Advanced Logic Synthesis for Electronics
www.ALSE.fr

Class-based SystemVerilog Verification

Formation de 2 jours

DOULOS

Présentation

SystemVerilog (IEEE 1800™) est le « super-langage » qui succède au langage Verilog pour la description matérielle (HDL) mais surtout pour la Verification (HVL).

On peut le décomposer en plusieurs parties qui répondent à des besoins et à des utilisations complémentaires :
1- La partie « Design », base du langage, indispensable aux concepteurs et pour la vérification unitaire
2- **La partie « Objet » ou Classes SystemVerilog pour Vérification (cette formation)**
3- La partie Méthodologies de Vérification dont UVM est aujourd'hui la convergence.

Cette formation de deux jours fait donc le lien entre les formations *SystemVerilog for Design* (3 jours) et *Enhanced UVM Adopter class* (4 jours) dont elle est une introduction indispensable.

Les Instructeurs ALSE sont aussi et surtout des Experts en Conception qui savent partager leur savoir-faire avec passion et sont particulièrement appréciés des participants.

Les exercices pratiques, qui occupent environ 50 % du temps de la formation, sont également des éléments clés du succès de ces formations, ils sont indispensables pour un réel apprentissage.
Ils assurent également un contrôle continu de l'acquisition des compétences.

Objectifs pédagogiques

- Apprendre et maîtriser la partie « **Objets** » du langage SystemVerilog (**classes**).
- Appliquer ce savoir pour les **classes de vérification** et la génération de **stimuli aléatoires contraints**.
- Découvrir les principes des **Librairies Méthodologiques de Vérification**.

Qu'apprendrez vous ?

- La partie « Objets » (Classes) du SystemVerilog appliquée à la Vérification (Transactions, Interfaces virtuels, TLM & Channels...)
- L'utilisation des Classes pour la génération de stimuli aléatoires contraints
- La couverture fonctionnelle, les moniteurs et les Checkers
- Les process et les événements.

Connaissances préalables requises

Une connaissance préalable du langage SystemVerilog (hormis la partie Objet) est indispensable.
Cette formation est donc une suite logique à la formation « *SystemVerilog for Design* » ou à « *SystemVerilog for VHDL Designers* ».

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 :

- Les Classeurs du cours théorique, avec Index.
  Ils constituent un Manuel de Référence concis.
- Les Cahiers des Exercices pratiques, qui permettent de mettre en œuvre les connaissances théoriques.
- Les « Doulos Golden Reference Guides » :
  Aide-mémoires SystemVerilog et UVM, très exhaustifs et pratiques (syntaxe, sémantique, trucs et astuces). Il sont destinés à devenir pendant longtemps les compagnons de votre clavier...

Programme

Class-based SystemVerilog for Verification

Introducing Classes for Transactions

• Constrained Random Verification • Representing Transaction Data • SystemVerilog Classes
• Object = Instance of Class • Constructor • Constructor Arguments

Class Members and Copying

• Static Data Members • Constant Data Members • Randomized Data Members
• Data Members of Class Type • Forward Typedef • Object Copy with new
• Shallow Copy • Deep or Shallow Copy?

Virtual Interfaces

• Test Harness and Testbench • Modules versus Classes • Creating the Testbench
• Virtual Interface • Building a test harness • Adding a clocking block
• Connecting the virtual interface • Accessing a Task through a Modport • Testbench Static Structure
• BFM or Driver Class • Testbench Object Structure

Extending Classes for Stimulus

• Improved Generator Class • Constrained randomization • Creating an Extended Class
• The Inheritance Relationship • Inheriting Class Members • Control Knobs and Constraints
• Methods of Extended Class • Derived-class Object, Base-class Variable
• Virtual Methods • General-Purpose Infrastructure

TLM and Channels

• Reusable Verification Environments • Transaction Level Modeling • Using Channels
• Generic Channel and Transaction Classes • Out-of-Block Declarations • Connecting Channels
• Getting Data from a Generic Channel • Safe Downcasting with $cast • Type Parameterization of Classes
• Running Components with fork...join • fork...join_none • Identifying Forked Processes

Component Hierarchy

• Testbench Component Hierarchy • Implementing Relationships • Base Classes (review)
• Abstract Class and Pure Virtual Methods • Interface Classes in IEEE 1800-2012
• Component Base Class • Launching a Task with fork...join_none • Customising a Component
• Constructing a Component

Monitors and Checkers

• Kinds of BFM-Like Component • Monitors and Checkers • Bus Protocol Checking
• Modports for Driver and Monitor • Monitor Implementation • Using the Monitored Transactions
• Checker Implementation • Mutual Exclusion • Semaphore Class • Checker with Mutual Exclusion

Functional Coverage

• Coverage Driven Verification • Verification Planning • From Features to Tests
• Covergroups • Embedded Covergroups • Procedural Sampling • Arguments and Options
• Coverage Bins • Bins and Coverage • Transition Coverage • Cross Coverage
• Cross Coverage and Labels • Cross Coverage Example • Controlling Cross Bins

More on Constraints (Optional Topic)

• Inline Constraints • Overriding Constraints • Procedural Control of Randomization
• Procedural Control of Constraints • Constraint Ordering • Function Calls within a Constraint
• Constraining Dynamic Arrays • Constraining an Array-of-Objects • Arrays within a Constraint
• Hierarchical Constraints • unique • Soft Constraints

Processes and Events (Optional Topic)

• The std Package • What is a “process”? • fork...join • fork...join_none • fork...join_any
• wait fork • disable fork • Identifying Processes • Fine-grain Process Control
• Process Control Example • Mailbox Class • Using Mailboxes • Enhanced Events