



FP6-IST-507219

# PROSYD

Property-Based System Design

Instrument: Specific Targeted Research Project

Thematic Priority: Information Society Technologies

## Port of Infineon Tools

(Deliverable D3.3/3, Public Version)

Due date of deliverable: June 30, 2005

Actual Delivery date: July 24, 2005

Start date of project: 01.01.2004

Duration: 3 years

Organisation name of lead contractor for this deliverable:

**OneSpin Solutions** (replacing Infineon Technologies as PROSYD partner from May 15, 2005, subject to Commission approval)

Revision: 1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2000-2006)		
Dissemination Level		
PU	Public	<input checked="" type="checkbox"/>
PP	Restricted to other programme participants (including the Commission Services)	<input type="checkbox"/>
RE	Restricted to a group specified by the consortium (including the Commission Services)	<input type="checkbox"/>
CO	Confidential, only for members of the consortium (including the Commission Services)	<input type="checkbox"/>

## Notices

For information, contact [klaus.winkelmann@infineon.com](mailto:klaus.winkelmann@infineon.com).

This document is intended to fulfil the obligations of the PROSYD project concerning deliverable 3.3/3, 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 2005. All rights reserved.

## Table of Revisions

Version	Date	Description and Reason	By	Affected Sections
0.1	June 14, 2005	Creation	Kw	all
0.9	June 24, 2005	Integrated C. Pichlers contribution	kw, cp	3
o.9p	July 11, 2005	Public version	kw	all
1.0	July 24, 2005	Approval by project management	Ce	Version number

## Authors

Christian Pichler, Klaus Winkelmann

## Executive Summary

The CVE property checker has been enhanced to support properties written in the PSL language.

## Purpose

The purpose of this document is to report the Infineon contribution to the result of task 3.3, which is “porting of existing tools”.

In this public version, the focus is on the users’ view, and some technical details have been omitted.

## Intended Audience

This report is intended for project partners intending to use Infineon property checker on PSL properties.

It is assumed that readers are familiar with PSL.

## Background

Task 3.3 aims at providing a state-of-the-art tool set for formal verification of designs, based on PSL properties. Four PROSYD partners provide tools for this task, based on their respective background technology.

# Contents

1	Introduction.....	1
1.1	Background .....	1
1.2	Starting-point.....	1
2	Technical approach.....	2
2.1	Basics of ITL .....	2
3	User view .....	3
3.1	Tool usage .....	3
3.2	Example run .....	3
4	Outlook .....	6
5	References.....	7

## Table of Figures

Figure 1:	Counterexample wave-form.....	5
-----------	-------------------------------	---

# Glossary

**DUV:** Design under Verification. An RT level model of a hardware design.

**HDL:** Hardware Design Language, such as VHDL, Verilog.

**ITL:** InTerval Language, Infineon's proprietary property specification language.

**Macro machine:** a finite state machine which remains in a stable state as long as all its inputs remain stable.

**Model Checking:** The automatic or almost automatic verification of a property of a model of a hardware component or in some cases of software.

**SAT:** satisfiability, the question whether a given set of Boolean formula has a solution.

**SAT solver:** algorithm that decides a SAT problem and returns a solution if it exists.



# 1 Introduction

---

## 1.1 Background

Task 3.3 aims at providing a state-of-the-art tool set for formal verification of designs, based on PSL properties. Four PROSYD partners provide tools for this task, based on their respective background technology.

---

## 1.2 Starting-point

At start of the PROSYD project, Infineon had an efficient industrial-strength property checker called GateProp, which is part of the CVE tool environment. GateProp checks properties written in the ITL language, a proprietary language of Infineon tailored to simply express so called finite-window properties, i.e. properties stretching over a finite number of time-steps.

# 2 Technical approach

This section describes the approach to porting GateProp to PSL. The basic decisions are:

- GateProp will continue to support the ITL language
- PSL is internally converted to ITL.

In this public version of the report, we omit the details of the latter conversion.

---

## 2.1 Basics of ITL

An ITL property is essentially a constraint on the design's signals over a finite time interval. To be valid, a property needs to hold for every observation window of the appropriate length, in every run. Using a proprietary combination of algorithms such as bounded model-checking, the tool checks each property and, if it is found to be not valid, produces a counter-example.

The basic concepts of the language are:

**Time steps:** A property is written over a number of time steps, from time  $t$  (i.e.  $t+0$ ) to a future time (e.g. time  $t+4$  after 4 clock cycles). Consequently, there are a few time constructs in the language (e.g. `at t`, `during[t, t+2]`, etc.).

Each property consists of a "prove" and an "assume" part.

**Assume part:** This part allows the designer to specify the working mode of the design under inspection. Assumptions such as 'no reset occurs at time  $t$ ' are typically necessary to investigate if the design exhibits a particular behavior. Further typical assumptions are 'at time  $t$  the input connection\_request is high' or 'there will be no write\_request during time interval  $t+1$  to  $t+5$ '.

**Prove part:** This part of the property specifies expected behavior. Typical assertions in the prove-part are 'the grant output is set at time  $t+5$ ' or 'the write\_acknowledge output will somewhere be issued within time interval  $t+1$  to  $t+3$ '.

There are a number of language extensions that are designed to result in *concise but intuitive* properties, including data quantifiers, a powerful macro mechanism and time variables. Local variables ("for", "freeze") can be declared in a property, and assigned any expression.

# 3 User view

This section details the knowledge required for using the PSL-GateProp tool., including the exact grammar.

---

## 3.1 Tool usage

The PSL front-end has been integrated with the development version of CVE. To use it, after installing CVE the following command is used:

```
> gateprop --psl -t <vunit-name> <model> <psl-file>
```

Here <model> is a file generated from the design sources by one of the CVE front-end tools. How to use those front-end tools, as well as further options for `gateprop` are detailed in [CVE05].

---

## 3.2 Example run

Here we present a small example. First a property file is shown. It refers to a small artificial example that was created by the consortium for the purpose of demonstrating the PSL design flow at the May 2005 review in Graz. The design represents a basic resource controller and has the following interface.

```
port (  
    reset    : in  std_ulogic;  
    clk     : in  std_ulogic;  
    req     : in  std_ulogic;  
    cancel  : in  std_ulogic;  
    go      : in  std_ulogic;  
    grant   : out std_ulogic  
);
```

The following properties are written to the file `simple.psl`:

```
-- PROSYD mini example  
-- properties by K. Winkelmann  
-- May 4, 2005  
  
vunit demo1 { -- this is supposed to fail, to show a buggy property  
    assert {req = '1'} |=> {grant = '1' [*3]};  
}  
  
vunit demo2 {  
    assert ({[*1]; grant = '1'} |=> {grant = '0'});  
}
```

```

vunit demo_always {
--psl: verification_window = 1000;
-- bounded proof
assert always not {[*1]; grant = '1'; grant = '1'}! ;
}

vunit demo2b { -- induction step
assume not {grant = '1'; grant = '1'}! ;
assert not {[*1]; grant = '1'; grant = '1'}! ;
}

-- detecting the design bug (simulation style!)
vunit demo4 {

assert { {[*1]; cancel = '1' };
        {go = '0'[*] ; go = '1' and req = '0'; req = '1'; [*3]}
        &&
        {cancel = '0' and reset = '0' [*]}
}
|->
{grant = '1'};
}

-- property declaration and induction proof in one vunit.
vunit demo2de{
property not_grant_after_grant_at_t (const tp) is
{[*tp]; grant = '1'} | => {grant = '0'};
assume not_grant_after_grant_at_t(1) or {[*1]; reset='1'} ;
assert not_grant_after_grant_at_t(2);
}

```

The following is results are produced when compiling and checking this example:

```

***** executing PROSYD SIMPLE DEMO - compilation *****
vhdl2gates -m syn_macro simple
VHDL2GATES Version 3.999-after(121)
Copyright (c) 1997-2005 by Infineon Technologies AG. All rights
reserved.
Tauri(tm) Copyright 1995-2004 FTL Systems, Inc. Copyright 1999-2004
FTL Systems UK Ltd. (10 . 5. 100 Release)
-I- Alternative subprogram bodies in library 'cve', package
'substitute' read
-I- Compiling to GAT...
-I- Starting exact computation of macro machine ...
-I- ... finished after 2 steps (0.00 sec CPU, 0.0 MB)
-I- GatWrite - GAT `simple-rtl' written to file `./gat/simple-
rtl.gat'.
-I- FSM contains 5 inputs + 1 output + 4 states + 20 internal nets.
-I- Gate-Equivalent: 63 (NAND) gate(s).
-I- Terminating with no errors and warnings, 8.17 sec CPU, 12.8 MB.
-R- VHDL Compiled with no warnings or errors

***** executing PROSYD SIMPLE DEMO - demol *****
gateprop --psl -x -t demol -D demol.dmp gat/simple-rtl.gat
simple.psl
GATEPROP Version 3.999-after(9)
Copyright (c) 1998-2005 by Infineon Technologies AG. All rights
reserved.
-I- File `gat/simple-rtl.gat': 129 lines read in 0.01 sec.
-I- Generating standard clocking scheme for clock 'clk'.
-I- Simulating clocking scheme.
-I- Property file `simple.psl' processed in 0.02 sec CPU.
-I- Size of assumption: < 20 vars, < 1000 gates.

```

```

-I- Size of commitment: < 20 vars, < 1000 gates.
-I- Size of property: < 20 vars, < 1000 gates.
-I- Assumption is not contradictory, checked in 0.02 sec.
-R- The property fails, 0.03 sec CPU in total, 6 MB used
-I- Wrote .vcd file to `demo1.dmp'.
-I- Wrote customization for Dinotrace to file `demo1.dino'.

***** executing PROSYD SIMPLE DEMO - demo2 *****
gateprop --psl -x -t demo2 -D demo2.dmp gat/simple-rtl.gat
simple.psl
GATEPROP Version 3.999-after(9)
Copyright (c) 1998-2005 by Infineon Technologies AG. All rights
reserved.
-I- File `gat/simple-rtl.gat': 129 lines read in 0.01 sec.
-I- Generating standard clocking scheme for clock 'clk'.
-I- Simulating clocking scheme.
-I- Property file `simple.psl' processed in 0.01 sec CPU.
-I- Size of assumption: < 20 vars, < 1000 gates.
-I- Size of commitment: < 20 vars, < 1000 gates.
-I- Size of property: < 20 vars, < 1000 gates.
-I- Assumption is not contradictory, checked in 0.01 sec.
-R- The property holds, 0.01 sec CPU in total, 6 MB used.

```

When a property fails, a trace is displayed as follows. Also other presentation, using commercial wave-form viewers are supported.

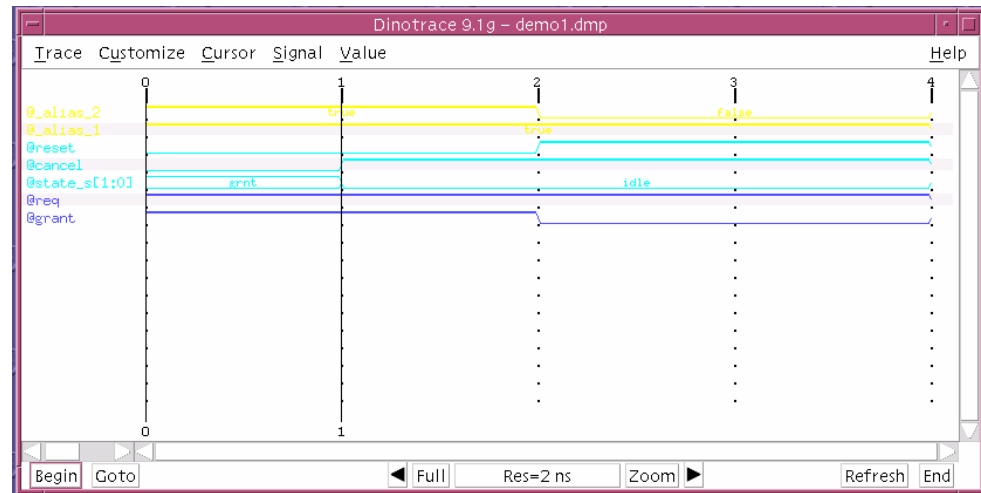


Figure 1: Counterexample wave-form

# 4 Outlook

The PSL front-end is available and stable and shows good performance on the existing test-bench. Next steps will be

- case-studies: further, more practical tests need to be carried out to see how well the tool can prove industrial designs.
- Proof scheme: to prove infinite properties with the bounded checker, induction has to be used. It will be studied how inductive proofs can be supported on top of the current implementation.
- Debugging: for productive use, the user needs a good debugger in case of failed properties. Concepts for such need to be studied and implemented.

# 5 References

- [D3.1/1] Research report on improved decision heuristics for high-performance SAT-based static property checking. PROSYD report 2004
- [D3.2/2] Research report on improved symbolic search strategies and model reduction for static property checking. PROSYD report 2005
- [D3.2/3] Research report on exploitation of RT information in static property checking algorithms. PROSYD report 2004
- [Bri01] R. Brinkmann, Using Symmetry for Problem Reduction in Bounded Model Checking on the Register-Transfer Level, SymCon 01, Paphos, Cyprus, 2001.
- [CVE05] Infineon CVE team, On-line documentation for the CVE tool set. Available as part of the CVE product.