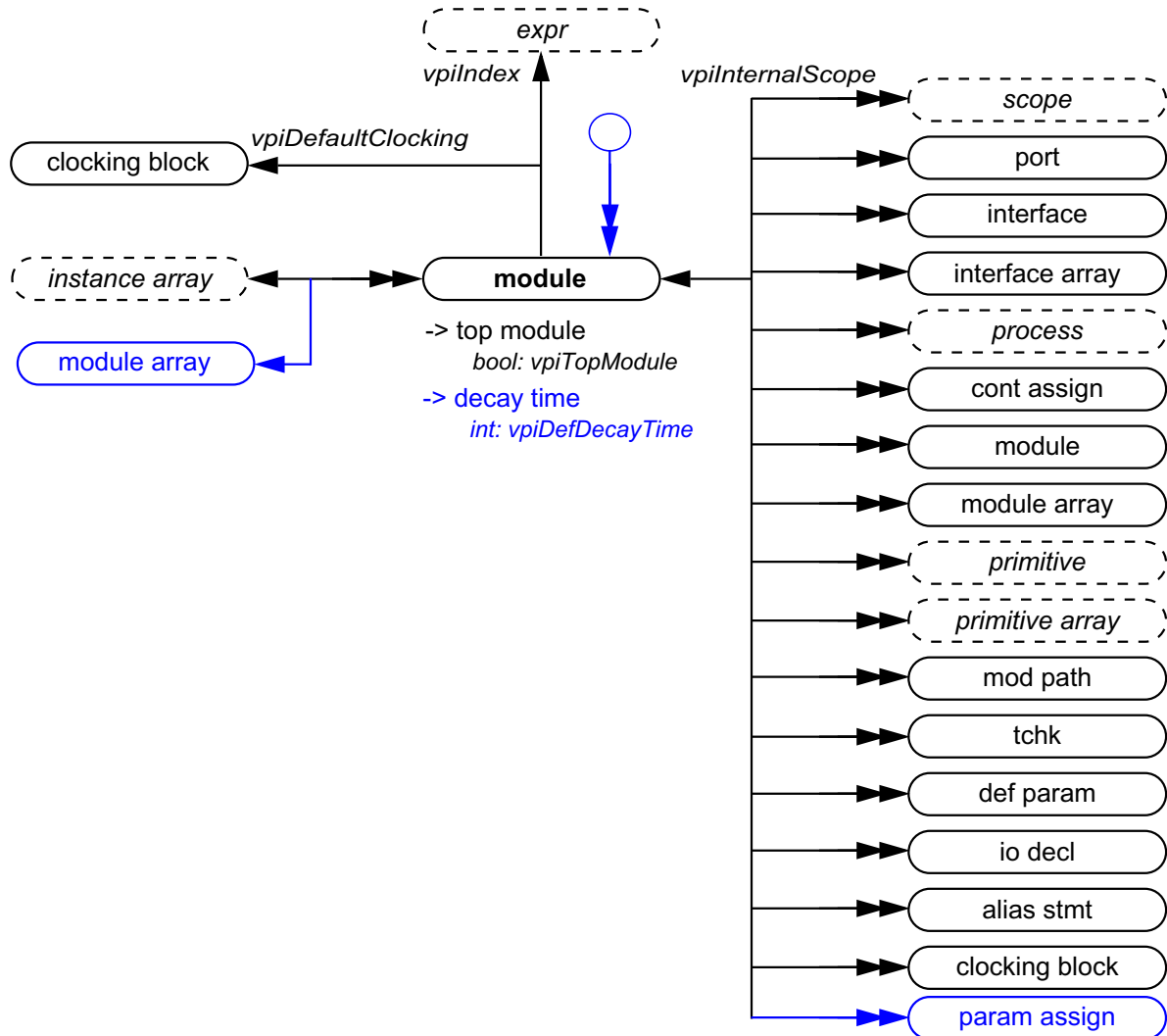


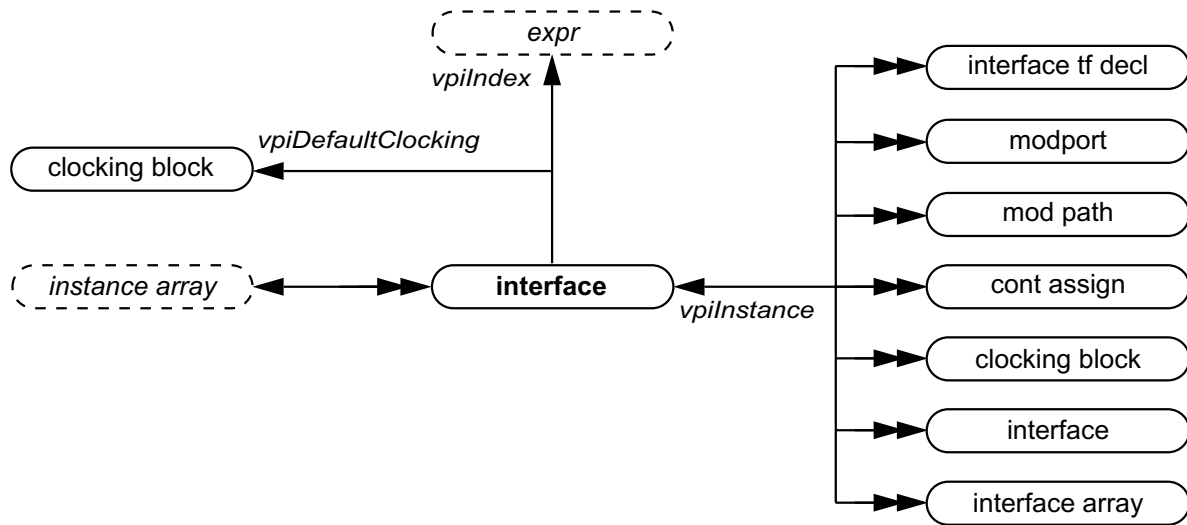
32.2 Module (supersedes P1364 26.6.1)



NOTES:

- 1) ~~vpiMemory~~ shall return array variable objects rather than ~~vpiMemory~~ objects. ~~The IEEE P1364 standard has made a similar update to the Verilog VPI (refer to note 1 in P1364, 26.6.9)~~ Top-level modules shall be accessed using **vpi_iterate()** with a **NULL** reference object.
- 2) If a module is an element within a module array, the **vpiIndex** transition is used to access the index within the array. If a module is not part of a module array, this transition shall return **NULL**.

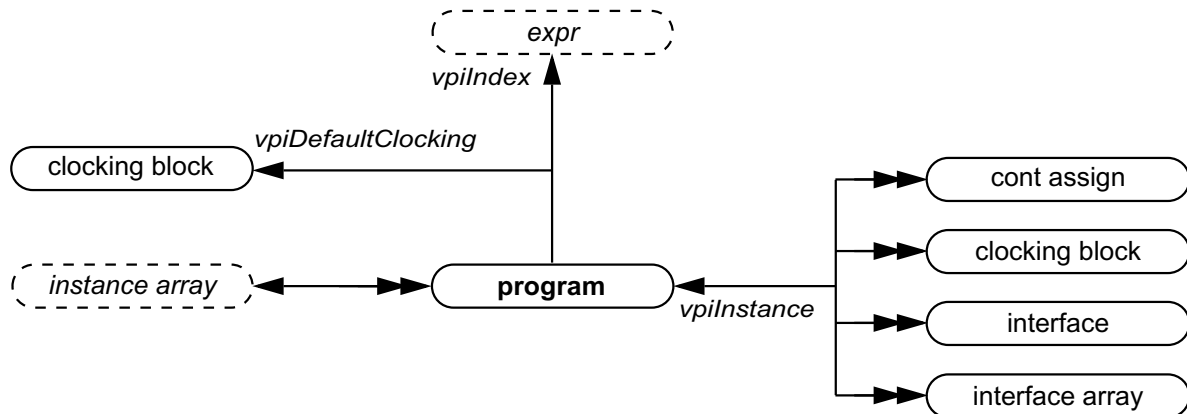
32.3 Interface



NOTE:

- 1) All interfaces are instances and all relations and properties in the Instances diagram also apply.
- 2) If an interface is an element within an instance array, the **vpilIndex** transition is used to access the index within the array. If an interface is not part of an instance array, this transition shall return NULL.

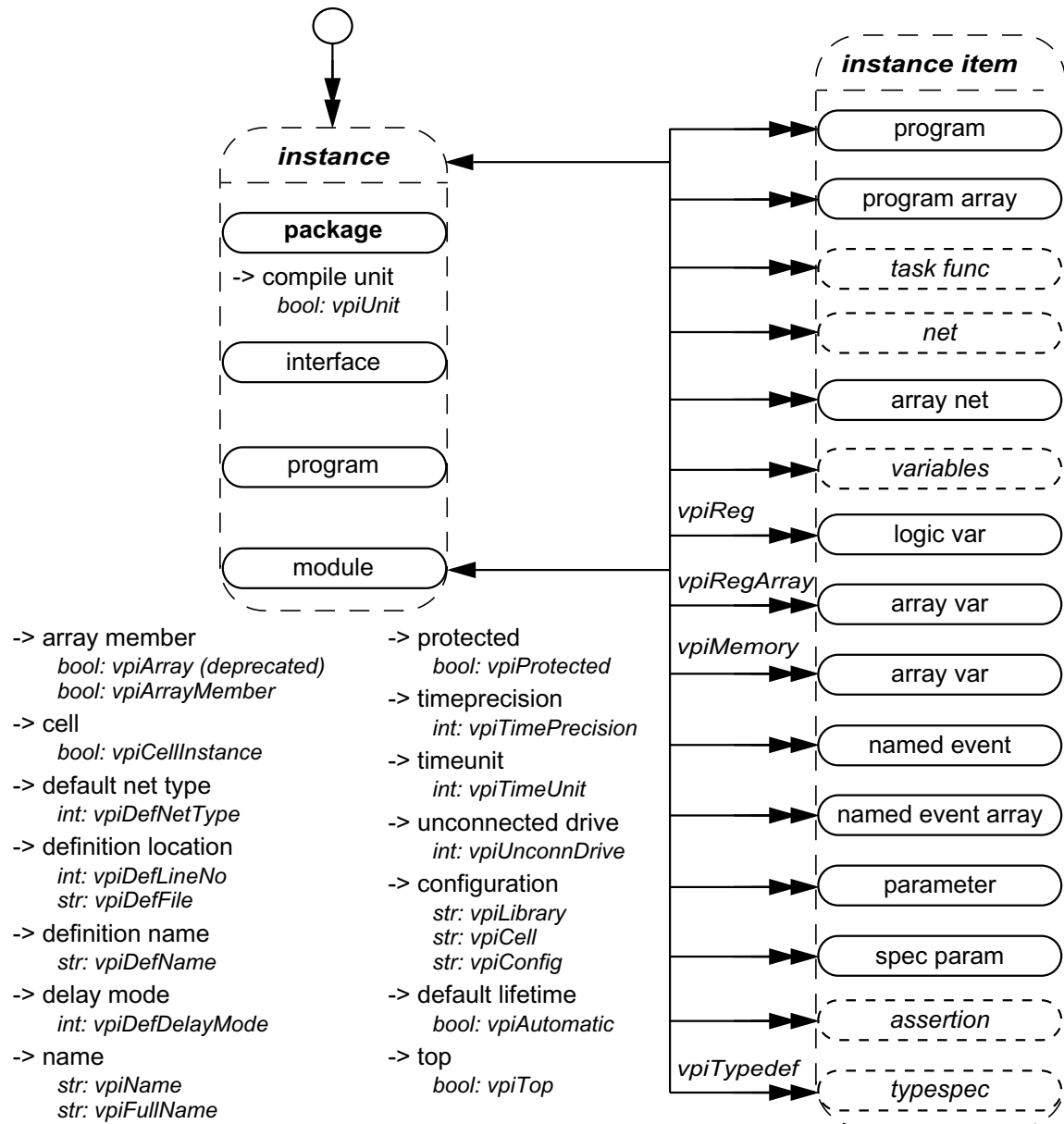
32.6 Program



NOTE:

- 1) All programs are instances and all relations and properties in the Instances diagram also apply.
- 2) If a program is an element within an instance array, the **vpilIndex** transition is used to access the index within the array. If a program is not part of an instance array, this transition shall return NULL.

32.7 Instance



NOTES:

- 1) The **vpiTypedef** iteration shall return the user-defined typespecs that have typedefs explicitly declared in the instance.
- 2) **vpiModule** shall return a module if the object is inside a module instance, otherwise NULL.
- 3) **vpiInstance** shall always return the immediate instance (package, module, program or interface) in which the object is instantiated.
- 4) **vpiFullName** for objects that exist within a compilation unit shall begin with '\$unit::' and therefore may be ambiguous. **vpiFullName** for a package shall be the name of the package and should end with "::"; this syntax disambiguates between a module and a package of the same name. **vpiFullName** for objects that exist in a package shall begin with the name of the package followed by "::". The separator "::" shall only appear

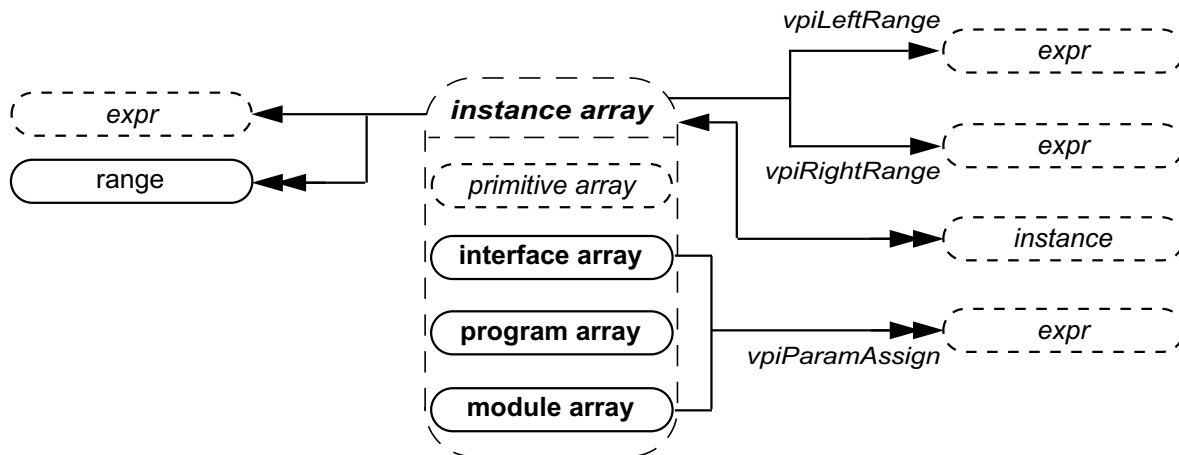
between the package name and the immediately following the name component. The . separator shall be used in all other cases.

5) The following items shall not be accessible via `vpi_handle_by_name()`:

- imported items
- objects that exist within a compilation unit

- 6) Passing a NULL handle to `vpi_get()` with `types` properties (should be fixed in 26.6.1 of 1364 too) `vpiTimePrecision` or `vpiTimeUnit` shall return the smallest time precision of all modules in the instantiated design.
- 7) The properties `vpiDefLineNo` and `vpiDefFile` can be affected by the `'line` compiler directive. See [P1364 19.7](#) for more details on the `'line` directive.

32.8 Instance arrays (supersedes P1364 26.6.2)



-> access by index

`vpi_handle_by_index()`
`vpi_handle_by_multi_index()`

-> name

`str: vpiName`
`str: vpiFullName`

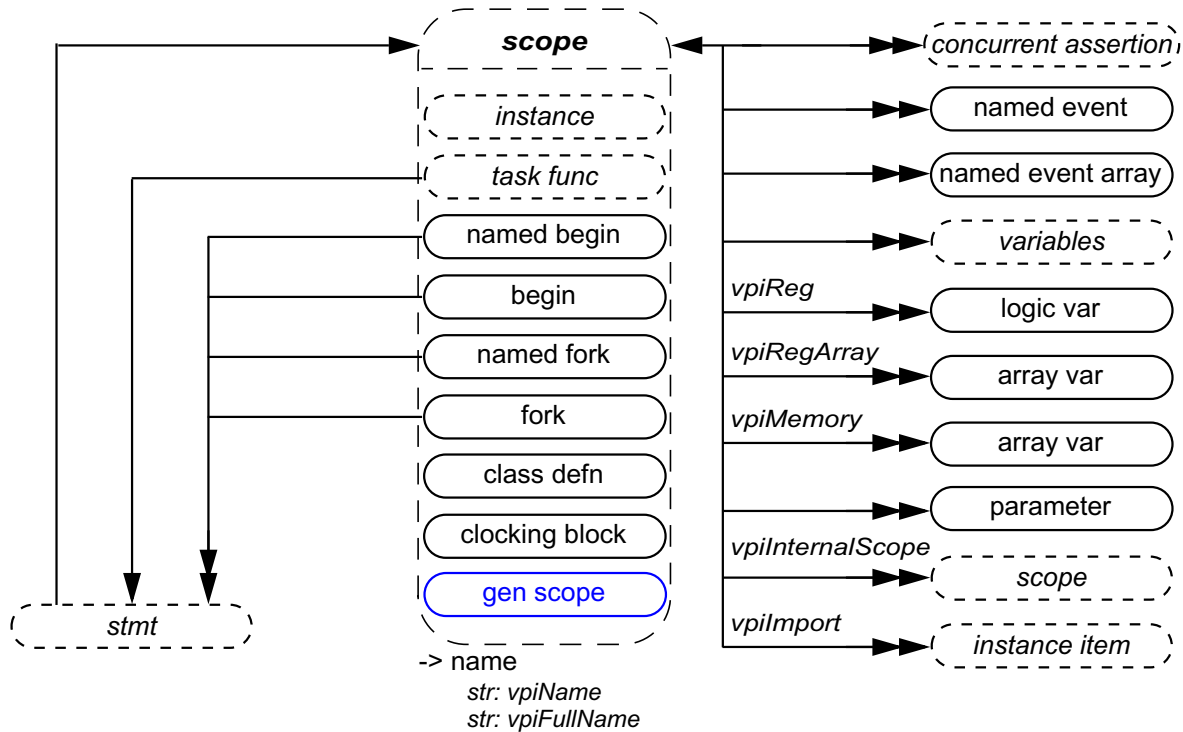
-> size

`boolint: vpiSize` (needs to be fixed in 1364 too)

NOTES:

- 1) Traversing from the instance array to **expr** shall return a simple expression object of type **vpiOperation** with a **vpiOpType** of **vpiListOp**. This expression can be used to access the actual list of connections to the ~~module or primitive~~ instance array in the Verilog source code.
- 2) Param assignments can only be obtained from non-primitive instance arrays.
- 3) To obtain all the dimensions of a multidimensional array, the range iterator must be used. Using the **vpiLeftRange/vpiRightRange** properties only returns the last dimension of a multidimensional array.

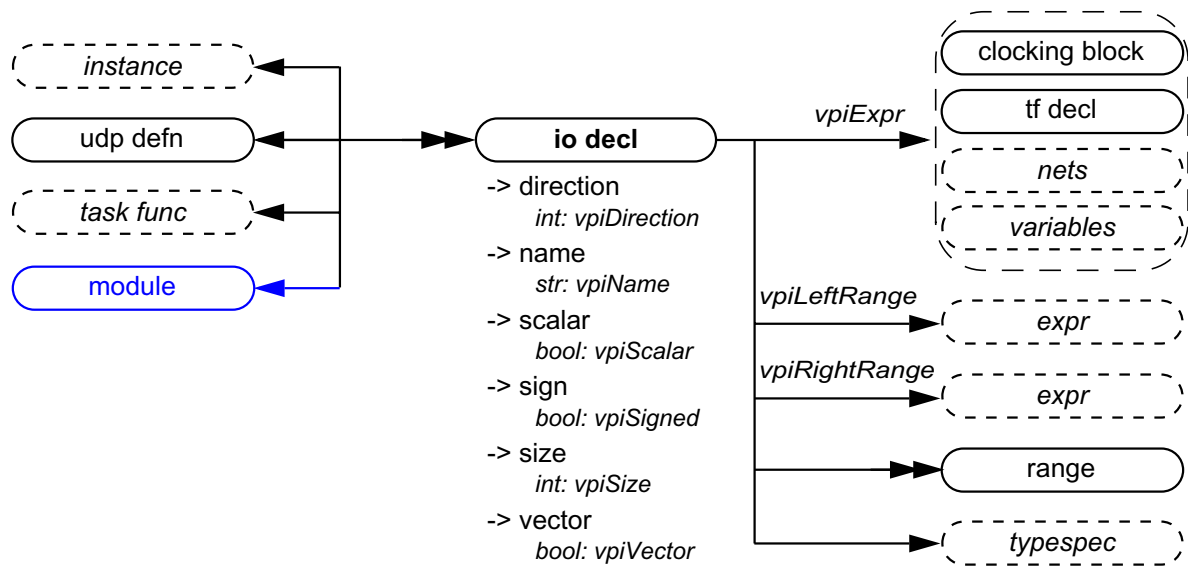
32.9 Scope (supersedes P1364 26.6.3)



NOTES:

- 1) Unnamed scopes shall have valid names, though tool dependent.
- 2) The **vpiImport** iterator shall return all objects imported into the current scope via import statements. Note that only objects actually referenced through the import shall be returned, rather than items potentially made visible as a result of the import. Refer to 19.2.2 for more details.
- 3) A task func can have zero or more statements (see 11.2 and 11.3). If the number of statements is greater than 1, the **vpiStmt** relation shall return an unnamed **begin** that contains the statements of the task or function. If the number of statements is zero, the **vpiStmt** relation shall return NULL.

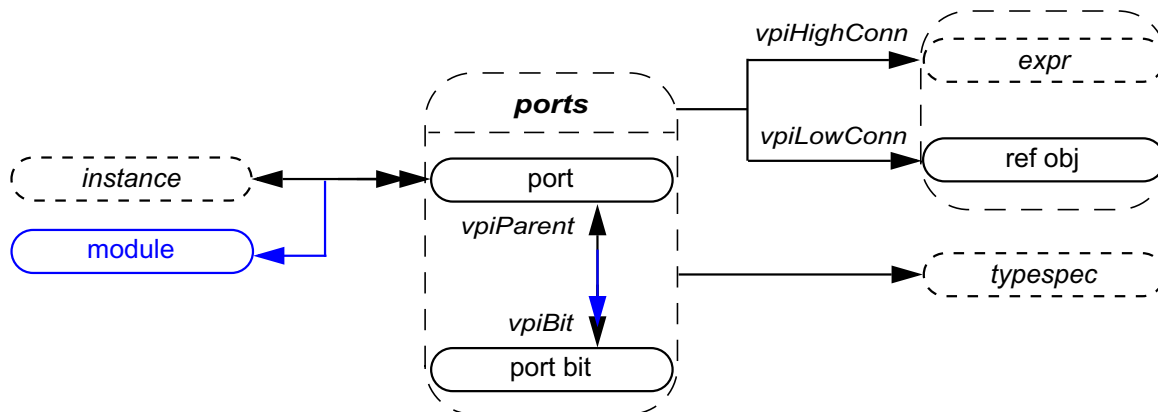
32.10 IO declaration (supersedes P1364 26.6.4)



NOTE:

— **vpiDirection** returns **vpiRef** for pass by **ref** ports.

32.11 Ports (supersedes P1364 26.6.5)

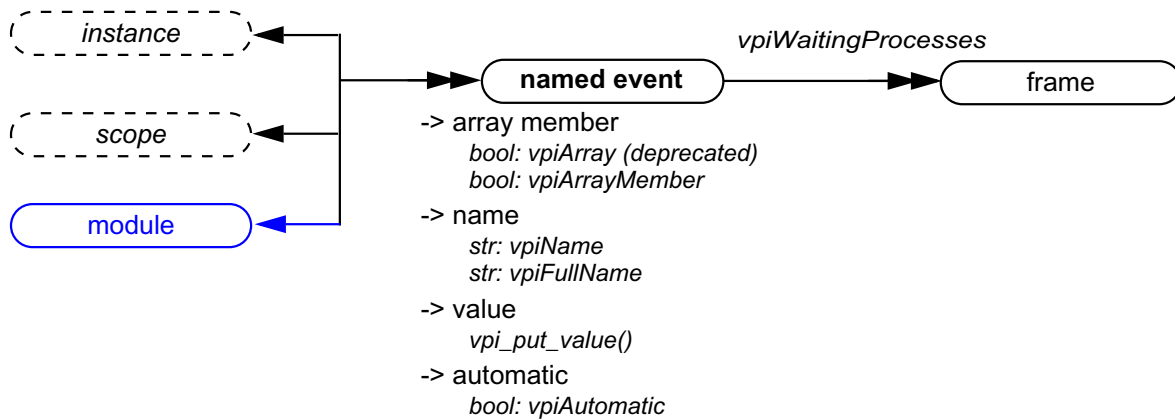


-> connected by name <i>bool: vpiConnByName</i>	-> port type <i>int: vpiPortType</i>
-> delay (mipd) <i>vpi_get_delays()</i> <i>vpi_put_delays()</i>	-> scalar <i>bool: vpiScalar</i>
-> direction <i>int: vpiDirection</i>	-> size <i>int: vpiSize</i>
-> explicitly named <i>bool: vpiExplicitName</i>	-> vector <i>bool: vpiVector</i>
-> index <i>int: vpiPortIndex</i>	-> access by index <i>vpi_handle_by_index()</i> <i>vpi_handle_by_multi_index()</i>
-> name <i>str: vpiName</i>	

NOTES:

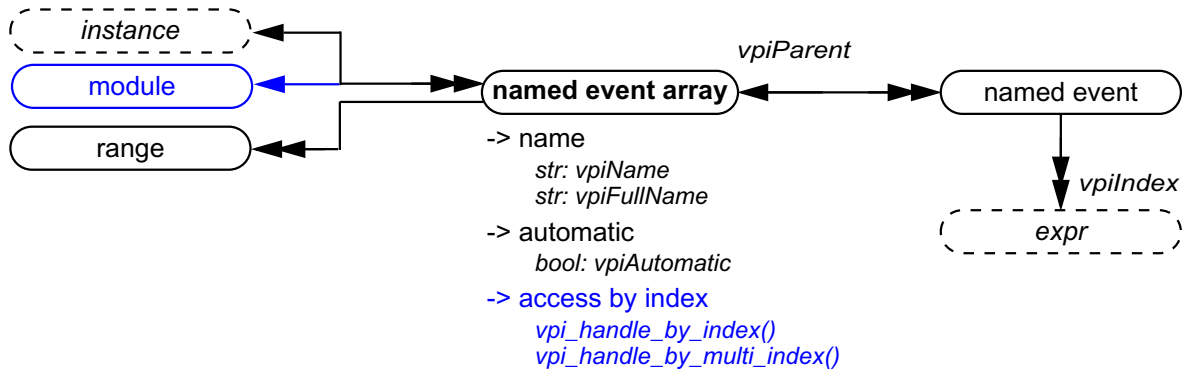
- 1) **vpiPortType** shall be one of the following three types: **vpiPort**, **vpiInterfacePort**, and **vpiModportPort**. Port type depends on the formal, not on the actual.
- 2) **vpi_get_delays**, **vpi_put_delays** delays shall not be applicable for **vpiInterfacePort**.
- 3) **vpiHighConn** shall indicate the hierarchically higher (closer to the top module) port connection.
- 4) **vpiLowConn** shall indicate the lower (further from the top module) port connection.
- 5) **vpiLowConn** of a **vpiInterfacePort** shall always be **vpiRefObj**.
- 6) Properties ~~scalar and vector~~ **vpiScalar** and **vpiVector** shall indicate if the port is 1 bit or more than 1 bit. They shall not indicate anything about what is connected to the port.
- 7) Properties ~~index and name~~ **vpiIndex** and **vpiName** shall not apply for port bits.
- 8) If a port is explicitly named, then the explicit name shall be returned. If not, and a name exists, then that name shall be returned. Otherwise, NULL shall be returned.
- 9) **vpiPortIndex** can be used to determine the port order. The first port has a port index of zero.
- 10) ~~**vpiHighConn** and **vpiLowConn** shall return NULL if the port is not connected.~~ **vpiLowConn** shall return NULL if the module or interface or program port is a null port (e.g. "**module** M();"). **vpiHighConn** shall return NULL if the instance of the module or interface or program does not have a connection to the port.
- 11) **vpiSize** for a null port shall return 0.

32.19 Named events (supersedes P1364 26.6.11)



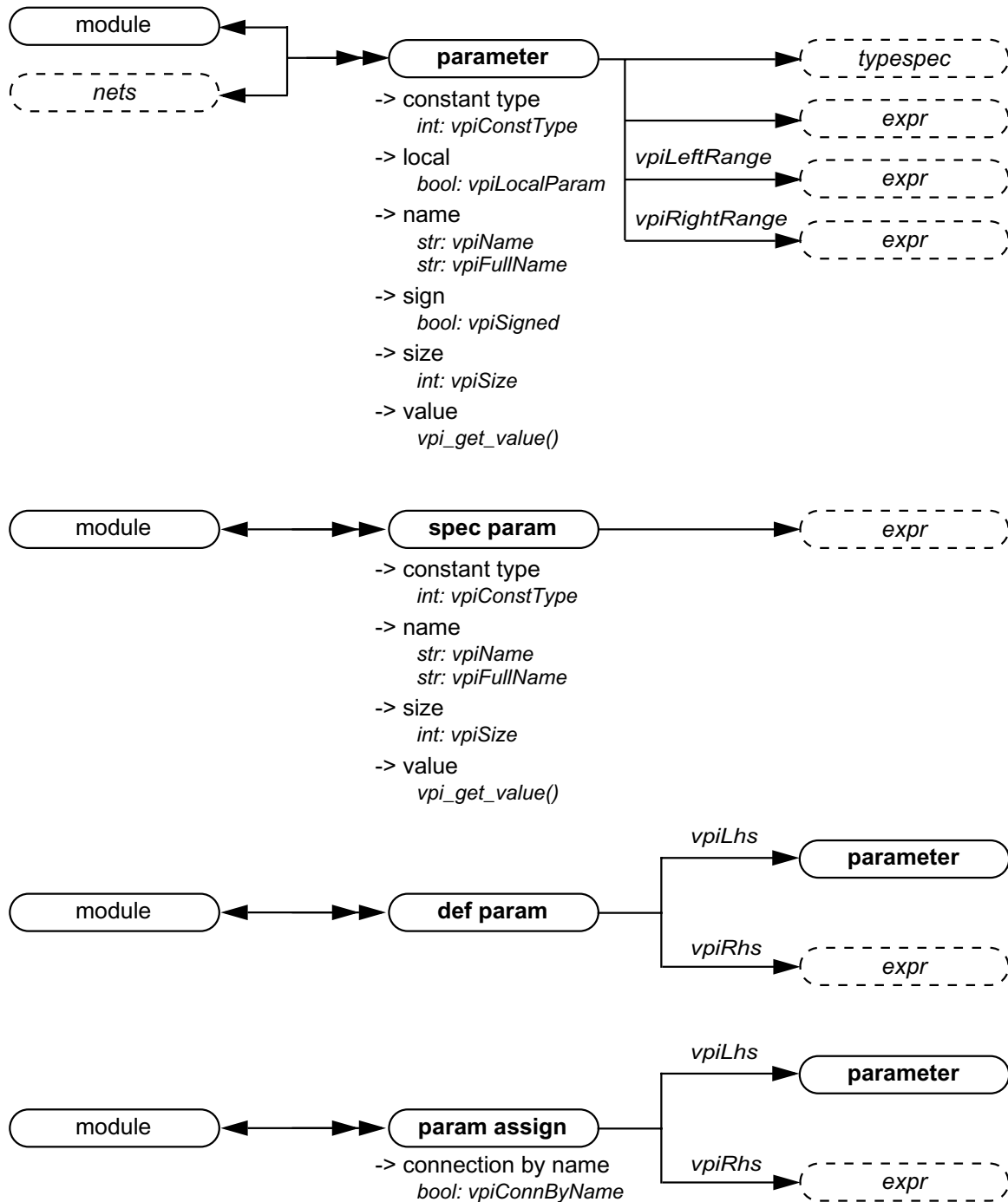
NOTE:

- The **vpiWaitingProcesses** iterator returns all waiting processes, identified by their frame, for that named event.



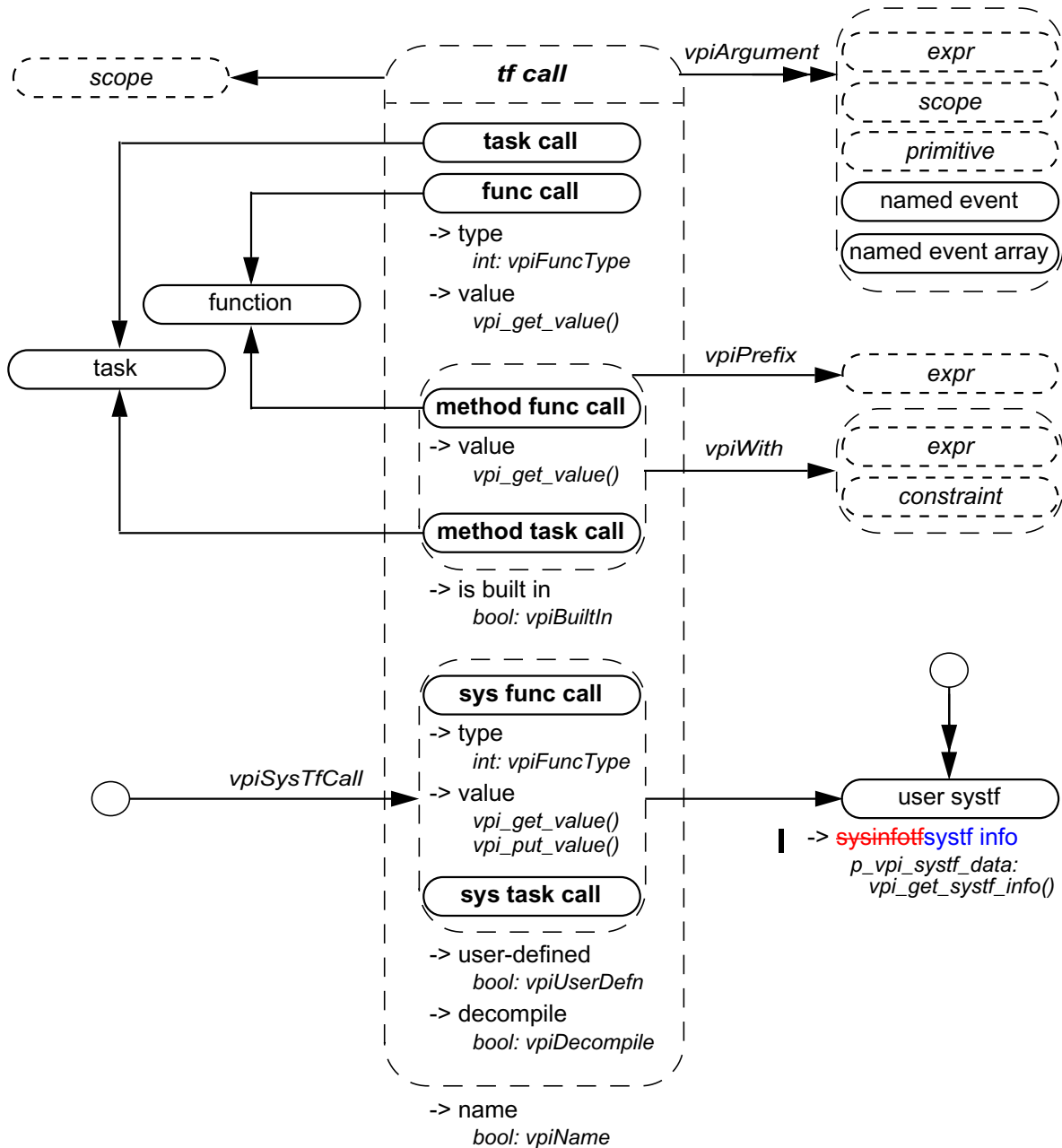
NOTE:

- **vpi_iterate(vpiIndex, named_event_handle)** shall return the set of indices for a named event within an array, starting with the index for the named event and working outward. If the named event is not part of an array, a NULL shall be returned.

32.20 Parameter (supersedes P1364 26.6.12)**NOTES:**

- 1) Obtaining the value from the object **parameter** shall return the final value of the parameter after all module instantiation overrides and defparams have been resolved.
- 2) **vpiLhs** from a param assign object shall return a handle to the overridden parameter.
- 3) If a parameter does not have an explicitly defined range or is a type parameter, **vpiLeftRange** and **vpiRightRange** shall return a NULL handle.

32.27 Task and function call (supersedes P1364 26.6.19)



NOTES:

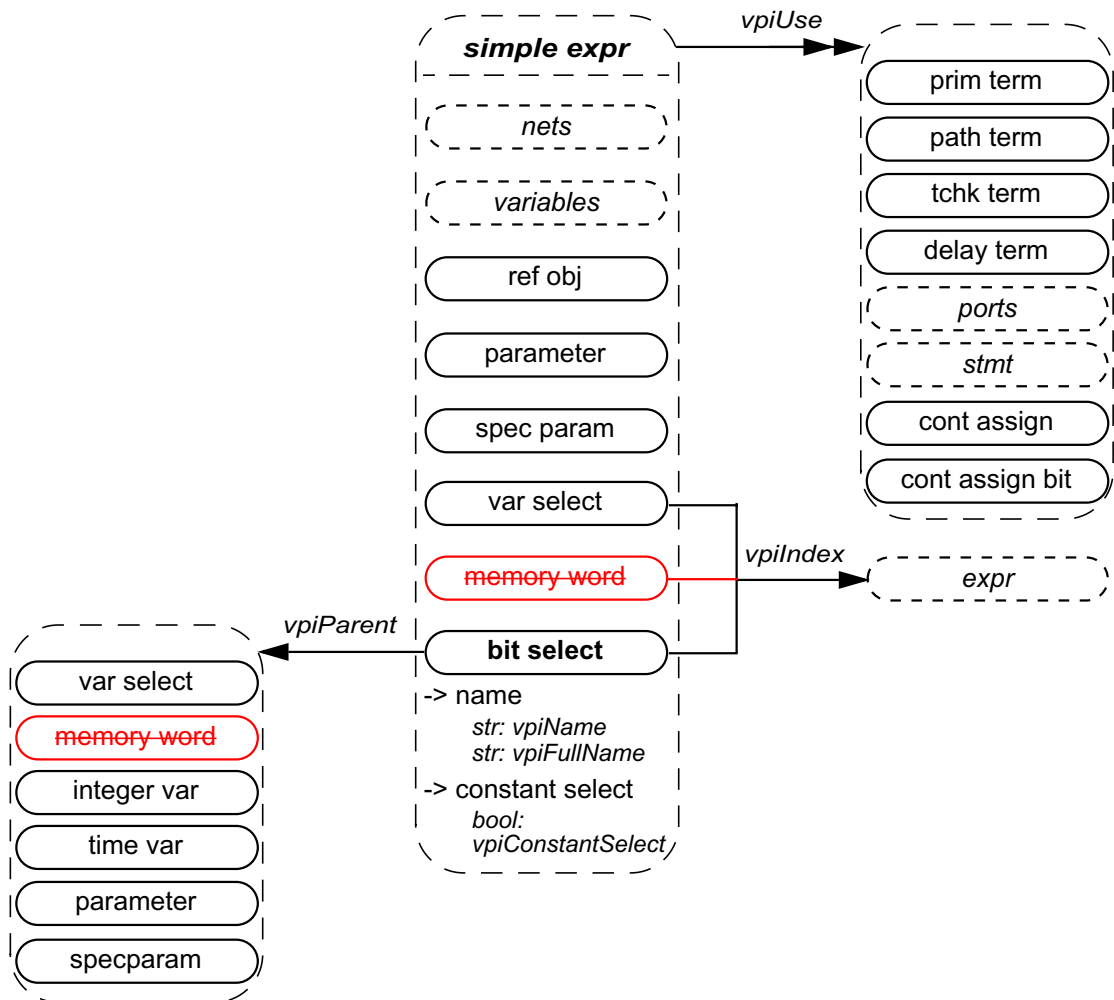
- 1) The **vpiWith** relation is only available for randomize methods (see 13.6) and for array locator methods (see 5.15.1).
- 2) For methods (method func call, method task call), the **vpiPrefix** relation shall return the object to which the method is being applied. For example, for the class method invocation

```
packet.send();
```

the prefix for the “send” method is the class var “packet”.

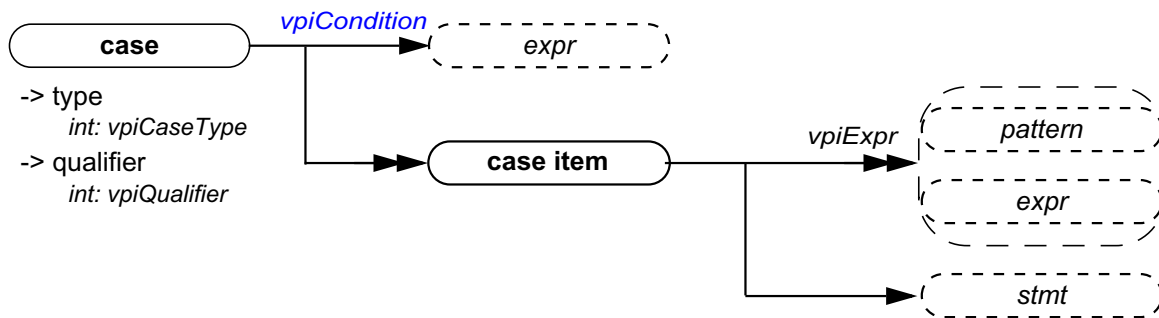
- 3) The system task or function that invoked an application shall be accessed with **vpi_handle(vpiSysTfCall, NULL)**
- 4) **vpi_get_value()** shall return the current value of the system function.
- 5) If the **vpiUserDefn** property of a system task or function call is true, then the properties of the corresponding systf object shall be obtained via **vpi_get_systf_info()**.
- 6) All user-defined system tasks or functions shall be retrieved using **vpi_iterate()**, with **vpiUserSystf** as the type argument, and a NULL reference argument.
- 7) Arguments to PLI tasks or functions are not evaluated until an application requests their value. Effectively, the value of any argument is not known until the application asks for it. When an argument is an HDL or system function call, the function cannot be evaluated until the application asks for its value. If the application never asks for the value of the function, it is never evaluated. If the application has a handle to an HDL or system function it may ask for its value at any time in the simulation. When this happens the function is called and evaluated at this time.
- 8) A null argument is an expression with a **vpiType** of **vpiOperation** and a **vpiOpType** of **vpiNullOp**.
- 9) The property **vpiDecompile** ~~will~~ shall return a string with a functionally equivalent system task or function call to what was in the original HDL. The arguments ~~will~~ shall be decompiled using the same manner as any expression is decompiled. See [32.39](#) for a description of expression decompilation.

32.38 Simple expressions (supersedes P1364 26.6.25)



NOTES:

- 1) For vectors, the **vpiUse** relationship shall access any use of the vector or part-selects or bit-selects thereof.
- 2) For bit-selects, the **vpiUse** relationship shall access any specific use of that bit, any use of the parent vector, and any partselect that contains that bit.

32.47 Case, pattern (supersedes P1364 26.6.36)**NOTES:**

- 1) The *case item* shall group all case conditions that branch to the same statement.
- 2) **vpi_iterate()** shall return NULL for the default case item since there is no expression with the default case.

