JEDA Validation Tools Suite
High Level ESL Model Validation Solution

JEDA Validation Tools Suite (JEDAsuite) automates high level model validation to accelerate ESL model development, reduce platform level debugging and thus reduce ESL project risks. It enables a predictable and reproducible validation solution through automation. JEDAsuite integrates coverage measurement tools, tools to generate performance traffic, means to implement checkers, to self-check protocols or other temporal dependencies in a unified environment.

Target customers for this solution are hardware engineers, who want to establish a high level model validation flow. High level model developers, who want to develop a comprehensive validation process for their model library. And finally the model user, who wants to use well tested models and implement checks to automatically detect critical user situations during their performance analysis.

Features

- Full SystemC IEEE 1666 and C/C++ support
- Eclipse–Base Development Environment
- Position Aware Function Coverage, Decision Coverage, Condition/Branch, Multi-condition coverage, Functional Data Coverage
- Temporal Checking for Transaction Level Models at various levels of abstraction
- Item, Cross, Transition Coverage
- Intelligent Traffic Generation for fast system level performance traffic specification
- Regression Test Ranking

JEDA Suite Target Users

JEDAsuite is targeted primarily to three different groups of ESL users:

System Architect:
- Measure coverage of test to assess the quality of the IP component or virtual platform used for performance evaluation and architecture decisions
- Specify and generate performance specific traffic patterns mimicking various user cases, like video or audio traffic, conferencing systems. The traffic pattern also mimic architecture specific constraints typical in memory controllers or interconnect structures
- Implement functional checks for components or interfaces, to automatically identify that specific conditions are never reached

Software Engineer:
- Measure coverage to assess the quality of virtual platforms used for SW development
- Check automatically for accesses to restricted memory addresses
- Identify un-used memory regions

Hardware Engineer:
- Eliminate unused code in high level model specification so that dead code will not propagate into RTL in High Level Synthesis flows.
- Generate comprehensive input vectors to increase coverage of the models
- Apply coverage metrics traditionally used for RTL verification to high level models to increase code quality as well as reduce the number of synthesis runs
• Construct checks for expected internal behavior or interface specifications for unit-tests. Reuse the checks inside virtual platforms.

Advanced Code Coverage Feature Highlights

JEDAsuite’s advanced code coverage component provides code coverage capability for high level SystemC and C++ models. The main goal of code coverage metrics inside the JEDA Validation Tools Suite is to achieve similar coverage in the SystemC and RTL code. This means it needs to identify in details each path through the SystemC code. Listed here are highlights on the features of the Advanced Code Coverage functionality:

Position Aware Coverage: Identifies for each function call and keeps track of line, function, condition and multiple condition separately for each time a function is called. With this feature one can easily trace all paths in an executable, which are triggered in a simulation run.

Exhaustive decision coverage: multi-bit condition evaluation allows identifying on a bit-by-bit basis that for example all states in a state machine have been executed. More importantly it identifies states, which are not reachable. This marks un-used code, which can be eliminated to improve RTL results.

SystemC instance coverage: Identifies for each SystemC instance separately line, function, condition, decision and multiple condition coverage.

Filtered viewing: for large designs performing analysis on smaller chunks of data help reduce the scope of what's being analyzed. This capability provides filtering mechanism by source files name, class name, base class name, function name and by SystemC module instance name basis

Functional Data Coverage Feature Highlights

JEDAsuite’s functional data coverage component measures the quality of an IP component model or virtual platform based on a user defined quality metric representing the model’s specification intend. Some of the functionalities available in this component include:

Item Coverage – measures if a single or range of values for a variable or register have or have not been executed. These values represent e.g. a CPU instruction opcode, a burst length, a memory address etc. So that the user can be sure that all important states in the system have been triggered.

Transition Coverage – measures the presence of a specific sequence of values on a given variable. Examples of transition coverage usage are: in TLM2.0 based IP model, checking that BEGIN_REQ(request_begin) is followed by END_REQ and BEGIN_RESP is followed by BEGIN_RESP.

Cross Coverage – measures occurrences of two or more item combined. It identifies if those combined coverage conditions are met. Examples of this can be a memory read to an address range need to occur while an interrupt is being serviced, to make sure that this corner case is tested.

Group Coverage – users have different needs for grouping and organizing the coverage metric in ways that make sense to them and represents the design intend. This feature provides the ability to group item, transition and cross coverage together to reflect specification requirements, which need to be tested.

JEDA Temporal Checker Feature Highlight

JEDAsuite’s Temporal Checker component provides users the ability to write native SystemC temporal checks for their models. Using this feature, one can specify the requirements and the expected behavior of the design in a very abstract way. Than it checks automatically that the model developer meets this specification in the SystemC models. The checkers allow to identifying architectural constraints, temporal dependencies and/or various other model bugs automatically to guarantee that the model is compliant with the specification.

The component comes with its own Temporal Assertion Development and Debug environment to help you write temporal checkers quickly.

Intelligent Traffic Generation Feature Highlight

This component employs two technologies created in JEDA: random constrain solver and random sequence generation mechanism. Using these underlying technologies, the component provides API's for the user to quickly specify configurable performance traffic with fine grain control.

The component is designed in such a way that the performance traffic characteristic definition is decoupled from the protocol used to send the
traffic. This allows having a single description of a video decoder traffic pattern and apply this to various protocol interfaces.

The random constraint solver technology is available today. The sequence generation mechanism will be available later in 2009.

Test Ranking Functionality

JEDAsuite includes a test ranking system that prioritizes regression tests based on various decision criteria that the user sets. This allows to quickly identifying the subset of tests that yield maximal coverage. The user identifies the target coverage and the tool identifies the set of tests, which runs the shortest time, or the least amount of tests.

Unified Development Environment

Eclipse analysis viewer: provides summary and detailed coverage results viewing capability for both the code-coverage and functional data coverage components of JEDAsuite

Temporal Checker debugger: this debug environment contains a summary view and a hierarchical debugger window. The latter directs users to the exact location of temporal check failure points helping them understand the cause of the failure.

Supported Simulators and Platforms

Platforms:
- Linux RHE 3.0 32-bit and 64-bit AMD

Simulators:
- OSCI SystemC
- Cadence IUS
- Coware SCIDE SCsh
- Forte BDW
- Mentor ModelSim, QuestaSim
- Synopsys VC