

# Assertions, A Verification Technique for VHDL-based FPGAs

Mary Harris

JHU/APL

Mary.E.Harris@jhuap.edu



MARLUG - Mid-Atlantic Region Local Users Group  
ANNUAL CONFERENCE - OCTOBER 5, 2004  
Johns Hopkins University Applied Physics Lab – Laurel, MD

# What Are Assertions?

- ✓ **Statement about design's intended behavior**
  - **Flag conditions requiring observation or intervention**
  - **Precise mathematical expression**
  - **Non-synthesizable**



# Why Use Them?

- ✓ **Ease debugging process**
  - **Improve observability**
    - ✓ **Detect deviant behavior**
    - ✓ **Isolate erroneous code**
    - ✓ **Inform designer of error condition**
- ✓ **Always analyzing design**
  - ✓ **Change in test vectors or design**
- ✓ **Good source of unambiguous documentation**
  - ✓ **Declare assumptions about how design is supposed to work**
- ✓ **Supported by static & dynamic verification tools**
  - ✓ **Entry into Formal Verification tools**

# Syntax

## v VHDL

**Label: ASSERT <expression of expected behavior>**

**Report <string of text to be written to transcript>**

**Severity <note, warning, error, fatal>;**

## v Example

**Assert count\_value <= max\_count**

**Report “counter has exceeded its maximum value”**

**Severity error;**

# Where Are Assertions Used?

- **Test benches**
- **Behavioral models**
  - √ **Memories**
    - Setup, hold, pulsewidth
  - √ **Interface Protocols**
- **Synthesizable RTL models**
  - √ **Reusable design elements, IP**
    - establish correct usage



# Real-world Examples

- ✓ **Message header contents**
- ✓ **Illegal FSM states**
- ✓ **Signal sequence conforms to protocol**
- ✓ **3-bit count 1 to 6 counter**
  - **detect count values of 0 or 7**

# Types

- ✓ **User-defined**
- ✓ **Proprietary**
  - **HVL-specific**
- ✓ **Standardized**
  - **Accellera Open Verification Library (OVL)**



# Reference Materials

- ✓ **Assertion-Based Design**  
**Harry Foster, Adam Krolnik, David Lacey**  
**Kluwer Academic Publishers**
- ✓ **EE Times Articles from 2003, 2004**
- ✓ **Verification Guild**  
— <http://verificationguild.com/>



# Do Assertions Work?

- ✓ **From “Assertion-Based Design”**
  - **34% of all bugs were found by assertions on DEC Alpha 21164 project**
  - **25% of all bugs were found by assertions on Cyrix M3(p2) project**
  - **85% of all bugs were found using OVL assertions on HP**



# Overhead

- √ **From “Assertion-Based Design”**
  - **1% - 3% increase to design coding phase**
  - **~ 10% increase in simulation time**



# OVL Assertions

- ✓ **Prewritten/tested**
- ✓ **Common assertion tests**
- ✓ **VHDL & Verilog**
- ✓ **Support formal verification tools**
- ✓ **Download from website**
  - <http://www.eda.org/ovl/download.html>

# OVL Assertions

**assert\_always**

**assert\_always\_on\_edge**

**assert\_change**

**assert\_cycle\_sequence**

**assert\_decrement**

**assert\_delta**

**assert\_even\_parity**

**assert\_fifo\_index**

**assert\_frame**

**assert\_handshake**

**assert\_implication**

**assert\_increment**

**assert\_never**

**assert\_next**

**assert\_no\_overflow**

**assert\_no\_transition**

**assert\_no\_underflow**

**assert\_odd\_parity**

**assert\_one\_cold**

**assert\_one\_hot**

**assert\_proposition**

**assert\_quiescent\_state**

**assert\_range**

**assert\_time**

**assert\_transition**

**assert\_unchange**

**assert\_width**

**assert\_win\_change**

**assert\_win\_unchange**

**assert\_window**

**assert\_zero\_one\_hot**



# Assert\_always

## v Assert\_always

The `assert_always` assertion continuously monitors the *test\_expr* at every positive edge of the triggering event or clock *clk*. It contends that a specified *test\_expr* will always evaluate TRUE. If *test\_expr* evaluates to FALSE, an assertion will fire (that is, an error condition will be detected in the code). The *test\_expr* can be any valid Verilog or VHDL expression (depending on the library you are using).

## v Syntax

```
assert_always [#(severity_level, options, msg)] inst_name  
(clk, reset_n, test_expr);
```

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.Numeric_Std.ALL;
USE work.ovl_assertlib.ALL;
ENTITY assert_always IS
GENERIC (severity_level: INTEGER := 0;
         options: INTEGER := 0;
         msg: STRING := "ASSERT
ALWAYS VIOLATION");
PORT (clk, reset_n, test_expr: IN
      std_ulogic);
END assert_always;

ARCHITECTURE ovl OF assert_always IS
SIGNAL valid: std_ulogic := '1';
SIGNAL rst_n: std_ulogic;
BEGIN
-- synopsys translate_off
-- // synopsys template
-- // ovl assertion_library
--
ASSERT valid = '1' REPORT msg
      SEVERITY ovlSevTab(severity_level);
--

```

```

--
rst_n <= ovl_reset_n when
      (ovl_reset_n_enable = '1') else
reset_n;
--
PROCESS
BEGIN
WAIT UNTIL clk'EVENT AND clk =
      '1';
valid <= '1';
IF (rst_n = '1') THEN
      IF (test_expr = '0') THEN
            valid <= '0';
      ELSE
            valid <= '1';
      END IF;
ELSE
      valid <= '1';
END IF;

END PROCESS;
-- synopsys translate_on
END ovl;

```



# Assertion Instantiation

```
Target_behaving <= '1' when DEVSEL# = '1' and
                    TRDY# = '1'           else
                    '0' when DEVSEL# = '1' and
                    TRDY# = '0'           else
                    '1';
```

```
Inverted_FRAME# <= not FRAME#;
```

```
Error_condition1: entity ovl_assertlib.assert_always
  GENERIC MAP (1,0,"Illegal PCI Target Condition")
  PORT MAP    (clk => pci_clk,
               reset_n => inverted_FRAME#,
               test_expr => Target_behaving);
```



# Assertion Best Practices

- ✓ **Capture design specification requirements**
- ✓ **Add while creating RTL model**
- ✓ **Formalize design intent**
- ✓ **Continually add them throughout design phase**
- ✓ **Analyze non-assertion based failures for new opportunities**
- ✓ **Co-locate with RTL code**



# Assertion Best Practices

- ✓ **Include in all IP**
- ✓ **Name all assert statements**
- ✓ **Incorporate method to disable all/some**
- ✓ **Consistent style to prevent synthesis errors**
- ✓ **Create library of tested assertions**
  - **Peer review**

**More suggestions in “Assertion-Based Design”**

# Synplicity work-around

```
-- synthesis translate_off
```

```
  Assert ()
```

```
  Report “”
```

```
  Severity fatal;
```

```
-- synthesis translate_on
```

# Summary

- ✓ **Assertions that are introduced at design conception offer the greatest potential to reduce the design verification cycle time.**
- ✓ **The benefits of using assertions in circuit design is proportional to the complexity of the design**
  - **Size of code**
  - **Number of people involved in design**
  - **Reusability**



