HandsOn Training

Intel® FPGA Technical Training

Developing Software for the Nios II Processor

Course Description

This course provides all theoretical and practical know-how to create and manage software projects in the Nios II Software Build Tools for Eclipse.

The course combines 50% theory and 50% practical work on Terrasic DE series evaluation board.

The course starts with an overview of the Nios II processor, a summary of FPGA hardware design flow, and the Nios II software design process and tools.

The course then teaches how to access peripherals through the Nios II Hardware Abstraction Layer (HAL), its specific APIs and file system.

The course then describes in details the Nios II BSP features such as HAL system header file, linker scripts, boot sequence, assessing and reducing code size, and Nios II exceptions.

The second part of the course focuses on practical use of advanced debug features such as JTAG debug core, multi-processor systems, and how to measure code performance with performance counter and profiler.

The course ends with hardware acceleration using custom instructions and custom components, and using Direct Memory Access (DMA).

Course Duration

2 days

Goals

1. Become familiar with Intel Nios II processor, its capabilities and when to use it
2. Understand SoC design hardware and software flow from specification to programming and final verification on the board
3. Develop software for Nios II processor
4. Accessing peripheral through Hardware Abstraction Layer (HAL)
5. Develop BSP for the Nios II processor
6. Debug simple and complex Nios II systems, including multi-processor systems
7. Use custom components and custom instructions with the Nios II design
8. Use DMA with the Nios II processor
9. Handle Interrupts and exceptions in Nios II processor
10. Measure code performance with performance counter and profiler

Intended Users

Software engineers who would like to develop application and BSP for the Nios II processor

Previous Knowledge

Embedded system software design

C/C++ and assembly programming

Course Material

1. Synthesizer and Place & Route: Quartus Prime
2. Nios II Embedded Design Suite
3. Terrasic Cyclone V GX Evaluation board
4. Course book (including labs)

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

Table of Contents

Day #1

- **Introduction to Nios II Processor Architecture and Design Tools**
  - What is the Nios II processor?
  - Nios II processor architectural summary
  - Nios II processor performance
  - Licensing of Nios II processor
  - Nios II classic vs Nios II Gen2

- **FPGA Hardware Design Flow Summarized**
  - FPGA system hardware design flow
  - Qsys system integration tool
  - .sopcinfo file
  - FPGA programming

- **Nios II Software Design Process**
  - Nios II processor design flow
  - Nios II software design process
  - Nios II Embedded Design Suite (EDS)
  - Nios II software build tools
  - Creating new software projects
  - Application and BSP projects
  - Application and BSP from template
  - Create BSP project by itself
  - Create application project by itself
  - Importing projects into Workspace
  - Creating a new source file in GUI
  - Importing source files to a project
  - Creating linked resources
  - Setting project properties
  - Application project properties
  - Setting compiler flags (App and BSP)
  - Add libraries to application project

- BSP project properties
- BSP editor
- Build properties for individual files
- Project directory structure
- .elf and system.h output files
- Program target hardware
- Running code on a target
- Run configurations window
- System ID peripheral check
- Nios II debugger
- Nios II debug perspective
- Debug windows
- Breakpoints
- Watchpoints
- Nios II instruction set simulator

❖ **LAB #1: Build Software and Run/Debug on Board**

- **Introduction to the Hardware Abstraction Layer (HAL)**
  - Nios II Hardware Abstraction Layer (HAL)
  - Software build tools project structure
  - Key features of the HAL
  - Runtime library

- **HAL Specific API**
  - System clock
  - Using system clock timer
  - Alarms
  - Alarm code example
  - Timestamp timers
  - High resolution timestamp code example

- **Accessing Peripherals from Nios II**
  - Data cache usage
  - Reading/writing hardware in Nios II
  - Altera-provided HAL types
  - nios2-elf-gcc data widths
  - Header files for Nios II peripherals

- Nios II custom peripherals
- Example Nios II program

- **HAL File System**
  - HAL file system introduction
  - HAL file system structure
  - HAL file system API
  - Example applications
  - C support
  - Accessing device directly

❖ **LAB #2: Accessing Peripherals and Utilizing a Timer**

- **HAL System Header File**
  - Nios II software project key files
  - HAL system header file
  - System header file generation
  - Example system.h (BSP and components)
  - Example system.h (memories)
  - Making hardware changes to a component

- **Linker Script**
  - Linker script: linker.x
  - Linker section names
  - Entry and exception locations
  - View/edit linker sections and regions
  - Positioning variables and functions in memory
  - Create new sections and regions
  - Customizing linker script directly
  - Hardware variables to memories
  - Stack & Heap

Day #2

- **Boot Sequence**
  - Initialization file
  - Boot sequence for Nios II systems
  - Example alt_sys_init.c
  - Boot copier
  - Flash programmer overview
  - Booting FPGA from Flash
  - Loading RAM and running code
  - Customizing boot sequence
  - Hosted vs free-standing applications
  - Modifying boot loader
  - BSP editor: drivers tab

- **Assessing & Reducing Code Size**
  - Command shell options to determining code size
  - Determining code size from .objdump file
  - Common options to reduce code footprint
  - Advanced BSP options to reduce code footprint
  - Use alt_* stdio routines with Lightweight API
  - Lightweight driver API restrictions
  - Example results
  - General recommendation

- **Nios II Exceptions**
  - Nios II exceptions introduction
  - Interrupt controller options
  - External interrupt controller priority
  - Enhanced HAL interrupt API
  - Recommendations for ISRs
  - HAL API for ISRs
  - Interrupt routine: ISR declaration code example
  - Main program code example
  - Improving interrupt response

  - ❖ **LAB #3: Create an Interrupt Routine and Assess and Reduce Code Size**

- **JTAG Dbug Core**
  - JTAG Debug core introduction
  - JTAG Debug core block diagram
  - Enabling the JTAG Debug core and appropriate debug features in Qsys
  - Eclipse debugger communication with target

- **Advanced NIOS II Debug Features**
  - Debug from any entry point
  - Multi-processor systems
  - Software for multi-processor systems
  - Projects for multi-processor systems

- **Measuring Code Performance**
  - Performance counter
  - GNU profiler
  - Performance counter unit in Qsys
  - Time and event counters
  - Performance counter macros
  - Performance counter use
  - Code example usage
  - Performance counter report
  - Generating raw data with GNU profiler
  - Examining GPROF profiling results in Eclipse
  - Profiler data output: flat profile
  - Profiler data output: call graph
  - View call hierarchy
  - Command shell commands to analyze gmon.out
  - Other debugging options

❖ **LAB #4: Performance Counter and Profiler Tools**

- **Custom Instructions & Custom Peripherals**
  - Using custom hardware to accelerate algorithms
  - CPU processing bottlenecks identification
  - Accelerate bottleneck functions in hardware
  - Nios II custom instructions
  - Custom instruction features

- C language software interface
- Custom peripherals

- **DMA**
  - Direct Memory Access
  - DMA attached to system interconnect
  - Typical DMA transaction
  - Benefits of using DMA in Altera FPGAs
  - Accelerating software case study: CRC algorithm
  - HAL DMA model
  - DMA command prototypes
  - Posting a transmit request code example
  - Posting a receive request code example
  - Manipulating DMA transmit and receive channels
  - Thoughts on data cache

- **Additional Topics**
  - Creating HAL compliant device drivers
  - Nios II RTOS support
  - Nios II EDS software components

❖ **LAB #5: Use DMA Controller to Move Bulk Data & Perform Custom Bit-swap Instruction**