HandsOn Training

Intel® FPGA Technical Training

Designing with NIOS II Processor using Qsys System Integration Tool

Course Description

This course will teach you how to quickly build Nios II soft processor designs for Intel FPGAs using Intel's Qsys system-level integration tool.

You will become proficient with Qsys and expand your knowledge of the Quartus Prime FPGA design software.

Since Qsys makes design reuse easy through standard interfaces, we will examine the Intel Avalon-Memory Mapped and Streaming Interfaces as well as introduce the AMBA AXI interface standard from ARM. System Console debugging tool is also covered in details.

The course teaches the Nios II architecture and its memory, peripherals, how to manage SoC system, how to configure system based on Nios II, how to transfer data through the Bus system and internal interconnect, how to connect external memories, how to build a system with Qsys, how to handle interrupts, how to develop software and ways for debugging.

The course continuous with a focus on appending custom instruction and custom components to enhance performance, use of simulation models (BFMs), and creating SoC test-benches.

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, 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).

The class provides a significant hands-on component, where you will gain exposure to tool usage as well as system and custom HDL component design.

Course Duration

5 days

Goals

1. Build digital systems in the Qsys tool
2. Integrate the files generated by Qsys into the Quartus Prime design flow
3. Become familiar with Intel Nios II processor, its capabilities and when to use it
4. Understand SoC design hardware and software flow from specification to programming and final verification on the board
5. Create custom components with Avalon-MM and Avalon-ST interfaces and integrate them into your system
6. Test custom components and entire systems with the Avalon Verification Suite in the Modelsim-Altera
7. Perform in-system control & debugging with System Console
8. Configure the SoC system (clocks, PLLs, Resets, cache, TCM, on-chip memory, off-chip memory, MMU/MPU, Peripherals)
9. Use Bus Functional Models (BFMs) to simulate SoC behavior
10. Use custom components and custom instructions with the Nios II design
11. Use DMA with the Nios II processor
12. Handle Interrupts and exceptions in Nios II processor
13. Measure code performance with performance counter and profiler
14. Develop software for Nios II processor
15. Accessing peripheral through Hardware Abstraction Layer (HAL)
16. Develop BSP for the Nios II processor
17. Debug simple and complex Nios II systems, including multi-processor systems
18. Optimize systems to maximize performance and help close timing
19. Customize components using Tcl
20. Exploit Qsys hierarchical capability to add flexibility & scalability to your design

Intended Users

FPGA engineers who would like to use Qsys to build complex systems on chip

Previous Knowledge

Intel FPGAs architecture

Quartus Prime software

SignalTap II Embedded Logic Analyzer

C/C++

ModelSim

VHDL/Verilog

Course Material

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

Table of Contents

Day #1

❖ Introduction to the Qsys System Integration Tool

- **What is Qsys?**
  - Traditional system design
  - Automatic interconnect generation
  - Qsys benefits
  - Target Qsys applications
  - Qsys vs SOPC Builder
  - SOPC Builder systems in Qsys

- **Qsys UI**
  - Component library
  - System contents
  - System inspector
  - Address map
  - Clock settings
  - Project settings
  - Generation
  - HDL example
  - Messages
  - Other useful Qsys commands

- **Using Qsys in FPGA Design Flow**
  - FPGA hardware design flow
  - Additional Qsys verification support
  - Qsys system generation

- **Qsys Files**
  - Qsys source files
  - Qsys output files

❖ **Lab #1: Introduction to Qsys Design Flow**

- **Introduction to Qsys Interconnect**
  - Qsys interconnect architecture
  - Qsys supported standard interfaces (Avalon, AXI)
  - Qsys interconnect implementation
  - Arbitration priority
  - Enables simultaneous multi-mastering
  - Qsys memory-mapped packet format
  - Packetized interconnect vs latency
  - NoC architecture
  - Master network interface
  - Slave network interface
  - Pipelining

- **Using Qsys IP**
  - Qsys standard interfaces for IP
    - Clock
    - Reset
    - Avalon-ST
    - Avalon-MM
    - Avalon-C
    - Avalon-TC
    - AXI

  - Qsys IP
    - Component library parameter editors
    - Basic components
    - Streaming components
    - Memory components
    - Tristate components
    - Bridge components
    - High-speed interface components
    - Processor components

❖ **Lab #2: Using Qsys IP**

Day #2

❖ Building Nios II Processor-Based Systems

- **Nios II Processor-Based Systems**
  - What is System On a Programmable Chip (SOPC)?
  - What is Nios II processor?
  - Typical system architecture
  - Nios II processor major features
  - Requirements for Nios II processor designs
  - Nios II processor block diagram
  - Licensing
  - Qsys automatic interconnect generation
  - Add and configure Nios II CPU in Qsys
  - Nios II Gen 2 processor changes and improvements
  - Nios II processor versions and performance comparison
  - Vector tab settings
  - Caches and memory interfaces tab
  - Nios II Gen 2 uncached peripheral region feature
  - Tightly coupled memory
  - Arithmetic instructions tab
  - Nios II multiplier/divider/shifter performance
  - MMU and MPU settings tab
  - JTAG debug tab
  - Advanced features tab
  - Internal interrupt controller
  - Vectored interrupt controller (VIC)
  - Shadow registers
  - Using bridges with a system
  - Migrate from Nios II classic to Nios II Gen2 processor

❖ **Lab #3: Creating a Nios II Processor System Using Qsys**

- **RTL Simulation with the Nios II Processor**
  - RTL simulation for a Qsys system
  - Enable testbench generation
  - Qsys testbench output files
  - Building for an RTL simulation
  - Running an RTL simulation
  - "msim_setup.tcl" simulation script
  - Component simulation options

❖ Custom Components & Processor Interfaces

- **Creating Custom Components**
  - What are custom components?
  - Custom components interconnect
  - Example custom components
  - Component editor GUI
  - "hw.tcl" file
  - Custom component integration into Qsys system
  - Edit component parameters

- **Processor Interfaces**
  - Interfacing FPGA with external processors
  - Memory mapped interface
  - High-speed serial interfaces
  - Good use of SPI/Avalon master bridge
  - Using the NIOS II processor
  - Using the SoC devices

❖ **Lab #4: Build a Data Path Hardware System in Qsys that Incorporates Custom Logic**

❖ **Lab #5: Add Control Plane to the Design and Test the System**

Day #3

❖ System Console, Verification, and Hierarchy with Qsys

- **System Integration with Qsys**
  - Avalon verification suite
  - Application Programming Interface (API)
  - Avalon verification suite testbench
  - Clock source BFM
  - Reset source BFM
  - Avalon-MM BFMs & API
  - Avalon-ST BFMs & API
  - Avalon-MM/ST monitors, testbenches and API
  - AXI verification IP
  - Component verification system
  - System verification

❖ **Lab #6: Testing Custom Components with BFMs**

❖ **Lab #7: Testing the System with BFMs**

- **System Verification with System Console**
  - What is System Console?
  - Usage examples
  - System Console interfaces
  - System Console GUI launch
  - Command line interface
  - Interactive usage tips

- **System Console services**
  - System Console usage flow
  - Qsys components for System Console
  - Linking sevice instances with Quartus II project
  - Linking Quartus project using Tcl script
  - JTAG debug commands
  - master service type
  - monitor service type
  - bytestream service type
  - Dashboards

❖ **Lab #8: Testing the System with System Console**

- **Utilizing Hierarchy in Qsys Designs**
  - Hierarchy in the Quartus II software
  - Hierarchy in Qsys
  - Scalable Qsys systems with hierarchy
  - Adding subsystems to component library
  - Add subsystems to system
  - Connecting systems together
  - Ways to create subsystems
  - Making changes to a subsystem
  - Bridging systems together with Avalon-MM pipeline bridges
  - Subsystem masters
  - Subsystem clocking
  - Subsystem parameterization
  - Tcl commands used in instance script

❖ **Lab #9: Qsys Hierarchy: Expand System to support four parallel video processing channels**

❖ **Lab #10: Qsys Hierarchy: Run four-channel system under Nios II software control**

Day #4

- **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 #11: 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 #12: 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

- **Nios II Processor Custom Instructions**
  - What are custom instructions?
  - Block diagram of a system with custom instructions
  - Custom instructions support in Nios II development tools
  - Create custom instructions in Qsys
  - Add custom instruction to system
  - C language software interface
  - Assembly language interface
  - Why custom instructions?
  - Floating point custom instructions
  - Floating point CI macros
  - Nios II custom instruction user guide
  - Custom instruction vs. custom component
  - Multi-cycle custom instructions
  - Accelerating CRC example

❖ **LAB #13: Custom Components and Custom Instructions**

- **Application Examples for the Nios II Processor**
  - Component tradeoff
  - State machine replacement
  - Custom microcontroller
  - General purpose system controller
  - I/O processing controller
  - Off-loading existing CPU
  - Build multi-processor systems
  - Multi-CPU architectures
  - Shared memory, mutex, and mailboxes
  - Software for multi-processor systems
  - Launch group for multi-processor systems
  - Sending interrupts
  - Control plane and data plane

❖ **LAB #14: Build and Explore a Multi-Nios II CPU System**

Day #5

- **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 #15: Create an Interrupt Routine and Assess and Reduce Code Size**

- **JTAG Debug 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 #16: Performance Counter and Profiler Tools**

- **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

❖ **LAB #17: Use DMA Controller to Move Bulk Data & Perform Custom Bitswap Instruction**