Fast-Track Verilog pour utilisateurs VHDL

Formation accélérée de 1 jour !

Présentation

“Fast-track Verilog pour utilisateurs de VHDL” est un cours de conversion intensif concentré en une seule journée qui enseigne l’utilisation du langage de programmation Verilog® pour mener à bien les projets de conception de composants programmables et ASICs. Ce cours est destiné aux personnes qui ont déjà suivi le cours Comprehensive VHDL ou ont déjà une bonne expérience de conception avec le langage VHDL.

En mettant en évidence les similarités et les différences entre les langages VHDL et Verilog et les flots de conception associés, l’apprentissage du langage Verilog est très rapide. La maîtrise du langage Verilog est un pré-requis avant d'aborder les formations SystemVerilog.

Malgré sa très courte durée, ce cours procure une excellente maîtrise du langage Verilog, il permet de comprendre, modifier, améliorer et vérifier tout code existant, et il permet d'utiliser ce langage pour toutes les tâches assurées par le VHDL.

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

A qui est destiné cette formation ?

- Aux ingénieurs ayant déjà une pratique du langage VHDL et qui souhaitent maîtriser le langage Verilog.
- Aux ingénieurs désireux d'évoluer ensuite vers SystemVerilog.

Objectifs pédagogiques

- Passage de la pratique du langage VHDL à la complète maîtrise du langage Verilog.

Contenu de la Formation

- Les différences et similarités entre VHDL et Verilog.
- Comment utiliser le langage Verilog pour la conception RTL et la synthèse logique.
- Comment écrire efficacement des bancs tests en Verilog pour vérifier les designs.
- Comment éviter de commettre des erreurs en générant du code Verilog pour la synthèse..

Connaissances requises

Les participants doivent avoir suivi la formation Doulos **Comprehensive VHDL** (ou avoir des connaissances équivalentes), et avoir déjà pratiqué.

Supports de cours

Les manuels de cours Doulos sont réputés pour être détaillés, précis et faciles d’utilisation. Leur style, leur contenu et leur exhaustivité sont uniques dans le monde de la formation. Ils sont souvent utilisés ensuite comme référence.

Sont compris dans la formation :

- Le Classeur du cours théorique, avec un Index. Il constitue un Manuel de Référence concis.
- Le Cahier d'exercices pratiques qui permet de mettre en œuvre les connaissances théoriques.
- Les fichiers des exercices et solutions
- Le « Doulos Golden Reference Guide », aide-mémoire Verilog complet et pratique (syntaxe, sémantique et astuces).

Programme de la Formation

Introduction

What is Verilog? ◆ Brief history and current status ◆ The PLI ◆ Scope of Verilog ◆ Design flow ◆ Verilog-2001 SystemVerilog ◆ Verilog books and Internet resources

Differences between VHDL and Verilog

“Philosophy” ◆ Red Tape ◆ Strong typing ◆ Determinism ◆ Data abstraction ◆ Structure vs behaviour - Nets vs registers ◆ Language structure – architecture, packages, configurations, files ◆ Identifiers ◆ Output ports ◆ Implicit wires ◆ Arrays ◆ Aggregates ◆ Signedness ◆ Operators ◆ Signal vs variable/nets ◆ Process vs initial/always ◆ if, case, loop differences ◆ File i/o ◆ Hierarchical names

Verilog Basics

Modules & ports ◆ Continuous assignments ◆ Comments ◆ Names ◆ Nets and strenghts ◆ design hierarchy ◆ Module instances ◆ Primitive instances ◆ Text fixtures ◆ $monitor ◆ Initial blocks ◆ Logic values ◆ Vectors ◆ Registers ◆ Numbers ◆ Output formatting ◆ Timescales ◆ Always blocks ◆ $stop and $finish ◆ Using nets and variables correctly

Combinational Logic

Event control ◆ If statements ◆ Begin-end ◆ Incomplete assignment and latches ◆ Unknown and don’t care ◆ Conditional operator ◆ Tristates ◆ Case, casez and casex statements ◆ full_case and parallel_case directives ◆ For, repeat, while and forever loops ◆ integers ◆ Self-disabling blocks ◆ Combinational logic synthesis

Sequential Logic

Synthesising flip-flop & latches ◆ Avoiding simulation race hazards ◆ Nonblocking assignments ◆ Asynchronous & synchronous resets ◆ Clock enables ◆ Synthesizable always templates ◆ Designing state machines ◆ State machine architectures ◆ Verilog code-based FSM strategy ◆ State encoding ◆ Unreachable states & safe design practices ◆ One-hot machines

Other features of Verilog

Verilog operators ◆ Part selects ◆ Concatenation & replication ◆ Shift registers ◆ Conditional compilation ◆ Parameterization and generate ◆ Hierarchical names ◆ Arithmetic operators and their synthesis ◆ Signed and unsigned values ◆ Memory arrays ◆ RAM modeling and synthesis ◆ $readmemb and $readmemh

Tasks and Functions

Understanding tasks ◆ Task arguments ◆ Task synchronization ◆ Tasks and synthesis

Test fixtures

File I/O – writing to files; File access using MCDs; Reading from files ◆ Automated design verification using Verilog ◆ Force and release ◆ Gate-level simulation ◆ Back annotation using SDF ◆ “Traditional” Verilog libraries◆ Configuration and libraries ◆ Command-line options ◆ Behavioural modelling

Behavioural Verilog

Algorithmic coding ◆ Synchronization using waits & event control ◆ Concurrent-disabling of always blocks ◆ Named events ◆ Fork & join ◆ High-level modelling using tasks, Implicit FSMs and concurrent-disabling ◆ Understanding intra-assignment controls ◆ Overcoming clock skew ◆ Blocking and nonblocking assignments ◆ Continuous procedural assignment

Gate Level Verilog

Structural Verilog ◆ Using built-in primitives ◆ Net types & drive strenghts ◆ UDPs ◆ Gate, net & path delays ◆ Specify blocks ◆ Smart paths ◆ Pulse rejection ◆ Cell library