

ADD

### 17.14.6 Inferred value functions

The following elaboration time system functions are available to query the inferred value of the clock event, reset and enable signals.

- `$inferred_clock` returns the value of the inferred clock event.
- `$inferred_disable` returns the value of the inferred disable expression.
- `$inferred_enable` returns the value of the inferred enable condition.

Inferred clock corresponds to the current resolved event expression that can be used in a clocking event definition. It is obtained by applying clock flow rules to the point where `$inferred_clock` is called. If there is no inferred event expression, `$inferred_clock` returns *false* (i.e., 1'b0).

Inferred disable expression is either the reset condition inferred from an always block with asynchronous reset or the **default disable** value (See 17.14 **Note to editor: This is after introducing the proposal #1648**).

Inferred enable condition returned by `$inferred_enable` is the inferred condition from an always or initial block if one exists, the same as the condition inferred in verification statements placed in an **if...else** or a **case** statement (see 17.13.5). It is *true* (i.e., 1'b1) when no condition can be inferred.

All inferred value functions are lazy, i.e., their evaluation is deferred until the latest possible time. It means that if the inferred value function is invoked within the body of a property expression or within a sequence expression, it is not evaluated there, but only when instantiated in an assertion because only then the clock flow rules can be applied. When an inference system function is used as the default value on a formal argument to a property or sequence, it is replaced by the inferred value at the point where the property or sequence is instantiated.

Consider the following example:

```
module m(logic a, b, c, d, rst1, clk1, clk2);

    default clocking @(negedge clk1); endclocking
    default disable rst1;

    property p_triggers(start_event, end_event, form, clk = $inferred_clock,
        rst = $inferred_disable, en = $inferred_enable);
        @clk disable iff (rst)
        en throughout (start_event ##0 end_event[->1]) | => form;
    endproperty

    property p_multiclock(clkw, clkx = $inferred_clock, clky, w, x, y, z);
        @clkw w ##1 @clkx x | => @clky y ##1 @$inferred_clock z;
    endproperty

    a1: assert property (p_triggers(a, b, c));
    a2: assert property (p_triggers(a, b, c, posedge clk1, '0) );

    always @(posedge clk2 or posedge rst) begin
        if (rst) ... ;
        else if (d)
            a3: assert property (p_triggers(a, b, c));
    end
end
```

```

a4: assert property(p_multiclock(negedge clk2, , posedge clk1, a, b, c, d);

endmodule

```

The above code is equivalent to

```

module m(logic a, b, c, d, rst1, clk1, clk2);

a1: assert property (@(negedge clk1) disable iff (rst1)
                    1'b1 throughout a ##0 b[->1] /=> c);
a2: assert property (@(posedge clk1) disable iff (1'b0)
                    1'b1 throughout a ##0 b[->1] /=> c) ;

always @(posedge clk2) begin
  a3: assert property
      (@(posedge clk2) disable iff (rst) d throughout (a ##0 b[->]) /=> c) ;
end

a4: assert property (@(negedge clk2) a ##1 @(negedge clk1) b |=>
                    @(posedge clk1) c ##1 @(posedge clk1) d);

endmodule

```

In assertion a1 the clock event is inferred from the default clocking, therefore `$inferred_clock` is `negedge clk1` for a1. In assertion a2 the clock event `posedge clk1` is passed explicitly to the property `p_triggers`, therefore no default value is inferred for it. In assertion a3 the clock event is inferred from the control event of the `always` block, therefore `$inferred_clock` is `posedge clk2` for a3.

The disable condition `rst1` is inferred for assertion a1 from the `default disable` statement, and the disable condition `rst` is inferred for assertion a3 from the `always` block. Assertion a2 uses explicit reset value '0 in which case the `disable iff` statement could be omitted altogether.

The inferred enabling condition for assertions a1 and a2 is '1, since they are not in the scope of any conditional statement; therefore the `throughout` operator could also be omitted for these assertions. The inferred enabling condition for assertion a3 is d.

In assertion a4, as the property `p_multiclock` is instantiated in the `assert property` statement, `clkw` is replaced by the actual argument (`negedge clk2`), `clkx` by the default argument value `$inferred_clk` which at the property instance position is the default clocking clock (`negedge clk1`). The third clock, `clky`, is replaced by the actual argument (`posedge clk1`) because it is explicitly specified. The fourth clocking event gets the value of `$inferred_clock` which is (`posedge clk1`) as determined by the clock flow rules at this position in the assertion, that is, by the preceding clocking event specification in the multiclock sequence of the consequent of the implication.