

- b) Compare the elapsed time to the specified limit;
- c) Report a timing violation if the elapsed time violates the limit.

The skew checks have two different violation detection mechanisms, event-based and timer-based. Event-based skew checking is performed only when a signal transitions, while timer-based skew checking takes place as soon as the simulation time equal to the skew limit has elapsed.

The **\$nochange** check involves three events rather than two.

### 15.3.1 \$skew

The **\$skew** timing check syntax is shown in [Syntax 15-9](#).

```

$skew_timing_check ::= (From Annex A - A.7.5.1)
    $skew ( reference_event , data_event , timing_check_limit [ , [ notifier ] ] ) ;
data_event ::= (From Annex A - A.7.5.2)
    timing_check_event
reference_event ::=
    timing_check_event
timing_check_limit ::=
    expression

```

*Syntax 15-9—Syntax for \$skew*

[Table 15-7](#) defines the **\$skew** timing check.

**Table 15-7—\$skew arguments**

Argument	Description
reference_event	Timestamp event
data_event	Timecheck event
limit	Non-negative constant expression
notifier (optional)	Reg

The **\$skew** timing check reports a violation in the following case:

$$(\text{timecheck time}) - (\text{timestamp time}) > \text{limit}$$

Simultaneous transitions on the reference and data signals ~~can never~~ *shall not* cause **\$skew** to report a timing violation, even when the skew limit value is zero.

The **\$skew** timing check is event-based; it is evaluated only after a data event. If there is never a data event (i.e., the data event is infinitely late), the **\$skew** timing check shall never be evaluated, and no timing violation shall ever be reported. In contrast, the **\$timeskew** and **\$fullskew** checks are timer-based by default, and they ~~shall~~ *should* be used if violation reports are absolutely required and the data event can be very late or even absent altogether. These checks are discussed in [15.3.2](#) and [15.3.3](#).

**\$skew** shall wait indefinitely for the data event once it has detected a reference event and it shall not report a timing violation until the data event takes place. A second consecutive reference event shall cancel the old wait for the data event and begin a new one.

After a reference event, the **\$skew** timing check shall never stop checking data events for a timing violation. **\$skew** shall report timing violations for all data events occurring beyond the limit after a reference event.

### 15.3.2 \$timeskew

The syntax for **\$timeskew** is shown in [Syntax 15-10](#).

```

$timeskew_timing_check ::= (From Annex A - A.7.5.1)
    $timeskew ( reference_event , data_event , timing_check_limit
        [ , [ notifier ] [ , [ event_based_flag ] [ , [ remain_active_flag ] ] ] ] );
data_event ::= (From Annex A - A.7.5.2)
    timing_check_event
event_based_flag ::=
    constant_expression
reference_event ::=
    timing_check_event
remain_active_flag ::=
constant_mintypmax_expression
    constant_expression (change also in Syntax 15-2, Syntax 15-11, A.7.5.2)
timing_check_limit ::=
    expression

```

Syntax 15-10—Syntax for \$timeskew

[Table 15-8](#) defines the **\$timeskew** timing check arguments.

Table 15-8—\$timeskew arguments

Argument	Description
reference_event	Timestamp event
data_event	Timecheck event
limit	Non-negative constant expression
notifier (optional)	Reg
event_based_flag (optional)	Constant expression
remain_active_flag (optional)	Constant expression

The **\$timeskew** timing check reports a violation only in the following cases:

$$(\text{timecheck time}) - (\text{timestamp time}) > \text{limit}$$

Simultaneous transitions on the reference and data signals ~~can never~~ *shall not* cause **\$timeskew** to report a timing violation, even when the skew limit value is zero. **\$timeskew** *shall also not report a violation if a new timestamp event occurs exactly at the expiration of the time limit.*

The default behavior for **\$timeskew** is timer-based. ~~Violations are~~ *A violation shall be* reported immediately upon an elapse of time after the reference event equal to the limit, and the check shall become dormant and report no more violations (even in response to data events) until after the next reference event. *However, if a data event occurs within the limit, then a violation shall not be reported and the check shall become dormant immediately.* This check shall also become dormant if it detects a *conditioned* reference event when its condition is false *and the remain active flag is not set.*

The **\$timeskew** check's default timer-based behavior can be altered to event-based using the ~~event-based flag~~ *event based flag* (Note: the change is to insert underscores. This is for all uses of "event based flag" and "remain active flag"). It behaves like the **\$skew** check when both the ~~event-based flag~~ *event based flag* and the ~~remain active flag~~ *remain active flag* are set. The **\$timeskew** check behaves like the **\$skew** check when only the ~~event-based flag~~ *event based flag* is set, except *that* it becomes dormant after reporting the first violation *or if it detects a conditioned reference event when its condition is false.*

Example:

```
$timeskew (posedge CP &&& MODE, negedge CPN, 50, event based flag,
remain active flag);
```

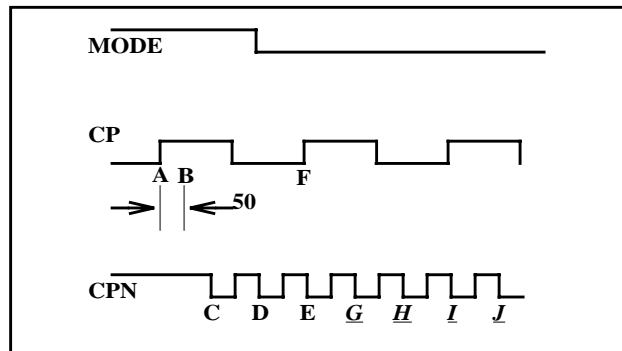


Figure 15-1—Sample **\$timeskew**

Case 1: ~~Event based flag~~ *event based flag* and ~~remain active flag~~ *remain active flag* not set.

After the first reference event on CP at A, a violation is reported at B as soon as 50 time units have passed, *turning the \$timeskew check dormant, and* no further violations are reported.

Case 2: ~~Event based flag~~ *event based flag* set, ~~remain active flag~~ *remain active flag* not set.

*After the first reference event on CP at A, the* The negative transition on CPN at point C ~~shall cause~~ *causes* a timing violation, *turning the \$timeskew check dormant, and no further violations are reported.* Subsequent negative transitions at points D and E do not cause violations. The second reference event at F occurs while MODE is false, *turning so* the **\$timeskew** check *remains* dormant *and no further violations are reported.*

Case 3: Both ~~event based flag~~ *event based flag* and ~~remain active flag~~ *remain active flag* set.

*After the first reference event on CP at A, the* The first three negative transitions on CPN at points C, D and E ~~shall~~ *cause* timing violations. The second reference event at F occurs while MODE is false, *but because the remain active flag is set, the \$timeskew check remains active and so additional violations are reported at*

*G, H, I and J. In other words, all negative transitions on CPN cause violations, which is identical to \$skew behavior. turning the \$timeskew check dormant, and no further violations are reported.*

*Case 4: event based flag not set, remain active flag set.*

*For the waveform depicted in Figure 15-1, \$timeskew has the same behavior in this case as in case 1. The difference between the two cases is illustrated by the following waveform.*

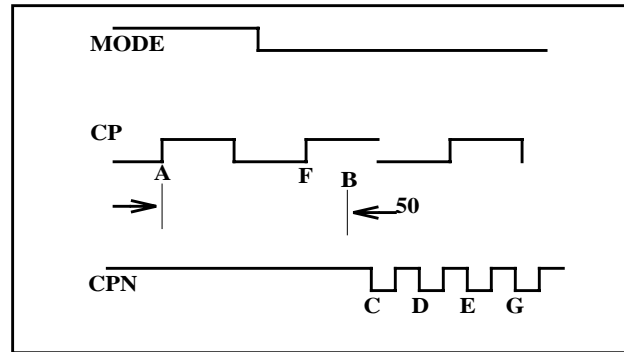


Figure 15-2—Sample \$timeskew with remain\_active\_flag set

*Although the reference event on CP at F occurs while MODE is false, it does not turn the \$timeskew check dormant since the remain\_active\_flag is set. A violation will hence be reported at time B, whereas for case 1, where the remain\_active\_flag is not set, the \$timeskew check would turn dormant at F and no violation would be reported.*

### 15.3.3 \$fullskew

The syntax for \$fullskew is shown in [Syntax 15-11](#).

```

$fullskew_timing_check ::= (From Annex A - A.7.5.1)
    $fullskew ( reference_event , data_event , timing_check_limit , timing_check_limit
        [ , [ notifier ] [ , [ event_based_flag ] [ , [ remain_active_flag ] ] ] ] );
data_event ::= (From Annex A - A.7.5.2)
    timing_check_event
event_based_flag ::=
    constant_expression
reference_event ::=
    timing_check_event
remain_active_flag ::=
    constant_mintypmax_expression
    constant_expression
timing_check_limit ::=
    expression

```

Syntax 15-11—Syntax for \$fullskew

[Table 15-9](#) defines the **\$fullskew** timing check arguments.

**Table 15-9—\$fullskew arguments**

Argument	Description
reference_event	Timestamp or timecheck event
data_event	Timestamp or timecheck event
limit 1	Non-negative constant expression
limit 2	Non-negative constant expression
notifier (optional)	Reg
event_based_flag (optional)	Constant expression
remain_active_flag (optional)	Constant expression

**\$fullskew** is ~~identical~~ *similar* to **\$timeskew**, except *that* the reference and data events can transition in either order. The first limit is the maximum time by which the data event ~~can~~ *should* follow the reference event. The second limit is the maximum time by which the reference event ~~can~~ *should* follow the data event.

The reference event is the timestamp event and the data event is the timecheck event when the reference event precedes the data event. The data event is the timestamp event and the reference event is the timecheck event when the data event precedes the reference event.

The **\$fullskew** timing check reports a violation only in the following case, where limit is set to limit1 when the reference event transitions first, and to limit2 when the data event transitions first:

$$(\text{timecheck time}) - (\text{timestamp time}) > \text{limit}$$

Simultaneous transitions on the reference and data signals shall ~~never not~~ cause **\$fullskew** to report a timing violation, even when the skew limit value is zero. *\$fullskew shall also not report a violation if a new timestamp event occurs exactly at the expiration of the time limit.*

The default behavior for **\$fullskew** is timer-based. Violations shall be reported immediately upon an elapse of time after the timestamp event equal to the limit. It then becomes dormant and reports no more violations, even in response to timecheck events, until after the next timestamp event. This check shall also become dormant if it detects a timestamp event when the associated condition is false.

*The default behavior for \$fullskew is timer-based (event based flag not set). A violation shall be reported immediately upon elapse of the time limit after the timestamp event if a timecheck event does not occur in this time, turning the timing check dormant. However, if a timecheck event does occur within the time limit, then no violation is reported and the timing check turns dormant immediately.*

*A reference event or data event is a timestamp event and starts a new timing window, unless it is a timecheck event occurring within the time limit after a preceding timestamp event, in which case it turns the timing check dormant, as stated above.*

*In the timer-based mode, a second timestamp event that occurs within the time limit starts a new timing window that replaces the first one, unless the second timestamp event has an associated condition whose value is false. In such a case, the behavior of \$fullskew depends on the remain\_active\_flag. If the flag is set, then the second timestamp event is simply ignored. If the flag is not set, and the timing check is active, then the timing check turns dormant.*

The **\$fullskew** check's default timer-based behavior can be altered to event-based using the event-based flag. It behaves like the **\$skew** check when both the event-based flag and the remain-active flag are set. The

**\$timeskew** check behaves like the **\$skew** check when only the event based flag is set, except it becomes dormant after it reports the first violation.

*The **\$fullskew** check's default timer-based behavior can be altered to event-based using the event based flag. In this mode, **\$fullskew** is similar to **\$skew**, in that a violation is reported not upon elapse of the time limit after the timestamp event (as in timer-based mode), but rather if a timecheck event occurs after the time limit. Such an event ends the first timing window and immediately begins a new timing window, where it acts as the timestamp event of the new window. A timecheck event within the time limit ends the timing window, turns the timing check dormant, and no violation is reported.*

*In the event-based mode, a second timestamp event that occurs before a timecheck event has occurred starts a new timing window that replaces the first one, unless the second timestamp event has an associated condition whose value is false. In such a case, the behavior of **\$fullskew** depends on the remain active flag. If the flag is set, then the second timestamp event is simply ignored. If the flag is not set, and the timing check is active, then the timing check turns dormant.*

*In both the timer-based and event-based modes, if the timestamp event has no condition or has a true condition, and the timing check is dormant, then the timing check is activated.*

Example:

```
$fullskew (posedge CP &&& MODE, negedge CPN, 50, 70,, event based flag,
remain active flag);
```

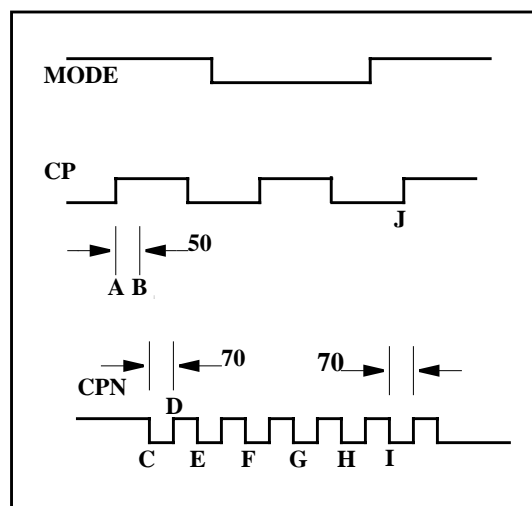


Figure 15-3—Sample **\$fullskew**

Case 1: ~~Event based flag~~ event based flag and ~~remain active flag~~ remain active flag not set.

The transition at A of CP while MODE is true begins a wait for a negative transition on CPN, and a violation is reported at B as soon as a period of time equal to 50 time units has passed. This resets the check and readies it for the next active transition.

A negative transition on CPN occurs next at C, beginning a wait for a positive transition on CP while MODE is true. At D a time equal to 70 time units has passed without a positive edge on CP while MODE is true, so a violation is reported and the check is again reset to await the next active transition.

A transition on CPN at E also results in a timing violation, as does the transition at F, because even though CP transitions, MODE is no longer true. Transitions at G and H also result in timing violations, but not the transition at I, because it is followed by a positive transition on CP while MODE is true.

*Case 2: Event based flag `event_based_flag` set, remain active flag not set.*

The transition at A of CP while MODE is true begins a wait for a negative transition on CPN, and a violation is reported at C on CPN because it occurs beyond the 50 time unit limit. This transition at C also begins a wait of 70 time units for a positive transition on CP while MODE is true. But for transitions on CPN at B through H there is no positive transition on CP while MODE is true, and so no timing violations are reported. The transition at I on CPN begins a wait of 70 time units, and this is satisfied by the positive transition on CP at J while MODE is true.

*Case 3: Event based flag and remain active flag both set.*

The transition at A of CP while MODE is true begins a wait for a negative transition on CPN, and a violation is reported at C on CPN, and it shall also begin a wait for a positive transition on CP while MODE is true. No such transition on CP ever takes place after CPN transitions C through H, but no violations are reported because CP never experiences a positive transition while MODE is true. Transition I also reports no violation because a positive transition at I on CP while MODE is true occurs within the 70 time unit skew limit.

*Although the waveform in this particular example does not show the role of the remain active flag, it should be recognized that this flag has a vital role in determining the behavior of the `$fullskew` timing check, just as it does for the `$timeskew` timing check.*

### 15.3.4 \$width

The `$width` timing check syntax is shown in [Syntax 15-12](#).

```
$width_timing_check ::= (From Annex A - A.7.5.1)
    $width ( controlled_reference_event , timing_check_limit
        [ , threshold [ , notifier ] ] ) ;
controlled_reference_event ::= (From Annex A - A.7.5.2)
    controlled_timing_check_event
threshold ::=
    constant_expression
timing_check_limit ::=
    expression
```

Syntax 15-12—Syntax for `$width`

[Table 15-10](#) defines the `$width` timing check.

Table 15-10—`$width` arguments

Argument	Description
reference_event	Timestamp edge triggered event
(data_event - implicit)	Timecheck edge triggered event
limit	Non-negative constant expression
threshold (optional)	Non-negative constant expression
notifier (optional)	Reg