



# Comprehensive Verilog®

## 4 jours



### Présentation

**Comprehensive Verilog®** est un cours de 4 jours traitant de l'application du langage Verilog® pour la conception de composants programmables et d'ASICs. Il couvre le langage Verilog®, le codage en vue de synthèse RTL (Register Transfer Level), la description des adaptateurs de test et l'utilisation d'outils Verilog®. Ce cours comprend également une présentation de SystemVerilog.

Doulos étant indépendant des fournisseurs d'outils, les participants peuvent choisir leurs outils de simulation, de synthèse, de conception de logique programmable durant les applications pratiques.

Les applications pratiques sont basées sur des exercices soigneusement préparés pour renforcer l'apprentissage. Ils comprennent en moyenne 50% du temps de la formation.

### Objectifs pédagogiques

- Formation pour la réalisation d'un premier projet Verilog.
- Consolidation et approfondissement de la connaissance du langage Verilog.
- Transfert des connaissances du langage VHDL aux applications basées sur le langage Verilog.

### Contenu de la formation

- Comment Verilog® se situe dans le design flow FPGA / ASIC.
- Comment utiliser le langage Verilog® pour la conception hardware et la synthèse logique.
- Comment décrire en Verilog® les adaptateurs de test pour vérifier vos conceptions.
- Comment éviter les erreurs classiques

### Connaissances requises

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

### 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 oeuvre vos connaissances.
- Le "Doulos Golden Reference Guide", aide-mémoire Verilog® complet et pratique (syntaxe, sémantique et astuces).
- Les "Tool Tour Guides" (pour mettre en oeuvre rapidement les outils et technologies de votre choix)
- Le CD-ROM Multimédia Interactif "Verilog Pacemaker" permettant une préparation au cours et un rafraîchissement ultérieur des connaissances.

---

## Structure et contenu

What is Verilog®? • Scope of Verilog • Design flow for ASICs, CPLDs and FPGAs • Introduction to synthesis • Synchronous design • Timing constraints • Verilog books and internet resources

### Modules

Modules & ports • Continuous assignments • Wire assignments • Comments • Names • Nets and strengths • Design Hierarchy • Module instances • Primitive instances • Text fixtures • \$monitor Initial blocks • Logic values • Vectors • Registers

### Numbers, Wires and Regs

Numbers • Output formatting • Timescales • Always blocks • \$stop and \$finish • Using wires and registers correctly

### Always blocks

Event control • If statements • Begin-end • Incomplete assignment and latches • FPGAs and latches • Unknown and don't care • Conditional operator • Tristates

### Procedural statements

Case, casez and casex statements • full\_case and parallel\_case directives • For, repeat, while and forever loops • integers • Self-disabling blocks • Combinational logic synthesis

### Clocks and Flipflops

Synthesising flip-flop & latches • Avoiding simulation race hazards • Nonblocking assignments • Asynchronous & synchronous resets • Clock enables • Synthesizable always templates • RTL synthesis technology • Inferring flip-flops • Making best use of RTL synthesis

### Operators, Parameters, Hierarchy

Bitwise, reduction, logical and equality operators • Part selects • Concatenation & replication • Shift registers • Conditional compilation • Parameters • Hierarchical names

### Finite State Machines

Designing state machines • State machine architectures • Verilog code-based FSM strategy • State encoding • Unreachable states & safe design practices • One-hot machines

### Synthesis of Arithmetic and Counters

Arithmetic operators and their synthesis • Signed and unsigned values • Adder architectures WYSIWYG arithmetic synthesis • Resource sharing • Integer and vector arithmetic

### Tasks, functions and memories

Understanding tasks • Task arguments • Task synchronization • Tasks and synthesis • Functions Memory arrays • RAM modelling and synthesis • \$readmemb and \$readmemh

---

## Structure et contenu... suite

### Test Fixtures

Designing test fixtures • Writing to files • File access using MCDs • Reading from files • The Verilog® Programming Language Interface (PLI) • Automated design verification using Verilog® • Force and release • Gate-level simulation • Back annotation using SDF • PLD and ASIC design flow • Verilog 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 • Understanding unsynthesizable Verilog constructs

### Project Management & Good Practice

Writing Verilog® for simulation and synthesis • Coding standards for synthesis • File organisation Design data control • Functional validation • Methodical testing • Hierarchical design • Gated clocks Asynchronous design

### Supplementary Subjects

#### The PLI

What is the PLI ? • What is the PLI for ? • How to use the PLI ? • TF, ACC and VPI routines • creating tests in C

#### Gate Level Verilog®

Structural Verilog® • Using built-in primitives • Net types & drive strengths • UDPs • Gate, net and path delays • Specify blocks • Smart paths • Pulse rejection • Cell library modelling

#### Verilog®-2001

What is Verilog®-2001 ? • “Cosmetic” changes • General enhancements • Parameterisation and generate • Configuration and libraries • File I/O

#### SystemVerilog

Background • Who is SystemVerilog for? • Current status of SystemVerilog • RTL enhancements • Interfaces • Assertions • Testbenches • C interface

Verilog is a registered trademark of Cadence Design Systems Inc

---

## Cours avancés

Si vous avez apprécié ce cours, et après une mise en pratique de celui-ci, N'hésitez pas à nous rejoindre aux cours :

Expert Verilog® (qui comprend les modules Expert Design et Expert Verification Verilog®)  
Comprehensive SystemVerilog  
Modular SystemVerilog (including tool specific variants)