Notes on SystemVerilog LRM Extracts

The following content consists of extracts taken directly from the official Language Reference Manual (LRM) of SystemVerilog. The sections containing text highlighted in red represent the specific content we believe could be replaced with our proposal.

For clarity, the original LRM page number corresponding to the beginning of each extracted section is included in that section's title (this page marker is editorial and is not part of the original LRM text).

11.4.4 Relational operators (p. 278)

[…]

When one or both operands of a relational expression are unsigned, the expression shall be interpreted as a comparison between unsigned values. If the operands are of unequal bit lengths, the smaller operand shall be zero-extended to the size of the larger operand.

When both operands are signed, the expression shall be interpreted as a comparison between signed values. If the operands are of unequal bit lengths, the smaller operand shall be sign-extended to the size of the larger operand. See 11.8.2 for more information.

If either operand is a real operand, then the other operand shall be converted to an equivalent real value and the expression shall be interpreted as a comparison between real values.

[…]

11.4.5 Equality operators (p. 279)

[…]

When one or both operands are unsigned, the expression shall be interpreted as a comparison between unsigned values. If the operands are of unequal bit lengths, the smaller operand shall be zero-extended to the size of the larger operand.

When both operands are signed, the expression shall be interpreted as a comparison between signed values. If the operands are of unequal bit lengths, the smaller operand shall be sign-extended to the size of the larger operand. See 11.8.2 for more information.

If either operand is a real operand, then the other operand shall be converted to an equivalent real value, and the expression shall be interpreted as a comparison between real values.

[…]

11.4.8 Bitwise operators (p. 281)

[…]

For the binary bitwise operators, if one or both operands are unsigned, the result is unsigned. If the operands are of unequal bit lengths, the smaller operand shall be zero-extended to the size of the larger operand.

If both operands are signed, the result is signed. If the operands are of unequal bit lengths, the smaller operand shall be sign-extended to the size of the larger operand. See 11.8.2 for more information.

For the unary bitwise negation operator, if the operand is unsigned, the result is unsigned. If the operand is signed, the result is signed.

[…]

11.6.1 Rules for expression bit lengths (p. 299)

The rules governing the expression bit lengths have been formulated so that most practical situations have a natural solution.

The number of bits of an expression (known as the *size* of the expression) shall be determined by the operands involved in the expression and the context in which the expression is given.

A *self-determined expression* is one where the bit length of the expression is solely determined by the expression itself—for example, an expression representing a delay value.

A *context-determined expression* is one where the bit length of the expression is determined by the bit length of the expression and by the fact that it is part of another expression. For example, the bit size of the right-hand expression of an assignment depends on itself and the size of the left-hand side.

Table 11-21 shows how the form of an expression shall determine the bit lengths of the results of the expression. In Table 11-21, `i`, `j`, and `k` represent expressions of an operand, and `L(i)` represents the bit length of the operand represented by `i`.

Table 11-21—Bit lengths resulting from self-determined expressions

Expression	Bit length	Comments
Unsized constant number	At least 32 bits	
Sized constant number	As given	
`i op j`, where op is: `+ - * / % & | ^ ^~ ~^`	$\max(\mathrm{L}(i), \mathrm{L}(j))$	
`op i`, where op is: `+ - ~ ++ --`	$\mathrm{L}(i)$	
`i op j`, where op is: `=== !== ==? !=? == != > >= < <=`	1 bit	Operands are sized to $\max(\mathrm{L}(i), \mathrm{L}(j))$
`i op j`, where op is: `&& || -> <->`	1 bit	All operands are self-determined
`op i`, where op is: `& ~& | ~| ^ ~^ ^~ !`	1 bit	All operands are self-determined
`i op j`, where op is: `>> << ** >>> <<<`	$\mathrm{L}(i)$	j is self-determined
`i ? j : k`	$\max(\mathrm{L}(j), \mathrm{L}(k))$	i is self-determined
`{i, ..., j}`	$\mathrm{L}(i) + \cdots + \mathrm{L}(j)$	All operands are self-determined
`{i {j, ..., k}}`	$i \times (\mathrm{L}(i) + \cdots + \mathrm{L}(j))$	All operands are self-determined

Multiplication may be performed without losing any overflow bits by assigning the result to something wide enough to hold it.

11.6.2 Example of expression bit-length problem (p. 300)

[...]

11.6.3 Example of self-determined expressions (p. 301)

```
logic [3:0] a;
logic [5:0] b;
logic [15:0] c;
initial begin
   a = 4'hF;
   b = 6'hA;
   $display("a*b=%h", a*b);    // expression size is self-determined
   c = {a**b};                 // expression a**b is self-determined
                               // due to concatenation operator {}
   $display("a**b=%h", c);
   c = a**b;                   // expression size is determined by c
   $display("c=%h", c);
end
```

Simulator output for this example:

```
a*b=16   // 'h96 was truncated to 'h16 since expression size is 6
a**b=1   // expression size is 4 bits (size of a)
c=ac61   // expression size is 16 bits (size of c)
```

11.7 Signed expressions (p. 301)

[…]

11.8.2 Steps for evaluating an expression (p. 302)

The following are the steps for evaluating an expression:

- Determine the expression size based upon the standard rules of expression size determination (see 11.6).
- Determine the sign of the expression using the rules outlined in 11.8.1.
- Propagate the type and size of the expression (or self-determined subexpression) back down to the context-determined operands of the expression. In general, any context-determined operand of an operator shall be the same type and size as the result of the operator. However, there are two exceptions:
  - If the result type of the operator is real and if it has a context-determined operand that is not real, that operand shall be treated as if it were self-determined and then converted to real just before the operator is applied.
  - The relational and equality operators have operands that are neither fully self-determined nor fully context-determined. The operands shall affect each other as if they were context-determined operands with a result type and size (maximum of the two operand sizes) determined from them. However, the actual result type shall always be 1 bit unsigned. The type and size of the operand shall be independent of the rest of the expression and vice versa.
- When propagation reaches a simple operand as defined in 11.5, then that operand shall be converted to the propagated type and size. If the operand shall be extended, then it shall be sign-extended only if the propagated type is signed.

11.8.3 Steps for evaluating an assignment (p. 303)

The following are the steps for evaluating an assignment:

- Determine the size of the right-hand side by the standard assignment size determination rules (see 11.6).
- If needed, extend the size of the right-hand side, performing sign extension if, and only if, the type of the right-hand side is signed.

11.8.4 Handling x and z in signed expressions (p. 303)

[…]