11.3.5 Operator expression short circuiting

REPLACE:

The operators shall follow the associativity rules while evaluating an expression as described in 11.3.2. However, if the final result of an expression can be determined early, the entire expression need not be evaluated. This is called *short-circuiting* an expression evaluation.

For example:

```
logic regA, regB, regC, result ;
result = regA & (regB | regC) ;
```

If `regA` is known to be zero, the result of the expression can be determined as zero without evaluating the subexpression `regB | regC`.

WITH:

The operators shall follow the associativity rules while evaluating an expression as described in 11.3.2. ~~However, if the final result of an expression can be determined early, the entire expression need not be evaluated. This is called *short-circuiting* an expression evaluation.~~ Some operators (`&&`, `||`, and `?:`) shall use *short-circuit evaluation*; in other words, some of their operand expressions shall not be evaluated if their value is not required to determine the final value of the operation. The detailed short-circuiting behavior of each of these operators is described in its corresponding section (11.4.7 and 11.4.11). All other operators shall not use short circuit evaluation – all of their operand expressions are always evaluated. When short-circuiting occurs, any side effects or runtime errors that would have occurred due to evaluation of the short-circuited operand expression shall not occur.

For example:

```
logic regA, regB, regC, result ;
function logic myFunc(logic x);
…
endfunction
result = regA & (regB | myFunc(regC)) ;
```

~~If~~ Even if `regA` is known to be zero, ~~the result of the expression can be determined as zero without evaluating the subexpression `regB | regC`~~ the subexpression `(regB | myFunc(regC))` will be evaluated and any side effects caused by calling `myFunc(regC)` will occur.

Note that implementations are free to optimize by omitting evaluation of subexpressions as long as the simulation behavior (including side effects) is as if the standard rules were followed.

11.4.7 Logical operators

ADD NEW PARAGRAPH AT END OF SECTION:

The `&&` and `||` operators shall use short-circuit evaluation as follows. The first operand expression shall always be evaluated. For `&&`, if the first operand value is logically false then the second operand shall not be evaluated. For `||`, if the first operand value is logically true then the second operand shall not be evaluated.

11.4.11 Conditional operator

REPLACE:

If *cond_predicate* is true, the operator returns the value of the first *expression*; if false, it returns the value of the second *expression*.

WITH:

If *cond_predicate* is true, the operator returns the value of the first *expression* without evaluating the second *expression*; if false, it returns the value of the second *expression* without evaluating the first *expression*.