# Beacon-RTL-VHDL

**interra** systems

## Comprehensive Test Suite for VHDL-RTL Compliance

### Key Advantages

- Field-proven test suite that has been used to mature several industry-standard EDA tools
- Developed in partnership with significant EDA majors
- Conforming to accepted definition and interpretation of RTL semantics
- Providing an unbiased quality analysis of EDA tools
- Complete coverage of RTL subsets and styles
- Comprehensive validation of a synthesis-based test tool

### Highlights

- Over 2,000 test cases along with test benches
- Expected simulation output for all positive test cases
- 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-RTL-VHDL delivers a comprehensive test suite to check RTL compliance in a VHDL-based EDA product.

Based on VHDL (VHDL'87, VHDL'93 and VHDL'02) standards, these test suites lets you characterize EDA tools for compliance and quality across various synthesis styles, RTL semantics, and supported/unsupported RTL subsets.

Enabling you to discover RTL non-compliance early in the product development and testing life cycle, Beacon-RTL-VHDL 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 the simulation output provided with Beacon-RTL-VHDL.
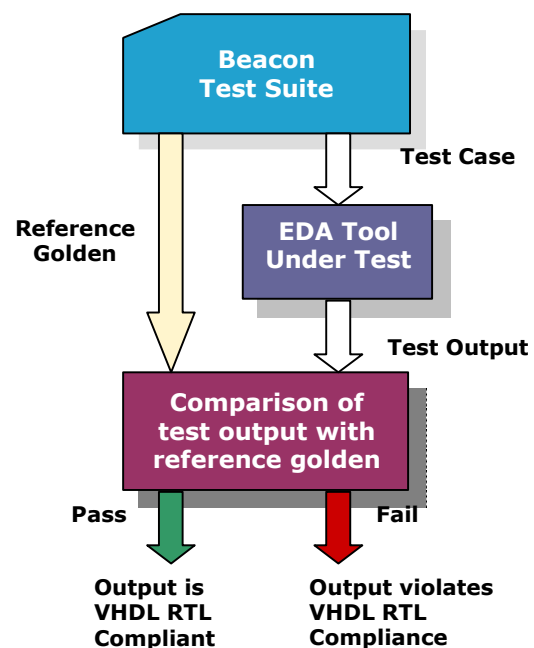
# The Beacon  -RTL  -VHDL Features

## Comprehensive    Test Suite

Beacon -RTL -VHDL is a comprehensive test suite covering syntax and semantics of VHDL        -based RTL synthesis. The test cases include several styles including complicated test cases that combine different synthesizable constructs with post     -synthesis    aspects, such as parallel output instances.

## Well -Categorized Test Suite

The test cases are well organized based on synthesizable constructs and styles.

### Synthesizable Construct        -based Test Cases

Construct  -based test cases target all synthesizable VHDL cons   tructs of varying usage. These test cases test the language coverage of an EDA tool.

### Style  -based Test Cases

Style -based test cases verify if an EDA tool is compliant with the various RTL modeling styles. These test cases include standard styles of RTL mod eling. Such as implicit style state machine, explicit style state machine, synchronous and asynchronous reset modeling.

### Negative and Ignored Test Cases

Negative test cases include styles that do not comply with the RTL subset.

Ignored test cases include        styles, which can be ignored by the tool, but which should generate an appropriate warning.

## 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 for all Tests

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 compa re the outputs. You can also apply the test bench to reference golden tool and compare the outputs.

| RTL subset/Style | Number of Test Cases |
|---|---|
| 3_state_inf | 5 |
| Attributes | 11 |
| Comb_machines | 6 |
| Concurrent, sequential | 315 |
| Configuration | 4 |
| Entity_architecture | 89 |
| Expression, types_decls | 407 |
| IEEE_UX01, IEEE _UX01Z | 27 |
| IEEE _X01, IEEE _X01Z | 28 |
| IEEE _signed, IEEE _unsigned | 98 |
| IEEE _std_logic, IEEE _std_ulogic | 30 |
| IEEE _stdlogic_vector, IEEE _stdulogic_vector | 31 |
| IEEE _small_int, | 12 |
| Ignore, unsup_con    st | 81 |
| Library | 1 |
| Miscellaneous | 255 |
| Negative | 79 |
| Package, sub_programs | 155 |
| Package_std_logic_arith | 5 |
| Package_std_logic_signed | 18 |
| Package_std_logic_unsigned | 16 |
| Pragma | 29 |
| Reg_inf | 94 |
| Seq_machines, state_machines | 35 |
| VHDL -93 | 244 |
| VHDL -2002 (Non   -synthesi  zable) | 90 |
| Total | 2,165 |

## Sample Test Case

```
-- Testcase : 20.8.4
-- A genera  te statement ( "for" generation scheme)
-- contains a declarative part.
entity decl_in_generate1 is
    port (          IN1, IN2 : bit_vector( 7 downto 0);
    OUTPUT : out bit_vector( 7 downto 0));
end entity decl_in_generate1;

architecture decl_in_generate1 of
decl_in_    generate1 is
    begin
      C : for i in IN1'range generate
      signal S : bit;            --   signal declaration
      begin
      S <= IN1(i) and IN2(i);
      OUTPUT(i) <= IN1(i) xor (S nand IN2(i));
    end generate C;
end architecture decl_in_generate1;
```