

## VHDL Logical Synthesis and Simulation for Xilinx FPGA design

Ref : 002572A

Duration : 5 days

### OBJECTIVES

- Comprehend the various possibilities offered by VHDL language
- Understand the logical synthesis notions
- Knowing the different writing style and their impact on the quality of synthesis results
- Knowing the performance that can be expected from Xilinx FPGA
- Learning how to configure compilation options and implementation constraints
- Manipulating the debug tools and implementation reports

### RELATED COURSES

- Designing for performance, ISE (002833A)
- DSP implementation techniques for Xilinx FPGAs (002838A)
- Designing a LogiCore PCI Express system (004552A) or with Multi-Gigabit RocketIO Transceivers (002843A)
- Microblaze (003149A) or PowerPC (002952A) system implementation

### PARTNERS

- This training course is approved by XILINX

### PREREQUISITES

- This training is intended to electronic engineers who already have a good knowledge in designing digital electronic circuits, 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, more particularly of Spartan 3 family and its derived.

### TRAINING MATERIALS

#### Software Configuration :

- Xilinx ISE Design Suite 11.3 Logic Edition

#### Recommended Hardware Configuration :

- Intel Core 2 or equivalent
- Windows XP
- 1 GB Free disk after software installation
- At least 1Go RAM
- Minimum Display resolution : 1024 x 768
- On Site training : video projector

Authorized  
Training Provider

### Contact

Tel : 05 62 13 52 32  
Fax : 05 61 06 72 60  
training@mvd-fpga.com

Course also available  
customized

Next sessions, see : <http://www.mvd-training.com/en/schedule.html>

### TOPICS

#### Architecture of Spartan 3 FPGA and its derived

- General structure
- CLB and slices notion
  - Combinatory logic and latches
  - Arithmetical logic
  - Distributed memory notion
  - Offset register SRL 16
- In/Out blocks
  - In/Out latches
  - DDR register
  - Spartan 3E, Spartan 3A timing and electric settings and specificities
- Dedicated RAM blocks and use modes
  - Customable FIFOs implementation
  - Other example of use
- Clocks distribution and DCMs
  - Global Buffer, local buffer (Spartan 3E/A)
  - DCMs and settings
- Dedicated multipliers and DSP48A blocks (Spartan\_3A-DSP)
- Configuration
  - Master, slave, SPI (Spartan3E/A)

#### Fundamental differences with Virtex 4 architecture

#### Introduction to Virtex 5 architecture

#### 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
    - Ios latches packing
    - CLB Pack factor
    - MAP-timing
  - PAR
    - Global effort
    - Extra effort
    - Multi Pass Place and Route
    - Other options
  - BITGEN
    - Configuration options
    - Startup options
- Implementation results analysis tools - constraints
  - FLOORPLANNER
    - How to analyze the placement quality
    - Placement constraints generation
    - By element
    - By functional block
  - FPGA EDITOR
    - Detail implementation checking
    - Arithmetical resources identification
    - Navigation
    - Checking important aspects of routing (clock lines, BUFGs access, DCMs, ...)
  - TIMING ANALYZER
    - Global analysis of the performance of a design
    - Crossprobing between Timing Analyzer and FPGA editor or Floorplanner
    - Conclusions for the design optimization
  - Introduction to CHIPSCOPE
  - Constraints file
    - Placement constraints and electric configuration of Ios
    - Basic timing constraints
      - PERIOD
      - OFFSET IN/OUT

- Advanced timing constraints
  - Multi-cycle path
  - False path
  - Constraints limits on path linking two different clock domains
- Advanced placement constraints
  - AreaGroup : use precautions
- Genericity and automatic re-configuration of re-usable modules
- Useful predefined allocations in logical synthesis
- Functions and procedures
- Definition of packages and libraries
- Practical labs

### Writing rules of VHDL code in logical synthesis

- Notion of entity / architecture
- Competing and sequential instructions
- Predefined types and objects
- Predefined operators and of use extended by using standardized packages
- Process
  - Importance of the sensitivity list
  - Sequential instructions : if, case, loop
  - Use of variables
- A few tricks to avoid
- Potential interpretation incoherencies between the logical synthesis and the simulation : how to avoid it
- Practical labs

### 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 ?
- Synchronous design –advantages-methodology-focusing
- Static analysis of the timing : how to use it ?
- Optimization of performance irrespective of the target
- Pipeline notion
- Asynchronous event management
  - Random
  - Data streams
- Practical labs

### Deepening of VHDL language for the optimization and the re-use of the code in logical synthesis

- Notion of variable and example of use

### DOCUMENTATION

Training manuals will be given to attendees during training in print.

### 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 labs

### Test benches and simulation

- A few basic rules for the writing of an efficient testbench
- VHDL instructions specific to simulation
  - Wait and its various forms
    - « Loop »
  - Assertions
  - Data types
  - Others
- Writing components models intended to make the simulation more realistic
- Use of existing models and simulation packages
- Practical labs
- 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
  - Storage of the simulation results in a file
- Command interpreter
- Generating information messages
- Practical labs