



STREP – IST 033709 VERTIGO

Verification & Validation of Embedded System Design Workbench

Deliverable D7.1 – Final Report

DUE DATE	30 November 2008		
ACTUAL DATE	30 June 2009		
START OF PROJECT	01 June 2006	DURATION	30 months
ABSTRACT	This deliverable reports the main results and achievements of VERTIGO project and a summary of dissemination actions.		
AUTHOR, COMPANY	VERTIGO Consortium		
WORKPACKAGE/TASK	WP7/T7.3		
FILING CODE	VERTIGO/Deliverables/08_D7.1		
KEYWORDS	Embedded System, Design Platform, TLM, SystemC, Assertion Based Verification, Assertion Coverage, Model Checking, SAT, High Level Decision Diagrams (HLDD), Hierarchical Petri Nets (HPN), Fault Injection, Extended Finite State Machines (EFSM), Transactor Generation (TGEN), Automatic TL Abstraction (A2T)		

Release	Date	Reason of change	Reviewer	Distribution
1	15/12/2008	First version	Consortium	Commission
2	02/02/2009	Second version after Commission's remark	Consortium	Commission
3	05/03/2009	Third version after Commission's rejection	Consortium	Commission
4	30/06/2009	Fourth version after Commission's rejection	STM, TEDA, AERIE, UNIVR	Commission

Partner acronyms:

STM	STMicroelectronics S.r.l. – Italy
TEDA	TransEDA Systems
AERIE	AerieLogic sarl
SOTON	University of Southampton
UNIVR	University of Verona
LIU	Linköping University
TUT	Tallinn University of Technology

Table of Contents

1.	VERTIGO positioning and consortium	1
2.	The bottleneck in the Electronics System Level Design for Embedded Systems on Chips and VERTIGO Objectives	2
2.1	VERTIGO objectives	4
2.2	VERTIGO goals	5
3.	VERTIGO implementation	6
3.1	Achievements	8
3.2	Summary of results	17
4.	Validation of VERTIGO tools & flow	22
4.1	imPROVE-HPK evaluation	22
4.1.1	Case studies	22
4.1.2	Evaluation items	23
4.1.3	Evaluation Execution	23
4.1.4	Results	25
4.1.5	Conclusions	26
4.2	Modelling and verification of the CRC block by means of Petri nets	26
4.3	Modelling and verification of CRC block with HLDDs	27
4.4	Coverability, Assertion Step Coverage, Assertion variable Coverage	28
4.5	Validation on the demo platform	29
5.	Main deviations from the DOW	33
6.	User evaluation and comparison vs the State of the Art	34
7.	VERTIGO – looking forward	36
8.	Dissemination of knowledge	37
9.	Glossary	38

1. VERTIGO positioning and consortium

VERTIGO has dealt with the development of technologies and tools to *verify* embedded systems built upon configurable platforms, within economical and technical constraints.

The VERTIGO program has involved several modelling, design and verification languages applied at TL and RTL, making it possible to support different methodologies for design integration onto a platform that provides a number of tools and models for IPs and communication. A major effort in VERTIGO has been to integrate dynamic and static verification around a coherent assertion based approach. Static techniques have been analysed in the context of Petri nets, Extended Finite State Machines (EFSM) and High Level Decision Diagrams (HLDD) in addition to the proven SAT approach. The Assertain product from TransEDA partner has served as a framework for collecting common coverage figures.

Several formal techniques have been investigated, that can contribute to different stages of the design flow modelling and verification (SW, TLM level, RTL level, module level, system level) and integrated with the simulation-based approach (dynamic verification). An Assertion Based Verification (ABV) method has been developed, that can be used both in static and dynamic verification with emphasis on TLM and the related metrics to measure their coverage. A SW/HW co-verification environment has been prototyped, capable of driving the development of SW routines for the purpose of the embedded platform test.

In order to ensure the project success, a strong focus was put on defining clear objectives. On the one hand, ST Italy provided several industrial test cases and two design flows – Top Down first and Bottom Up second. On the other hand, two deliverables were intended to provide precise requirements which were, desirably, to be achieved.

The whole project has been oriented toward realizing the desired design flows and achieving those objectives.

The VERTIGO consortium consists of 7 partners: STMicroelectronics S.r.l. (ST Italy), TransEDA Systems, AerieLogic, University of Southampton, University of Verona, Linköping University, Tallinn University of Technology. STMicroelectronics S.r.l. is the coordinator of the project and is in charge of the application of the project methodologies and tools in the validation of industrial designs. AerieLogic provides the core set of tools for the verification at RTL. University of Southampton provides the SAT formal technology, to be applied mainly at RTL. Tallinn University of Technology provides HLDD based algorithms and tools. Linköping University provides Petri Net based algorithms and tools. University of Verona provides EFSM based algorithms and tools, the enhanced Property Coverage Checker (PCC) and the overall methodology to coordinate Transaction Level (TL) and synthesisable level (RTL) verification. Finally, TransEDA Systems is in charge of evaluating the results of VERTIGO for commercial deployment, by selecting the most promising techniques developed by the academic partners. TransEDA provides the core set of dynamic verification tools used by design companies worldwide.

VERTIGO started on June 1st 2006 for a duration of 30 months.

The project coordinator is:

STMicroelectronics S.r.l.
Via Olivetti, 2
20041 Agrate Brianza
Italy
Contact name: umberto.rossi@st.com

2. The bottleneck in the Electronics System Level Design for Embedded Systems on Chips and VERTIGO Objectives

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.

Embedded systems, implemented in a single chip, can compete in terms of logical complexity with the more sophisticated ASICs or general purpose chips, thanks to the implementation of very powerful CPU cores – e.g. ARM, PowerPC and a number of very different specialized peripheral block.

We see a convergence in the specification and design methodology at System Level that implies the use of high level specification & design language for an application that will run in SW and HW. Also HW Design Languages (HDLs) have raised the level of abstraction, e.g. consider the popular System C and System Verilog; hence chip designs suffer the same fate as software code with highly labour intensive debugging cycles.

In the SW domain, refinement & verification techniques to deal with the increased complexity are available; In the HW domain synthesis & verification techniques to deal with huge number of logic gates are available.

But a gap still exists between the SW and HW domains, that is at the very moment when a SW-like high level description has to be translated into a synthesizable HDL that is later fed to the physical implementation process. Such a translation is often manual or assisted by some automatic flows in specific cases – e.g. data path. A fundamental problem is to ensure the consistency between the verified high level specification and the HDL description in the Register Transfer Level (RTL) format that constitutes the entry point to the physical implementation.

Example of approaches used to fill the gap can be found in several commercial products.

With Esterel, for instance, a powerful System level specification language is available, that can be both executed and formally verified. Thanks to the rigorous formal structure of Esterel language, the description can be directly translated to (synthesizable) RTL. This approach is great for control related parts and works fine for HW blocks (IPs) of reasonable size – say few 10's K gates. In order to facilitate the integration with other design and verification environments a number of language translators are provided, including the commercial HDL languages and SystemC.; an example is provided by the “joint solution” for integrating Synfora's algorithmic synthesis and Esterel's controller design capabilities.

A number of different commercial products are targeting the assembly of the full SoC based on appropriate descriptions of building blocks. The main target is the provision of a SW/HW design & verification environment that allows to identify the optimal SW/HW partition for an embedded application. The availability of models, e.g. in SystemC language or in a proprietary language, is the key point to gain access to this methodology.

What we saw in the commercial flows, at the time VERTIGO project was conceived, was either a smooth path for the implementation of a single HW IP block, i.e. the case of Esterel synthesis or algorithmic synthesis, or assembly flows based on models of IP block that must be proven equivalent to the existing RTL; in the second case, the only solution provided for verification is provided by massive simulations.

At STMicroelectronics such flows have been applied at specific segments, e.g. set-top-box systems, imaging systems, wireless systems and other consumer products; in those cases standard applications and benchmarks exist to drive the system simulation and finally verify the SoC.

VERTIGO project was conceived for different application segments: Office/Computer applications and Automotive applications. In both these cases, there is no access to the

customer proprietary final application, therefore there are no general directives that can be used to verify the correctness of the final SoC. As a consequence, in Computer and Automotive segments there has not been enough motivation, in the past, to move away from RTL for system specification & verification. An example of Configurable Platform for Automotive applications is reported in Figure 1.

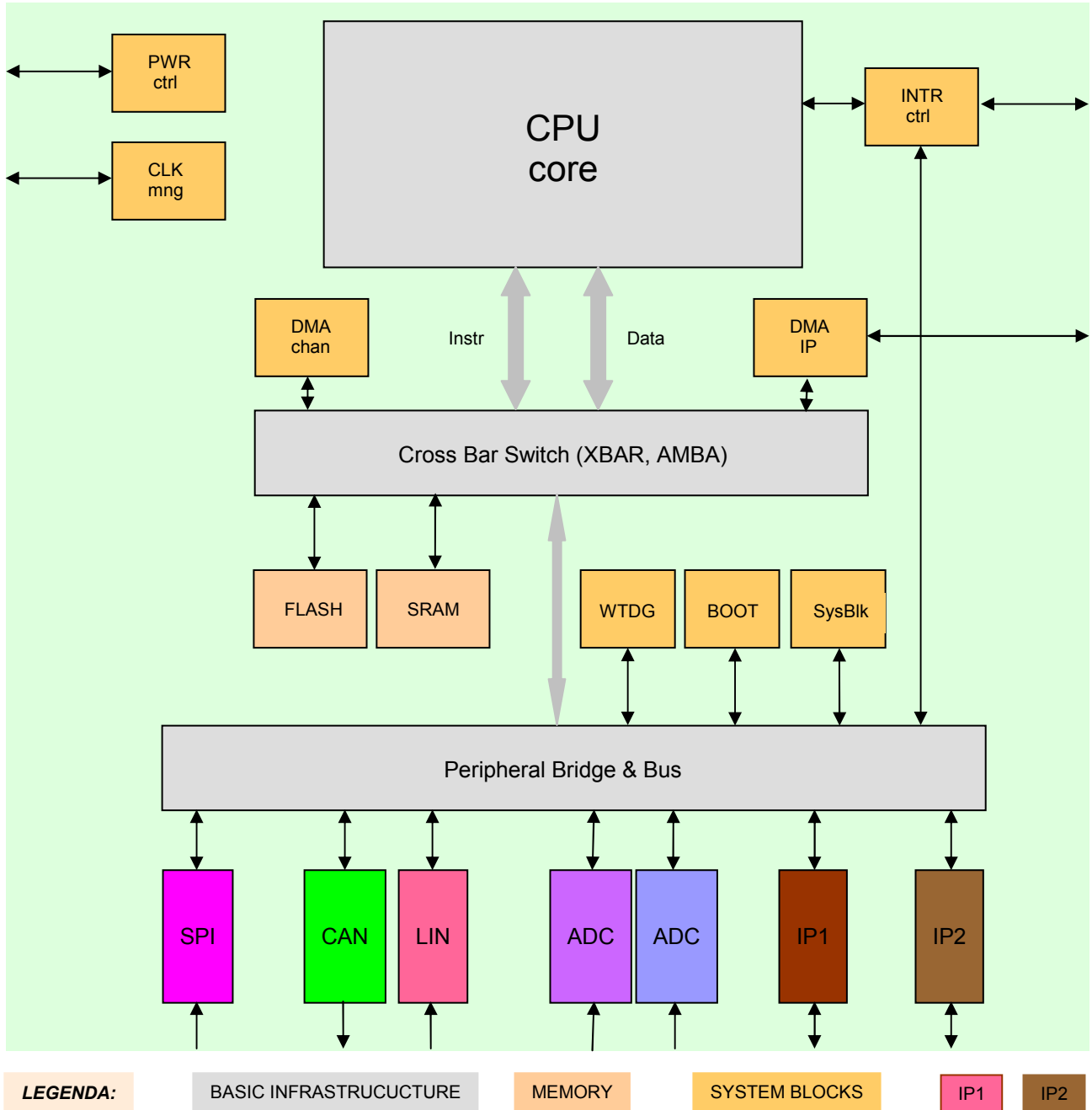


Figure 1: Configurable Platform for Automotive applications

The system infrastructure is built upon 16/32 bit architecture. Customization affects part of the system blocks, the size of SRAM and Flash memories, IP blocks for customer specific needs, e.g. the ADCs, and finally the set of standard peripherals that are used network the SoC in the automotive systems, e.g. by means of SPI, CAN and LIN protocol bases

communication. The complete platform is being currently designed at Register Transfer Level (RTL), which is the entry point for an synthesizable HDL description.

Typical problems that are faced during the product design cycle are reported.

- 1) The platform is *configurable*, as it allows to build a number of different variants of application customized both in SW – via the Flash and the Boot contents – and HW – via the assembly of specialized IP blocks. It would be convenient for the customer to have at disposal an high level, e.g. Transactional Level (TL) Model that allows to examine in advance the capabilities of the product being assembled and start developing SW once the specification is made – therefore reducing the customer application development cycle. An alternative to the TL Model is represented by HW emulation, but this is typically more expensive and more difficult to deliver to the final application developers.
- 2) The TL models of the platform components must be validated vs the existing RTL, and this poses a significant verification problem. The verification problem can exist both in Top Down flow, e.g. when a new set of IP blocks is being designed starting from a TL specification, or in Bottom Up flow, e.g. when a set of existing IP block has to be enriched with TL models in order to be exported in the TL platform.
- 3) The assembled product must be designed and verified without knowing the customer application(s). No example applications/benchmarks are available, therefore the verification problem is to be addressed by considering the architectural consistency uniquely.
- 4) The usage of protocols for block communication requires a number of checks that can be made automatic thanks to the use of standard assertions. An important feature is to improve the reporting of assertion coverage by means of simulation and to improve the capabilities of algorithms when static formal verification is applied. Simulation based checks provide a measure of coverage – according to some chosen metrics – whereas Formal checks provide by definition 100% coverage.

2.1 VERTIGO objectives

Given the problem stated above, the objectives of VERTIGO project can be summarized in the following lines:

development of a systematic methodology to combine a simulation-based approach (dynamic verification) together with *formal methods* (static verification) integrated into an IP-cores and platform based design flow, for the purpose of producing a SW kit applied to the platform validation.

Such system level based design verification flow must solve three main problems:

- Verification of the correct interaction between all IP-cores and of the system in the *networked* environment, driven by coverage metrics
- Production of a *SW layer* for the purpose of the *embedded platform* test
- Verification of the correct *modelling* of system-level IP-cores and their correct mapping into RTL descriptions, driven by formal verification

An effort is necessary also for the SoC modelling techniques, applied at System (specification), Transaction (Communication) and Register (RT, Implementation) *Levels*. 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.

2.2 VERTIGO goals

The original *VERTIGO* goals were:

1. develop several formal techniques that can contribute to different stages of the design flow modelling and verification (SW, TLM level, RTL level, module level, system level) and integrate with the simulation-based approach (dynamic verification).
2. develop an Assertion Based Verification (ABV) method that can be used both in static and dynamic verification with emphasis on TLM and the related metrics to measure their coverage.
3. prototype a simple SW/HW co-verification environment capable of driving the development of SW routines for the purpose of the embedded platform test

In order to reach the goals, four key targets were identified at the beginning of the project:

- 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

3. VERTIGO implementation

The profiles of the 7 partners in VERTIGO provide the capabilities to address different modelling and verification techniques at specification and implementation levels. Capabilities are provided by the partners' developed tools that are reported in the table.

Tool	Purpose	Provider	At start	Development during VERTIGO
AIF/HIF	API & tools for data HDL translation & manipulation	UNIVR	Available (AIF) Prototypal (HIF)	Extension for supporting conversion from SystemC TLM 2.0 and Verilog to HIF and vice versa, and for manipulation of EFSM, PRES+ and HLDD models.
Laerte++	Random-based ATPG	UNIVR	available	Definition and implementation of a new pseudo-deterministic engine based on EFSM and CLP.
PCC	Property completeness evaluator	UNIVR	available	Definition and integration of vacuity and redundancy analysis features
ACIF	RTL fault injector	UNIVR	available	Extended with a new TLM fault model
EFSM gen.	Extraction of Extended FSM	UNIVR	available	Definition of a new EFSM model (S ² EFSM) which allow a more uniform traversing of the state space for ATPG
A2T	RTL to TLM abstractor	UNIVR	conceived	Implementation of the tool, and its extension to support SystemC TLM 2.0 coding style
DGEN	Device Driver Generation	UNIVR	conceived	Implementation of the tool, SystemC TLM 2.0-compliant code. Application of generated driver for effective reuse of testbenches
TGEN	Automatic Transactor Generation	UNIVR	prototypal	Extension for generating SystemC TLM 2.0-compliant code. Application of generated transactor for effective reuse of testbenches and assertions
Qemu	Integrate SystemC descriptions with Linux based emulation platform	Fabrice Bellard	available	Qemu used to implement the final platform
HSN	HW/SW/Network co-simulation	UNIVR	available	Support multiple-ISS & timing accurate co-simulation
Decider	HLDD based verification – dynamic & formal	TUT	available	Capturing RTL & TLM descriptions in commercial HDLs. Temporal assertions. Model manipulation in order to support coverage Analysis. Integration with Laerte++
SAT solvers	Formal analysis	SOTON	available CQuest	New solver and techniques: "Hinotos", PBO, MaxSAT, MUS

MCSAT v1.0	SAT formal verification	SOTON	available	MCSAT v2.0 with integrated UMC (Unbounded Model Checking) capabilities
PRES+	Petri net modelling and verification	LIU	available	Capture Real Time properties, Hierarchy, IP interactions at mixed abstraction levels. Use of static analysis to boost simulation coverage. Worst-case execution time based on simulation.
imPROVE-HDL	Model Checker	AERIE	available	New architecture & algorithms PSL & SVA assertion support Graphical configuration and Enhanced report generation
VNCover VNCheck ImPROVE-HDL	Discrete Dynamic Verification Discrete Rule Checker Model Checker	TEDA AERIE	available	Integration by TEDA of a new Static & Dynamic Analyzer named "Assertain" Assertain uses ImPROVE-HDL for coverability & advanced lint rules
A/F check	Automatic extractor for Alternate Function check at the SoC periphery from IP-XACT format	ST	conceived	Check the IP connectivity, through the SoC periphery, toward the extern.

Modelling and description have been based on SystemC, VHDL, Verilog and SystemVerilog. Concerning Assertion Based Verification (ABV), both SystemVerilog Assertion (SVA) and Property Specification Language (PSL) assertions have been used. Therefore, the first step was to implement a coherent infrastructure that comprises such a heterogeneous set of languages. The solution was found in the HIF format, that is an improved and extended version of the AIF format developed in the IST-FP5-SYMBAD project. See figure 2.

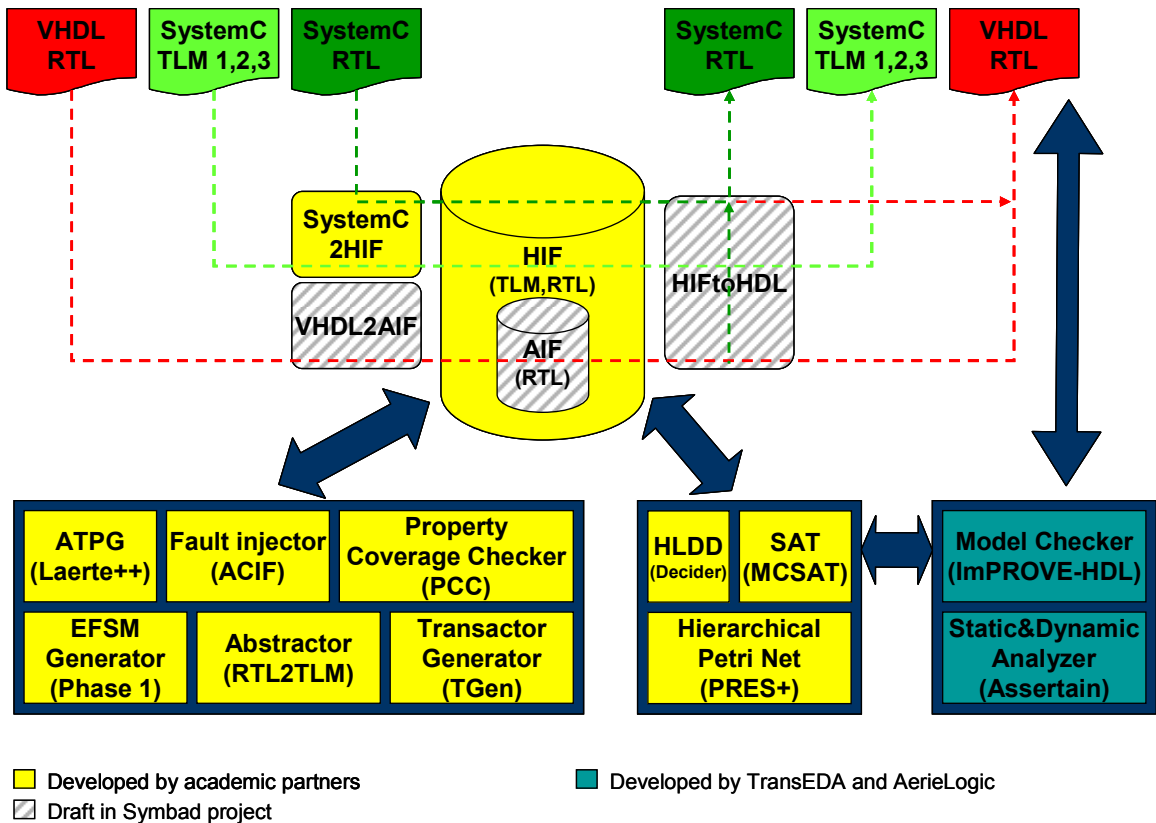


Figure 2: Language integration in VERTIGO

Each academic tool provider developed, at the beginning of VERTIGO, dedicated translators from/to the HIF format. Existing specific translators from academic tools from/to commercial languages/tools existed, like MCSAT \leftrightarrow improve-HDL, that were re-used. Commercial partners, TransEDA and AerieLogic, gained access directly through commercial languages. During the last 6 months of VERTIGO a new SystemC to HIF translation tool was developed by Verona University for supporting both TLM 2.0 and RTL, due to the intervening TLM 2.0 standard.

A major effort in VERTIGO was to integrate dynamic and static verification around a coherent assertion based approach. Static techniques have been analysed in the context of Petri nets, Extended Finite State Machines (EFSM) and High Level Decision Diagrams (HLDD) in addition to the proven SAT approach. The Assertain product from TransEDA has served as a framework for collecting common coverage figures.

3.1 Achievements

Project results and achievements are grouped according to the four key targets identified at the beginning of the project.

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

This target implies the use of a Top Down design approach where the System is refined from a High level description capturing process concurrency as well as timing aspects; for this purpose the Petri-net based Representation for Embedded Systems was chosen. In order to set a starting point for simulation and (manual) synthesis the SystemC TLM 2.0

was identified. This flow has been applied at block level only. The behaviour of a block in the demo platform has been formally specified, including timing properties for the integration with the rest of the system., by means of Petri-nets. Model Checking verification has been performed by means of Linköping's PRES+ tool. In particular the property of liveness has been proven for this block.

KT1 IMPLEMENTATION DETAILS

This target has dealt with Petri Nets. In particular the goal was the extension of the basic Petri Nets model to include hierarchy and timing notations. This allows to model the interaction of complex IP-cores and the verification of timing properties of the global system.

Extension of the classical Petri net modelling techniques included:

- Hierarchy to reduce complexity
- Timing notations to facilitate analysis of properties
- Interaction of complex IP-cores
- Incremental verifications
- Verification at different abstraction levels
- Analysis of worst case execution time via simulation

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

Also this target has been limited at the block level. Instead of considering generic properties, the attention was set on the flow of IP abstraction from RTL to TL. Equivalence checking RTL to TL is achieved by means of a correct-by-construction flow upon which the abstraction tool A2T is based. A2T tool is still prototypal at the end of VERTIGO, but it is able to translate a RTL VHDL description (without hierarchy levels) into a SystemC TL.

KT2 IMPLEMENTATION DETAILS

A2T

A definition of equivalence based on events and sequence of events has been given, suitable for the TLM vs. RTL context. It can be applied to:

- TLM vs. RTL context
- prove the correctness of abstraction of refinement methodologies

Rules defined for automatic abstraction from cycle-accurate RTL designs to transaction-based TLM PV descriptions, involving Clock abstraction and Collapsing of RTL states

Implementation of A2T tool and application to three test cases provided by ST partner.

The core technology used by A2T tool is based on EFSM based techniques, although EFSM have been used also as computational model for ATPG and for the generation of transactors for TLM to RTL refinement

EFSM

Extension of the EFSM generation tool, named Phase1, to support the extraction of EFSM models from complex multi-process designs

The EFSM model has been analyzed and a classification of the different kinds of transitions that can appear in the EFSM have been performed with respect to the hardness to be traversed

According to the previous classification, a tool for generating a particular kind of EFSM (S²EFSM) starting from an HDL description has been developed, which allows a deterministic ATPG to traverse the state space of the design under verification in a more uniform way

A backjumping-based algorithm has been implemented to deterministically traverse EFSMs without requiring their stabilization. The backjumping-based strategy, avoiding stabilization, reduces the risk of state explosion, and it improves the traversability of unstabilized EFSMs

Constraint logic programming-based strategies and SAT-based strategies have been implemented as an alternative to backjumping to fire the guards of EFSM transitions in forward mode

KT3.to integrate dynamic and static verification around a coherent Assertion Based Verification approach at RTL

Several features have been developed to achieve this target: Assertion Variable Coverage – i.e. telling what signals in a passing assertion sub-expression are actually toggling, Assertion Step Coverage – i.e. efficiently and accurately counting the number of steps exercised for a temporal assertion, Coverability – i.e. reporting what lines of code are not reachable and cannot be covered by dynamic simulation.

Further investigation regarded the use of HLDD technique to implement another measure of code coverage able to highlight when the effect of VERILOG non-blocking assignments do not propagate in downstream logic.

The fault injection technique has been introduced to validate the coverage of patterns that are used to validate RTL description vs TL description equivalence.

Finally, automatic transactor generation – transactor is a protocol translator that allows the communication between TL and RTL blocks – has been provided to efficiently build a model of platform where TL blocks and RTL blocks co-exist.

KT3 IMPLEMENTATION DETAILS

This key target includes several kinds of core technology developments.

SAT & hybrid techniques

Development of new techniques for SAT, of new algorithms for extensions of SAT, and of SAT-based hybrid solvers. This work serves as the foundation for the development of static verification approaches, namely model checking solutions:

- MCSAT with UMC capability

- New version of MCSAT integrated into imPROVE-HDL

- BMC-like approach for model checking B models

- New robust SAT solver Hinotos
- Improved pseudo-Boolean optimization algorithms
- New algorithms for Maximum Satisfiability
- New algorithms for extracting MUS's
- MaxSAT to allow for multiple property checking
- Hinotos to replace SAT solver in MCSAT

imPROVE-HDL

New BMC algorithm replaces the old one – several upfront reductions and optimizations – significant reduction in memory consumption and significant speed-up

Integration of an UMC capability

Recoding of imPROVE-HDL internal flow

Completely new internal architecture – RTL replaces a low-level format as common internal language – Most Java code replaced with C/C++ more efficient code.

Specification of the imPROVE-HDL report

HTML format, includes several functional coverage

imPROVE-HPK

Assertions checking the interfaces of a block, and especially bus-protocol assertions, are a corner stone for a strong Assertion-based methodology. The functionalities of the protocol are well defined. The properties, coverage scenarios as well as the formal environment that model the protocol can all be prepared and validated once for all. They can be reused from one verification phase to another or from one project to another with minimal effort. The tedious assertions debugging loop and formal environment debugging loop can be avoided.

Libraries of assertions, called Hardware Protocol Kits (HPKs), targeting the Amba protocol (AXI, AHB and APB) and the OCP protocol have been provided by AerieLogic. These HPKs work with a specialized version of imPROVE-HDL called imPROVE-HPK. All imPROVE-HDL features work with the HPK, especially PSL and SVA can be added on top of the HPK if needed.

The user has to fill-in a parameters file for each design's interface. This file includes a description of the interface HPK signal names to link to the design's own signal names, and a description of the protocol features supported by the interface. To further ease the completion of this file and to avoid potential mistakes or misunderstandings on some parameters, a Graphical Interface has been developed.

Once the parameter file is available, the process is mostly automatic. The formal environment and the properties and coverage scenarios are automatically instantiated from the HPK in accordance with the chosen parameters. A regression script is automatically created that enables to launch the entire list of properties and coverage scenarios.

HLDD

- Definition of a new, Temporally extended HLDD (T-HLDD) model
- Developed HLDD model manipulations for code coverage analysis
- Implemented HLDD-based assertion checker
- Implemented code coverage measurement tool using HLDD models

Development of efficient HLDD-based coverage metrics based on high-level decision diagram manipulations

Definition of an HIF-based interface for converting EFSMs towards HLDDs and vice versa in order to allow the integration between Decider and Laerte++

Integration of EFSM-based and HLDD-based ATPG engines into Laerte++

PSL assertion simulation & Assertion coverage

Fault modelling & injection

High-level fault models are used to perturb the behaviour of complex IP-cores

The well-known RTL bit coverage fault model has been analyzed to evaluate its adaptability to be adopted at TLM. In this context, drawbacks of directly applying bit coverage at TLM have been identified

A new TLM fault model derived from the bit coverage have been proposed. The new fault model will differently address the TLM communication protocol of the DUV, modelled by means of an S²EFSM, and the TLM DUV functionality, modelled by means of high-level functions. This process involves the definition of "mutant".

Definition of mutants for TLM constructs and of a new mutation model for TLM designs:

- TLM communication primitives formalized by EFSM

- Mutations involve the EFSM of TLM primitives

- Identification of relations between mutations and typical design errors

Automatic injection of TLM mutants

Extension of ACIF

Mutation analysis can be used for measuring the quality of test suites defined for verifying SystemC descriptions.

TLM mutation model applied for Functional verification of IP cores, Assertion cover checking in HW-SW co-simulation systems and Equivalence checking between TLM and RTL models, based on transactors and simulation

Definition of an interface for the exchange of fault lists between Decider and Laerte++

PCC property completeness evaluator

Definition of:

- a technique to reduce the size of property set to be checked by removing redundant properties. In particular, the approach exploits fault simulation to identify properties which are logical consequence of other properties in the same set

- a technique to identify vacuous properties, i.e., properties that pass vacuously. The approach relies on fault simulation of checker automatically generated from the properties to be analyzed

Transactor generation

TLM to RTL assertion automatic translation

Use of TLM to RTL transactors – to be used for both static and dynamic verification

TLM satellites translated into RTL - to be used for both static and dynamic verification

Assertion coverage metrics

Definition of precise assertions coverage metrics for Dynamic verification

Step coverage

Variable coverage

Fault coverage (fault on the design – fault on the assertion)

Integration of SystemVerilog in Assertain

Integration of VN-Check (rule checker) and VN-Cover (code coverage) into Assertain

Integration of imPROVE-HDL (formal tool) into Assertain and provision of Coverability analysis

Assertions reduction analysis in PCC

HLDD-based assertion coverage

HSN HW/SW/network simulator

Initially developed in the ANGEL project, to support assertion-based verification. VERTIGO extensions concern implementation of a way for integrating checkers into HSN, to apply assertion-based verification to check the correctness of HW and SW modules during the co-simulation. Moreover, HSN HW/SW co-simulation environment was refined by adding the concept of time in the instruction set simulator (ISS) where the SW runs. Finally, the tool has been extended to support co-simulation between SystemC and multiple instances of ISS. This feature was mandatory to allow the integration of Laerte++ into the HSN environment, since Laerte++ needs two ISS instances work in parallel, one connected to the SystemC fault-free design and the other connected to the SystemC faulty design

KT4.to federate partner provided techniques in the design of a novel static verification environment operating at RTL for IP verification and IP interaction

A premise to reach this target was the definition of a common format to all of the VERTIGO developments, achieved by means of HIF – shown in previous figure 2.

Integration of academic tools and industrial tools relying on the HIF format and the related suite of tools has been performed. Then, a systematic methodology to automatically generate SystemC checkers linkable with Laerte++ has been implemented. This allowed a mixed static/dynamic strategy for assertion-based verification. Thus, PSL properties used by AERIE and TEDA tools for static verification can be converted into checkers and provided to Laerte++ for dynamic verification

Automatic Property Generation for protocol interfaces have been provided, together with the capability of formally proving the produced property, either exhaustively or within a bounded limit. An environment for extracting Formal Coverage metrics has been provided.

Automatic Test pattern Generation has been implemented based on fault injection coverage metric.

KT4 IMPLEMENTATION DETAILS

Integration of developed prototypes has been performed in two ways: *tight* integration and *networked* integration.

Tight integration of Laerte++ and Decider for ATPG by combining EFSM, CLP and HLDD

Tight integration into new product set Assertain of:

- VNCover
- VNCheck

- o imPROVE-HDL

An example of Assertain Verification Closure report is illustrated in figure 3

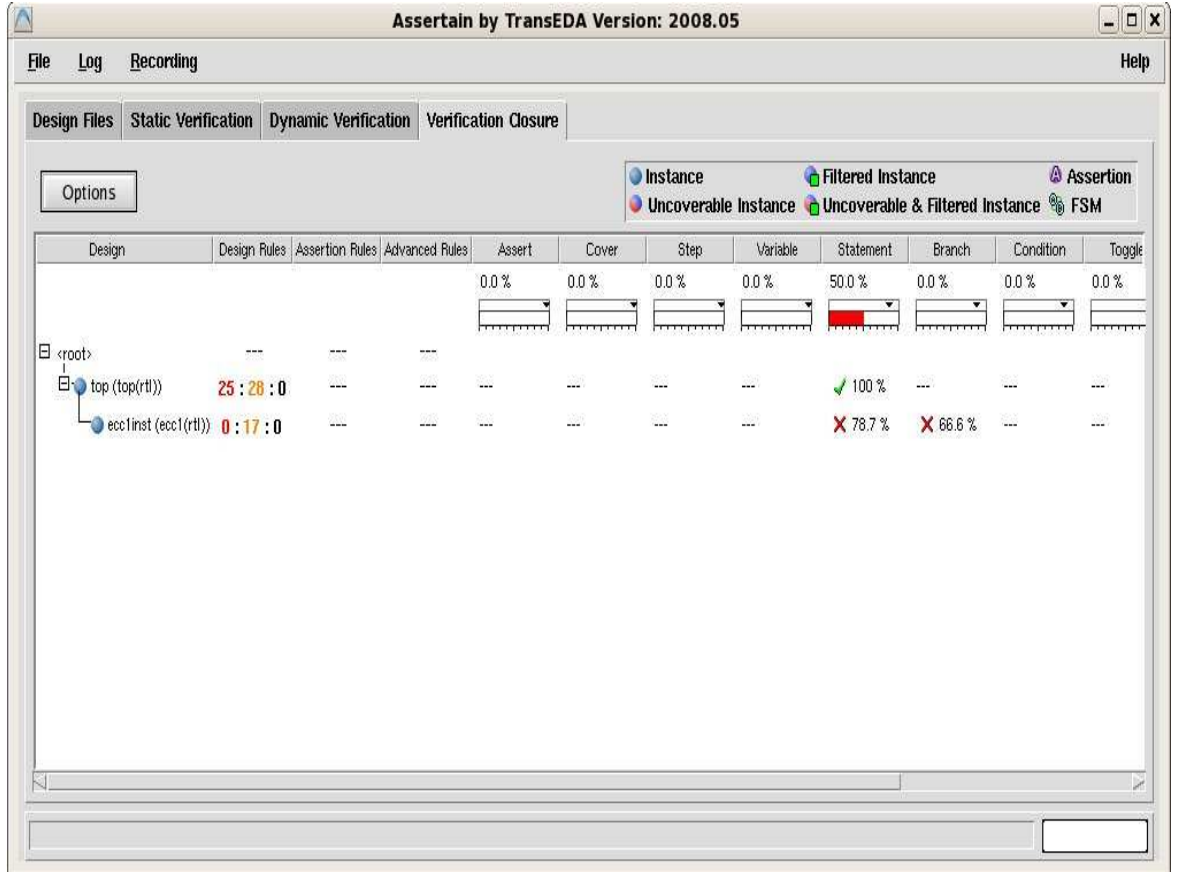


Figure 3: Verification Closure with Assertain tight integration

Networked integration for Assertain with

- University of Verona's Laerte++
- Tallinn University of Technology's HLDD Assertion & Coverage Analyzer
- University of Southampton

TransEDA have produced an object orientated integration engine based on Linux Red Hat EL5, PHP and MySQL which allows the test runs of each tool encapsulated as a HIF file to be uploaded and rendered as a Flex GUI – http based. This allows the work of each VERTIGO contributor to be integrated.

Test runs can be grouped and charted to show verification closure

An example of Verification Closure report including academic tools is illustrated in figure 4

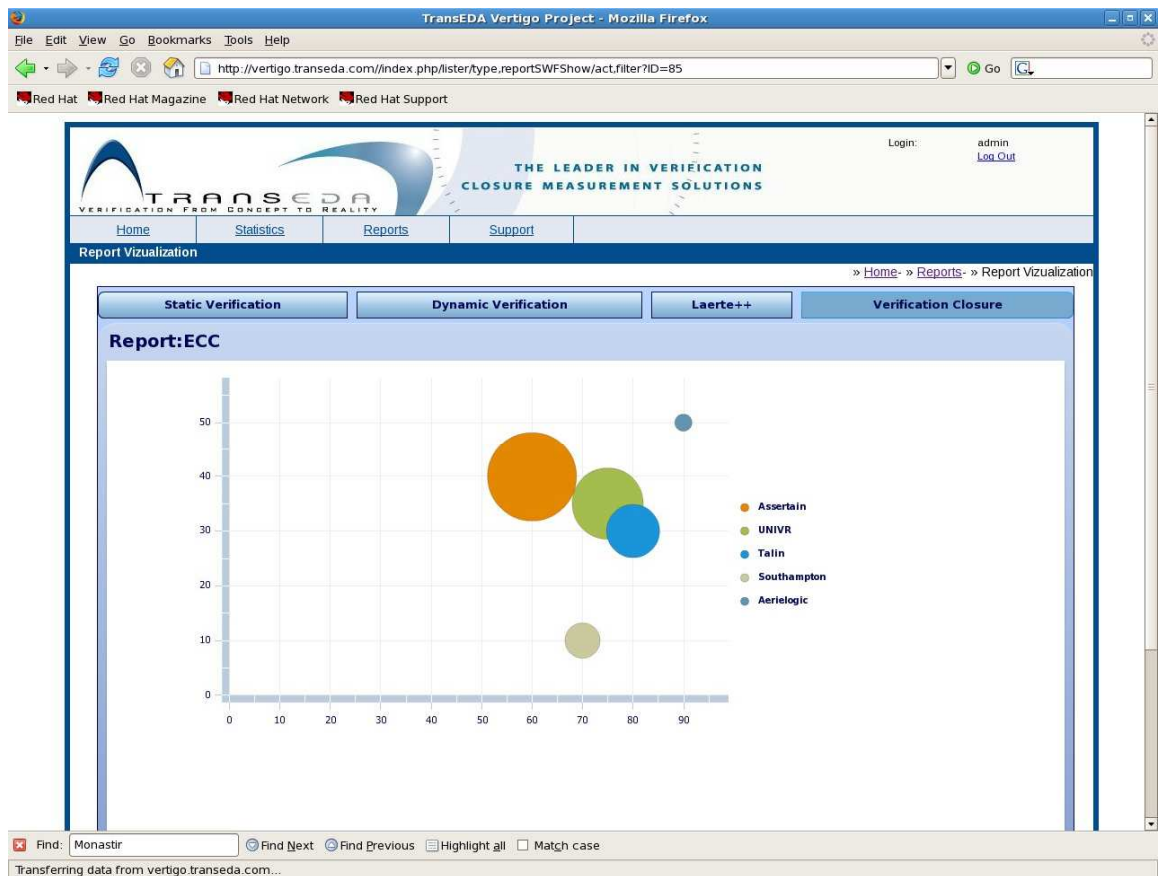


Figure 4: Verification Closure with Assertain networked integration

In the networked integration the results of each verification run are stored in an object orientated database. The entire verification run set can then be retrieved and graphed to focus in on design issues.

Networked tool integration implemented architecture is reported in Figure 5.

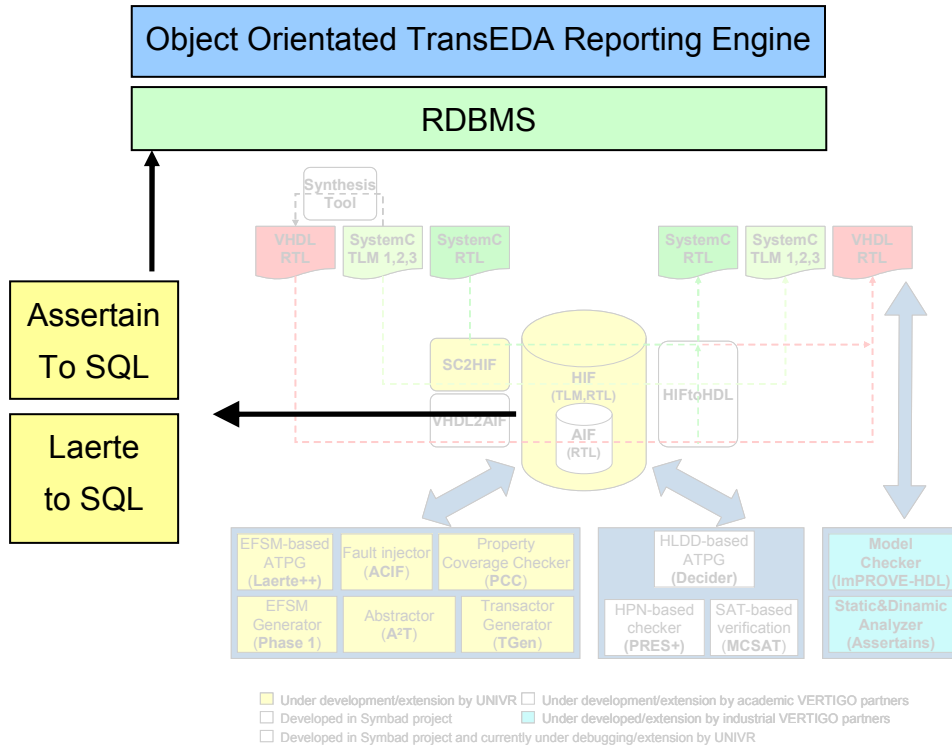


Figure 5: networked tool architecture

3.2 Summary of results

One of the underlying means to achieve the Vertigo project was to enhance the formal verification algorithms used for static analysis. This work focused on three topics:

- capacity:

 - size of the designs that can be handled

- depth for bounded model checking:

 - number of cycles that can be exhaustively searched to produce a bounded proof

- efficiency for unbounded model checking:

 - percentage of properties for which an unbounded proof is found that is valid whatever the number of cycles is (in opposition to a bounded proof that is valid only on a certain number of cycles after reset)

The reference tool used for the comparative study is the property checker imPROVE-HDL provided by AerieLogic at the beginning of the Vertigo project, and believed to be a very strong SAT based static checker at that time. The experiments were performed on several tens of industrial blocks ranging from 10K gates to 400K gates, and several properties per block (total of 300 properties), on a 1.2Ghz computer with 4Gb of memory.

The memory consumption was reduced by a factor of three during the project, enabling to process for instance the 400K gates problem on several tens of cycles, which was impossible with the reference tool. On average, the depth of search for bounded proof at iso time was enhanced by 50% (average 80 cycles exhaustive search instead of 55 cycles with the reference tool), and the speed at iso depth was enhanced by a factor of two (43K seconds of run instead of 79K seconds with the reference tool).

For unbounded proof, the experiments showed an increase of 50% in the number of properties proved correct (70% of total instead of 49% with the reference tool). However the time consumption is far greater (average 30mins instead of average 2min for the reference tool).

From the perspective of SAT technology implementation, the main achievements can be summarized in this way, over test suites provided by AerieLogic and University of Southampton:

- New BMC (Bounded Model Checking) capability reduced memory consumption (x3) and significantly speeded up execution (x2 – x10)

- New UMC (Unbounded Model Checking) capability increased infinite proof capacity (x2)

PER PARTNER RESULTS

TransEDA (TEDA)			
Feature	Requirement	Result	Reference
Integration of partner tools into Assertain in order to achieve verification closure through VERTIGO flow	Integration of enhanced imPROVE-HDL. Provision of links to academic partner tools for a loose integration with Assertain	Integration of tools within a Flex GUI from academic and industrial partners.	Assertain 2008.05
Integrate Coverability feature	Increment dynamic coverage by means of formal proof	Achieved 100% line coverage on ECC, CRC, SPI blocks + internal testcases	Assertain 2008.05
Enhance assertion coverage metrics	Use of formal analysis for Assertion Step Coverage and Assertion Variable Coverage	Achieved 100% ASC and 100% AVC on assertions used with CRC, internal testcases	Assertain 2008.05

AerieLogic (AERIE)			
Feature	Requirement	Result	Reference
PSL and SVA	Support of simple subset for Verilog, SV and VHDL designs	Completed unit testing + several industrial testcases	imPROVE-HDL before VERTIGO
Capacity	Reduce memory consumption by 2/3	Achieved on 45 industrial testcases	imPROVE-HDL before VERTIGO
Bounded proofs	Reduce solution time by 4/5	1/2 X achieved on 45 industrial testcases	Average 1/2 achieved
Full proofs	Improve by 2 factor full proofs	Achieved on 45 industrial testcases	imPROVE-HDL before VERTIGO

University of Southampton (SOTON)

Feature	Requirement	Result	Reference
SAT based Model Checking with BMC and UMC proofs	Integration of new techniques in MCSAT and new SAT solvers	See imPROVE-HDL enhancements	Competing SAT based model checkers Integration into improveHDL Journal publications SAT 2008 paper
SAT encodings	Identified properties of some SAT encodings	Improvements in the run time of SAT solvers when properties are exploited	CP 2007 paper
Model enumeration in PB and integer domains	Developed new algorithms for model enumeration in pseudo-Boolean and integer domains	These techniques can be applied in model checking at higher levels of abstraction	SAT 2006 paper
Unsatisfiability proofs	Used in unbounded model checking and new MaxSAT solvers	Improvements in imPROVE-HDL MaxSAT solver	Integration into improveHDL SAT 2006, DATE 2008, SAT 2008 papers
New MaxSAT Algorithms	New unsatisfiability-based MaxSAT algorithms	Among best performing solvers. Can be used for multiple property checking	DATE 2008 and SAT 2008 papers

University of Verona (UNIVR)			
Feature	Requirement	Result	Reference
HIF intermediate format for VERTIGO common description language	Support of TLM/RTL and SystemC/VHDL languages with related translators	Fixed all known bugs out of AIF of SYMBAD project (Codegen tool)	HIFSuite v1.0 developed out of SYMBAD project
Abstraction Refinement of HDL descriptions and Assertions	Integration of EFSM, A2T and TGen in a consistent flow	Achieved IP abstraction – ECC, CRC – and verify interface correctness in 1 day	Manually requires 1 week

Automatic Test Pattern Generation for TLM/RTL designs	Integration of a deterministic EFMS-based ATPG engine	Achieved 100% transition coverage w.r.t. genetic-based engine reaching 50% on ITC benchmark suite and VERTIGO reference platform.	Laerte++ out of the SYMBAD project
Verification qualification for TLM models	Design and Implementation of a mutation model suited for TLM	Stated the actual coverage of examples taken from OSCI TLM 2.0 library	N.A.
Improvement in the model checking and assertion-based verification processes	Hybrid integration of Dynamic and Static techniques in new Property Coverage Checker. Recognize redundancy and vacuity	Saved more than 50% evaluation time w.r.t pure symbolic techniques	PCC out of the SYMBAD project
HW/SW co-simulation	Enhancement of the HW/SW co-simulation environment including timing aspects and multiple instances of ISS. Automatic generation of device drivers	To simulate HW and SW before actual HW is available. Applied in the reference platform demo	HSN tool out of the ANGEL project

University of Linköping (LIU)			
Feature	Feature	Feature	Feature
Timing properties	Timing properties	Timing properties	Timing properties
Performace	Performace	Performace	Performace
Mixed static & dynamic techniques	Mixed static & dynamic techniques	Mixed static & dynamic techniques	Mixed static & dynamic techniques
Capacity	Capacity	Capacity	Capacity

Tallinn University of Technology (TUT)			
Feature	Requirement	Result	Reference
Performance	Assertion checking speed up	100% speed up achieved	Decider before VERTIGO
Capacity	Reduce memory consumption	HLDD size halved	Decider before VERTIGO
HLDD-based ATPG	Increase coverage	Coverage increased by 50% on average	Decider before VERTIGO
Code coverage	Increase code coverage at the same cost of simulation time	10% increase achieved	Decider before VERTIGO

4. Validation of VERTIGO tools & flow

A preliminary validation of AerieLogic's imPROVE-HPK for *automatic property extraction* was conducted by STMicroelectronics.

The Alternate Functions check has been implemented by ST for verifying the configurable logic that is used to customize the chip periphery of the Automotive platform. The Alternate Functions check automatically extracts properties to be verified from an IP-XACT format description, that are then proven by a commercial formal verification tool.

Modeling of the "CRC" block by means of Petri nets and application of PRES+ for Model Checking

Modeling of the "CRC" block by means of HLDDs in order to simulate and demonstrate an "unconventional" coverage measure alternative to the classical based on RTL modeling.

Application of Coverability feature plus Assertion Step Coverage and Assertion Variable Coverage to CRC, ECC and SPI blocks.

Then a final validation involved all partners over a simple model of the platform that was shown during the demo session, referred to as the *demo platform*, in the last VERTIGO review.

In the following sections more details are reported.

4.1 imPROVE-HPK evaluation

imPROVE-HPK has been tried on a real piece of design that ST could not export to the partners for confidentiality reason. This design is the so called "AHB Interconnection Matrix" which is taken as a complex communication Infrastructure in the Office application platform.

Purposes of the validation were to judge the effectiveness of imPROVE-HPK in preparing the verification environment:

- Specify the desired AHB connections for each AHB port (e.g. master or slave) in the design to the verification environment
- Specify the configuration
- Friendliness of the imPROVE-HPK GUI and the underlying property library

imPROVE-HPK can be interfaced with the AerieLogic native imPROVE-HDL formal engine or also with commercial tools that support PSL or SVA as standard for property specification. This is part of the evaluation targets.

4.1.1 Case studies

Repeater block

The Repeater block is a simple bridge, inside the Office application platform, with 1 Master + 1 Slave interfaces. This was chosen as a simple bench to practice with the environment, before addressing the complex AHB subsystem.

AHB Interconnection Matrix subsystem

The main characteristics of the AHB interconnection are:

- 10 masters
- 27 AHB slaves
- 18 APB slaves

Design Circuit Elements:

- | | |
|------------------------------|--------|
| - Inputs (arrays expanded): | 17152 |
| - Outputs (arrays expanded): | 4052 |
| - Sequentials: | 9417 |
| - Combinationals: | 497145 |

The structure of this subsystem is so-called “multilayer”, as there is the capability to support transfers from multiple masters in parallel – in a simple subsystem, with #M masters and #S slaves just 1 transaction may happen at each time. The subsystem is also a matrix, because selected paths may be activated in parallel. This complex structure results in a number of decoders and arbiters that are embedded in the infrastructure itself, instead of the single decoder and the single arbiter like in the simple subsystem.

Some AHB slaves act as bridges to APB buses, thus making the infrastructure more complex – although this is a case featured also in simple schemas.

4.1.2 Evaluation items

The evaluation targets involved two main areas:

- Protocol properties
- Exporting imPROVE-HPK to commercial tools – through PSL

4.1.3 Evaluation Execution

Repeater block

The setup for this design has been very fast through the improve-HPK GUI. A first run reported a number of false negatives, i.e. errors highlighted by the tool that at a later inspection revealed inexistent, that required to add user defined constraints.

One aspect that became apparent is that there are a number of different flavors concerning AHB interfaces. The most common interfaces are:

- “IP master” → on top of common signals: HBUSREQ (output), HGRANT (input), HLOCK (output), HSPLIT (input) – the last two signals are seldom used in our experience
- “IP slave” → on top of common signals: HSEL (input), HREADYIN (input)

In the case of blocks acting as interconnects, e.g. the bridges, the interfaces are, in general, mixed or the blocks may present two symmetrical interfaces – like the case of the Repeater that has two slave-like interfaces, as far as the signals are concerned. In those cases what really makes the difference is not only the signal list but the controls and the checks that are applied at those signals.

The situation faced with the Repeater block is depicted in the figure6:

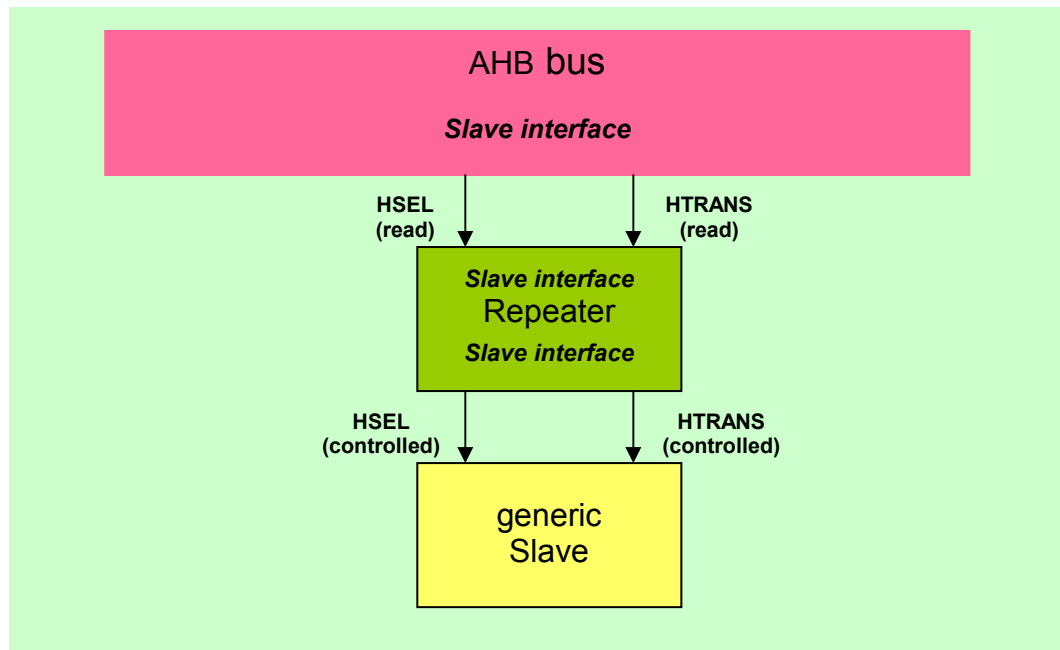


Figure 6

The interface of the *AHB bus* to the *Repeater* is a slave one. As the *Repeater* has just to buffer the signals to a *generic Slave*, also the *Repeater* to *generic Slave* interface is a slave one; however in the *Repeater*'s interface from the *AHB bus* the *HSEL* and *HTRANS* signals are read by the *Repeater* (and must be controlled by the verification environment) whereas in the *Repeater*'s interface to the *generic Slave* the *HSEL* and *HTRANS* signals must be controlled by the *Repeater* (and read or checked by the verification environment).

With this setup, the *HTRANS* signal must be checked by the verification environment – in the *Repeater* to *generic Slave* interface - only when *HSEL* is high (active), because the *HTRANS* is not significant when the peripheral is not selected and the lack of this constraint-in-the-check leads to a false negative in the protocol checking of the *Repeater*. With the current version of *imPROVE-HPK* library it has been necessary to add a patch in the verification environment.

In addition, the verification environment requires that the signals that are checked stay stable while *HREADY* signal is low (inactive). Nonetheless, this is not a requirement of the *AMBA* specification for slave ports; the result is that, during the verification of the *Repeater* to *generic Slave* interface, a false negative is exposed – *Control_When_Wait_State* property.

Another issue that we have faced is due to an over-constraint in the tried version of *imPROVE-HPK* library. According to the *AMBA AHB* specification a master can abort a transaction, upon "ERROR" condition on *HRESP* signal, at any time. In *improve-HPK* it is possible to choose either *always abort* (i.e. at the first cycle the *HRESP* signal feature and "ERROR" condition) or *never abort* transaction. This caused two problems

- a functional bug of the repeater is revealed only when the master aborts at the second cycle of "ERROR", and therefore cannot be found by *imPROVE-HPK* with either abort policy. This is the case when the verification environment drives the *Repeater* block

- a false negative is exposed because the Repeater block always aborts read transactions but never write transactions, therefore it is not possible to mix read and write transactions in a single verification. This is the case when the verification environment checks the Repeater block

AHB Interconnection Matrix subsystem

The full interconnect is much more complex than the simple repeater. In this case the imPROVE-HPK library needs to be heavily adapted. In one day it was possible to configure one master interface and one slave interface. It was not possible to complete this task during the support period from AerieLogic.

Exporting imPROVE-HPK via PSL

Properties were exported in PSL format to be checked by commercial tools. The export and setup of commercial tools with those properties is simple.

- 45 properties were exported
- 11 properties were proven by the chosen commercial tool
- 2 were proven by imPROVE-HDL

It is not possible to make an exact performance comparison; the two tools were left to run a couple of days each.

4.1.4 Results

Office application platform needs to verify interconnecting blocks, therefore it is unlikely that IP-master or IP-slaves will be considered.

Main Pros and Cons are reported, and may certainly be the occasion for further discussion between AerieLogic and ST.

Pros and Cons

imPROVE-HPK – PROs

- very easy setup of standard interfaces
- possibility to export properties to other tools
- performance evaluation properties (optional)

imPROVE-HPK – CONS

- flexibility on some assertion definitions
 - (e.g. an AHB master is supposed to always/never cancel a burst when an ERROR is received, this may hide some bugs)

imPROVE-HDL – PROs

- possibility to replay the trace with an external simulator
- coverage metrics provided, based on cone of influence of single properties

imPROVE-HDL – CONS

- one failure on Repeater was not identified (possibly due to small bounded proof limit)
- only 2 properties proven (11 for commercial)
- poor debug capabilities
- no GUI in imPROVE-HDL

4.1.5 Conclusions

The imPROVE-HPK package provides a good automation support to verification in case IP-master or IP-slave interfaces, that is when a “standard” environment drives the IP’s interface. In such a context the GUI has been proven a fast and effective way to set up the verification environment.

In the case of interconnecting blocks, e.g. complex glue logic functions, it is often necessary to deal with hybrid interfaces, i.e. featuring both master and slave functions, which may require to add specific user constraints, in order to adapt the imPROVE-HPK package to the actual design. This needs a very good knowledge of the AHB protocol.

4.2 Modelling and verification of the CRC block by means of Petri nets

The CRC block implements the computation of the Cyclic Redundancy Check out of a data stream and is provided with a standard register interface.

CRC has been modelled by means of Petri nets and then verified. The description includes also timing information, i.e. the definition of time limits by which transitions can occur.

PRES+ engine provided a graphical representation of the model that shows how the different transitions, or *events*, can be fired according to the availability of the necessary tokens in the feeding *conditions*.

The interesting part, however, is the execution of model checking. In the model checking section of PRES+ it is possible to specify some simple CTL (Computational Tree Logic) formulae. Three relevant analyses have been made for the CRC model:

- Liveness properties:
 - (AF result)
 - AG (input -> AF result)

meaning that eventually a result will be produced.

These properties were proven false in a first version of the model as an unlimited transition time was set in the starting event. Once that time has been bounded, the two properties pass

- One performance property. The property:
(EF <= 100 result)

States that it is possible to obtain a result within 100 time units – during a time unit several concurrent events occur. PRES+ shows this is not true, whereas by substituting 100 with 200 the property turns out to be true. In the end, by dichotomy it is possible to find that the shortest time is 165.

4.3 Modelling and verification of CRC block with HLDDs

The CRC block has been described with HLDDs via HIF translation. With the HLDD model it is possible to perform simulation and perform coverage analysis. The used simulation patterns are the same as the ones used for RTL verification, automatically produced by Laerte++ - see next section 4.4.

The coverage provided by HLDD simulation includes two of the traditional coverage measures performed at RTL, that is code coverage, expression coverage and toggle analysis – flipping bits.

HLDDs can be refined in *Minimized* or *Reduced* formats – indicating two different levels of compaction. It can be shown that the coverage obtained by means of Minimized format is very similar to the conventional coverage, whereas the *Reduced* format allows to obtain a partial path coverage information. In the figure 7 it is shown an example of uncovered items – obtained when just a subset of the full verification patterns have been applied to the CRC.

It can also be shown that the coverage computed on HLDDs can be back-annotated on the original VHDL description.

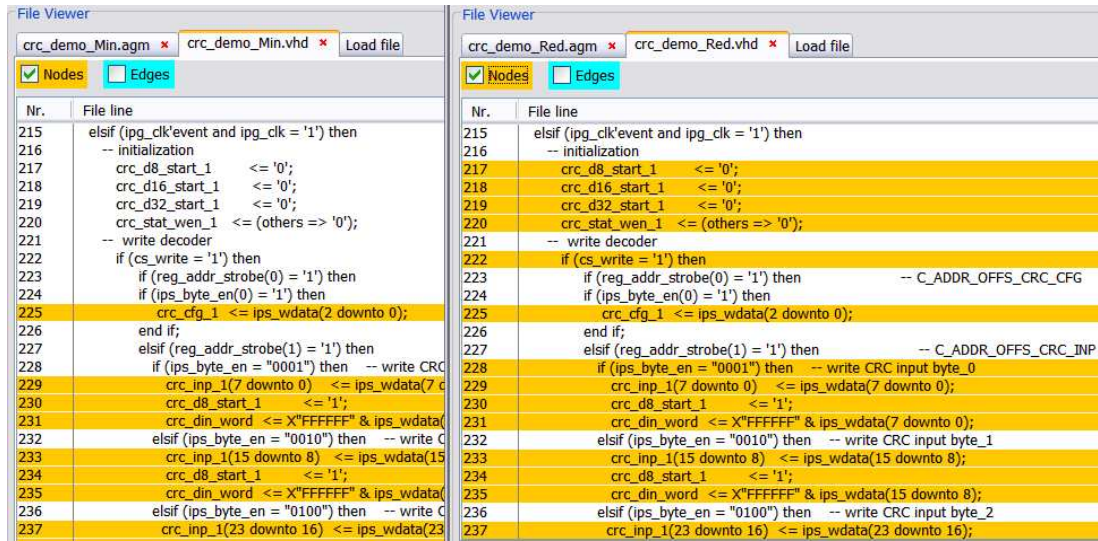


Figure 7: comparison between Minimized HLDD and Reduced HLDD coverages

In the figure, more highlighted (uncovered) VHDL file lines for the reduced HLDD based experiments (right-hand picture) than for the minimized HLDD based ones (left-hand picture) are shown. For example, some of the lines that have been highlighted extra during the reduced HLDD based measurements are: 217-220, 222, 228. The uncovered items at lines 217-220 refer to the fact that the initial values, assigned by means of non-blocking assignments, are never used in the following computation of CRC.

4.4 Coverability, Assertion Step Coverage, Assertion variable Coverage

One problem the verification engineer has to face with is to understand how to stop the production of stimuli during the process of the functional verification. At a certain point, it becomes vital to understand if a certain line of code or condition can be covered or not. What is important is to possibly reach 100% of explained coverage, i.e. by discarding the lines/conditions that cannot be reached. Reach-ability is typical problem that can be investigated with formal technology, i.e. the Integration of imPROVE-HDL inside Assertain environment.

An example of this reach-ability analysis for the ECC block is reported in figure 8.

```

228 p_ips_read2_1 : process (cs_read, reg_addr_strobe,
                           crc_cfg_1,crc_inp_1,crc_cstat_1,crc_outp_1,gnd)
229 begin
230   if (cs_read = '1') then
231     if (reg_addr_strobe(0) = '1') then          -- C_ADDR_OFFS_CRC_CFG
232       ips_rdata <= gnd(31 downto 3) & crc_cfg_1;
233     elsif (reg_addr_strobe(1) = '1' ) then      -- C_ADDR_OFFS_CRC_INP
234       ips_rdata <= crc_inp_1;
235     elsif (reg_addr_strobe(2) = '1' ) then      -- C_ADDR_OFFS_CRC_CSTAT
236       ips_rdata <= crc_cstat_1;
237     elsif (reg_addr_strobe(3) = '1') then       -- C_ADDR_OFFS_CRC_OUTP
238       ips_rdata <= crc_outp_1;
239   else
240     ips_rdata <= (others => '0');
241   end if;
242 else
243   ips_rdata <= (others => '0');
244 end if;
245 end process p_ips_read2_1;
    
```

Figure 8: Reach-ability analysis on ECC block

Assertion Step Coverage and Assertion variable Coverage have been tried on an assertion describing the computation flow of CRC through the data stream written to the registers. It can be shown that a hole in ASC hid an actual bug in the VHDL description.

4.5 Validation on the demo platform

The *demo platform* is reported in the figure 9.

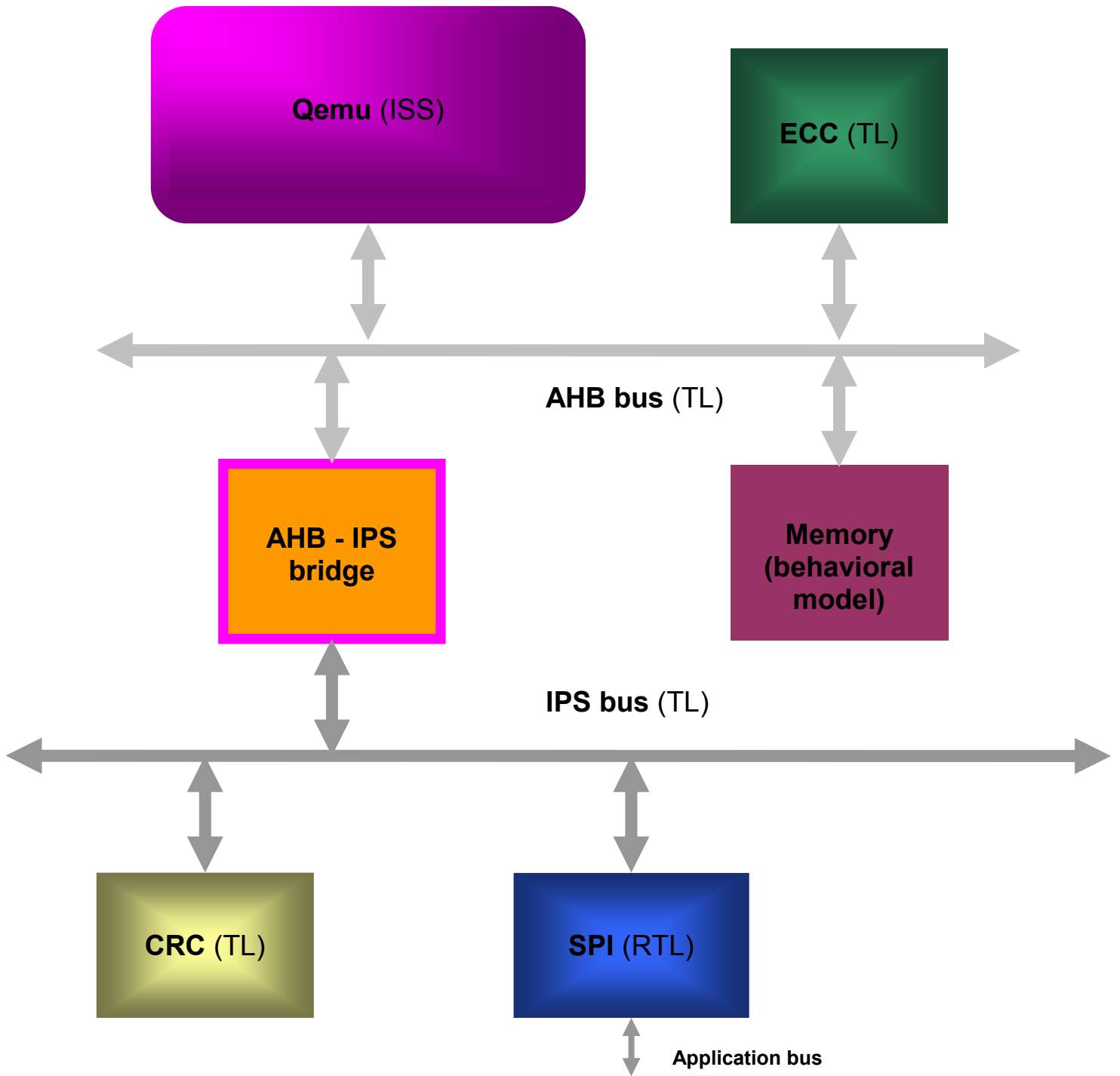


Figure 9: the demo platform

Blocks appear at different abstraction levels and are described in SystemC. Qemu models the processor plus the host operating system. The other blocks appear both at Transactional or Register Transfer levels.

Various applications of VERTIGO developed tools were performed.

- Translation of ECC, CRC, SPI blocks into HIF format at RT level (RTL)
- After the translation, Automatic Test Pattern Generation for the verification of the obtained RTL and coverage evaluation by means of Fault Injection metric by using Laerte++
 - Fault injection and fault-list visualization by means of ACIF tool
- Automatic transactor generation from the IPS (TL) bus to the SPI (RTL) block by means of the TGen tool
- Abstraction from RTL to TL for ECC and CRC blocks by means of A2T tool
- Co-simulation of SystemC modules with Qemu ISS model
 - Assisted generation of device drivers for ECC, CRC and SPI modeled by means of DGEN tool

A picture reporting the SW architecture of the platform simulation is reported in figure 10.

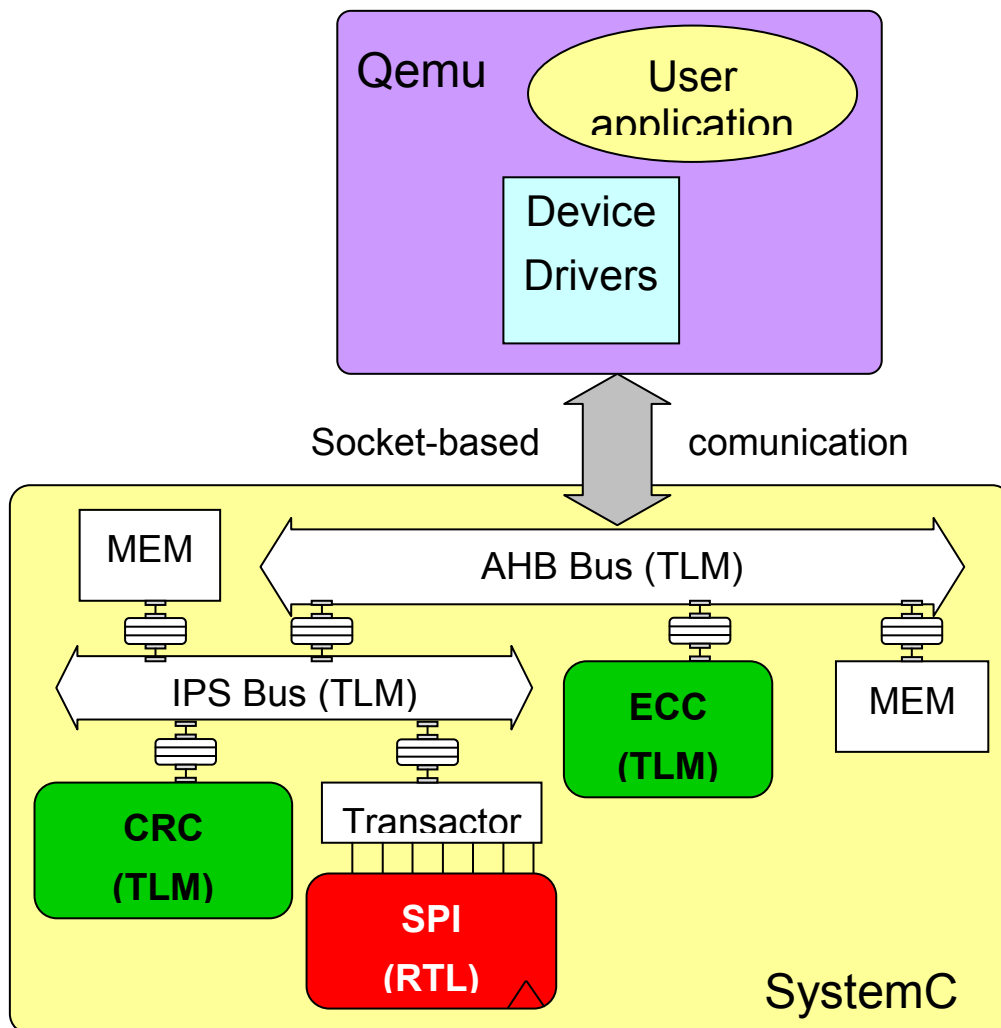


Figure 10: SW architecture of the simulation platform

- Verification of the whole platform by using a mixed RTL/TL fault model. Use of ATPktG tool to inject & visualize faults at TL

- ATPG in the SW/HW co-verification by means of the HSN integration of Laerte++ with Qemu. With this integration Laerte++ takes advantage of control signal and data signal information to generate efficient test patterns

The figure 11 shows how Laerte++ has been integrated with Qemu for the purpose of the ATPG.

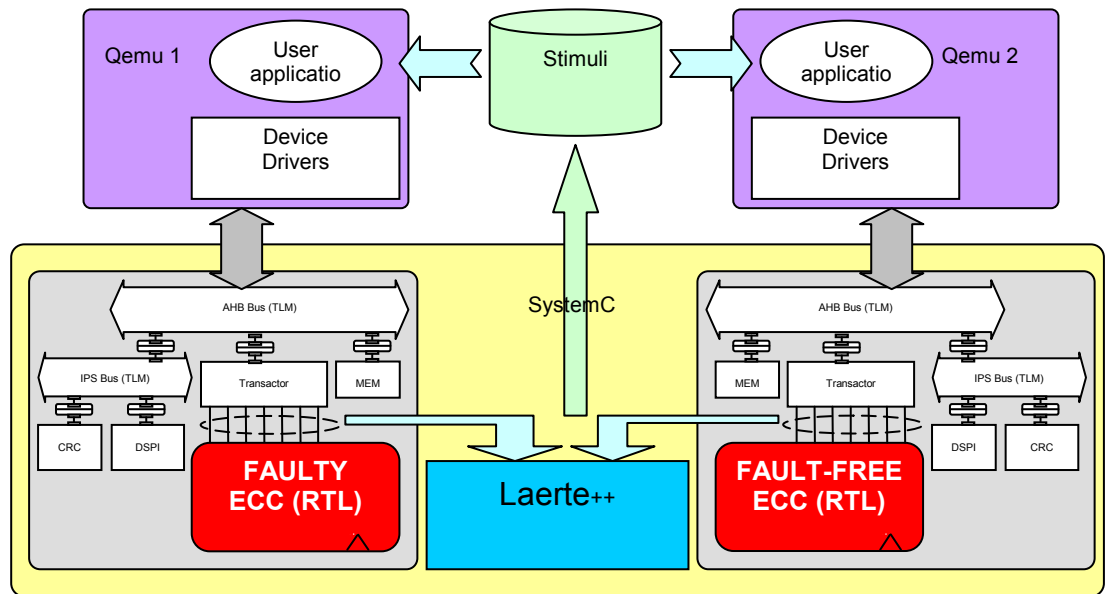


Figure 11: Laerte++ integration with HSN environment for ATPG on ECC block

5. Main deviations from the DOW

The comparison of the stated objectives and the obtained results, shows that some targets have been only partially addressed. Different causes contributed:

- STMicroelectronics reserved two platforms (Office Application and Automotive application) to be investigated during VERTIGO but the effort to analyze a single platform was enough to saturate the whole project
- As a matter of fact, in the middle of VERTIGO the Office Application platform became obsolete and the move to the Automotive platform became mandatory. This fact involved a change from a Top Down design & verification approach to a Bottom Up verification approach.
- Although most of techniques developed during VERTIGO can be employed both in Top Down and Bottom Up approaches, methodologies concerning modeling & verification at high level suffered from this shift – see KT1 and KT2 targets. The consequence is that those methodologies have been applied at block level instead of System level.

A good example of adaptation in this variable context is represented by the A2T tool development. Instead of performing an equivalence check between a RTL description vs TL model, it was decided to use the underlying technology to implement an automatic abstractor that produces a correct-by-construction TL model.

A significant effort has been performed to integrate the partners implementation techniques into a single environment. For the purpose TransEDA's Assertain product was the natural candidate. In the end a tight integration is present for commercial tools or for collaborations already in place before VERTIGO start – see in KT4 tight integration and networked integration. As an example, the integration chain of Southampton University's SAT-based tools into AerieLogic's imPROVE-HDL and consequently into TransEDA's Assertain has been strengthened. On the other hand, bilateral integrations have been achieved, like: Decider and Laerte++ or Laerte++ and HSN.

6. User evaluation and comparison vs the State of the Art

During VERTIGO several modeling and verification techniques have been investigated, as stated in the project objectives. However three main points remain partially unaddressed:

- The high level model of system behavior through a system view. reason:
 - Shift from Top-Down flow to Bottom-Up flow for the STMicroelectronics flow and limited time left to investigate Petri net modeling issues
- Identifying a clear semantic for moving (a set of) TL properties to (a set of) RTL properties and open the way to adopt a property restricted equivalence checking criterion. This is a very ambitious target and was partially addressed in the Bottom-Up flow by means of the A2T tool
- Although several ABV techniques have been developed, a tight integration of the related tools has not been achieved in Assertain.

In the following table some specific features that were asked by STMicroelectronics are commented.

User's viewpoint			
VERTIGO Feature	Plus & Minus / Satisfaction	Market Offer	Still to go ?
Coverability	<i>Good</i>	Partial support found in commercial tools	
Fault Injection ATPG	Powerful extension to TL. <i>Good</i> , in lab environment	Certitude product by CERTESS (RTL only)	Push EDA vendors to adopt this technology and provide user-friendly integration
Automatic Property Generation	<i>Fair</i> for standard properties & interfaces	Major vendors today provide libraries integrated with their products. A key success factor could be a comparison of the cost	Address deviations from the standard protocols.
Formal Proof	<i>Fair</i> proof and size capacity for protocol related properties. Less proof capacity compared to BDD based approach on medium size problems.	All major EDA vendors provide today competitive solutions based on sophisticated integrations of different techniques	The amount of necessary investment in this area by the EDA vendor position is cutting off SME's

<p>Assertion Coverage Assertion Signal Coverage Assertion Step Coverage</p>	<p><i>Fair</i></p>	<p>No support found in other commercial tools</p>	
<p>Automatic Abstraction - A²T</p>	<p><i>Does not support hierarchy</i></p>	<p>Abstraction products have been on the market for some years, but the approach is based on the knowledge of architectural details – e.g. datapaths</p>	<p>This is yet a research topic</p>
<p>TL mixed simulation</p>	<p><i>Excellent</i> with the integrating of HSN environment. A good quality of the starting master testbench is necessary to achieve a significant result</p>	<p>No support found in commercial tools.</p>	<p>Needs to be extended beyond master-to-slave interfaces</p>
<p>Coverage Features and Unified Coverage Report</p>	<p>Unified report including Dynamic Simulation and Formal Proof available through Flex allowing fast customization and charting. This is <i>Sufficient</i> but not competitive for a commercial deployment</p>	<p>Major EDA vendors are providing a merger of coverage metrics from Dynamic Simulation and Formal Verification. This is far beyond the original VERTIGO targets and possibly requires a huge investment</p>	<p>Original VERTIGO targets are obsolete vs the current commercial offer. HLDD contribution exploitation should be further investigated to see if there is a potential to enrich coverage metrics</p>

In the next chapter 7 it is reported how the VERTIGO partners intend to proceed to address part of the open points at the end of the project.

7. VERTIGO – looking forward

The culmination of the VERTIGO project for the industrial partners TransEDA and AerieLogic is the integration within the same toolset 'Assertain' of the dynamic code coverage of TransEDA and the static formal verification techniques of AerieLogic and University of Southampton.

Additionally TransEDA will extend the functionality of Assertain through a technical co-operation with Tallinn University of Technology. The HLDD algorithms developed by Tallinn University of Technology will be incorporated into Assertain in 2009.

Assertain will replace the TransEDA's VNCover toolset which is used worldwide by over 60 companies. TransEDA expects to roll out the results of the successes obtained with VERTIGO in 2009.

A major focus for AerieLogic during VERTIGO was to strengthen its formal property checker imPROVE-HDL. This was achieved with the following tasks:

- a) integration of a new BMC developed in collaboration with University of Southampton. imPROVE-HDL can now address design blocks of several hundreds of thousands gates.
- b) integration of a prototype UMC developed in collaboration with University of Southampton. The results are promising, with average 60% properties proved instead of 30% with inductive proof alone.
- c) PSL and SVA assertions language support. Both languages are working in an industrial context on VHDL, Verilog and SystemVerilog designs.

This enabled AerieLogic to provide a high capacity formal tool for integration in the TransEDA platform Assertain. AerieLogic also uses imPROVE-HDL for formal verification services.

imPROVE-HDL will be extended towards TLM levels within the COCONUT (FP7-2007-IST-1-217069) european project started in 2008, including AerieLogic, University of Southampton and University of Verona among the partners.

Another task initiated by AerieLogic under VERTIGO is an automated flow for protocol compliance using formal verification. This is promising for property checking usages that AerieLogic is going to enhance. Part of these developments are conducted under COCONUT.

Technologies developed by University of Verona during VERTIGO were mainly related to:

- a) mutation analysis and testing, which has been included in the Laerte++ prototype;
- b) property coverage, which has been included in the PCC prototype;
- c) TLM/RTL modelling and integration, which has been included in HIFSuite, A²T and TGEN.

Technologies developed inside Laerte++ are under analysis by an EDA startup. It is particularly interested in the TLM fault model and in its possible application.

Commercial rights of HIFSuite have been recently acquired by EDALab s.r.l., a spin-off of the University of Verona whose mission consists of exploiting academic research and developing solutions in the field of modeling and verification of networked embedded systems. In this context, EDALab is currently engineering HIFSuite features. It is expected that HIFSuite will be marketed by EDALab by the end of 2009.

Finally, PCC, Laerte++, HIFSuite A²T and TGEN have been included as starting technologies in COCONUT. In particular, A²T and TGEN are expected to be commercialized by EDALab at the end of the COCONUT.

8. Dissemination of knowledge

The VERTIGO partners participated in several events during the whole project. Their publications and when applicable their presentations are available on the VERTIGO public web site, at the URL: <http://www.vertigo-project.eu>.

Publication statistics

Total number of publications created in the frame of the VERTIGO project is 62. This includes 15 journal papers, 36 conference publications and 11 workshop presentations. 4 scientific publications have been jointly submitted by two VERTIGO partners.

Publications by project's year

	Journal	Conference	Workshop
1st year	LIU → 1 SOTON → 1 TUT → 0 UNIVR → 0 (tot: 10)	LIU → 0 SOTON → 4 TUT → 1 UNIVR → 1 <u>total</u> → <u>6</u>	LIU → 0 SOTON → 0 TUT → 0 UNIVR → 2 <u>total</u> → <u>2</u>
2nd year	LIU → 1 SOTON → 3 TUT → 1 UNIVR → 4 (tot: 30)	LIU → 2 SOTON → 7 TUT → 3 UNIVR → 4 <u>total</u> → <u>16</u>	LIU → 0 SOTON → 0 TUT → 3 UNIVR → 2 <u>total</u> → <u>5</u>
Final ½ year	LIU → 1 SOTON → 2 TUT → 1 UNIVR → 1 (tot: 20)	LIU → 0 SOTON → 2 TUT → 5 UNIVR → 5 <u>total</u> → <u>12</u>	LIU → 0 SOTON → 0 TUT → 1 UNIVR → 2 <u>total</u> → <u>3</u>
Total: 60	16	34	10

Publications by academic partner

	Journal	Conference	Workshop
LIU (tot: 5)	3	2	-
SOTON (tot: 19)	6	13	-
TUT (tot: 15)	2	9	4
UNIVR (tot: 21)	5	10	6

9. Glossary

AHB: *the Advanced High-performance Bus, a part of the AMBA hierarchy of buses specification.*

AMBA: *the Advanced Microcontroller Bus Architecture, ©ARM Limited 1999.*

APB: *the Advanced Peripheral Bus, a part of the AMBA hierarchy of buses specification.*

Arbiter: *the Arbiter ensures that only one master at a time is allowed to initiate data transfer on the bus.*

Assertion: *1. A property that the DUV has to fulfil. 2. In some contexts (e.g. in dynamic verification), synonym of Property.*

Assertion Checker: *A procedural component aimed at checking whether a given assertion is violated by the DUV during testbench simulation.*

Assertion Based Verification: *A verification task using Assertion as fundamental elements to check DUV correctness. Can refer either to Model Checking or to Simulation instrumented by assertion checkers*

Assumption: *A property that the DUV verification takes for granted; assumptions are properties used to model input behaviours.*

ATPG: *Automated Test Pattern Generation, a definition coming from Testing activity. In the functional verification domain it represents a process of helping the verification engineer to produce a set of legal stimuli to the DUV*

BDD: *Binary Decision Diagram. Graph representation of discrete functions.*

Behaviour: *A sequence of input-state-output tuples describing design functioning through time.*

Block: *A component in a design. A block may contain instances of other blocks.*

BMC: *Bounded Model Checking. Proof techniques to verify a property only on a bounded number of cycles. The user decides if the bound is large enough for the proof to be considered complete.*

Decider: *Decider is a data variable-oriented deterministic ATPG based on HLDDs.*

Design: *A model of a piece of hardware, described in some Hardware Description Language (HDL).*

DUV: *Design Under Verification. The subject of a verification process (it has to be a model for the set of properties used as assertions).*

Formal Verification: *A technique aimed at verifying that the DUV is correct with respects to all the elements of set of possible input value sequences usually specified implicitly by means of assumptions, or considered to be totally free in absence of assumptions. Compared with more traditional verification techniques, like logic simulation in presence of a testbench, Formal Verification is usually characterized by an extremely higher coverage, but often affected by an applicability limitation due to smaller size of treatable problems w.r.t. testbench simulation.*

HDL: *Hardware Description Language. It allows to describe the behaviour of a piece of hardware both in procedural and concurrent fashion. (Netlist represents a concurrent fashion).*

HIF: *Hardware Intermediate Format. It is a database format for TLM and RTL descriptions, resulting as an extension from Symbad's AIF*

HLDD: *High-Level Decision Diagrams. Graph representation of discrete functions that are a generalisation of Boolean Decision Diagrams. Nodes represent variables in the design, edges correspond to value ranges, which represent activation conditions. Supported data types are not limited to Boolean domain but include integers, bit vectors, enumeration types etc.*

Inductive proof: *proof techniques based on recurrence. Suppose the property is true on any $n-1$ cycles, is it then always true on the n^{th} cycle? When such a proof is found, it is complete.*

IP: *Intellectual Property. In the context of VERTIGO IP is usually synonym of RTL Block*

Master: *A Master is able to initiate read and write operations by providing an address and control information. Only one master is allowed to actively use the bus at any one time.*

Model Checking: *A formal verification technique aimed at checking that a DUV satisfies a set of temporal properties.*

Petri net: *A graphical modelling language that conveniently and intuitively captures both flow and concurrency.*

PRES+: *(Petri net based representation of embedded systems) A representation, based on Petri nets, to capture real-time embedded systems. Tokens are augmented with values and transitions with time delay intervals, guards and functions. This allows to capture both data and control flows as well as real-time aspects.*

Property: *A collection of logical and temporal relationships between expressions involving design signals, that represents a set of behaviours.*

Property Checking: *1) Model Checking synonym. 2) (less frequent) ABV synonym.*

Protocol: *A set of rules governing communication between participants in a system.*

PSL: *(Property Specification Language) a language aimed at easing the expression of temporal properties, built on top of a propositional temporal logic (LTL), meant to be used for both specification of designs properties and for environment description, either by means of assumptions, or through the writing of additional procedural code. PSL v1.1 has been ratified as a standard by IEEE-1850 committee.*

RTL: *Register Transfer Level. It is a style of HDL description that includes high-level constructs and clearly separates Combinational and Sequential resources. Often RTL is synonymous of synthesisable description, i.e. the one that can be automatically translated into physical gates.*

Slave: *A Slave responds to a read or write operation within a given address range. The slave signals back to the active master the success, failure or waiting of the data transfer.*

SAT: *(Boolean) Satisfiability. Given a Boolean expression E , decide if there is some assignment to the variables in E such that E is true. A Boolean expression is composed of Boolean variables, (logical) negation (NOT), (logical) conjunction (AND) and parentheses for grouping*

SystemC: *a C++ class library used to model electronics systems.*

SVA: *(SystemVerilog Assertions) the assertive part of SystemVerilog.*

SystemVerilog: *A HDL modelling language directly derived from Verilog, augmented with data abstraction, configuration features derived from VHDL, and a rich temporal assertion language. SystemVerilog is currently under ratification as a standard by IEEE-1800 committee.*

Testbench: *The combination of DUV, stimulus generation, expected output estimation, and output comparison. Usually meant to be used in the context of an HDL simulator.*

Transaction: *a composition of events that represents a basic communication token, whatever the abstraction level is involved. Read and Write operations are usually considered as transactions.*

TLM: *Transaction Level Modeling*

UMC: *Unbounded Model Checking. Proof techniques to verify a property whatever the number of cycles is. When such a proof is found, it is complete.*

Validation: *The process of verifying the function and performance of a design interacting with the surrounding environment, e.g. the SW running on it. In some circumstances Validation means the process of verifying the actual silicon chip.*

Verification: *The process of falsifying or verifying the functional and performance requirements of a design, be it chip, board or system. Many different kinds of verification tools are in use today, including simulation, formal verification, emulation and rapid prototyping.*

Verilog: *One of two standardized hardware description languages (IEEE-1364), used to specify the structure and behaviour of electronic systems in textual format.*

VHDL: *(VHSIC Hardware Description Language, where VHSIC stands for Very High Speed Integrated Circuit) One of two standardized hardware description languages (IEEE-1076), used to specify the structure and behaviour of electronic systems in textual format*

