Using VPI Interface

VPI (Verilog Procedural Interface) provides routines for accessing Verilog-AMS design data.

Enables dynamic interaction between Verilog-AMS and external software tools.

Facilitates integration with simulation and CAE (Computer-Aided Engineering) systems.

Common applications:

Delay calculators and annotators

Custom debugging tools

Co-simulation with other simulators

PI Callbacks – Dynamic Interaction Mechanism

VPI callbacks enable dynamic interaction with Verilog-AMS HDL products.

Allow user-defined applications to be triggered by simulation events.

Example use cases:

Call my_monitor() when a net value changes.

Call my_cleanup() at the end of simulation.

Enable advanced applications:

Co-simulation integration

Specialized timing checks

Complex debugging tools

Accessing Objects with VPI Routines

VPI routines allow access to objects in an instantiated Verilog-AMS design.

Each instance has unique, accessible objects (e.g., m1.w vs. m2.w).

Enables inspection and manipulation of:

HDL objects (e.g., wires, modules)

Simulation-specific objects (e.g., runtime values, event queues)

VPI as a simulation interface:

Goes beyond hierarchical language interfaces

Provides runtime information not available in static HDL analysis

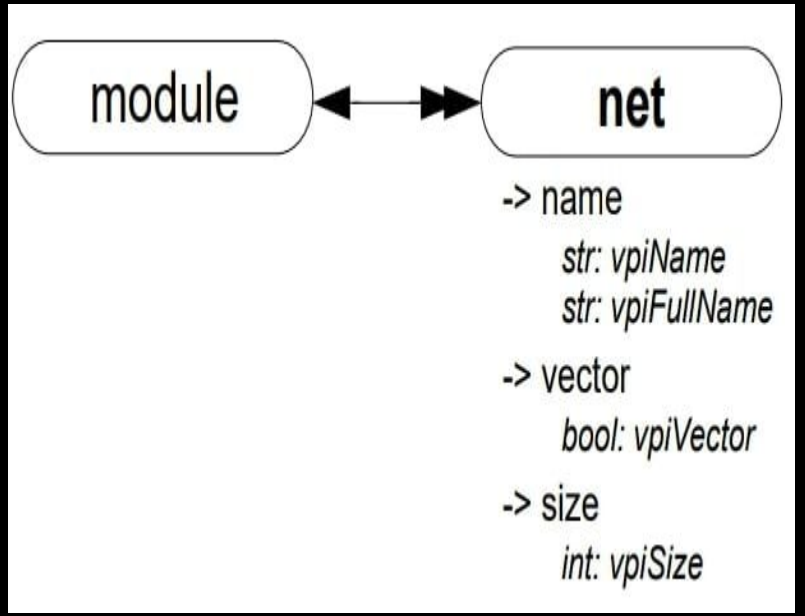
Slide 4
One-to-many relationships handled using:

vpi_iterate() to create an iterator

vpi_scan() to loop through objects

Getting properties
vpi_get() – returns integer/Boolean properties

vpi_get_str() – returns string properties



VPI routines are grouped based on their core use in Verilog-AMS simulation and analysis:

- *Simulation-Related Callbacks*
  Handle runtime events (e.g., value changes, simulation end)

- *System Task/Function Callbacks*
  Register and manage user-defined system tasks/functions

- *HDL Hierarchy Traversal*
  Navigate through instantiated Verilog-AMS design hierarchy

- *Object Property Access*
  Retrieve object attributes (e.g., names, types, values)

- *Object Access from Properties*
  Identify objects using property relationships

- *Delay Processing*
  Get/set propagation delays using structured data

Callbacks methods:
vpi_register_cb():Register a simulation-related callback
vpi_remove_cb():Remove a simulation-related callback
vpi_get_cb_info():Get information about a simulation-related callback
vpi_handle():Obtain a handle for an object with a one-to-one relationship
vpi_scan()/vpi_iterate():Obtain handles for objects in a one-to-many relationship

vpi_handle_multi():Obtain a handles for an object in a many-to-one relationship

vpi_get_delays():Retrieve delays or timing limits of an object
vpi_put_delays():Write delays or timing limits to an object