PROPOSAL: Fix misleading comments on queue operators

Relates to: SV Mantis item 520
Applies to: IEEE 1800-2008 draft 4 as modified by SV Mantis 1702

Version 1, 13 Jan 2008

Version 2, 21 Jan 2008

Fixed one oversight (replace comma with colon in queue part-select in 6.24.3); improved formatting of example in 7.11.4

Proposed change

Mantis 1702 (now resolved) has addressed the original concerns of this item by legitimizing unpacked array concatenation. However, there remain some examples in the LRM where unpacked array concatenation is used in assignment to a queue and the associated comments wrongly imply that an operation such as insertion is being performed. This proposal suggests various re-wording and modification of those examples to remove possible confusion or conflict with 1702's definition of queue element reference persistence (see new clause **7.11.3** introduced by the proposal for 1702).

Jonathan Bromley, 13 Jan 2008

MODIFY example in 6.24.3 (top of page 109 of LRM draft 4):

```
p = ... // initialize control packet
stream = {stream, Bits'(p)} // append packet to unpacked queue of bits
stream.push_back(Bits'(p)); // append packet to unpacked queue of Bits

Bits b;
Control q;
q = Control'(stream[0]); // convert stream back to a Control packet
stream = stream[1:$]; // remove packet from stream
stream.pop_front(b);  // get packet (as Bits) from stream
q = Control'(b);      // convert packet bits back to a Control packet
```

MODIFY example in 6.24.3 (last line of page 109 of LRM draft 4):

```
channel = channel[ size, $ ]; // remove packet data from stream
channel = channel[ size : $ ]; // update the stream so it now lacks that packet
```

DELETE the first block of code examples in 7.11.1:

```
int q[$] = { 2, 4, 8 };
int p[$];
int e, pos;
e = q[0]; // read the first (leftmost) item
e = q[$]; // read the last (rightmost) item
q[0] = e; // write the first item
p = q; // read and write entire queue (copy)
q = { q, 6 }; // insert '6' at the end (append 6)
q = { e, q }; // insert 'e' at the beginning (prepend e)
q = q[1:$]; // delete the first (leftmost) item
q = q[0:$-1]; // delete the last (rightmost) item
q = q[1:$ 1]; // delete the first and last items
q = {}; // clear the queue (delete all items)
q = { q[0:pos-1], e, q[pos,$] }; // insert 'e' at position pos
q = { q[0:pos], e, q[pos+1,$] }; // insert 'e' after position pos
```

EDIT reference in sub-clause 7.11.3 as added by Mantis 1702 (not in Draft 4):

As a consequence of this clause, inserting elements in a queue using unpacked array concatenation syntax, as illustrated in the examples in ~~7.11.1~~**7.11.4**, will cause all references to any element of the existing queue to become outdated.

EDIT reference in last sentence of sub-clause 10.10.1 as inserted by Mantis 1702 (not in Draft 4):

Unpacked array concatenation is especially useful for writing values of queue type, as shown in the examples in ~~7.11.1~~**7.11.4**.

ADD new sub-clause 7.11.4:

7.11.4 Updating a queue using assignment and unpacked array concatenation

As described in **7.11**, a queue variable may be updated by assignment. Together with unpacked array concatenation, this offers a flexible alternative to the queue methods described in **7.11.2** when performing operations on a queue variable.

The following examples show queue assignment operations that exhibit behaviors similar to those of queue methods. In each case the resulting value of the queue variable shall be the same as if the queue method had been applied, but any reference to elements of the queue will become outdated by the assignment operation (see **7.11.3**):

```
int q[$] = { 2, 4, 8 };
int e, pos, junk;

//   assignment                    //  method call yielding the
//                                 //  same value in variable q
// ------------------------------ // -------------------------
q = { q, 6 };                     //  q.push_back(6)
q = { e, q };                     //  q.push_front(e)
q = q[1:$];                       //  q.pop_front(junk) or q.delete(0)
q = q[0:$-1];                     //  q.pop_back(junk) or q.delete(q.size-1)
q = { q[0:pos-1], e, q[pos:$] }; //  q.insert(pos, e)
q = { q[0:pos], e, q[pos+1:$] }; //  q.insert(pos+1, e)
```

Some useful operations that cannot be implemented as a single queue method call are illustrated in the following examples. As in the examples above, assignment to the queue variable outdates any reference to its elements.

```
q = q[2:$];   // a new queue lacking the first two items
q = q[1:$-1]; // a new queue lacking the first and last items
q = {};       // clear the queue (delete all items)
```