Synthèse logique et simulation VHDL pour Conception de FPGA Xilinx™

5 jours

OBJECTIFS

- Appréhender les multiples possibilités offertes par le langage VHDL
- Comprendre les notions de synthèse logique
- Connaître les styles d'écritures et leur impact sur la qualité des résultats de synthèse
- Connaître les performances pouvant être attendues des FPGA Xilinx
- Apprendre à paramétrer les options de compilation et les contraintes d'implémentation
- Manipuler les outils de debug et les rapports d'implémentation

FORMATIONS CONNEXES

- Vivado™ Design Suite : Analyse statique de timing (STA) et Xilinx Design Constraints (XDC)
- Vivado™ High Level Synthesis
- Zynq™ All Programmable SoC : Architecture Système
- Conception avec les familles Xilinx™ Série-7

PARTENAIRES

XILINX. | Authorized Training Provider

PRÉREQUIS

- Cette formation s'adresse aux ingénieurs électroniciens ayant déjà de bonnes connaissances en conception de circuit d'électronique numérique, désireux d'acquérir une solide méthodologie de conception, et de tirer le meilleur parti du langage VHDL, ainsi que des outils de synthèse et de simulation associés pour développement de FPGA Xilinx.

CONFIGURATIONS

- Configuration logicielle :
  - Xilinx Vivado™ Logic Edition 2015.x
- Configuration matérielle :
  - Ordinateur récent (i5 ou i7)
  - Windows 7 64b
  - Minimum 8Go de mémoire vive
  - Résolution d'affichage minimum 1024x768, recommandée 1920x1080
- Pour les formations sur site, prévoir un vidéo projecteur

CHAPITRES

1ER JOUR

- Architecture des FPGAs Spartan3A, Spartan6, Virtex5, Virtex6 et Série-7
  - Structure générale
  - Notions de CLB et de slices
    - Logique combinatoire et bascules
    - Logique arithmétique
    - Notions de mémoire distribuée
    - Registres à décalage SRL
  - Blocs d'entrée sortie
    - Bascules d'entrée/sortie

        - Registres DDR
        - Paramétrages électriques et de timing
    - Blocs de RAM dédiée et modes d'utilisation
        - Implémentation de FIFOs paramétrables
        - Autres exemples d'utilisation
    - Distribution d'horloges, DCMs et PLLs
        - Buffer globaux, buffers locaux
        - DCMs, PLLs et paramétrage
    - Multiplieurs dédiés et blocs DSP48
    - Configuration
        - Maitre, esclave, SPI, BPI, JTAG

2ND JOUR

- Règles d'écriture du code VHDL en synthèse logique
    - Notion d'entité/architecture
    - Instructions concurrentes et séquentielles
    - Objets et types prédéfinis
    - Opérateurs prédéfinis et d'utilisation étendue par l'utilisation de packages standardisés
    - Instructions concurrentes : when, with select, for generate
    - Travaux pratiques

3ÈME JOUR

- Règles d'écriture du code VHDL en synthèse logique (suite)
    - Les process
        - Importance de la liste de sensibilité
        - Instructions séquentielles : if, case, loop
        - Utilisation de variables
        - Quelques pièges classiques à éviter
        - Incohérences potentielles d'interprétation entre la synthèse logique et la simulation : comment s'en affranchir ?
- Gestion de la hiérarchie pour une meilleure réutilisation
    - Organisation de design par modules fonctionnels : quel découpage choisir ?
    - Notions d'inférence et d'instanciation
        - Quand doit-on instancier primitives ou macros ?
    - Précautions à prendre pour un code évolutif et/ou réutilisable
    - Importance du choix de noms des modules et des nets pour faciliter l'implémentation physique, la simulation et la mise au point
    - Doit-on préserver la hiérarchie lors de la synthèse logique ?
    - Travaux pratiques

4ÈME JOUR

- Approfondissements sur le langage VHDL pour l'optimisation et la réutilisation du code en synthèse logique
    - Notions de variables et exemples d'utilisation
    - Généricité et paramétrage automatique des modules réutilisables
    - Attributs prédéfinis utiles en synthèse logique
    - Fonctions et procédures
    - Définition de packages et librairies
    - Travaux pratiques
- Méthodologie de conception hardware en synthèse logique
    - Conception asynchrone et pièges classiques
        - Métastabilité et aléas de fonctionnement
        - Limitations de la simulation fonctionnelle et timing sur les designs asynchrones : comment s'en affranchir ?
    - Gestion d'évènements asynchrones
        - Aléatoires
        - Flots de données
    - Conception synchrone
    - Analyse statique de timing : comment l'utiliser ?
    - Optimisation de performances indépendamment de la cible
    - Notions de pipeline
    - Travaux pratiques
- Outils d'implémentation et de mise au point
    - Flot d'implémentation et de génération de bitstream
    - Analyse des rapports
    - Principales options d'implémentation
    - Outils d'analyse de résultats d'implémentation

5ÈME JOUR

- Testbenches et simulation
    - Quelques règles de base pour l'écriture d'un testbench efficace
    - Instructions VHDL spécifiques à la simulation
        - Wait et ses différentes formes
        - Boucles
        - Assertions
        - Types de données
        - Autres
    - Ecriture de modèles de composants destinés à rendre la simulation plus réaliste
    - Utilisation de modèles et packages de simulation existants
    - Travaux pratiques
    - Ecriture et lecture de fichiers ASCII
        - Affectation d'un flot de données à partir d'un fichier
        - Stockage des résultats de simulation dans un fichier
        - L'interpréteur de commandes
    - Génération de messages d'information
    - Travaux pratiques

NOTES

- Les supports de cours seront fournis sur papier à chaque participant pendant la formation.

CONTACT

Tel : 05 62 13 52 32
Fax : 05 61 06 72 60
training@mvd-training.com