SV-EC issue 2380: Assignment compatibility of unpacked arrays

Applies to: IEEE 1800-2008 ballot draft

Version 1, 31-May-2009: Initial proposal

Version 2, 08-Jun-2009: Re-affirm element equivalence for assignment compatibility

Version 2A, 10-Jun-2009: Various errata fixed. Changes from version 2 marked by sidebars.

Background and discussion

In IEEE 1800-2005, arrays had to be *equivalent* to be assignment compatible. Mantis items 1702 (unpacked array concatenation) and 1447 (various changes relating to arrays) have made the existing LRM text self-contradictory, and clearly require that arrays can be assignment compatible if their elements are of *assignment compatible* type and their shapes match.

This proposal reaffirms the requirement for type equivalence of elements if arrays are to be assignment compatible.

Details of the proposal

ADD this new paragraph between the two existing paragraphs of 6.22.3:

… by truncation or rounding.

Unpacked arrays are assignment compatible with certain other arrays that are not of equivalent type. Assignment compatibility of unpacked arrays is discussed in detail in **7.6**.

Compatibility can be in one direction only. …

UPDATE clause 7.6 as follows:

For the purposes of assignment, a packed array is treated as a vector. Any vector expression can be assigned to any packed array. The packed array bounds of the target packed array do not affect the assignment. A packed array cannot be directly assigned to an unpacked array without an explicit cast.

Associative arrays are assignment compatible only with associative arrays, as described in **7.10.9**. A fixed-size unpacked array, dynamic array or queue, or a slice of such an array, shall be assignment compatible with any other such array or slice if all the following conditions are satisfied:

— the element types of source and target shall be equivalent;

— if the target is a fixed-size array or a slice, the source array shall have the same number of elements as the target.

Here *element* refers to elements of the slowest-varying array dimension. These elements may themselves be of some unpacked array type. Consequently, for two arrays to be assignment compatible it is necessary (but not sufficient) that they have the same number of unpacked dimensions.

Assignment compatibility of unpacked arrays is a weaker condition than type equivalence because it does not require their slowest-varying dimensions to be of the same unpacked array kind (queue, dynamic or fixed-size). Note, however, that this weaker condition applies only to the slowest-varying dimension. Any faster-varying dimensions must meet the requirements for equivalence (see **6.22.2**) for the entire arrays to be assignment compatible.

~~Assigning to a fixed-size unpacked array requires that the source and the target both be arrays with the same number of unpacked dimensions, the length of each dimension be the same, and each element be of an equivalent type. The same requirements shall be in effect if either or both of the arrays are slices. Assigning to a fixed-size unpacked array requires that the source and the target both be arrays with the same number of unpacked dimensions, the length of each dimension be the same, and each element be of an equivalent type. The same requirements shall be in effect if either or both of the arrays are slices. Assignment is done by assigning each element of the source array to the corresponding element of the target array. Element correspondence is defined as leftmost to leftmost, rightmost to rightmost, irrespective of index values.~~

Assignment shall be done by assigning each element of the source array to the corresponding element of the target array. Correspondence between elements is determined by the left-to-right order of elements in each array. For example, if array A is declared as **int** A[7:0] and array B is declared as **int** B[1:8], the assignment A = B; will assign element B[1] to element A[7], and so on. ~~Assigning fixed-size unpacked arrays of non-equivalent type to one another shall result in a compiler error.~~

~~See 6.22.2 for equivalent types.~~ If the target of the assignment is a queue or dynamic array, it shall be resized to have the same number of elements as the source expression and assignment shall then follow the same left-to-right element correspondence as described above.

```
int A[10:1]; // fixed-size array of 10 elements
int B[0:9]; // fixed-size array of 10 elements
int C[24:1]; // fixed-size array of 24 elements
A = B; // ok. Compatible type and same size
A = C; // type check error: different sizes
```

An array of wires can be assigned to an array of variables, and vice versa, if ~~they have the same number of unpacked dimensions, the same number of elements for each of those dimensions, and an equivalent type of elements. Assignment is done by assigning each element of the source array to the corresponding element of the target array.~~ the source and target arrays' data types are assignment compatible.

```
~~wire [31:0] W [9:0];~~
~~assign W = A;~~
~~initial #10 B = W;~~
  logic [7:0] V1[10:1];
  logic [7:0] V2[10];
  wire  [7:0] W[9:0];  // data type is logic [7:0] W[9:0]
  assign W = V1;
  initial #10 V2 = W;
```

~~Associative arrays are only assignment compatible with associative arrays (see 7.10.9). Fixed-size unpacked arrays, dynamic arrays and queues are assignment compatible with each other under the following conditions:~~

~~— Their element types shall be assignment compatible. Note that if the element type is any variety of array, then this is a recursive condition.~~

~~— If the destination is fixed-size, then the source and destination sizes shall be equal.~~

When a dynamic array or queue is assigned to a fixed-size array, the size of the source array cannot be determined until run time. An attempt to copy a dynamic array or queue into a fixed-size array target having a different number of elements~~Since the size of a dynamic array or queue can only be determined at run time, failure of the above condition~~ shall result in a run time error and no operation shall be performed. Example code showing assignment of a dynamic array to a fixed-size array follows.

```
int A[2][100:1];
```

…

MODIFY the second bullet point of clause 10.10 as follows (insert word "unpacked"):

— an item whose self-determined type is an unpacked array whose slowest-varying dimension's element type is assignment compatible with the element type of the target array shall represent as many elements as exist in that item, arranged in the same left-to-right order as they would appear in the array item itself;

MODIFY the final sentence of sub-clause 11.2.2 as follows:

~~To be copied or compared, the type of an aggregate expression shall be equivalent. See 6.22.2.~~
If the two operands of a comparison operator are aggregate expressions, they shall be of equivalent type as defined in **6.22.2**. Assignment compatibility of aggregate expressions is defined in **6.22.3** and, for arrays, **7.6**.