SVA Quick Reference

Product Version: IUS 11.1
Release Date: December 2011

This quick reference describes the SystemVerilog Assertion constructs supported by Cadence Design Systems. For more information about SystemVerilog Assertions, see the Assertion Writing Guide.

Note: Numbers in parentheses indicate the section in the IEEE 1800-2005 Standard for SystemVerilog for the given construct.

## Binding

bind target bind_obj [ (params) ] bind_inst (ports);

(17.15) Attaches a SystemVerilog module or interface to a Verilog module or interface instance, or to a VHDL entity/architecture. Multiple targets supported. Example:

bind fifo fifo_full v3(clk,empty,full);
bind top.dut.fifo1 fifo_full v3(clk,empty,full);

endsequence

endproperty

## Immediate Assertions

[ label: ] assert boolean_expr [ action_block ];

(17.2) Tests an expression when the statement is executed in the procedural code. Example:

enable_set_during_read_op_only : assert (@(posedge clk) rst |-> P1);

endproperty

property P1;
property P6 (bit AA, BB='true, EN=1);

endproperty

[ identifier: ] ( ( argument_list ) )

(17.11) Creates an instance of a property declaration. Example:

property P1; @(event) a &= b #!1 !a & !b;
endproperty

property P2; @(posedge clk) rst |-> P1;
endproperty

## Declarations

sequence identifier [ argument_list ];

sequence_expr [ seq_op sequence_expr ] ... ;

endsquence [ : identifier ];

(17.6) Declares a sequence expression that can be used in property declarations. Local variables are permitted. Example:

sequence RunSeq (bit Req=0, bit Ack=0); Req #[@13] Ack;
endsquence

property identifier [ argument_list ];

[ clock_expr ] [ disable_clause ] property_expr;

endproperty [ : identifier ];

(17.11) Declares a condition or sequence to be verified during simulation. Local variables are permitted. Example:

property P6 (bit AA, BB='true, EN=1);

@ (posedge clk)
EN |-> (BB ||1 c) |-> (AA &&[1:2] (d||AA));
endproperty

[ identifier: ] ( ( argument_list ) )

(17.11) Creates an instance of a property declaration. Example:

property P1; @(event) a &= b #!1 !a & !b;
endproperty

property P2; @(posedge clk) rst |-> P1;
endproperty

## Clock Expressions

@ ( ( {posedge | negedge} clock | expression );

(17.14) Declares an event or event expression to use for sampling assertion variable values. Multiple clocks (17.12), and clocks inferred from an always block containing only assertions, are supported. Examples:

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

assert property (@ (posedge clk) (a #1 b) |=> @ (posedge clk2) (c #1 d) );
always @ (posedge clk) begin
assert property (@(a #1 b) |=> (c #1 d) ); assert property (@[a][3]) |=> ~c );
cover property (@(a #1 b #1 c) |=> (d[2:4] ) );

## Default Clocking Blocks

default clocking [ clk_identifier ]

(17.14) Specifies the clock or event that controls property evaluation. Example:

default clocking master_clk @ (posedge clk);

property p4; (a |-> #2 b); endproperty

assert property (p4);
endclocking

## Disable Clause

disable iff ( boolean_expr )
default disable iff ( boolean_expr )

(17.11) Specifies a reset expression. Checking of the property is terminated asynchronously when the expression is true. Example:

property P4; @ (negedge clk) disable iff (rst)
(c) |-> (#(max-1:$) d));
endproperty

## Property Expressions

sequence_expr |-> property_expr

(17.12) The property expression must be true in the last cycle that the sequence expression is true (overlapping). Example:

property property P5 (AA);
@ (negedge clk)
disable iff (rst)
(c) |-> (#(max-1:$) d));
endproperty

property_expr and property_expr

(17.11) Returns true if both property expressions are true. Example:

0(c) v |-> (w #1 (d x) ) and (y #1 z)

not property_expr

(17.11) Returns the opposite of the value returned by the property_expr. Example:

property abcd
@ (posedge clk) a |-> not (b #1 c #1 d));
endproperty

if ( expression ) expression_expr1 [ else expression_expr2 ]

(17.11) If expression is true, property_expr1 must hold; property_expr1 does not need to hold when expression is false. If expression is false, property_expr2 must hold, if it exists. Example:

property P2; @ (negedge clk) if (a b) else d |-> c;
endproperty

## Sequence Operators

sequence_expr1 and sequence_expr2

(17.7.4) Both sequences must occur, but the end times of the operands can be different. Example:

(a #2 b) and (c #2 d #2 e) ;

[ property ]

sequence property ( prop_expr ) [ action_block ];

(17.13.1) Checks a property during verification. Example:

property P5 (AA);
@ (negedge clk) (b #1 c) |=> (AA ||1[1:2] (d||AA));
endproperty

[ label: ] assume property ( prop_expr ) [ action_block ];

(17.13.2) Constrains the inputs considered for the property during verification. In simulation, treated like assert. Example:

A1: assume (@ (em^n) rst);

[ label: ] cover property ( prop_expr ) [ pass_statement ];

(17.11) Specifies a reset expression. Checking of the property is terminated asynchronously when the expression is true. Example:

property P4; @ (negedge clk)
disable iff (rst)
(c) |-> (#(max-1:$) d));
endproperty

[ label: ] cover property ( seq_expr ) [ pass_statement ];

(17.13.3) Monitors the property or sequence for coverage and reports statistics. The statement is executed when the property succeeds. Cover property reports all matches. Example:

C1: cover property (@ (event) a |-> b # (2:5) c);

## Expect Statement

expect ( prop_expr ) [ action_block ];

(17.16) Blocks the current process until the property succeeds or fails. Example:

expect( @(posedge clk) #[@1:10] top.TX_Monitor_data == value ) success = 1;
else success = 0;

## Disable Clause

disable iff ( boolean_expr )
default disable iff ( boolean_expr )

(17.11) Specifies a reset expression. Checking of the property is terminated asynchronously when the expression is true. Example:

property P4; @ (negedge clk) disable iff (rst)
(c) |-> (#(max-1:$) d));
endproperty

## Property Expressions

sequence_expr |-> property_expr

(17.12) The property expression must be true in the last cycle that the sequence expression is true (overlapping). Example:

property property P5 (AA);
@ (negedge clk)
disable iff (rst)
(c) |-> (#(max-1:$) d));
endproperty

property_expr and property_expr

(17.11) Returns true if both property expressions are true. Example:

0(c) v |-> (w #1 (d x) ) and (y #1 z)

not property_expr

(17.11) Returns the opposite of the value returned by the property_expr. Example:

property abcd
@ (posedge clk) a |-> not (b #1 c #1 d));
endproperty

if ( expression ) expression_expr1 [ else expression_expr2 ]

(17.11) If expression is true, property_expr1 must hold; property_expr1 does not need to hold when expression is false. If expression is false, property_expr2 must hold, if it exists. Example:

property P2; @ (negedge clk) if (a b) else d |-> c;
endproperty

## Sequence Operators

sequence_expr1 and sequence_expr2

(17.7.4) Both sequences must occur, but the end times of the operands can be different. Example:

(a #2 b) and (c #2 d #2 e) ;
**Sequence Operators (cont’d)**

```verilog
first_match (sequence_expr, seq_match_item)
```

*(17.7.7)* Evaluation of one or more sequences stops when the first match is found. Example:

```verilog
sequence s1;
first_match(a #1 b[1:0] => c)
endsequence
```

**sequence_expr1 intersect sequence_expr2**

*(17.5.7)* Both sequences must occur, and the start and end times of the expression must be the same. Example:

```verilog
(a #2 b) intersect (c #2 d #2 e)
```

**sequence_expr1 or sequence_expr2**

*(17.6.7)* At least one of the sequences must occur. Example:

```verilog
(b #1 c) | (d[#1:2] #1 e) | (f[#1:2]
```

**boolean_expr throughout sequence_expr**

*(17.7.8)* A condition must hold true for the duration of a sequence. Example:

```verilog
(a #2 b) throughout read_sequence
```

**sequence_expr1 within sequence_expr2**

*(17.7.9)* The seq_expr1 must match at some point within the timeframe of sequence_expr2. Example:

```verilog
(a #2 b #3 c) within write_enable
```

**Cycle Delays**

```verilog
##integral_number
##identifier
#(constant_expression)
#(constant_expression)
#(constant_expression)
```

*(17.5)* Specifies the number of clock ticks from the current clock tick until the next specified behavior occurs. Example:

```verilog
property p5 (AA);
@posedge clk (b #1 c) => (AA #1[1:2] (d||AA));
endproperty
```

**Local Variables in Sequences and Properties**

*(17.8, 17.9)* The seq_match_item is executed when seq_expression is matched. The match item can be a subroutine call. Example:

```verilog
sequence data_check:
  int x;
  a #1 {a, x=data_in} #1 b[0:0]
  #1 b & (data_out=x);
endsequence
```

**Sequence Methods**

```verilog
sequence_instance.
```

*(17.12.6)* Identifies the endpoint of a sequence. Example:

```verilog
wait (AB.triggered) || BC.triggered);
```

**Assertion Severity Tasks**

```verilog
$fatal ([0112.] message [args])
```

*(17.2)* Fatal message task; messages can be strings or expressions. You can call this task from the action block of an assertion. Example:

```verilog
$fatal (0);
```

```verilog
$error (message [args])
```

*(17.2)* Error message task; messages can be strings or expressions. You can call this task from the action block of an assertion. Example:

```verilog
$error("a == %s, b == %s", test.inst.a, test.inst.b);
```

```verilog
$warning (message [args])
```

*(17.2)* Warning message task; messages can be strings or expressions. You can call this task from the action block of an assertion. Example:

```verilog
$warning("a == %s, b == %s", test.inst.a, test.inst.b);
```

**System Functions**

```verilog
$onehot (bit_vector)
```

*(17.10)* Returns true if and only one bit of the expression is high. Example:

```verilog
property p4 (Arg)
  @posedge clk $onehot(Arg);
endproperty
```

```verilog
$onehot0 (bit_vector)
```

*(17.10)* Returns true if no more than one bit of the expression is high. Example:

```verilog
property p5 (Arg)
  @posedge clk $onehot0(Arg);
endproperty
```

```verilog
$sisunknown (bit_vector)
```

*(17.10)* Returns true if any bit of the expression is X or Z. Example:

```verilog
property p6 (Arg)
  @posedge clk $sisunknown(Arg);
endproperty
```

**Assertion-Control System Tasks**

```verilog
$sassertoff ([levels [list_of_mods_or_assns]]);
$sasserton ([levels [list_of_mods_or_assns]]);
$sassertkill ([levels [list_of_mods_or_assns]]);
```

*(22.6)* Controls assertion checking during simulation. Example:

```verilog
$sassertoff (0, top.mod1, top.mod2.net1);
```