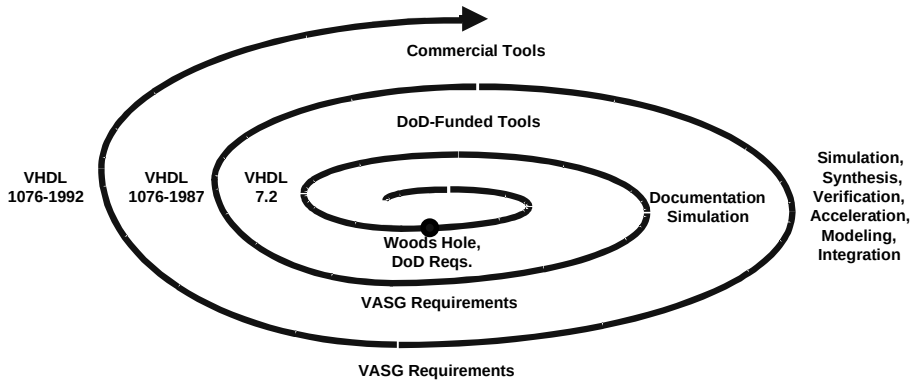
VHDL

- Standard for all stages of hardware design.
- Different levels of abstraction.
- Mixture of description at different levels.
- Can be used to specify both data and control path.
- Describe the specification of:
  - inputs
  - synthesis results
- For logic and high-level synthesis tools.

Cycles of VHDL Evolution



Language Features

- Scope/Range of the hardware described
- Design Management
- Timing Description
- Architectural Description
- Designer Interface Description
- Designer Environment Description
- Language Extensibility

Language Features

- Scope/Range of the hardware to be described:
  - From algorithmic description to logic gate description
  - Combinational & Sequential
  - Sequential circuits
    - Clocked - synchronous
    - Unclocked - asynchronous

Language Features

- Scope/Range of the hardware to be described:
  - Specification of
    - Explicit clock - synchronous
    - Events - asynchronous
  - Creation of simulation models corresponding to the design level being modeled
  - Mixed-level & mixed-mode

Language Features

- Design Management
  - Design methodology
  - Hierarchical design for complex structures
  - **Top-down** and/or **Bottom-up** approach
    - Separation between
      - *Data flow*
      - *Control flow*

Language Features

- Design Management
  - Language should support:
    - Modular design
      - aggregation of modules
    - Generic design description
      - descriptions which can be parametric with respect to some aspects
    - Abstract data types
      - abstraction of the electrical signal behavior

Language Features

- Timing Description
  - Timing concepts oriented toward hardware simulation at any level of design
    - hierarchy
      - different timing aspects at the different hierarchical levels
    - abstraction
      - different timing aspects are considered

Language Features

- Timing Description
- Language should support:
  - Event-driven & Cycle-driven
  - Timing granularity
  - Propagation delays specification
- Information on:
  - rise time
  - fall time
  - signal periodicity
  - delay

Language Features

- Architectural Description
  - **Functional description**
    - functional relationship of a design future output signal values to its present and past input signal values
      - *Behavioral* Description
      - *Data-Flow* Description
  - **Structural description**
    - interconnection of design elements

Language Features

- Designer Interface Description
  - I/O ports
    - handle any kind of data
  - All levels of abstraction & hierarchy
    - handle any kind of description
  - CAD tools
    - fit in any kind of existing design flow

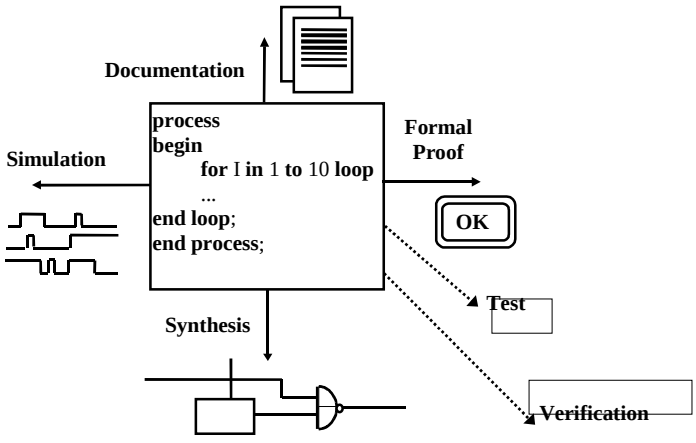
Language Features

- Designer Environment Description
  - Description of the **environment** the hardware will operate in
    - package data
    - design process
    - logic design rules
    - circuit-and-technology data
    - physical-design data
    - simulation data

Language Features

- Language extensibility
  - Independent of:
    - specific technology
    - preferred design methodology
    - design style
  - Technology advance
  - Flexibility
  - Definition of additional primitive data types

Application Domains



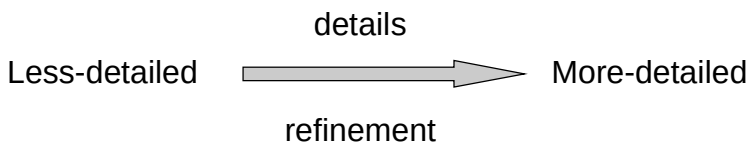
Application Domains

- MODELLING Design Definition (Documentation)
- SIMULATION Design Behavior
- SYNTHESIS Design Implementation

- FORMAL PROOF
- VERIFICATION
- TESTING

Application Domains

- MODELLING
  - The specification of a complex design can be achieved by adopting a Top-Down design methodology
  - Such a strategy is realized through the Incremental Design method
  - A flow of operations is repeatedly applied to the design until an appropriate level of abstraction is reached:

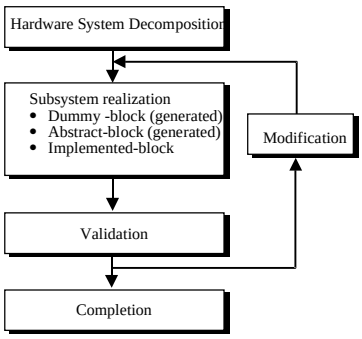


Application Domains

- SIMULATION & SYNTHESIS
  - VHDL allows the user to apply a methodology described as a **loop**, starting with a behavioral design description for simulation
  - Once simulated and accepted, this description is further partitioned
    - more and more details are added
    - the results are re-simulated.
  - The level of detail will reach a point where the synthesis system can take over and **automatically** produce a hardware implementation

Application Domains

- System Design with Incremental Design
  - This method defines basic design activities, valid for most of the applications, application-specific design constraints, and design options.



Incremental Design

- Design Activities
  - Hardware System Decomposition
  - Subsystem realization
    - Dummy-block
    - Abstract-block
    - Implemented-block
  - Validation
  - Modification

Incremental Design

- Design Constraints
  - Abstraction Levels Constraints

    determine the top level and the further addressed levels of abstraction of the design process. The following levels could occur as top-level

    - Architecture or System level

      the system is decomposed into components and the environment is described

      - characteristics (e.g. performance)
      - timing behavior in a global view (e.g. asynchronous communication between processes)

Incremental Design

- Design Constraints
  - Abstraction Levels Constraints
    - Algorithmic level

      the description at the partitioning of the system from the previous step. Each component of the system is implemented by an algorithm

      - timing behavior in a global view (e.g. asynchronous communication between processes)
    - Functional block level or RTL level

      the point of view in the description is the view of the design objects

      - timing behavior at a more detailed level (e.g. cycles of process execution)

Incremental Design

- Design Constraints
  - Constraints from the Design Process

    after HDL model generation, the model has to be validated, using simulation and to be implemented at a lower level of abstraction, using synthesis tools

    - These tools require a specific description style, especially synthesis tools, to be efficient or even correct
  - Hardware System Constraints
    - use of specific elements ➠ components libraries
    - timing behavior ➠ design complexity

Incremental Design

- Design Options
  - ***Application-oriented*** Design

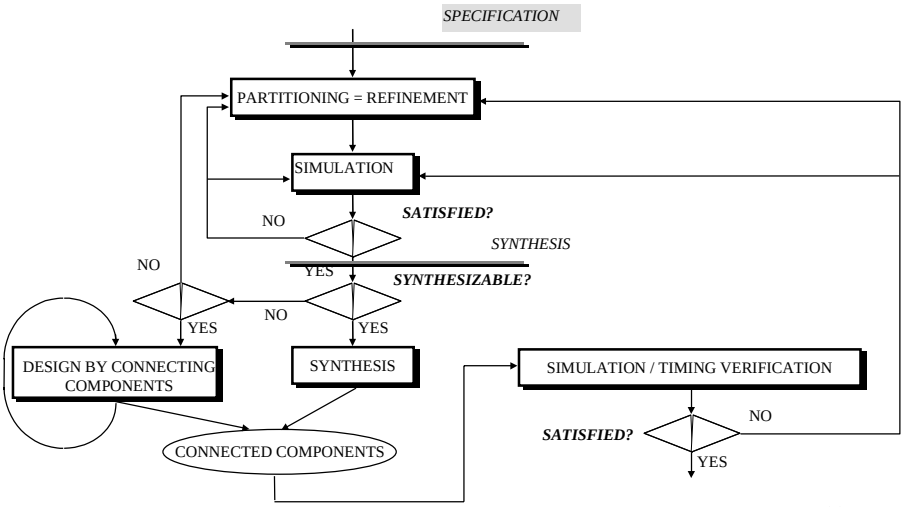
    the design object should be in the centre of interest and not the design environment
    - for designing a CPU, the design system advises and supports first the building of the entity and then the use of processes to describe a FSM
  - ***Language-oriented*** Design

    design environments cannot handle all kinds of designs in an efficient way
    - the language must be transparent to the designer

Incremental Design

- Design by ***simulation and synthesis*** methodology
  - all synthesis steps are verified by simulation
  - unaccepted refinements imply design loops
  - when the design satisfies the requirements the design process ends

Design Methodology



General Design Flow

- Starting point: design specification expressed with (not always) **specific formalisms**
- Simulation of the design specification to validate correct design behavior
- Design partitioning
- Modules definition
- Modules simulation
- Synthesis
- Implementation simulation

General Design Flow

- **STEPS**
  - Production Specification
  - Algorithm Development
  - Design Decomposition
  - Module Implementation
  - Gate-Level Verification
  - Layout
  - Post Layout Verification
- **TASKS**
  - Writing of specification documents
  - Behavioral modeling and simulation
  - Block Diagrams, RTL Level Modeling
  - Schematic Capture, Synthesis & Optimization
  - Simulation and Timing Analysis
  - Netlisting to an ASIC vendor
  - Gate level simulation and timing analysis

General Design Flow

- Each step may be iterated several times until the desired results are obtained
- The process is then applied to another module of the design until all units have been completed
- The iteration allows to correct design errors earlier in the design process
  - reduction of time and costs
  - with respect of error detection at silicon or component implementation level

General Design Flow

- With this methodology based on the two VHDL applications, ***Simulation*** & ***Synthesis***, the final implementation can be
  - considered correct by construction
  - functionally verified by simulation
  - validated by formal verification

Assumptions

- The presented methodology is based on some fundamental **assumptions** that allow the designer to know the quality of the final result
  - The synthesis system produces hardware that behaves correctly and exactly like purely behavioral models previously simulated
    - First create and simulate then get them automatically designed
  - Problems are related to timing and depend on the lack of clear synthesis semantics

Assumptions

- Even when describing behavior for simulation, the designer works with synthesis in mind (at least at later stages of the design process)
  - The designer may give a description that suits a synthesis tool rather than another, knowing the *limitation of a particular automatic system*
- Partitioning the design into "reasonable" chunks of behavior interacting with each other
  - What is reasonable?
    5k equivalent gates per individual module

Assumptions

- The technology information must be **available** and used by the synthesis system as early as possible in the synthesis process
- Technology information allows
  - efficient ***partitioning***
  - ***performance/costs*** effective designs

Application Domains - Prospective

- FORMAL PROOF
  - to verify that the description matches some higher-level specifications that are known to be correct
- VERIFICATION
  - promising application area

Validation is the check that the system we are building is a satisfactory solution to the real-word problem. Verification checks the internal consistency of two consecutive steps, by allowing one to prove that an implementation is correct with respect to its specification

Application Domains - Prospective

- TESTING
  - test pattern generation is already being considered hardware test
- INTEGRATION
  - of analysis, diagnosis, synthesis and verification tools
- Hardware/Software modeling in VHDL?

VHDL and Modeling Aspects

- The elements of the TOP-DOWN design methodology can be identified as
  - ABSTRACTION
  - HIERARCHY
  - MODULARITY
  - REUSABILITY
  - EFFICIENCY

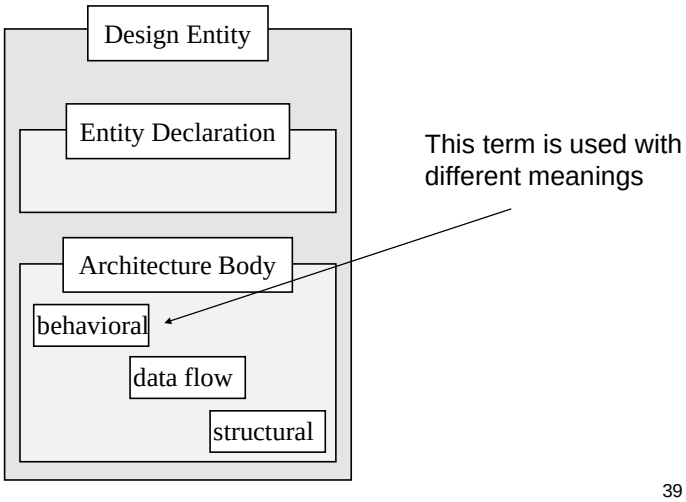
VHDL Modeling Aspects

- ABSTRACTION
  - Today Top Down Design refers to the practice of using a Hardware Description Language to ***capture*** the functionality of a design at an abstract level.
  - An abstraction will group details (in a module) that describe the function of design unit but does not describe how the design unit is implemented.

VHDL Modeling Aspects

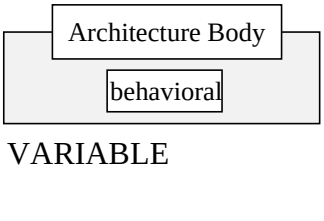
- ABSTRACTION
  - The primary abstraction level of a VHDL hardware model is the design entity, which can represent a cell, a chip, a board, or subsystems.
    - **Entity**
    - **Architecture**
  - Design description methods:
    - Behavioral ➠ high-abstraction level
    - Data flow and Structural ➠ lower abstraction levels.

VHDL Modeling Aspects



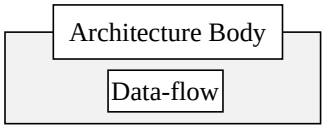
VHDL Modeling Aspects

- ABSTRACTION
  - Design Description Methods: behavioral
    - Functional behavior of an hardware design in terms of signals and responses to various stimuli.
    - Algorithmic description with no underlying structure.
  - Constructs:
    - PROCESS
    - PROCEDURE
    - FUNCTION



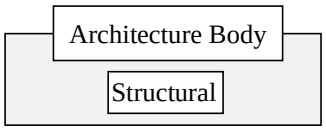
VHDL Modeling Aspects

- ABSTRACTION
  - Design description methods: data flow
    - Definition of the design functionality through the information flow from inputs to outputs by means of arithmetic and logical relations.
    - Concurrent execution of the defined relations.
  - Constructs:
    - BLOCK
    - PROCEDURE
    - FUNCTION



VHDL Modeling Aspects

- ABSTRACTION
  - Design description methods: structural
    - Topological description of the interconnection of components for realizing the circuit functionality completely detached from the components' functionality.
  - Constructs:
    - COMPONENT declaration
    - COMPONENT instantiation
    - CONFIGURATION



VHDL Modeling Aspects

- ABSTRACTION (behavioral example)

```
ENTITY ex_adder IS
    PORT( op1, op2, carryin: IN bit;
        output, carryout: OUT bit);
END ex_adder;
ARCHITECTURE beh_arc OF ex_adder IS
BEGIN
    PROCESS (op1, op2, carryin)
        variable result: integer range 0 to 3 := 0;
    BEGIN
        result := bit_to_integer(op1) + bit_to_integer(op2)
                + bit_to_integer(carryin);
        output <= boolean_to_bit((result mod 2)=1);
        carryout <= boolean_to_bit(result > 1);
    END PROCESS ;
END beh_arc ;
```

VHDL Modeling Aspects

- ABSTRACTION (data-flow example)

```
ARCHITECTURE str_dataflow OF ex_adder IS
      SIGNAL tmp1, tmp2, tmp3: bit ;
BEGIN
      tmp1 <= op1 XOR op2;
      output <= tmp1 XOR carryin after 1ns ;
      tmp2 <= op1 AND op2;
      tmp3 <= tmp1 AND carryin after 1ns ;
      carryout <= tmp2 OR tmp3 after 2ns ;
END str_arc ;
```


VHDL Modeling Aspects

- ABSTRACTION (structural example)

```
ARCHITECTURE str_arc OF ex_adder IS
      COMPONENT half_add IS
            PORT( in1, in2:   IN bit;
                  out, carry: OUT bit) ;
      END COMPONENT ;
      COMPONENT or_gate IS
            PORT( a, b: IN bit;
                  o:    OUT bit) ;
      END COMPONENT ;
      SIGNAL tmp1, tmp2, tmp3: bit ;
BEGIN
      I1: half_add PORT MAP (op1,op2,tmp1,tmp2) ;
      I2: half_add PORT MAP (tmp2,carryin,tmp3,output) ;
      I3: or_gate PORT MAP (tmp1,tmp3,carryout) ;
END str_arc ;
```

VHDL Modeling Aspects

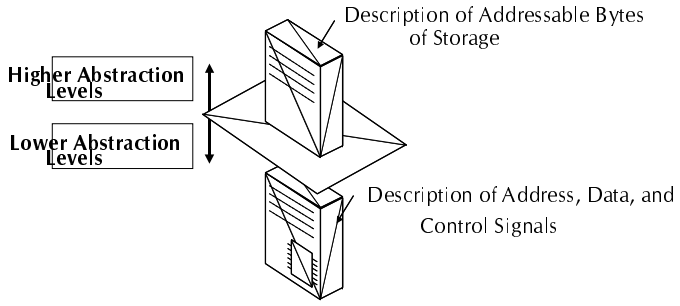
- ABSTRACTION
  - Example:
    - A **Read Only Memory** device is described at a high level as a series of address locations with corresponding data bytes stored in each location
      - no care about address lines, data lines, or control lines
      - focus on the data byte assignments to selected addresses

VHDL Modeling Aspects

- ABSTRACTION
  - In the lower-level module, it is possible to describe how each signal on the ROM pins must be configured to read or program each data storage location
    - To change the data stored in a given location go to the higher-level module and modify the hex value associated with an address rather than redefine the states of many data lines

VHDL Modeling Aspects

- ABSTRACTION (ROM example)



VHDL Modeling Aspects

- ABSTRACTION
  - ***Information Hiding***
    - Complements abstraction
    - Focuses the attention of the designer on relevant information
      - the extracting of the functional details from a given module and masking of the implementation details
    - Protects proprietary information

VHDL Modeling Aspects

- HIERARCHY
  - Approach of **splitting** an initial, complex problem into simpler, more manageable sub-problems that can be worked out separately to achieve a solution to the initial problem.

VHDL Modeling Aspects

- HIERARCHY
  - Design Partitioning
    - Related to the problem of defining a hierarchy to handle the design process within the methodology
  - ***Starting point:*** behavioral description of the entire device
  - ***Target:*** description identifying elements, which properly connected perform the same global functionality
    - Concurrent decomposition
    - Sequential decomposition

VHDL Modeling Aspects

- HIERARCHY - PARTITIONING
  - How to do the partitioning ?
    - There is no defined methodology
      - ***Designer experience***
  - *Hints*:
    - Identification of operations to be performed
      - control path & data path
    - Behavioral Specification
      - Simulation
        - Synthesis to get an idea of used elements

VHDL Modeling Aspects

- HIERARCHY - PARTITIONING
  - Subprograms - functions and procedures
    - A subprogram allows the user to decompose the hardware description into behavioral descriptions or operations using algorithms for computing values.
    - In a TOP-DOWN approach it is possible to decompose the solution into its primary functions:
      - **high-level**: "what happens"
      - **low-level**: "how it happens"

VHDL Modeling Aspects

- HIERARCHY - PARTITIONING
  - ***Simulation***
    - synchronization among elements
    - data exchange among elements
  - ***Synthesis***
    - the process may autonomously create a partitioning and a hierarchy while processing a VHDL description

VHDL Modeling Aspects

- REUSABILITY
  - The complexity of designed electronic devices underlines the need to adopt effective methods to cut the design time
    - definition of **libraries** of standard cells

VHDL Modeling Aspects

- EFFICIENCY
  - The complexity of developed models results in an increased **CPU time** consumed to simulate them
  - This raises the issue of efficiency of VHDL concepts used for modeling:
    - use variable instead of signals whenever possible
    - use signal attribute returning a value rather than a signal
    - efficient process statement

VHDL Modeling Aspects

- DOCUMENTATION
  - VHDL may be used to document a design. Communication medium
    - ***man-to-tool***
    - ***man-to-man***
    - ***tool-to-tool***
  - VHDL is used as a *format*

VHDL Modeling Aspects

- DOCUMENTATION
  - There is a need for a common representation format that can be used to represent:
    - system specification
    - partial design data generated by various synthesis tools
    - synthesis results produced by those co-operating tools
  - Format serves as a common foundation constituting a synthesis environment

An input design specification written in VHDL may be mapped on this foundation, and similarly, the synthesis results may be mapped onto VHDL from this foundation

VHDL Description Styles

- A comparison (1)
  - The behavioral description more generally constitutes the starting point for the specification of systems of a relevant complexity
    - Adopted in the Top-Down design methodology
    - Apt for the immediacy of describing in an almost algorithmic way, complex devices, independent of the final implementation
    - High abstraction level
    - Used for validating the correctness of the VHDL description with respect to the specifications

VHDL Description Styles

- A comparison (2)
  - The data flow description requires the knowledge of the input/output relations, in terms of arithmetical and logical data manipulation
    - The specification provides hints of what the final realization will look like
    - Apt for describing Data-path
  - The structural description defines the topology of the circuit in terms of the components constituting it and of their interconnections
    - Component functionality is not required at this level of description

VHDL Description Styles

- A comparison (3)
  - The adopted description style depends on several factors, among which:
    - specific knowledge of the device functionality and possible (desired) implementation
    - target of the VHDL description
      - evaluation of alternatives
      - determination of a sound implementation
      - …