European formal verification tools for model correctness

J. Mermet, A. Morawiec

European Formal Verification Tools for Model Correctness

Jean Mermet and Adam Morawiec
Université Joseph Fourier
Laboratoire TIMA
Modélisation et Vérification des Systèmes Digitaux
BP 53, 38041 Grenoble Cedex, France
Phone: +33 4 76 63 56 71
Fax: +33 4 76 44 04 54
E-mail: Jean.Mermet@imag.fr, Adam.Morawiec@imag.fr

ABSTRACT

This article gives a survey on formal hardware verification tools developed in Europe. It describes the main objectives and domains of application of the formal methods for the verification of electronic system-on-a-chip models and designs. Further, it attempts to introduce some classification scheme for the existing commercial or prototype tools, which is based on the different verification aspects: proof technique, type of the circuit, abstraction level, verification objective, mathematical model, input format, diagnostic method incorporated, verification execution and shows the particular results for each of these aspects. Finally, some conclusions concerning the possible future trends in development are drawn.

1. SYSTEM DESIGN FLOW

The design of electronic circuits leveraged by the deep submicron technology development allowing very high chip densities and driven by the need for more and more complex applications is moving towards the system design.

Electronic system design aiming at the design of the single-chip systems can be accelerated by the integration of cores which encapsulate the intellectual property (IP) and by their interfacing with the rest of the system to be built. The cores are the black-box models of intellectual property building blocks supplied by the IP providers/developers and the third-party library providers. The system design process requires system-building tools, bringing all the necessary elements and encompassing the application expertise of library providers for the development of the systems-on-a-chip. This must include the integration and interfacing of the IP cores. These procedures of integration and interfacing needs to be formalized by the standardization work, which may happen in the frame of the VSI Alliance initiative, where the focus concentrate along the six axes: on-chip buses, IP protection, test, system level design and verification, mixed signal and implementation-level verification.

In the development of building block models, the verification of the model correctness plays a crucial role in guarantying the modeler that the results obtained are in compliance with the modeler's intent and/or the design specification, as well as in making the IP block user confident that the block behavior is fully compliant to the specification. The model development is a multi-skill process. Not only design ability and HDL mastering are necessary but model correctness assessment is now mandatory.

2. THE NEED FOR THE FORMAL MODEL/DESIGN VERIFICATION

Traditionally, the verification of the design/model correctness is done by the simulation, which despite of the progress in simulation performance due to the introduction of the methods like e.g. cycle-based simulation, cannot cope with the complexity of the deep submicron designs. The test vectors employed in simulation can cover only a reduced number of all circuit states, thus, there is a significant part of the design which remains unverified. The formal methods applied to hardware

verification offer the significant gain in the verification time, and, what is more important, can give a total certitude that the specified property is accomplished by the system, when the verification succeeds.

In the design process there is a need for different types of verification according to the design step or level of abstraction: the verification of the design idea / intent of the modeler (the specification validation); the verification of some properties of the model; the comparison of the more detailed model (implementation) done at the lower level of abstraction with the system model provided at the higher abstraction level; the comparison of two equivalent design descriptions given at the same level of abstraction.

3. Formal Verification Tools

Several university and industry research centers have tried to develop the tools founded on formal methods, which could help the designer to examine the functional and/or dynamic (temporal) correctness of the design/model. This section contains some results of the survey on formal verification tools developed in Europe. It will introduce the classification scheme for the tools and some of their properties which are useful for the application of this tools in modeler's practice.

3.1. European tools available

The following list contains the names of the formal verification tools developed in Europe which has been taken into account in the survey [1]:

BSNVERI, FVG BDD Tool (Eindhoven University of Technology, Netherlands);); C@S (University of Karlsruhe, Germany); CheckOff-E, CheckOff-M, LAMBDA (Abstract Hardware Limited, UK); ELLA Verification Environment (University of Manchester; UK); FANCY (Institut National Polytechnique de Lorraine, France); HOL (University of Cambridge, UK); ICOS (OFFIS, Oldenburg, Germany); LOVERT (Darmstadt University, Germany); MONA (University of Aarhus, Denmark); SMOCK, VHDL2NQTHM, PREVAIL[2] (Université Joseph Fourier, Grenoble, France); Synchronized Transitions (ST) (Technical University of Denmark); SVP Specification and Verification of Processors (Université de Nice, France); TACHE (Université de Provence, France); Verification Condition Generator for VHDL (VCG) (Universidad Politecnica de Madrid, Spain); VERENA (University of Passau, Germany); VFORMAL (Compass Design Automation); YATC (Philips, Netherlands).

3.2. Description of the classification scheme

The principal formal verification tools characteristics are divided into four groups: *the technical characteristics*, *the environment characteristics*, *the performances* and *the system requirements*, and finally, *the bibliographical information*.

In the first group we consider the type of the circuit to be verified, the abstraction level accepted by the tool, the objective of the verification, the mathematical model used by the tool and the formal proof method.

The environment characteristics precise for each tool the different usage aspects of the tool like the input format, , the diagnostic method incorporated in the tool, the user interface and the helps available, and finally the execution methods.

The third type of characteristics taken into account in the survey are the performances and the system requirements. The performances of the tools depend on several particular to the tool characteristics like the input format, the abstraction level of the input description, the proof method and mathematical model used, the system environment, etc., thus, they remain relative for each verification system, so they will not be shown in this article.

Finally, the bibliographical information related to each tool is presented in the references to this article.

3.3. Survey results

3.3.1. Formal proof method

There are five main approaches to formal verification which are applied in the European tools: tautology checking, equivalence checking, model checking, theorem proving and symbolic simulation. All of these approaches assume given a specification of the circuit reflecting its expected function and an implementation reflecting the designed hardware. The formal verification consist in proving that the circuit implementation is correct with respect to the specification.

Tautology checking

The equivalence in terms of function of the specification and the implementation is checked. The implementation is correct when its outputs are equal to the outputs of the specification when to the inputs of both of them are supplied the same input sequences. Both implementation and specification are given at the same level of abstraction.

Equivalence checking

The equivalence in terms of function of the specification and the implementation is checked. Here the specification and implementation may be given at different levels of abstraction.

Model checking

The specification is in the form of a logic formula, the truth of which is determined with respect to a semantic model provided by an implementation. This method allows to verify specific functional or temporal properties of the verified circuit which can be essential to assure its correct behavior.

Theorem proving

The specification and implementation are represented as the formulas in logic. The relationship between a specification and an implementation is specified as a theorem in logic which is proved within the context of a proof calculus (using the libraries of axioms and the deductive apparatus).

Symbolic simulation

The symbolic simulation uses a model of hardware and simulation engine to proof the equivalence between the design specification and implementation. These both circuit representations are simulated concurrently and the results of the simulation are compared to establish the proof. This type of simulation differs from the conventional one because it employs the symbols in place of the actual values of circuit signals or variables.

The table 1 presents the formal proof method used in the European verification tools.

Formal verification tool	Formal proof method				
	TP	TC	EC	MC	SS
BSNVERI			■		
C@S	■			■	
CheckOff-E			■		
CheckOff-M				■	
ELLA Verification Environment				■	■
FANCY			■		
FVG BDD Tool			■		
HOL	■				
ICOS	■	■		■	
LAMBDA	■				
LOVERT		■	■		
MONA				■	
SMOCK				■	
ST	■				

Formal verification tool	Formal proof method (continued)				
	TP	TC	EC	MC	SS
SVP	■		■		
TACHE		■			
VCG					
VERENA	■				
VFORMAL			■	■	
VHDL2NQTHM	■				
YATC		■			

TP theorem proving
TC tautology checking
EC equivalence checking
SS symbolic simulation
MC model checking

Table 1: Formal proof method.

3.3.2. Type of models to be verified

There can be three major groups of the models for which the formal verification is possible: combinatorial, synchronous sequential, asynchronous sequential circuits.

Almost all the examined tools can be applied to the verification of the combinatorial circuits except the symbolic model checker SMOCK. Also most of the tools verify the synchronous sequential circuits. The only exception is BSNVERI, which is a specialized tool for the combinatorial circuits. Some tools can be in addition applied to the verification of the asynchronous sequential design; these are C@S, CheckOff-E, CheckOff-M, HOL, MONA, ST, VCG and VFORMAL.

Formal verification tool	Circuit type		
	combinatorial	sequential synchronous	sequential asynchronous
BSNVERI	■		
C@S	■	■	■[1]
CheckOff-E	■	■	■
CheckOff-M	■	■	■
ELLA Verification Environment	■	■	
FANCY		■	
FVG BDD Tool	■	■[2]	
HOL	■	■	■
ICOS	■	■	
LAMBDA	■	■	
LOVERT	■	■	
MONA	■	■	■
SMOCK		■	
ST	■	■	■
SVP	■	■	
TACHE	■	■	
VCG	■	■	■
VERENA	■	■	
VFORMAL	■	■	■
VHDL2NQTHM	■	■	
YATC		■	

[1] necessary correct modeling
[2] necessary the same state encoding

Table 2: Acceptance of the circuit type by the formal verification tools.

3.3.3. Models abstraction levels

The abstraction level accepted by the tools extends from system level, throughout behavioral and gate level to switch level.

Almost all the tools can be applied for the verification of the designs specified at the behavioral, register-transfer and gate level. The exceptions are BSNVERI, TACHE and VFORMAL which do not accept the behavioral level description, FANCY, FVG BDD Tool and YATC which do not accept neither behavioral nor RTL level, as well as VHDL2NQTHM which is not appropriate for the gate level circuit specification and VCG accepting only the behavioral circuit description.

Only some of the tools allow to verify the design at the system level. Those are C@S, MONA, ST and SVP. From the other side HOL and ST accept switch level design description.

Formal verification tool	Modeling level				
	system	beha-vioral	RTL	gate	switch
BSNVERI			■	■	
C@S	■	■	■	■	
CheckOff-E		■	■	■	
CheckOff-M		■	■	■	
ELLA Verification Environment	?				
FANCY				■	
FVG BDD Tool				■	
HOL		■		■	■
ICOS		■	■	■	
LAMBDA		■		■	
LOVERT		■	■	■	
MONA	■	■		■	
SMOCK		■	■	■	
ST	■	■		■	■
SVP	■	■	■	■	
TACHE			■	■	
VCG		■			
VERENA		■		■	
VFORMAL			■	■	
VHDL2NQTHM		■	■		
YATC				■	

Table 3: Modeling level of abstraction.

3.3.4. Objective of the verification

The objective of the verification may be the verification of the function of the system or the dynamic properties.

All European formal verification tools have been created for the verification of the design function. Some of them (C@S, ELLA Verification Environment, HOL, ICOS, SMOCK and VCG) permit to verify some temporal qualitative properties.

3.3.5. Mathematical model

The mathematical model used by the tool can be based on a type of a formal logic, on an omega-automata or a process algebra [1, 2, 3].

The different kinds of logic are most often used as a mathematical model of the design. The propositional logic is used in the C@S, FANCY and TACHE tools. Various types of the predicate logic have been widely employed: first-order logic in ST and VHDL2NQTHM (Boyer-Moore logic), second-order monadic logic in MONA and higher-order logic in HOL and LAMBDA. Also different types of temporal logic have found a wide use in hardware verification tools. The linear temporal logic has been applied in C@S and ICOS, the variation of branching time temporal logic have been applied to the verification task in C@S (the CTL* and QCTL logic) and in ICOS (the CTL logic). Another mathematical model used is omega-automata, on which ELLA Verification Environment is based, as well as a tool incorporated in C@S environment. ELLA is also using another mathematical model based on process algebra.

Several above mentioned formal verification tools use the different types of mathematical models combined with the data representation based on binary decision diagrams (BDDs) supported by optimization and reduction algorithms to manage the design complexity [6, 7].

Formal verification tool	Mathematical model used			
	logic	BDD	auto-mata	process algebra
BSNVERI		■		
C@S	• propositional • LTL • CTL* • QCTL		■	
CheckOff-E		■		
CheckOff-M		■	■	
ELLA Verification Environment			■	■
FANCY	propositional (Boolean term rewrite system)	■	■	
FVG BDD Tool		■		
HOL	higher-order			
ICOS	• LTL • CTL	■		
LAMBDA	higher-order			
LOVERT		■	■	
MONA	monadic second-order	■		
SMOCK		■		
ST	predicate			
SVP	• first-order • higher-order (for specific cases)			
TACHE	propositional	■	■	
VCG	temporal predicate / Kripke model			
VERENA	higher-order			
VFORMAL		■		
VHDL2NQTHM	Boyer-Moore			
YATC		■		

Table 4: Mathematical model used.

3.3.6. Model description formalism

The formal verification tools use, as an input description formalism, either the standardized languages/formats: the hardware description languages (VHDL and Verilog) and the interchange formats (e.g. EDIF) or proprietary formats or languages. The table 5 presents the details on the

description formats accepted by the tools. Since VHDL is very popular hardware description language in Europe, it is also very often used as the design description language for the formal verification.

Formal verification tool	Input format	
	VHDL	other
BSNVERI		proprietary language
C@S		proprietary language
CheckOff-E	■	EDIF, (Verilog)
CheckOff-M	■	EDIF, Verilog properties: CIL
ELLA Verification Environment		specific language Core ELLA
FANCY		ISCAS'89, BLIF
FVG BDD Tool		intermediate format
HOL		specific language HOL
ICOS	■	timing diagrams state diagrams
LAMBDA		language L2
LOVERT	■	language SMAX
MONA		logic formulas
SMOCK	■	language SMAX
ST		synchronized transitions notation
SVP		FHSL (close to the subsets of VHDL and Verilog)
TACHE	subset	
VCG		
VERENA	■	language VIOLA
VFORMAL	■	
VHDL2NQTHM	■	
YATC		Verilog, NDL, EDIF

Table 5: Input format.

3.3.7. Diagnostic method

The diagnostic facility incorporated in the formal verification tool allows to find the design errors when the verification fails. Most of the tools provide the counter-example which is generated when non-equivalence is discovered and shows the situation in which the proof fails. Some of the tools furnish the sequences of input vectors to show the state where the behavior of compared description is different.

The PREVAIL environment incorporates the specific diagnostic tool allowing for finding and correcting some types of the errors in the input description.

Formal verification tool	Diagnostic method		
	counter-example	test vectors / sequences	other
BSNVERI	■		
C@S	■		
CheckOff-E		■	
CheckOff-M		■	
ELLA Verification Environment		?	

Formal verification tool	Diagnostic method (continued)		
	counter-example	test vectors / sequences	other
FANCY	no		
FVG BDD Tool	■		
HOL			proof trace
ICOS	?		
LAMBDA		■	
LOVERT	■	■	state functions generation
MONA	?		
SMOCK		■	
ST	■		
SVP		■	presentation of the two non equivalent expressions
TACHE		■	
VCG	no		
VERENA	■		
VFORMAL		■	information about unwanted situations
VHDL2NQTHM			proof trace
YATC	■		

Table 6: Diagnostic methods.

3.3.8. Verification execution

The table 7 below shows the manner in which the verification is executed. Most of the formal hardware verification methods allow for a completely automated verification process. However, some of them based on the theorem proving, which presents the most general approach to hardware verification, need to be supported in the verification process.

Formal verification tool	Verification execution	
	automated	user guided
BSNVERI	■	
C@S	■	■
CheckOff-E	■	
CheckOff-M	■	
ELLA Verification Environment	?	?
FANCY	■	
FVG BDD Tool	■	
HOL		■
ICOS		■
LAMBDA		■
LOVERT	■	
MONA	?	?
SMOCK	■	
ST	■	■
SVP	■	■

Formal verification tool	Verification execution (continued)	
	automated	user guided
TACHE	■	
VCG	■	
VERENA		■
VFORMAL	■	
VHDL2NQTHM	■	
YATC	■	

Table 7: Verification execution.

4. CONCLUSIONS

Formal verification methods and tools can play a significant role in the modeling of electronic systems. European universities and research centers have developed a wide range of formal hardware verification tools which can be used to verify the different aspects of the correctness of the model of a design at different levels of abstraction and for different types of the circuit.

The survey presented in this article gives a comprehensive view of the state-of-the-art research in the domain of formal verification of hardware done in Europe. This view is supposed to be useful to model developers who are spending a fast growing amount of time for this job.

It shows that researches examine different approaches to formal verification which are adapted to the particular objectives of the verification. This is because there is no unique method which can be applied for all kind of circuits, for all aspects of verification process and for all levels of abstractions.

We can observe the trend to incorporate into a single environment several tools which can deal with distinct aspects of the verification in order to built a complete verification framework (C@S, ICOS, PREVAIL). The development of the formal verification environments, with a kind of adaptive choice of the appropriate verification method, can be one of the possible future objectives.

The survey will help the modeler to identify the tools applicable for the verification of a specific type of models. In particular it will help choosing the suitable verification tool for a model whose specification is provided at a specific level of abstraction.

We can see already some first approaches to the verification of the temporal properties of a design. Although they concern at present only the qualitative verification of dynamic properties, they establish a significant step towards quantitative temporal properties verification, which is a crucial issue in submicron models.

Most of the tools examined are still in the prototype phase of development. However, some of them have reached already a high level of maturity and have been commercialized. As the examples can serve CheckOff-E, CheckOff-M, LAMBDA, VFORMAL and YATC.

5. REFERENCES

GENERAL

[1] J.L. HEIN: « Discrete Structures, Logic, and Computability », Jones and Bartlett Publishers, Inc., 1995

[2] J.P. CLEAVE: « A Study of Logics », Oxford Science Publications, Clarendon Press, Oxford, 1991

[3] A. GUPTA: « Formal Hardware Verification Methods: A Survey. », Formal Methods in System Design, 1, 1992, pp. 151 - 238

[4] P. CAMURATI, P. PRINETTO: « Formal Verification of Hardware Correctness: Introduction and Survey of Current Research. », IEEE Computer, pp. 8 - 19, July 1989

[5] C.-J. SEGER: « An Introduction to Formal Hardware Verification », http://www.cs.ubc.ca/tr

[6] D. BORRIONE, H. EVEKING: « Formal Verification of Hardware Designs », Journal of the Brazilian Computer Society, Special Issue on Electronic Design Automation, November 1995

[7] D. BORRIONE, F. CORNO, P. PRINETTO: « A Tutorial: Formal Methods for VHDL », Euro-DAC'94 with Euro-VHDL Conference, Grenoble, 1994

[8] V. STAVRIDOU: « Formal Methods in Circuit Design », Cambridge University Press, 1993

[9] G. MUSGRAVE, R. HUGHES, D. DUNCOMBE: « Review of Verification Techniques », http://www.ahl.co.uk

[10] A. SALEM: « Vérification formelle des circuits digitaux décrits en VHDL », Thèse de doctorat, UJF, Grenoble, October 1992

[11] R. KUMAR, C. BLUMENRÖHR, D. EISENBIEGLER, D. SCHMID: « Formal Synthesis in Circuit Design - A Classification and Survey », Formal Methods in Computer-Aided Design, FMCAD '96, pp. 294-309, Springer Verlag, November 1996

BSNVERI

G.L.J.M. JANSSEN: « Application of BDDs in Formal Verification », Proceedings of the 22nd International School and Conference on CAD, pp. 49-53, Yalta-Gurzuff, Ukraine, Mai 1995

G.L.J.M. Janssen: « Direct Interpretation of Hardware Descriptions by Boolean Function Manipulation with Applications to Verification », Proceedings IEEE/ProRISC Workshop on Circuits, Systems and Signal Processing, Houthalen, Belgique, 8-9 Avril 1992

C@S

K. Schneider and T. Kropf: « A unified approach for combining different formalisms for hardware verification », Proceedings of the International Conference on Formal Methods in Computer-Aided Design (M. Srivas, ed.), LNCS, Palo Alto, USA, Springer Verlag, November 1996

K. Schneider and T. Kropf: « Verifying Hardware Correctness by Combining Theorem Proving and Model Checking », Technical Report SFB358-C2-2/96, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, December 1996

CheckOff-E, CheckOff-M, LAMBDA

R.L. Harris, M.P. Fourman, G. Musgrave: « Core Tools for the Next Generation of Electronics CAD », Proceedings of the Electronic Design Automation Conference, A.P.Ambler(ed.), 1988

R.B. Hughes, G. Musgrave: « Design-Flow Graph Partitioning for Formal Hardware/Software Codesign », Codesign Computer-Aided Software/Hardware Engineeing, J. Rozenblit et K; Buchenrieder (eds.), IEEE Press, 1995

ELLA Verification Environment

H. Barringer, G. Gough, B. Monahan, A. Williams, M. Arcus, A. Armstrong, M. Hill: « A Design and Verification Environment for ELLA », CHDL95, Tokyo, Japan, August 1995

H. Barringer, G. Gough, B. Monahan, A. Williams: « Formal Verification Support for the ELLA Hardware Description Language », CHARME95, Frankfurt, Germany, October 1995

FANCY

S. KRISCHER: "Methodes de verifcation de circuits digitaux", Ph.D. thesis, INPL Nancy, CRIN/INRIA-Lorraine, 1994

FVG BDD Tool

C.A.J. van Eijk, G.L.J.M. Janssen: « Exploiting structural similarities in a BDD-based verification method », Proceedings of 2nd International Conference on Theorem Provers in Circuit Design (TPCD94) (T. Kropf and R. Kumar, eds.), Vol. 901 of Lecture Notes in Computer Science, Bad Herrenalb, Allemagne, pp. 110-125, Springer Verlag, 1995.

C.A.J. van Eijk, J.A.G. Jess: « Exploiting Functional Dependencies in Finite State Machine Verification », Proc. of the European Design and Test Conference ED&TC 1996, Paris, France, 11-14 Mars 1996, pp. 9-14

HOL

M. J. C. Gordon: « HOL: A Proof Generating System for Higher-Order Logic », Current Trends in Hardware Verification and Automated Theorem Proving, edited by G. Birtwistle and P. A. Subrahmanyam, Springer-Verlag, 1989, pp. 73-128

M. J. C. Gordon, T. F. Melham (eds.): « Introduction to HOL: A theorem proving environment for higher order logic », Cambridge University Press, 1993

ICOS

R. Schlör: « A Prover for VHDL-based Hardware Design », IFIP State-of-the-Art Seminar on « Hardware Specification, Verification, and Synthesis », Bangalore, Inde, Janvier 1996

W. Damm, B. Josko, R. Schlör: « Specification and Verification of VHDL-based System-level Hardware Designs », IFIP State-of-the-Art Seminar on « Hardware Specification, Verification, and Synthesis », Bangalore, Inde, Janvier 1996

LOVERT

A. Bartsch, H. Eveking, H. J. Faerber, M. Kelelatchew, J. Pinder, U. Schellin: « LOVERT - A Logic Verifier of Register-Transfer Descriptions », Proceedings of Formal VLSI Correctness Verification, L. Claesen, ed., pp. 247-256, North-Holland, 1990

H. Eveking: « Axiomatizing Hardware Description Languages », International Journal of VLSI Design, Vol. 2, No. 3, pp. 263-280, 1990

MONA

D. Basin, N. Klarlund: « Hardware Verification Using Monadic Second-Order Logic », Technical Report RS-96-7, BRICS, 1995

J. G. Henriksen, M. Jørgensen, J. Jensen, N. Klarlund, R. Paige, T. Rauhe, and A. Sandholm: « MONA: Monadic Second-Order Logic in Practice , 1st TACAS Workshop, Aarhus, 1995

PREVAIL

D. Borrione, L. Pierre, A. Salem: « Formal Verification of VHDL Descriptions in the PREVAIL environment », IEEE Design and Test of Computers, Juin 1992

D. Borrione, H. Bouamama, D. Deharbe, C. Le Faou, A. Wahba: HDL-Based Integration of Formal Methods and CAD Tools in the PREVAIL Environment", Proceedings of the International Conference on Formal Methods in Computer-Aided Design (M. Srivas, ed.), LNCS, Palo Alto, USA, Springer Verlag, November 1996

SMOCK

D. Déharbe, D. Borrione: « Symbolic Model Checking of VHDL Design Entities », Raport RR 925 -I-, Institut IMAG, Décembre 1993

D. Déharbe: « Model Checking of Finite State Machines : Extensions and Applications to VHDL Designs », First Asian-Pacific Conference on Hardware Description Languages: Standards and Applications, Brisbane, Australia, December 1993

ST - Synchronized Transitions

J. Staunstrup: « A Formal Approach to Hardware Design », Kluwer Academic Publishers, 1994

Staunstrup, N. Mellergaard: « Localized Verification of Modular Designs », Formal Methods in System Design, Vol.6, p. 295-320, Kluwer Academic Publishers, 1995

SVP

L. Arditi : « Spécification et preuve des microprocesseurs », Ph.D. Thesis, Université de Nice, Sophia Antipolis, Laboratoire I3S, October 1996

L. Arditi et Hélène Collavizza : « Towards Verifying VHDL Descriptions of Processors », in « European Design Automation Conference with EURO-VHDL », Brighton, UK, September 1995, IEEE Computer Society Press, pages 414-419

TACHE

C. Bayol, J.-L. Paillet: « Using TACHE for proving circuits », Participants Proceedings, IMEC-IFIP International Workshop on Applied Formal Methods for Correct VLSI Design, Vol. 2, pp. 585-589, Leuven, Belgique, 13-16 November 1989

C. Bayol, J.-L. Paillet: « TACHE : a Tautology CHEcker for the P-Calculus », R.R. MAIUP N° 98-07, Marseille, France

VCG - Verification Condition Generator for VHDL

P.T. Breuer, L. Sanchez Fernandez, C. Delgado Kloos: « Proof Theory and a Validation Condition Generator for VHDL », Proceedings from Euro-DAC'94 with Euro-VHDL'94, IEEE CS Press, 1994

P.T. Breuer, L. Sanchez Fernandez, C. Delgado Kloos: « A simple denotational semantics, proof theory and a validation condition generator for VHDL », Formal Methods in System Design, 7 (1 & 2), July 1995

VFORMAL

O. Coudert, C. Berthet, and J. C. Madre : « Verification of Sequential Machines Using Boolean Functional Vectors », Proceedings of IFIP International Workshop on Applied Formal Methods for Correct VLSI Design, L. Claesen, ed., Leuven, Belgium, pp. 111-128, Novembre 1989.

« VFORMAL User Guide », Compass Design Automation, March 1995

VHDL2NQTHM

R.S. Boyer, J.S. Moore: « A Computational Logic Handbook », Academic Press, New York, 1988

D. Borrione, H. Bouamama, R. Suescun: « Validation of the Numeric_Bit package using the NQTHM theorem prover », Proceedings of APCHDL'96 Conference, 1996

YATC

C.A.J. van Eijk, G.L.J.M. Janssen: "Exploiting Structural Similarities in a BDD-based Verification Method.", Proc. 2nd International Conference on Theorem Provers in Circuit Design '94

[1] The survey do not take into consideration the formal synthesis tools presented e.g. in [11].

[2] PREVAIL is the formal hardware verification environment integrating the LOVERT, SMOCK and VFORMAL tools, as well as a diagnostic system.