Verilog
Hardware Description Language

한국외국어대학교 전자정보공학부
SoC Research Lab.
조 경 순

kscho@hufs.ac.kr

What is Verilog Language ?

- A hardware description language that provides a means of specifying a digital system at a wide range of levels of abstraction
- Supports the early conceptual stages of design with its behavioral level of abstraction and the later implementation stages with its structural level abstraction.
- Provides hierarchical constructs, allowing the designer to control the complexity of a description.

History of Verilog

- 1981
  - A CAE software company called Gateway Design Automation was founded by Prabhu Goel.
  - One of Gateway's first employees was Phil Moorby, who was an original author of GenRad's Hardware Description Language (GHDL) and HILO simulator.
- 1983
  - Gateway released the Verilog Hardware Description Language known as ***Verilog HDL*** or simply ***Verilog*** together with a Verilog simulator.
- 1985
  - The language and simulator were enhanced and the new version of the simulator was called ***Verilog-XL***.

History of Verilog

- 1983 to 1987
  - The Verilog-XL simulator gained a strong foothold among advanced, high-end designers mainly because it was fast, especially at the gate level and could handle designs in excess of 100,000 gates.
- 1987
  - Another startup company, Synopsys, began to use the proprietary Verilog behavioral language as an input to their synthesis product.
  - The IEEE released the ***VHDL*** standard, drawing attention to the possibilities of top-down design using a behavioral HDL and synthesis.
- December 1989
  - Cadence bought Gateway.

History of Verilog

- Early 1990
  - Cadence split the Verilog HDL and the Verilog-XL simulator into separate products, and then released the Verilog HDL to the public domain.
  - Cadence did this partly to compete with VHDL, which was a nonproprietary HDL, and mostly because Verilog users wanted to share models and knowledge about Verilog, which was not easy with a proprietary language.
  - ***Open Verilog International*** (***OVI***) was formed to control the language specification.
  - OVI is an industry consortium comprised of both Verilog users and CAE vendors.

History of Verilog

- 1990
  - Nearly all ASIC foundries supported Verilog and most used Verilog-XL as a golden simulator.
  - The golden simulator is the one that an ASIC vendor will use to sign-off a chip against, and guarantee that a manufactured chip will meet the same timing as that of the simulated model.
- 1993
  - Of all designs submitted to ASIC foundries in this year, 85% were designed and submitted using Verilog. (Source EE Times.)
- 1995
  - The Verilog language was reviewed and adopted by the IEEE as ***IEEE Standard 1364-1995***.

VHDL vs Verilog : Capability

- Hardware structure can be modeled equally effectively in both VHDL and Verilog.
- The choice of which to use is not based solely on technical capability but on:
  - Personal preferences
  - EDA tool availability
  - Commercial, business and marketing issues.
- The modeling constructs of VHDL cover a slightly higher levels of behavioral abstraction than those of Verilog.

VHDL vs Verilog : Compilation

- VHDL
  - Multiple design units (entity-architecture pairs) that reside in the same system file, may be separately compiled if so desired.
  - It's a good design practice to keep each design unit in its own system file.
- Verilog
  - The Verilog is still rooted in its native interpretative mode.
  - Compilation is a means of speeding up simulation, but has not changed the original nature of the language.
  - Care must be taken with both the compilation order of code written in a single file and the compilation order of multiple files.
  - Simulation results can change by simply changing the order of compilation.

VHDL vs Verilog : Data Types

- VHDL
  - A multiple of language or user-defined data types can be used.
  - Dedicated conversion functions are needed to convert objects from one type to another.
  - VHDL models may be easier to write and clearer to read because of its multiple data types such as enumerated data types.
- Verilog
  - Verilog data types are very simple, easy to use and very much geared towards modeling hardware structure as opposed to abstract hardware modeling.
  - All data types are defined by the Verilog language and not by the user.
  - Verilog may be preferred because of the simplicity of its data types.

VHDL vs Verilog : Design Reusability

- VHDL
  - Procedures and functions may be placed in a package so that they are available to any design unit that uses them.
- Verilog
  - There is no concept of packages in Verilog.
  - Functions and procedures used within a model must be defined in the module statement with which it will be used.
  - To make functions and procedures generally accessible from different module statements, they must be placed in a separate system file and included using the `include compiler directive.

VHDL vs Verilog : Easiest to Learn

- Starting with zero knowledge of either language, Verilog is probably the easiest to grasp and understand.
  - This assumes that the Verilog compiler directive language and the Programming Language Interface (PLI) language are not included.
  - If these languages are included, they can be looked upon as two additional languages that need to be learned.
- VHDL may seem less intuitive at first for two primary reasons:
  - It is very strongly typed, which is a feature that makes it robust and powerful for the advanced user after a longer learning phase.
  - There are many ways to model the same circuit, especially those with large hierarchical structures.

VHDL vs Verilog : Annotation

- A spin-off from Verilog is the Standard Delay Format (SDF), a general-purpose format used to define the timing delay in a circuit.
- The format provides a bidirectional link between chip layout tools, and either synthesis or simulation tools in order to provide more accurate timing representations.
- The SDF format is now an industry standard in its own right.

VHDL vs Verilog : High-Level Constructs

- VHDL has more high-level constructs and features than Verilog.
  - Package statements for model reuse
  - Configuration statements for configuring design structure
  - Generate statements for replicating structure
  - Generic statements for generic models that can be individually characterized, for example, bit width
- Except for being able to parameterize models by overloading parameter constants, there is no equivalent to the high-level VHDL modeling statements in Verilog.

VHDL vs Verilog : Language Extensions

- The use of language extensions will make a model nonstandard and most likely not portable across other design tools.
- Sometimes necessary in order to achieve the desired results.
- VHDL
  - Has an attribute called ‘foreign that allows architectures and subprograms to be modeled in another language.
- Verilog
  - The PLI is an interface mechanism between Verilog models and Verilog software tools.
  - A designer, or more likely, a Verilog tool vendor, can specify user-defined tasks or functions in the C programming language, and then call them from the Verilog source description.

VHDL vs Verilog : Libraries

- VHDL
  - A library is a storage area in the host environment for compiled entities, architectures, packages and configurations.
  - Useful for managing multiple design projects.
- Verilog
  - There is no concept of a library in Verilog.
  - Due to its origin as an interpretive language.

VHDL vs Verilog : Low-Level Constructs

- VHDL
  - Simple logical operators are built into the language.
  - NOT, AND, OR, NAND, NOR, XOR and XNOR
  - Any timing must be separately specified using the after clause.
- Verilog
  - The Verilog language was originally developed with gate-level modeling in mind, and so has very good constructs for modeling at this level.
  - Also has a very good constructs for modeling the cell primitives of ASIC and FPGA libraries, e.g., User-Defined Primitives (UDP), truth tables and the specify block for specifying timing delays across a module.

VHDL vs Verilog : Managing Large Designs

- VHDL
  - Configuration, generate and package statements, together with the generic clause, all help manage large design structures.
- Verilog
  - There are no statements in Verilog that help manage large designs.

VHDL vs Verilog : Operators

- The majority of operators are the same between the two languages.
- VHDL
  - Has the mod operator that is not found in Verilog.
- Verilog
  - Does have very useful unary operators that are not predefined in VHDL.
  - A loop statement can be used in VHDL to perform the same operation as a Verilog unary reduction operator.

VHDL vs Verilog : Parameterizable Models

- VHDL
  - A specific bit width model can be instantiated from a generic n-bit model using the generic clause.
  - The generic model will not synthesize until it is instantiated and the value of the generic given.
- Verilog
  - A specific bit width model can be instantiated from a generic n-bit model using overloaded parameter values.
  - The generic model must have a default parameter value defined.
  - In the absence of an overloaded value being specified, it will still synthesize, but will use the default parameter settings.

VHDL vs Verilog : Procedures and Tasks

- VHDL
  - Allows concurrent procedure calls.
- Verilog
  - Does not allow concurrent task calls.

VHDL vs Verilog : Readability

- A matter of coding style and experience than language feature
- VHDL
  - Its roots are based on Ada.
- Verilog
  - Its constructs are based approximately 50% on C and 50% on Ada.
- C programmers may prefer Verilog over VHDL.

VHDL vs Verilog : Structural Replication

- VHDL
  - The generate statement replicates a number of instances of the same design unit or some subpart of a design, and connects it appropriately.
- Verilog
  - There is no equivalent to the generate statement in Verilog.

VHDL vs Verilog : Test Harnesses

- Designers typically spend about 50% of their time writing synthesizable models and the other 50% writing a test harness to verify the synthesizable models.
- Test harnesses are not restricted to the synthesizable subset and so are free to use the full potential of the language.
- VHDL has generic and configuration statements that are useful in test harnesses, that are not found in Verilog.

VHDL vs Verilog : Verboseness

- VHDL
  - Because VHDL is a strongly-typed language, models must be coded precisely with defined and matching data types.
  - Models are often more verbose and the code often longer than its Verilog equivalent.

VHDL vs Verilog : Verboseness

- Verilog
  - Signals representing objects of different bit widths may be assigned to each other.
  - The signal representing the smaller number of bits is automatically padded out to that of the larger number of bits, and is independent of whether it is the assigned signal to or not.
  - Unused bits will be automatically optimized away during the synthesis process.
  - Has the advantage of not needing to model quite so explicitly as in VHDL.
  - Has the disadvantage that unintended modeling errors will not be identified by an analyzer.