



*FP6-IST-507219*

## **PROSYD:**

*Property-Based System Design*

Instrument: Specific Targeted Research Project

Thematic Priority: Information Society Technologies

### **Summary Report on Experiences with WP2 (Deliverable 2.3/3)**

Due date of deliverable: December 31, 2006

Actual submission date: December 25, 2006

Start date of project: January 1, 2004

Duration: Three years

Organisation name of lead contractor for this deliverable: IBM

Revision 1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	<b>Public</b>	<input checked="" type="checkbox"/>
PP	<b>Restricted to other programme participants (including the Commission Services)</b>	<input type="checkbox"/>
RE	<b>Restricted to a group specified by the consortium (including the Commission Services)</b>	<input type="checkbox"/>
CO	<b>Confidential, only for members of the consortium (including the Commission Services)</b>	<input type="checkbox"/>

## Notices

For information, contact Mark Moulin, [markm@il.ibm.com](mailto:markm@il.ibm.com)

This document is intended to fulfil the contractual obligations of the PROSYD project concerning deliverable 2.3/3 described in contract number 507219.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

© Copyright PROSYD 2006. All rights reserved.

## Table of Revisions

Version	Date	Description and Reason	By	Affected Sections
0.1	19.11.2006	First draft by authors	R.Bloem, M.Moulin, K.Winkelmann	All
0.2	28.11.2006	Second draft by authors	R.Bloem, M.Moulin, K.Winkelmann	All
0.3	30.11.2006	Edited Introduction	R.Bloem	Introduction
1.0	25.12.2006	Final version	Cindy Eisner	All

## Authors

Roderick Bloem  
Mark Moulin  
Klaus Winkelmann

## Executive Summary

This document provides a summary of PROSYD experience with case studies on the property-based synthesis and error-localization tools derived in work package WP2. Two synthesis and four error-localization case studies are discussed. The performance and usefulness of algorithms and tools are evaluated; and the conditions on tools efficiency are discussed.

## Purpose

The purpose of this document is to describe the work done with case studies on the property-based synthesis and error-localization tools. The document summarizes the experiences of using the tools on six examples, and draws conclusions on the tools' applicability and dissemination, and future research and development.

## Intended Audience

This document is intended for researchers working on property-driven synthesis and error-localization tools, as well as for formal verification engineers and hardware designers who use such tools. Basic knowledge of the property specification language, PSL, or LTL is assumed.

## Background

The WP2 is devoted to property-based design, which includes synthesis of a design from a property and debugging properties that fail during verification. In deliverable D2.3/1, a novel property-based synthesis algorithm was tested on two case studies. In deliverable D2.3/2 a tool for helping a designer detect bugs, based on failure of a property, was tested on four case studies. This document provides an overall view and conclusions drawn from the WP2 case studies.

# Contents

Table of Figures .....	iv
Glossary .....	v
1 Introduction.....	1
2 Summary on experience with WP2.....	4
3 Discussion and Conclusions .....	6
4 References.....	8

## Table of Figures

Figure 1: The PROSYD design flow.....	1
Figure 2: Design Flow.....	2

# Glossary

## **Assertion**

A property that is expected to hold true on a specific design.

## **BuFi**

Bug Finder. A tool for automatic localization of fault candidates for sequential circuits at the gate or HDL level. The tool was developed by Stefan Staber of Graz University.

## **Design**

A model of a piece of hardware, described in a hardware description language (HDL).

## **Failure/Bug/Error**

An incorrect result.

## **HDL (Hardware Description Language)**

One of several specialized high-level languages used by semiconductor designers to describe the features and functionality of chips and systems prior to handoff to the IC layout process. Currently, the two standard HDLs in use worldwide are Verilog HDL and VHDL.

## **Property**

A collection of logical and temporal relationships between expressions involving design signals, representing a set of behaviours.

## **PSL**

Property Specification Language, the language for specifications of designs upon which PROSYD is based.

## **Specification**

The process of defining the expected behaviour of a hardware design.

## **Synthesis**

The process of automatically generating a design from a given specification. Formally, checks if the given specification is realizable and finds a witness.

## **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.



# 1 Introduction

The aim of PROSYD is to provide an integrated design flow, based on PSL. As shown in Figure 1, the design flow is divided into three phases: specification, design, and verification. The purpose of the specification work package is to provide the user with a means of efficiently writing good specifications. The verification work package deals with methods to ensure the correctness of a handwritten circuit efficiently. The design work package is concerned with the construction of correct circuits. It consists of two major tasks: error localization and property synthesis [1-7]. Both aim to generate correct circuits, as opposed to verifying the correctness of a given circuit.

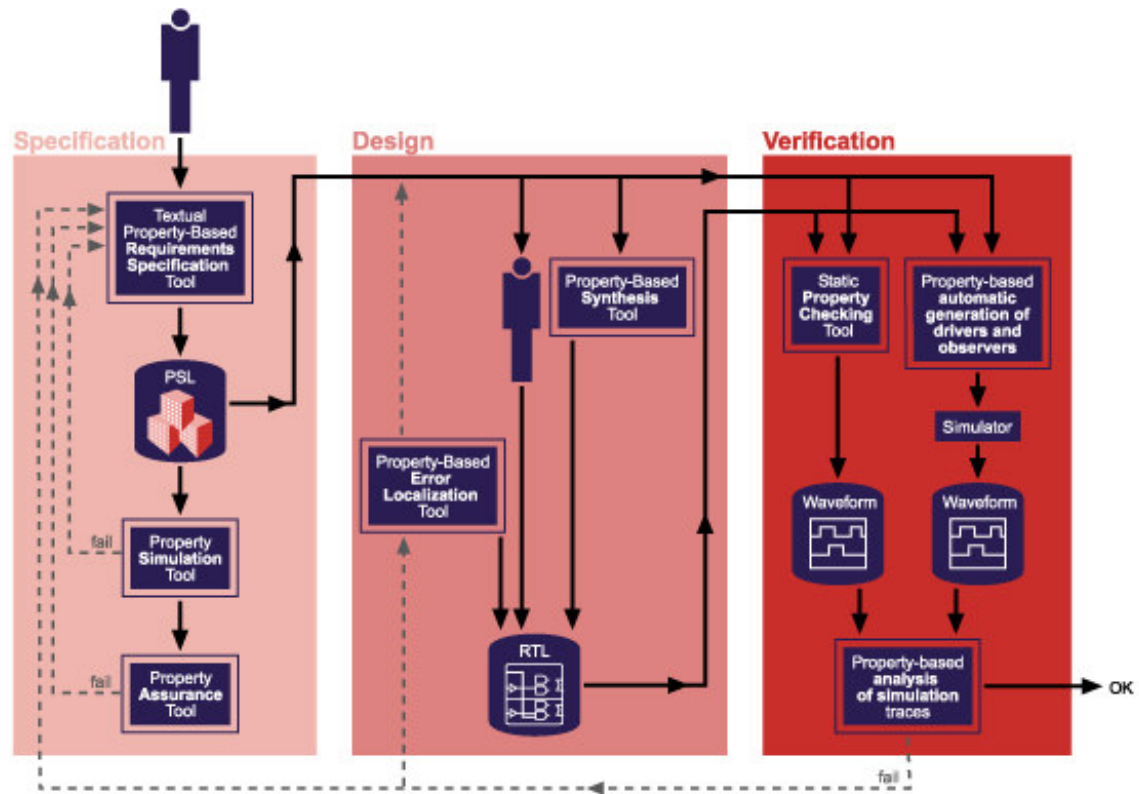


Figure 1: PROSYD design flow

As the complexity of digital designs increases, debugging takes an ever greater role in the design process. Debugging consists of fault detection, localization, and repair. Thus, apart from fault detection, which is handled in the verification work package, we need to focus on automatic means for **property-based fault localization** and repair.

When a design is faulty, the dynamic and static verification tools studied in the verification work package yield an error trace showing an input sequence for which the design violates its specification. Even when such a trace is available, finding and fixing a fault remains a time-consuming chore, which accounts for 50-80% of the time spent in verification and about one third of the overall time spent in designing a system (see Figure 2).

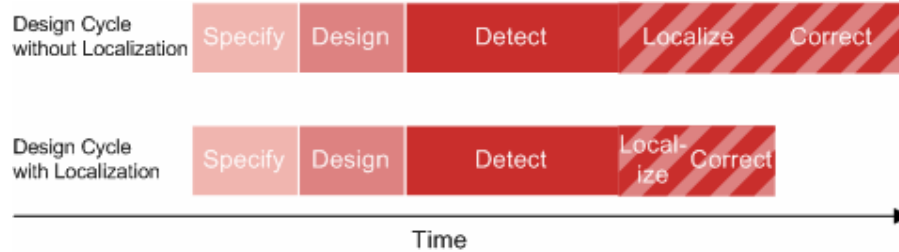


Figure 2: Design Flow

Property-based fault localization is the process of isolating parts of the design likely to be responsible for the failure of a verification run. Thus, the designer's attention is focused on that fraction of the code where the fault is suspected to be. Thus, it aims to reduce design time by reducing the time spent in debugging.

Although debugging is a major task in system design, it received relatively little attention in the formal verification community prior to the advent of PROSYD. Within PROSYD, we tackled the problems of fault localization both theoretically [5] and practically, in the form of a tool, called BuFi (for Bug Finder) [7].

BuFi was evaluated at OneSpin and at IBM [2]. The case studies revealed both benefits and limitations of the tool. At OneSpin, we performed the case study on a priority-driven arbiter. This case study helped us further develop the tool, by extending the range of faults with which it can deal. Furthermore, we evaluated the effort needed for debugging a protocol processor and found that the time needed for manual fault localization is relatively minor. This emphasizes the need for further research into strengthening the tool to return fewer potential fault locations.

At IBM, we evaluated BuFi on a FIFO arbiter [8] and a buffer. The case study analyzes how a user can help the tool to adapt more efficiently to the design for which it is applicable, using four tests to measure the effect of changes to the tool input.

Our work on fault localization has been acknowledged internationally; our paper on fault localization won the best paper award at the 2006 Haifa Verification Conference [9]. It should be noted that the methodology is very novel and needs further research before it can be included in the standard design flow.

The goal of **property-based synthesis** is even more ambitious: to automatically construct a correct design from its specifications, thus removing the need for formal verification. This way, we can avoid hand-coding HDL code altogether. We can also provide a designer with a fully functional circuit as soon as the specifications are available. This has two major benefits. First, the user can immediately validate the specification. Second, the designer can build optimized blocks of logic in a fully operational environment, which leads to fewer bugs when integration is performed. It should be noted that although the need for verification of the design is removed, property-based synthesis does not remove the need for validation, i.e., of making sure that the specification meets the user's requirements.

Thus, property-based synthesis supports the development of systems with higher quality, within shorter times, and at lower costs.

Until recently, property-based synthesis has been considered a purely theoretical subject. It was studied in theoretical computer science conferences; implementations did not exist. Only in recent years, and to a major extent within the PROSYD project, has property synthesis been made to work for realistic examples. We have developed several synthesis algorithms, which are described, along with the necessary theory, in [3]. We have also provided a very fast tool for a subset of LTL as well as the first tool that can synthesize full LTL [4].

We present a synthesis case study of the tool in [1]. The case study applies PSL to synthesize two designs from their specifications: the arbiter for ARM's on-chip AMBA Advanced High-Performance Bus [10], and on a generalized buffer (GenBuf) from IBM [11]. During the work, we built formal specifications for the circuits. Usually, the standard (English language) specifications are informal and incomplete, and many aspects of the circuit are not defined or missing.

This problem was apparent for the ARM specification of the AMBA arbiter and less critical for GenBuf of IBM, for which a formal specification had been provided by IBM as part of their verification efforts. The formal specifications of the AMBA arbiter and GenBuf are short, readable, and easy to modify, which is as they should be according to the methodology proposed in D1.1/1 [12] on property-driven methodology for VLSI design. It should be noted that this is the first time an industrial example is automatically synthesized from its specifications.

Our research on property synthesis has received considerable attention. The algorithms have been presented at the 2006 conference on Verification, Model Checking, and Abstract Interpretation [13], and at the 2006 conference on Formal Methods in Computer-Aided Design [14]. A report of the case study has been accepted for the conference on Design Automation and Test in Europe [15].

In summary, we tackled two very novel topics in property-based design. Further research is needed in both areas. In particular, we need to find methods to increase the discriminatory power of property-based fault localization and to further increase the capacity of the property-based synthesis tools. We would like to stress, however, that considering the novelty of the problems and the approaches that we developed, both should be considered a great success. Property synthesis, in particular, turns out to be far more practical than we expected at the beginning of the project. We will continue to pursue this subject with vigour and expect to see major advances in the years to come.

## 2 Summary of Experience with WP2

This section discusses highlights of the experiences described in D2.3/1 [1] and D2.3/2 [2].

The TU Graz has spent 21PM on a synthesis tool development in D2.2/3 [6], and 4PM on case studies D2.3/1 [1]. The TU Graz approach to the property-based synthesis based on an automatic high-level synthesis that generates correct-by-construction Verilog code directly from the PSL specification. D3.2/1 [1] demonstrates the viability of the synthesis approach on examples of the ARM AMBA bus [10] and the IBM generalized buffer (GenBuf) [11].

The AMBA bus represents a moderate size characteristic industrial case [10]. According to the TU Graz testing protocol, a synthesis of an arbiter for two masters takes about half a minute, and synthesis for three arbiters takes about ten minutes. The output is a Verilog gate-level description of about 200kB for a two-master arbiter and 800kB for a three-master arbiter.

The generalized buffer of IBM GenBuf [11] was developed and used by IBM as a tutorial design for the RuleBase PE verification platform. TU Graz synthesized the control logic of GenBuf according to the original complete specification. Generally, the GenBuf consists of a controller, a FIFO, and a multiplexer. During the case study, only the controller was synthesized from its specification. The FIFO and multiplexers were considered as standard pieces of logic with standard implementation

In the case study's conclusion, [1] TU Graz points that the AMBA arbiter specification was informal and incomplete. They say that it is not always trivial to translate an informal specification to PSL formulas. There was less of a problem with GenBuf, where a good specification in PSL had already been constructed. The tool proved the correctness of the property-based synthesis concept. It can be graded as an academic tool of a good quality. Still, much effort must be invested in tool maturity and dissemination among hardware design centres. The next target is to receive tractable circuits for large unit size systems.

During the error-localization case studies D2.3/2 [2], IBM spent 4PM and OneSpin spent 2PM on the error-localization tool evaluation. TU Graz spent 4PM on technical and methodological support to the partners. OneSpin worked mostly with the first prototype of the tool, and provided much useful feedback. IBM began to work with a second prototype, and invested a lot in the stabilization and maturity process. The interoperability of the partners during this case study created a unique development atmosphere, which permit an academic exploratory idea to grow within a short space of time into a self-contained prototype of the tool.

OneSpin spent a total of two person-months evaluating the error localization algorithm and tool developed by TU Graz. OneSpin created the evaluation from the scope of industrial verification projects. It also provided representative examples for TU Graz to test the algorithms.

At the beginning of the evaluation, the protocol processor design was chosen as a test bench, but it was rejected as too complex for the first prototype of the error localization tool (algorithm). Therefore, a much simpler design of a priority-driven

arbiter was selected. The case study with the arbiter design helped TU Graz to extend and generalize the algorithm, and resulted in correctly localizing errors for the arbiter. It demonstrated the capabilities and also some limitations of the approach, which were unlikely to be gained from a larger case study. This was no surprise as the error model used in this initial version of the algorithm did not take missing-line errors into account since its fault model was limited to incorrect expressions. It turned out that the failure of a single property provides too little information for a focused diagnosis, and a more complete set of properties is needed.

Next, the error-localization tool was tested on the protocol processor design described in D1.4/1 [16]. OneSpin concluded that using their technology, it isn't difficult to localize the bug, understand its cause at the RT level, and fix it. Hence, while a good automated error localization tool would be nice, it is not amongst the most pressing issues in the overall verification activity.

IBM spent a total of four person-months evaluating the error localization algorithm and tool developed by TU Graz. IBM created the evaluation on the Buffer design from the scope of research and development verification projects. Buffer is a derivative design of GenBuf.

During the test runs, the tool marked approximately 25% of the code, and all the diagnoses included the erroneous line. Since the design is small and has few effective lines of code, many of the lines could affect each of the failed properties. A few insights were derived from the case study for efficient use of the tool: a complete set of specifications is extremely useful; and given too many diagnoses, one can improve the results by adding stronger assertions.

Altogether, based on this case study, the tool is helpful when the bug is caused by an error in a single line. The tool is less applicable when the bug is a result of a combination of errors in several lines of code or when the bug is caused by a conceptual mistake. When using the tool, it could be very helpful to have relevant specifications of the design.

The second case study of a real FIFO design [8] was designed to bridge two clock domains. The design was written in Verilog, and IBM and Graz attempted to adapt the design to Bufi. There were several technical difficulties in doing this, mostly because it is a real design that uses complicated Verilog constructs for reasons of performance. The adaptation (rewriting) required substantial human resources, so it was decided only to define a future expansion of Bufi capabilities to a real subset of Verilog. On one hand, this case study shows the limitations of Bufi and its distance from minimal capability as an industrial and academic tool; but on other hand, it points to directions for future development.

## 3 Discussion and Conclusions

The experiences with property-based synthesis and error-localization tools demonstrate that the PROSYD research and development team have withstood many challenges during the course of WP2. As opposed to other packages where the developments were based on relatively well developed methodologies, algorithms, and tools; the WP2 activity for the first time in EDA research connected the synthesis and error-localization tasks, and declared them a goal not only for academic research, but also for industrial tooling. We think that this upgrade of property-based synthesis and error-localization research will draw the attention of academic and industrial partners to these issues.

The evaluation results show that both techniques acquired a good algorithmic basis during the PROSYD research. Property-based synthesis provided a final product—a synthesized circuit. This circuit belongs to the industrial AMBA bus design. TU Graz derived a methodology of selecting the necessary properties for easy and efficient implementation D2.3/1 [1], D2.2/1 [4], D2.2/3 [6]. This methodology connects a synthesis task with WP1 methodologies, where the same AMBA AHS bus was used in D1.1/1 [16]. A comparison of this case study with a GenBuf example shows the necessity of efficient construction of a complete set of properties. The interesting point is that while a property-driven synthesis was considered at the beginning of the project as exploratory research, the results went beyond this framework and the real end-products—synthesized circuits—were produced.

In comparison to the synthesis technique, the error-localization technique had a more developed background. Its goal was to help the designer debug the counter-example traces. The resulting tool, Bufi D2.2/4 [7], D2.2/2 [5] shows good potential for following development before issue to market. On the other hand, the case study results are relatively modest. The tool was checked on tutorial and industrial designs. It points correctly to the problematic area, which is obviously the same area that would be manually chosen by an experienced designer. Hence, the next tasks for the error-localization tool will be improving performance and sharpening the bounds of problematic areas. Here, we admit that the error-localization tool is less market-mature than the synthesis tool, which is the opposite of our projections at the beginning of the project.

While both tools are far from being disseminated through design and verification houses, they can be distributed as a part of PROSYD tools to increase interest in the property-driven synthesis and error-localization directions. It is important to note that we are situated at the beginning of development, and PROSYD activity only opens the development throughout CAD centres.

In all, PROSYD partners spent a total of 41PM on developing the tools and 17PM on case studies. The relatively low ratio compared to development and case studies for other packages shows that developers anticipate much useful feedback from case studies. The TU Graz, OneSpin, and IBM teams tested the tools on carefully chosen designs. The synthesis case study on the AMBA bus is important as a case study on open source code. The Buffer block derived from the IBM GenBuf synthesis case study was used in verification case studies. These relatively simple

examples with a closed PSL specification allow a deep analysis of testing on a small size design and filtered faults in algorithms implementation. They also point to two methodological issues, i.e., the importance of specification completeness, and the possibility of over-performance of error-localization when the tool marks even pieces of design code distantly related to the bug. The protocol processor from OneSpin was a tough test for the error-localization tool, and this test example will serve as a test bench for a future tool version.

The case studies provide direction for future development and research, and correspond to the PROSYD presentation of property-driven synthesis and error-localization as necessary parts of today's model checking development and research.

The property-driven synthesis and error-localization case studies provide good evaluations of the novel research and development during the course of PROSYD project. The case studies were done on tutorials and real designs by TU Graz, OneSpin, and IBM partners. The results show that property-driven synthesis and error-localization provide prospective trends in model checking. The synthesis tool gave a good impression, providing a tractable circuit for tutorial design. The error-localization tool points to issues that have to be addressed before error-localization can become standard practice in hardware verification. It will be useful to disseminate the results of this study among EDA centres to facilitate further research and development in these fields.

## 4 References

- [1] R. Bloem, B. Jobstmann, and A. Pnueli. Evaluation of Tools and Methodology for Property-Based Logic Synthesis, December 2006. PROSYD D2.3/1.
- [2] S. Moran, S. Staber, K. Winkelmann, and K. Yorav. Evaluation of Tools Developed for Error Localization, December 2006. PROSYD D2.3/2
- [3] G. Auerbach, M. Moulin, B. Jobstmann, and R. Bloem. Property-Based Design and Implementation May 2005. PROSYD D2.1/1.
- [4] R. Bloem, B. Jobstmann, and A. Pnueli. Property-Based Logic Synthesis for Rapid Design Prototyping, September 2005. PROSYD D2.2/1.
- [5] R. Bloem and S. Staber. Property-based Error Localization, April 2005. PROSYD D2.2/2.
- [6] R. Bloem and B. Jobstmann. Manual for Property-Based Synthesis Tool, August 2006. PROSYD D2.2/3.
- [7] R. Bloem and S. Staber. Manual for Property-Based Error Localization Tool, September 2006. PROSYD D2.2/4.
- [8] Marvell Technology Group Ltd. URL: <http://www.marvell.com>
- [9] S. Staber, G. Fey, R. Bloem, and R. Drechsler, Fault Localization for Property Checking, *Proc. Haifa Verification Conference*, 2006.
- [10] ARM Ltd. AMBA Specification (Rev. 2). Available from [www.arm.com](http://www.arm.com), May 1999.
- [11] IBM. GenBuf tutorial. Available from [www.haifa.ibm.com/projects/verification/RBHomepage/tutorial3/](http://www.haifa.ibm.com/projects/verification/RBHomepage/tutorial3/).
- [12] S. Ruah, A. Fedeli, C. Eisner, and M. Moulin. Methodology Document on Property-Driven Specification of VLSI design, May 2005. PROSYD D1.1/1.
- [13] N. Piterman, A. Pnueli, and Y. Sa'ar, Synthesis of Reactive Designs, *Proc. Verification, Model Checking, and Abstract Interpretation (VMCAI'06)*, pp. 364-380, 2006.
- [14] B. Jobstmann and Roderick Bloem, Optimizations for LTL Synthesis, *6th Conference on Formal Methods in Computer Aided Design (FMCAD'06)*, 2006.
- [15] R. Bloem, S. Galler, B. Jobstmann, N. Pitermann, A. Pnueli, M. Weiglhofer, Automatic Hardware Synthesis from Specifications – A Case Study, *Proc. Design Automation and Test in Europe (DATE'07)*, To Appear.
- [16] G. Auerbach, L. Benalycherif, A. Fedeli, D. Fisman, A. McIsaac, K. Winkelmann. Case Studies in Property-Based Requirements Specification, November 2006. PROSYD D1.4/1.