Beacon-VHDL-2008

Comprehensive Test Suite for VHDL-2008 Compliance

Key Advantages

- Backed by Interra's field-proven expertise in developing HDL-based test suites for VHDL and Verilog
- Developed in partnership with significant EDA majors
- Conforming to accepted definition and interpretation of the language
- Providing an unbiased quality analysis of EDA tools
- Some partial and some complete coverage of each of new construct and styles
- Comprehensive validation of syntax, synthesizability, and simulation semantics

Highlights

- Over 2500 test cases along with test benches
- Golden output for comparison
- Detailed test plans with cross-reference to test cases
- Well organized test cases highlighting testing objectives
- Both positive and negative test cases

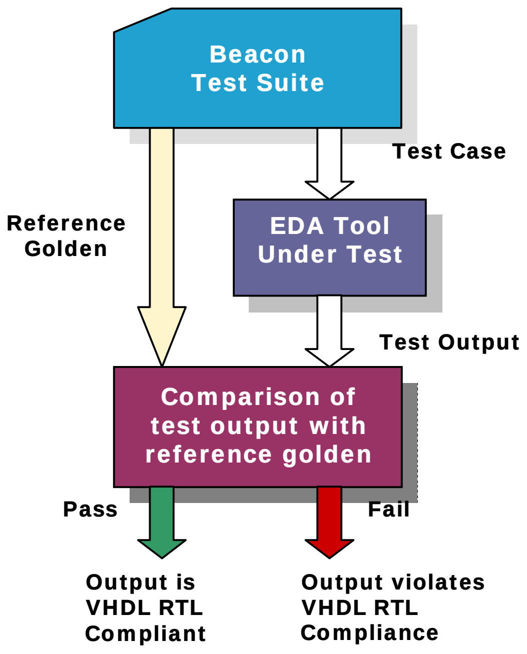
Addressing the needs of EDA tool developers to quickly evaluate the quality of their products, Interra's Beacon-VHDL-2008 delivers a comprehensive test suite based on VHDL IEEE 1076-2008 standards.

You can use Beacon-VHDL-2008 to characterize EDA tools for coverage and quality across various language constructs and styles.

Enabling you to discover language and RTL non-compliance early in the product development and testing life cycle, Beacon-VHDL-2008 offers:

- Reduced development costs and time-to-market EDA products
- Development of standard-based products
- Precise evaluation of bugs and errors in the product
- Measure of product quality
- Unbiased feedback on product quality
- Regression tests for quality assurance

The test cases and test benches can be applied to the EDA tool under evaluation and results can be compared with golden reference that is provided with Beacon-VHDL-2008.


Beacon Test Suite
Test Case
Reference Golden
EDA Tool Under Test
Test Output
Comparison of test output with reference golden
Pass
Fail
Output is VHDL RTL Compliant
Output violates VHDL RTL Compliance

The Beacon-VHDL-2008 Features

Comprehensive Test Suite

Well organized test plan of synthesizable and non-synthesizable test cases. It also enables you to evaluate syntax, semantics, and simulation aspects of a VHDL-2008 based tool.

Syntax and Semantics Cover

Includes over 2,500 test cases that cover constructs and styles.

Language Construct	Number of Test Cases
Design Entity and Configuration	179
Concurrent Statement	290
Declaration	384
Combined Construct	43
Design Unit	36
Expressions	738
Lexical Elements	141
Names	205
Predefined Language	304
PSL	14
ScopeAndVisibility	25
Sequential Statement	369
Specifications	18
Subprogram and Package	232
Tool Directive	33
Types	177
VHPI (VHDL Programming interface)	30
Total	**3218**

Test Case Covered

Positive test cases with test benches and Negative test cases,Most of the new constructs of VHDL-2008 has been covered, some construct partially and some construct fully.

Well Documented Test Plans

The test plans describe all test objectives and are categorized by sections. Each test case has a reference to the section number of the test plan.

Test Benches and Golden Reference

Provides test benches to instantiate test cases and apply vectors on inputs. Outputs are captured after an appropriate interval and written on to a file. You can easily apply the test bench to the test tool and compare the outputs.

Sample Test Case

--** Purpose:	External signal name : The external path is a package path name : Use external name as a prefix : In an expression.
--** LRM :	Sections 8.7
--** TestPlan:	Sections 6.2.2.1.1.1.1
--** Kind :	Semantic
--** Status:	SIMULATION_SHOULD_PASS

```
********************************************
*
package pack is
  signal s1 : bit_vector(0 to 7) :=
"10110011";
  signal s2 : bit_vector(0 to 7) := (others
=> '1');
end package pack;

entity packPathName1 is
  port(in1,in2 : in bit_vector(0 to 7);
       out1 : out bit_vector(0-to 7));
end entity packPathName1;

architecture arch of packPathName1 is
begin
  process(<< signal @work.pack.s2 :
bit_vector(0 to 7)>>, in2)
  begin
    for idx in 0 to 7 loop
      if idx mod 2 = 1 then
        out1(idx) <=
            << signal @work.pack.s1 :
              bit_vector(0 to 7)>>(idx) or
              << signal @work.pack.s2 :
              bit_vector(0 to 7)>>(idx);
        else
          ...
       end if;
     end loop;
  end process;
end architecture arch;
```

Interra Systems, Inc.
20195 Stevens Creek Boulevard, Suite 250, Cupertino, CA 95014
Phone: +1 408 873 1212, Fax: +1 408 873 1211
Email: info@interrasystems.com Web: www.interrasystems.com