

## Motivation

The events used in concurrent assertions are edge-sensitive: `@clk` means “changing `clk`”, `@(posedge clk)` means “rising `clk`”, etc. Nevertheless in Annex F describing the formal semantics of assertions the clocks are level-sensitive: `@clk` means when “high `clk`”, `@1` means “each position of the word (trace)”. This proposal is aimed to explain how the SystemVerilog level sensitive events are converted to the level-sensitive clocks.

This proposal is based on Mantis 1682 and is aligned to Draft4.

ADD

### 16.2.1 Clock control

Note to the editor: Please, shift the subsequent clause numbering accordingly.

In this clause (Annex F) the clock controls are considered level-sensitive, and in the `@c` notation `c` is assumed to be a boolean. However, in SystemVerilog the notation `@c` is used to designate an edge-sensitive event control. To describe how level-sensitive event controls are converted to the level sensitive clock controls we introduce operator  $\tau$  defining rewriting rules from an edge-sensitive clock control to a level-sensitive clock control.  $b, b_1, \dots$  denote a boolean expression, and  $e, e_1, \dots$  denote an event expression.

- $\tau(\$global\_clock) = 1$
- $\tau(b) = \$changing\_gclk(b)$ , for  $b \neq \$global\_clock$ , see [reference to global clocking](#).
- $\tau(posedge\ b) = \$rising\_gclk(b)$ , see F.2.3.6
- $\tau(negedge\ b) = \$falling\_gclk(b)$ , see F.2.3.6
- $\tau(e\ \mathbf{iff}\ b) = \tau(e)\ \&\&\ b$
- $\tau(e_1\ \mathbf{or}\ e_2) = \tau(e_1)\ \|\ \tau(e_2)$
- $\tau(e_1, e_2) = \tau(e_1)\ \|\ \tau(e_2)$

For example, the SystemVerilog event control `@(posedge clk)` corresponds to `@($rising_gclk(clk))` in the formal semantics description.