http://www.alse-fr.com

Formation Pratique
Intel SoC-FPGAs

Introduction Les **SoC-FPGAs Intel** (Cyclone V, Arria V, Arria10, Stratix10...) qui embarquent les **processeurs applicatifs ARM Cortex** (**A9-MP** et Quad-core **A53**) représentent un saut majeur de technologie et de performance qui permet d'associer les bénéfices et la puissance de traitement des FPGAs avec les processeurs les plus puissants du marché, sur la même puce. C'est pourquoi nous avons créé une formation modulaire de quatre jours qui permet dans ce temps réduit de prendre rapidement en main les concepts fondamentaux ainsi que les outils associés. Cette formation fournit également des applications complètes et typiques qui serviront de base solide à vos projets.

Contenu Cette formation est proposée en **trois parties** pour un total de **quatre jours**.

La **première** partie (1 jour) concentre l'essentiel des concepts et de l'utilisation de **Platform Designer** (ex **Qsys)** pour construire et mettre au point la plateforme hardware/software du côté « FPGA » (hardware programmable) du composant SoC.

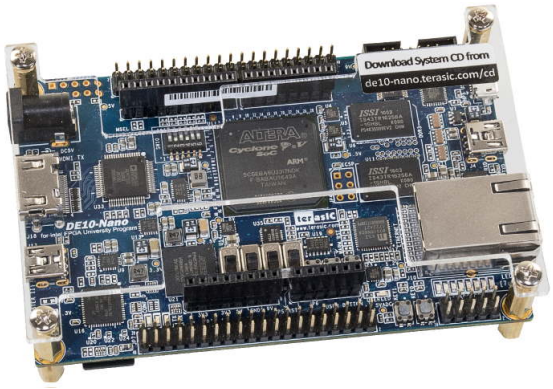
La **deuxième** partie (2 jours, non divisible) couvre :
- Les aspects matériels essentiels (configuration, horloges, reset, activation, configuration...) du côté HPS « ARM » et du côté « Programmable » , ainsi que leurs interactions.
- L'architecture de la partie HPS (processeurs et périphériques, mémoires, modules de debug, caches, sources et séquences de boot...)
- Les outils de vérification (Simulation SystemVerilog, BFMs etc)
- L'environnement de développement logiciel et de mise au point ARM DS-5.
- Les flot de conception combiné matériel+logiciel, et la mise en œuvre du système complet.

Enfin la **troisième** partie (1 jour) est conseillée à tous les ingénieurs (Hardware et Software). Elle est consacrée à la distribution **Linux Embarqué** construite par Intel. Elle est accessible même aux concepteurs encore assez novices : au besoin un cursus de préparation personnelle préalable est proposé aux inscrits qui souhaitent acquérir les bases de Linux avant le training.

L'ensemble de cette formation est **indispensable** pour construire des applications fiables et performantes qui tirent bénéfice des riches possibilités de l'architecture SoC : construction d'architecture et de *fabric* optimisées, mise en place de flots de données -notamment pour les échanges HPS ↔ FPGA-, paramétrage des interconnexions, constructions de périphériques custom, intégration dans le champs d'adressage HPS, techniques de mise au point, principe et mise en œuvre de l'accélération matérielle etc.

Les aspects Matériel **et** Logiciel sont abordés dans ce stage pour permettre aux ingénieurs hardware et software d'avoir une vue complète des deux mondes, ce qui est *indispensable* pour concevoir un système embarqué efficace et optimisé.

Théorie et Pratique alternent à travers de très nombreux **exercices** décrits pas-à-pas, implémentés et testés sur un **Kit SoC-FPGA peu coûteux** ($130) et facilement disponible (DE10-Nano kit, ci-contre. Ces exercices **indispensables** pour un réel apprentissage, **assurent** aussi **un contrôle continu de l'acquisition des compétences.**



Pré-requis Connaissances basiques en langage C et, de préférence, en conception Intel-FPGA (VHDL). Bases Linux souhaitées pour la journée 4.

Durée Trois à quatre jours de 7 heures effectives chacun.

Horaires Typiquement : 9h00 → 18h00

Contrôle Cette formation est sanctionnée par un **contrôle continu** des connaissances et par une **fiche de présence**.

Objectifs

Cette formation vous permettra (entre autres) de répondre aux questions suivantes :

- ✔ Comment tirer avantage du FPGA et de la logique programmable ?
- ✔ Construire des périphériques dans la logique programmable ?
- ✔ Piloter mes périphériques FPGA depuis les processeurs ?
- ✔ Exemples de Drivers Linux ?
- ✔ Transférer efficacement et sûrement des données entre processeurs et FPGA ? Quid du DMAC ?
- ✔ Comprendre la distribution Linux Embarqué et les outils de développement ?
- ✔ Débugger Hardware et Software ?
- ✔ Construire sa propre carte et adapter l'écosystème fourni ?

Partie I « FPGA SoC System Design using Platform Designer » - 1 jour

- Acquérir le **savoir-faire essentiel** pour utiliser **Platform Designer** (ex **Qsys)** et **construire l'architecture de la partie programmable** des SoC FPGA : découvrir les périphériques standard de la logique programmable, le processeur soft-core Nios II, et une introduction aux périphériques « Custom ».
- Savoir mettre au point la partie programmable en simulation RTL ainsi que sur cible avec SystemConsole. Utilisation de l'environnement de développement logiciel Nios II EDS / Eclipse pour valider simplement et rapidement la partie FPGA (bare metal).

Partie II « SoC FPGA Hardware design + Software design » - 2 jours

- Comprendre et maîtriser l'architecture interne des SoC FPGAs Intel avec processeurs embarqués ARM Cortex A9-MP (et A53) pour développer des systèmes complets, utiliser les outils fournis par Intel.
- Comprendre la configuration du système avec les BSP Editor et les étapes de démarrage (Boot).
- Comprendre l'environnement logiciel embarqué (dont ARM-DS5) proposé par Intel et savoir développer des applications logicielles prenant en compte les parties programmables développées par l'utilisateur.

Partie III « Linux Embarqué pour Intel SoC FPGAs » - 1 jour

- Prendre rapidement en main la distribution Linux Embarqué proposée par Intel, comprendre les fondamentaux, découvrir le *Device Tree Generator*, voir comment prendre en compte un périphérique utilisateur côté FPGA, et savoir développer et débugger une application Linux simple.

Pré-requis

- Connaissances de base en Électronique Numérique et en Conception de Logique Programmable.
- Connaissances de base (rudiments) en Langage C.
- Si possible rudiments de Conception FPGA (VHDL ou Verilog) et outils Intel.
- Pour le jour 4 : les bases de Linux (acquisition possible par auto-apprentissage préalable).

NB : la formation « Conception avec Quartus » n'est pas un pré-requis, elle peut être suivie séparément.

Remarque: les Exercices pratiques de la formation utilisent un Kit peu coûteux et facilement disponible (non fourni), ce qui permet de pouvoir les reproduire ultérieurement après la formation.

http://www.alse-fr.com

Formation
Intel SoC FPGA

Programme *

Partie I :
Conception de la partie Programmable avec Qsys & Nios II

Journée 1

- **Les bases de la Conception d'un « System on Programmable Chip » avec Platform Designer**
  Introduction à l'écosystème Intel SoPC.
  Introduction et utilisation de Platform Designer (ex Qsys) pour construire un *System on Chip*.
  Introduction aux processeurs NIOS II et à leur environnement.
  Les Périphériques Utilisateur usuels (SystemID, Jtag UART, On-chip Memory, PIO, Timer, Nios...)
  Exercice : Création d'un système SoPC complet avec périphériques Avalon-MM en utilisant Qsys, puis test sur le SoCKit.
- **Platform Designer** (Qsys) **Interconnect Fabric**.
  Principes, Protocoles supportés, paramètres et maximisation de performance.
  Avalon MM et les différents modes de transfert,
  Avalon ST, Périphériques mixtes et datapath processing engines.
  Découvrir et utiliser le GUI et l'éditeur d'architecture.
- Les **Périphériques « Custom »**
  Principe et utilité des périphériques utilisateur.
  Créer facilement un périphérique utilisant l'interfaces Avalon MM.
  Le Component Editor. Créer et publier le périphérique custom.
  Exercice Pratique : Conception VHDL et mise en œuvre d'un Périphérique utilisateur custom
- Validation de la plate-forme matérielle par **(co)simulation HDL avec ModelSim**.
  Création des modèles de simulation, modélisation des composants périphériques,
  Utilisation du Simulateur ModelSim Intel Edition.
  Exercice optionnel : Simulation RTL par ModelSim AE du système SoPC.
- *OPTION* : Introduction à l'utilisation de **SystemConsole**
  Principe et utilisation basique de SystemConsole pour accéder au système Qsys.
  Exercice Pratique : Mise au point sur cible à l'aide de l'outil System Console
- APPENDICE optionnel :
  Introduction à l'Environnement de Développement Logiciel **Nios II EDS / Eclipse**
  Construire un projet logiciel. Composer le code. Compiler et générer le code.
  Charger, Exécuter et Débugger le code avec Eclipse.
  Exercice Pratique : Création d'un projet, codage et exécution du test pour le périphérique custom

NB : cette journée démontre qu'il peut être très intéressant -au moins dans certaines phases au début du projet- d'utiliser un softcore Nios II dans un système SoC !

* : Contenu et répartition sont fournis uniquement à titre indicatif.
Ils sont également susceptibles d'être modifiés ou adaptés par l'instructeur.

Partie II : Intel SoC FPGAs – Aspects Hardware & Software

Journée 2 – Aspect Hardware

- Présentation des **familles Intel SoC FPGA** avec Processeurs embarqués **ARM Cortex** (A9-MP et A53) et des **Kits** de développement.
- Présentation des principaux constituants du **système HPS** (processeurs, périphériques intégrés, les interfaces mémoires, les différents niveaux de caches, les interconnexions et les flux de données, les modules de gestion d'intégrité et de debug...). Le modèle Mapping Mémoires.
- **Le Système HPS**
  Sources d'horloge, PLLs, reset, FPGA Manager, FPGA configuration, System Manager, Scan manager, source de boot, mode Safe, L3 Interconnect, Bridges AXI, configuration des flux de communication avec la partie « User » (Programmable), L4 Peripheral bus.
  Les périphériques HPS (Ethernet, Flash, USB, SPI, CAN, I²C, UARTs, Timers Wtachdog, GPIO...).
  On-Chip Ram/Rom, SDRAM controller.
  Le contrôleur DMAC quand l'utiliser (et non).
- Dichotomie : Logic Programmable (**User Logic**) ↔ HPS (**Hard Processor System**).
  Les différentes interfaces disponibles.
- Construction de **la partie Programmable et configuration du bloc HPS**.
  Problématiques de pin-assignment et pin-sharing/muxing.
  Comprendre les différences et utiliser à bon escient les différents liens possibles.
  Les standards supportés : Avalon-MM, Avalon ST, AXI-4, AXI-3, AHB, APB
  Mapper des périphériques Custom dans le champ d'adressage HPS
  Accéder depuis le FPGA aux périphériques HPS.
  Exercice pratique : Golden Hardware Reference Design avec Périphérique Utilisateur.
- Introduction aux **BFMs SystemVerilog** et à l'environnement de **Vérification**.
  Exercice pratique : Utilisation et simulation du système Qsys complet dans Modelsim-AE
- *OPTION (si l'agenda le permet et si pas déjà couvert en jour 1)* :
  Introduction à l'utilisation de **SystemConsol** pour accéder au système Qsys.
  Exercice Pratique : Mise au point sur cible à l'aide de l'outil System Console

Journée 3 – Développement logiciel sur SOC-FPGA ARM

- Comment **accéder** aux **périphériques HPS** et **FPGA** en fonction du master utilisé (MPU, FPGA, HPS)
- Passage d'informations (***Handoff***) **Hardware → Software** : principe et fichiers.
- Présentation de l'**Environnement de développement ARM : DS-5**
  Fonctions générales, composition de code, compilation, Maîtriser le lancement et la synchronisation avec la cible, savoir récupérer suite à une désynchronisation.
  Exercice 0 : HPS « Hello World »
- Comprendre toutes les **étapes** du **démarrage** du système (du Boot initial jusqu'à l'Operating System Linux) ainsi que les options des différentes phases.
- Le **BSP Editor**.
  Comprendre le fonctionnement du **Second Stage Boot Loader** (SSBL) et les différentes tâches qu'il assume.
  Exercice pratique : génération du preloader (SSBL sur Cyclone V), inspection, insertion d'un code utilisateur et debug bare-metal avec DS-5.
- **HPS Flash Programmer.**
- Bare-Metal : ***SoCAL*** et ***HWLIB***. *État actuel et roadmap. Limitations. AMP vs SMP.*
  Exercice : Compilation eet debug en Bare-Metal
- **Mise au point combinée** Matériel-Logiciel.
  Exercice pratique : Debugging Adaptatif et Cross-triggering
- Introduction à l'offre actuelle en **Operating Systems Embarqués** et de la distribution **Linux** proposée par **Intel** (cette dernière fait l'objet de la troisième partie du training).

Partie III – Linux Embarqué pour Intel SoC FPGAs

Journée 4

- Rappel des OS disponibles sur SoC FPGAs.
- Qu'est-ce que **Linux Embarqué** ? Principes et Jargon. BareMetal vs OS.
  Rappels des concepts et des commandes essentielles qui seront utilisées dans la journée.
- Description rapide de la **distribution Intel**.
  Le projet ***Yocto*** : pour qui et pour quoi. Intel SoC Linux BSP.
  Présentation et utilisation de ***Buildroot***.
  Démonstration de customisation et de génération de la distribution. Création de l'image SDMMC.
  Exercice pratique : Comment construire l'image Linux, programmer la SD-Card et booter Linux.
- Les principales **commandes** shell **Linux**. Les principaux services disponibles.
  Le File System, tty, BusyBox, Runlevels, Boot Network management, montage des FS,
  process control et Kernel logs.
  Exercice pratique : Prise en main le l'OS Linux embarqué sur SoCkit.
  Boot, ligne de commande, services, gestion du réseau Ethernet, montage de volume, etc
- **uBoot**. Principe, utilité, les commandes, environnement et variables, scripts, Mise en œuvre.
  Exercice pratique : booter depuis une image réseau.
- Accès au périphériques : le **Device Tree et les Drivers**.
  Comment prendre en compte les périphériques utilisateur (custom) situés dans la partie programmable sans recompilation du noyau.Exemple, Génération du Device Tree, support soft IPs.
  Les Drivers : User space vs Kernel space. Types et classes de drivers:Caractères, blocs et paquets.
  Interface avec le DT. Compiler et insérer un module.
  Exercice pratique : création d'un driver simple pour un périphérique custom avec interface registres.
- L'Environnement de **Cross-Développement Logiciel**. grâce au plugin Buildroot pour Eclipse.
  Installation et configuration du plugin.
  Créer le projet, Coder, compiler, exécuter et debugger une **Application Linux** faisant appel aux périphériques user.
  Remote debugging.
  Exercice pratique : Utilisation du driver précédent dans une application.

-=oOo=-