

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

F.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 boolean functions on the input alphabet, and in the `@c` notation `c` is assumed to be a boolean. However, in SystemVerilog the notation `@c` is commonly used to designate a value-change sensitive event control. To describe how value-change sensitive event controls are converted to boolean we introduce operator τ 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
- $\tau(e) = \$future_gclk(b)$, for a named event e (see 15.5), and for a dummy `bit` variable b associated with the event e , such that b has value 1 in the time slots when the event e is triggered, and value 0 in all other time slots.
- $\tau(e\ \mathbf{iff}\ b) = \tau(e) \ \&\&\ b$
- $\tau(e_1\ \mathbf{or}\ e_2) = \tau(e_1) \ ||\ \tau(e_2)$
- $\tau(e_1, e_2) = \tau(e_1) \ ||\ \tau(e_2)$

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