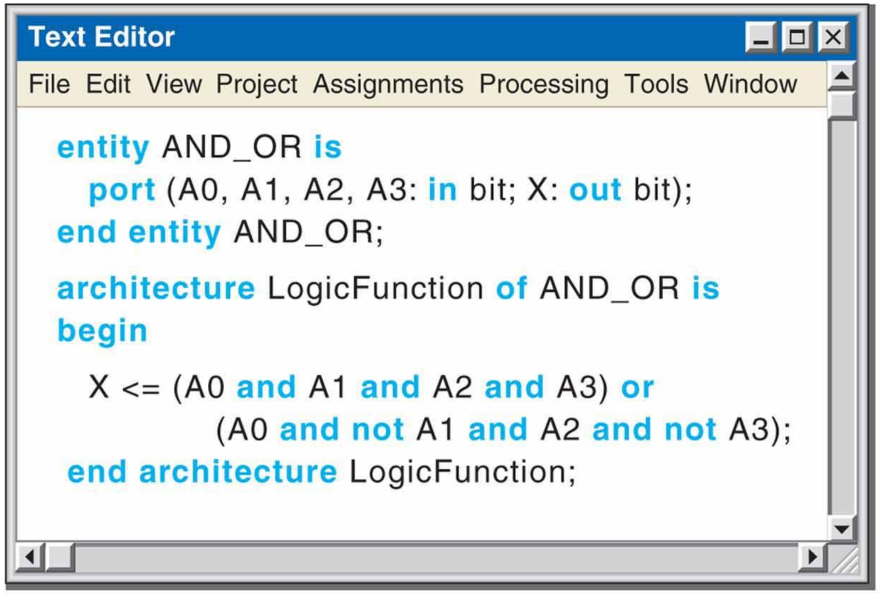
Lecture 27

Office Hours:
BKD 3601-7
Monday 9:00-10:30, 1:30-3:30
Tuesday 10:30-11:30

ECS371.PRAPUN.COM

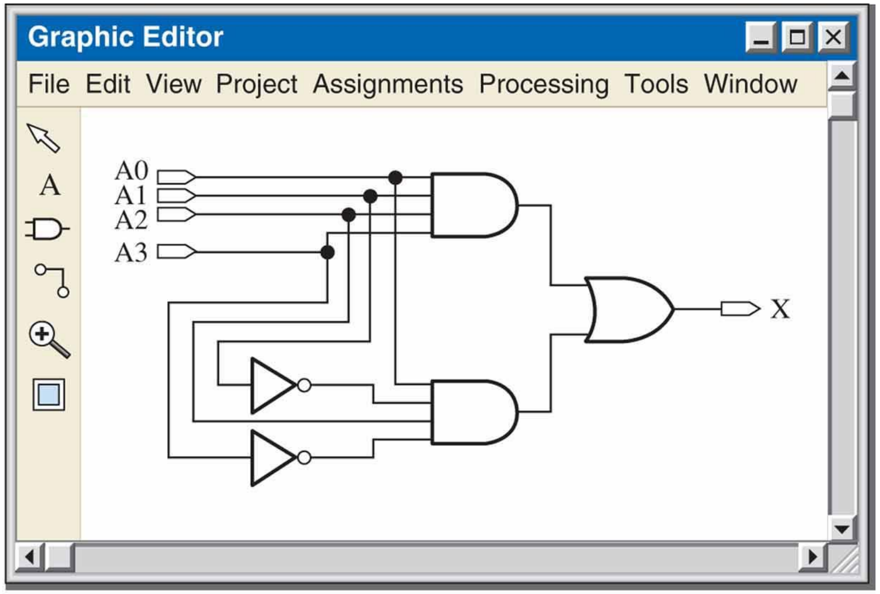
Design Entry

- Examples of text and schematic entry screens



```
entity AND_OR is
   port (A0, A1, A2, A3: in bit; X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
   X <= (A0 and A1 and A2 and A3) or
            (A0 and not A1 and A2 and not A3);
 end architecture LogicFunction;
```

(a) Text entry using VHDL to describe an AND-OR logic circuit



(b) Schematic entry of the same AND-OR logic circuit entered in (a)

- In text entry, the design is entered via a hardware description language such as VHDL

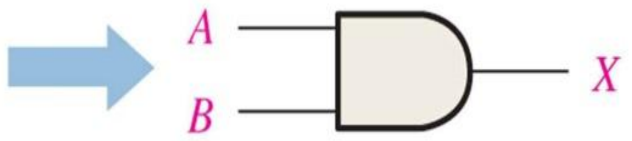
Introduction to VHDL

- VHDL is a language for describing digital electronic systems.
- It arose out of the United States Government's **Very High Speed Integrated Circuits (VHSIC)** program, initiated in 1980.
  - Subsequently adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the US.
- The V stands for VHSIC (Very High Speed Integrated Circuit).
- HDL stands for hardware description language.
  - HDL based design has established itself as the modern approach to design of digital systems
  - VHDL and Verilog HDL are two dominant HDLs.
- Revisions: VHDL-87, VHDL-93, VHDL-2002, VHDL-2008

Example: 2-input AND gate

- It is relatively easy to write programs to describe simple logic circuits in VHDL.
- The logical operators are the following VHDL keywords: and, or, not, nand, nor, xor, and xnor.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```



Three Approaches

- In VHDL, there are three approaches to describing logic:

1. Structural	Description is like a schematic (components and block diagrams).
2. Data flow	Description is equations, such as Boolean operations, and registers.
3. Behavioral	Description is specifications over time (state machines, etc.).

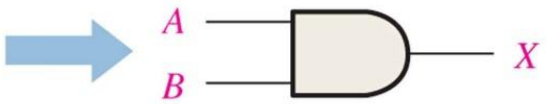
Entity and Architecture

- All VHDL files require an entity declaration and an architecture body.
  1. **Entity**: a declaration of a module's inputs and outputs.
     - Describe a given logic function in terms of its external inputs and outputs, called **ports**.
  2. **Architecture**: a detailed description of the module's internal structure or behavior.
     - Describe the internal operation of the logic function.
- May designers like to think of a VHDL entity declaration as a "wrapper" for the architecture, hiding the details of what's inside while providing the "hooks" for other modules to use it.
- The VHDL file name must be the same as the entity name.

Entity Element

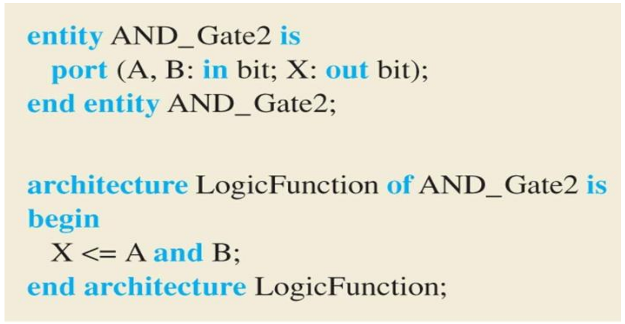
- In its simplest form, the entity element consists of three statements:
  1. The first statement assigns a name to a logic function;
  2. the second statement, called the **port statement** which is indented, specifies the inputs and outputs;
  3. and the third statement is the end statement.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```



Port

- A port in VHDL is a connection from a VHDL design entity to the outside world.
- The direction or directions in which a port may operate is called its **mode**.
- A VHDL port may have one of four modes:
  1. IN (input only),
  2. OUT (output only),
  3. INOUT (bidirectional), and
  4. BUFFER (output, with feedback from the output back into the design entity).
- The mode of a port is declared in the port statement of an entity declaration or component declaration.



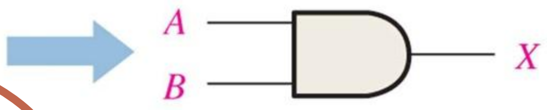
```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```

Architecture Declaration

- The *entity-name* in this definition must be the same as the one given previously in the entity declaration.
- The ***architecture-name*** is a user-selected identifier, usually related to the entity name; it can be the same as the entity name if desired.

```
architecture architecture-name of entity-name is
   type declarations
   signal declarations
   constant declarations
   function definitions
   procedure definitions
   component declarations
begin
   concurrent-statement
   ...
   concurrent-statement
end architecture-name;
```

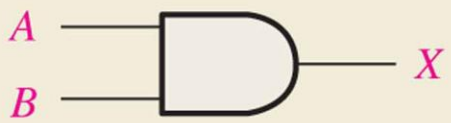
```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```



Recall: Three basic approaches to writing a VHDL program:

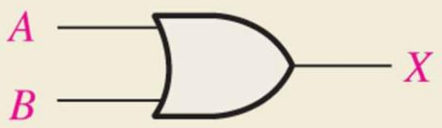
1. Behavioral
2. Data flow
3. Structural

Data Flow Approach



2-input AND gate

```
entity AND_gate is
  port (A, B: in bit; X: out bit);
end entity AND_gate;

architecture ANDfunction of AND_gate is
begin
  X <= A and B;
end architecture ANDfunction;
```

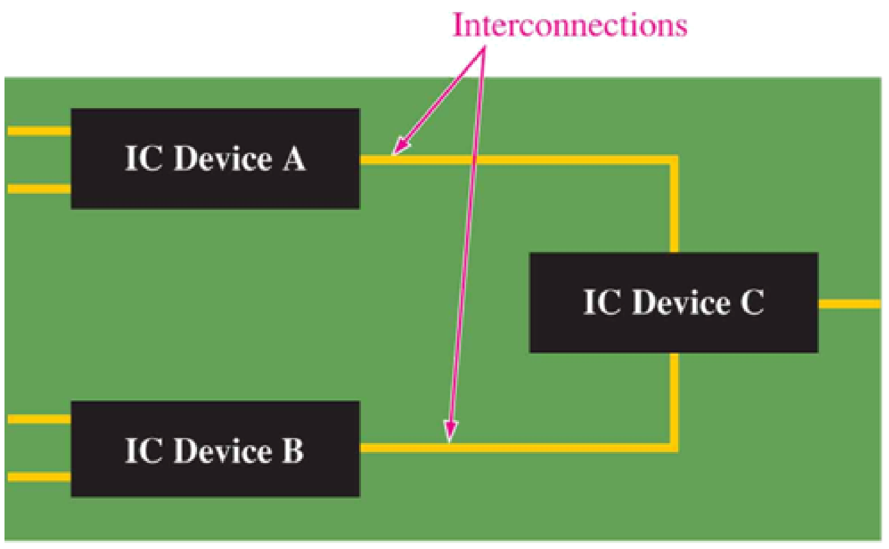


2-input OR gate

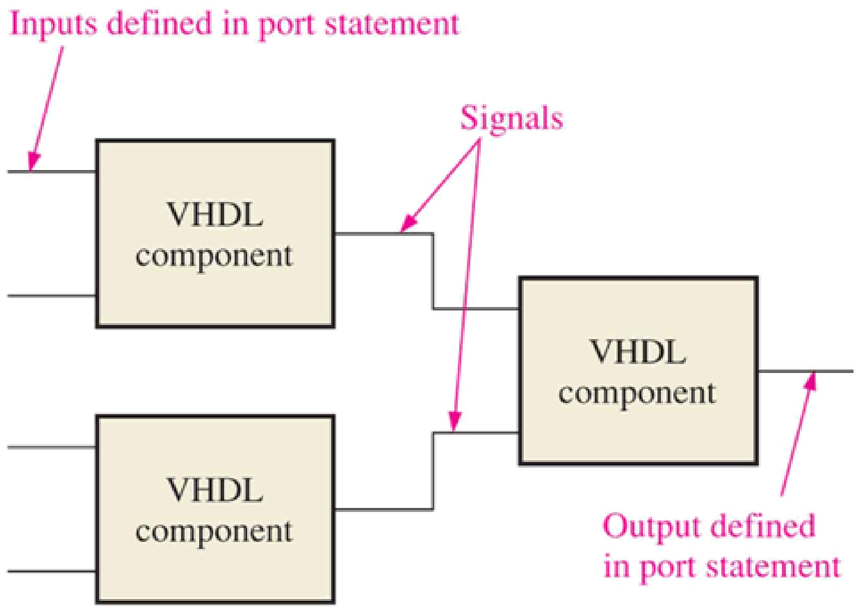
```
entity OR_gate is
  port (A, B: in bit; X: out bit);
end entity OR_gate;

architecture ORfunction of OR_gate is
begin
  X <= A or B;
end architecture ORfunction;
```

Structural Approach

- Can be compared to installing IC devices on a circuit board and interconnecting them with wires.
- Describe logic functions and specify how they are connected together.



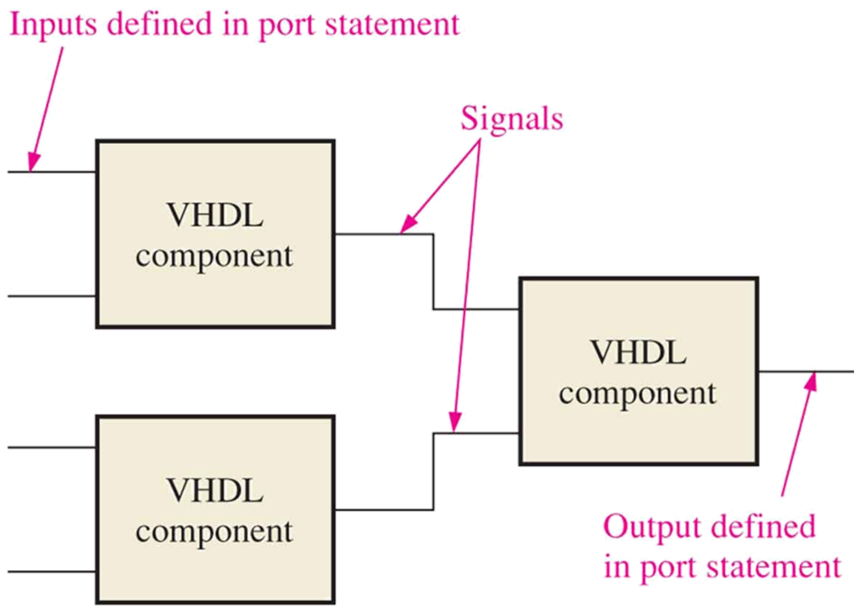
(a) Hardware implementation with fixed-function logic


Inputs defined in port statement
Signals
VHDL component
VHDL component
VHDL component
Output defined in port statement

(b) VHDL structural implementation

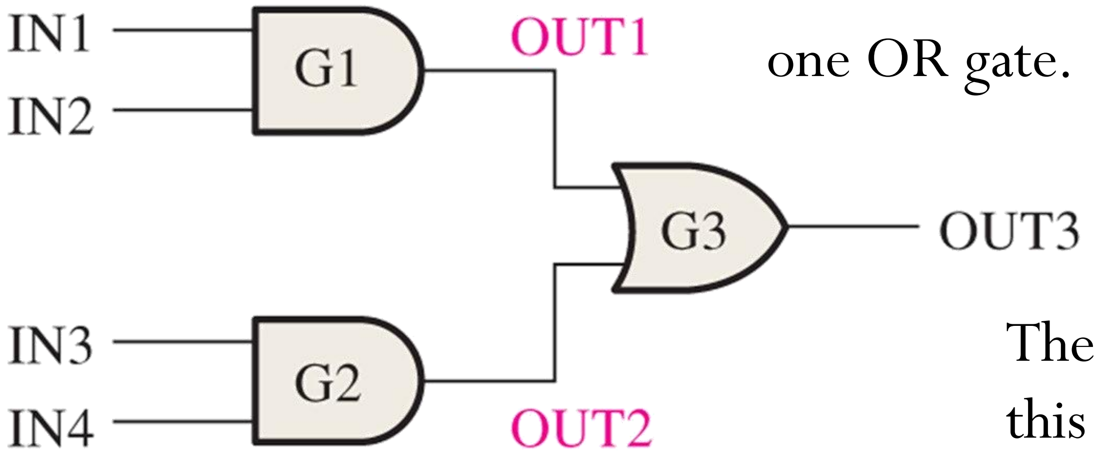
VHDL Components and Signals

- The **VHDL component** is a way to predefine a logic function for repeated use in a program or in other programs.
  - The component can be used to describe anything from a simple logic gate to a complex logic function.
  - For example, can create a component for an AND gate and then use it as many times as you wish
- The **VHDL signal** can be thought of as a way to specify a "wire" connection between components.


Inputs defined in port statement
Signals
VHDL component
VHDL component
VHDL component
Output defined in port statement

Example:

- Goal: Implement a simple SOP logic circuit



This circuit has two AND gates and one OR gate.

```
entity AND_OR_Logic is
   port (IN1, IN2, IN3, IN4: in bit; OUT3: out bit);
end entity AND_OR_Logic;
```

The VHDL program for this circuit will have

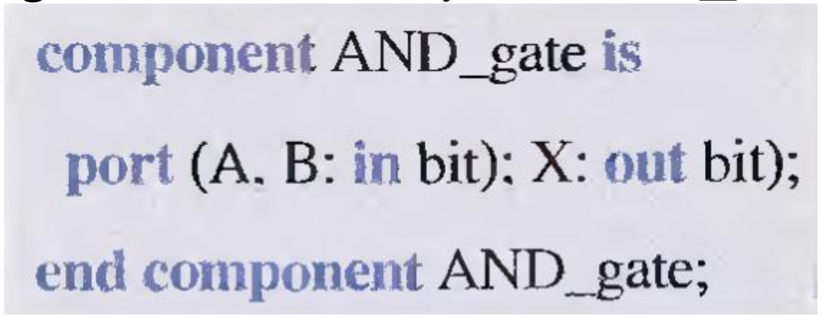
1. two components (AND_gate, OR)
2. two signals (OUT1, OUT2)

Component Declaration

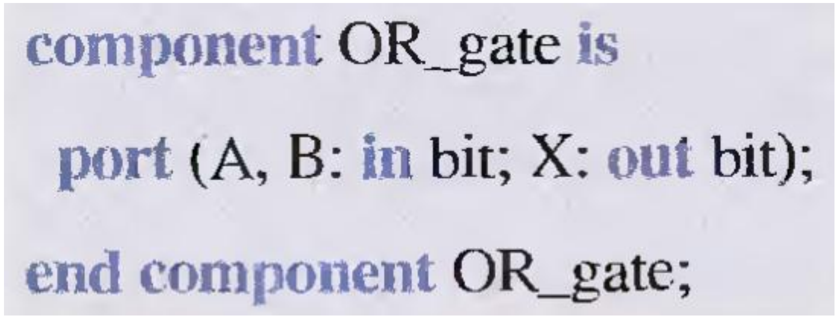
The VHDL program for any logic function can become a component and used whenever necessary in a larger program with the use of a component declaration of the following general form.

```
component name_of_component is
   port (port definitions);
end component name_of_component;
```

Recall that we have previously defined VHDL data flow descriptions of a 2-input AND gate with the entity name `AND_gate` and a 2-input OR gate with the entity name `OR_gate`.



```
component AND_gate is
  port (A. B: in bit): X: out bit);
end component AND_gate;
```



```
component OR_gate is
  port (A, B: in bit; X: out bit);
end component OR_gate;
```

Component Declaration (2)

```
component AND_gate is
  port (A. B: in bit): X: out bit);
end component AND_gate;
```

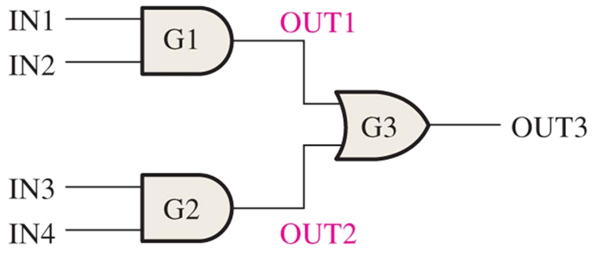
The port statement in the component declaration must correspond to the port statement in the entity declaration of the AND gate.

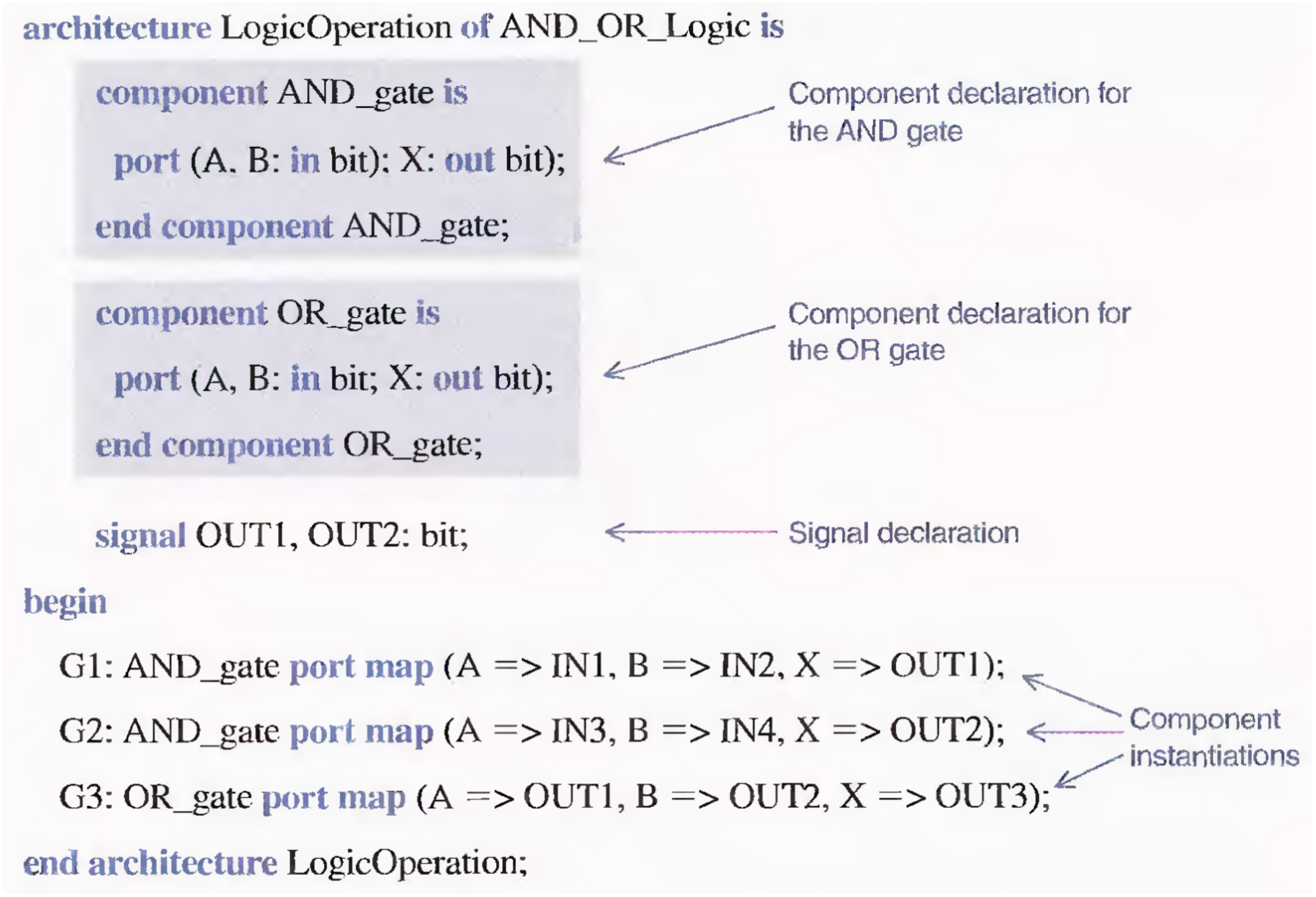
```
component OR_gate is
  port (A, B: in bit; X: out bit);
end component OR_gate;
```

```
entity AND_gate is
  port (A, B: in bit; X: out bit);
end entity AND_gate;

architecture ANDfunction of AND_gate is
begin
  X <= A and B;
end architecture ANDfunction;
```

```
entity OR_gate is
  port (A, B: in bit; X: out bit);
end entity OR_gate;

architecture ORfunction of OR_gate is
begin
  X <= A or B;
end architecture ORfunction;
```

Example (Revisited)





```
architecture LogicOperation of AND_OR_Logic is
    component AND_gate is
      port (A, B: in bit); X: out bit);
    end component AND_gate;

    component OR_gate is
      port (A, B: in bit; X: out bit);
    end component OR_gate;

    signal OUT1, OUT2: bit;
begin
  G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
  G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
  G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

Component declaration for the AND gate

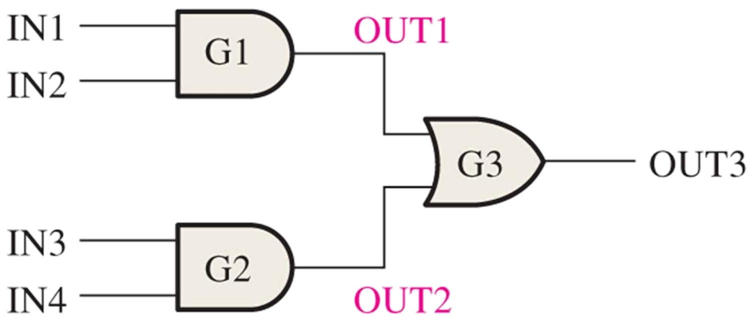
Component declaration for the OR gate

Signal declaration

Component instantiations

Using Signals

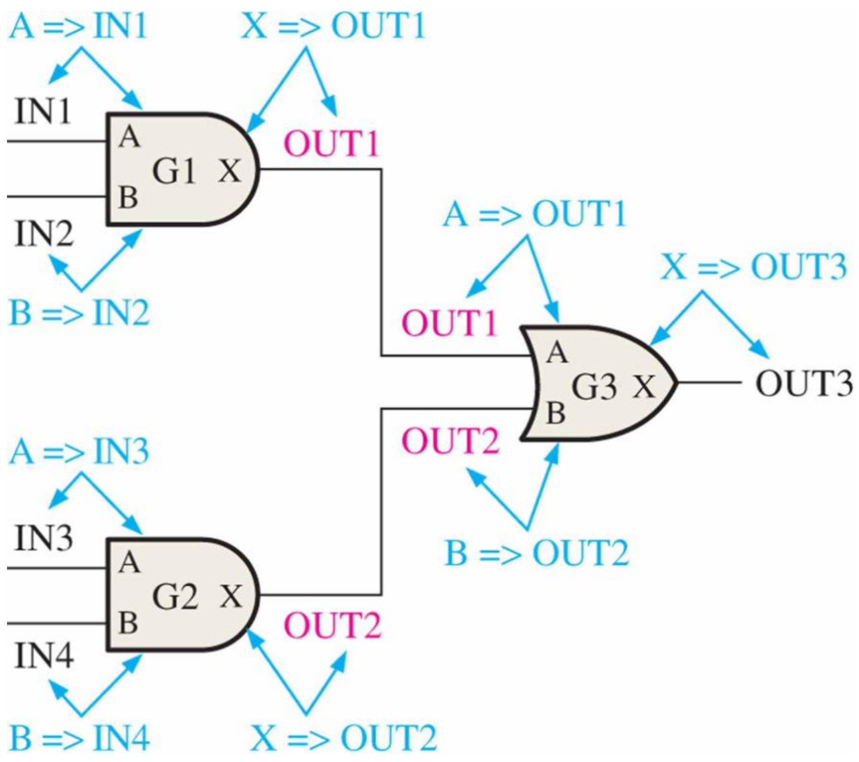
- In VHDL, signals are analogous to **wires** that interconnect components on a circuit board.
- Signals are the **internal connections** in the logic circuit and are treated differently than the inputs and outputs.
- Whereas the inputs and outputs are declared in the entity declaration using the port statement, the signals are ***declared within the architecture*** using the signal statement.

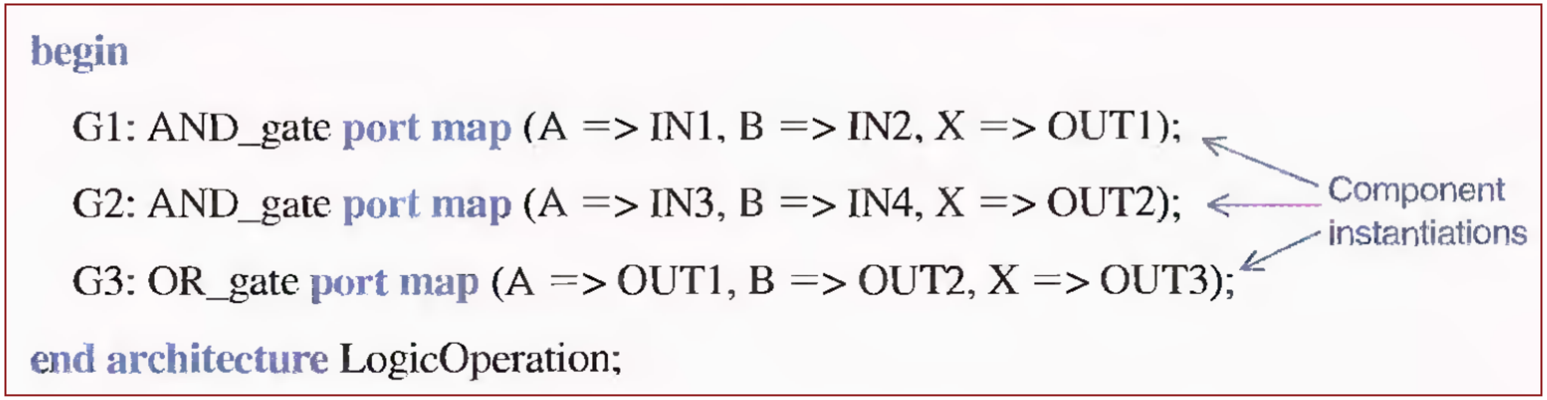


```
signal OUT1, OUT2: bit;
```

Using Components

- Write a **component instantiation** statement for each instance in which the component is used.
- The component instantiations appear between the keyword begin and the end statement.

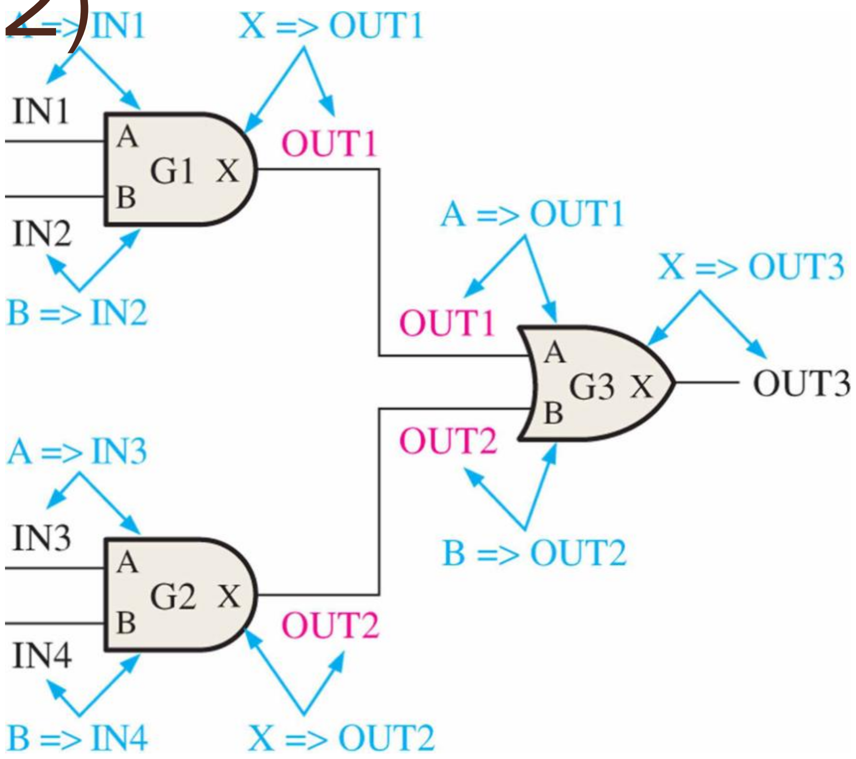


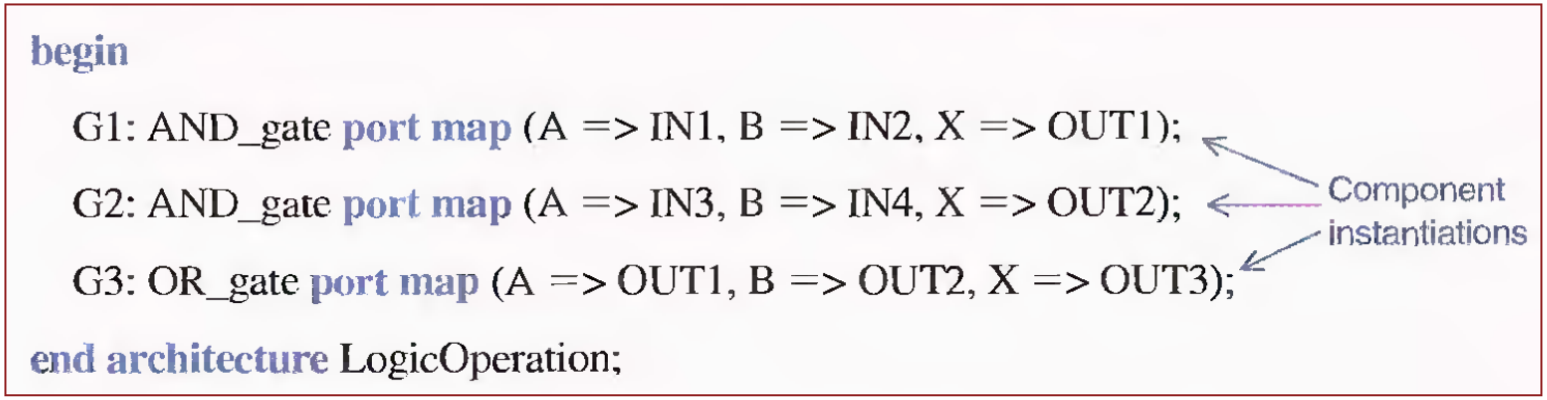


```
begin
   G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
   G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
   G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

Using Components (2)

- For each instantiation an identifier is defined, such as G1, G2, and G3 in this case.
- Then the component name is specified.
- The port map essentially makes all the connections.



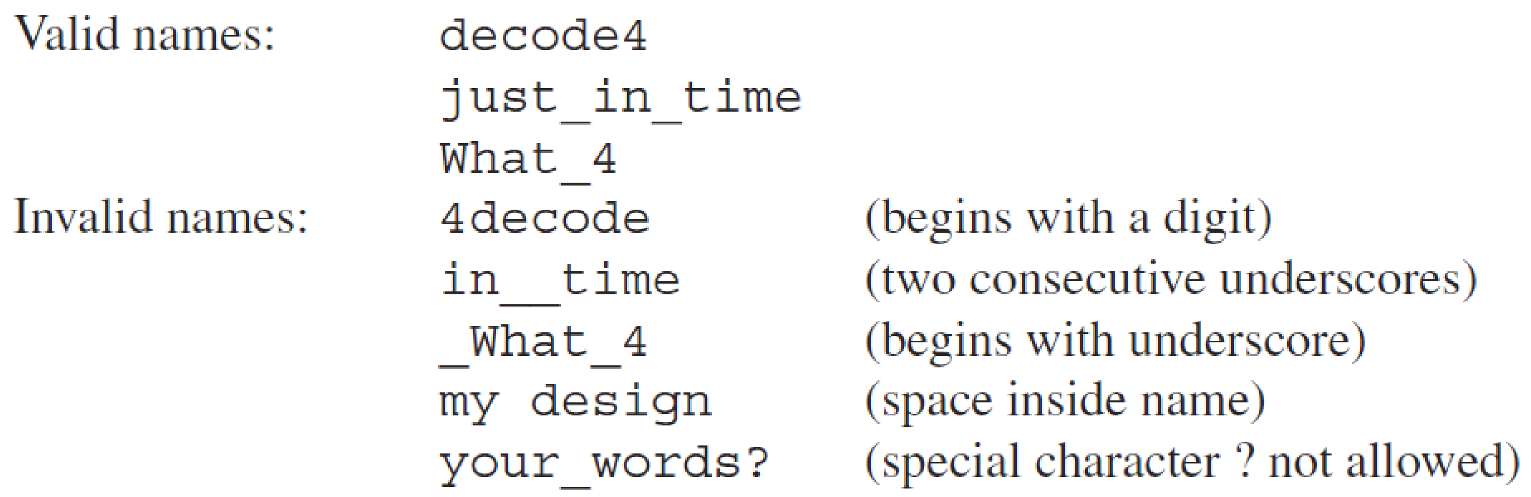


```
begin
    G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
    G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
    G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

Component instantiations

Valid Names

- A valid name in VHDL consists of a letter followed by any number of letters or numbers, without spaces.
- VHDL is not case sensitive.
- An underscore may be used within a name, but may not begin or end the name.
- Two consecutive underscores are not permitted.
- *Example*



Valid names:	`decode4`	
	`just_in_time`	
	`What_4`	
Invalid names:	`4decode`	(begins with a digit)
	`in__time`	(two consecutive underscores)
	`_What_4`	(begins with underscore)
	`my design`	(space inside name)
	`your_words?`	(special character ? not allowed)