EZVerify

The Industry's First OpenVera® Productivity Tool Suite

The recent years have seen an exponential increase in the complexity of electronic chip design. Verification has become the most challenging aspect for the product team, sometimes taking up to 70% of the design cycle. VeriEZ Solutions is committed to increasing the **efficiency of verification**, allowing companies to shorten their verification cycle.

Design and verification engineers have continuously tried to combat aggressive schedules with innovative techniques and methodologies. The use of hardware verification languages such as OpenVera is an example of a new and viable verification methodology. **EZVerify is a high-performance verification productivity tool suite for OpenVera-based design and verification teams.**

To enable efficient verification, EZVerify uses a three-pronged attack:
i) identifies verification coding errors early in the flow, giving beginner and experienced users alike the opportunity to fix such errors
ii) allows users to build custom coding policies using the application programming interface (API) to an in-memory knowledge base of input modules
iii) provides a comprehensive document of verification information by analyzing the input modules (new, legacy or external verification IP)

EZVerify consists of three components:
1) **EZCheck™** - a static lint checker for OpenVera-based modules
2) **EZApi™** - a complete API to an in-memory knowledge base of input OpenVera modules
3) **EZReport™** - an HTML document containing verification knowledge extracted from the OpenVera modules

Benefits

- Accelerates verification
- Allows companies to implement a corporate-wide coding policy to generate correct, consistent and reusable OpenVera modules
- Enables smooth integration of geographically dispersed verification teams
- Performs quantitative analysis of external verification IP
- Provides easy mechanism to understand and integrate legacy verification modules

Features

- Full language support for OpenVera
- Fits into existing verification flows
- Provides in-memory knowledge base that can be accessed with an intuitive API
- Highly extensible - robust API provides foundation for infinite user extensions
- High-performance - components run very fast
- Platforms - Solaris and Linux

About veriEZ Solutions

VeriEZ Solutions, Inc. is a privately held company that develops and markets software solutions that enable efficient chip verification. It is the developer of EZVerify™, the industry's first OpenVera® productivity tool-suite. EZVerify includes a configurable and extensible static lint checker (EZCheck™) and verification knowledge extractor (EZReport™).

EZVerify detects verification errors, improves quality, enables verification reuse and shortens verification cycles. VeriEZ's products are easily incorporated into existing verification flows. More information is available on the company website, www.veriez.com.

Corporate

2086 Walsh Avenue Suite C2
Santa Clara CA 95050
Ph: 1.408.988.1604
Fx: 1.408.988.2711
Em: info@veriez.com

EZVerify™

Verify Faster

EZCheck
Static Linter

Using high-level languages for verification is a popular and particularly efficient methodology to reduce verification time.

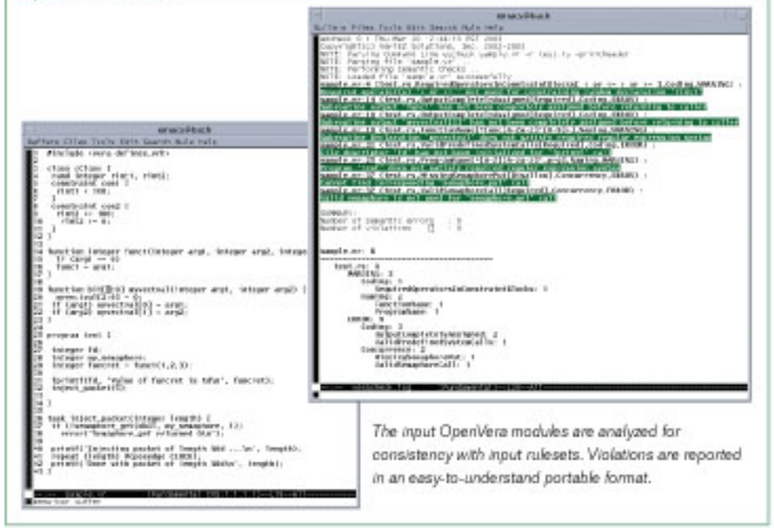
The use of a high-level language for hardware verification is similar to the widespread use of a hardware description language (HDL) for hardware design. Consequently, high-level design verification engineers face the same problems and challenges as their counterparts in hardware design. Linting is a widely accepted technology used by hardware designers. Designers routinely depend on linters to detect simulation/synthesis mismatch, combinational feedback and latch inference problems in HDL designs. EZCheck applies lint methodology to OpenVera designs.

EZCheck can detect problems such as erroneous usage of concurrency constructs, missing function output assignments, thread-dependent code, and several other serious errors that are not normally uncovered without several hours of debugging. In addition to identifying errors in OpenVera code, EZCheck is invaluable in allowing companies to design and enforce a coding standard for OpenVera modules. It consists of 200+ rules that user may use to require a certain coding style, enforce a design-wide naming policy and/or restrict usage of certain constructs. A subset of rules can be assimilated into a "ruleset", enhancing modularity and enabling the use of different rulesets for different applications.

Use Model

The input to EZCheck is one or more OpenVera modules along with one or more rulesets. The output is a list of violations that pinpoints areas in the input that are not consistent with the required rulesets.

Operation



The input OpenVera modules are analyzed for consistency with input rulesets. Violations are reported in an easy-to-understand portable format.

200+ rules

EZCheck comes standard with over 200 predefined rules. Predefined rules may be classified into 4 broad categories:

- **Verification errors:** Rules in this category detect errors in OpenVera code that are not detected by a standard compiler. Examples include incomplete (or missing) assignments to function outputs, incomplete case statements and incorrect class constructs and hierarchy.
- **Language rules:** Rules in this category allow users to restrict or disallow the usage of certain language constructs. This feature is significant because a) some language constructs are more error-prone than others and b) downstream tools used in the verification flow may not be able to handle certain constructs.
- **Naming rules:** Rules in this category allow users to establish a module-wide naming policy. Consistent naming schemes will result in manageable and reusable OpenVera components. For example, enumeration item names can require the enumeration declaration name as a prefix to prevent clashes in the design namespace.
- **Usage rules:** Rules in this category enable the creation of consistent, reusable and easily maintainable code. For example, rules to encourage consistent class design are especially useful for engineers who come from a hardware background.

Pre-packaged Rulesets

EZCheck comes standard with two pre-packaged rulesets, Basic and Best Practices.

- **Basic ruleset:** The Basic ruleset is a collection of predefined rules that warns the user of critical errors in the design. Examples of errors detected by this ruleset include:
  - Incomplete assignments to function outputs
  - Common mistakes in use of concurrency constructs, missing timeouts
  - Thread dependent code
  - Multiply-driven/undriven signals
  - Use of X or Z in relational, arithmetic and case operations
- **Best Practices ruleset:** The Best Practices ruleset is a collection of predefined rules that allows the user to follow certain policies that will result in maintainable, readable and reusable code. In addition to rules from the Basic ruleset, it also includes several rules to enhance maintainability. Examples of rules in this ruleset are as follows:
  - Naming requirements for files, interfaces and programs
  - Disallowing/requiring optional arguments in subroutines
  - Disallowing global subroutines and variables
  - Limiting levels of class hierarchy
  - Requiring/disallowing skew for all interface outputs and inputs

User customization

EZCheck is highly customizable, allowing several levels of customization.

1. **Input:** EZCheck has an intuitive input interface that is highly configurable. Users may run the design through multiple rulesets at the same time. Individual rules may be disabled from the command line. In addition, the files to be read in may be provided in an input file (similar to the popular .f file format in Verilog).
2. **Rule customization:** Every rule in the tool may be customized; for e.g., the rule that disallows certain types of declarations in an OpenVera design can be used to disallow bit/reg declarations in one ruleset and integer declarations in another. Or, the maximum levels of class hierarchy can be set to 3 in one ruleset and 5 in another.
3. **User-defined rules:** The advanced user can use EZApi (a separate product) to create new rules. These rules can be used in rulesets just like predefined rules.
4. **Custom rulesets:** The user may create an infinite number of rulesets by combining any number of predefined and user-defined rules.

EZApi
Custom Rule Builder

EZApi is a powerful component in the EZVerify tool suite.

It consists of an in-memory internal representation of data objects that captures the information in OpenVera modules. EZApi is a set of C++ routines that provides unrestricted and unlimited access to the internal representation.

Use Model

By using EZApi, users can build custom rules and policies that can be used with the EZCheck lint utility. It is a robust and proven API - predefined rules provided for EZCheck have been developed using EZApi.

Features

- Intuitive C++ API
- Full language support for OpenVera
- Coding examples included for easy setup
- Robust and proven

EZReport
Verification Knowledge Extractor

EZReport creates a comprehensive document that summarizes the verification aspects of a given OpenVera module.

It takes as input a list of OpenVera modules, and creates an HTML document with the following information:

- Class hierarchy
- Top-level program
- Synchronization mechanisms
- Global variables and subroutines
- Interfaces

Use Model

EZReport is useful to understand external verification IP and legacy IP. Verification engineers can make reuse decisions by using the comprehensive document created by EZReport. Users may also customize the document created to include additional proprietary information such as links to specific files and illustrations.