Hardware Description Languages

Last updated 12/18/24

Hardware Description Languages

- Specialized programming languages used to describe the structure, design, and operation of digital and electronic systems
  - Typically used to design Integrated circuits (ICs) and field-programmable gate arrays (FPGAs)
  - Allow designers to model hardware components at different levels of abstraction
    - Behavioral, register-transfer level (RTL), and structural
  - Enable simulation of digital designs to verify their functionality
  - Enable synthesis of digital designs to generate hardware for implementation on devices
  - Designed for concurrent execution
    - Reflecting the parallel nature of hardware components

Hardware Description Languages

- Early versions
  - Mid 1980's
  - Focused on hardware description and simulation
  - VHDL
    - VHSIC Hardware Description Language
      - Very High Speed Integrated Circuit Hardware Description Language
    - Developed in a DOD/Corporate collaboration
  - Verilog
    - Verification – Logic → verilog
    - Developed by Gateway Design Automation
  - Both have since been made public and are covered by IEEE standards
  - Both are still in use today

Hardware Description Languages

- Advances
  - Private companies created tools to synthesize logic from the VHDL descriptions
    - Once synthesis was introduced – the language was effectively broken into 2 parts
      - Synthesizable VHDL
      - Un-synthesizable VHDL
  - Additional signal types and libraries were added
  - Support for analog hardware was added
  - Support for system level abstractions was added

Hardware Description Languages

- Current status
  - Dozens of HDLs exist today – a handful are widely used
  - VHDL
    - Used by DOD/Government contractors
    - Common in industrial manufacturers
  - Verilog
    - Similar capabilities to VHDL but more concise
    - Commonly used by commercial/consumer developers
  - System Verilog
    - Verilog with additional Object-Oriented hardware constructs
  - SystemC
    - C++ (object oriented) class library with objects designed to simulate and synthesize logic systems
    - Especially effective for very large/complex systems
    - Supports hardware/software interaction

Hardware Description Languages

- Synthesizable constructs
  - Code that can be converted to hardware
  - Concurrent logic
  - Sequential logic (processes with edge detection)
  - Structural code

- Un-synthesizable constructs
  - Code that cannot be converted to hardware
    - Allowed for simulation, descriptive or compilation purposes
  - Time – wait, delay
  - Loops – while, for, …
  - Initial conditions