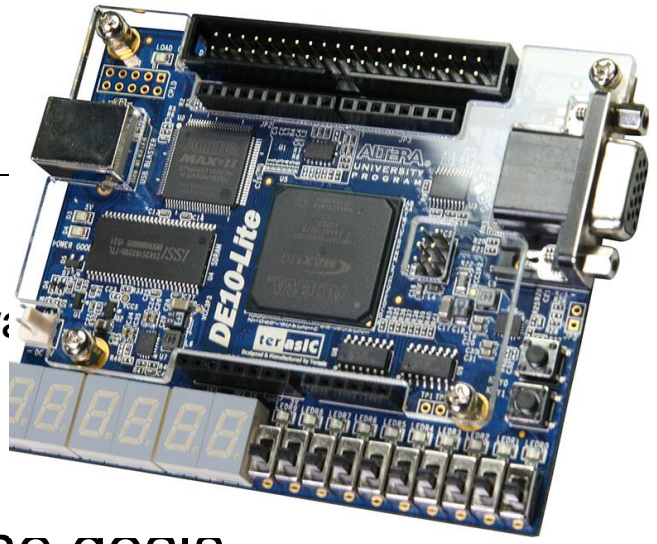
VHDL-Overview

Christophe Bobda

VHDL



VHDL: Very high speed integrated circuits **H**ardwa

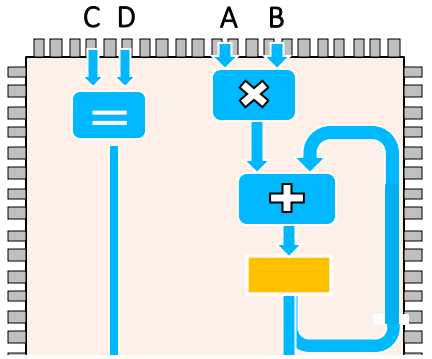
~~Very Hard Design Language~~

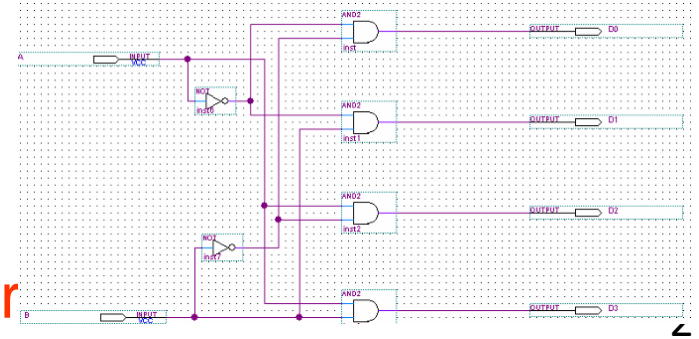
Hardware-Description Language with the goals

- Simulation
- IC synthesis

Description is done under the aspects

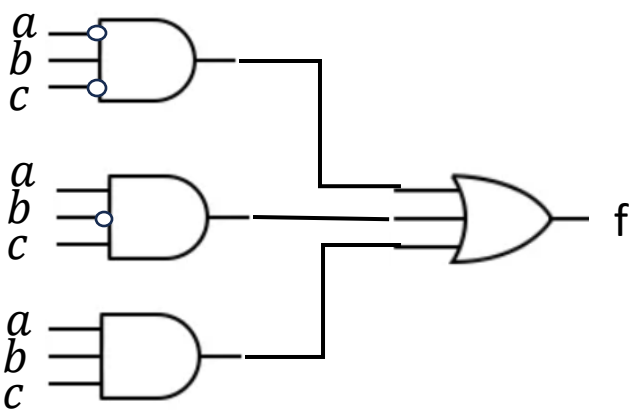
- Behavior
- Structure
- Hierarchy
- Parallelism

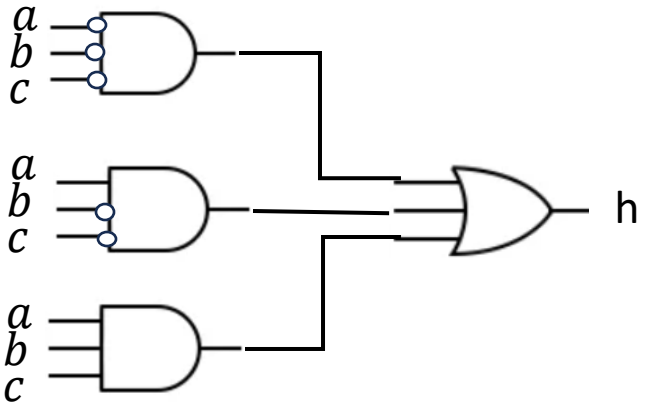




Only synthesizable Subset is consider

a	b	c	f	h
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1





f = $\bar{a}b\bar{c} + a\bar{b}c + abc$

h = $\bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + abc$

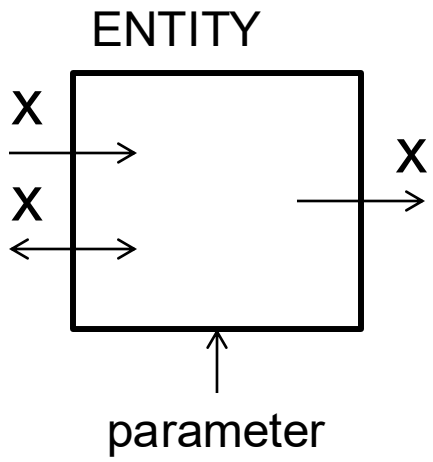
a	b	f	g	h
0	0	1	0	1
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1

$f = \bar{a}\bar{b} + a\bar{b}$

g $= a\bar{b} + ab$

h $= \bar{a}\bar{b} + \bar{a}b + ab$

Entity – Wrapper and interface definition



logic encapsulation of a unit Interface

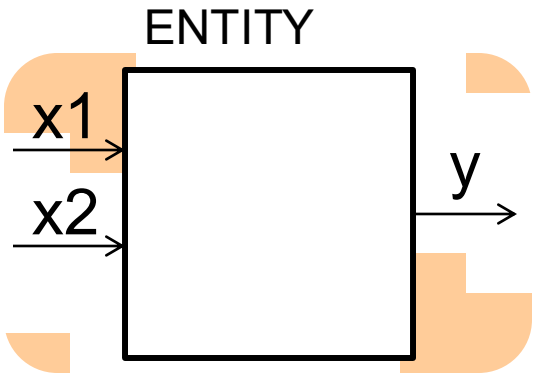
Definition: **Port** → X

Port are defined through **Identifier**,

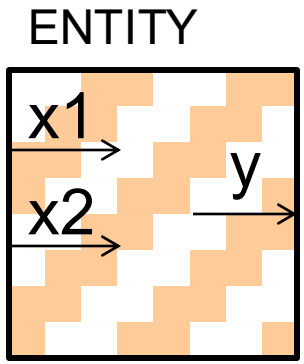
Direction und **Data-type**

Direction**: in, out, inout, buffer**

Data-type: bit, std_logic(_vector), ADT



View from outside



View from inside

Signals can be connected to ports from inside or from outside.

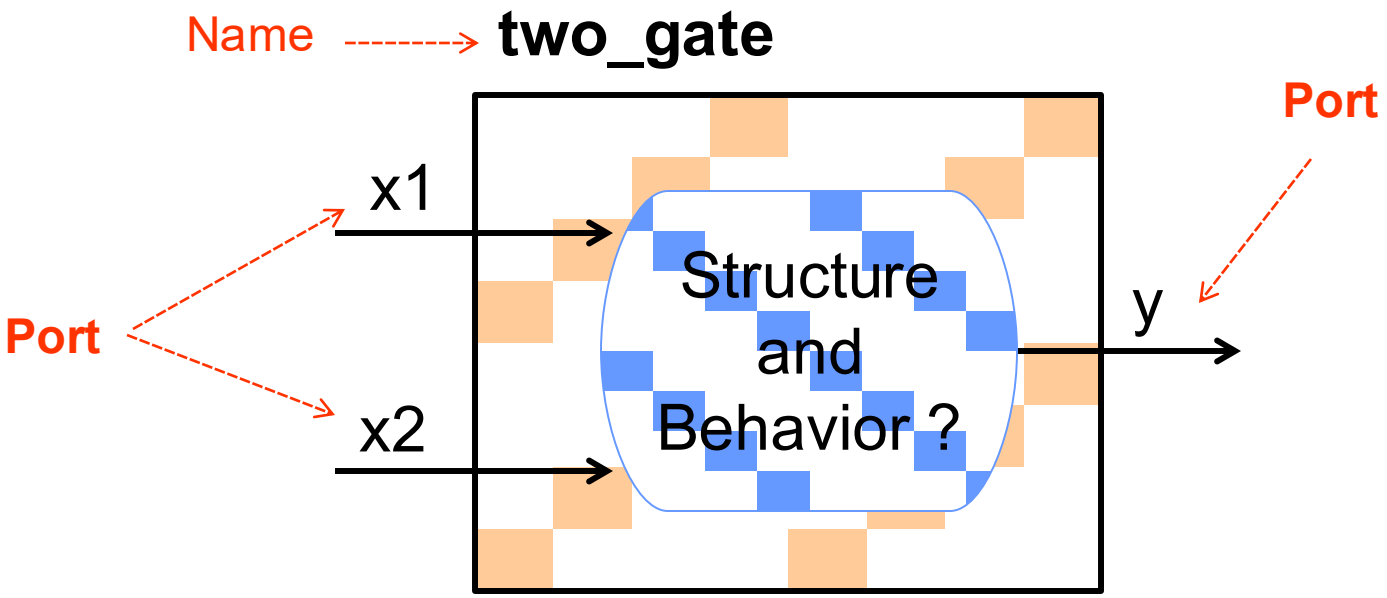
Entity as „Black Box“

Viewed from outside an entity is a „black box“

→ The structure and behavior (functionality) is hidden

Example below: Entity with two inputs and one output port

Functionality not defined yet



Entity – Definition

Keywords in blue

Library to be used

Which part of the library should be

Prefix

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

Entity-Definition

Name of the Entity

Two input-signals with names: x1 and x2

Interface

```
ENTITY two_gate IS
PORT ( x1, x2 : IN  std_logic;
       y      : OUT std_logic );
END two_gate;
```

Name of the Entity

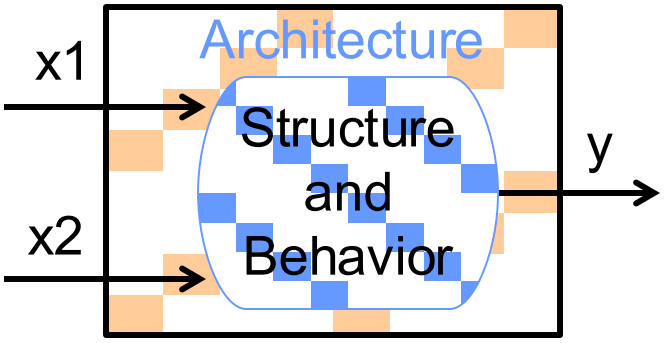
Output-Signal with name: y

Data-type of the signal (multi-value logic)

VHDL is case insensitive !

Architecture – Behavior of the Entity

The behavior of an Entity is described in the **Architecture**.



The Architecture

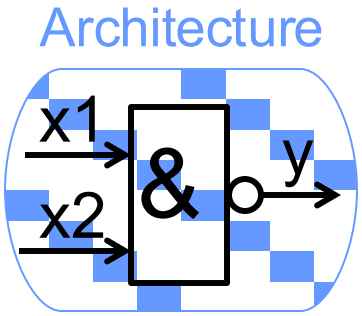
- has a unique name,
- signals and components can be internally declared,
- is assigned to a unique Entity.

```
ARCHITECTURE <Architecture name> OF <Entity-Name> IS
[Declarations]
BEGIN
    [Architecture Statements]
END <Architecture name>;
```

Architecture – Example

NAND-Gate

based on the previous Entity-Definition:



Signal assignment

Architecture name

Entity name

```
ARCHITECTURE two_gate_nand OF two_gate IS
BEGIN

    y <= NOT ( x1 AND x2 );

END two_gate_nand;
```

Behavioral description:

one statement with pre-defined and pre-implemented functions: AND, OR, NOT

Architecture name

XOR-Entity

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


ENTITY xor_gate IS

PORT( x1, x2    : IN      std_logic;
          y     : OUT     std_logic );
END xor_gate;

ARCHITECTURE behavioral OF xor_gate IS
BEGIN

    y <=  x1 XOR x2;

END behavioral;
```

VHDL Operators

Signal Assignment: y <= x1

Boolean Operators:

- AND/and: logical and
- OR/or: logical or
- NAND/nand: logical complement of and
- NOR/ nor: logical complement
- XOR/xor: logical exclusive
- XNOR/ xnor : logical complement of exclusive

Random Boolean statement can be specified in VHDL.
Use parentheses to enforce precedence

```
y <=  (x1 XOR NOT(x2 XOR X3)) NAND y;
```

Process – Parallel statements

VHDL describes parallel processes

- All signal assignments take place simultaneously

VHDL		VHDL
A <= B; C <= D;	=	C <= D; A <= B;

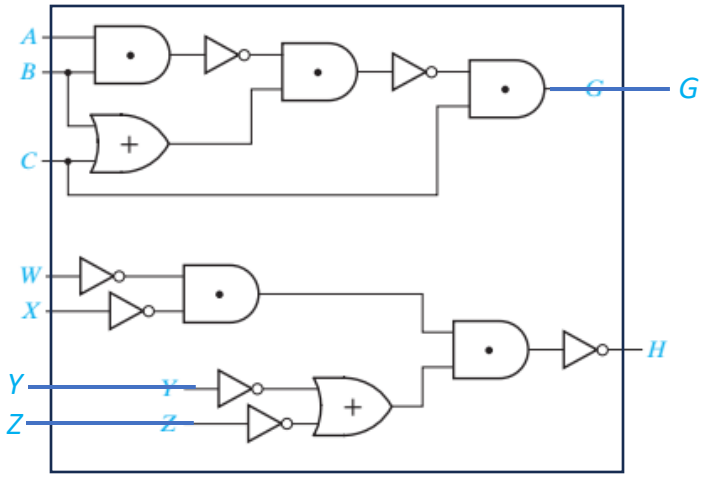
VHDL		JAVA
A <= B; C <= A;	≠	A = B; C = A;

Example

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

ENTITY my_circuit IS

PORT( a, b       : IN       std_logic;
          x, y, z    : OUT    std_logic );
END my_circuit ;

ARCHITECTURE behavioral OF my_circuit IS
BEGIN
    y <=  a XOR (a AND NOT(b));
    z <=  b OR NOT b;
    x <=  b OR NOT b;
END behavioral;
```

Example

Provide the VHDL design for the circuit below




```
ENTITY example2 IS
PORT( A : IN              std_logic;
      w,y, b,c : IN       std_logic;
      z,x      : IN       std_logic;
      H     : OUT         std_logic
      g        : OUT      std_logic );
END entity example2;
```

Entity Declaration

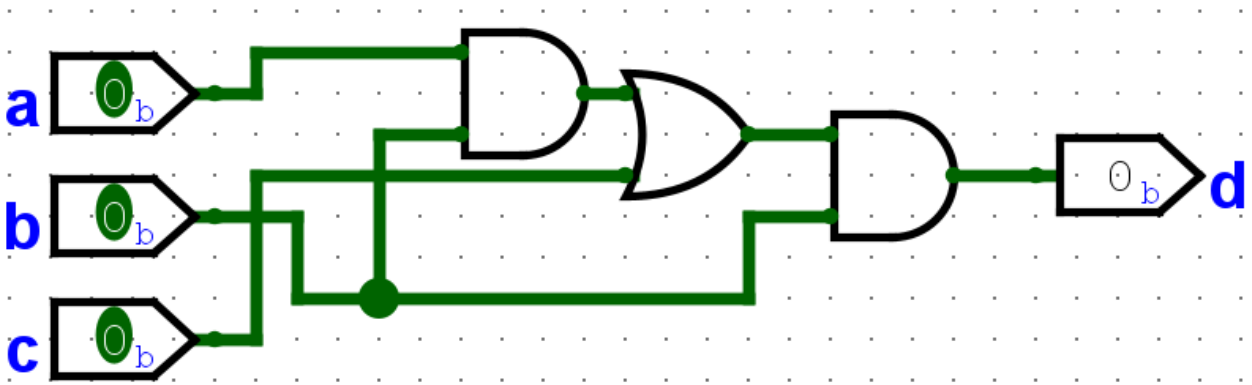
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY example2 IS
PORT( A, B, c : IN        std_logic;
      w,y     : IN        std_logic;
      z       : IN        std_logic;
      x       : IN        std_logic;
      H, g    : OUT       std_logic );
END example2;
```

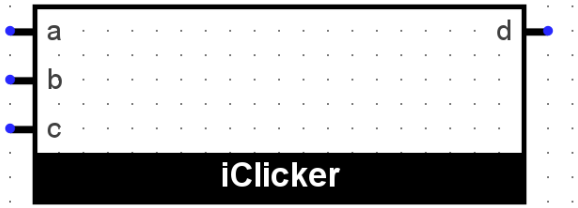
Architecture Implementation

```
architecture structural arc IS
begin

g <=  c AND NOT((b OR c) AND (NOT(a AND b))));
H <= NOT(NOT Y OR NOT z) AND (NOT W AND NOT X));

END architecture;
```

iClicker Question

Given the following circuit diagram, choose the correct implementation in VHDL:






```
library ieee;
use ieee.std_logic_1164.all;

entity iClicker is
port (      a : in std_logic;
            b : in std_logic;
            c : in std_logic;
            d : out std_logic
      );
end iClicker;
```

```
architecture structural is
begin
```

A `d <= a and b nor c`

B `d <= (a and c) or (b and c)`

C `d <= ((a and b) or c ) and b)`

D `d <= a and b`

```
end architecture;
```

Design Flow

Modern design is done using CAD (computer-aided design) or EDA (Electronic Design Automation) tools

- FPGA: AMD Vivado, ~~Intel~~ Altera Quartus
- VLSI: Synopsys DC, Cadence,

Design Flow: See Lab0

- Design Entry
- Functional Simulation
- Synthesis (Pins and Library assignment, compilation)
- Post-synthesis simulation (timing analysis, verification)
- Place and Route (Layout)
- Fabrication/Bitstream Download (FPGA)

Signals

- A signal defines a connection in hardware. Can be used to connect components within an entity
- A signal has a datatype, which is the of possible values + operations on those values
- Signal Declaration

  SIGNAL *signal_name1[, signal_name2, …]* : *signal_type*;

- Declaration can be done in
  - Package
  - Port-section of an entity
  - Declaration section of an entity
- Signal Assignment is performed with symbol <=

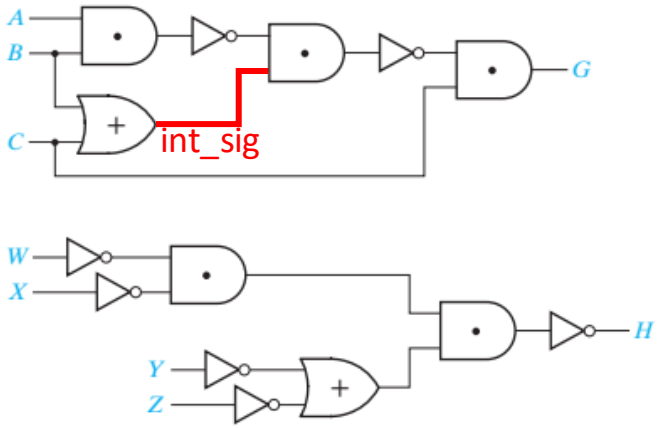
  a <= b xor c

Internal Signals

```
ENTITY Position IS
PORT(   A          : in   std_logic_vector( 2 downto 0 );
        B          : out std_logic_vector( 0 to 2 )  );
END Position;

ARCHITECTURE behv OF Position IS
SIGNAL signal1, signal2: std_logic;
SIGNAL signal3 : std_logic_vector(4 downto 0);

Begin
signal1 <= A(0) AND B(2);
B(0) <= signal1 XOR A(1);
...
```

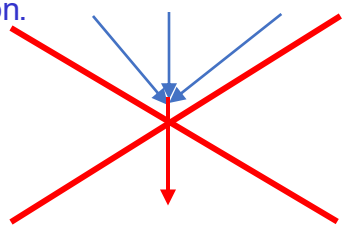
Example

Provide the VHDL design for the circuit below



Signals not connected are floating and will be removed by the compiler

Assigning multiple values to a signal simultaneously will creat and multi-drive situation.



Entity Declaration

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

ENTITY example2 IS
PORT( A, B, c : IN      std_logic;
      w,y     : IN      std_logic;
      z       : IN      std_logic;
      x       : IN      std_logic;
      H, g    : OUT     std_logic );
END example2;
```

Architecture Implementation

Write(assign a value) to the signal

Read from the signal

```
architecture structural arc IS
  signal int_sig: std_logic;
begin
   int_sig <= (b OR c)
   g <=  c AND (int_sig AND (NOT(a AND b)));
   H <= NOT(NOT Y OR NOT z) AND (NOT W AND NOT X));
 END architecture;
```

Variables

- A variable defines a memory location. Can be used to store values.
- A variable has a datatype, which is the set of all possible values + operations on those values
- Variable Declaration

  VARIABLE *var_name1[, var_name2, …]* : *variable_type*;

- Declaration can be done exclusively in a process
- Variable assignment is performed with symbol :=

  a := b xor c

```
PROCESS(a, b)
  VARIABLE c: bit;

  c := b and c;
```

Datatypes

Datatype

- Define a set of values a signal or a variable can take, along with operations on those values.
- Predefined datatype
  - bit (std_logic): '0', '1'
  - bit_vector (std_logic_vector): "00101", "001001"
  - integer: 10, 123, 56
  - character: 'a', 'v', 'n'
  - string: "anmggzem"
  - Boolean: FALSE, TRUE
  - real: 1.23, 234.6, -0.34
- User-defined datatype
  - SATES
  - CAN_PARAMETERS
  - NETWORK_PACKET
  - …

Multi-Value Logic

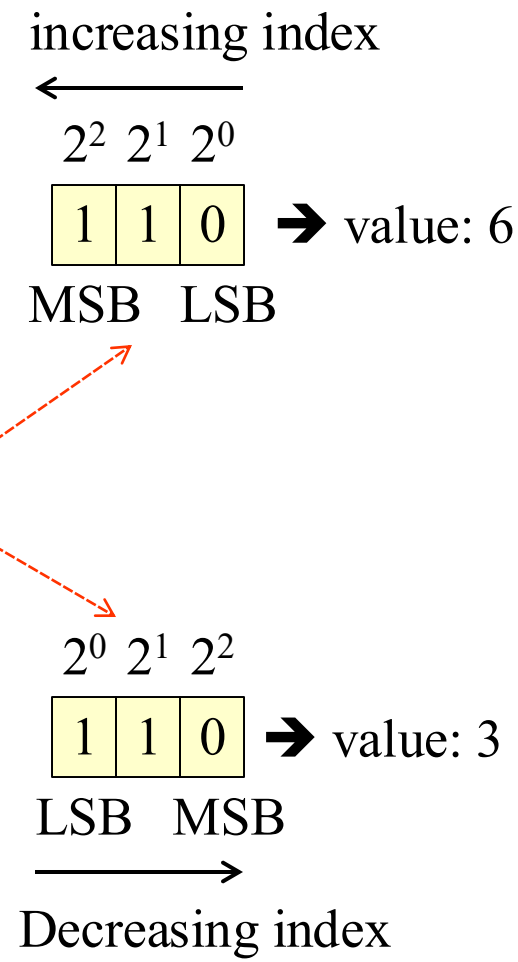
- Standard logic IEEE 1164
  - Std_logic instead of bit
  - Declaration in Package „standard_logic_1164“
  - Possible values:
    - ´U´ not initialized
    - ´X´ unknown
    - ´0´ Logic 0
    - ´1´ Logic 1
    - ´Z´ high impedance
    - ´W´ unknown (weak signal)
    - ´L´ logic 0 (weak signal)
    - ´H´ Logic 1 (weak signal)
    - ´-´ Don't care
  - Standard_logic_vector: Vektor von standard_logic values

Bit vectors

Single signals can be grouped in a vector:

std_logic → std_logic_vector( )

Example: 3-bit coded position




```
ENTITY Position IS
PORT(   A       : in   std_logic_vector( 2 downto 0 );
        B       : out std_logic_vector( 0 to 2 )  );
END Position;
```

3-bit wide, binary coded Bit-vector,
write: x **downto** y means decreasing index from the MSB to the LSB
(MSB: most significant bit, LSB: least ...)

Bit (Std_Logic) Vectors

- Declaration
  - signal Z_Bus: bit_vector(3 downto 0);
  - signal Z_Bus: std_logic_vector(0 to 3);

- Assignment:
  - “By position”
    - Z_Bus(3) <= C_Bus(1);
    - Z_Bus(1) <= C_Bus(1);
    - Z_Bus(3 downto 2) <= “01“
  - “Concatenation”:
    - Z_Bus <= A & B & C & D;
  - “Aggregates”:
    - Z_Bus <= (A, B, C, D);
  - “Specifying by element index“:
    - Z_Bus <= (3 => ‘ 1‘, 1 downto 0 => ‘ 1‘, 2 => ‘ 0‘);
  - “others“:
    - Z_Bus <= (3 => ‘ 1‘, 1 => ‘ 0‘, other => ‘ 0‘);

VHDL Operators

- Logic operators
  - AND, OR, NAND, NOR, XOR (equal precedence)
  - NOT (higher precedence)
- comparison
  - < (smaller than), <= (smaller or equal than), = (equal), >= (bigger or equal) , > (bigger than), /= (different)
- Arithmetic operators
  - + (Addition), - (subtraction), * (Multiplication), / (Division), ** (Exponent), abs (Magnitude), mod (Modulo), rem (Remainder)

Want To Learn More ?

Consider Undergraduate Research With Us/Other Labs

https://smartsystems.ece.ufl.edu/