17.7.4 Immediate past and future sampled value functions

This subclause describes the system functions available for accessing the immediate past and future values of an expression as sampled by the global clock. *They can only be used when global clocking is defined* (see Editor please insert reference to global clocking). These functions include the capability to access the previous (resp. the next) value as sampled by the global clock tick that precedes (resp. follows) immediately the time at which the function was called. Sampling of an expression is explained in 17.3. The following functions are provided:

Immediate past:

$past_i(expression)
$rose_i(expression)
$fell_i(expression)
$stable_i(expression)
$changed_i(expression)

Immediate future:

$future_i(expression)
$rising_i(expression)
$falling_i(expression)
$steady_i(expression)
$changing_i(expression)

The behavior of the immediate past sampled-value functions can be defined using the sampled-value functions as follows:

```
$past_i(v)     = $past(v, , , @$global_clocking)
$rose_i(v)     = !$past(v, , , @$global_clocking)          &&   $sampled(v)
$fell_i(v)     = $past(v, , , @$global_clocking)           &&   !$sampled(v)
$stable_i(v)   = $past(v, , , @$global_clocking)           ==   $sampled(v)
$changed_i(v)  = $past(expression, , , @$global_clocking)  !=   $sampled(v)
```

The future sampled-value functions are similar except that they use the subsequent value of the expression. `$future_i(v)` is the sampled value of `v` at the next global clocking tick.
The other functions are defined as follows:

```
$rose_i(v)     = !$sampled(v)  &&   $future_i(v)
$fell_i(v)     = $sampled(v)   &&   !$future_i(v)
$stable_i(v)   = $sampled(v)   ==   $future_i(v)
$changed_i(v)  = $sampled(v)   !=   $future_i(v)
```

The use of these functions is limited to assertion features only. It shall be a compilation error to invoke these functions outside of property expressions. The immediate past and future value functions may not be nested.

The main reason for introducing the next value functions is their efficiency in formal verification: the performance of many formal verification tools is better when the next values of the variables are referenced in the assertions, and not the past ones. Also, using the next value functions may sometimes make assertions more intuitive. In formal tools the next value is the sampled value at the next global clocking tick. Therefore, to maintain consistency between simulation and formal tools, when a variable is updated it should happen before the next global clocking tick.

An action block of an assertion containing next value functions is performed at the time moment when all the next values are actually computed, that is, at the global clocking tick that follows the assertion clock tick at which the final boolean expression of the assertion is evaluated.

Example 1:

The Table 17-1 shows the values returned by the next value functions for `sig` at different time moments.

The following assertion states that the signal may change only on rising clock:

```
assert property (@$global_clock $ischanging(sig) |-> $isrising(clk))
   else $error(”sig is not stable”);
```

In the Figure 17-4 we see that this property is violated at time 80.

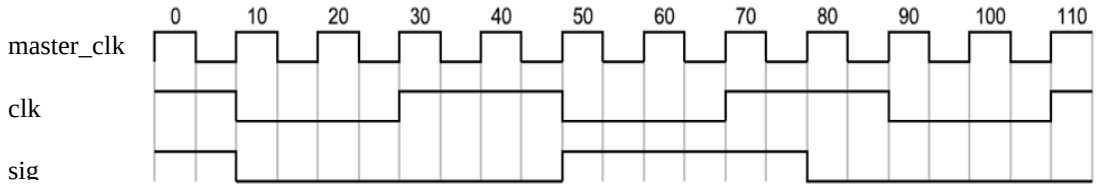

Figure 17-4 Next value change

Table 17-1: Next value functions

Time	sig	$future_i	$rising_i	$falling_i	$changing_i	$steady_i
10	1’b1	1’b0	false	true	true	false
30	1’b0	1’b0	false	false	false	true
50	1’b0	1’b1	true	false	true	false
80	1'b1	1'b0	false	true	true	false

Example 2:

The following assumptions states that a signal sig must remain stable between two rising edges of a clock clk as sampled by global clocking.

```
a: assume property(@$global_clocking
          $rising_i(clk) ##1 (!$rising_i(clk)[*1:$]) |-> $stable(sig) );
```

22.9 Assertion system functions

Note to the editor - add the following text at the end of this sub-clause:

The following functions allow to access the immediate past and future values as sampled by the global clock or detect immediate changes in sampled values of an expression.

Immediate past:

$past_i(expression)

$rose_i(expression)
$fell_i(expression)
$stable_i(expression)
$changed_i(expression)

Immediate future:

$future_i(expression)
$rising_i(expression)
$falling_i(expression)
$steady_i(expression)
$changing_i(expression)

These functions are discussed in 17.7.4.