



## STREP – IST 033709 VERTIGO

---

Verification & Validation of Embedded System Design  
Workbench

### Deliverable D1.2 – Revision of Verification requirements and VERTIGO verification flow

DUE DATE	30 July 2008		
ACTUAL DATE	30 July 2008		
START OF PROJECT	01 June 2006	DURATION	30 months
ABSTRACT	This document contains the end-user requirements, that drive the development of VERTIGO flow, and the evaluation criteria to assess the usability of the flow. It reports also requirements for the partners' developments and the related measurement criteria		
AUTHOR, COMPANY	Stephen Scholefield TransEDA, Umberto Rossi STMicroelectronics		
WORKPACKAGE/TASK	WP1/T1.3		
FILING CODE	VERTIGO/Deliverables/08_D1.2_VERTIGO		
KEYWORDS	Design Platform, TLM, SystemC, Assertion Based Verification, Model Checking, Requirements		

DOCUMENT HISTORY

Release	Date	Reason of change	Reviewer	Distribution
1	12/07/2008	First draft	Rossi	Internal
2	14/07/2008	Bottom Up / Top Down flow revision	Rossi	Internal
3	17/07/2008	Verification Requirements inserted	Rossi	Internal
4	19/07/2008	Verification Flow	Scholefield	Internal
5	21/07/2008	Picture clean-up	Rossi	Internal
6	22/07/2008	Move to a marketing revision	Scholefield	Internal
7	28/07/2008	Reviewers' comments from draft		
7	28/07/2008	Introduction extension with summary of VERTIGO targets and tool integration	Scholefield	Internal
8	28/07/2008	Quantifiable targets	All	Internal
9	29/07/2008	Final Integration	Rossi	CO

# 1. Introduction

---

This document is the revision of the former Deliverable D1.2 due at M18, on November 30<sup>th</sup> 2007, that has been rejected by the reviewers at the 2<sup>nd</sup> year review held in Verona on July 4<sup>th</sup> 2008. The reviewers have asked a substantial remake of the Deliverable and have given clear indications on the expected results.

Chapter 2 describes high level requirements that identify the problems and the needs to justify the VERTIGO development.

Chapter 3 details the verification features and the verification requirements that must be satisfied in order to assess VERTIGO developments and flow.

Chapter 4 reports the revision of VERTIGO flow

Finally, Chapter 5 reports the expected verification/modelling domains where the partners will contribute

## **Review of VERTIGO concepts and global targets**

Chip designs are becoming more complex and the price of design errors ever higher – particularly so if the chip reaches the mass consumer market where a product recall or product delay can see competitors leap-frog to market leadership.

The turf wars between Nintendo, Sony and Microsoft in the games console market show a series of leap-frogs with leadership changing place at each round. Deliver a complex chip ahead of the market and the rewards are immense – get a chip design wrong and the safer less complex design becomes the new leader.

Chips are designed in hardware description languages and hence suffer the same fate as software code with highly labour intensive debugging cycles.

The tools proposed in project Vertigo are aimed at reducing debugging cycles and adding computing power to crack the complexity of chip design – the prize is a high value tool set capable of being sold to the largest chip designers in the world.

The VERTIGO project deals with modelling and verification techniques, applied at the SoC (System on a Chip) design, working at System (specification), Transaction (Communication) and Register (RTL, Implementation) *Levels*.

Such techniques have been in use for several years but the main difficulty is to move from one level to another and to state criteria of correspondence among them. Existing criteria may be implicit with more precise specification languages that automatically synthesise Register implementation (e.g. Esterel) or with pre-defined platform in which IP blocks are assembled to provide the final product (e.g. CoWare). However, automatic synthesis only works top-down for IP blocks, whereas with pre-defined platforms the classical verification methodology is strictly bounded to massive simulations.

At STMicroelectronics such flows have been applied at specific segments, e.g. set-top-box systems, imaging systems; in those cases standard applications and benchmarks exist to drive the system simulation and finally verify the SoC. Segments like Office/Computer applications or Automotive, however, are affected by a wider diversification and there is no access to the customer proprietary final application. As a consequence, in Computer and Automotive segments there has not been enough motivation, in the past, to move away from RTL for system verification.

The purpose of VERTIGO is easing the move among the different *Levels*, relying both on existing standards or commercial flows and on new developments for general or specific needs.

Key targets to achieve the purpose are (re-phrased from the DoW):

- KT1.** ability to express descriptive properties at System or Transaction Level, i.e. true behaviour of the design which is not explicit in the specification
- KT2.** to introduce coverage criteria for the Transaction Level properties and to compare the results of similar properties one at Transaction Level and the other one at RTL
- KT3.** to integrate dynamic and static verification around a coherent Assertion Based Verification approach at RTL
- KT4.** to federate partner provided techniques in the design of a novel static verification environment operating at RTL for IP verification and IP interaction

VERTIGO developments and flow are to be tried on industrial designs provided by STMicroelectronics, in addition to benchmarks provided by partners

**Partners contributions and tool integration**

Partners contribute with state-of-the-art technologies and new developments during VERTIGO progress. A necessary prerequisite is to provide a uniform modelling language for the design description in HDL, to which all partner tools may interface. This language is HIF, an extension of AIF format developed during the SYMBAD IST-2001-34607 project. HIF format and HIF access utilities ensure integration beyond the existing parsers of standard languages like, VHDL, Verilog or System Verilog. The picture in Fig.1 reports a view of the main VERTIGO tools and their integration.

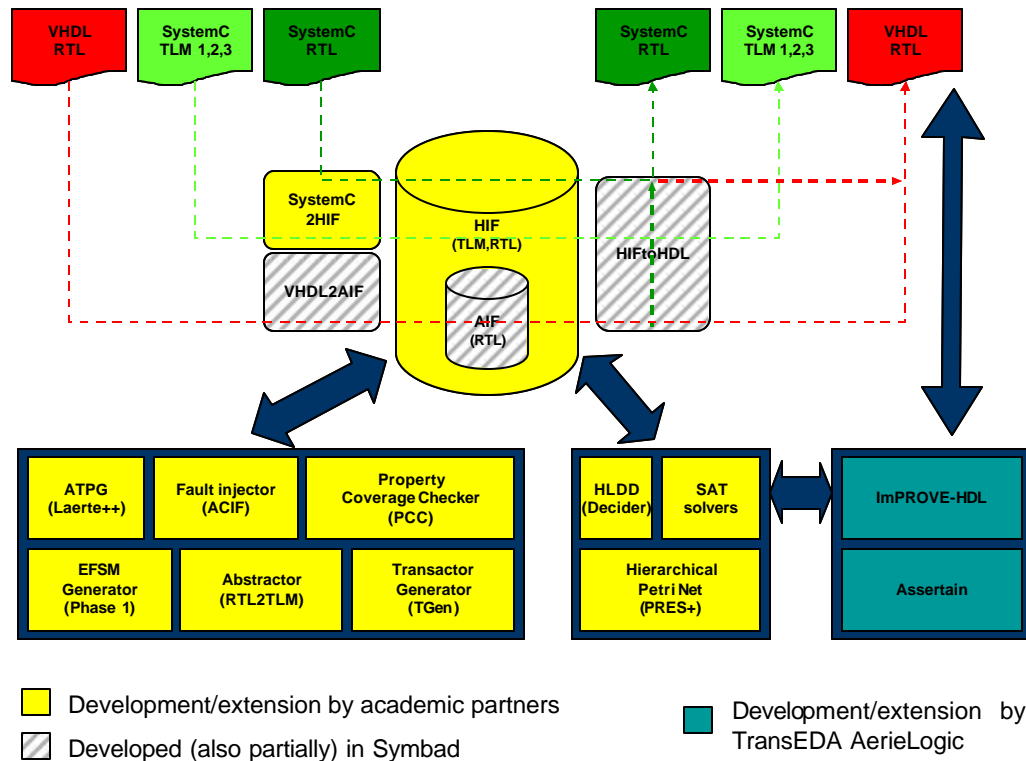


Figure 1 Tool Integration

The final VERTIGO workflow is to be implemented in TransEDA's Assertain integration. A tight integration is forecasted with existing commercial tools, VN-Cover, VN-Check, imProve-HDL and University of Southampton's SAT tools (via improve-HDL). The tools developed by the Universities of Linköping, Tallinn and Verona will be brought together in the Assertain workflow as a loose integration of each partner's executables. The output will be rendered in HTML with each run held in a back-end database. A common database is to be used to store the design iterations. Different tool results should be uploaded to the database from ever they are generated across the Network. Results should be grouped and scored with the objective of achieving a balanced Verification Signoff.

## 2. End User Requirements

---

Winning in RFQ (Request For Quotation) in SoC industry today, requires the ability to provide the potential customer with a model of the system that can be used to validate basic application, the claims of performance and, in some cases, of reliability and safety. The ability of developing basic SW - e.g. simple boot sequence, simple drivers – before silicon availability is becoming a must for the SoC integrator itself and a clear requirement from marketing.

The achievement of these targets involves set of flows that span from High level System Specification, SW/HW co-design, Behavioural Synthesis etc. Eventually the result is the customization of an existing platform, plus a set of IP blocks taken from a catalogue of RTL and sometimes TL models, plus a (small) set of IP blocks built on purpose.

The final target is to verify the correct interaction of the IP blocks in the system and, for the new produced models of IP blocks, to verify the correctness at RTL and TL with emphasis on the consistency between these two levels. The development of basic SW layer is considered as a part of the stimuli that are used to validate the system.

### 2.1 Target test candidate - a configurable platform

---

A Configurable Platform is the starting point to define products and variants in a family of applications. An example of a platform built around a basic infrastructure is reported in Fig.2

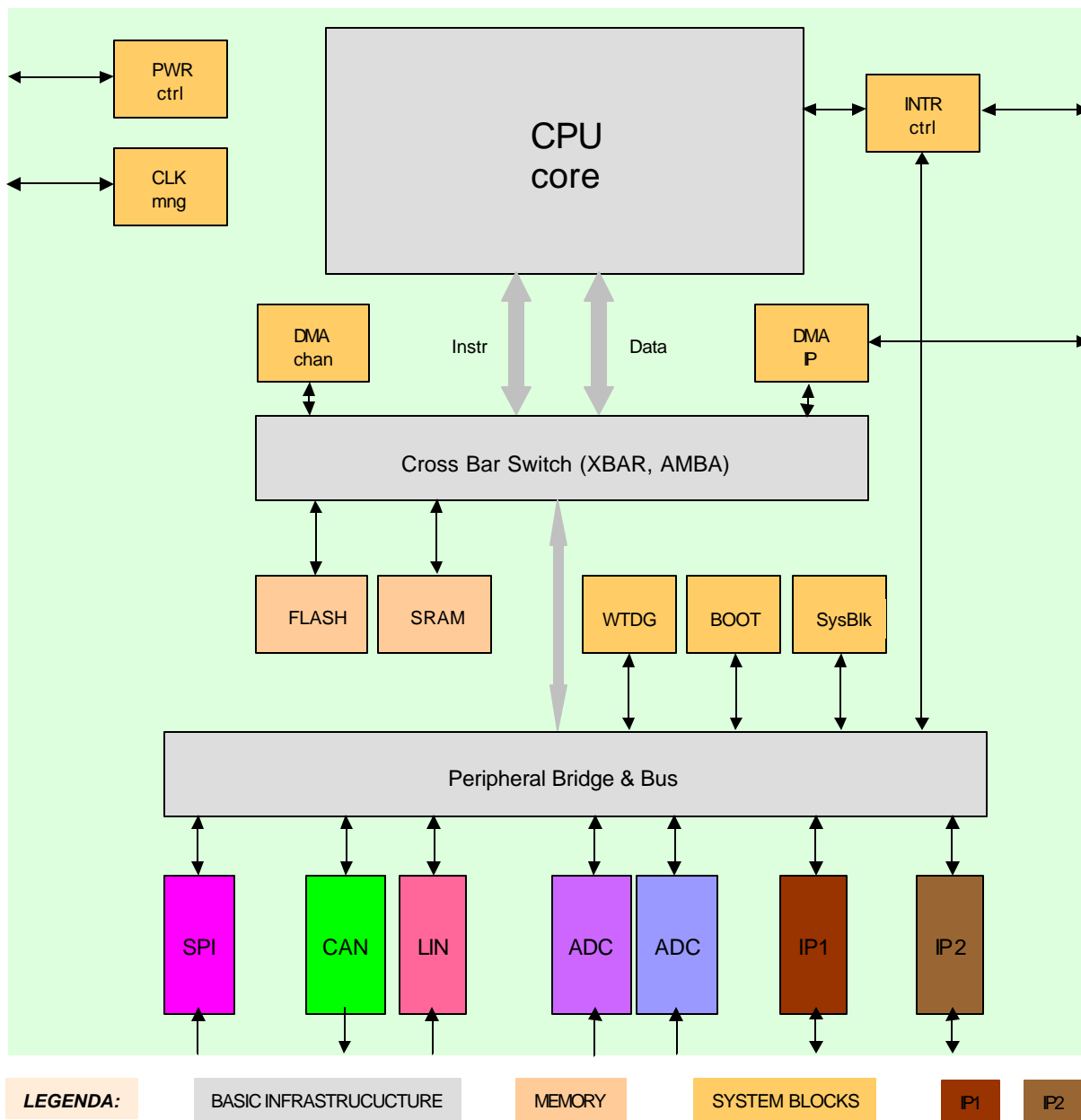


Figure 2 Configurable Platform

A set of system parameterized blocks is provided to complete the basic structure functionality, e.g. Interrupt Controller, DMA Controller, Data acquisition DMA, Watchdog Timer, Boot Assist Module. The final product is finalized by means of the IPs connected to the peripheral bus, like SPI, CAN, LIN network interfaces, ADC interfaces, timers etc.

## 2.2 Soc integration approaches

### Bottom Up flow

Bottom up flow is relevant when the platform model is quite well defined, as shown in Fig.2. With this definition of Bottom Up, most of the IPs are available at RTL, usually because they have been previously successfully integrated in production silicon. On those IPs it is necessary to qualify existing verification patterns to comply with more demanding coverage targets – that may have not be taken into account at the time of the original verification – or to generate new patterns that show the correct integration in the new target SoC.

Bottom Up flow comes into play with the IP *abstraction* process. This may take a long time when a rich legacy of RTL IP blocks is available from the catalogue but a limited number of TLM models is present. The final target is to reach the maximum simulation speed at the platform level and release an executable model that protects Intellectual Property. A compromise to the abstraction of the IP block is represented by connecting an RTL simulation model with the TLM platform, keeping the simulation speed at an acceptable level.

The TLM platform may include a mix of TL and RTL IP models depending on the availability in the catalogue. The basic infrastructure, ISS core plus busses, must simulate at full speed.

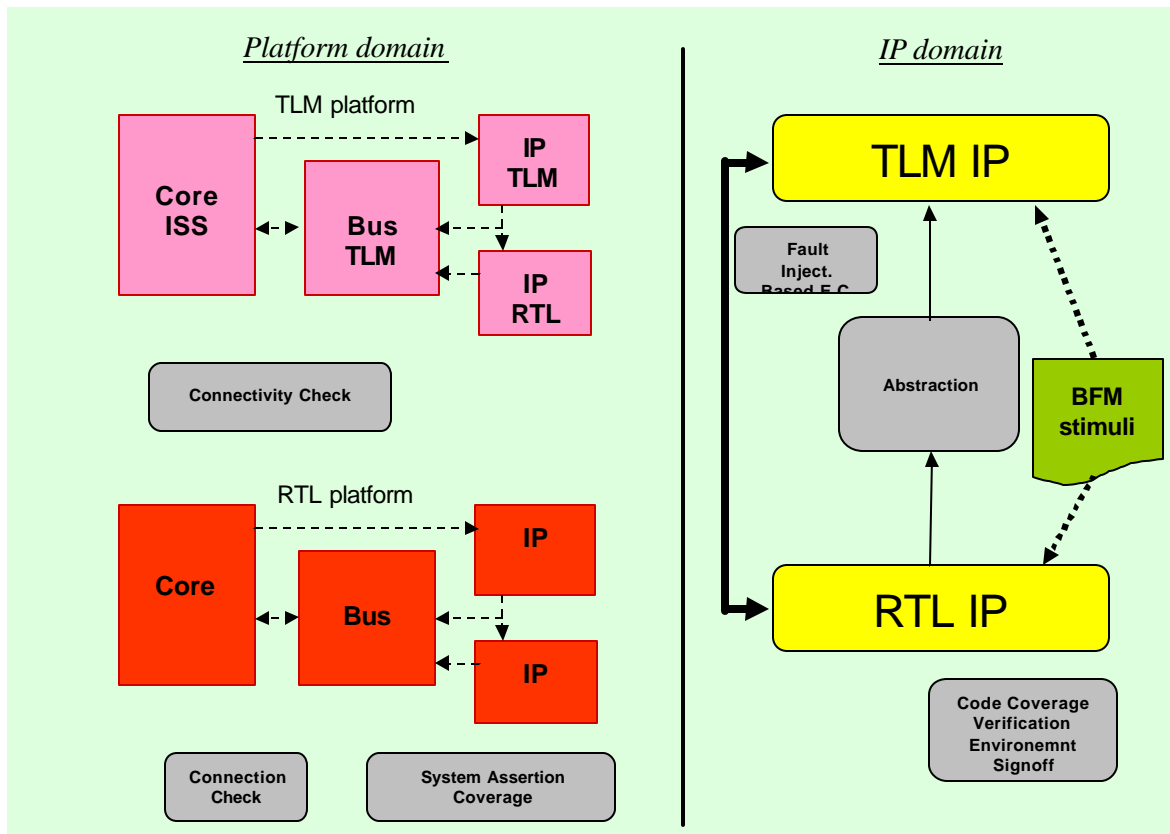


Figure 3 Bottom Up flow

## Top Down flow

Top Down flow deals with the refinement from specification to working product. In the case of VERTIGO we can manage IP block refinement only, that involves the creation of a new IP block not available in the catalogue.

The IP block is defined as an executable specification at TLM and the RTL model is provided either by Manual or Behavioural synthesis. In case of manual synthesis the capability of assuring consistency of RTL model vs TLM becomes critical.

The RTL IP block needs to be thoroughly verified, therefore a capability of Formal Model Checking is provided in order to cover 100% critical mission features.

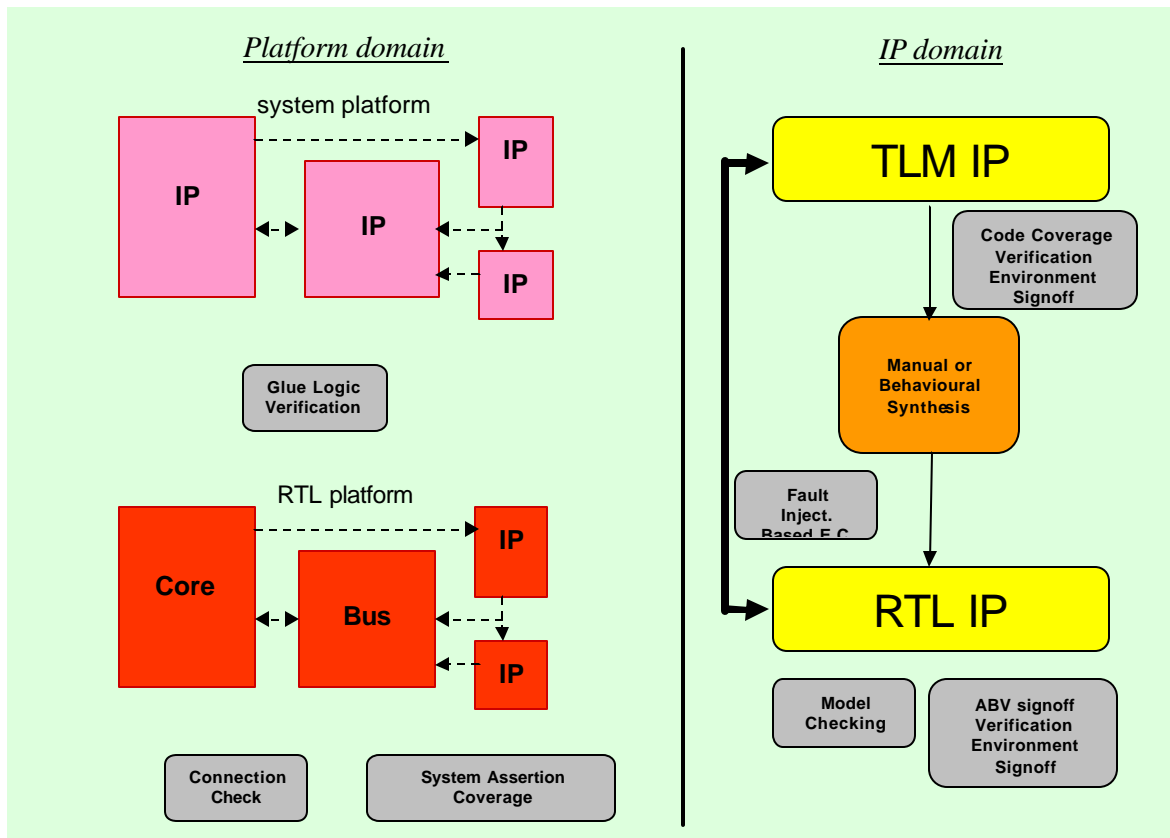


Figure 4 Top Down flow

In the platform domain it is possible to perform checks involving abstract functions, provided that the system is modelled in Petri Net notation. With the RTL model it is possible to perform check on connection and System Assertion Coverage. There is no support to link the system definition to the RTL in the platform domain.

### 3. Verification requirements

---

#### flow Top Down – Bottom Up

Verification feature	Verification requirement	Domain	Key Targets
spot unreachable lines in the code	Target 100% (explained) code coverage analysis	IP/RTL	KT3
spot uncoverable sub-expressions			
spot parts of code unverified by assertions	Assertion based verification signoff	IP/RTL	KT3
spot un-exercized parts of complex assertions	Target 100% assertion coverage analysis	IP/RTL Platform/RTL	KT3
spot un-exercized paths of complex behaviour	Find bugs close to corner case situations	IP/RTL	KT3, KT4
Measuring observability (in addition to controllability) of test patterns	Qualify verification environment and reach 90% observability	IP/TLM - RTL	KT2, KT4
automatic generation of properties vs reference bus	Model IP integration vs bus infrastructure – AHB	IP/RTL Platform/RTL	KT3, KT4
automatic generation of properties for PAD muxing logic	Model IP integration vs PADs	Platform	KT3, KT4
formal or bounded proof	Verification closure on integration properties described above	IP/RTL Platform/RTL	KT3

### flow Top Down

Verification feature	Verification requirement	Domain	Key Target
cause-effect with conditions and temporal frame support	Formally specify behaviour for IP integration	Platform/System	KT1
Equivalence Check RTL vs TLM	Provide coverage to check manual synthesis	IP/RTL	KT2
formal or bounded proof	100% confidence on property coverage	IP/RTL	KT3

### flow - Bottom Up

Verification feature	Verification requirement	Domain	Key Target
abstraction support	Translate legacy IP blocks to TL with different IP styles, flat and hierarchical	IP/RTL	KT2
Equivalence Check TLM vs RTL	Check abstraction result	IP/RTL – TLM	KT2
integrate RTL models in TL platform	Perform RTL simulation by means of TLM testbench Speed-up simulation at integration level	Platform/TLM	KT2
measuring observability of test patterns	Check IP connections at TL integration and reach 100% explained coverage	Platform/TLM	KT2

### 3.1 Table of key goals vs ST testcases

Feature	Requirement	Qualitative Measure	Quantitative Measure	Test Case
Verification of I/O mux for Alternate Functions implementation	(I) Alternate Functions description (II) automation of property production from specification (III) formal proof	Complete the flow from (I) to (III) and possibly find link with the SPIRIT proposal	Achieve 100% coverage with formal. Complete the job in 1 night vs 1 week for the full SoC test regression	JPC560xB – SoC for car body applications
RTL Abstraction & Verification	(I) automatic abstraction capability (II) equivalence checking (dynamic)	Definition of assertion based equivalence criteria	Complete the job in 1 day vs the manual 1 week.	ECC block CRC block SPI block
Glue Logic Verification	Spotting critical latency in interconnect	Spot latency related problem	Save design re-spin due to non discovered bug	Boot control chain in JPC560xx
Formal Proof - full proofs and bounded proofs	Complete proofs for both triggering conditions and property assertions	To complete 3 out of the 5 present bounded proofs obtained with commercial tools	To achieve the result in 1 night – we believe the properties be true – vs the inconclusive 3 days	MPS block
Dynamic and Static coverage for ABV	Unified coverage report	Quality of the unified report – see section 4.1 . Ease comprehension of uncovered items to quickly fix patterns / assertions	Achieve 100% assertion coverage. No visible degradation in simulation speed by analysing complex assertions	DEMO platform assembled with CRC, ECC and SPI around QEmu

### 3.2 Measurement criteria for partners' developments

TransEDA			
Feature	Qualitative Measurement	Quantitative Measurement	Reference
Integration of partner tools into Assertain in order to achieve verification closure through VERTIGO flow – see section 4	Integration of enhanced improve-HDL. Provision of links to academic partner tools for a loose integration with Assertain		Assertain 2006.01-2-00
Integrate Coverability feature	Increment dynamic coverage by means of formal proof	Achieve 100% line coverage on ECC, CRC, SPI blocks + internal testcases	Assertain 2006.01-2-00
Enhance assertion coverage metrics	Use of formal analysis for Assertion Step Coverage and Assertion Variable Coverage	Achieve 100% ASC and 100% AVC on assertions used with ECC, CRC, SPI + internal testcases	Assertain 2006.01-2-00

AerieLogic			
Feature	Qualitative Measurement	Quantitative Measurement	Reference
Support of PSL and SVA assertion languages	Full support of PSL simple subset in Verilog and VHDL flavors. Support of equivalent subset in SVA. Support embedded PSL and SVA. Support external PSL		improve-HDL v2.7
Formal proof capacity	500 assertions on 45 industrial testcases	Reduce the memory consumption to 1/3 <sup>rd</sup>	improve-HDL v2.7
Bounded proofs	500 assertions on 45 industrial testcases	Reduce the solution time to 1/5 <sup>th</sup> average	improve-HDL v2.7
Full proofs	500 assertions on 45 industrial testcases	Improve 100% the number of proofs	improve-HDL v2.7

University of Southampton			
Feature	Qualitative Measurement	Quantitative Measurement	Reference
New SAT based Model Checking addressing both Bounded (BMC) and Unbounded (UMC) proofs	Integrate new techniques MCSAT and MaxSAT plus new solvers like Hinotos to serve for bounded and full proofs	See improve -HDL enhancements	Competing SAT based model checkers

University of Verona			
Feature	Qualitative Measurement	Quantitative Measurement	Reference
HIF intermediate format for VERTIGO common description language	Support of TLM/RTL and SystemC/VHDL languages with related translators	Fixing all known bugs out of AIF of SYMBAD project	AIF features out of SYMBAD project
Abstraction Refinement of HDL descriptions and Assertions	Integration of EFSM generator, A2T abstractor and TGen in a consistent flow	To achieve IP abstraction – ECC, CRC, SPI – in 1 day	Manually requires 1 week
Methodology to achieve TLM/RTL block equivalence	Definition of RTL to TLM equivalence to be applied for checking the correctness of A2T abstraction or manual abstraction		N.A.
Improvement in the model checking and assertion-based verification processes	Hybrid integration of Dynamic and Static techniques in new Property Coverage Checker (PCC) tool. Ability to recognize redundant and vacuous properties.	Save more than 50% evaluation time w.r.t pure symbolic techniques	PCC out of the SYMBAD project
Verification qualification for TLM models	Design and Implementation of a mutation model suited for TLM	State the actual coverage of examples taken from OSCI TLM 2.0 library	N.A.
Automatic Test Pattern Generation for TLM/RTL designs	Integration of a deterministic EFMS-based ATPG engine	Achieve 100% transition coverage w.r.t. genetic-based engine reaching 50% on ITC benchmark suite	Laerte++ out of the SYMBAD project

University of Linköping			
Feature	Qualitative Measurement	Quantitative Measurement	Reference
Verification of IP block interaction	HIF interface link. Transactor based verification of IP blocks at different abstraction levels. Worst case execution time analysis based on simulation techniques		PRES
Hybrid Dynamic / Static Verification	Extension of the classical Petri nets to include Hierarchy, Timing notation and IP block interaction. Integration in PRES+	Double the coverage vs pure dynamic technique for an equal amount of verification time	PRES

University of Tallinn			
Feature	Qualitative Measurement	Quantitative Measurement	Reference
Abstraction / Refinement of HLDD models	Integration of manipulation techniques into HLDD libraries and HIF	Passing HIF regression test suite provided by Verona University	HLDD library at start of project
HLDD based coverage metrics	Definition of HDL coverage metrics for HLDDs. Inclusion of constraint solving in the process.	Passing HIF regression plus ITC benchmarks	DECIDER
HLDD based bounded model checking	Capturing PSL assertions in addition to HDL model and implementation of Assertion checking	Increase 100% speed for assertion check. Increase 50% the coverage obtained with HLDD-based test generation.	DECIDER
HLDD based coverage metrics	Introduce Modified Condition (MC) and Decision Coverage (DC) metrics	Provide an increase of 10% in coverage at the same simulation cost of classical code coverage	DECIDER
HLDD based description capacity	New HLDD minimization algorithms introduced	Reducing 50% (halve) the size of HLDD models. To be checked on ITC benchmarks	DECIDER

## 4. Revision of VERTIGO verification flow

---

The proposed verification flow is based on the flow of TEDA tool Assertain which is widely used in Industries ranging from Aerospace to Consumer electronics. The flow will be extended with the contributions of each partner. The purpose of this is that Assertain integrates dynamic and static results into an unified coverage database, thus allowing a view of the overall progress towards verification closure.

Closure is an iterative process and is based on a confidence level that all problems have been found. Closure can be seen as a funnel and tools must be used judiciously as we move down the funnel to closure. Additionally some tools simply can not cope with the width of the funnel at the start of the verification as the number of paths is too large, but come into their own as the funnel narrows.

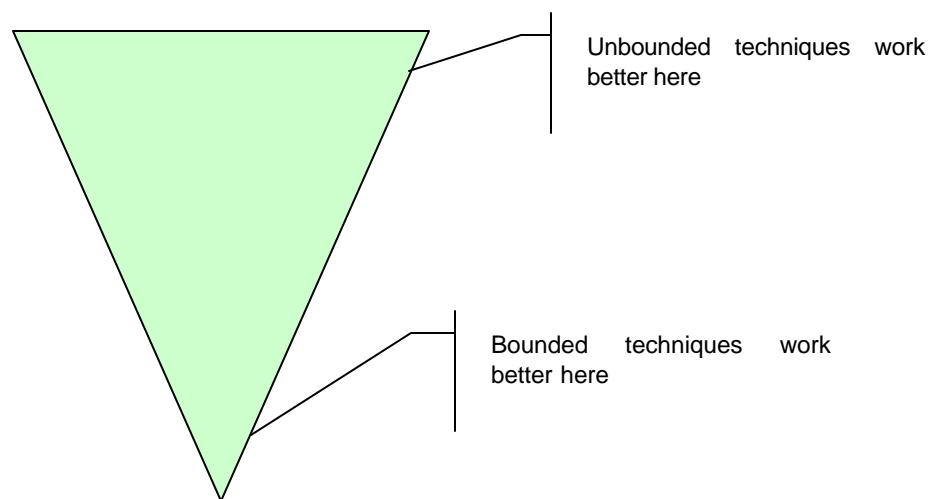


Figure 5 Funnel leading to Verification Closure

## 4.1 Description of Assertain verification flow

---

Assertain supports ABV using additional dynamic assertion coverage metrics: step, variable and fault coverage. Assertion step coverage measures control of the assertions, variable coverage provides a functional view of the structural coverage results and assertion fault coverage measures the coverage of the user's design by the assertions.

Assertain support RTL and behavioural Verilog and VHDL; SystemVerilog for assertion coverage. In order to promote interoperability and reuse, common input and output formats using the standard XML language will be used. This will allow a simpler integration with a well-defined format between Assertain and partner's tools.

There are three fundamental stages in Assertain:

Step One : Dynamic verification

Step Two: Static verification

Step Three: Verification closure

Assertain follows the same scheme, described below:

When a design is first read, customizable static rules are automatically verified on the RTL files. This includes not only common lint such as syntax and semantic checking, but also complex rules about design and coding style, naming conventions, documentation, etc. In addition, automatic formal rule checking can be performed on user's choice to exhaustively verify the structural consistency of various design constructs such as FSMs, busses, pragmas, arrays, etc. Finally, specific rules target the assertions to give the user a first feedback of the assertion density.

When the static checks are clean, engineers start their dynamic simulation with Assertain monitoring code and assertion coverage information using the industry's most complete set of metrics. Recorded coverage is displayed in the user-friendly GUI to provide critical feedback on how good the test suite exercises the design, the assertions and the functional coverage points. Tests grading and optimization can then be used to extract the most efficient regression test suite, from the coverage point of view.

When coverage reaches a point close to the sign-off target, Assertain enables users to run coverability analysis on the remaining uncovered items to get a more accurate view of what is coverable and what is not. In order to further speed up the convergence process toward sign-off, the requirement traceability links between specifications, assertions, RTL code and test benches allow functional bugs and coverage holes to be rapidly identified.

## 4.2 VERTIGO flow integration

The VERTIGO verification flow spreads on three design levels: system level, transaction level and RTL.

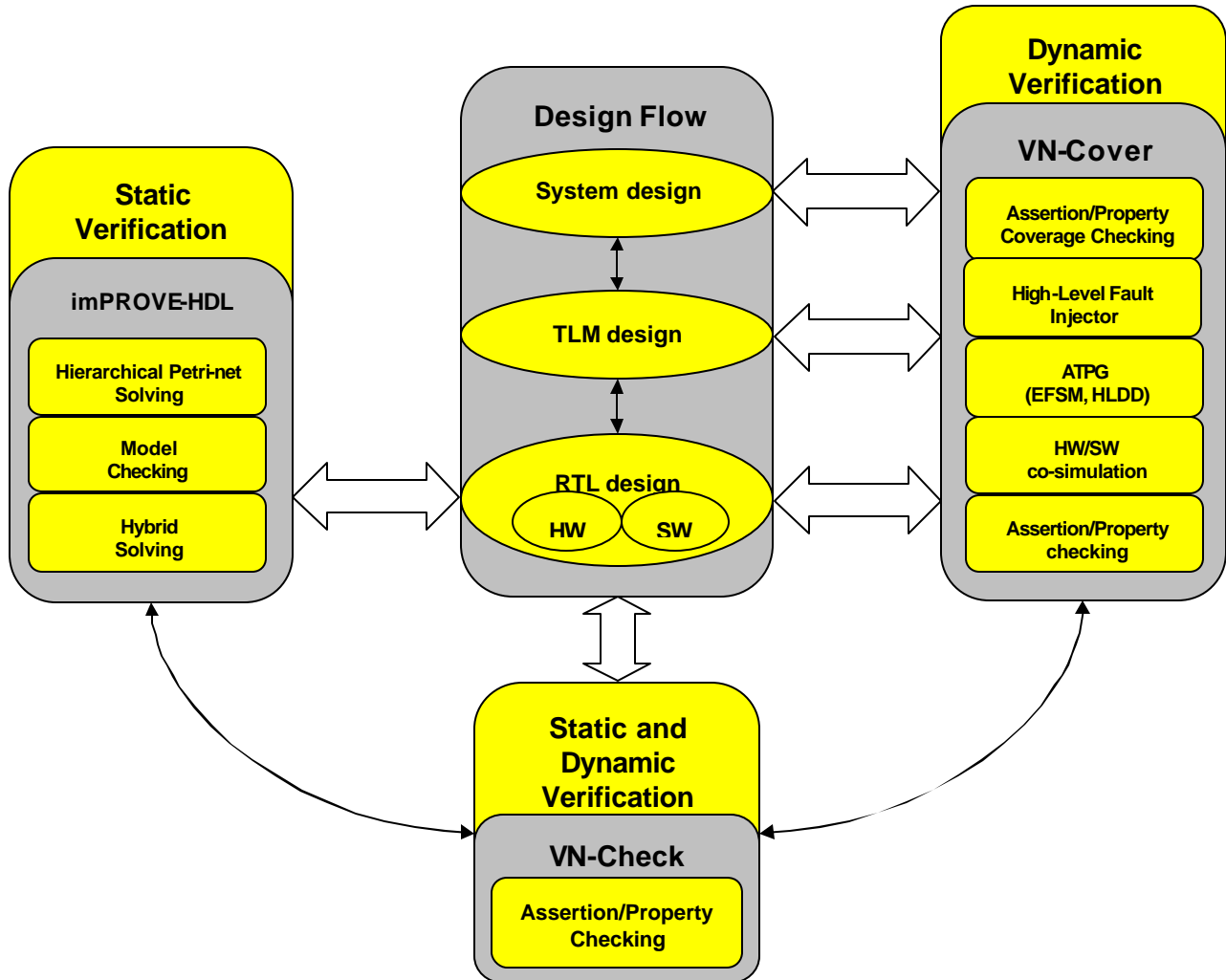


Figure 6 VERTIGO flow

The VERTIGO static and dynamic verification flow will include a complete integration of imPROVE-HDL and Assertain under a single GUI and coordinated in a clever way around RTL languages (System-Verilog and VHDL) and an ABV methodology. Dedicated links and associated methodologies to other partners tools will also be provided, as reported in Fig.6. Integration will be based on command line calls to these tools with data interchange via the HIF interface.

## 5. Partner contributions vs. building blocks

The relevant blocks of the environment are reported in the picture, including the expected contributions from partners.

