This course explains how to design with VHDL on Xilinx FPGAs using ISE Design Suite

Objectifs

- Comprehend the various possibilities offered by VHDL language
- Implementing state machine
- Reusing components
- Understand the logical synthesis notions
- Knowing the different writing style and their impact on the quality of synthesis results
- Learning how to write test bench for simulation
- Knowing the performance that can be expected from Xilinx FPGA
- Identify critical paths and verify compliance with the timings
- Learning how to configure compilation options and implementation constraints
- Manipulating the debug tools and implementation reports

Prerequisites

- Knowledge of digital technology
- Concepts of Boolean algebra
- Some programming concepts are desirable (whatever language)
- This training is intended to electronic engineers who are willing to acquire a strong designing methodology, and to take the best of VHDL language and the associated synthesis and simulation tools for designing Xilinx FPGA

Course environment

- One PC for two trainees
- Xilinx ISE Design Suite 13.3 Logic Edition

Related courses

- MicroBlaze implementation, reference N2 - IEEE1588 - Precise Time Protocol course
- Spartan-6 / Virtex-6 Integrated PCI Express Block, reference N2 - IEEE1588 - Precise Time Protocol course
- Designing with Ethernet MAC logicores, reference N2 - IEEE1588 - Precise Time Protocol course
## Plan

### Day 1

### From the logic gate to the FPGA

- **Structure of an Integrated Circuit**
  - SSI (small scale integration), TTL
  - MSI (medium scale integration), PALs, GALs, PLDs
  - LSI (large scale integration), CPLDs
  - VLSI (very large scale integration), ASICs, ASSPs, FPGAs

- **Development of logical architectures**

- **Technology constraints**
  - Interconnection methods (SRAM, Fuse, AntiFuse, Flash)
  - Clock distribution
  - Logic element types

- **Timing issues**

### Spartan6 and Virtex6 FPGA architecture

- **General structure**

- **CLB and slices notion**
  - Combinatory logic and registers
  - Arithmetical logic
  - Distributed memory
  - Shift register SRL

- **In/Out blocks**
  - In/Out registers
  - DDR registers
  - Timing and electric settings and specificities

- **Dedicated RAM blocks and use modes**
  - Customable FIFOs implementation
  - Other example of use

- **Clocks distribution, DCMs & PLLs**
  - Global Buffer, local buffer
  - DCMs, PLLs and settings

- **Dedicated multipliers and DSP48 blocks**

- **Configuration**
  - Master, slave, SPI, BPI, JTAG

### VHDL Contributions

- **Interest of VHDL programming**

- **Different steps of the design**
  - Programming
  - Simulation
  - Synthesis
  - Mapping
  - Place and Route
  - Timing Analysis
  - Bitstream generation
Day 2

**Basic concepts of VHDL**
- Notion of entity / architecture
- IEEE library use
- Predefined types and objects
  - Ports, signals, variables
- Different styles of architecture
- Component instantiation
- Practical lab

**Combinational logic in VHDL**
- Tools for modeling components
- Concurrent and sequential instructions
  - Allocation
  - Process(Selection list, Sequential instructions, Variables)
- Predefined operators and use extended by using standardized packages
- Concurrent instructions : when, with select, for generate
- Practical lab

Day 3

**Sequential logic in VHDL**
- Flip-flop reminder
- Reset management
- Tri-state buffers
- Synchronous process
- Practical lab

**Hardware designing methodology in logical synthesis**
- Asynchronous conception and classic tricks
  - Metastability and hazards of functioning
  - Limits of functional simulation and timing on asynchronous designs: how to get over them?
- Asynchronous event management
  - Random
  - Data streams
- Synchronous design –advantages-methodology-focusing
- Static timing analysis: how to use it?
- Optimization of performance irrespective of the target
- Pipeline notion
- Practical lab

**Writing rules of VHDL code in logical synthesis**
- A few tricks to avoid
- Potential interpretation incoherencies between the logical synthesis and the simulation : how to avoid it
Day 4

Hierarchy management for a better use

- Organization of design by functional modules: what routing to choose
- Inference and instancing notions
  - When is it important to instantiate primitives or macros?
- Precautions for an evolutionary and/or re-usable code
- Importance of modules’ name selection and of the nets to facilitate the physical implementation, the simulation and the tuning
- Does the hierarchy have to be preserved during the logical synthesis?
- Practical lab

Advanced VHDL language for optimization and code re-use in logical synthesis

- Notion of variable and example of use
- Genericity and automatic configuration of re-usable modules
- Useful predefined attributes in logical synthesis
- Functions and procedures
- Definition of packages and libraries
- Practical lab

Implementation and tuning tools

- Implementation stream and bitstream generation
  - Translate
  - Map
  - Place and Route (PAR)
  - BitGen
- Analysis of MRP and PAR reports
- Main implementation options
  - MAP
  - PAR
  - BITGEN
- Implementation results analysis tools - constraints
  - PlanAhead
  - FPGA EDITOR
  - TIMING ANALYZER
  - Introduction to CHIPSCOPE
  - Constraints file

Day 5

The state machines

- Mealy and Moore machines
  - Graphic representations
  - Implementation
  - VHDL translation
- Design principles of an FSM with two processes
- Reset of a state machine
- Simulation usage to verify the design
- Resource use optimization
- Practical lab
**Test benches and simulation**

- A few basic rules for the writing of an efficient test bench
- VHDL instructions specific to simulation
  - Wait and its various forms
  - "Loop"
  - Assertions
  - Data types
  - Timing verification
  - Others
- Writing components models intended to make the simulation more realistic
- Use of existing models and simulation packages
- Practical lab
- Integration of "pseudo logic" in order to facilitate the interpretation of the simulation results
- Writing and reading of ASCII files
  - Allocation of a data flow from a file - Test vector generation
  - Storage of the simulation results in a file
- Command interpreter
- Generating information messages
- Practical lab

---

**Renseignements pratiques**

- **Duration**: 5 jours
- **Cost**: 2100 € HT