Changes to text in section 8.4 of SystemVerilog LRM version 3.1a

In Verilog, an **`if (`***expression***`)`** is evaluated as a boolean, so that if the result of the expression is 0 or X, the test is considered false.

SystemVerilog adds the keywords **`unique`** and **`priority`**, which can be used before an **`if`**. If either keyword is used, it shall be a run-time ~~error~~ warning for no condition to match unless there is an explicit **`else`**. For example:

```
unique if ((a==0) || (a==1)) $display("0 or 1");
else if (a == 2) $display("2");
else if (a == 4) $display("4"); // values 3,5,6,7 cause ~~an error~~ a run-time warning

priority if (a[2:1]==0) $display("0 or 1");
else if (a[2] == 0) $display("2 or 3");
else $display("4 to 7"); //covers all other possible values, so no ~~error~~ warning
```

A **`unique if`** indicates that there should not be any overlap in a series of **`if`**...**`else`**...**`if`** conditions, i.e. they should be mutually exclusive, allowing the expressions to be evaluated in parallel. A software tool shall issue ~~an error~~ a warning if it determines that more than one condition is, or can be, true. A software tool shall also issue ~~an error~~ a warning if it determines that no condition is true, or it is possible that no condition is true, and the final **`if`** does not have a corresponding **`else`**.

A **`priority if`** indicates that a series of **`if`**...**`else`**...**`if`** conditions shall be evaluated in the order listed. In the preceding example, if the variable a had a value of 0, it would satisfy both the first and second conditions, requiring priority logic. A software tool shall also issue ~~an error~~ a warning if it determines that no condition is true, or it is possible that no condition is true, and the final **`if`** does not have a corresponding **`else`**.

The **`unique`** and **`priority`** keywords apply to the entire series of **`if`**...**`else`**...**`if`** conditions. In the preceding examples it would have been illegal to insert either keyword after any of the occurrences of **`else`**. To nest another if statement within such a series of conditions, a **`begin`**...**`end`** block should be used.

In Verilog, there are three types of case statements, introduced by **`case`**, **`casez`** and **`casex`**. With SystemVerilog, each of these can be qualified by **`priority`** or **`unique`**. A **`priority case`** shall act on the first match only. A **`unique case`** shall check for overlapping case items, allowing the case items to be evaluated in parallel. A **`unique case`** shall issue a warning message if more than one case item matches the case expression. If the case is qualified as **`priority`** or **`unique`**, the simulator shall issue a warning message if no case item matches. These warnings can be issued at either compile time or run time, as soon as it is possible to determine the illegal condition.

Note: by specifying **`unique`** or **`priority`**, it is not necessary to code a **`default`** case to trap unexpected case values. For example:

```
bit [2:0] a;
unique case(a) // values 3,5,6,7 cause a run-time warning
      0,1: $display("0 or 1");
      2: $display("2");
      4: $display("4");
endcase

priority casez(a) // values 4,5,6,7 cause a run-time warning
      3'b00?: $display("0 or 1");
      3'b0??: $display("2 or 3");
endcase
```