VHDL Basics

Δαδαλιάρης Αντώνιος
dadaliaris@cs.uth.gr

VHDL: Ports

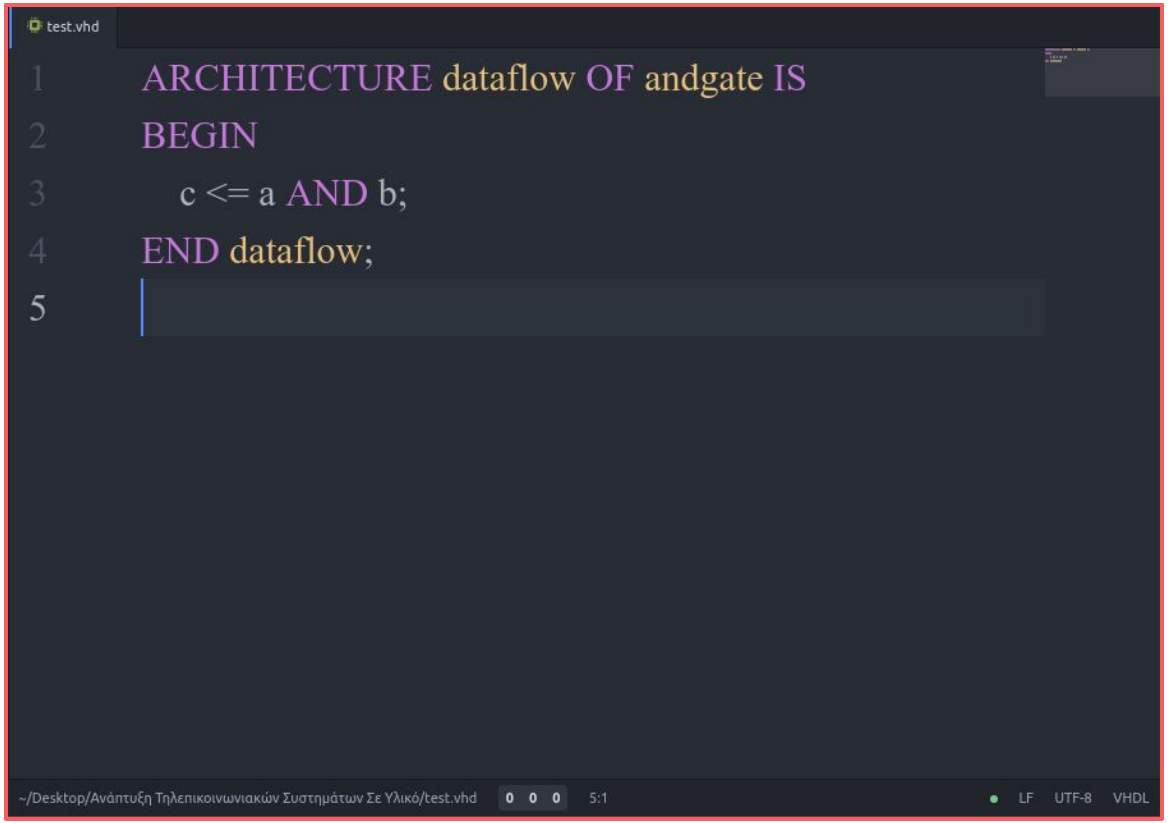
- Port names:
  - Letters, digits and/or underscores
  - All port names must start with a letter
  - Case insensitive
- Port types:
  - IN
  - OUT
  - INOUT
  - BUFFER

VHDL: Architectures

- The architecture specifies the design's internal implementation.
- We can have multiple architectures for one entity.
- We usually design multiple architectures with each architecture optimized for different performance metrics:
  - Area
  - Power
  - Timing
  - ....

VHDL: Architectures (cont.)

VHDL: Architectures (cont.)



VHDL: Signals

- Signals can represent either wires or storage elements.
- Signals can only be defined inside an architecture.
- Every signal declared must be associated with a data type.
- Every signal has attributes that can be useful at certain occasions.
- Since VHDL is a strongly-typed language only explicit type conversion is supported.

VHDL: Signals - Multi-Valued Logic

- Multi-valued logic adds values to the classical boolean logic in order to model the state of signals more accurately.
- Values:
  - ‘U’: uninitialized
  - ‘-’: don’t care
  - ‘1’: forcing 1
  - ‘0’: forcing 0
  - ‘X’: forcing unknown
  - ‘H’: weak 1
  - ‘L’: weak 0
  - ‘W’: weak unknown
  - ‘Z’: high impedance

VHDL: Multi-Valued Logic

- There are two standardized data types that use multi-valued logic:
  - std_ulogic
    - std_ulogic_vector
  - std_logic
    - std_logic_vector

VHDL: Built-in Data Types

- The basic built-in data types supported in VHDL are the following:
  - BIT and BIT_VECTOR
  - INTEGER
  - REAL
- Although these types can be used for simulating a design, they are not particularly useful in synthesis.
- You should always use the multi-valued logic data types.

VHDL: Operators

Operator	Description	Operand #1	Operand #2	Result
a ** b	Exponentiation	Integer	Integer	Integer
abs a	Absolute Value	Integer		Integer
not a	Negation	Boolean/bit/bit_vector		Boolean/bit/bit_vector
a * b	Multiplication	Integer	Integer	Integer
a / b	Division	Integer	Integer	Integer

VHDL: Operators (cont.)

Operator	Description	Operand #1	Operand #2	Result
a mod b	Modulo	Integer	Integer	Integer
a rem b	Remainder	Integer	Integer	Integer
+ a	Identity	Integer		Integer
- a	Negation	Integer		Integer
a + b	Addition	Integer	Integer	Integer

VHDL: Operators (cont.)

Operator	Description	Operand #1	Operand #2	Result
a - b	Subtraction	Integer	Integer	Integer
a & b	Concatenation	1D array	1D array	1D array
a sll b	Shift Left Logical	bit_vector	Integer	bit_vector
a srl b	Shift Right Logical	bit_vector	Integer	bit_vector
a sla b	Shift Left Arithmetic	bit_vector	Integer	bit_vector

VHDL: Operators (cont.)

Operator	Description	Operand #1	Operand #2	Result
a sra b	Shift Right Arithmetic	bit_vector	Integer	bit_vector
a rol b	Rotate Left	bit_vector	Integer	bit_vector
a ror b	Rotate Right	bit_vector	Integer	bit_vector
a = b	Equal To	any	any	Boolean
a /= b	Not Equal To	any	any	Boolean

VHDL: Operators (cont.)

Operator	Description	Operand #1	Operand #2	Result
a < b	Less Than	any	any	Boolean
a <= b	Less Than Or Equal To	any	any	Boolean
a > b	Greater Than	any	any	Boolean
a >= b	Greater Than Or Equal To	any	any	Boolean
a and b	Logic and	Boolean/bit/bit_vector	Boolean/bit/bit_vector	Boolean/bit/bit_vector

VHDL: Operators (cont.)

Operator	Description	Operand #1	Operand #2	Result
a or b	Logic or	Boolean/bit/bit_vector	Boolean/bit/bit_vector	Boolean/bit/bit_vector
a xor b	Logic xor	Boolean/bit/bit_vector	Boolean/bit/bit_vector	Boolean/bit/bit_vector
a nand b	Logic nand	Boolean/bit/bit_vector	Boolean/bit/bit_vector	Boolean/bit/bit_vector
a nor b	Logic nor	Boolean/bit/bit_vector	Boolean/bit/bit_vector	Boolean/bit/bit_vector
a xnor b	Logic xnor	Boolean/bit/bit_vector	Boolean/bit/bit_vector	Boolean/bit/bit_vector

VHDL: System Views

- There are three different ways we can view a digital system:
  - Behavioral view
  - Structural view
  - Physical view

VHDL: System Views (cont.)

- There are three different ways we can view a digital system:
  - Behavioral view
    - Describing functionality and the way inputs and outputs behave
    - The design is practically a black box (we know the functionality but we do not have a clue about its architecture)

VHDL: System Views (cont.)

- There are three different ways we can view a digital system:
  - Structural view
    - Describing the internal implementation of the design
    - Circuits, subcircuits, gates and the way they interconnect
    - Practically a block diagram

VHDL: System Views (cont.)

- There are three different ways we can view a digital system:
  - Physical view
    - Enriched structural view
    - Additional information
      - Subcircuit/Gate sizes
      - Interconnect routing
      - ….

VHDL: Objects

- Every element that can store a value is called an object.
- The main VHDL objects are the following:
  - SIGNAL
  - VARIABLE
  - CONSTANT
  - FILE

VHDL: Objects (cont.)

- SIGNAL:
  - Declaration: SIGNAL signal_name: DATA_TYPE;
  - Assignment: signal_name <= value;
  - We can “assume” that signals act like wires
  - The inputs and outputs declared in the ENTITY of a design are considered signals

VHDL: Objects (cont.)

- VARIABLE:
  - Declaration: VARIABLE variable_name: DATA_TYPE;
  - Assignment: variable_name := value;
  - Variables are not mapped to actual hardware.
  - We should only declare, assign values and use variables inside processes (see sequential statements).

VHDL: Objects (cont.)

- CONSTANT:
  - Declaration and assignment at the same time.
  - CONSTANT const_name: DATA_TYPE := value;
  - You should only use constants only to enhance the readability of your code.