



Comprehensive VHDL

5 jours



Présentation

“Comprehensive VHDL” est un cours de 5 jours pour apprendre à utiliser le langage VHDL pour la conception des FPGA et ASIC. Ce cours a été mis à jour et restructuré pour refléter les plus récentes méthodes de conception. Les participants peuvent suivre les modules individuels ou le cours complet de 5 jours.

- **VHDL for FPGA Design (jours 1 à 3)** : forme le participant de façon pratique à la conception de FPGA en apportant les bases nécessaires en VHDL. L’orientation de ce cours est sur le flot VHDL vers FPGA. Ce module fournit également les éléments de base nécessaires pour les concepteurs d’ASIC et de FPGA souhaitant appliquer les possibilités plus avancées décrites dans le module suivant. Les participants traitant d’applications FPGA emporteront une infrastructure très souple de projet qui comprend un jeu de scripts, d’exemples de circuits, de modules et des fichiers de contrainte qui peuvent être utilisés, adaptés et étendus pour leur propre application.
- **VHDL Avancé (jours 4 et 5)** : s’appuie sur le module précédent pour préparer le participant à la conception de FPGA et ASIC complexes. Il traite en particulier de l’utilisation de VHDL pour les circuits hiérarchisés, pour la réutilisation des circuits, et la création de bancs de test plus puissants.

Doulos étant une société indépendante, les participants peuvent utiliser les outils de conception de leur choix durant les applications pratiques qui occupent 50 % du temps de la formation. Ces applications sont présentées sous forme d’exercices soigneusement préparés, afin de faciliter l’acquisition des connaissances.

Objectif pédagogique

- Maîtrise de la conception VHDL des composants programmables ou des ASICs.
- Formation pour la réalisation d’un premier projet VHDL
- Consolidation et approfondissement de la connaissance du langage VHDL.

Contenu de la formation

VHDL pour FPGA

- Syntaxe et sémantique du langage VHDL pour la conception des FPGA.
- Comment adopter un style d’écriture efficace et sûr pour les outils de synthèse actuels.
- Comment tirer parti, depuis le langage VHDL, des spécificités des composants FPGA.
- Comment écrire des bancs de tests VHDL simples.
- Le flot de conception VHDL avec les outils de simulation, de synthèse et de placement routage.
- Comment écrire du code VHDL de qualité reflétant les meilleurs standards de l’industrie.

VHDL Avancé

- Instructions du langage VHDL s’appliquant aux circuits complexes FPGA et ASIC.
- Instructions du langage VHDL et styles de codage pour écrire des bancs de test complexes.
- Comment coder des circuits hiérarchisés utilisant plusieurs bibliothèques de conception VHDL.
- Comment écrire du code VHDL paramétré et réutilisable en utilisant les génériques et les types de données.
- Comment utiliser les simulations au niveau porte.

Connaissances requises

Les participants doivent avoir suivi la formation **Essential Digital Design Techniques** (ou équivalent), ou avoir de bonnes connaissances en informatique et en électronique digitale. Aucune connaissance préalable du langage est demandée.

Les participants suivant seulement le cours **VHDL Avancé** doivent avoir une expérience de réalisation de circuit et avoir suivi le cours VHDL pour FPGA Design ou un équivalent.

Supports de cours

Les manuels de cours Doulos sont réputés pour être les plus détaillés et les plus faciles d'utilisation. Leur style, leur contenu et leur exhaustivité sont uniques dans le monde de la formation HDL. Ils sont souvent utilisés comme référence après avoir suivi les cours de formation. Sont compris dans la formation :

- Les notes de cours indexées constituant un manuel de référence complet.
 - Le cahier d'applications rempli d'exemples et d'applications pratiques pour vous aider à mettre en œuvre vos connaissances.
 - Le « Doulos Golden Reference Guide », aide-mémoire VHDL complet et pratique (syntaxe, sémantique et astuces »).
 - Les « Tool Tour Guides » (pour mettre en œuvre rapidement les outils de votre choix).
 - Le guide de conception pour les principales technologies ASIC et FPGA-PLD.
-

Structure et contenu

VHDL pour FPGA

Introduction

The scope and application of VHDL • design and tool flow • FPGAs • the VHDL world

Getting Started

The basic VHDL language constructs • VHDL source files and libraries • The compilation procedure • Synchronous design and timing constraints

FPGA Design Flow (Practical exercises using a hardware board)

Simulation • Synthesis • Place-and-Route • Device programming

Design Entities

Entities and Architectures • Std_logic • Signals and Ports • Concurrent assignments • Instantiation and Port Maps • The context Clause

Processes

The Process statement • Sensitivity list versus Wait • Signal assignments and delta delays • Register transfers • Default assignment • Simple Testbenches

Synthesising Combinational Logic

If statements • Conditional signal assignments and Equivalent process • Transparent latches • Case statements • Synthesis of combinational logic

Types

VHDL types • Standard packages • Integer subtypes • Std_logic and std_logic_vector • Slices and concatenation • Integer and vector values

Synthesis of Arithmetic

Arithmetic operator overloading • Arithmetic packages • Mixing integers and vectors • Resizing vectors • Resource sharing

Synthesising Sequential Logic

RISING_EDGE • Asynchronous set or reset • Synchronous inputs and clock enables • Synthesisable process templates • Implied registers

FSM Synthesis

Enumeration types • VHDL coding styles for FSMs • State encoding • Unreachable states and input hazards

Memories

Array types • Modelling memories • IP Generators • Instantiating generated components • Implementing ROMs

Basic TEXTIO

TEXTIO • READ and WRITE • Using TEXTIO for testbench stimulus and outputs • STD_LOGIC_TEXTIO

VHDL Avancé

More about types

Variables • Loops • Std_logic and resolution • Array and integer subtypes • Aggregates

Managing Hierarchical Designs

Hierarchical design flow • Library name mapping • Component declaration • Configuration • Hierarchical configurations • Compilation order

Parameterised Design Entities

Array and type attributes • Port Maps • Generics and Generic Maps • Generate statement • Generics and generate.

Procedural Testbenches

Subprograms • Procedures • Functions • Parameters and Parameter Association • Package declarations • Package bodies • Subprograms in packages • Subprogram overloading • Operator overloading • Qualified expressions • RTL Procedures.

Text-File-Based Testbenches

Assertions • Opening and closing files • Catching TEXTIO errors • Converting between VHDL types and strings • Checking simulation results • Initialising memories • Foreign bodies.

Gate Level Simulation

Rationale for gate level simulation • VITAL tool flow • Reuse of RTL testbench at gate level • Comparison of RTL and gate level results • Behavioural modelling