

Synthèse logique et simulation VHDL pour Conception de FPGA Xilinx

Ref : 002572A

Durée : 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

- Optimisation des performances (002833A)
- Techniques d'implémentation de fonctions DSP pour FPGA Xilinx (002838A)
- Conception de systèmes avec PCI-e (004552A) ou rocket-IO (002843A)
- Implémentation de System-On-Chip à base de Microblaze (003149A) ou PowerPC (002952A)

PARTENAIRES

- Cette formation est approuvée par XILINX

PRE-REQUIS

- 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, et plus particulièrement les familles Spartan3 et dérivées.

MATERIEL DE FORMATION

Configuration logicielle :

- Xilinx ISE Design Suite 11.3 Logic Edition

Configuration matérielle recommandée :

- Intel Core 2 ou équivalent
- Windows XP
- 1 Go d'espace disque disponible après installation des logiciels
- Au minimum 1Go de mémoire vive
- Résolution d'affichage : au moins 1024 x 768
- Pour les formations sur site, prévoir un vidéo projecteur

Authorized
Training Provider

Contact

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

**Le contenu peut-être
adapté sur site**

Prochaines sessions, voir ici : <http://www.mvd-training.com/fr/schedule.html>

PROGRAMME

Architecture des FPGA Spartan3 et familles dérivées

- Structure générale
 - Notions de CLB et de slices
 - Logique combinatoire et bascules
 - Logique arithmétique
 - Notions de mémoire distribuée
 - Registres à décalage SRL16
- Blocs d'entrée sortie
 - Bascules d'entrée/sortie
 - Registres DDR
 - Paramétrages électriques et de timing et spécificités Spartan3E, Spartan3A
- Blocs de RAM dédiée et modes d'utilisation
 - Implémentation de FIFOs paramétrables
 - Autres exemples d'utilisation
- Distribution d'horloges et DCMs
 - Buffer globaux, buffers locaux (Sparta 3E/A)
 - DCMs et paramétrage
- Multiplieurs dédiés et blocs DSP48A (Spartan_3A-DSP)
- Configuration
 - Maître, esclave, SPI (Spartan3E/A)

Différences fondamentales avec l'architecture Virtex 4

Introduction à l'architecture Virtex 5

Outils d'implémentation et de mise au point

- Flot d'implémentation et de génération de bitstream
 - Translate
 - Map

- Place and Route (PAR)
 - BitGen
- Analyse des rapports MRP et PAR
- Principales options d'implémentation
 - MAP
 - Empaquetage des bascules d'los
 - CLB Pack factor
 - MAP-timing
 - PAR
 - Effort global
 - Extra effort
 - Multi Pass Place and Route
 - Autres options
 - BITGEN
 - Options de configuration
 - Options de Startup
- Outils d'analyse de résultats d'implémentation - contraintes
 - FLOORPLANNER
 - Comment analyser la qualité du placement
 - Génération de contraintes de placement
 - Par éléments
 - Par blocs fonctionnels
 - FPGA EDITOR
 - Vérification de détails d'implémentation
 - Identification des ressources arithmétiques
 - Navigation
 - Vérification d'aspects importants du routage (lignes d'horloges, accès aux BUFGs, DCMs ...)
 - TIMING ANALYZER
 - Analyse globale de la performance d'un design
 - Crossprobing entre le Timing Analyzer et FPGA Editor ou Floorplanner
 - Conclusions à tirer pour optimisation du design
- Introduction à CHIPSCOPE
- Fichier de contraintes
 - Contraintes de placement et de configuration électrique des los
 - Contraintes basiques de timing

- PERIOD
- OFFSET IN/OUT
- Contraintes avancées de timing
 - Chemins multi-cycle
 - Faux chemins
 - Limitations sur les Contraintes sur chemins reliant 2 domaines d'horloge différents
- Contraintes avancées de placement
 - AreaGroup : précautions d'utilisation
- Généricité et re-paramétrage automatique des modules réutilisables
- Attributs prédéfinis utiles en synthèse logique
- Fonctions et procédures
- Définition de packages et librairies
- Travaux pratiques

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
- 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 ?
- 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 ?
- Conception synchrone - avantages - méthodologie - mise au point
- Analyse statique de timing : comment l'utiliser ?
- Optimisation de performances indépendamment de la cible
- Notions de pipeline
- Gestion d'évènements asynchrones
 - Aléatoires
 - Flots de données
- Travaux pratiques

Approfondissements sur le langage VHDL pour l'optimisation et la réutilisation du code en synthèse logique

- Notions de variables et exemples d'utilisation

DOCUMENTATION

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

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

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 " Loop "
 - 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
- Intégration de " pseudo logique " afin de faciliter l'interprétation des résultats de simulation
- 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