Advanced Logic Synthesis for Electronics
www.ALSE.fr

# SystemVerilog for Design

## Formation de 3 jours

DOULOS

### Présentation

SystemVerilog (IEEE 1800™) est un « super-langage » dérivé du langage de description matériel Verilog qu'il étend sur de nombreux aspects pour en faire le langage universel du présent et du futur, aussi bien pour la conception que pour la vérification des circuits et systèmes numériques. Pour la conception (synthèse, RTL), les améliorations sont conséquentes. Mais c'est surtout dans les domaines de l'Architecture, la Modélisation et la Vérification de systèmes complexes que ce langage montre toute sa puissance (Types de données, Objectisation et Classes, Assertions SVA, Cover-driven Constrained Random generation, etc).

*System Verilog for Design* est une Formation dense qui permet d'acquérir les bases solides pour utiliser ce langage avec profit dans le contexte d'une utilisation RTL. Elle est pratiquement suffisante pour les Ingénieurs de Conception (codage RTL et vérification unitaire par bancs de tests). Et elle offre les bases indispensables aux ingénieurs de Vérification et modélisation qui souhaiteraient aborder plus tard les compléments (Les Classes et les Méthodologies de Vérification par exemple).

Les participants peuvent utiliser au choix différents outils de Conception, Synthèse et Simulation (FPGA ou ASIC) durant les exercices pratiques qui occupent environ 50 % du temps de la formation. Ces exercices progressifs, complets et soigneusement choisis, facilitent et renforcent l'acquisition des connaissances.

Les Instructeurs ALSE sont aussi et surtout des Experts en Conception qui utilisent à journée entière les langages qu'ils enseignent pour concevoir et vérifier des systèmes complexes. Ils savent partager leur savoir-faire avec passion et sont particulièrement appréciés des participants.

### Objectifs pédagogiques

- Comprendre les évolutions et les besoins qui ont conduit à la richesse du langage SystemVerilog.
- Apprendre et maîtriser l'ensemble du langage SystemVerilog hormis la parti objet (classes).
- Savoir utiliser efficacement le langage pour la conception RTL et la vérification unitaire.
- Évoluer des techniques de tests unitaires vers des méthodes plus sophistiquées permises par les nombreuses extensions du langage.
- Comprendre l'évolution des outils de conception et de vérification et l'adoption du SystemVerilog.
- Les responsables de groupes pourront préparer efficacement les transitions méthodologiques.

### Qu'apprendrez vous ?

Le cours est structuré en différentes sections :

- *Fundamentals of SystemVerilog for Design* apprend à utiliser SystemVerilog pour la conception RTL (synthèse), et aborde l'utilisation du langage pour la vérification.
- *SystemVerilog Assertions* enseigne la partie du langage qui est dédiée aux différents Layers des Assertions, et permet d'en tirer avantage pour construire des modèles et des règles de vérification.
- *Module-based SystemVerilog Verification* montre comment utiliser SystemVerilog pour adresser les challenges de la vérification des designs actuels dont la complexité exige des bancs de test et des modèles sophistiqués. Les types de données avancés, la génération aléatoire contrainte et la couverture fonctionnelle sont les éléments-clés de cette méthodologie.

### ☞ Connaissances requises (Attention)

*Une connaissance préalable sérieuse du langage Verilog est indispensable.*

### 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 HDL. 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 des Exercices pratiques, qui permet de mettre en œuvre les connaissances théoriques.
- Le « Doulos Golden Reference Guide » : Aide-mémoire SystemVerilog
  très exhaustif et pratique (syntaxe, sémantique, trucs et astuces).
  Il est destiné à devenir pendant longtemps le compagnon de votre clavier...

# Programme

## Fundamentals of SystemVerilog for Design (Jour 1)

**The SystemVerilog Data Type system**
- enum
- typedef ◆ struct
- union ◆ packed/unpacked
- Packages and $unit
- Using arrays in SystemVerilog
- array and structure literals, assignment patterns

**Nets and variables**
- Key changes in Verilog-2005 and SystemVerilog
- Continuous assignments to variables
- Modified driver and connection rules
- Data types on ports and nets

**Modules and processes**
- Port connection shorthand
- Type parameters
- Synthesis idioms for processes for better expressiveness
- Miscellaneous improvements to the language

**Design applications of interfaces**
- The interface construct
- Interfaces to encapsulate communication
- modports
- Synthesis of interfaces and modports
- Imported functions for design

…/...

## SystemVerilog Assertions (Jour 2)

### Introduction to Assertions

- Assertions, properties, sequences
- Clocking and sampling
- Property implication
- Uses of assertions
- Simulation of assertions
- Formal tools

### Assertion methodology

- Methodology consequences of assertion-based design and verification
- Assertion and assumption
- Benefits of assertions to the designer
- Protocol checkers

### Introduction to SVA syntax

- Writing simple assertions of your own
- Sequences and the ## operator
- Repetition and time ranges
- Sequence fusion
- Overview of temporal operators
- Local variables and actions in assertions

### Packaging assertions

- Assertions in Interfaces and Modules
- The *bind* construct
- Deploying Verification IPs, particularly assertion-based IPs.

## Module-based SystemVerilog Verification (Jour 3)

### Verification for Design groups

- Bus functional models
- Testbench architecture in classic Verilog
- Stimulus and response timing
- Using SystemVerilog to construct module-level testbenches
- Clocking and program blocks ◆ Testbench applications of interfaces
- Building libraries of stimulus patterns (sequences)
- Writing test cases to control the testbench

### Dynamic data types

- Strings
- Queues
- Dynamic arrays ◆ Associative arrays
- Queue and array methods
- foreach loop

### Testbench automation

- Brief introduction to testbench automation concepts
- Randomisation, checking and coverage
- The need for constraints
- Randomisation of stimulus data using std::randomize and traditional Verilog distribution functions
- Procedural randomisation: randcase, randsequence ◆ Collecting functional coverage data