Objectives

The goal of this proposal is to introduce case statements inside assertions , for example:

```
property p1(logic a,b);
  s(a,b) |=>
   case (val)
     3'b011: a ##1 b;
     3'b100: a |=> b;
     default: a ##2 b;
   endcase
endproperty
```

Change log:

Addressing comments from JH (12-Feb-08):

1. Added property_expr ::= property_statement, this would allow code such that in the example above (in the objective) without semicolon after the endcase. Adding this however basically says that you are free to add semicolon at the ends of property_expr or property_statement but you don't have to put them after case or if-else (so "if (b) p;;; else q" is legal), this is backward compatible but also allows inseting semicolons inside if-else.
2. Operator precedence for "case" isn't defined, this is consistent with the procedural statements operator precedence.

Addressing comments from DB (13-Feb-08):

1. At F.2.3.5: the ≡ symbol is missing

Addressing comments from DK (18-Feb-08):

1. removed : The case expression given in parentheses shall be evaluated exactly once and before any of the case item statements. From 16.12.17 since it seems to be related only to expression with side effects.

2. Fix fonts in h) and in 16.5.1 on page 4.- made "*(b!==b1 && b!==b2)" Italic in "h)", and keywords to Courier /Italic/9 in 16.5.1*

3. changed h) on page 4 to refer to expression_ instead of b_

18-Feb-08: shifted "16.12.17 case" to be before "16.12.16 recursive properties" in 1932 by note to editor

Addressing comments from BT (18-Feb-08):

1. "change isn't backwards compatible in terms of VPI" – made changes such that original VPI diagram is kept and add support for case operator in the property expr VPI diagram, the proposal now doesn't depend on more extension to VPI.

Addressing comments from BT (24-Feb-08):

1. add #define vpiCaseProperty <editor to fill in> and #define vpiCasePropertyItem <....> -- Note also rename it "case property item" instead ...

Addressing comments from DK (24-Feb-08):

1. Syntactical meta-symbols [] and {} are shown in new productions in black–made them blue
2. Page 4—" . The set of semantic leading clocks of case (b) b1:q1 b2:q2 endcase is {inherited}." The statement should have general form with n clauses and optional default.
3. F.2.3.5 Other derived operators (last page)
   I would prefer to provide a recursive definition:
   "(case (b) b1: P1 … bn: Pn endcase) ≡
   (if (b===b1) P1 … else if (b===bn) Pn)" -->
   "(case (b) b1: P1 b2: P1 … bn: Pn endcase) ≡
   (if (b===b1) P1 else case (b) b2: P2 … bn: Pn endcase)", e.t.c.
4. note fonts, subscripts are sometimes rendered as regular fonts.

Addressing comments from SB (10-Mar-08):

1. "I don't think the BNF footnote is needed. Regular cases have the same restriction with such a BNF footnote."

Addressing comments from JH (12-Mar-08):

1. extra semicolon in assertion statement –A possible solution is to have both property_spec and property_statement_spec, where the former is unchanged, i.e.

   property_spec ::= [clocking_event] [disable iff (expression_or_dist)] property_expr

   and the latter requires a property_statement in place of the property_expr.
   Then use property_spec in the assertion statements (unchanged) and use property_statement_spec in the property declaration, getting rid of the explicit semicolon there.
2. additional comments on 13-Mar-08:
   a. change the definition of vacuity to include n case items with optional default
   b. change in the multi clocks section (16.15.1) the default to be optional.

The proposal has been aligned to P1800-2008draft4 – assuming 1932 has passed and been implemented (The entire proposal is conditional by 1932 passing)

16.12 Declaring properties

REPLACE in Syntax 16-14

property_declaration ::=
property property_identifier [**(** [property_port_list] **)**] **;**
{ assertion_variable_declaration }
property_spec **;**
endproperty [**:** property_identifier]

property_expr ::=
…
| **if (** expression_or_dist **)** property_expr [**else** property_expr]
…

WITH

property_declaration ::=
property property_identifier [**(** [property_port_list] **)**] **;**
{ assertion_variable_declaration }
~~property_spec ;~~ property_statement_spec
endproperty [**:** property_identifier]

property_statement_spec ::=
[clocking_event] [**disable iff (** expression_or_dist **)**] property_statement

property_statement ::=
property_expr **;**
| **case (** expression_or_dist **)** property_case_item { property_case_item } **endcase**
| **if (** expression_or_dist **)** property_expr [**else** property_expr]

property_case_item::=
expression_or_dist { **,** expression_or_dist } **:** property_statement
| **default** [**:**] property_statement

property_expr ::=
…
~~| **if (** expression_or_dist **)** property_expr [**else** property_expr]~~
| property_statement
…

Add**:** (Note to editor: This clause numbering is taken from 1932, please insert before "16.12.16 Recursive properties" and shift the numbers accordingly

16.12.16 case

The *case* property statement is a multiway decision that tests whether a boolean expression matches one of a number of other boolean expressions and branches accordingly.

property_statement ::= *// from A.2.10*
...
| **case (** expression_or_dist **)** property_case_item { property_case_item } **endcase**
…

property_case_item::=
expression_or_dist { **,** expression_or_dist } **:** property_statement
| **default** [**:**] property_statement

Syntax 16-17—property statement case syntax (excerpt from Annex A)

(Note to editor – please shift number of tables accordingly)
The *default* statement shall be optional. Use of multiple default statements in one property case statement shall be illegal.

A simple example of the use of the case property statement is the decoding of variable `wait` to produce a delay between the check of two signals as follows:

```
property p_wait(logic [1:0] wait);
  case (wait)
    2'd0: a && b;
    2'd1: a ##2 b;
    2'd2: a ##4 b;
    2'd3: a ##8 b;
    default: 0; // cause a failure if wait has x or z values
  endcase
endproperty
```

The case item expressions shall be evaluated and compared in the exact order in which they are given. If there is a default case item, it is ignored during this linear search. During the linear search, if one of the case item expressions matches the case expression given in parentheses, then the property statement associated with that case item shall be evaluated, and the linear search shall terminate. If all comparisons fail and the default item is given, then the default item property statement shall be executed. If the default property statement is not given and all of the comparisons fail, then none of the case item property statements shall be evaluated and the evaluation of the case property statement from that start point succeeds and returns true (vacuously).

16.14.7 Non-vacuous evaluations

REPLACE

g) An evaluation attempt of an instance of a property is non-vacuous iff the underlying evaluation attempt of the *property_expr* that results from substituting actual arguments for formal arguments is non-vacuous.

WITH

g) An evaluation attempt of an instance of a property is non-vacuous iff the underlying evaluation attempt of the *property_expr* that results from substituting actual arguments for formal arguments is non-vacuous.

h) An evaluation attempt of a property of the form

case (*expression_or_dist*)
  expression_or_dist$_1$:*property_stmt*$_1$
  …
  expression_or_dist$_n$: *property_stmt*$_n$
  [**default** : *property_stmt*$_d$]
endcase

is non-vacuous iff either *(expression_or_dist===expression_or_dist*$_1$*)* and the underlying evaluation attempt of *property_stmt*$_1$ is non-vacuous, or *(expression_or_dist!== expression_or_dist*$_1$ *&& expression_or_dist=== expression_or_dist*$_2$*)* and the underlying evaluation attempt of *property_stmt*$_2$ is non-vacuous, or …, or the default is present and *(expression_or_dist!==expression_or_dist*$_1$ *&& expression_or_dist!==expression_or_dist*$_2$ *&& … && expression_or_dist!==expression_or_dist*$_n$*)* and the underlying evaluation of *property_stmt*$_d$ is non-vacuous.

16.15.1 Clock resolution in multiclocked properties

REPLACE

— The set of semantic leading clocks of **if** (*b*) *q1* else *q2* is {*inherited*}.

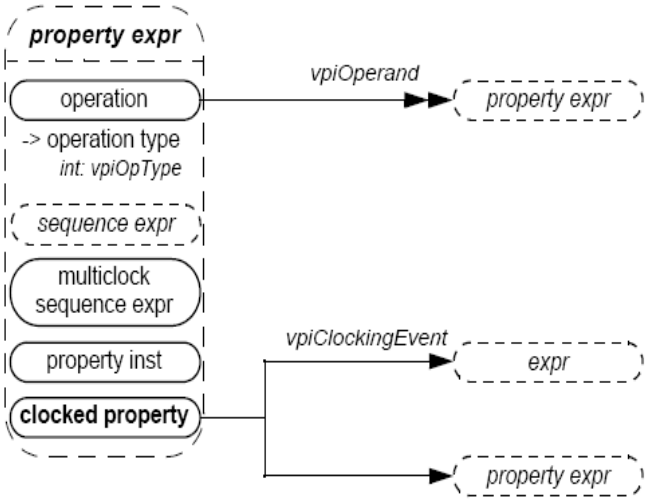
WITH

— The set of semantic leading clocks of **if** (*b*) *q1* else *q2* is {*inherited*}.

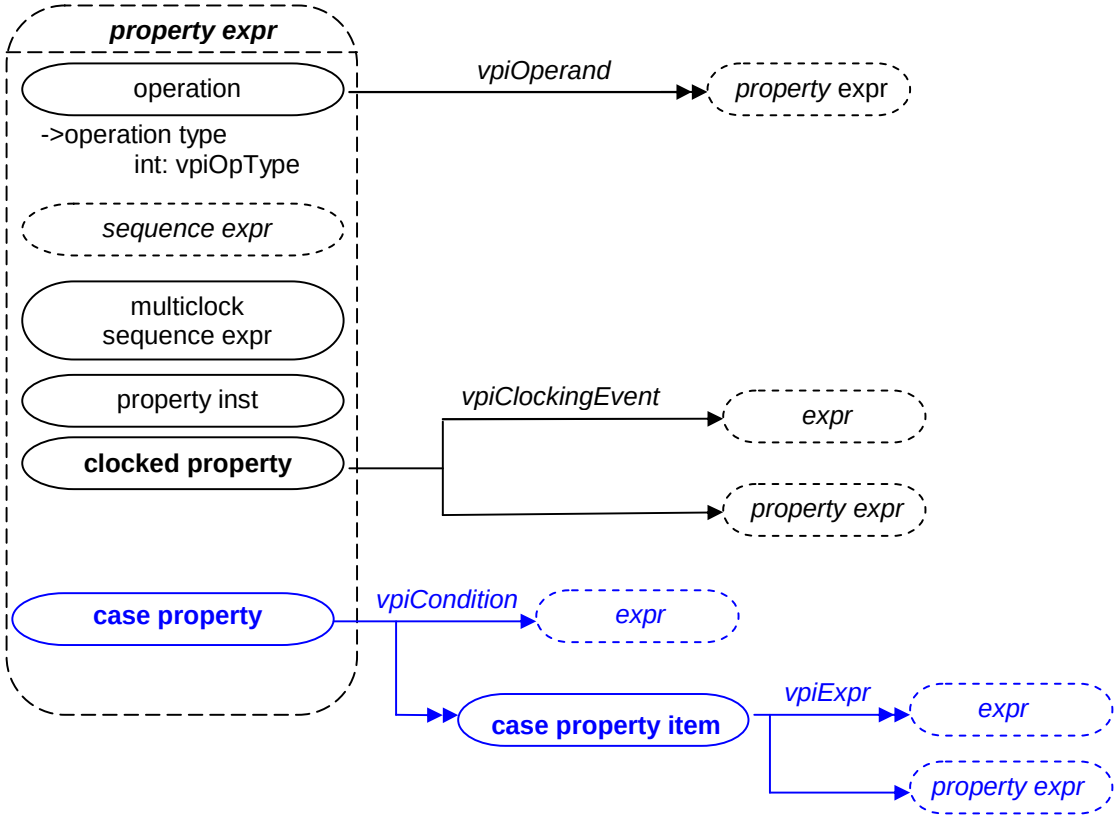
— The set of semantic leading clocks of **case** (*b*) *b*$_1$:*q*$_1$ … *b*$_n$:*q*$_n$ [**default:** *q*$_d$] **endcase** is {*inherited*}.

36.45 Property specification

CHANGE:



TO:



Add to "Details:"

3) The case property item shall group all case conditions that branch to the same property statement.

4) **vpi_iterate()** shall return NULL for the default case item because there is no expression with the default case.

A.2.10 Assertion declarations

REPLACE

property_declaration ::=
property property_identifier [**(** [property_port_list] **)**] **;**
{ assertion_variable_declaration }
property_spec **;**
endproperty [**:** property_identifier]

property_expr ::=
…
| **if (** expression_or_dist **)** property_expr [**else** property_expr]
…

WITH

property_declaration ::=
property property_identifier [**(** [property_port_list] **)**] **;**
{ assertion_variable_declaration }
~~property_spec ;~~ property_statement_spec
endproperty [**:** property_identifier]

property_statement_spec ::=
[clocking_event] [**disable iff (** expression_or_dist **)**] property_statement

property_statement ::=
property_expr **;**
| **case (** expression_or_dist **)** property_case_item { property_case_item } **endcase**
| **if (** expression_or_dist **)** property_expr [**else** property_expr]

property_case_item::=
expression_or_dist { **,** expression_or_dist } **:** property_statement
| **default** [**:**] property_statement

property_expr ::=
…
~~| **if (** expression_or_dist **)** property_expr [**else** property_expr]~~
| property_statement
…

N.2 Source code

REPLACE

```
#define vpiPropertyInst 660
#define vpiSequenceDecl 661
```

WITH

```
#define vpiPropertyInst 660
#define vpiSequenceDecl 661
#define vpiCaseProperty editor to fill  /* property case */
#define vpiCasePropertyItem editor to fill  /* property case item */
```

F.2.3.5 Other derived operators

REPLACE

— (**if** (*b*) *P*1 **else** *P*2) ≡ ((*b* |-> *P*1) **and** (!*b* |-> *P*2))

WITH

— (**if** (*b*) *P*1 **else** *P*2) ≡ ((*b* |-> *P*1) **and** (!*b* |-> *P*2))

— (**case** (*b*) b_1: P_1 **endcase**) ≡ (if (*b*===b_1) P_1)

— (**case** (*b*) **default:** P_d **endcase**) ≡ (P_d)

— (**case** (*b*) b_1: P_1 **default:** P_d **endcase**) ≡ (if (*b*===b_1) P_1 **else** P_d)

— (**case** (*b*) b_1: P_1 … b_n: P_n **endcase**) ≡ (if (*b*===b_1) P_1 **else case** (*b*) b_2: P_2 **…** b_n: P_n **endcase**)

— (**case** (*b*) b_1: P_1 … b_n: P_n **default:** P_d **endcase**) ≡ (if (*b*===b_1) P_1 **else case** (*b*) b_2: P_2 **…** b_n: P_n **default:** P_d **endcase**)