



H1 - Lattice Mico32 FPGA embedded processor

Implementing and programming a processor core in an FPGA

Goals

- Understand the process of creating a system with an FPGA-embedded processor
 - Assembling the hardware platform
 - Installation of FPGA
 - Programming of application software
 - Final Integration
- Master the particulars of the Open Source MICO32 core:
 - Software architecture
 - Wishbone bus
 - Standard peripherals: UART, Ethernet (10/100/1000) ...
- Learn to program a the platform using Eclipse.
- Discover installing uClinux and micrium uC/OSII on the platform

This course teaches you how to control the creation of a platform using an embedded CPU. All exercises are done on a board with the Lattice ECP2 Open Source MICO32 core; installation and use of Micrium uC/OSII and uClinux will be presented quickly.

Course Material

- A Windows PC per two trainees
 - The Lattice Diamond FPGA programming tool
 - The MSB tool for creating Mico32 Open Source platform
 - The software development environment (based on Eclipse)
 - The installation of the platform on uClinux and uC/OSII
- A Lattice ECP2 target board
- Printed handouts of the course slides
- Labs presentation and solutions

Prerequisite

- Basic understanding of processor architecture
- Good knowledge of VHDL programming ([V1 - VHDL Language Basics](#) course level)
- Knowledge of embedded programming in C (if possible [L2 - C language for Embedded MCU](#) course level)
- For a good understanding of installing uClinux it is desirable to have a basic knowledge of
 - Embedded Linux (see [D1 - Embedded Linux with Buildroot and Yocto](#) course) to understand the uClinux boot process
 - Linux programming (see [D0 - Linux user mode programming](#) course) for Linux programming exercises

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First day

Platform development process

- Definition of the platform
 - Creation of the physical architecture of the system
 - Choice of characteristics and design
- VHDL code generation of the platform
- Implementation of the platform
 - Verification of the platform
 - Creation of the bitstream for programming the FPGA
- Programming the platform
 - Choice of software infrastructure
 - Programming in C and assembler

Platform definition with the Mico System Builder

- Selecting core options
 - Caches
 - Interrupt controller
- Memory selection and configuration
 - Internal RAM and ROM
 - External flash, serial and parallel
 - SRAM and external DDRAM
- Choosing and configuring devices
 - GPIOs
 - Timer
 - UART, SPI, I2C
 - Ethernet
 - DMA Controller
- Bus arbitration strategies
 - Private or shared busses
 - Static or dynamic arbitration
- Configuring the platform
 - Connection of peripheral busses to
 - Connection of peripheral interrupt controllers and DMA
 - Choice of addresses and interrupts
- Checking the consistency of the platform
- VHDL code generation of the platform

Second day

Platform implementation

- Behavioral Simulation
 - Using the testbench generated by the MSB
- Synthesis according to the chosen FPGA
 - Definition of inputs / outputs
 - Placement
 - Routing
- After routing simulation

- Checking timings
- Control processing speed
- FPGA programming
 - Generating the bitstream
 - Transfer to the target

Platform programming

- C programming
 - The Eclipse-based programming environment
 - Program Runtime Environment
 - Programming and code generation constraints
 - Linker memory definition
- Simulation on the development station
 - Using the MICO32 platform simulator
- Transfer to the target
- Cross debugging
 - Use of GDB to cross-debug the program on the target

Install and use of micrium uC/OSII

- Installing micrium uC/OSII on the target platform
 - Specific platform requirements for uC/OSII
 - Configuration uC/OSII to fit the platform
 - Create a simple program
 - Recompile
 - Transfer on the target
 - Cross-debug

Third day

Install and use uClinux

- Installing u-boot on the target
 - Specific platform requirements for u-boot and uClinux
 - Configuring u-boot to fit the platform
 - Recompile u-boot
 - Transfer on the target
 - Auto-test of the platform by u-boot
- Installing uClinux
 - Configure the Linux kernel
 - Choice of boot parameters
- Creation of programs for uClinux
 - Compilation under Eclipse for uClinux
 - Cross-debug

Creating custom components

- The Wishbone bus
 - Bus topology and signals
 - Master interfaces
 - Slave interfaces
- Defining custom input/output components
 - Creation of the component VHDL code
 - Integration into the Mico System Builder
 - Creation of a platform including the new component
- Use of custom components in software

- Creating a program using a component
- Notion of driver

System deployment

- Deployment of bitstreams in SPI flash
 - Using the JTAG port to program the flash
- Deployment of tested code in parallel flash
 - Creation of the flash programming infrastructure
 - Reconfiguration of the application for execution from flash
 - Deployment of an integrated application
 - Deployment of a complete uClinux image