Proposal : immed_assume directive (Enhancement #2)

The immed_assume directive is to be used to instruct verification tool or environment to constrain the design under test. The expression in the immed_assume directive is non-temporal such that only boolean expression is allowed and it should hold for all times. Although the verification tool is not obligated to prove the validity of each individual immed_assume statement, it should provide a means to report or identify the errors where there are conflicting immed_assume expressions. Such conflicting expressions associated with immed_assume directives are not the valid expressions.

```
BNF:
  immed_assume_statement :: =
     immed_assume expression;
  expression ::= bool_expression
```

Proposal : parameter value assignment (Enhancement #7)

The support of parameter value assignment benefits users in many ways. The most important one is the ease for commonly-used library creation. The supported features are shown as follows:

- Allow parameter value passing by position as well as by name
- If parameter value passing is by name, actual parameters can be in any order in the parameter list and parameters can be omitted. The actual parameter value is preceded by “para_id=”.
- If parameter value passing is by position, the order of actual parameters needs to be the same as that of the formal parameters and actual parameters can only be skipped with respect to the following rules. There are two alternatives to skip a subset of parameters with this method. First, in the parameter declaration, this subset of parameters to be skipped shall be preceded by the remaining parameters. Secondly, a parameter can be skipped with the replacement of “,”.
- Allow to pass int constant, boolean, clk_event, sequence and etc.

```
BNF:

property_declaration::=

  property property_id [property_formal_list] ;
        { parameter_declaration }
        { assertion_variable_declaration }
        property_spec;
  endproperty [:property_id]
```

```
  parameter_declaration ::=
    parameter_list;

  parameter_list ::= parameter_statement {; parameter_statement }
  parameter_statement ::= int_para
                        | bool_para
                        | clock_event_para
                        | sequence_para;
  int_para ::= parameter int para_id = int_value;
  bool_para ::= parameter bool para_id = bool_value;
  clock_event_para ::= parameter clock_event para_id = clock_event_kind;
  clock_event_kind ::= posedge | negedge | edge;
  sequence_para ::= parameter sequence para_id = sequence_exp;
```

Example:

```
property follow(en, clk, leader, follower);
  parameter int min_lat = 0;
  parameter int max_lat = 0;
  parameter clock_event sample_clock_edge = posedge;

  @(sample_clock_edge ck) en |->
         (‘true [*0:$] ##1 leader ##[min_lat:max_lat] follower) [*1:$] ;
endproperty

// SVA follow instance
follow_negedge :
  assert  property ( #(max_lat=5,min_lat=3,clock_event=negedge )
                                 follow(en, clk,leader,follower)
                          )
                    ;
                    else $display("%m follower reponses late");
```