

## H1 - Lattice Mico32 - Processeur embarqué

*Mise en oeuvre et programmation d'un processeur Soft-Core dans un FPGA*

### Objectifs

- Comprendre les phases de création d'un système intégrant un cœur de processeur
  - Assemblage de la plateforme matérielle
  - Installation du FPGA
  - Programmation du logiciel applicatif
  - Intégration finale
- Maîtriser les particularité du cœur Open Source MICO32:
  - Architecture logicielle
  - Bus Wishbone
  - Périphériques standard: UART, Ethernet (10/100/1000)...
- Apprendre à programmer une plateforme à cœur programmable à travers Eclipse.
- Découvrir l'installation de Micrium uC/OSII et uClinux sur la plateforme

Ce cours apprend à maîtriser la création d'une plateforme utilisant un CPU embarqué. Tous les exercices sont faits sur une carte Lattice ECP2 avec le cœur Open Source MICO32; l'installation de uc/OSII et de uClinux seront présentées rapidement.

### Matériel

- Un PC Windows par binôme avec
  - L'outil Lattice Diamond de programmation du FPGA
  - L'outil de création de plateforme Open Source Mico32
  - L'environnement de développement logiciel (basé sur Eclipse)
  - L'installation de uClinux et uC/OSII pour la plateforme Mico32
- Une carte cible Lattice ECP2
- Support de cours imprimé
- Présentation et solutions des exercices

### Pré-requis

- Compréhension de base de l'architecture des processeurs
- Bonne connaissance de la programmation VHDL (niveau cours [V1 - Les bases du langage VHDL](#))
- Connaissance de la programmation embarquée en C (si possible niveau cours [L2 - C language for Embedded MCUs](#))
- Pour une bonne compréhension de l'installation de uClinux il est souhaitable d'avoir une connaissance de base de
  - Linux Embarqué (voir cours [D1 - Linux embarqué avec Buildroot et Yocto](#)) pour comprendre le processus de démarrage de uClinux
  - la programmation Linux (voir cours [D0 - Programmation en mode utilisateur Linux](#)), pour les exercices de programmation Linux

### Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais).
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

### Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

# Course Outline

## Premier jour

### Méthodologie de développement d'une plateforme

- Définition de la plateforme
  - Création de l'architecture matérielle du système
  - Choix des caractéristiques et dimensionnement
- Génération du code VHDL de la plateforme
- Implémentation de la plateforme
  - Vérification de la plateforme
  - Création du bitstream de programmation du FPGA
- Programmation de la Plateforme
  - Choix de l'infrastructure logicielle
  - Programmation en C et assembleur

### Définition de la plateforme avec Mico System Builder

- Choix des options du cœur
  - caches
  - contrôleur d'interruptions
- Choix et configuration de la mémoire
  - RAM et ROM internes
  - Flash externe, série ou parallèle
  - SRAM ou DDRAM externe
- Choix et configuration des périphériques
  - GPIOs
  - Timer
  - UART, SPI, I2C
  - Ethernet
  - Contrôleur de DMA
- Stratégies d'arbitrage du bus
  - Bus privés ou partagés
  - Arbitrage fixe ou dynamique
- Configuration de la plateforme
  - Connection des périphériques aux bus
  - Connection des périphériques aux contrôleurs d'interruptions et de DMA
  - Choix des adresses et des interruptions
- Vérification de la cohérence de la plateforme
- Génération du code VHDL de la plateforme

## Second jour

### Implémentation de la plateforme

- Simulation comportementale
  - Utilisation du TestBench généré par le MSB
- Synthèse en fonction du FPGA choisi
  - Définition des entrées/sorties
  - Placement
  - Routage
- Simulation après routage

- Vérification des timings
- Contrôle des vitesses de traitement
- Programmation du FPGA
  - Génération du bitstream
  - Transfert sur la cible

## Programmation de la Plateforme

- Programmation en C
  - L'environnement de programmation basé sur Eclipse
  - Environnement d'exécution des programmes
  - Contraintes de programmation et de génération des programmes
  - Définition de la mémoire pour l'éditeur de liens
- Simulation sur la station de développement
  - Utilisation du simulateur de plateforme MICO32
- Transfert sur la cible
- Debug croisé
  - Utilisation de GDB pour débogger le programme sur la cible

## Installation et utilisation de uC/OSII

- Installation de micrium uC/OSII sur la cible
  - Besoins spécifiques à uC/OSII à satisfaire sur la plateforme
  - Configuration de uC/OSII en fonction de la plateforme
  - Création d'un programme simple
  - Recompilation
  - Transfert sur la cible
  - Debug croisé

## Troisième jour

### Installation et utilisation de uClinux

- Installation d'u-boot sur la cible
  - Configuration en fonction de la plateforme
  - Recompilation de u-boot
  - Transfert sur la cible
  - Auto-test de la plateforme par u-boot
- Installation de uClinux
  - Configuration du noyau Linux
  - Choix des paramètres de boot
- Création de programmes pour uClinux
  - Compilation sous Eclipse pour uClinux
  - Debug croisé

### Création de composants spécifiques

- Définition de composants d'entrée/sortie spécifiques
  - Création du code VHDL du composant
  - Intégration dans le Mico System Builder
  - Création d'une plateforme incluant le nouveau composant
- Utilisation de composants spécifiques
  - Création d'un programme manipulant le composant
  - Notion de driver

### Déploiement du système

- Déploiement du bitstream en flash SPI
  - Utilisation du port JTAG pour programmer la flash
- Déploiement du code testé en flash parallèle
  - Création de l'infrastructure de programmation de la flash
  - Reconfiguration de l'application pour exécution en flash
  - Déploiement d'une application intégrée
  - Déploiement d'un système uClinux complet