DIZAIN-SYNC

TRAINING GUIDE

dizain-sync

Dizain-Sync is an independent consultantcy organisation in electronic design & development, with more than 30 years experience.
With our knowledge we offer a unique perspective on the combination of PLM, EDA, Design and Training services.

Dizain-Sync's mission is to maximize customers productivity by improving Design Methodology, Design Flow, Design and Designers.

PART OF
dieco electronics

Overview training courses

4

HDL courses
- Professional VHDL
- Advanced VHDL
- Haskell

7

System Verilog courses
- Introduction to Verilog
- System Verilog for Verification
- System Verilog Assertions

10

UVM courses
- Introduction to Universal Verification Methodology
- Open verification Methodology

12

SystemC courses
- SystemC Modeling with introduction to TLM 2.0
- Introduction to SystemC Verification
- Advanced SystemC Verification

18

Support languages
- Introduction to C++
- Introduction to TCL/TK
- Python

20

LTspice courses
- LTspice Basic
- LTspice Advanced

Professional VHDL

Duration
3 Days

Trainer
Bert Molenkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

Introduction

In this 3-day course you will learn how to write efficient VHDL for modelling, implementation and verification of FPGA and digital ASIC designs. The training doesn't only cover the language defined in the IEEE 1076 standard, but also includes topics focusing on simulation and synthesis of your design.

Hands-on exercises during the training will help you practice what you've learned, optionally while using the simulation or synthesis tools of your choice. Experience shows that there is a pleasant interaction between the students and the trainer during the training. For many digital designers this course has proven to be a very good start for using VHDL in their daily work!

Prerequisites
A digital design background and preferably some programming experience in C or another language.

Audience
Designers of digital systems.

Subjects

VHDL Introduction

- VHDL, the history
- Properties of VHDL
- Alternative VHDL descriptions of the sr latch
- Test Bench
- VHDL Analysis, Elaboration, and Simulation

VHDL in more detail

- Data types
- Operators
- Overloading
- Subprograms
- Packages
- Analysis order
- Sequential statements
- Concurrent statements
- Modeling delay
- Generic descriptions
- Multiple driven signal
- Port map pitfalls
- Qualification

Synthesis of VHDL

- Synthesis Tools
- What is not supported
- What is supported
- Synchronous model
- Combinational model
- Latches
- Tri-states

Standard Libraries

Finite State Machines

Advanced VHDL

Duration
2 Days

Trainer
Bert Molenkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

Introduction

Increase your VHDL proficiency by learning advanced techniques that will help you write more robust and reusable code. This comprehensive course is intended for designers who already have some experience with VHDL.

This training is more than just the theory that is shown below. There is room to discuss your own case during the training. Besides that you can ask the trainer to explain more about a topic of your own choice.

Audience

The training is very suitable for those who have worked some time with VHDL. The training is less suitable for the beginning VHDL user. For them, the training Professional VHDL would be better.

Subjects

VHDL-1993/2002/2008 and advanced topics

- More regular syntax
- Shared variable
- Impure functions
- Component instantiation and configuration
- Operators added since VHDL'93
- Waveform
- Signal related attributes
- Postponed processes
- Guarded signals and disconnection

Property Specification Language (PSL)

Fixed point package (VHDL'93 style)

- Introduction Fixed Point
- Representation of fixed point numbers in VHDL
- Examples
- Available operators
- Rouding and saturation
- Pitfall

Floating point package (VHDL'93 style)

Final State Machine

Verilog

- History
- Global differences between Verilog HDL and VHDL
- Modeling styles
- Test environment
- Data types
- Operators

The Intel Quartus Prime Timing Analyzer

- Introduction
- Design
- Constrain the design
- Timing Analyzer
- Combinational Logic

Introduction to Haskell and Clash

Duration
3,5 Days

Trainer
Qbaylogic

Location
Borne, Nijmegen

SCAN AND SIGN UP FOR THE TRAINING

Introduction

Haskell is an advanced functional programming language that, in combination with the Haskell-to-VHDL/Verilog compiler called Clash, enables straightforward reasoning about the functionality and performance of your circuits. Haskell's clear semantics and concise syntax enable a design methodology where you can go from a high-level golden reference to a solution meeting the performance requirements using correct-by-construction transformations. For problems where such a route is less appropriate, Haskell offers powerful testing frameworks instead, based on constraint random testing and automatic shortest counterexample generation.

In this course you will learn how to effectively use Haskell to create high-level models and subsequently transform them into efficient FPGA or ASIC designs. The training includes many hands-on exercises to help you put into practice what you have learned. The course can have its focus adjusted to suit the experience of the audience.

Prerequisites
Either a digital design background with some programming experience, or an existing programming background.

Audience
Designers of digital systems.

Training overview

Haskell introduction

- Functional programming
- Polymorphism
- Algebraic data types
- Higher-order functions
- Type inference
- Interactive interpreter

Haskell for circuit design

- Sized data types
- Combinational circuits
- Synchronous circuits
- State machines
- Space vs Time trade-off
- VHDL/Verilog generation

Testing in Haskell

- Unit testing
- Property-based testing
- Constraint random testing
- Automated stimulus shrinking

System verilog Courses

Introduction to Verilog

Duration
3 Days

Trainer
Gert-Jan Tromp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

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.
We can offer this class with most popular simulators.

Introduction

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

These courses 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.

Subjects

- 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)

Prerequisites

A digital design background and preferably some programming experience in C or another language.

Audience

Digital 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.

System verilog Courses

SystemVerilog for Verification

Duration
4 Days

Trainer
Gert-Jan Tromp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

Introduction

This SystemVerilog (IEEE 1800) is a significant new language based on the widely used and industry standard Verilog hardware description language. The SystemVerilog extensions enhance Verilog in a number of areas, providing productivity improvements for RTL designers, verification engineers and for those involved in system design and architecture. The course stresses a methodology for implementing SystemVerilog in your verification environment. The course is a consistent mix of lecture and lab-exercises.

Prerequisites

Introduction to Verilog training course or equivalent experience.

Audience

Experienced Verification Engineers who wish to learn about verification with SystemVerilog.

Day 1 Introduction to Verification with SystemVerilog

- Language Enhancements
- SystemVerilog Data Types
- Arrays & Structures
- SV Scheduler
- Program Control
- Lab-Sparse memory
- Hierarchy
- Tasks & Functions
- Dynamic Processes
- Inter-process Sync & Communication
- Lab-Mailboxes

Day 2 Classes

- Class Basics
- Constructors
- Lab-OOP
- Virtual Methods
- Inheritance
- Parameterization
- Polymorphism
- Lab-Polymorphism

Day 3 Interfaces

- Lab-Virtual Interfaces
- Randomization & Constraints
- Randomize
- Constraints
- Random Sequences
- Lab-Randomization
- Functional Coverage
- Covergroups
- Coverpoints and Cross
- Lab-Covergroups

Day 4 SVA

- Concurrent Assertion Basics
- Lab-Assertion Basics
- Boolean Expressions
- Sequences
- Lab-Sequences
- Lab-Data Values
- Properties
- Verification Directives
- Lab-bind

System verilog Courses

SystemVerilog for Assertions

Duration
1 Day

Trainer
Gert-Jan Tromp or
Paul Eijkelkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

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

This 1-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.

Prerequisites
Students are expected to be already familiar with the Verilog language.

Subjects

SystemVerilog Assertions

- Immediate / Concurrent
- Severity system tasks
- SystemVerilog Event Scheduler
- Concurrent Assertions
- Boolean expressions
- System Functions

Sequence Blocks

- Sequence Operators
- Repetition [*N][*m:n]
- Non-Consecutive Repetition [=N][=N:M]
- Goto Repetition [->N] [->N:M]
- Value Change Functions
- Relating sequences
- Seq. expressions: and, or, intersect
- Sequence expressions
- throughout, within, .ended
- Sequence controls
- Data-use within a sequence

Property block

- Implication |-> |=>
- Sequential antecedents
- Multi-clock support
- matched

Verification directives

- Clock inference and specification
- Controlling Assertions
- Bind directive
- Reactive SV testbenches

UVM Courses

Universal Verification Methodology (UVM)

SCAN AND SIGN UP FOR THE TRAINING

Duration
3 Days

Trainer
Gert-Jan Tromp

Location
Borne, Nijmegen
Amsterdam

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

This 3-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

Prerequisites

SystemVerilog for Verification course or equivalent experience using SystemVerilog.

Audience

Verification & Design engineers.

Subjects

- Introduction to UVM
- Transaction-level Communication
  - TLM Interfaces, Channels, Port & Exports
- Basic Testbench Structure
  - Components
  - Phasing
  - Start and end of simulation
- Dynamic Construction - Introduction to the UVM Class Factory
- Connecting to the DUT
- Generating Reports and Messaging
- Modeling Transactions
- Analysis
  - UVM Analysis components
  - Scoreboards, coverage collectors, predictors
- Hierarchy
  - UVM Components and Hierarchy
  - Hierarchical API
- Creating a Configurable Test Environment
  - Factory Overrides
  - Resources, configurations
- Stimulus generation
  - Sequences
  - Scenarios (testing patterns)
- UVM registers
  - Register model development
  - Register model integration
  - Register model usage

UVM Courses

Open Verification Methodology (OVM)

Duration
3 Days

Trainer
Gert-Jan Tromp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

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.

Introduction

This 3-day course is for engineers interested in developing SystemVerilog verification environments using the Open Verification Methodology (OVM).

Subjects

- Introduction to OVM
- Transaction-level Communication
  - TLM Interfaces
  - TLM Channels
  - Port & Exports
- Basic Testbench Structure
  - Components
  - Threaded_component
  - Environment
  - Phases
- Dynamic Construction
  - Introduction to the OVM Class Factory
- Connecting to the DUT
  - Transactors
  - Virtual Interfaces
  - Working with BFMs
- Generating Reports and Messaging
- Modeling Transactions
- Adding Analysis Components
  - Analysis components
  - Scoreboards
  - Coverage Collectors
- Control Blocks
  - Hierarchy
  - OVM Component and Hierarchy
  - Hierarchical API
- Creating a Configurable Test Environment
  - Factory Overrides
  - Configuration
- Managing Test cases
  - Layered Stimulus (Scenarios)
  - Programmable Transaction Sequences
- Other OVM Classes
- Mixed Language Simulation
  - Using SystemC TLM models in a SystemVerilog Test Environment
  - Design Patterns

Prerequisites
SystemVerilog for Verification course or equivalent experience.

Audience
Verification & Design engineers.

SystemC courses

SystemC Introduction for Modeling with TLM 2.0

Duration
3 Days

Trainer
Gert-Jan Tromp or
Paul Eijkelkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

Introduction

A 3-day workshop for engineers who are new to SystemC or those who may be self-taught, with an interest in learning SystemC for modeling purposes.

Course Description

This 3-day workshop introduces the student to the SystemC C++ class library and the TLM 2.0 modeling standard. It is intended for engineers who are new to System Cor those who may be self-taught, with an interest in learning SystemC for modeling purposes.The student will learn how to write, compile, execute, and debug system and hardware descriptions with SystemC, and will receive thorough and in-depth coverage of the concepts of the Accellera/IEEETLM 2.0 modeling standard.This course is mixed lecture and exercises, with an exercise for nearly every topic.

Prerequisites

Introduction to C++ (2 days) training course
Course may be taken immediately before this course.

Audience

Hardware, software and systems engineers who have a good working knowledge of C++ and SystemC, and want to learn to use the OSCI TLM-2.0 standard.

Course Outline

Introduction to SystemC

- Core Library Basics
  - Modules
  - Communication (channels, ports,and exports)
  - Module Constructor (and exercise)
  - Simulationo Scheduler
  - Events and Event Queues
- Modeling Behavior
  - Method Processes (and exercise)
  - Thread Processes (and exercise)
  - Module Instantiation (in module) (and exercise)
  - Simulation Initialization
- Core Library Elements
  - SystemC Data Types
  - Primitive Channels
- User defined channels (and exercise)
  - Custom Constructors

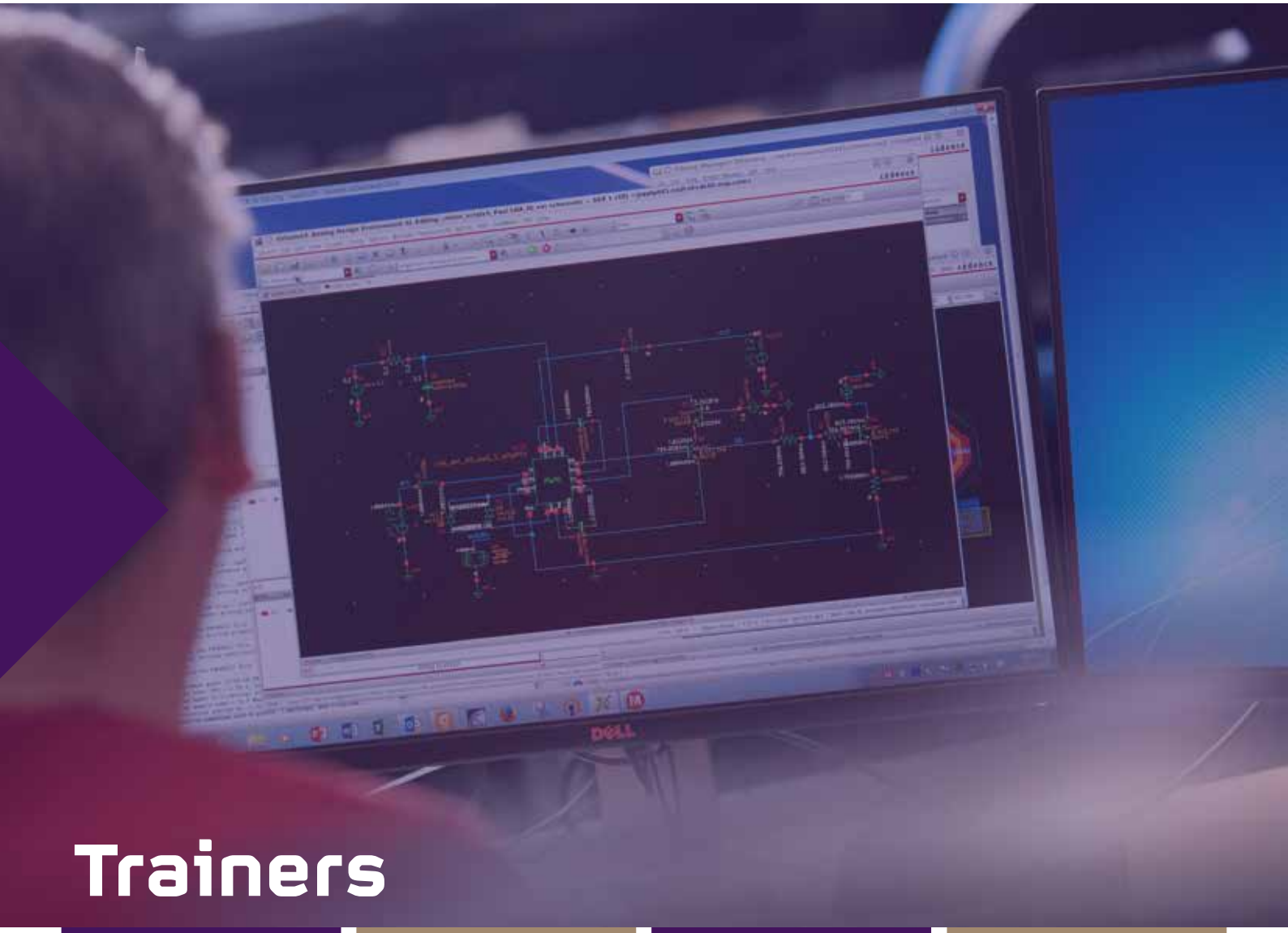
Exports
Dynamic Processes (and exercise)

Introduction to the IEEETLM 2.0 Standard

- TLM 2.0 Overview
  - Interface functions
  - Socketso Generic payload
  - Protocol
- Interfaces
  - Transport
  - DMI
  - Debug
- Sockets
  - Initiator and Target
  - Socket Binding
  - Hierarchy, Multi-connect
  - Topology Examples
- Generic Payload Overview
  - Attributes
- LT Coding style (and exercise)
  - Transport Interface
  - Temporal Decoupling
  - Quantum Keeper
- AT Coding Style (and exercise)
  - Protocol PhasesoForward, Backward, and Return Paths
  - Base Protocol (2-phase)
  - Payload Event Queue (PEQ)
- DMI interface DMI Hint
  - DMI Data Structure
  - Invalidating DMI
- Debug Interface (and exercise)
  - Debug Transport Interface
- Convenience Sockets (and exercise)
  - Simple Sockets
  - Tagged Sockets
  - Multi-pass through Sockets
- Generic Payload In-dept
  - Byte Enable
  - Streaming
  - Endianness
  - Memory Management

Generic Payload Extensions (and exercise)

- Base Protocol In-depth
  - 4-state and Variants



Trainers

Gert-Jan Tromp

Gert-Jan Tromp is an expert in electronic design and verification, embedded software, design tools development, technical writing and digital systems testing. He has more than 20 years of experience as a trainer and consultant for Dizain-Sync.

Bert Molenkamp

Bert Molenkamp is a teacher of the faculty Electrical Engineering, Mathematics and Computer Science at the University of Twente. His field of research is Digital System Design, especially focussing on the use of VHDL in the design process of a digital system, and synthesis aspects of VHDL. He is a VHDL trainer at Dizain-Sync B.V. since 1989.

Paul Eijkelkamp

Paul Eijkelkamp is a consultant of Dizain-Sync. His expertise lies in the field of simulation models, hardware description languages like SystemC, VHDL and Verilog, programming languages and setting up EDA environments. He has more than twenty years of experience in providing training.

Jaap Fijnvandraat

Jaap Fijnvandraat is an expert in modelling and simulation in several disciplines. In Signify he was responsible for the international roll-out and support of the Design Analysis environment. As such he has developed and given a lot of trainings, especially in the area of Electronic Design simulation. In July 2018 Jaap started his own company, FEMS Consultancy, offering his expertise on modelling and simulation.

Location overnight Hotel van der Valk Hengelo

Location Nijmegen

Location of the course

The training location is at our office in Borne. Lunch is also served here. The overnight stays, lunch and dinner are in the Hotel van der Valk. In case you have multiple people to be trained, we can also come to you for an on-site training, saving travel cost and optimizing the time for the trainees. Please contact us for the possibilities.

Location Amsterdam

Foto: Kireyonok Yuliya

SystemC courses

Introduction to SystemC for Verification

Duration
3 Days

Trainer
Paul Eijkelkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

It is intended for engineers who are new to SystemC or those who may be self-taught, with an interest in learning SystemC for verification purposes. The student will learn how to write, compile, execute and debug system and hardware descriptions and testbenches with SystemC. This course is mixed lecture and exercises, with an exercise for nearly evaery topic.

Introduction

This 3-day workshop introduces the student to the SystemC C++ class library and to the SystemC Verification library.

Subjects

- Introduction
- SystemC modeling
- Basic modeling structure
- Getting started - running & debugging
- Modules
  - Channels, ports, interfaces
  - Module constructor
  - Events
  - Processes in general
  - Thread processes
  - Method processes
  - Module instantiation (in modules)
- sc_main
- SystemC data types
- Primitive channels
- SystemC Verification
  - Data Introspection
  - Randomization
  - Constraints
  - Callbacks
  - Sparse arrays
  - Customizing Data Generation

Prerequisites

Introduction to C++ (1 day) training course.

Hands-On Labs

A good portion of class time will be spent applying principles learned in lecture to hands-on labs.

Audience

Desig engineers, system engineers and software engineers.

SystemC courses

Advanced SystemC for Verification

Duration
3 Days

Trainer
Paul Eijkelkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

This 3-day workshop focuses on verification and test bench techniques using SystemC and the SystemC Verification Library.

Introduction

This 3-day workshop is intended for engineers who are familiar with SystemC with an interest in using SystemC for Advanced Verification.

Subjects

- Verification concepts
- Testbench structure
- Testbench strategies
- Stimulus generation
  - Transaction-based stimulus
  - Constrained randomization of data
  - Randomizing control flow
  - Dynamic construction with parameters
  - Using dynamic processes for stimulus
  - Transactors
- Mixed-language simulation (tool specific)
- Verification
  - Shadow models
  - Monitors
  - Assertions
  - Checkers
- Analysis
  - Intro to C++ Standard Template Library - Applying STL: Scoreboards
  - Transaction-level logging / viewing
  - Coverage
    - Code / line coverage (tool specific)
    - Functional coverage (tool specific)
  - Reactivity
  - Closing the loop: feedback to stimulus

Prerequisites

Introduction to SystemC for Verification training course (or equivalent experience).

Hands-On Labs

A good proportion of class time will be spent on practical lab exercises.

Audience

Desig engineers, system engineers and software engineers.

SystemC courses

Introduction to C++

Duration
1 day, 2 days
or 3 days

Trainer
Paul Eijkelkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

We offer 1-day, 2-day and 3-day versions of the course to meet the needs of the student with different levels of C++ experience.
The following subjects are for the 3-day course. The 1-and 2-day versions will skip certain subjects depending on the need.

Introduction

Our C++ courses are intended to introduce (or refresh) engineers who will be using C++ for design, modeling or verification purposes such as using the SystemC class library or the SystemC Verification library.

Subjects

- Introduction
- Getting Started
- Preprocessor and Libraries
- Program Structure
- Basic Language Elements
- More Data Types
- I/O
- Pointers & References
- Classes - Modeling “objects”
- Constants
- Function overloading
- Initialization & cleanup
- Operator Overloading
- Templates
- Template specialization
- Useful utilities in the Standard Template Library [STL] (+ exercise)
- Inheritance

Prerequisites

A digital design background and preferably some programming experience in C or another language.

Hands-On Labs

A good portion of class time will be spent applying principles learned in lecture to hands-on labs.

Support languages

Introduction to Python

Duration
3 Days

Trainer
Jokin Segundo Barbarro

Location
Borne, Nijmegen

SCAN AND SIGN UP FOR THE TRAINING

Introduction

A 3-day all-hands-on-deck workshop to get acquainted with the basics (and not so basics) of the Python scripting language, with a clear focus on learning by doing, and making scripts that help automate everyday tasks.

Course description:

The objective of the course is to study the foundational aspects of Python programming with the focus set on script writing, data manipulation, and program organization. After completion, students should be able to start writing useful Python programs on their own and explore the multitude of applications of Python that might be of their interest ("throwaway" scripting, lab automation, system administration, data science, scientific programming, or web frameworks, just to name a few.)

Prerequisites

This course intends to present the fundamentals of Python to people that already have at least basic programming experience. No specific programming language knowledge is required, but a basic idea of programming concepts (e.g., conditionals and loops), and operating system use (working with files, directories, and the terminal) are assumed.

Audience

Engineers, scientists, and sysadmins that want to learn the basics of data processing and scripting with Python to later apply it to their field of expertise.

Course outline:

Setup of Python.

- Introduction to Python: scripts, basic data types and flow control.
- Data structures, containers, and collections.
- Organizing your program: functions, exceptions, modules.
- Classes and Object-Oriented Programming (OOP) in Python.
- Generators and other advanced topics.
- Testing, debugging, and packaging code.

Support Languages

Introduction to Tcl/Tk

Duration
2 Days

Trainer
Paul Eijkelkamp

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

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.

Introduction

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 GUI capabilities through Tk toolkit.

Prerequisites

Student will need to have a working knowledge of UNIX operating system. Also, basic programming experience in C or UNIX shells is highly recommended

Hands-On Labs

A good proportion of class time will be spent on practical lab exercises.

Audience

Desig engineers, system engineers and software engineers.

Subjects

- Day 1
  - Introduction
  - Getting Started
  - TCL Basics
  - TCL Commands
  - String processing
  - Lists
  - TCL I/O
  - TCL Arrays
  - Procedures
  - TCL in the Unix environment
- Day 2
  - TK Basics
  - Arranging Widgets with Pack
  - Arranging widgets with Grid
  - Tk Events and Binding
  - Basic TK Widgets
  - Menus
  - Scrollbars
  - Listboxes
  - Text widget
  - Canvas widget
  - Combining TCL and C

LTspice courses

LTspice Basic

SCAN AND SIGN UP FOR THE TRAINING

Duration
1 Day

Trainer
Jaap Fijnvandraat

Location
Borne, Nijmegen
Amsterdam

Introduction

In this 1-day training workshop you will learn about the special features of LTspice for setting up simulations and performing analysis results. It is a hands-on training. Each subject starts with a short introduction after which guided exercises help to understand and apply the new features. Even for those already familiar with LTspice this course has proven to reveal new features that make the use of LTspice yet more efficient.

Training overview

LTspice Introduction
- Strength of LTspice
- Positioning of LTspice

Schematic entry
- Basic elements and actions

Performing an analysis
- DC, AC, transient analysis
- Measurements
- Sweeping parameters

Hierarchical models
- Using a schematic model versus a library model
- Special LTspice elements

Information resources

Prerequisites
Some experience with Spice-like circuit simulation.

Audience
Electrical engineers, wanting to exploit the strength of LTspice for simulating their designs.

LTspice courses

LTspice Advanced

Duration
2 Days

Trainer
Jaap Fijnvandraat

Location
Borne, Nijmegen
Amsterdam

SCAN AND SIGN UP FOR THE TRAINING

Introduction

In this 2-day training workshop you will learn about the advanced use of LTspice for special purposes, especially for modelling (sub-)circuits. Part of the workshop is using LTspice functions for modelling a Pulse Width Modulator to drive a Half Bridge.

Prerequisites

Having the knowledge of the LTspice Basic course (having experience with measurements, sweeping parameters, hierarchical models). Regular user of LTspice with at least half a year of real experience.

Audience

Electrical engineers who want to learn more in-depth the capabilities of LTspice in modelling sub-circuits and in performing simulations for parameter studies.

Also more advanced use of parameters – e.g. for statistics – is demonstrated. The use of Initial Conditions for (re)starting simulations is treated as well. Furthermore some special techniques (FFTs and Averaged Modelling) are shown. The course will help one to build a behavioural model of (own-designed) sub-circuits or (bought-in) ASICs.

Training overview

Tracking of parameters

- Sensitivity analysis
- Design of experiments

Statistical simulations

- Parameter distributions
- Noise in the time domain

Advanced Behavioural Modelling

- Mathematical functions
- Differentiation and integration

Averaged modelling

Performing a Fast Fourier Transformation
Initial Conditions and Restarting

- ICs on nodal voltages
- ICs on state elements

SHAPING THE FUTURE

Contact

New - Private course

Would you prefer some extra personal attention? Then you can book a private lesson or a private course. You do not have to wait for full classes and you can start whenever you want in consultation with the trainer.

- The lessons are one-on-one and so you get all the attention.
- The lessons are provided on your wishes and level.
- You can bring your own case.

Contact us to discuss dates and prices

Tailor made training

Beside our standard training program we can also provide customer specific training. For example training on your design flow, data management, simulation or build automation.

More information

Would you like more information about these courses please contact US or visit our website

training@dizain-sync.com

www.dizain-sync.com

Dizain-Sync B.V
Oostermaat 2
7623 CS Borne
The Netherlands
+31 (0)74 2650 050
info@dizain-sync.com