



→ OPENVERA

# LRM: Assertions Update for Portability

Bruce Greene



# Agenda

- **Methodology**
- **LRM Update Overview**
- **Language Changes**
- **New Syntax**
- **Migration Steps**
- **Conclusion**

## What is Assertion-Based Verification?

- **Powerful Verification Methodology of using Assertions across a broad range of verification tools**

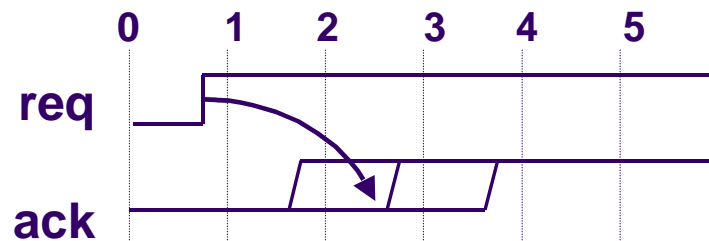
**Simulation -- Coverage -- Testbench – Formal -- IP**

**Higher verification efficiency for finding more bugs**

## What is an Assertion?

A concise description of complex behavior:

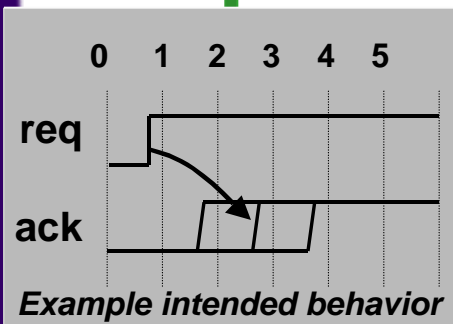
“After request is asserted, acknowledge must come 1 to 3 cycles later”



```
// OpenVera Assertions (OVA) Code for Protocol:  
clock posedge clk {  
  event: req_cycle: if (posedge req) then #[1..3] posedge ack;  
}
```

# Assertion Languages are superior to HDL

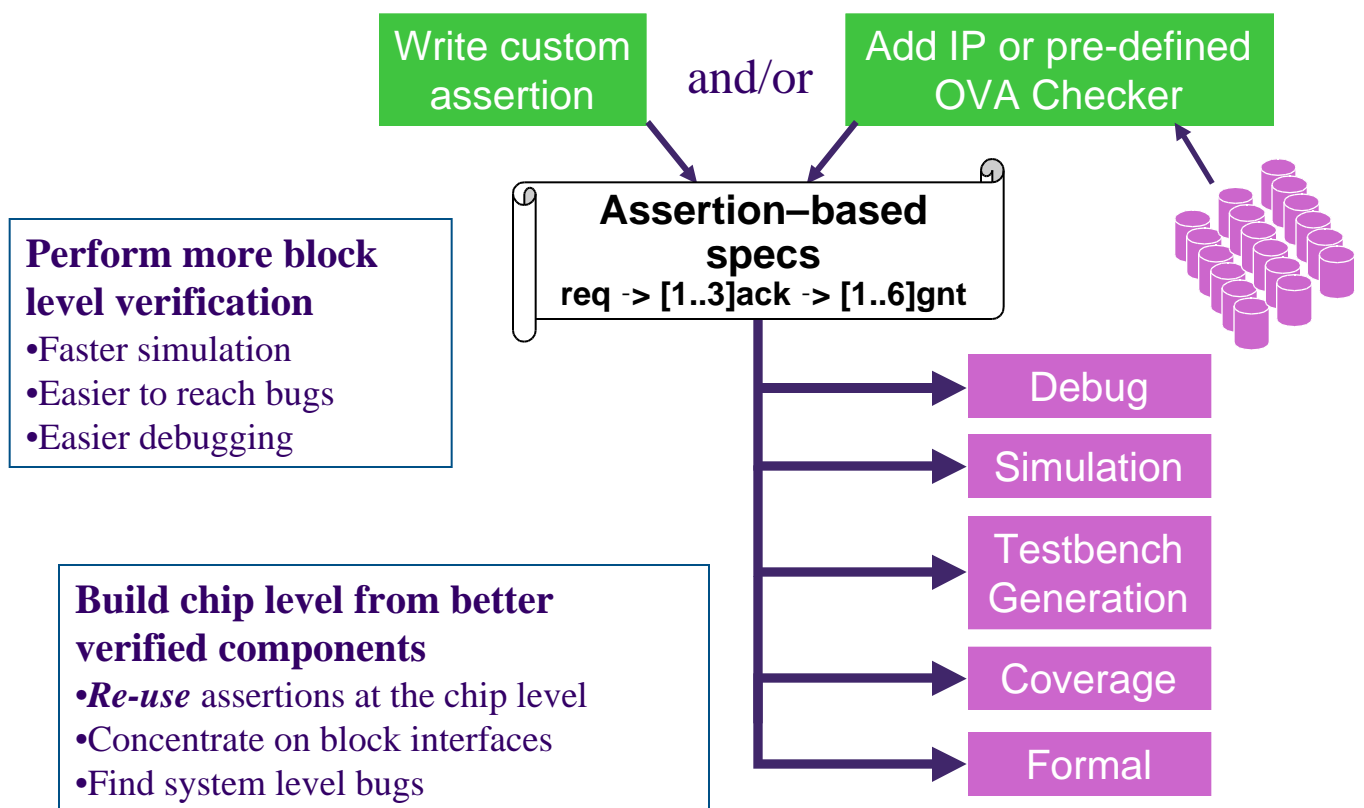
Easy to capture and debug designer's intent / assumptions



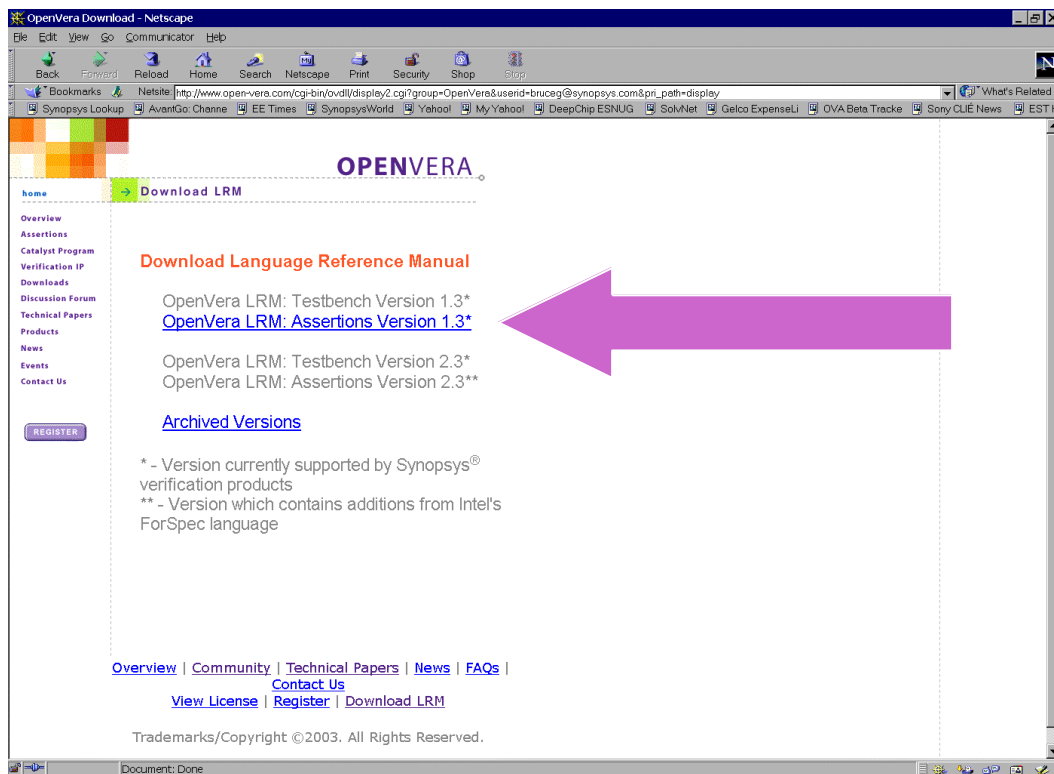
```
clock posedge clk {  
  event req_cycle: posedge req #[1..3] posedge ack;  
}
```

```
sample_inputs : process (clk)  
begin  
  if rising_edge(clk) then  
    STROBE_REQ <= REQ;  
    STROBE_ACK <= ACK;  
  end if;  
end process;  
protocol: process  
  variable CYCLE_CNT : Natural;  
begin  
  loop  
    wait until rising_edge(CLK);  
    exit when (STROBE_REQ = '0') and (REQ = '1');  
  end loop;  
  CYCLE_CNT := 0;  
  loop  
    wait until rising_edge(CLK);  
    CYCLE_CNT := CYCLE_CNT + 1;  
    exit when ((STROBE_ACK = '0') and (ACK = '1')) or (CYCLE_CNT = 3);  
  end loop;  
  if ((STROBE_ACK = '0') and (ACK = '1')) then  
    report "Assertion success" severity Note;  
  else  
    report "Assertion failure" severity Error;  
  end if;  
end process protocol;
```

# Assertions-Based Flow

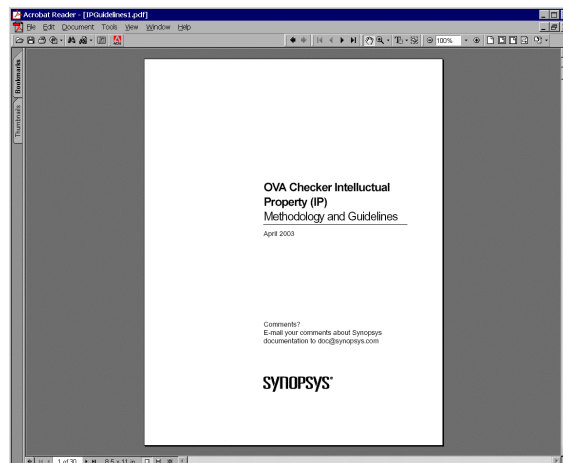


# OpenVera Assertions v1.3 LRM Update



# OVA IP Methodology and Guidelines

- Available on <http://open-vera.com> under
  - technical papers section



## Motivation

- **To support consistent usage of OVA across various tools**
- **Current syntax prevents one from uniformly**
  - **supporting 3rd party simulators**
  - **supporting VHDL**
  - **supporting IP**
- **Core Language Syntax remains unaffected**

# OVA Gaining Momentum

- **OVA supported in:**
  - **Synopsys Products**
    - VCS™, VCS-MX, VERA®
  - **EDA tools suppliers**
    - @HDL, Aldec, Axis Systems, Novas, Valiosys
  - **Verification IP providers**
    - Cold Spring Engineering, Netsys, Silicon Interfaces
  - **More coming soon...**

# Language Changes in a Nutshell

- **“Scope”, “Module” obsolete**
  - replaced with “unit” and unit bindings
  - bind module, bind instances
- **Units will now be typed**
  - allows for better interface specification
  - XMR will be allowed only when “unit” instantiated
- **Note: All core language constructs do not change!**

# Unit Definition

Name of unit

parameters

```
unit my_checker
  #(parameter string msg = "assertion triggered")
  (logic a, logic b);
...
...
endunit
```

port list

# Bindings

- **Module based bindings**

```
bind module module_name : unit_instance;
```

- **Instance based bindings**

```
bind instances instance1, ..., instanceN : unit_instance;
```

# Example

```
bind module top : my_checker inst1 (tb.u1.u2.clk ,en,data);
```

Verilog Module Name

OVA Unit Name

```
bind instances tb.u1,tb.u2 : my_checker inst1 (tb.u1.u2.clk,en,data);
```

Verilog Instance Names

OVA Unit Name

## Migration Steps – Custom OVA

- 1. Change module/scope to “unit” with typed ports**
- 2. Add “bind” construct**
  - 1. bind module (replacement for module)**
  - 2. bind instances (replacement for scope)**
- 3. Move any XMR’s to “bind” construct**

## Migration Steps - example

```
module top {  
  clock posedge tb.u1.u2.clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
}
```

## Migration Steps – module -> unit

```
module top {  
  clock posedge tb.u1.u2.clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
}
```

```
unit my_checker ...  
  clock posedge tb.u1.u2.clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
endunit
```

## Migration Steps – add types

```
module top {  
  clock posedge tb.u1.u2.clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
}
```

```
unit my_checker (logic clk, logic enable , logic [7:0]  
data);  
  clock posedge tb.u1.u2.clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
endunit
```

## Migration Steps – remove XMR's

```
module top {  
  clock posedge tb.u1.u2.clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
}
```

```
unit my_checker (logic clk, logic enable , logic [7:0]  
data);  
  clock posedge clk {  
    event e_range : if (enable) then #1 ((data<=max) ;  
  }  
  assert a_range : check(e_range);  
endunit
```

## Migration Steps – add bindings

```
module top {  
    clock posedge tb.u1.u2.clk {  
        event e_range : if (enable) then #1 ((data<=max) ;  
    }  
    assert a_range : check(e_range);  
}
```

```
unit my_checker (logic clk, logic enable , logic [7:0]  
data);  
    clock posedge clk {  
        event e_range : if (enable) then #1 ((data<=max) ;  
    }  
    assert a_range : check(e_range);  
endunit  
  
bind module top : my_checker inst1 (tb.u1.u2.clk ,en,data);
```

## Migration Steps – Template Based

- 1. Create unit “wrapper” around template code with parameter/signal list**
- 2. Add “edge\_select” macro**
- 3. Call template within unit**

## Create Unit Wrapper

```
template one_hot(en=1,clk,state,strict=0, msg = "ONE_HOT") : {  
  clock clk {  
    event ova_e_one_hot :  
      if (en) then  
        ((count(state) == 1) || (!strict && (count(state) == 0)));  
      }  
    assert ova_c_one_hot : check( ova_e_one_hot, msg );  
  }  
}
```

```
unit ova_one_hot
```

```
endunit
```

## Add Parameters

```
template one_hot(en=1,clk,state,strict=0, msg = "ONE_HOT") : {  
  clock clk {  
    event ova_e_one_hot :  
      if (en) then  
        ((count(state) == 1) || (!strict && (count(state) == 0)));  
      }  
    assert ova_c_one_hot : check( ova_e_one_hot, msg );  
  }  
}
```

```
unit ova_one_hot #(  
  parameter integer strict=0, bw=2, edge_expr=0;  
  parameter string msg = "assertion triggered")  
  
endunit
```

## Add Port List

```
template one_hot(en=1,clk,state,strict=0, msg = "ONE_HOT") : {
    clock clk {
        event ova_e_one_hot :
            if (en) then
                ((count(state) == 1) || (!strict && (count(state) == 0)));
            }
        assert ova_c_one_hot : check( ova_e_one_hot, msg );
    }
}
```

```
unit ova_one_hot #(
    parameter integer strict=0, bw=2, edge_expr=0;
    parameter string msg = "assertion triggered")
    (logic en,logic clk, logic [bw-1:0] state);

endunit
```

## Add edge\_select macro

```
template one_hot(en=1,clk,state,strict=0, msg = "ONE_HOT") : {
  clock clk {
    event ova_e_one_hot :
      if (en) then
        ((count(state) == 1) || (!strict && (count(state) == 0)));
      }
    assert ova_c_one_hot : check( ova_e_one_hot, msg );
  }
}
```

```
unit ova_one_hot #(
  parameter integer strict=0, bw=2, edge_expr=0;
  parameter string msg = "assertion triggered")
  (logic en,logic clk, logic [bw-1:0] state);
  edge_select(edge_expr, clk);

endunit
```

# Call template

```
template one_hot(en=1,clk,state,strict=0, msg = "ONE_HOT") : {  
    clock clk {  
        event ova_e_one_hot :  
            if (en) then  
                ((count(state) == 1) || (!strict && (count(state) == 0)));  
            }  
        assert ova_c_one_hot : check( ova_e_one_hot, msg );  
    }  
}
```

```
unit ova_one_hot #(  
    parameter integer strict=0, bw=2, edge_expr=0;  
    parameter string msg = "assertion triggered")  
    (logic en,logic clk, logic [bw-1:0] state);  
    edge_select(edge_expr, clk);  
    one_hot(en, clk_edge, state, strict, msg);  
endunit
```

## Summary

- **Greater portability with LRM changes**
- **Core language constructs unaffected**
- **Portability across simulators using OVASim**
- **OVA gaining momentum among customers, EDA vendors, and IP providers**