If the DMA path is true (i.e., *$rose(dma_command) is true, and (dma_req ##[1:3] dma_ack) is true*), but bus_req is false, then the ap_DMA_BusAccess assertion is vacuous; this is because the antecedent of the followed-by is true but its consequent is vacuous.

3.10.4 nexttime, s_nexttime

The (**nexttime**[k] property_expression)family of operators specify an occurrence of a property expression that starts after *n* cycles, and specify the outcome of the results under various conditions including the lack of or existence of *n* cycles and the outcome of the property expression. There are four variations of the *nexttime* operators, two of which are the weak operators (without the prefix "**s_**"), and two of which are the strong operators (with the prefix "**s_**"). Those include:

```
property_expr ::=
  nexttime property_expr    // Weak Nexttime
| nexttime [ constant _expression ] property_expr // Weak Nexttime
| s_nexttime property_expr  // Strong Nexttime
| s_nexttime [ constant_expression ] property_expr  // Strong Nexttime
```

Since the **nexttime** property_expr is equivalent to the **nexttime[1]** property_expr, its description will be implied when addressing the non-indexed **nexttime**.

🕮 **Rule:** *[1] The indexed weak nexttime property* **nexttime** [constant_expression] property_expr *evaluates to true if, and only if, either there are not* constant_expression *clock ticks or* property_expr *evaluates to true beginning at the last of the next* constant_expression *clock ticks*.

🕮 **Rule:** *[1] The indexed strong nexttime property* **s_nexttime** [constant_expression] property_expr *evaluates to true if, and only if, there exist* constant_expression *clock ticks and* property_expr *evaluates to true beginning at the last of the next* constant_expression *clock ticks.*

The **nexttime** property operators are equivalent to the followed-by operator with the antecedent equal to the sequence (**##**k 1'b1). Specifically,

Property	Equivalent property
nexttime [k] **property_expr** *// k is a constant_expression*	((**##**k 1'b1) **#-#** property_expression)

Example:

```
ap_a_cd: assert property(a |-> s_nexttime [2] (c ##3 d) );
```

The above assertion states that if a==1'b1 at cycle *n*, the assertion requires that c==1'b1 two cycles later at cycle *n+2*, and d==1'b1 three cycles after that at cycle *(n+2) + 3*. The "strong" form of **nexttime** (**s_nextttime**) implies that (n+2) and (n+3) cycles must occur, otherwise the assertion fails.

3.10.4.1 Vacuity

Table 3.9.9.1 shows the possible conditions on the number of clocks and the evaluations of the *property_expression* P for **(nexttime [*k*] P)** and **(s_nexttime [*k*] P)**

Table 3.9.9.1 nexttime and s_nextttime evaluations

Number of *k* clocks in simulator and evaluation result of property P	**(nexttime [*k*] P)**	**(s_nexttime [*k*] P)**
* *k* clocks exist and P is true nonvacuously	True Nonvacuous	True Nonvacuous
* *k* clocks exist and P is vacuous	True vacuously	True vacuously
* *k* clocks exist and P is false	False	False
* *k*==0, P is not evaluated to completion	True vacuously	False
* One more clock exists, but less than *k* clocks	True vacuously	False