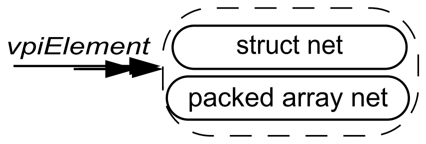
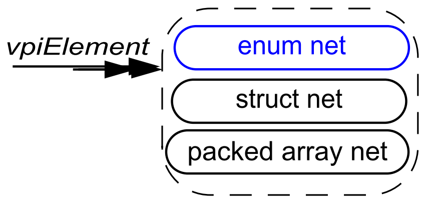
36.15 Nets

REPLACE:



WITH:



36.15 Nets [continued]

REPLACE:

1) Any net declared as an array with one or more unpacked ranges is an array net. Any packed struct net declared with one or more explicit packed ranges is a packed array net. The range iterator for a packed array net returns only the explicit packed ranges for such a net. It shall not return the implicit range of the packed struct net elements themselves. For example:

WITH:

1) Any net declared as an array with one or more unpacked ranges is an array net. Any packed struct net or enum net declared with one or more explicit packed ranges is a packed array net. The range iterator for a packed array net returns only the explicit packed ranges for such a net. It shall not return the implicit range of the packed struct net elements themselves. For example:

....

REPLACE:

2) The boolean property **vpiArray** is deprecated in this standard. The **vpiArrayMember** property shall be TRUE for a net that is an element of an array net. It shall be FALSE otherwise. The **vpiPackedArrayMember** property shall be TRUE for a packed struct net or a packed array net that is an element of a packed array net.

WITH:

2) The boolean property **vpiArray** is deprecated in this standard. The **vpiArrayMember** property shall be TRUE for a net that is an element of an array net. It shall be FALSE otherwise. The **vpiPackedArrayMember** property shall be TRUE for a packed struct net, an enum net, or a packed array net that is an element of a packed array net.

....

REPLACE:

22) **vpi_iterate(vpiIndex, net_handle)** shall return the set of indices for a net within an array net, starting with the index for the net and working outward. If the net is not part of an array (the **vpiArrayMember** property is FALSE), a NULL shall be returned. The **vpiIndex** iterator shall work similarly for packed array net elements (packed struct nets or packed array nets whose **vpiPackedArrayMember** property is TRUE). The indices returned shall start with the index of the element and work outward until the **vpiParent** packed array net is reached (see detail 28). The indices retrieved for packed array net elements shall be the same as those shown in the example for detail 29 for each of the sub-elements returned by **vpiElement**. The indices will be retrieved in right-to-left order as they appear in the text.

WITH:

22) **vpi_iterate(vpiIndex, net_handle)** shall return the set of indices for a net within an array net, starting with the index for the net and working outward. If the net is not part of an array (the **vpiArrayMember** property is FALSE), a NULL shall be returned. The **vpiIndex** iterator shall work similarly for packed array net elements (packed struct nets, enum nets, or packed array nets whose **vpiPackedArrayMember** property is TRUE). The indices returned shall start with the index of the element and work outward until the **vpiParent** packed array net is reached (see detail 28). The indices retrieved for packed array net elements shall be the same as those shown in the example for detail 29 for each of the sub-elements returned by **vpiElement**. The indices will be retrieved in right-to-left order as they appear in the text.

36.15 Nets [continued]

REPLACE:

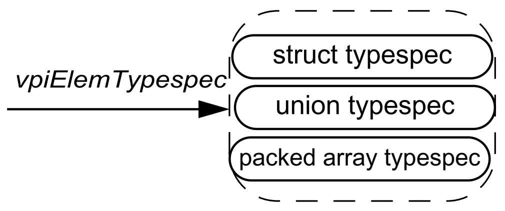
29) The **vpiElement** transition shall be used to iterate over the sub-elements of packed array nets. Unlike **vpiNet** iterations for **vpiArrayNet** objects, **vpiElement** shall retrieve elements for only one dimension level at a time. This means that for multi-dimensioned packed array nets, **vpiElement** shall retrieve elements which are themselves also **vpiPackedArrayNet** objects. **vpiElement** can then be used to iterate over the sub-elements of these objects and so on, until the leaf level struct nets are returned. In other words, the datatype of each element retrieved by **vpiElement** is equivalent to the original **vpiPackedArrayNet** object's datatype with one leftmost packed range removed. For example, consider the following **vpiPackedArrayNet** object:

WITH:

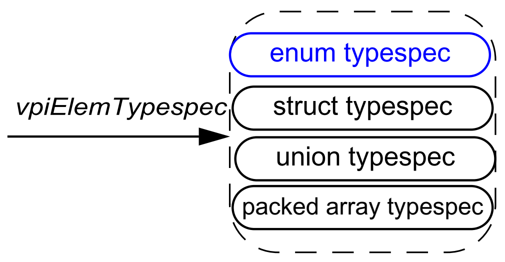
29) The **vpiElement** transition shall be used to iterate over the sub-elements of packed array nets. Unlike **vpiNet** iterations for **vpiArrayNet** objects, **vpiElement** shall retrieve elements for only one dimension level at a time. This means that for multi-dimensioned packed array nets, **vpiElement** shall retrieve elements which are themselves also **vpiPackedArrayNet** objects. **vpiElement** can then be used to iterate over the sub-elements of these objects and so on, until the leaf level struct nets or enum nets are returned. In other words, the datatype of each element retrieved by **vpiElement** is equivalent to the original **vpiPackedArrayNet** object's datatype with one leftmost packed range removed. For example, consider the following **vpiPackedArrayNet** object:

36.21 Typespec

REPLACE:



WITH:



36.21 Typespec [continued]

REPLACE:

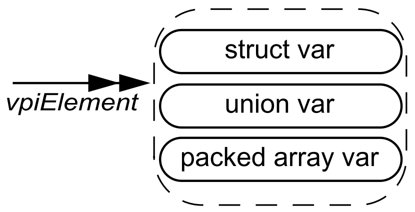
8) The **vpiElemTypespec** transition shall be used to unwind the typespec of an unpacked array (array typespec) or a packed array (packed array typespec, or a bit or logic typespec with one or more dimensions) one dimension level at a time. This means that for a multi-dimensional array typespec (a typespec with more than one unpacked range), **vpi_handle(vpiElemTypespec, array_typespec_handle)** shall initially retrieve a **vpiArrayTypespec** equivalent to the original typespec with its leftmost unpacked range removed. Subsequent calls to the **vpiElemTypespec** method continue the unwinding until a typespec object is retrieved that has no unpacked ranges remaining. Similarly, when the **vpiElemTypespec** is applied to a typespec of a multi-dimensional packed array object, a **vpiPackedArrayTypespec** (or **vpiBitTypespec** or **vpiLogicTypespec**) is retrieved which is equivalent to the original typespec with its leftmost packed range removed, and so on, until a typespec without an explicit packed range is retrieved. When the **vpiElemTypespec** relation is applied to a **vpiStructTypespec**, **vpiUnionTypespec**, or a **vpiBitTypespec** or **vpiLogicTypespec** with no ranges present, it shall return NULL. This allows packed or unpacked array typespecs constructed with multiple typedefs to be unwound without losing name information. Consider the complex array typespec defined below for “arr”:

WITH:

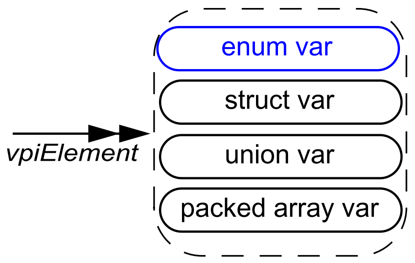
8) The **vpiElemTypespec** transition shall be used to unwind the typespec of an unpacked array (array typespec) or a packed array (packed array typespec, or a bit or logic typespec with one or more dimensions) one dimension level at a time. This means that for a multi-dimensional array typespec (a typespec with more than one unpacked range), **vpi_handle(vpiElemTypespec, array_typespec_handle)** shall initially retrieve a **vpiArrayTypespec** equivalent to the original typespec with its leftmost unpacked range removed. Subsequent calls to the **vpiElemTypespec** method continue the unwinding until a typespec object is retrieved that has no unpacked ranges remaining. Similarly, when the **vpiElemTypespec** is applied to a typespec of a multi-dimensional packed array object, a **vpiPackedArrayTypespec** (or **vpiBitTypespec** or **vpiLogicTypespec**) is retrieved which is equivalent to the original typespec with its leftmost packed range removed, and so on, until a typespec without an explicit packed range is retrieved. When the **vpiElemTypespec** relation is applied to a **vpiStructTypespec**, **vpiUnionTypespec**, **vpiEnumTypespec**, or a **vpiBitTypespec** or **vpiLogicTypespec** with no ranges present, it shall return NULL. This allows packed or unpacked array typespecs constructed with multiple typedefs to be unwound without losing name information. Consider the complex array typespec defined below for “arr”:

36.17 Packed array variables

REPLACE:



WITH:



....

REPLACE:

1) **vpiPackedArrayVar** objects shall represent packed arrays of packed struct var and union var objects. The properties **vpiVector**, and **vpiPacked** for these objects and their underlying struct var or union var elements shall always be TRUE (see 36.16).

WITH:

1) **vpiPackedArrayVar** objects shall represent packed arrays of packed struct var~~ and~~, union var, or enum var objects. The properties **vpiVector**, and **vpiPacked** for these objects and their underlying struct var~~ or~~, union var, or enum var elements shall always be TRUE (see 36.16).

....

REPLACE:

2) For consistency with other variable-width vector objects, the **vpiSize** property for **vpiPackedArrayVar** objects shall be the number of bits in the packed array, not the number of struct or union elements. The total number of struct var or union var elements for a packed array var can be obtained by computing the product of the **vpiSize** property for all of its packed ranges.

WITH:

2) For consistency with other variable-width vector objects, the **vpiSize** property for **vpiPackedArrayVar** objects shall be the number of bits in the packed array, not the number of struct, enum, or union var elements. The total number of struct var, enum var, or union var elements for a packed array var can be obtained by computing the product of the **vpiSize** property for all of its packed ranges.

36.17 Packed array variables (continued)

REPLACE:

3) The **vpiElement** transition shall be used to iterate over the sub-elements of packed array variables. Unlike **vpiVarSelect** or **vpiReg** transitions for **vpiArrayVar** objects, **vpiElement** shall retrieve elements for only one dimension level at a time. This means that for multi-dimensioned packed arrays, **vpiElement** shall retrieve elements which are themselves also **vpiPackedArrayVar** objects. **vpiElement** can then be used to iterate over the sub-elements of these objects and so on, until the leaf level struct or union vars are returned. In other words, the datatype of each element retrieved by **vpiElement** is equivalent to the original **vpiPackedArrayVar** object's datatype with the leftmost packed range removed. For example, consider the following **vpiPackedArrayVar** object:

WITH:

3) The **vpiElement** transition shall be used to iterate over the sub-elements of packed array variables. Unlike **vpiVarSelect** or **vpiReg** transitions for **vpiArrayVar** objects, **vpiElement** shall retrieve elements for only one dimension level at a time. This means that for multi-dimensioned packed arrays, **vpiElement** shall retrieve elements which are themselves also **vpiPackedArrayVar** objects. **vpiElement** can then be used to iterate over the sub-elements of these objects and so on, until the leaf level struct, enum, or union vars are returned. In other words, the datatype of each element retrieved by **vpiElement** is equivalent to the original **vpiPackedArrayVar** object's datatype with the leftmost packed range removed. For example, consider the following **vpiPackedArrayVar** object:

....

REPLACE:

4) The **vpiPackedArrayMember** property shall be `TRUE` for any struct var, union var, or packed array var whose **vpiParent** is a packed array var (see detail 29 of 36.16).

WITH:

4) The **vpiPackedArrayMember** property shall be `TRUE` for any struct var, union var, enum var, or packed array var whose **vpiParent** is a packed array var (see detail 29 of 36.16).