

## 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 v1(clk,empty,full);
bind top.dut.fifo1 fifo_full v2(clk,empty,full);
bind fifo:fifo1,fifo2 fifo_full v3(clk,empty,full);
```

### 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
(state >= `start_read && state <= `finish_read);
else $warning("Enable set when state => %b",
state);
```

### Declarations

**sequence identifier [ argument\_list ] ;**  
**sequence\_expr [ seq\_op sequence\_expr ] ... ;**  
**endsequence [ : identifier ]**

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

```
sequence BusReq (bit REQ=0, bit ACK=0);
REQ ##[1:3] ACK;
endsequence
```

**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);
@(negedge 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
```

### Directives

**[ label : ] assert property (prop\_expr) [ action\_block ] ;**

(17.13.1) Checks a property during verification. Example:

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

**[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 (@(ena) !rst);
```

**[label:] cover property (prop\_expr) [ pass\_statement ] ;**  
**[label:] cover sequence (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 sequence 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;
```

### 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);
endproperty
```

```
assert property ( @(posedge clk1) (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]) );
end
```

### Default Clocking Blocks

**default clocking [ clk\_identifier ]**  
**{ identifier | clk\_expression } ;**  
**clocking\_items**

**end clocking**  
**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**

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

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

**sequence\_expr | => property\_expr**

(17.11.2) The property expression must be true in the first cycle after the sequence expression is true. Example:

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

**property\_expr and property\_expr**

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

```
@(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) property\_expr1 [ else property\_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 | => c;
else
d | => e;
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)****first\_match** (*sequence\_expr* [, *seq\_match\_item*])

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

```
sequence s1;
  first_match(a ##1 b[->1]:N) ## c);
endsequence
```

**sequence\_expr1 intersect sequence\_expr2**

(17.7.5) Both sequences must occur, and the start and end times of the sequence expressions must be the same. Example:

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

**sequence\_expr1 or sequence\_expr2**

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

```
(b ##1 c) or (d[*1:2] ##1 e) or f[*2]
```

**boolean\_expr throughout sequence\_expr**

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

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

**sequence\_expr1 within sequence\_expr2**

(17.7.9) *sequence\_expr1* must match at some point within the timeframe of *sequence\_expr2*. Example:

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

**Sequence Methods****sequence\_instance**. [*ended* | *matched* | *triggered*]

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

```
wait (AB.triggered) || BC.triggered);
  if (AB.triggered) $display("AB triggered");
```

**Cycle Delays****##integral\_number****##Identifier****##(constant\_expression)****##[const\_expr : const\_expr]****##[const\_expr : \$]**

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

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

**Local Variables in Sequences and Properties****(seq\_expression {, seq\_match\_item}) [repetition\_op]**

(17.8, 17.9) The *seq\_match\_item* is executed when *seq\_expression* is matched. The match item can be a subroutine call. Example:

```
sequence data_check;
  int x;
  a ##1 (!a, x=data_in) ##1 !b[*0:$]
  ##1 b && (data_out=x);
endsequence
```

**Repetition****[\* const\_or\_range\_expression ]**

(17.7.2) Consecutive repetition. Example:

```
(a[*2] ##2 b[*2]) | => (d)
```

**[-> const\_or\_range\_expression ]**

(17.7.2) Goto repetition. Example:

```
a ##1 b[->5] ##1 c
```

**[= const\_or\_range\_expression ]**

(17.7.2) Non-consecutive repetition. Example:

```
s1 | => (b [=5] ##1 c)
```

**Shortcuts**

R[\*] is the same as R[\*0:\$]

##[\*] is the same as ##[0:\$]

R[+] is the same as R[\*1:\$]

##[+] is the same as ##[1:\$]

**Assertion Severity Tasks****\$fatal** ([ 0 | 1 | 2, ] *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:

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

**\$error** (*message* [, *args* ] ) ;**\$warning** (*message* [, *args* ] ) ;**\$info** (*message* [, *args* ] ) ;

(17.2) Non-fatal message tasks; messages can be strings or expressions. You can call these tasks from the action block of an assertion. Example:

```
$error("Unsupported memory task command %b",
  m_task);
$warning("Enable is set during non-read op:
  state=>%b", state);
```

**System Functions****\$onehot** (*bit\_vector*)

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

```
property p1(Arg)
  @(posedge clk) $onehot(Arg);
endproperty
```

**\$onehot0** (*bit\_vector*)

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

```
property p2(Arg)
  @(posedge clk) $onehot0(Arg);
endproperty
```

**\$isunknown** (*bit\_vector*)

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

```
property p3(Arg)
  @(posedge clk) $isunknown(Arg);
endproperty
```

**\$countones** (*bit\_vector*)

(17.10) Returns the number of bits in a vector that have the value 1. Example:

```
property p4(Arg)
  @(posedge clk) $countones(Arg) == 4;
endproperty
```

**Sampled-Value Functions****\$sampled**(*expression*)

(17.7.3) Returns the sampled value of the expression at the current clock cycle. Example:

```
property propA
  @(posedge clk) (a ##1 b);
endproperty
p1: assert (propA)
  $display("%m passed");
else $warning("a == %s; b == %s",
  $sampled(test.inst.a),
  $sampled(test.inst.b));
```

**\$rose**(*expression*)

(17.7.3) Returns true if the sampled value of *expression* changed to 1 during the current clock cycle. Example:

Example:

```
(a ##1 b) |-> $rose(test.inst.sig4);
```

**\$fell**(*expression*)

(17.7.3) Returns true if the sampled value of *expression* changed to 0 during the current clock cycle. Example:

```
(a ##1 b) |-> $fell(test.inst.c);
```

**\$stable**(*expression*)

(17.7.3) Returns true if the sampled value of *expression* remained the same during the current clock cycle. Example:

```
(a ##1 b) |-> $stable(test.inst.c);
```

**\$past**(*expression* [, *n\_cycles*])

(17.7.3) Returns the sampled value of *expression* at the previous clock cycle or the specified number of clock ticks in the past. Example:

```
(a == $past(test.inst.c, 5))
```

**Assertion-Control System Tasks****\$assertoff** [ ( *levels* [, *list\_of\_mods\_or\_assns* ] ) ] ;**\$asserton** [ ( *levels* [, *list\_of\_mods\_or\_assns* ] ) ] ;**\$assertkill** [ ( *levels* [, *list\_of\_mods\_or\_assns* ] ) ] ;

(22.8) Controls assertion checking during simulation. Example:

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