



**The SystemVerilog  
Assertions (SVA)  
Reference Guide**

# Tutorial: Building a Simple Assertion

---

This tutorial explains how to construct a simple (i.e. non-sequential) concurrent SystemVerilog Assertion.

## Boolean Expression

Construct a SystemVerilog boolean expression (i.e. any expression allowed in an `if` statement condition) to define the design behavior. Terminate the expression with a semi-colon:-

```
!(RD_EN && WR_EN);
```

i.e. `RD_EN` and `WR_EN` cannot be simultaneously high.

Normal HDL visibility and case-sensitivity rules apply and any subprograms; overloaded functions; out-of-module references etc. allowed in a SystemVerilog boolean expression can be used.

## Clocking Expression

A clocking expression *must* be defined to control when the assertion is checked. The clocking expression is defined with the clock operator `@` and is placed *before* the boolean expression:-

```
@(posedge CLK) !(RD_EN && WR_EN);
```

i.e. `RD_EN` and `WR_EN` cannot be simultaneously high on the rising edge of `CLK`

Any RTL clock expression format is supported.

## Completing the Property

Enclose the clocking and boolean expressions in `property`. `endproperty` and name the property:-

```
property P1;  
    @(posedge CLK) !(RD_EN && WR_EN);  
endproperty
```

## Assert the Property

Assert the named property to tell the simulator to start checking the assertion. The property name must be enclosed in brackets and labelling the assert statement is recommended.

```
A1 : assert property (P1);
```

# Tutorial: Building a Simple Assertion

## Conditional Properties

Properties can be made conditional by using implication operators ( $|=>$ ,  $|->$ ) in the boolean expression.  $|->$  is the same cycle implication operator:-

```
property P2;
  @(posedge CLK)
  REQ && ~ACK |-> BUSY;
endproperty
A2 : assert property (P2);
```

i.e. if REQ is high and ACK is low, then BUSY must be high in the *same* evaluation cycle (at the same rising edge of CLK).

Next cycle implication uses  $|=>$ :-

```
property P3;
  @(posedge CLK)
  REQ && ~ACK |=> BUSY;
endproperty
A3 : assert property (P3);
```

i.e. if REQ is high and ACK is low, then BUSY must be high at the *next* evaluation cycle (at the next rising edge of CLK).

In both cases, REQ && ~ACK is the *enabling* condition (or antecedent), BUSY is the *fulfilling* condition (or consequent). If the enabling condition is true, then the fulfilling condition must also be true for the property to pass.

Note that a simple boolean property using same cycle implication can usually be expressed as a single, unconditional boolean condition. For example, for property P2 above, the unconditional form is:-

```
property P2_UNCON;
  @(posedge CLK)
  (REQ && ~ACK && BUSY) | !(REQ && ~ACK);
endproperty
A2 : assert property (P2_UNCON);
```

However next cycle conditions and multi-cycle conditions (sequences) are easier to describe with implication operators.

## Tutorial: Sequences

---

Sequences consist of multiple boolean expressions joined together over several evaluation cycles to define a multi-cycle design behaviour.

Sequence are constructed using the cycle delay operator (##N).

```
@(posedge clk) req ##1 ack ##1 busy
```

This defines the sequence of req true on the first rising edge of clk, followed by ack true in the second, followed by busy true in the next cycle. The clock operator defines the evaluation cycles for the sequence.

Sequences can be separately declared which allows them to be used as building blocks in other sequences.

```
sequence S1;  
    B ##1 C ##1 D;  
endsequence  
sequence S2;  
    S1 ##1 J ##1 K;  
endsequence
```

In the code fragment above, the sequence S2 is equivalent to the sequence:-

```
B ##1 C ##1 D ##1 J ##1 K
```

It is recommended to name sequences for two reasons.

1. You can identify the sequence when managing assertions in your verification tool.
2. The sequence can be used as a building block in another sequence, property, endpoint or verification directive.

Note: a named sequence is normally used as a building block of a property.