



Incisive[®] Enterprise

Specman Elite Testbench[®]

The shr_chkcv Package

Version 1.0

©1998-2007 Cadence Design Systems, Inc. All rights reserved worldwide.

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Incisive[®] Enterprise Specman Elite[®] Testbench contains technology licensed from, and copyrighted by: University of California and is © 1990, The Regents of the University of California. All rights reserved. Free Software Foundation, Inc. and is © 1989, 1991 Free Software Foundation, Inc., 51 Franklin St., Boston, MA 02110. All rights reserved. Open Source Initiative and is © 2004 by the Open Source Initiative, Law Offices of Lawrence E. Rose, 702 Marshall Ave. Ste. 301, Redwood City, CA 94063. All rights reserved. University of Colorado and is © 1995-2004, Regents of the University of Colorado, Boulder, CO 80309 USA. All rights reserved. Sun Microsystems, Inc. and is © Sun Microsystems, Inc., 4150 Network Cir., Santa Clara, CA 95054. All rights reserved. Scriptics Corporation and is © Scriptics Corporation. All rights reserved. loi Kim Lam and is © 1993-1995, loi Kim Lam. All rights reserved. Rudolf Usselmann, and is © 2000-2002 by Rudolf Usselmann. All rights reserved.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer;
2. The publication may not be modified in any way;
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement;
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration

Patents: Cadence Product Incisive Enterprise Specman Elite Testbench, described in this document, is protected by U.S. Patents 6,920,583; 6,918,076; 6,907,599; 6,687,662; 6,684,359; 6,675,138; 6,530,054; 6,519,727; 6,502,232; 6,499,132; 6,487,704; 6,347,388; 6,219,809; 6,182,258; and 6,141,630. Other patents pending.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Coverage of Checks and Expects

1 Introduction

shr_chkcov is a shareware that enables coverage collection of the *e* checks and expects in Specman 6.x. It is a new version of an existing package that worked on top of Specman 4.3.x, but, due to changes in the *e* infrastructure, was broken in Specman 5.x.

The shareware's capabilities provide a short-term solution, and a first step on the way to a complete assertion coverage solution that will be implemented once integration of the assertion coverage concept with our *v*Planning approach is enhanced.

shr_chkcov allows you to collect and view coverage for each of the checks in the environment that use the built-in Specman checking constructs. This enables you to determine whether any of the checks have not been evaluated in the set of simulation runs that were used to create the currently loaded coverage database.

The user can specify that coverage is not to be collected or analyzed for all checks, or a subset of the checks, in the environment.

2 Coverage of checks and expects

There are two types of assertion checks in *e*:

1. Temporal checks defined with the **expect** construct define the correct behavior over multiple cycles. The temporal expression defined with the **expect** construct is always expected to succeed. Temporal checks are struct members and are continuously active during a run.

Example:

```
expect TE1=>TE2
```

defines a temporal rule that TE2 must succeed when evaluated starting one cycle after TE1.

Coverage of Checks and Expects

Coverage of checks and expects

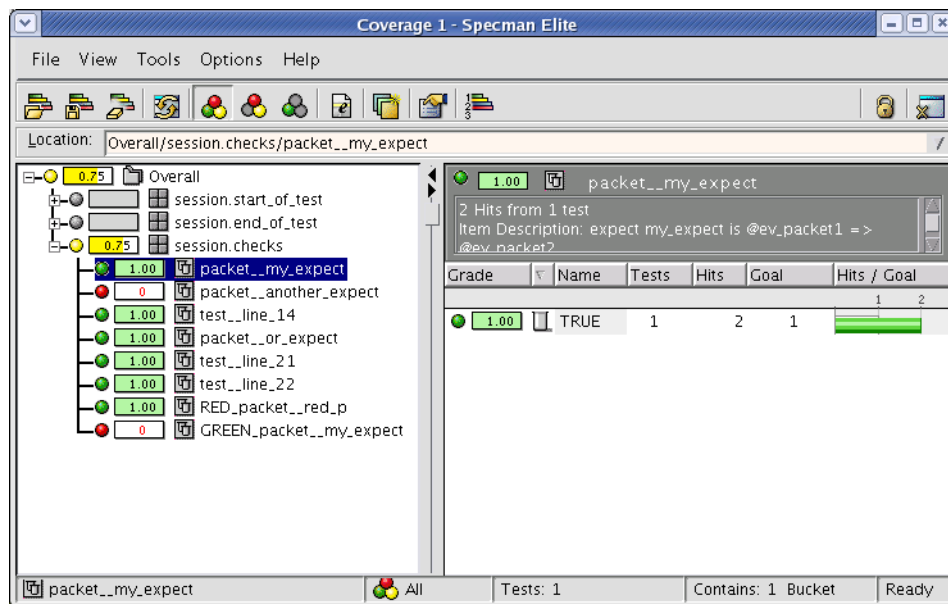
2. Data checks defined with the **check that** construct define the expected values at a particular point in time. Data checks are actions, and must be included in a method or a Time Consuming Method (TCM). They are activated only when the **check that** action is executed when the method is called.

Example:

```
check that 'data_out' == 'data_in';
```

Using the shr_chkcov shareware, a new cover group will be added under “session”, named “checks” (similar to the session.events cover group). This group will include all of the checks and expects in the design, each one of them represented as an item. Each item name is module-name__lineNumber (for example, test__line_20). Named expects will have their name used as the item name, in the following format: structName__checkName (for example, packet__my_expect). If a named check is defined in a when sub-type, the subtype name is used before the struct name (for example, GREEN_packet__my_expect).

Each item has only one bucket, named “TRUE”, indicating whether the check/expect was evaluated as described below, at least once, in one of the simulation runs in the coverage results database.



The semantics for the coverage is as follows:

- A check is covered if the **check that** action was executed.
- An expect is covered if the left hand side was evaluated to TRUE, and the ' \Rightarrow ' was triggered. For example, the expect statement:

```
expect TE1=>TE2
```

will be considered as covered if TE1 was evaluated to TRUE.

The coverage is collected per-type only. The shr_chkcov shareware does not distinguish between different instances.

3 shr_chkcov Usage

3.1 Using shr_chkcov with Your Environment

To use the shr_chkcov shareware, you simply need to load/import the top file. It is important that the shareware is loaded before loading the user's VE containing the checks/expects. You also can compile the shareware on top of Specman and then use this new executable when loading/compiling your environment.

For example:

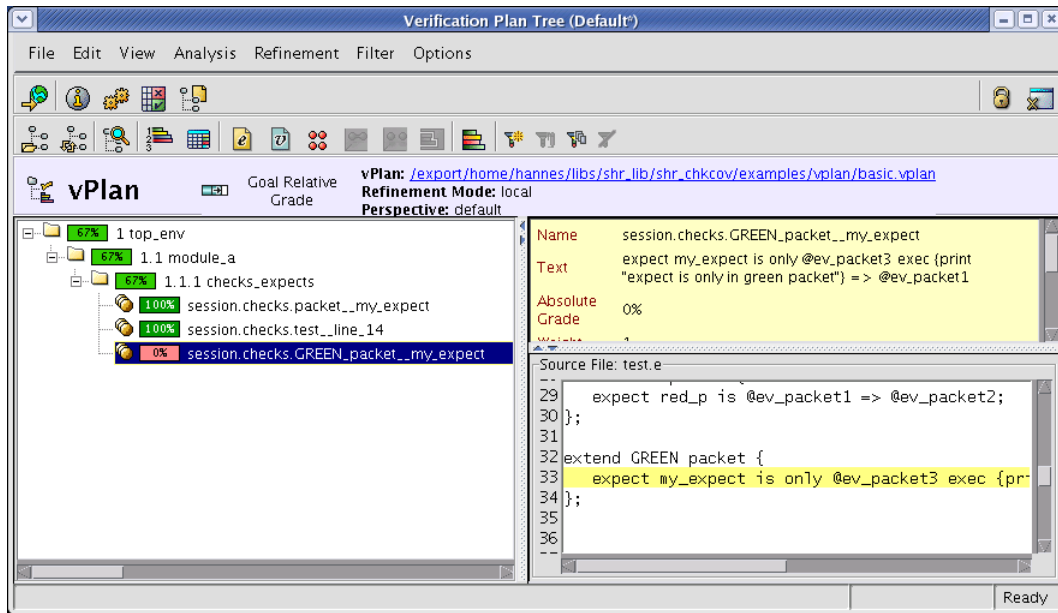
```
load shr_chkcov/e/shr_chkcov_top.e
load my_env/sve/xyz_config_top.e
```

3.2 Connecting Checks and Expects Coverage to a vPlan

To include the coverage points of checks and expects in your vPlan, use the full path notation of the coverage database, according to the naming described above.

For example:

```
section top_env {
  section module_a {
    section checks_expects {
      coverage: session.checks.packet__my_expect;
      coverage: session.checks.test__line_14;
      coverage: session.checks.GREEN_packet__my_expect;
    }
  }
}
perspective default {
  top_section: top_env;
}
```



3.3 Configuring Coverage Buckets

It is possible to configure (set on/off coverage) of a specific check or expect, using the `set_cover()` method. The method supports wildcards as well.

For example:

```
extend sys {  
  init() is also {  
    set_cover("session.checks.packet*", FALSE);  
    set_cover("session.checks.packet__my_expect", TRUE);  
  };  
};
```

3.4 Limitations

Be aware of the following issues and limitations:

- Assertion coverage semantics is not tightly defined by its nature.
- This solution does not try to define the full semantics for assertion coverage (if such exists), but supplies only a minimal set of capabilities that were included in the previous shareware.

- Coverage of checks and expects is collected per-type only.
- Minimal support level will be supplied for the utility.
- Minimal testing has been done for the utility.
- The proposed solution might not be compatible with future implementations of an enhanced solution.
- Coverage is collected before the execution of the check, and the check will be considered as covered even if the check fails (see below). If the user changed the severity of the check to IGNORE, coverage is not the right indication for the check if it was successful or not.
- An expect is covered if the left hand side was evaluated to TRUE, and the ' \Rightarrow ' was triggered, and we don't check that the right hand side was evaluated to TRUE as well. So for $TE1 \Rightarrow TE2$, if the simulation ends after TE1 succeeds but before TE2 succeeds, our coverage will record this as covered.