Белорусский государственный университет
информатики и радиоэлектроники

4-ый курс специальности ПОИТ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЦИФРОВОГО ПРОЕКТИРОВАНИЯ

2020

Introducing to VHDL

HDL – Hardware Description Language

VHDL, Verilog, SystemVerilog, SystemC, Handel-C, CUPL, AHDL, ABEL, PALASM, etc.

VHDL => VHSIC HDL, VHSIC => Very High Speed Integrated Circuit

ADA, Pascal-like language

U.S Department of Defense in 1980^s

IEEE Standard 1076

Introducing to VHDL

VHDL main features:

1) VHDL supports the whole design process:
   - System Level;
   - RT Level;
   - Circuit Level;
   - Logic gates Level;

2) VHDL is suitable for specification in
   - Behavioral domain;
   - Structural domain.

3) Precise simulation semantics is associated with the VHDL language constructs.

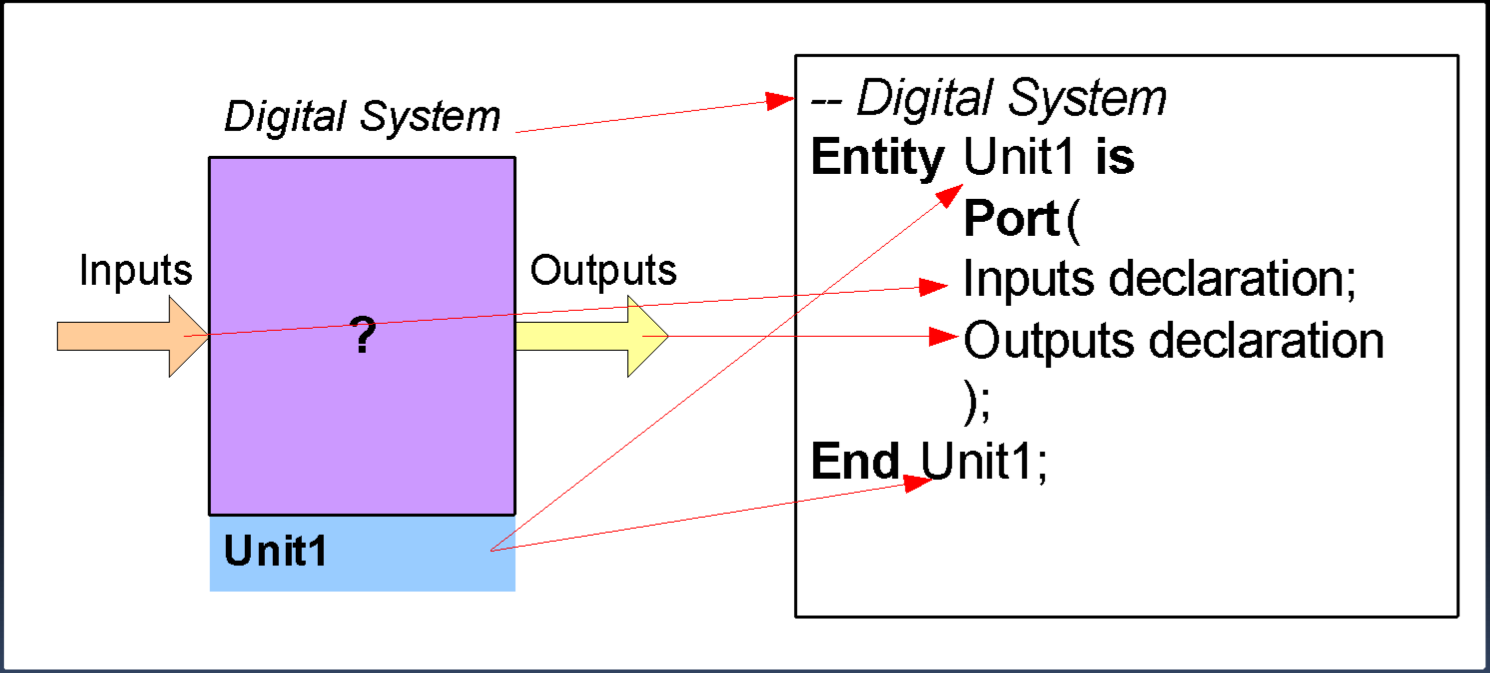
Introducing to VHDL

Synthesized and Not Synthesized descriptions

Behavioral Description	Z<=A and B;	Synthesized description, which uses signals, built-in logic operators, processes as a sequential subroutines and at the same time process can be a description for sequential circuit.
Structural Description	U1: AND_GATE port map (A,B,Z)	Synthesized description, which describes the digital system as a set of components and the interconnections between these components.
Time Description	Z<=A and B after 10 ns;	Not Synthesized description, which describes the behavior of the digital system in time using gates delay slots and signals transportation time.
Mixed Description	Component description as Z<= A and B.	Can be both: synthesized and not synthesized description. Simultaneously uses several types of HDL description.

Introducing to VHDL

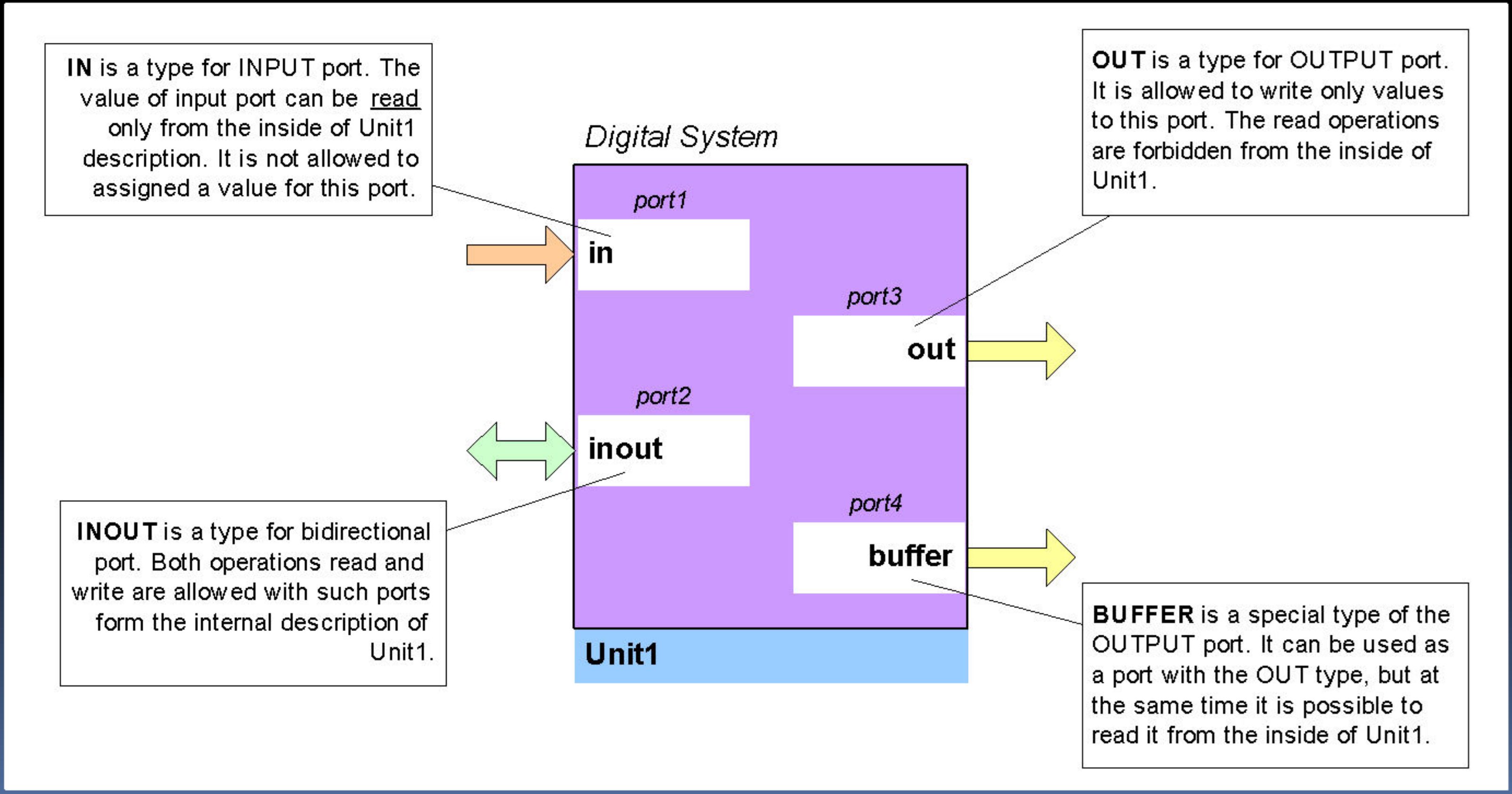
VHDL operates with COMPONENTs. Each COMPONENT has its own ENTITY and ARCHITECTURE.



```
-- Digital System
Entity Unit1 is
         Port(
         Inputs declaration;
         Outputs declaration
         );
End Unit1;
```

Introducing to VHDL

Ports: port is an object, which is used to provide the connection between the internal structure of the component and external environment.



IN is a type for INPUT port. The value of input port can be read only from the inside of Unit1 description. It is not allowed to assigned a value for this port.

OUT is a type for OUTPUT port. It is allowed to write only values to this port. The read operations are forbidden from the inside of Unit1.

INOUT is a type for bidirectional port. Both operations read and write are allowed with such ports form the internal description of Unit1.

BUFFER is a special type of the OUTPUT port. It can be used as a port with the OUT type, but at the same time it is possible to read it from the inside of Unit1.

Introducing to VHDL

Ports declaration : each port should have a type of a signal. There are several signal types in VHDL, but most important is STD_LOGIC type. Any object of this type can assume next values:

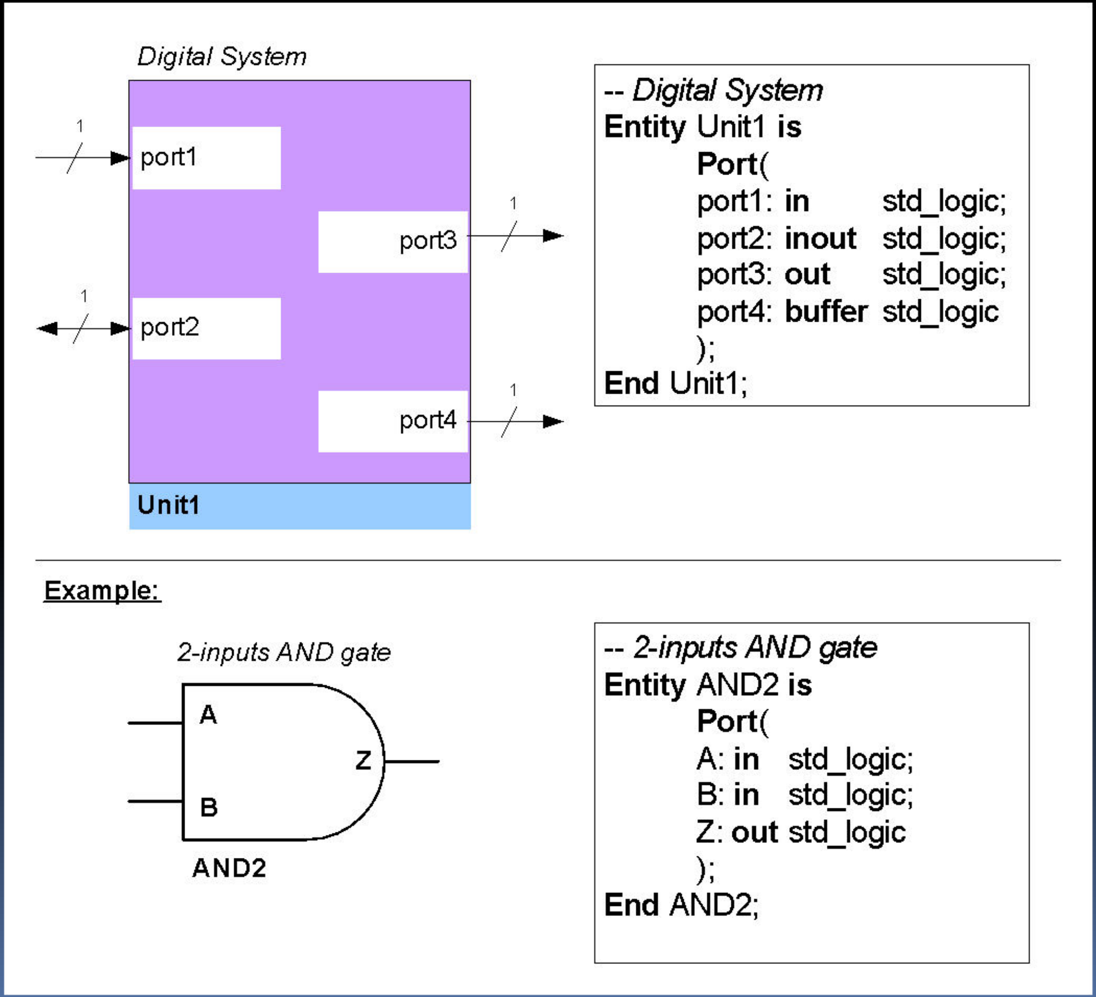
'U' – Uninitialized value;
'X' – Forcing unknown;
'0' – Forcing 0;
'1' – Forcing 1;
'Z' – High impedance;
'W' – Weak unknown;
'L' – Weak 0;
'H' – Weak 1;
'-' – Don't care.

Another example is BIT type, which uses only two values:

'0' – Logic 0;
'1' – Logic 1.

Introducing to VHDL

Ports declaration:



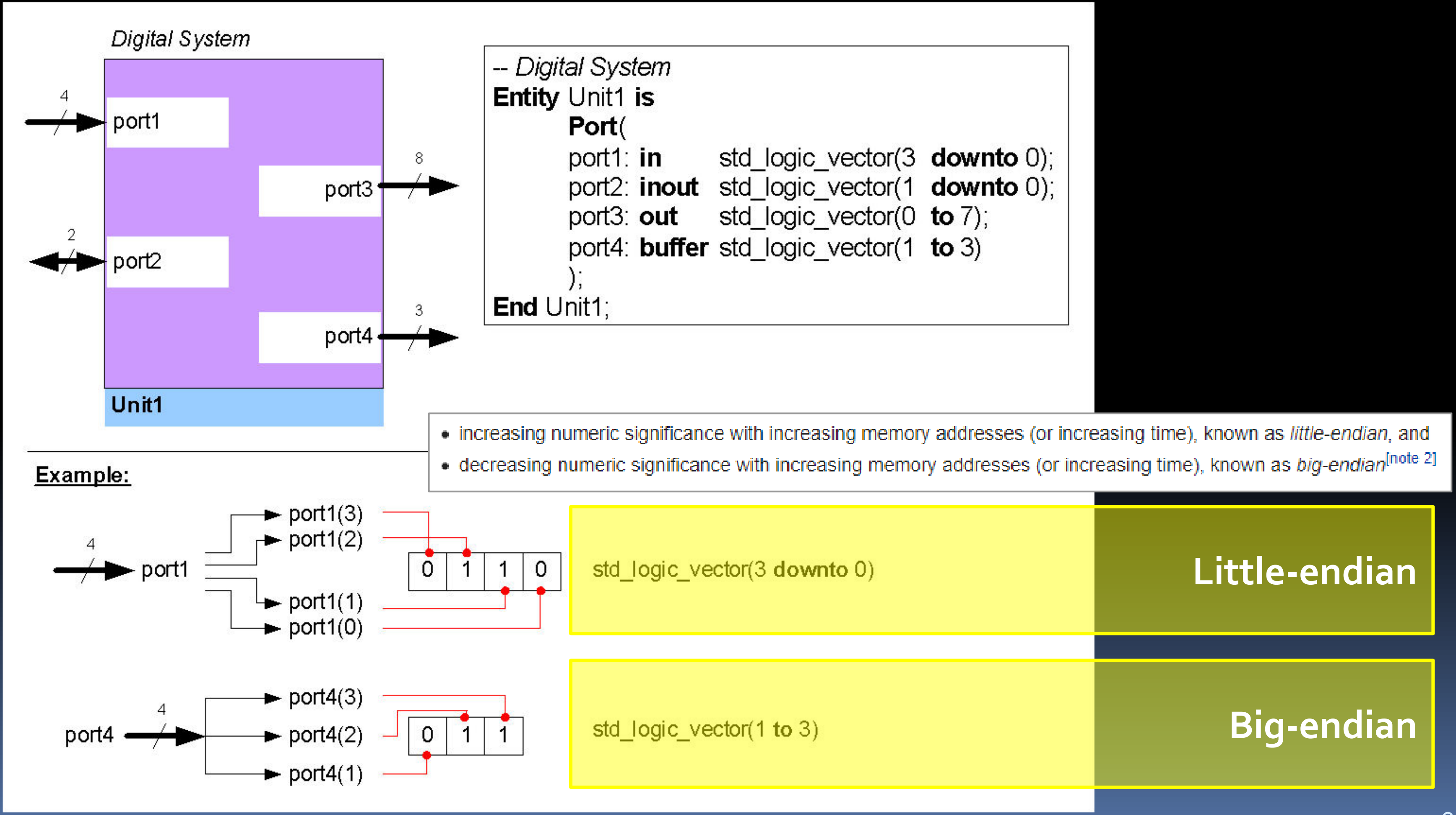
```
-- Digital System
Entity Unit1 is
        Port(
        port1: in       std_logic;
        port2: inout    std_logic;
        port3: out      std_logic;
        port4: buffer   std_logic
        );
End Unit1;
```

Example:

```
-- 2-inputs AND gate
Entity AND2 is
        Port(
        A: in    std_logic;
        B: in    std_logic;
        Z: out  std_logic
        );
End AND2;
```

Introducing to VHDL

Ports declaration: STD_LOGIC_VECTOR type



```
-- Digital System
Entity Unit1 is
        Port(
        port1: in      std_logic_vector(3  downto 0);
        port2: inout   std_logic_vector(1  downto 0);
        port3: out     std_logic_vector(0  to 7);
        port4: buffer  std_logic_vector(1  to 3)
        );
End Unit1;
```

- increasing numeric significance with increasing memory addresses (or increasing time), known as *little-endian*, and
- decreasing numeric significance with increasing memory addresses (or increasing time), known as *big-endian*[note 2]

std_logic_vector(3 **downto** 0)

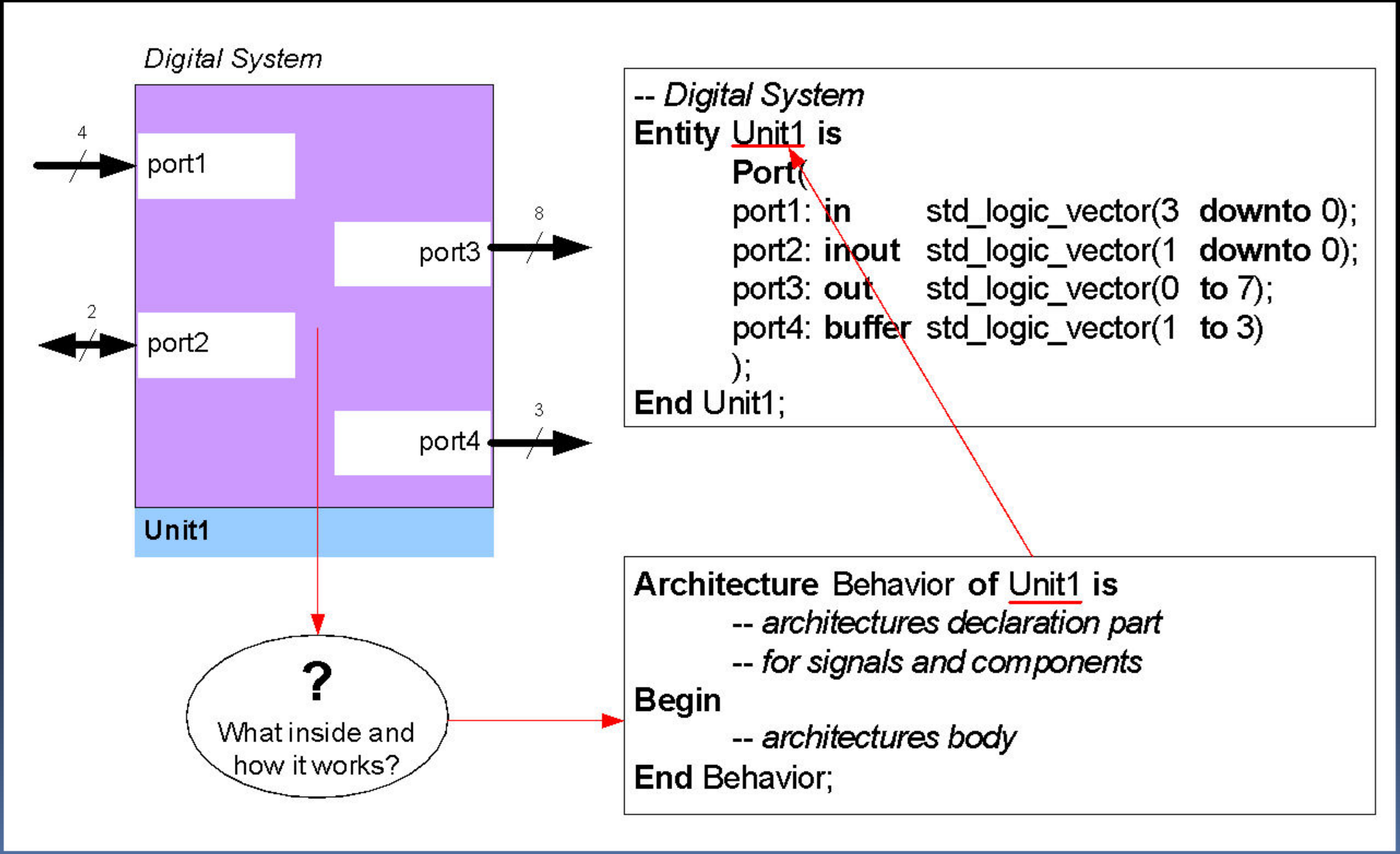
Little-endian

std_logic_vector(1 **to** 3)

Big-endian

Introducing to VHDL

Architecture: this block of VHDL represents the description of declared entity.

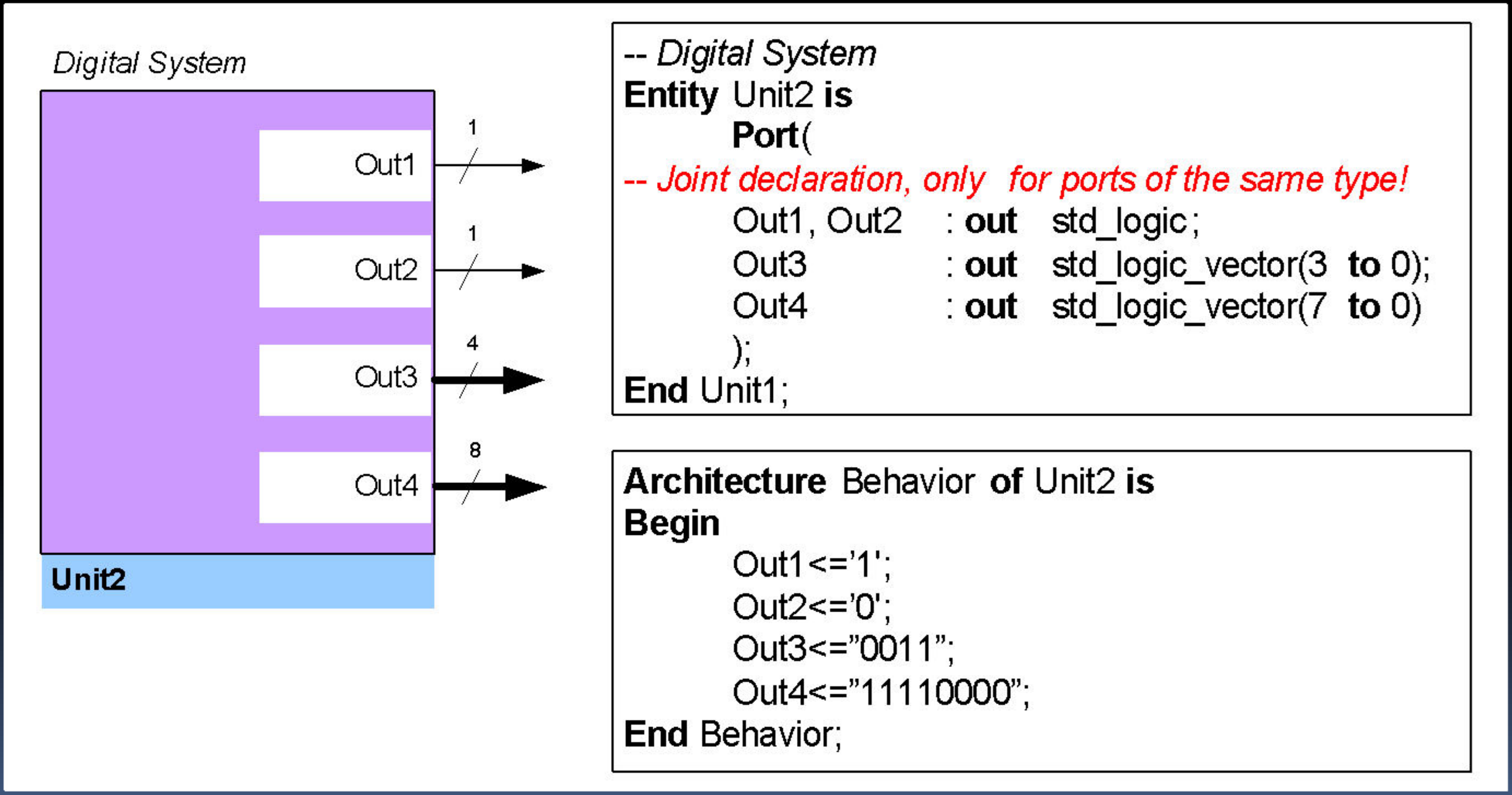


```
-- Digital System
Entity Unit1 is
        Port(
        port1: in     std_logic_vector(3  downto 0);
        port2: inout  std_logic_vector(1  downto 0);
        port3: out    std_logic_vector(0  to 7);
        port4: buffer std_logic_vector(1  to 3)
        );
End Unit1;
```

```
Architecture Behavior of Unit1 is
        -- architectures declaration part
        -- for signals and components
Begin
        -- architectures body
End Behavior;
```

Introducing to VHDL

Architecture:

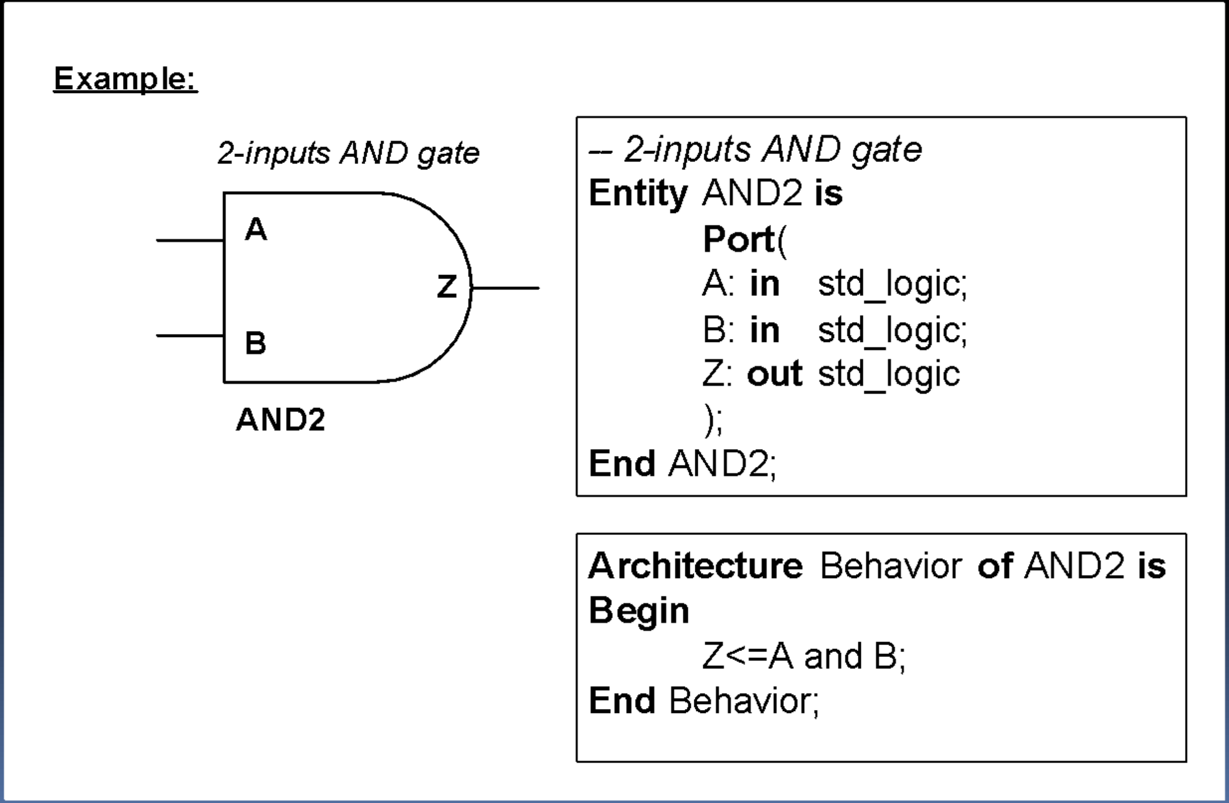


```
-- Digital System
Entity Unit2 is
        Port(
-- Joint declaration, only  for ports of the same type!
        Out1, Out2   : out   std_logic;
        Out3         : out   std_logic_vector(3  to 0);
        Out4         : out   std_logic_vector(7  to 0)
        );
End Unit1;
```

```
Architecture Behavior of Unit2 is
Begin
        Out1<=’1’;
        Out2<=’0’;
        Out3<=”0011”;
        Out4<=”11110000”;
End Behavior;
```

Introducing to VHDL

Architecture: there are several built-in logic operators in VHDL, which can be used for ports and for signals. They are: AND, OR, NOT, XOR, NAND, NOR, XNOR.



Example:

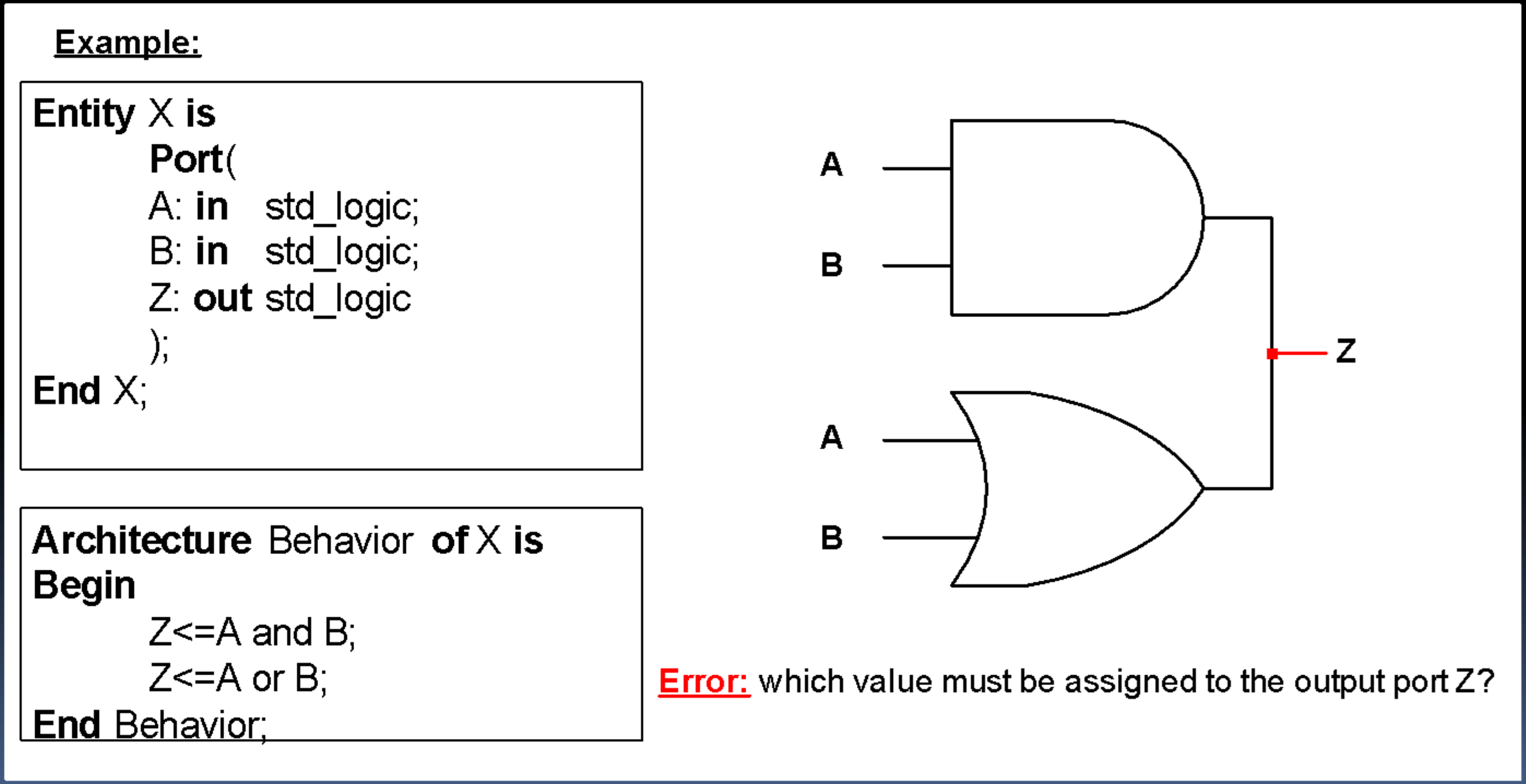
2-inputs AND gate
A
B
Z
AND2

```
-- 2-inputs AND gate
Entity AND2 is
          Port(
          A: in   std_logic;
          B: in   std_logic;
          Z: out std_logic
          );
End AND2;
```

```
Architecture Behavior of AND2 is
Begin
          Z<=A and B;
End Behavior;
```

Introducing to VHDL

Architecture: each expression inside the architectures body is parallel expression.



Example:

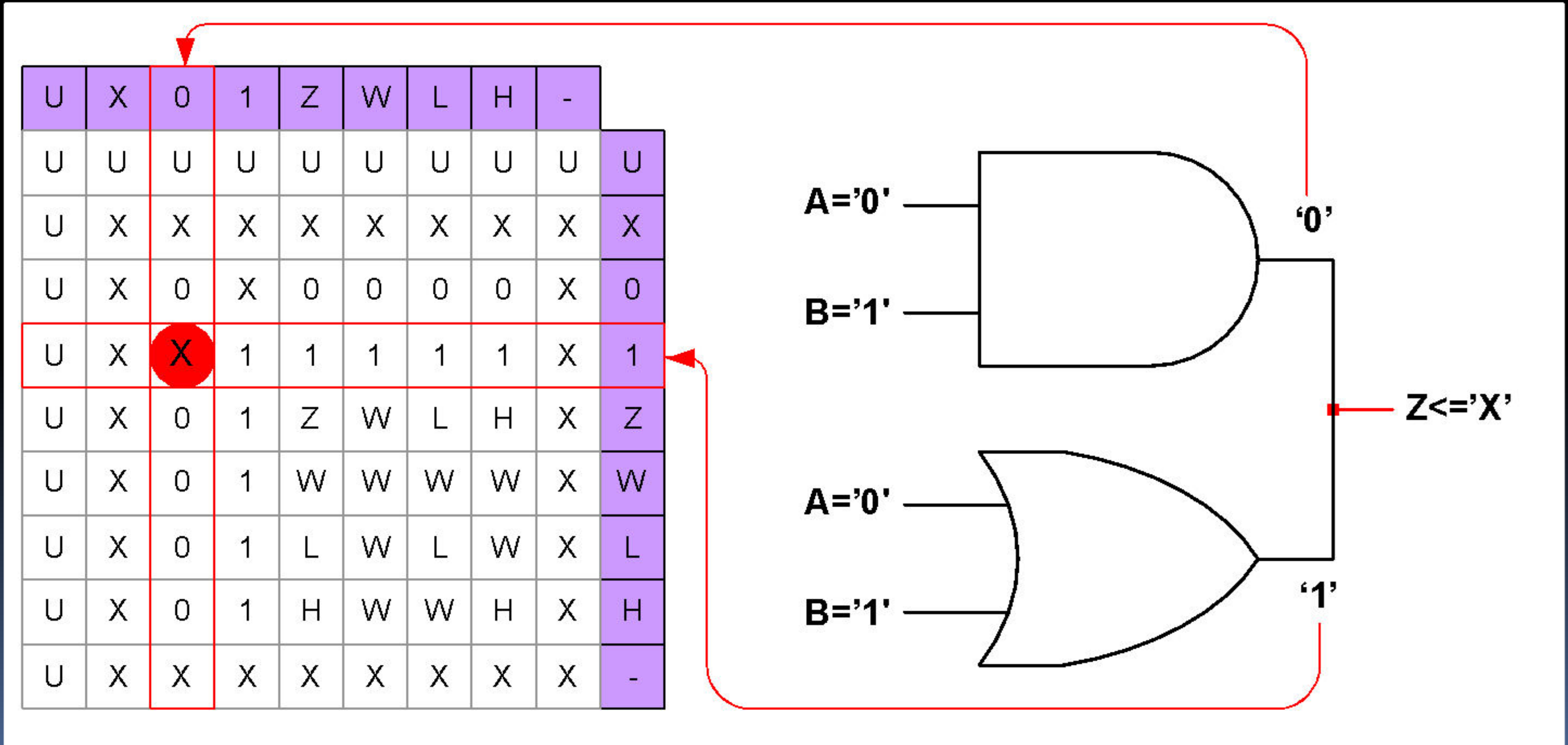
```
Entity X is
        Port(
        A: in   std_logic;
        B: in   std_logic;
        Z: out std_logic
        );
End X;
```

```
Architecture Behavior of X is
Begin
        Z<=A and B;
        Z<=A or B;
End Behavior;
```

Error: which value must be assigned to the output port Z?

Introducing to VHDL

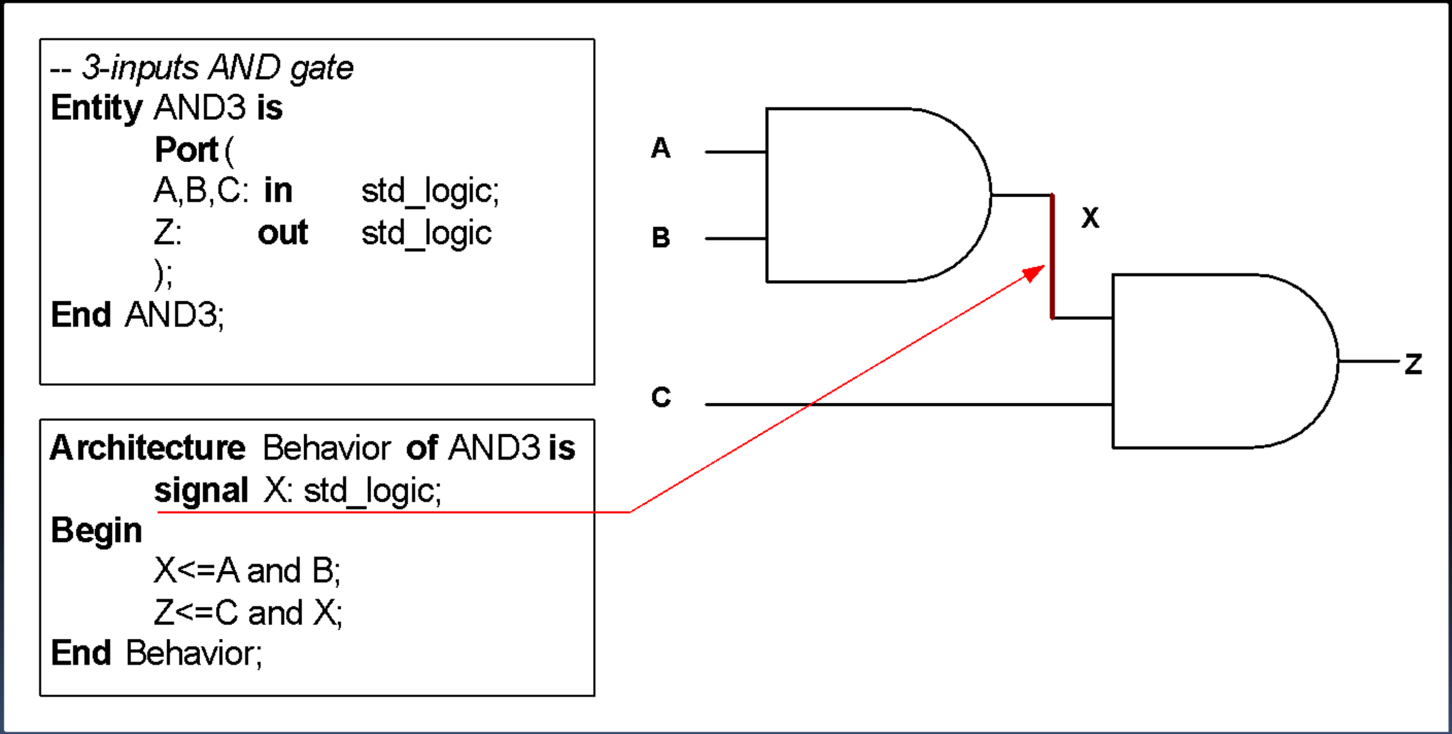
Architecture: for STD_LOGIC type there are special rules how to simulate the situations with the multiple drivers for one port or signal. These rules are represented as the Resolution Table for STD_LOGIC objects:



U	X	0	1	Z	W	L	H	-	
U	U	U	U	U	U	U	U	U	U
U	X	X	X	X	X	X	X	X	X
U	X	0	X	0	0	0	0	X	0
U	X	X	1	1	1	1	1	X	1
U	X	0	1	Z	W	L	H	X	Z
U	X	0	1	W	W	W	W	X	W
U	X	0	1	L	W	L	W	X	L
U	X	0	1	H	W	W	H	X	H
U	X	X	X	X	X	X	X	X	-

Introducing to VHDL

Architecture: special objects – signals

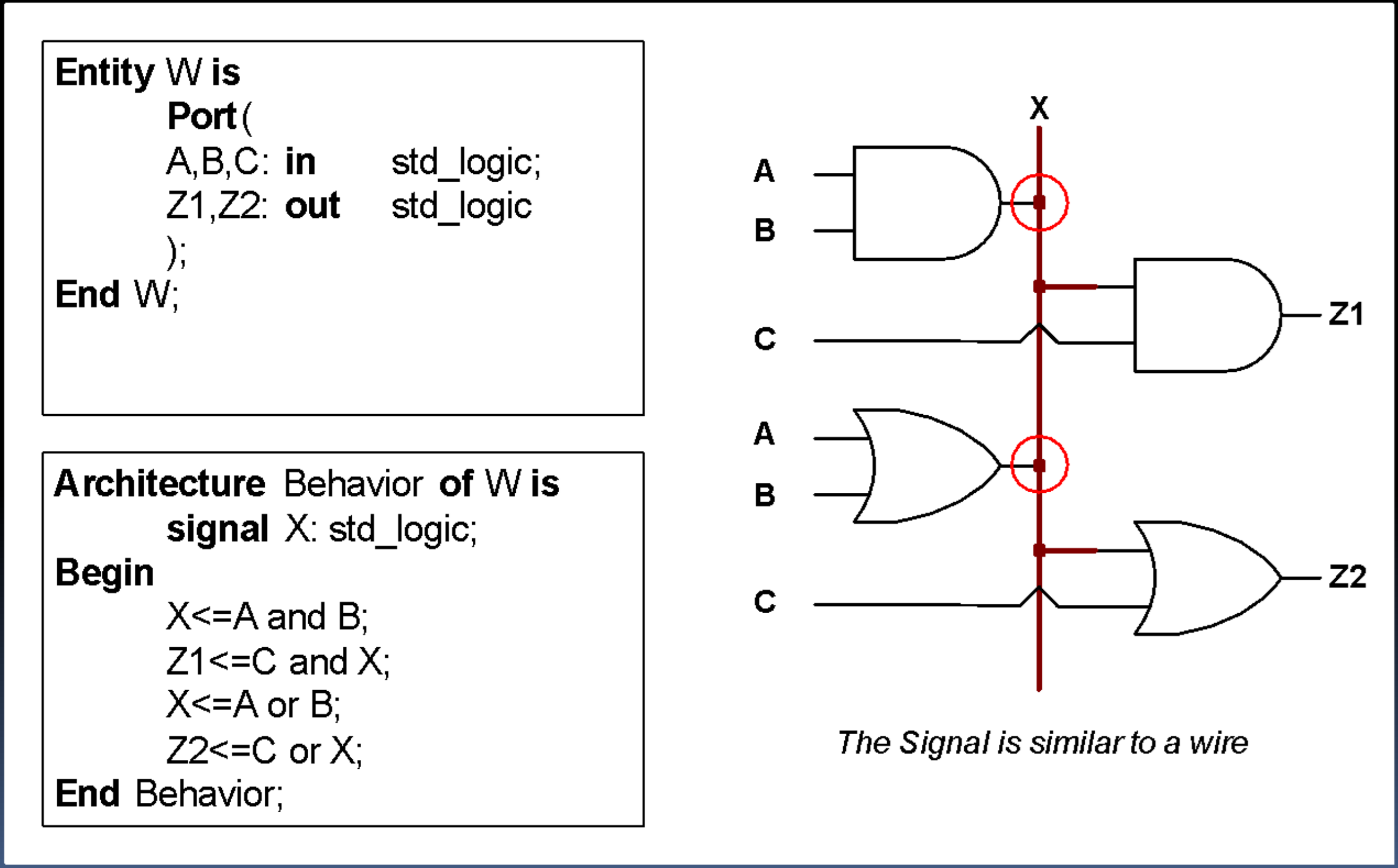


```
-- 3-inputs AND gate
Entity AND3 is
        Port (
        A,B,C:  in      std_logic;
        Z:      out     std_logic
        );
End AND3;
```

```
Architecture Behavior of AND3 is
        signal X: std_logic;
Begin
        X<=A and B;
        Z<=C and X;
End Behavior;
```

Introducing to VHDL

Architecture: signals are similar to wires (parallel objects)



```
Entity W is
        Port (
        A,B,C:  in      std_logic;
        Z1,Z2:  out     std_logic
        );
End W;
```

```
Architecture  Behavior  of W is
        signal  X: std_logic;
Begin
        X<=A and B;
        Z1<=C and X;
        X<=A or B;
        Z2<=C or X;
End Behavior;
```

The Signal is similar to a wire