Designing with Verilog and SystemVerilog

FPGA 1

LANG-VSV-ILT (v1.0) **Course Specification**

Course Description

This comprehensive course is a thorough introduction to the Verilog language. The emphasis is on writing Register Transfer Level (RTL) and behavioral source code. This class addresses targeting Xilinx devices specifically and FPGA devices in general. The information gained can be applied to any digital design by using a top-down synthesis design approach. This course combines insightful lectures with practical lab exercises to reinforce key concepts. You will also learn advanced coding techniques that will increase your overall Verilog proficiency and enhance your FPGA optimization. This course covers Verilog 1995 and 2001.

In this five-day course, you will gain valuable hands-on experience. Incoming students with little or no Verilog knowledge will finish this course empowered with the ability to write efficient hardware designs and perform high-level HDL simulations.

This comprehensive course also thorough introduction to SystemVerilog constructs for synthesis.

Level – FPGA 1

Course Duration – 5 days

Price – $4000 or 40 Xilinx training Credits

Course Part Number – LANG-VSV-ILT

Who Should Attend? – Engineers who want to use Verilog effectively for modeling, design, and synthesis of digital designs. This course also provides an introduction to SystemVerilog (an extension of the Verilog standard) for design.

Prerequisites

- Basic digital design knowledge

Follow-up to LANG-VSV-ILT course

- Verification with SystemVerilog

Alternative to LANG-VSV-ILT course

- Designing with Verilog
- Designing with SystemVerilog

Software Tools

- Vivado® Design or System Edition 2017.1

Hardware

- Architecture: N/A*
- Demo board: Kintex® UltraScale™ FPGA KCU105 or Kintex-7 FPGA KC705 board*

* This course does not focus on any particular architecture. Check with North Pole Engineering, Inc. for the specifics of the in-class lab board or other customizations.

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

- Write RTL Verilog code for synthesis
- Write Verilog test fixtures for simulation
- Create a Finite State Machine (FSM) by using Verilog
- Target and optimize Xilinx FPGAs by using Verilog
- Use enhanced Verilog file I/O capability
- Run a timing simulation by using Xilinx Simprim libraries
- Create and manage designs within the Vivado Design Suite environment
- Download to the evaluation demo board
- Describe the features and benefits of using SystemVerilog for RTL design
- Identify the new data types supported in SystemVerilog
- Use an enumerated data type for coding a finite state machine (FSM)
- Explain how to use structures, unions, and arrays
- Describe the new procedural blocks and analyze the affected synthesis results
- Define the enhancements and ability to reuse tasks, functions, and packages
- Identify how to simplify module definitions and instantiations using interfaces
- Examine how to efficiently code in SystemVerilog for FPGA design simulation and synthesis
- Target and optimize Xilinx FPGAs by using SystemVerilog
- Synthesize and analyze SystemVerilog designs with the Vivado Design Suite
- Download a complete SystemVerilog design to an evaluation board

Course Outline

This course has more material than can be covered in five days. All slides and lab instructions will be provided to students. Some instructors may present material out of sequence, offer custom presentations demonstrations and labs based on student consensus.

Day 1

- Introduction to Verilog IEEE 1800-2005 and earlier {Lecture}
- Verilog Keywords and Identifiers {Lecture}
- Verilog Data Values and Number Representation {Lecture}
- Verilog Data Types {Lecture}
- Verilog Buses and Arrays {Lecture}
- Verilog Modules and Ports {Lecture, Lab, Demo}
- Verilog Operators {Lecture}
- Continuous Assignment {Lecture}
- Gate-Level Modeling {Lecture}
- Procedural Assignment {Lecture}
- Blocking and Non-Blocking Procedural Assignment {Lecture, Lab}
- Procedural Timing Control {Lecture}

Day 2

- Verilog Conditional Statements: if_else {Lecture, Lab}
- Verilog Conditional Statements: case {Lecture}
- Verilog Loop Statements {Lecture}
- Introduction to Verilog Testbenches {Lecture, Lab}
- System Tasks {Lecture}
- Verilog Sub-Programs {Lecture}
- Verilog Functions {Lecture}
- Verilog Tasks {Lecture}
- Verilog Compiler Directives {Lecture}
- Verilog Parameters {Lecture, Lab}
- Verilog Generate Statement {Lecture}

Day 3

- Verilog Timing Checks {Lecture}
- Finite State Machines {Lecture}
- Mealy Finite State Machine {Lecture, Lab}
- Moore Finite State Machine {Lecture, Lab}

- FSM Coding Guidelines {Lecture}
- File I/O: Introduction {Lecture}
- File I/O: Read Functions {Lecture, Lab}
- File I/O: Write Functions {Lecture}
- Targeting Xilinx FPGAs {Lecture, Lab}
- User-Defined Primitives {Lecture}
- Programming Language Interface {Lecture}

Day 4

- Introduction to SystemVerilog IEEE 1800-2009
- Data Types
- Demo: SystemVerilog Integer Data Types
- **SV Lab 1:** SystemVerilog Data Types
- Structures, Unions, and Arrays
- **SV Lab 2**: Structures and Unions
- Additional Operators in SystemVerilog
- Procedural Statements and Flow Control
- **SV Lab 3:** always_ff and always_comb Procedural Blocks

Day 5

- Functions, Tasks, and Packages
- **SV Lab 4**: Functions, Tasks, and Packages
- Interfaces
- Targeting Xilinx FPGAs
- **SV Lab 5:** Interfaces and Design Download

Lab Descriptions

The labs for this course provide a practical foundation for creating synthesizable RTL code. All aspects of the design flow are covered in the labs. The labs are written, synthesized, behaviorally simulated, and implemented by the student. The focus of the labs is to write code that will optimally infer reliable and high-performance circuits. The labs culminate in a functional calculator that students verify in simulation.

- **SV Lab 1:** SystemVerilog Data Types – Use enumerated data types to build a finite state machine and perform synthesis to analyze the results.
- **SV Lab 2:** Structures and Unions - Learn about packed and unpacked structures and unions and how to access their members.
- **SV Lab 3:** always_ff and always_comb Procedural Blocks – Learn to use the new procedural blocks always_comb, always_ff, and always_latch to produce the intended synthesized results.
- **SV Lab 4:** Functions, Tasks, and Packages - Create a new package and import that package into the module.
- **SV Lab 5:** Interfaces and Design Download – Use an interface to simplify the module inputs and outputs. Download and verify the design in-circuit.

Register Today

NPE, Inc. delivers public and private courses in locations throughout the central US region; including Iowa, Illinois, Kansas, Minnesota, Missouri, North Dakota, South Dakota and Wisconsin.

Visit **www.npe-inc.com/training**, for full course schedule and training information.

NORTH POLE npe ENGINEERING

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and Xilinx training credits.

Student Cancellation Policy

- Students cancellations received more than 7 days before the first day of class are entitled to a 100% refund. Refunds will be processed within 14 days.
- Student cancellations received less than 7 days before the first day of class are entitled to a 100% credit toward a future class.
- Student cancellations must be sent here.

NPE Course Cancellation Policy

- We regret from time to time classes will need to be rescheduled or cancelled.
- In the event of cancellation, live on-line training may be offered as a substitute.
- NPE may cancel a class up to 7 days before the scheduled start date of the class; all students will be entitled to a 100% refund.
- Under no circumstances is NPE responsible or liable for travel, lodging or other incidental costs. Please be aware of this cancellation policy when making your arrangements.
- For additional information or to schedule a private class contact us here.