ASIC / FPGA tasarım süreçlerinde kullanılan , VHDL, Verilog HDL ve SystemVerilog eğitimleri istenen zorluk seviyesinde hazırlanan standart ya da ihtiyaçlara göre özelleştirilmiş şekilde sunulmaktadır. Mevcut standart eğitimlerimizi aşağıda bulabilirsiniz.

Temel sayısal tasarım kavramlarıyla entegre edilerek, FPGA tasarım ve doğrulama süreçlerinin laboratuvar uygulamaları eşliğinde anlatıldığı eğitimleri inceleyebilirsiniz.

Tüm bu konularda daha fazla bilgi için lütfen bizimle temasa geçiniz.

SystemVerilog For Verification (4 Days)

Education Duration : 4 Days

Education Language : English

Tranier : Nigel Woolaway

Prerequsite : -

Target Audionce : Experienced Verification Engineers

Content :

This 4-day course is aimed at experienced Verification engineers who wish to learn about verification with SystemVerilog. The course stresses a methodology for implementing these features in your verification environment.

This course is taught for all the leading simulators although not all simulators will support every feature immediately. The course is a consistent mix of lecture and lab-exercises. Targeted quizzes and labs are designed to reinforce the course material.

SystemVerilog Assertions (1 Day)

Education Duration : 1 Day

Education Language : English

Tranier : Nigel Woolaway

Prerequsite : -

Target Audionce : Experienced Verification Engineers

Content :

This one day course is targeted at Design and Verification engineers who wish to deploy Assertion based Verification within their next project. Assertion Based Verification is becoming a cornerstone of good design and verification practice. SystemVerilog is one of the first languages to feature a 100% native temporal assertion syntax, making it extremely well integrated with the language. Our course stresses a methodical approach to learning and developing good coding style.
This course, which is taught for all the leading simulators is a consistant mix of lecture and lab-exercises. Targetted quizzes and labs are designed to reinforce the course material. Although the

content of this class overlaps the final day of our SystemVerilog for Design and SystemVerilog for Verification courses, both SVA and our course are applicable to Verilog projects with no other SystemVerilog content.

Introduction to Universal Verification Methodology (UVM) (4 Days)

Education Duration : 4 Days

Education Language : English

Tranier : Nigel Woolaway

Prerequsite : -

Target Audionce : System Verilog Verification Engineers

Content :

This 4-day course is for engineers interested in developing SystemVerilog verification environments using the latest Universal Verification Methodology (UVM).
Students will first learn:

Basic testbench structure

How to model communication at the transaction level (TLM)

How to write analysis components such as Scoreboards and Coverage Collectors

Strategies for connecting to RTL designs

After mastering the basics, students will learn best-practice techniques to maximize the reusability of their test environments. Topics include:

Using the UVM factory

Managing complexity using hierarchy and factory overrides

Making reusable testbenches

Developing test cases using UVM sequences

Using UVM Registers

Introduction to VHDL For RTL Design (4 Days)

Education Duration : 4 Days

Education Language : English

Tranier : Nigel Woolaway

Prerequsite : -

Target Audionce : VHDL Tasarıma yeni başlayanlar

Content :

A 4-day course teaching designers to write efficient, accurate RTL code for synthesis as well as basic testbenching and verification techniques.

This course is intended for designers who are new to VHDL. It focuses on teaching good RTL coding style for synthesis but also discusses basic testbenching and verification techniques.
This course continuously mixes lecture and exercise. There is a simulation exercise for most topics providing a very hands-on experience.

Advanced VHDL (3 Days)

Education Duration : 3 Days

Education Language : English

Tranier: Nigel Woolaway

Prerequsite : "Introduction to VHDL for RTL Design"

Target Audionce : Experienced VHDL Designers

Content :

A 3 days course emphasizing behavioral techniques, testbench strategies and design management.
The 3-day Advanced VHDL class is aimed at experienced VHDL users who wish to take their use of the language to a higher level. The course is a consistant mix of lecture and lab-exercises.
A pre-requisite for this course is the Introduction to VHDL course or equivalent experience.

Introduction to Perl

This 2-day class will introduce the student to the Perl programming language. Upon completion of this class, the student will be able to write useful Perl programs to automate operating system tasks and perform sophisticated text manipulation. Students will also have a brief introduction to Perl's GUI capabilities through Perl/Tk. The format of the class is mixed lecture/lab, with lab exercises immediately following each major topic. The lab exercises are intended to reinforce the preceding lecture topic(s), and are designed to be directly applicable in an EDA context.

Topics Covered

- What is Perl?
- Parts of a Perl program
- Creating a Perl program
- Scalar Data
- Numeric and String Literals
- Variables
- Evaluation and Assignments
- Operators
- Managing Data in Files

- Standard file descriptors
- Opening files
- Reading and writing files
- Formatting
- File handling functions
- Program Flow Control
- Statement blocks
- 'if' and 'unless' statement
- 'while' and 'until' loops
- 'for' loops
- Using 'foreach'
- Lists and Arrays
- Using lists
- Creating and using arrays
- List and array functions
- Slices
- More Program Flow Control
- last, next, redo
- Using labels
- The 'goto' statement

Hashes

- Creating and using hashes
- Hash functions
- Hash slices
- Regular Expressions
- Using regular expressions
- Character matching
- Anchors and quantifiers
- Using backreferences
- split and join
- User-defined Functions
- Creating a Function

- Pass argument by value call
- Pass argument by reference call
- Private variables and scoping
- Special Variables
- Record handling
- Formats
- Regular expressions
- Perl in the Unix environment
- Process information
- File and Directory manipulation
- Process and time commands
- DBM Databases
- Opening and closing databases
- Using a DBM database
- Perl Modules
- Installing modules
- Graphical Interface
- Creating windows, widgets and methods
- Sockets

After completing this comprehensive training, you will have the necessary skills to:

- Write well-crafted, reusable Perl script to automate EDA-related tasks
- Explain and apply complex regular expressions to parse input
- Use Perl data structures to store and search for data
- Automate Linux tasks and manipulate files
- Create a basic graphical user interface in Tk for your Perl script

Who Should Attend

Engineers who want to use Perl to increase productivity by automating tasks in support of EDA tools.

Prerequisites

A working knowledge of the Linux operating system

Basic programming experience in C or Linux shells is recommended

TCL and TK (2 Days)

This 2-day class will introduce the student to the Tcl programming language and to the GUI capabilities of the Tk toolkit. Upon completion of this class, the student will be able to write useful Tcl programs to automate operating system tasks and add scripting capabilities to C programs.

Students will also be introduced to Tcl's GU capabilities through Tk toolkit.

The format of the class is mixed lecture/lab, with lab exercises immediately following each major topic. The lab exercises are intended to reinforce the preceding lecture topic(s), and are designed to be directly applicable in an EDA context.

Topics Covered

- TCL uses and comparison to other languages
- tclsh and wish interactive shells
- Command syntax
- Data types
- Evaluating expressions
- Control flow commands
- String processing
- Lists
- File access
- TCL Arrays
- Procedures
- TK Basics
- Arranging widgets with pack and grid
- TK events and binding
- Basic TK widgets
- Menus
- Scrollbars
- Listboxes
- The text widget
- The canvas widget
- Combining TCL and C

After completing this comprehensive training, you will have the necessary skills to:

- Describe the important differences between Tcl and keyword-based languages such as C or Perl

- Write well-crafted, reusable Tcl scripts to automate tasks
- Apply complex regular expressions to parse input and generate code
- Use the Tk library to add a graphical user interface around your script
- Extend Tcl by adding new commands backed by custom C code

Who Should Attend

Anyone wishing to use the power of TCL/TK to develop custom scripts and user interfaces for EDA related tasks.

Prerequisites

A working knowledge of the Linux operating system

Basic knowledge of a programming language is an advantage

Introduction to Digital Design (1 Day)

This one day course provides an overview of the digital design process for both FPGAs and ASICs.

Topics Covered

- Introduction
- History of digital design
- ASIC vs FPGA
- The digital design flow
- Stages of a typical digital design flow
- Specifications
- The importance of a good specification
- Linking the specification to implementation
- Linking the specification to verification
- Device I/O
- I/O standards
- Signal integrity
- Design entry
- Design languages
- Schematics
- Intellectual property (IP)
- Functional verification
- Verification languages

- Simulation
- Assertions
- Property checking
- Clock, reset and power domains
- Synthesis
- The synthesis process
- Timing constraints
- Test
- Preparing for test
- ATPG methodologies
- Layout
- Floorplanning
- Special nets
- Physical synthesis
- Static timing analysis
- Timing constraints
- Timing corners
- Physical verification
- DRC and LVS
- Parasitic extraction
- Production
- FEOL and BEOL
- Lithography limitations
- Production test
- Failure analysis
- Failure mechanisms
- Random vs systematic
- Identifying failure locations

This course is designed to be very interactive and students are encouraged to ask questions on all aspects of the digital design process. Because of the high-level view of the process there are no lab exercises and the course is also very modular to enable customisation of the contents to satisfy specific requirements. Detailed training courses are available for all of the topics presented.

Who Should Attend

Engineers and managers new to digital design and in need of an overview of the entire digital design process from specification through design languages, functional verification, synthesis, test pattern generation, layout, timing analysis, physical verification, production and failure analysis.

Prerequisites

A keen interest in the digital design flow

Introduction to Verilog (3 Days)

A 3-day course teaching designers to write efficient, accurate RTL code for synthesis as well as basic testbenching and verification techniques.

This 3-day course is intended for designers who are new to Verilog and who wish to become familiar with the language with a particular emphasis on writing RTL code for synthesis. We also cover how to construct testbenches for unit level verification of your RTL code.

This course continuously mixes lecture and exercise. There is a simulation exercise for most topics providing a very hands-on experience. Synthesizable constructs are clearly identified and appropriate synthesis coding techniques discussed.

Topics Covered

- Verilog modeling
- Using your Simulator
- Verilog basics
- Procedural assignments
- Design a sequential pipe
- Synthesizing your design
- Operators
- Programming statements
- Sensitivity lists
- Continuous assignments
- Primitives
- Tasks
- Functions
- Timing accuracy
- Verification using Verilog
- Bi-directionals
- Synthesis issues

• Finite State machines (exercise)

The course is a consistent mix of lecture and lab-exercises. Targeted quizzes and labs are designed to reinforce the course material.

Who Should Attend

Design engineers wishing to adopt the Verilog language for digital design and basic functional verification.

Prerequisites

Basic knowledge of digital design is essential, experience with a programming language is beneficial.

System Verilog for Design (2 Days)

A 2 day course with an optional 1 day introduction to Verilog for Design for designers unfamiliar with the Verilog language.

The course is aimed at RTL designers who wish to learn about the new features of SystemVerilog for RTL design including the use of concurrent assertions for verification of functional behaviour

Attendees will learn the new synthesisable constructs and features available in SystemVerilog.

Synthesis tools are not required.

Main Topics Covered

• Data Types

• Tasks and Functions

• Arrays and Structures

• Reducing RTL Ambiguity

• RTL Programming

• Hierarchy

• More Synthesis Constructs

• Interfaces

• System Verilog Assertions

Optional introductory topics

• Structure

• Data types

• Modules

• Hierarchy

• Procedural block

• Procedural Assignments

• if..else & case

• Continuous Assignments

• Tasks & Functions

• Finite State Machines

The course is a consistent mix of lecture and lab-exercises. Targeted quizzes and labs are designed to reinforce the course material.

Who Should Attend

Designers wishing to adopt the synthesisable aspects of System Verilog.

Prerequisites

RTL design experience is essential

Building Interfaces with Arria 10 High-Speed Transceivers (1 Day)

Course Description

In this course, you will learn how you can build high-speed, gigabit interfaces using the 20-nm embedded transceivers found in Arria® 10 FPGA families. You will be introduced to the transceiver architecture and how the transceivers are configured to support various high-speed protocols. You will learn how to optimize and debug both the digital and analog sections of your transceiver design. You will gain an understanding of the transceiver reconfiguration interface that you can use dynamically adjust transceiver settings to add flexibility to your transceiver design. Lastly, you will learn how to create application and control logic that effectively manages Arria 10 transceiver resources.

At Course Completion

You will be able to:

Implement high-speed serial protocols in Arria® 20-nm embedded transceivers

Simulate transceiver operation using a generated simulation setup script file

Improve transceiver usage and avoid transceiver design issues by applying an understanding of device architecture to design situations

Optimize analog settings to improve link behavior using Altera tools

Employ transceiver reconfiguration to dynamically change transceiver behavior in-system

Skills Required

Familiarity with FPGA/CPLD design flow

Familiarity with the Quartus Prime Pro design software

Familiarity with FPGA architecture

Familiarity with high-speed interfaces and transmission protocols is helpful, but not required

Prerequisites

We recommend completing the following courses:

SignalTap II Logic Analyzer: Introduction & Getting Started

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

The Quartus Software Debug Tools

OpenCL™ on FPGAs for Parallel Software Programmers (1 Day)

Course Description

This course will teach you how to accelerate algorithms on FPGAs using the OpenCL™ framework. In this class, we will cover the FPGA technologies that make it an ideal coprocessor to boost performance. We will discuss how to use the Intel® FPGA SDK for OpenCL to synthesize OpenCL constructs into custom logic to easily leverage the advantages of FPGA accelerated computing. We will go over the constructs of the OpenCL standard & the Intel FPGA flow that automatically converts kernel C code into hardware that interacts with the host. In the hands-on labs, you'll write OpenCL programs targeting FPGAs. *OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of Khronos

At Course Completion

You will be able to:

Describe the FPGA technologies that can be used to accelerate algorithms

Understand the advantages of using the Intel FPGA OpenCL solution

Write OpenCL host and kernel code to accelerate algorithms on FPGAs

Compile, debug, and run OpenCL programs using the Intel FPGA SDK for OpenCL

Skills Required

Basic understanding of a parallel computing platform such as CUDA* or OpenCL *Other names and brands may be claimed as the property of others

Designing with DSP Builder for Intel® FPGAs (1 Day)

Course Description

Learn the timing-driven Simulink design flow for implementing high-speed DSP designs. This course is focused on the hands-on development of highly-optimized DSP algorithms using the advanced blockset capability of the latest DSP Builder—an interface between the Intel® Quartus® Prime software and Mathworks MATLAB & Simulink tools. You will analyze and design your DSP algorithm using the DSP Builder for Intel FPGAs MATLAB and Simulink. You will learn how to easily explore

architecture and performance tradeoffs with system-level constraints. You will also verify the functionality and the performance of the generated hardware in the Intel Quartus Prime software.

At Course Completion

You will be able to:

Implement DSP algorithms using DSP Builder for Intel FPGAs

Incorporate IP and Primitive cores in a design

Explore design architecture and performance tradeoffs using system-level constraints

Incorporate a DSP Builder model into a Platform Designer system

Verify the hardware performance and implementation in the Intel Quartus Prime software

Skills Required

Familiarity with DSP fundamentals and design

Familiarity with Intel® Quartus Prime software is helpful, but not necessary

Familiarity with Mathworks MATLAB and Simulink is helpful, but not necessary

Familiarity with digital modem design is helpful, but not necessary

Introduction to High-Level Synthesis with Intel® FPGAs (1 Day)

Course Description

In the class, you will learn how to use the Intel® HLS Compiler to synthesize, optimize, and verify design components for Intel FPGAs. We will first discuss the benefits of HLS then talk about features of the Intel HLS Compiler. You will learn how to use the compiler options, the generated reports, and the final generated files to integrate the IP within an Intel Quartus® project. Lastly you will learn how to effectively optimize your IP using the generated reports.

At Course Completion

You will be able to:

Use the Intel HLS Compiler to synthesize an Intel Quartus®-compatible component

View reports to debug & optimize the component

Co-simulate your HLS component using an RTL simulator with a software testbench

Integrate the HLS-generated component within an FPGA design

Understand the various interfaces available & be able to select the optimal one for various types of components

Effectively use various data types & math support features

Understand how the compiler pipelines loops

Skills Required

Basic understanding of the C programming language

Basic understanding of FPGAs and the Intel Quartus Development Environment

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Pro Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

High-Level Synthesis Advanced Optimization Techniques (1 Day)

Course Description

In the class, you will learn how to use advanced techniques using the Intel® HLS Compiler to create an optimized IP for Intel® FPGAs. We will cover using recommended techniques to improve loop pipelining performance. We will discuss how the Intel HLS compiler generates and optimizes local memory architecture as well as how to best guide the compiler to create never-stall local memories. Lastly, we will use several real-life design examples to demonstrate the optimization flow.

At Course Completion

You will be able to:

Use the Intel HLS Compiler generated HTML reports to locate performance bottlenecks in a component

Effectively pipeline loops by removing data and memory dependencies

Use Pragmas to control HLS loop performance

Optimize Local Memory Architecture

Use all the optimization tool available in the Intel HLS Compiler to create a high-performance FPGA IP

Skills Required

Basic understanding of the C programming language

Basic usage with the Intel HLS Compiler; specifically attendance of the "Introduction to High-Level Synthesis with Intel FPGAs" course

Prerequisites

We recommend completing the following courses:

Introduction to High-Level Synthesis with Intel® FPGAs

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Pro Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

Enabling FPGA Accelerators Using the Acceleration Stack for Intel® Xeon® CPU with FPGAs (1/2 Day)

Course Description

The Acceleration Stack for Intel® Xeon® CPU with FPGAs is a robust collection of software, firmware, tools and hardware intended to make it easier to develop and deploy Intel FPGAs for workload optimization in data center and cloud environments. In this training, we will discuss and practice how software developers can write host code that can communicate with the FPGA accelerator transparently using the Open Programmable Acceleration Engine (OPAE) and walkthrough how FPGA and accelerator developers can build, test and integrate their Accelerator Functional Units (AFUs) into the FPGA.

At Course Completion

You will be able to:

Understand the components of the Acceleration Stack and how to use them

Write software applications using the OPAE layer to run acceleration workloads on an FPGA accelerator

How OpenCL abstracts away the acceleration stack to build heterogenous accelerator systems

Understand the basics of creating an AFU workload for the FPGA using a variety of programming models

Know how to get started using the Acceleration Stack for workloads like Machine Learning

Skills Required

Basic understanding of FPGAs and/or software development

Optimizing OpenCL™ for Intel® FPGAs (16 Hours Course) (2 Days)

Course Description

This course covers optimization techniques to implement a high performance OpenCL™ solution on FPGAs. We'll use various debug & analysis tools available in the Intel® FPGA SDK for OpenCL to boost performance of OpenCL kernels. The first half of the course focuses on the optimization of single work-item kernels & the utilization of Intel FPGA channels & OpenCL pipes. The second half of the course focuses on the optimization of ND range kernels & the effective utilization of various memory systems implemented by the Intel FPGA kernel compiler. Throughout the course we will discuss good OpenCL coding design practices for FPGAs & use Intel FPGA SDK for OpenCL features to improve OpenCL performance on FPGAs. *OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of Khronos

At Course Completion

You will be able to:

Use debugging & optimization tools

Execute multiple OpenCL kernels in a task parallel fashion

Boost performance of single work-item kernels

Use single work-item kernels to implement parallel programming algorithms

Use Intel FPGA channels or OpenCL pipes to increase communication performance

Boost performance of NDRange kernels

Improve usage of memory architectures

Improve host-device communication efficiency

Use good OpenCL coding practices

Boost data processing efficiency

Skills Required

Attendance of the Introduction to OpenCL for Intel FPGAs or a good understanding of the OpenCL standards

Prerequisites

We recommend completing the following courses:

Introduction to OpenCL™ for Intel® FPGAs

Introduction to Parallel Computing with OpenCL™ on FPGAs

Running OpenCL™ on Intel® FPGAs

Writing OpenCL™ Programs for Intel® FPGAs

Developing a Custom OpenCL™ BSP (1 Day)

Course Description

Using the Intel® FPGA SDK for OpenCL, software developers can quickly and efficiently use OpenCL to generate custom hardware for an FPGA. Leveraging this capability for your custom FPGA board requires building an Intel FPGA SDK for OpenCL-compatible custom Board Support Package (BSP).

This training will cover the requirements of a custom BSP. You will learn the steps necessary to convert an Arria® 10 reference platform BSP to the hardware & software deliverables needed for an Arria 10-based custom BSP.

At Course Completion

You will be able to:

Identify reference platform contents: design files, script files, Quartus® project files

Know the custom-BSP hardware and software deliverables

Understand OpenCL BSP-specific Qsys IP components

Setup and verify OpenCL BSP development environment

Know the steps necessary to customize the Arria 10 reference platform

Skills Required

FPGA design knowledge: architecture (including clocking, global routing, I/Os), high-speed design, timing analysis, Qsys design, floorplanning using LogicLock regions, Tcl scripting

Familiarity with the following interfaces: Avalon® (both streaming and memory-mapped), PCIe, external memory (DDR3 or DDR4)

Basic OpenCL coding knowledge

Basic C coding knowledge

Prerequisites

We recommend completing the following courses:

Introduction to OpenCL™ for Intel® FPGAs

Introduction to OpenCL™ for Intel® FPGAs (1 Day)

Course Description

OpenCL™ is a standard for writing parallel programs for heterogeneous systems. With the Intel® FPGA SDK for OpenCL, OpenCL constructs are synthesized into custom logic for optimal acceleration on FPGA devices. This course introduces the basic concepts of parallel computing. It covers the constructs of the OpenCL standard and the Intel FPGA flow that automatically converts kernel C code into hardware that interacts with the host. In hands-on labs, you'll write programs to run in emulation mode as well as on an FPGA board. *OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of Khronos

At Course Completion

You will be able to:

Describe high-level parallel computing concepts and challenges

Understand the advantages of using the Intel FPGA OpenCL solution

Know the basics of the OpenCL standard

Write simple programs in OpenCL

Compile, debug, and run OpenCL programs using the Intel FPGA SDK for OpenCL

Skills Required

Basic understanding of the C programming language

Partial Reconfiguration with Intel® FPGAs (1 Day)

Course Description

One advantage of an FPGA is the ability to change its function through reconfiguration. This normally replaces the entire FPGA design. What if you could reconfigure just part of the overall design, replacing blocks with different functionality while the main design is still running? In this class, you'll learn how to implement Partial Reconfiguration (PR) in an Intel® Arria® 10 or Intel Stratix® 10 FPGA. You'll know how to change the functionality of a portion of the device, while the rest operates without interruption. Explore the benefits and limitations of PR, understand design guidelines, and learn the steps to enable this feature. Through lab exercises, you will prepare a design for PR using the Intel Quartus® Prime Pro software, complete the design, and test the feature on a board.

At Course Completion

You will be able to:

Understand the Partial Reconfiguration (PR) design flow

Prepare a design for PR

Design or instantiate a PR host

Know about the available IP for use with PR

Generate required PR programming files through design compilation

Debug a PR design

Understand the limitations of PR

Skills Required

Completion of "The Intel® Quartus® Prime Software: Foundation" course OR a working knowledge of the Intel Quartus Prime software

Knowledge of a Hardware Description Language (Verilog or VHDL)

Prerequisites

We recommend completing the following courses:

Design Optimization Using Quartus II Incremental Compilation (Legacy Course)

Introduction to Verilog HDL

Introduction to VHDL

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Advanced Qsys System Integration Tool Methodologies (1 Day)

Course Description

In this class, you will learn advanced features of Altera's Qsys system level integration tool in the Quartus® Prime design software v. 15.1. You will learn how to simulate Qsys systems in ModelSim-Altera Edition using Avalon bus functional simulation models (BFMs), exercise, monitor, and debug systems with System Console, and build hierarchical Qsys systems. You will also learn how to further customize your components through Tcl scripting. Optimization techniques for improving performance and closing timing are also discussed. The class provides a significant hands-on component, where you will gain exposure to tool usage as well as system design.

At Course Completion

You will be able to:

Test custom components and entire systems with the Avalon Verification Suite in the ModelSim-Altera Edition simulation tool

Perform in-system control & debugging with System Console

Optimize systems to maximize performance and help close timing

Customize components using Tcl

Exploit Qsys' hierarchical capability to add flexibility & scalability to your design

Skills Required

Background in digital logic design

Working knowledge of the Quartus design software

Knowledge of HDL coding methodology (helpful but not mandatory)

Completion of the "Introduction to the Qsys System Integration Tool" class

Prerequisites

We recommend completing the following courses:

Introduction to the Platform Designer System Integration Tool

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Using the Intel® Quartus® Prime Standard Edition Software: An Introduction

Related Courses

Below are the related courses you may be interested in:

Migrating to the Quartus Prime Pro Edition Software

Introduction to the Platform Designer System Integration Tool (1 Day)

Course Description

This class will teach you how to quickly build designs for Intel® FPGA devices using the Platform Designer system-level integration tool (formerly known as Qsys), part of the Intel Quartus® Prime software. You will become proficient with using Platform Designer and learn how to quickly integrate "off-the-shelf" IP and custom logic into a system. Platform Designer makes design reuse easy through the use of standard interfaces, so you will learn about the interfaces supported by the tool: Avalon® Memory Mapped and Streaming as well as an introduction to the Arm* AMBA* AXI interface standard. 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.

At Course Completion

You will be able to:

Build digital systems in the Platform Designer tool

Integrate the files generated by Platform Designer into the Intel Quartus Prime software design flow

Create custom components with Avalon-MM and Avalon-ST interfaces and integrate them into your system

Skills Required

Background in digital logic design

Working knowledge of the Intel Quartus Prime software

Knowledge of HDL coding methodology (helpful but not mandatory)

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Using the Intel® Quartus® Prime Standard Edition Software: An Introduction

Related Courses

Below are the related courses you may be interested in:

Migrating to the Quartus Prime Pro Edition Software

Intel® Quartus® Prime Software: Pro Edition Features for High-End Designs (1 Day)

Course Description

In this course, you will learn about new features in the Pro Edition software to help you plan, implement, and close timing when using Intel Arria® 10 and Stratix® 10 FPGAs. You will learn how to migrate a design to the Intel Quartus Prime Pro Edition software from the standard edition, or plan a

new design from scratch using the Interface Planner. You will learn how to take advantage of new compilation flow stages and incremental optimization for faster design compiles. You will learn how to use block-based design to ease team-based design work and to take advantage of partial reconfiguration. New features for timing analysis will be covered including the clock domain crossing (CDC) viewer. A scripted flow for the tools will be emphasized throughout the course, including the labs.

At Course Completion

You will be able to:

Plan a design using the Interface Planner & Logic Lock regions

Use incremental optimization stages such as post-place physical synthesis & post-route fix-up to increase performance

Describe the steps needed to implement partial reconfiguration

Understand the Pro Edition software timing reports

Evaluate the sufficiency of clock crossing logic using the CDC Viewer

Understand command-line & TCL scripting interfaces available in the Pro Edition software & learn to use them

Skills Required

The Quartus Prime Software: Foundation class or familiarity with the Intel Quartus Prime software

Familiarity with Verilog or VHDL synthesizable design structures

Familiarity with timing constraint concepts

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Pro Edition) (Online Training)

Related Courses

Below are the related courses you may be interested in:

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training) (1 Day)

Course Description

You will learn how to use the Intel Quartus Prime Pro Edition software to develop an FPGA design from initial design to device programming. You will create a new project, input new or existing design files, and compile your project. You'll learn how to search for compilation information, use settings and assignments to adjust the results of compilation, and manage I/O-related assignments using the Pin Planner and the Interface Planner. You will also learn about device programming files and how to

program an FPGA device on your board. You will learn techniques to help you plan your design. You will employ Intel Quartus Prime features that can help you achieve design goals faster. You will also learn how to plan and manage I/O assignments for your target device.

At Course Completion

You will be able to:

Make pre-project decisions to prepare for an Intel Quartus Prime design

Create, manage, and compile Intel Quartus Prime projects

Use Intel Quartus Prime tools to view the results of compilation

Review compilation results in various Intel Quartus Prime software reports and graphical viewers

Plan and manage device I/O assignments using Pin Planner and Interface Planner

Understand device programming files and their use with the Intel Quartus Prime Programmer

Skills Required

Background in digital logic design

Ability to describe a hardware system using VHDL, Verilog, or EDA schematic tool

Follow-on Courses

Upon completing this course, we recommend the following courses (in no particular order):

Partial Reconfiguration with Arria 10 FPGAs

Related Courses

Below are the related courses you may be interested in:

The Quartus Prime Software: Foundation (Pro Edition) (Online Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

The Intel® Quartus® Prime Software: Foundation for Xilinx* Vivado* Design Suite Users (1 Day)

Course Description

You will learn how to use the Intel® Quartus® Prime Pro Edition software & correlate these steps to the Xilinx* Vivado* Design Suite to develop an FPGA design from initial design to device programming. You'll create a new project, input new or existing design files, & compile your project. Learn how to search for compilation information, use settings and assignments to adjust the results of compilation, & manage I/O-related assignments using the Pin Planner and the Interface Planner. You will learn about device programming files & how to program an FPGA on your board. You will learn techniques to help you plan your design. You will employ the Intel Quartus Prime software features to help you achieve design goals faster. You'll learn to plan & manage I/O assignments for your target device.

At Course Completion

You will be able to:

Identify Intel Quartus Prime software features to replace Xilinx Vivado Design Suite features

Make pre-project decisions to prepare for an Intel Quartus Prime design

Create, manage & compile Intel Quartus Prime projects

Use Intel Quartus Prime tools to view results of compilation

Review compilation results in various Intel Quartus Prime software reports & graphical viewers

Plan & manage device I/O assignments using Pin Planner & Interface Planner

Understand device programming files

Skills Required

Background in digital logic design

Ability to describe a hardware system using VHDL, Verilog, or EDA schematic tool

Related Courses

Below are the related courses you may be interested in:

The Quartus Prime Software: Foundation (Pro Edition) (Online Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Using Intel® SoC FPGAs (1 Day)

Course Description

This class will teach you how to design with Intel® FPGA SoC FPGAs using the Intel Quartus® Prime software and how to develop software for these devices. It will cover the Hard Processor System (HPS) architecture for Intel Stratix® 10, Arria® 10, Cyclone® V, and Arria V devices, including an overview of the Arm* Cortex*-A53 and A-9. You will learn to add and configure the processor component in a Platform Designer system. You will then learn to implement and configure the first-stage and second-stage bootloaders (based on U-Boot), and how to build and boot to the Linux* OS. You will learn the boot stages of each SoC FPGA family. At the completion of the course, you will have the knowledge necessary to immediately start using Intel SoC FPGA devices in your own designs or on development kits

At Course Completion

You will be able to:

Create, manage, and compile an SoC based FPGA using the Platform Designer tool

Compile the first

and second-stage bootloaders

Build a Linux* OS boot image

Use the SoC with a Linux* OS boot image

Skills Required

FPGA knowledge is helpful, but not required

Basic knowledge of using the command line in the Linux* OS is helpful, but not required

Prerequisites

We recommend completing the following courses:

Introduction to the Platform Designer System Integration Tool

Follow-on Courses

Upon completing this course, we recommend the following courses (in no particular order):

Developing Software for an ARM-based SoC

Related Courses

Below are the related courses you may be interested in:

SoC Hardware Overview: System Management, Debug, and General Purpose Peripherals

SoC Hardware Overview: the Microprocessor Unit

Advanced Optimization with Intel® Stratix® 10 FPGA Hyperflex™ Architecture (1 Day)

Course Description

Are you targeting an Intel® Stratix® 10 device & want to learn how your design can reach maximum core performance?

In this course, you will learn design techniques to unleash the full potential of the Intel Stratix 10 FPGA Hyperflex architecture using Hyper-Optimization. Learn how to use the Intel Quartus® Prime Pro Edition software to identify logic structures limiting retiming and thus design performance. Learn to modify your coding style & logic structures &, as a result, allow your design to achieve clock rates of up to 2 times when compared to a non-optimized design, without changing overall design functionality.

Note: While the focus of this course is the Intel Stratix 10 device family, many techniques can be used to improve performance in other FPGA devices.

At Course Completion

You will be able to:

Learn to interpret complex retiming reports to locate & understand critical chains, design paths requiring further optimization for improved performance

Learn Hyper-Optimization techniques to restructure design logic to take advantage of the Intel Stratix 10 Hyperflex architecture (or any FPGA architecture) using techniques such as 1) Unrolling loops, 2) Pre-computation to reduce loop size, 3) Shannon's Decomposition, 4) Time-domain multiplexing retiming, 5) Hyper-Folding, 6) Loop pipelining

Skills Required

Familiarity with FPGA/CPLD design flow

Familiarity with the Intel Quartus Prime Pro Edition design software

Familiarity with Verilog or VHDL synthesizable design structures

Completion of the “Performance Optimization with Intel Stratix 10 FPGA Hyperflex Architecture” course

Prerequisites

We recommend completing the following courses:

Performance Optimization with Intel® Stratix® 10 FPGA Hyperflex™ Architecture

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

Timing Closure with the Quartus II Software

The Intel® Hyperflex™ FPGA Optimization Workshop (1 Day)

Course Description

This workshop builds on the tools and techniques learned in the Performance Optimization with Stratix® 10 Hyperflex Architecture and Advanced Optimization with Stratix 10 Hyperflex Architecture training courses. This workshop provides an environment where an FPGA designer can sharpen their optimization skills by exercising those techniques on provided design blocks, while proctored by an Intel FPGA instructor. Note: this class is 90% lab time with only 10% lecture.

At Course Completion

You will be able to:

Use Fast Forward Compile reports to influence optimization decisions on designs targeting the Intel Hyperflex architecture

Improve clocking speed by implementing Intel Hyperflex architecture design recommendations on FPGA logic

Skills Required

Good HDL coding skills

Familiarity with the Intel® Stratix® 10 FPGA Hyperflex™ architecture

Completion of the "Performance Optimization with Intel Stratix 10 FPGA Hyperflex Architecture" and "Advanced Optimization with Intel Stratix 10 FPGA Hyperflex Architecture" instructor-led training courses or all equivalent online training courses (see curricula)

Prerequisites

We recommend completing the following courses:

Advanced Optimization with Intel® Stratix® 10 FPGA Hyperflex™ Architecture

Performance Optimization with Intel® Stratix® 10 FPGA Hyperflex™ Architecture

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

Timing Closure with the Quartus II Software

Performance Optimization with Intel® Stratix® 10 FPGA Hyperflex™ Architecture (1 Day)

Course Description

In the Performance Optimization with Intel® Stratix® 10 FPGA Hyperflex™ Architecture course, you will learn Intel Quartus® Prime Pro software features and some basic design techniques that will enable your designs to take advantage of the Intel Stratix 10 FPGA Hyperflex architecture. In the training, you will learn two steps to improving your performance with the Intel Hyperflex architecture, namely Hyper-Retiming and Hyper-Pipelining, with each step allowing you to move your design up the performance curve.

Note: While the focus of this course is the Intel Stratix 10 device family, many of the techniques you will learn can be used to improve performance in other device architectures.

At Course Completion

You will be able to:

Describe the Intel Stratix 10 device architecture

Enable the Intel Quartus Prime Pro software features that take advantage of the Intel Hyperflex architecture

Evaluate possible design improvements using the Intel Quartus Prime Pro software's Fast Forward Compile feature

Improve your Intel Stratix 10 FPGA design performance by understanding and enabling Hyper-Retiming

Improve your Intel Stratix 10 FPGA design performance by implementing zero-latency Hyper-Pipelining

Skills Required

Familiarity with FPGA/CPLD design flow

Familiarity with the Intel Quartus Prime Pro design software

Familiarity with Verilog or VHDL synthesizable design structures

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

Timing Closure with the Quartus II Software

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer (1 Day)

Course Description

You will learn how to constrain & analyze a design for timing using the timing analyzer in the Intel® Quartus® Prime software v. 17.1. This includes understanding FPGA timing parameters, writing Synopsys* Design Constraint (SDC) files, generating various timing reports in the timing analyzer & applying this knowledge to an FPGA design. Besides learning the basic requirements to ensure that your design meets timing, you will see how the timing analyzer makes it easy to create timing constraints to help you meet those requirements.

At Course Completion

You will be able to:

Understand the timing analyzer timing analysis design flow

Apply basic and complex timing constraints to an FPGA design

Analyze an FPGA design for timing using the timing analyzer

Write and manipulate SDC files for analysis and controlling the Intel Quartus Prime compilation

Skills Required

Completion of "The Intel Quartus Prime Software: Foundation" online or instructor-led course OR a working knowledge of the Intel Quartus software

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Advanced Timing Analysis with TimeQuest (1 Day)

Course Description

Using the Quartus® II software version 15.0 and building upon your basic understanding of creating Synopsys Design Constraint (SDC) timing constraints, this class will guide you towards understanding, in more depth, timing exceptions. You will learn how to apply timing constraints to more advanced

interfaces such as source synchronous single-data rate (SDR), double-data rate (DDR) and LVDS, as well as clock and data feedback systems. You will discover how to write timing constraints directly into an SDC file rather than using the GUI and then enhance the constraint file using TCL constructs. You will also perform timing analysis through the use of TCL scripts.

At Course Completion

You will be able to:

Write Tcl script files to automate constraining and analysis of FPGA designs

Apply timing exceptions to real design situations

Properly constrain and analyze the following design situations: source synchronous interfaces, external feedback designs, and high-speed interfaces containing dedicated SERDES hardware

Skills Required

Experience with PCs and the Windows operating system

Completion of "The Quartus II Software Design Series: Timing Analysis" course OR a working knowledge of the TimeQuest timing analyzer and basic SDC commands

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

Timing Closure with the Quartus II Software (1 Day)

Course Description

One of the greatest and most frustrating FPGA design challenges is closing timing. It is common to find, after performing a timing analysis on an FPGA design, that one or more timing reports show a timing failure. How can this be corrected? The answer is not always obvious. This class teaches techniques used by Design Specialists to close timing on designs that “push the envelope” of performance. Examples include thoroughly analyzing the design for timing failures, adjusting settings and assignments according to tool recommendations, selecting correct clock resources and writing HDL code for optimal performance. For Stratix 10, consider these courses instead: Performance Optimization with Stratix 10 HyperFlex Architecture, Advanced Optimization with Stratix 10 HyperFlex Architecture

At Course Completion

You will be able to:

Employ best practices for closing timing on an FPGA design in the Quartus II software

Analyze a TimeQuest-generated timing report as a starting point for timing closure

Use the tools available in the Quartus II software to help in meeting timing

Choose settings/assignments to get the best performance

Identify the most common types of timing failures and how to solve them

Skills Required

Experience with PCs and the Windows operating system

Completion of "The Quartus II Software Design Series: Foundation" course OR a working knowledge of the Quartus II software

Completion of “The Quartus II Sotware Design Series: Timing Analysis” course OR a working knowledge of Synopsys Design Constraints (SDC) and the TimeQuest timing analyzer

Prerequisites

We recommend completing the following courses:

The Intel® Quartus® Prime Software Design Series: Timing Analysis with Timing Analyzer

The Intel® Quartus® Prime Software: Foundation (Instructor-led / Virtual Training)

The Quartus Prime Software: Foundation (Standard Edition) (Online Training)

Timing Analyzer: Introduction to Timing Analysis

Advanced Verilog HDL Design Techniques (1 Day)

Course Description

You will learn efficient coding techniques for writing synthesizable Verilog for programmable logic devices (FPGAs and CPLDs). While the concepts presented mainly target Altera® FPGA devices using the Quartus® II software, many can be applied to other devices and synthesis tools as well. You will gain experience in behavioral and structural coding while learning how to effectively write common logic functions including registers, memory, and arithmetic functions. You will learn how to use Verilog constructs to parameterize your design, increasing their flexibility and reusability. You will be introduced to testbenches and Verilog constructs used when building them. The exercises will use the Quartus II software to synthesize Verilog code and the ModelSim®-Altera tool for simulation.

At Course Completion

You will be able to:

Implement synthesizable sequential and combinatorial RTL code

Design finite state machines using multiple encoding schemes

Develop simple testbenches for verification

Use tools in the Quartus II software to synthesize code and verify results

Run functional simulations in the ModelSim-Altera software

Skills Required

Completion of the "Introduction to Verilog HDL" course or some prior knowledge and use of Verilog hardware description language (HDL)

Background in digital logic design

Understanding of synthesis and simulation processes

Prerequisites

We recommend completing the following courses:

Introduction to Verilog HDL

Verilog HDL Basics

Introduction to Verilog HDL (1 Day)

Course Description

This class is a general introduction to the Verilog language and its use in programmable logic design, covering the basic constructs used in both the simulation and synthesis environments. By the end of this course, you will have a basic understanding of the Verilog module, data types, operators and assignment statements needed to begin creating your own designs, using both behavioral and structural approaches. In the hands-on laboratory sessions, you will get to practice the knowledge you have gained by writing simple but practical designs. You will check your designs by compiling in the Quartus® II software version 10.1 and simulating in the ModelSim®-Altera® tool.

At Course Completion

You will be able to:

Create a basic Verilog module

Understand the difference between simulation and synthesis environments

Understand Verilog data types and operators and their uses

Model hardware and test using behavioral modeling constructs

Model hardware and test using structural modeling constructs

Skills Required

Background in digital logic design

Knowledge of simulation is a plus

Prior knowledge of a programming language (e.g., "C" language) is a plus

No prior knowledge of Verilog HDL or the Quartus II software is needed

Advanced VHDL Design Techniques (1 Day)

Course Description

You will learn & practice efficient coding techniques for writing synthesizable VHDL for programmable logic devices (FPGAs & CPLDs). While the concepts presented will mainly target Altera® FPGA devices using the Quartus® II software, many can be applied to other devices & synthesis tools. You will gain experience writing behavioral & structural code & learn to effectively code common logic functions including registers, memory, & arithmetic functions. You will use VHDL constructs to parameterize your designs to increase their flexibility & reusability. You will also be introduced to testbenches, VHDL constructs used to build them, & common ways to write them. The exercises will use the Quartus II software version to process VHDL code & ModelSim®-Altera software for simulation.

At Course Completion

You will be able to:

Develop coding styles for efficient synthesis when:

Targeting device features

Inferring logic functions

Using arithmetic operators

Writing state machines

Use Quartus II software RTL Viewer to verify correct synthesis results

Incorporate Altera structural blocks in VHDL designs

Write simple testbenches for verification

Create parameterized designs

Skills Required

Completion of the "Introduction to VHDL" course or some prior knowledge and use of VHDL

Background in digital logic design

Understanding of synthesis and simulation processes

Prerequisites

We recommend completing the following courses:

Introduction to VHDL

VHDL Basics

Introduction to VHDL (1 Day)

Course Description

This one-day class is a general introduction to the VHDL language and its use in programmable logic design, covering constructs used in both the simulation and synthesis environments. By the end of this course, you will have a basic understanding of VHDL so that you can begin creating your own designs, using both behavioral and structural approaches. In the hands-on laboratory sessions, you will get to practice the knowledge you have gained by writing simple but practical designs. You will check your designs by compiling in the Quartus® II software v. 13.1 and simulating in the ModelSim®-Altera® tool.

At Course Completion

You will be able to:

Implement basic VHDL constructs

Use VHDL design units: entity, architecture, configuration and package

Create behavioral and structural models in VHDL

Skills Required

Background in digital logic design

Knowledge of simulation is a plus

Prior knowledge of a programming language (e.g., "C" language) is helpful, but not required

No prior knowledge of VHDL or Quartus II software is needed