

Minor Corrections

1.1 Page 204 in Section 17.4

Change

There are certain restrictions on the expressions that can appear in concurrent assertions.

Expressions are allowed to include function calls, but certain semantic restrictions are imposed.

- Functions that appear in expressions may not contain output or **ref** arguments (**constref** are allowed).
- Functions should be automatic (or preserve no state information) and have no side effects.

1.2 Page 207, Section 17.5

Change

Sequences ~~and sequence expressions~~ can be composed by concatenation, analogous to a concatenation of lists. The concatenation specifies a delay, using ##, from the end of the first sequence until the beginning of the second sequence.

1.3 Page 207, Section 17.5

Change

~~The context in which a sequence occurs determines when the sequence is evaluated. The first expression in a sequence is checked at the first occurrence of the clock tick at or after the expression that triggered evaluation of the sequence. Each successive element (if any) in the sequence is checked at the next subsequent occurrence of the clock.~~

1.4 Page 213 in Section 17.7.2

Change

~~An empty sequence shall be illegal.~~

1.5 Page 214 in Section 17.7.2

Change

- ~~If n is 0, then there must be either a prefix, or a suffix concatenation term (i.e., not the only term in the expression) to the repeated sequence~~
- ~~The match shall not be empty~~

1.6 Page 217 in Section 17.7.4

Change

The binary operator **and** is used when both operand expressions are expected to **succeed match**, but the end times of the operand expressions can be different.

The two operands of **and** are sequence expressions. The requirement for the **succeed match** of the **and** operation

is that both the operand expressions must **succeed match**. The operand expressions start at the same time. When one of the operand expressions **succeeds matches**, it waits for the other to **succeed match**. The end time of the composite expression is the end time of the operand expression that completes last.

When `te1` and `te2` are sequences, then the expression:

```
te1 and te2
```

— **Succeeds matches** if `te1` and `te2` **succeed match**.

1.7 Page 217 in Section 17.7.4

Change

The following example is an expression with ~~the and~~ operator **and**, where the two operands are single sequence evaluations.

1.8 Page 218 in Section 17.7.4

Change

3) Figure 17-5 shows the attempt to examine at clock tick 8 when both operand sequences start and **succeed match**. All five sequences for the first operand sequence match, as shown in a time window, at clock ticks 9, 10, 11, 12 and 13 respectively. The second operand sequence matches at clock tick 12.

1.9 Page 219 in Section 17.7.4

Change

If `te1` and `te2` are sampled booleans (not sequences), the expression `(te1 and te2)` **succeeds matches** if `te1` and `te2` are both evaluated to be true.

1.10 Page 219 in Section 17.7.4

Change

The binary operator **intersect** is used when both operand expressions are expected to **succeed match**, and the end

1.11 Page 220 in Section 17.7.4

Change

The two operands of **intersect** are sequence expressions. The requirements for the success of the **intersect** operation are:

— Both the operand expressions must **succeed match**.

1.12 Page 254 in Section 17.12.4

Change

A concurrent assertion statement can be used outside of a procedural context. It can be used within a module as

a *module_common_item*, an interface as a *module_common_item*, or a program as a *non_port_item*. A concurrent assertion statement is either an **assert**, **assume** or a **cover** statement.