Advanced Logic Synthesis for Electronics
www.ALSE.fr

Vérification avec cocotb

Formation de 4 jours

DOULOS

Présentation

Cocotb : "**co**-routine based **co**-simulation **t**est **b**ench" est un package Python qui permet de faire de la simulation RTL pilotée en Python.

Python est devenu un langage universel très populaire dans les domaines scientifiques et industriels et il dispose d'une librairie immense d'extensions (plus de 400 000).

Cela permet ainsi de modéliser sans effort dans l'environnement cocotb des périphériques, des types de données et des protocoles complexes pour créer des bancs de test puissants et efficaces.

- Cocotb permet d'écrire des bancs de test en Python.
- Cocotb est indépendant du/des langages HDL utilisé(s) : VHDL, Verilog ou SystemVerilog.
- Cocotb fonctionne avec pratiquement tous les simulateur HDL
- Cocotb est Open Source et Gratuit !
- Cocotb est très utilisé en vérification FPGA, mais il peut aussi être utilisé dans des projets ASIC ou SoC.

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 IPs et des systèmes complexes. Ils savent partager leur forte expérience et leur savoir-faire avec passion, et sont particulièrement appréciés des participants.

Lieu	Enseignement en ligne par un Instructeur qui dispense le cours interactivement, répond aux questions en temps réel, et surveille ensuite la bonne exécution des exercices en fournissant l'aide individuelle éventuellement nécessaire.
Matériel	PC Windows ou Linux (ou Mac) avec un écran confortable. Pour suivre le cours : salle équipée en vidéo conférence, ou utilisation de Teams. Un environnement calme et non perturbé est indispensable.
Exercices	Les exercices seront exécutés dans le cloud, pas de logiciels ni de licence à installer.

Objectif pédagogique

- Maîtriser cocotb pour être directement opérationnel avec cette méthodologie de vérification.

Qu'apprendrez vous ?

- Principes de la vérification Hardware.
- Rappels essentiels du langage Python.
- Les principes de cocotb - Construire l'environnement de test
- Stratégies de vérification : tests directs, conditions aux limites, bancs de tests text-based, random contraint, coverage, création de composants réutilisables de vérification etc...
- Utilisation de packages tiers.

Connaissances requises

Il est fortement recommandé d'avoir des notions de base sur le langage Python. Une connaissance de la vérification hardware (bancs de test et simulateurs HDL) est également attendue.

Au besoin, vous pouvez contacter ALSE pour vérifier que cette formation vous conviendra.

© 2026 ALSE

Programme détaillé [1] de la formation
Vérification avec cocotb

Jour 1

Bases de la vérification

⊳ Vérification : définition ⊳ Quel est le processus de vérification ?
⊳ Introduction au VHDL et à SystemVerilog ⊳ Niveaux d'abstraction courants
⊳ Vérification fonctionnelle ⊳ Approches de vérification ⊳ Simulation d'un système numérique
⊳ Outils de conception et de vérification ⊳ Flot simple de conception et de vérification
⊳ Visualiseurs de chronogrammes ⊳ Banc de test élémentaire

Bases de Python et de cocotb

⊳ À propos de Python ⊳ Caractéristiques de Python ⊳ L'univers Python ⊳ python.org ⊳ PyPI
⊳ À propos de cocotb ⊳ Pourquoi utiliser Python et cocotb pour la vérification ?
⊳ Installation de cocotb
⊳ Co-simulation avec cocotb ⊳ Développement de cocotb ⊳ IDE et simulateurs pris en charge
⊳ Interpréteur Python ⊳ Chaînes de caractères et commentaires
⊳ Fonctions ⊳ Variables et fonctions dynamiques ⊳ Coroutines
⊳ Premiers pas avec un test cocotb ⊳ Opérateurs de comparaison et booléens
⊳ Vérification des résultats ⊳ Attente de l'écoulement du temps
⊳ Exemple complet de test cocotb élémentaire ⊳ Exécution d'un test ⊳ Résumé des tests

Exercice 1 : Banc de test cocotb élémentaire

Valeurs et mise en forme

⊳ Nombres en Python ⊳ Chaînes de caractères en Python ⊳ Mise en forme simple des chaînes
⊳ Conversions de types ⊳ Valeurs logiques en VHDL ⊳ Valeurs logiques en SystemVerilog
⊳ Valeurs booléennes ⊳ Représentation des nombres positifs
⊳ Vecteurs logiques en VHDL ⊳ Vecteurs logiques en SystemVerilog
⊳ Entiers en Python ⊳ Instructions for ⊳ Expressions conditionnelles
⊳ Objets ⊳ Objets représentant les éléments du HDL ⊳ Accès à l'intérieur du design
⊳ Opérateurs ⊳ Lignes et continuation ⊳ Mots-clés réservés ⊳ Fonctions intégrées

Exercice 2 : Génération de stimuli supplémentaires

[1] Contenu non contractuel susceptible d'être mis à jour ou adapté par l'instructeur

© 2026 ALSE

Structure d'un test cocotb

⊳ Importation ⊳ Instruction from...import ⊳ Packages
⊳ Structure générale d'un test cocotb ⊳ Paramètres du décorateur de test
⊳ Makefiles cocotb ⊳ Définition des variables du Makefile ⊳ Éléments obligatoires du Makefile
⊳ TOPLEVEL et MODULE ⊳ Quelques options supplémentaires utiles du Makefile
⊳ Exemple de Makefile cocotb élémentaire ⊳ Sensibilité à la casse
⊳ Niveaux de journalisation ⊳ Seuils de journalisation ⊳ Hiérarchie de journalisation
⊳ Redirection des messages ⊳ Synchronisation temporelle des stimuli et des vérifications
⊳ Structuration des tests avec des fonctions ⊳ Structuration des tests avec des coroutines
⊳ Autres mécanismes de compilation

Exercice 3 : Banc de test mieux structuré

Élaboration d'un plan de vérification

⊳ Planification de la vérification ⊳ Tout commence par la spécification ⊳ Que faut-il vérifier ?
⊳ Définition du plan de vérification ⊳ Ne cherchez pas à tout planifier

Jour 2

Contrôle du flot et synchronisation

⊳ Instructions if ⊳ match ⊳ break ⊳ continue ⊳ Instructions while
⊳ Arguments des fonctions ⊳ Vérification des types des arguments de fonction
⊳ Façons de lancer des coroutines ⊳ Génération d'une horloge ⊳ Temporisation dans un design synchrone
⊳ Attente du front d'horloge ⊳ Horloges dans plusieurs tests
⊳ Déclencheur ReadOnly ⊳ Exemple : compteur VHDL de 8 bits ⊳ Vérification du compteur
⊳ Journalisation propre à chaque tâche

Exercice 4 : Vérification de la logique séquentielle

Chaînes de caractères et listes

⊳ Indexation des chaînes ⊳ Tranches de chaînes ⊳ Listes ⊳ Opérations sur les listes
⊳ Découpage et concaténation de chaînes ⊳ Méthodes des chaînes
⊳ Largeur de champ, justification et remplissage ⊳ Base numérique, virgule et signe

Manipulation des valeurs HDL

⊳ Logique en VHDL et SystemVerilog ⊳ Types logiques de cocotb
⊳ Nombres signés ⊳ Conversion entre nombres positifs et négatifs ⊳ Exemples en complément à deux
⊳ Extension de signe ⊳ Signé et non signé en HDL ⊳ Intervalle
⊳ LogicArray ⊳ Exemples de LogicArray ⊳ Manipulation directe de LogicArray
⊳ Affectation d'éléments et de tranches de LogicArray ⊳ Attribut .range de LogicArray
⊳ Conversions de LogicArray en entier non signé ⊳ Conversions de LogicArray en entier signé
⊳ Conversions de LogicArray en chaînes de caractères ⊳ Comparaison des représentations de LogicArray

Exercice 5 : Gestion de différents types de valeurs

Fichiers et exceptions

⊳ Bancs de test et fichiers texte ⊳ Écriture dans un fichier ⊳ Lecture d'un fichier
⊳ Qu'est-ce qu'une exception ? ⊳ Gestion des exceptions ⊳ En savoir plus sur les exceptions
⊳ Gestionnaire de contexte ⊳ Module os ⊳ os.environ ⊳ os.path ⊳ pathlib
⊳ Exécution de programmes du système d'exploitation ⊳ Configuration du banc de test

Exercice 6 : Fichiers texte

Packages de vérification supplémentaires

⊳ Approches ascendante et descendante ⊳ Méthodes de vérification
⊳ Stratégies : analyse pour définir les tests ⊳ Stratégies : limites et cas extrêmes
⊳ Stratégies : tests de régression ⊳ Stratégies : tests de résistance
⊳ Différents types de stimuli ⊳ Cas de test ⊳ Amélioration de la vitesse de simulation

© 2026 ALSE

Jour 3

Collections de valeurs

⊳ Autres opérations sur les listes ⊳ Boucles et listes ⊳ Tri de listes ⊳ Suppression et retrait d'éléments
⊳ Comparaison de listes ⊳ Tuple : liste immuable ⊳ Séquence ou itérateur ⊳ Objets itérables et itérateurs
⊳ Compréhensions de listes ⊳ Dictionnaires ⊳ Opérations sur les dictionnaires ⊳ Clés et valeurs
⊳ Autres opérations sur les dictionnaires ⊳ Exemple de vérification avec des listes
⊳ Exemple de vérification avec des dictionnaires ⊳ Accès aux listes HDL

Exercice 7 : Stratégies de vérification

Classes et objets

⊳ Classes ⊳ Objets multiples ⊳ Attributs et arguments de méthodes ⊳ Constructeurs
⊳ Variables de classe et variables d'instance ⊳ Façons d'appeler les méthodes ⊳ Variables et objets
⊳ Chaînes de documentation ⊳ Annotations de type ⊳ Héritage
⊳ Redéfinition de méthodes ⊳ Appels de méthodes virtuelles
⊳ Relations entre classes ⊳ Copie d'objets d'instance ⊳ Banc de test sous forme de classe

Exercice 8 : Banc de test basé sur des classes

Vérification au niveau transactionnel

⊳ Banc de test élémentaire ⊳ Les bancs de test monolithiques manquent de flexibilité
⊳ Modélisation fonctionnelle de bus (BFM) ⊳ Dissimulation de l'interface du DUT
⊳ Exemple de classe pilote ⊳ Définition de constantes
⊳ Banc de test au niveau transactionnel

Exercice 9 : Pilote APB

Transactions et randomisation

⊳ Classe de transaction ⊳ Méthodes spéciales ⊳ Méthode spéciale de comparaison
⊳ Quelques méthodes spéciales utiles
⊳ Méthodologie de vérification ⊳ Comment générer de meilleurs stimuli ?
⊳ Module random ⊳ Randomisation d'une transaction ⊳ Transactions aléatoires bien définies
⊳ Contraintes de pondération ⊳ Exclusion de valeurs ⊳ Contraintes plus complexes
⊳ Ajout de contraintes à une transaction ⊳ Stimuli aléatoires prévisibles

Exercice 10 : Transactions et randomisation

Jour 4

Composants de vérification réutilisables

⊳ Principes de conception d'un banc de test ⊳ Exemple d'environnement de vérification
⊳ Différents types de composants ⊳ Communication de transactions (bloquantes/non-bloquantes)
⊳ File d'attente cocotb comme canal ⊳ Exemple de classe génératrice de stimuli
⊳ Exemple de classe pilote ⊳ Récupération et pilotage des transactions
⊳ BFM de surveillance ⊳ Exemple de classe de surveillance
⊳ Classe initiale du banc de test de premier niveau
⊳ Accélération du développement des composants
⊳ Stabilité de la randomisation

Exercice 11 : Composants de vérification

© 2026 ALSE

Vérification et couverture

⊳ Moniteurs et vérificateurs ⊳ Que vérifions-nous ? ⊳ Fin de la simulation
⊳ Exemple de vérificateur pour une interface unique ⊳ Mécanismes de vérification
⊳ Finalisation de la méthodologie de vérification
⊳ Couverture fonctionnelle ⊳ Points de couverture ⊳ Bins de couverture ⊳ Classe de couverture générique
⊳ Composant collecteur de couverture ⊳ Classe de base du collecteur de couverture
⊳ Restitution de la couverture ⊳ Exemple de classe de couverture des transactions
⊳ Configuration de la couverture et de la randomisation
⊳ Finalisation du banc de test de premier niveau

Exercice 12 : Vérification et couverture

Données matricielles avec NumPy

⊳ Utilisation de Python pour manipuler des données de haut niveau ⊳ Tableau NumPy à une dimension
⊳ Tableau ou matrice à deux dimensions ⊳ Initialisation de tableaux ⊳ Séries arithmétiques
⊳ Copie de la forme d'un tableau existant ⊳ Tableaux aléatoires ⊳ Opérations élément par élément
⊳ Combinaison de tableaux et de scalaires ⊳ Broadcasting ⊳ Broadcasting en deux dimensions
⊳ Vectorisation d'une fonction ⊳ Tableau d'indices ⊳ Indexation avec un tableau de booléens
⊳ Autres opérations

Manipulation de données de haut niveau

⊳ Introduction à Pandas ⊳ Structures de données Pandas ⊳ Séries Pandas ⊳ DataFrames Pandas
⊳ Opérations de type SQL avec Pandas ⊳ Statistiques élémentaires avec Pandas
⊳ Transformations de données avec Pandas ⊳ Introduction à Matplotlib ⊳ Sous-graphiques Matplotlib
⊳ Tracés avec lignes, couleurs et marqueurs ⊳ Types de graphiques dans Matplotlib
⊳ Tracé d'un histogramme avec Matplotlib
⊳ Représentation d'un tableau sous forme de grille avec Matplotlib
⊳ Autres packages Python

Exercice 13 : Utilisation de packages Python supplémentaires

Méthodologies de vérification structurées

⊳ Méthodologies de vérification standard ⊳ Verification IP ⊳ pyuvm ⊳ Introduction à Forastero
⊳ Convention de nommage des ports Forastero ⊳ Exemples Forastero
⊳ Solveurs de contraintes ⊳ lambda ⊳ Introduction à constrainedrandom
⊳ Exemple avec constrainedrandom ⊳ Domaines de constrainedrandom
⊳ Introduction à PyVSC ⊳ Exemple de génération de stimuli aléatoires avec PyVSC
⊳ Exemple de covergroup avec PyVSC
⊳ Autres packages de couverture ⊳ Bibliothèque de vérification Apheleia

-=oOo=-

© 2026 ALSE