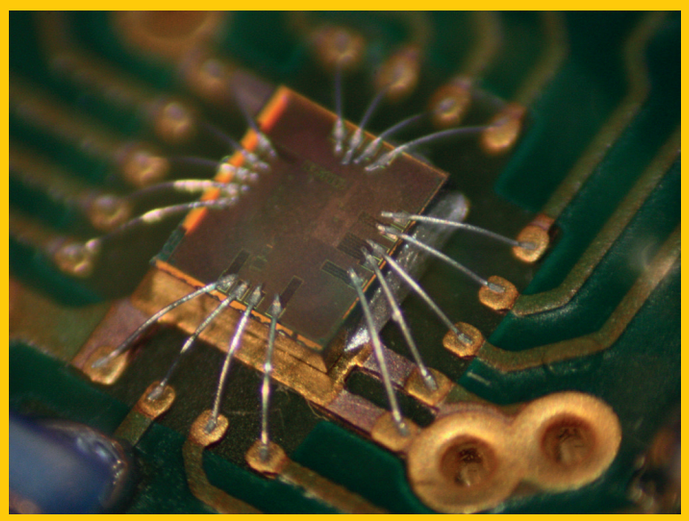
VDE VERLAG

ANALOG 2016

266

ITG-Fachbericht

Beiträge der
15. ITG/GMM-Fachtagung
12. – 14. September 2016
in Bremen



Informationstechnische Gesellschaft im VDE (ITG)

GMM

VDI

ITG INFORMATIONSTECHNISCHE GESELLSCHAFT IM VDE

VDE

Titelfoto: S. Mahlknecht; TU Wien: WUR Receiver

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.dnb.de** abrufbar.

ISBN 978-3-8007-4265-3

9 783800 742653

ISSN 0932-6022

www.vde-verlag.de

CD-Produktion: DMS – Disk Media Service, Berlin
Produced in Germany

ITG-Fachbericht **266**

ANALOG 2016

Beiträge der 15. ITG/GMM-Fachtagung
12. – 14. September 2016 in Bremen

Veranstalter:
Informationstechnische Gesellschaft im VDE (ITG)
VDE/VDI-Gesellschaft Mikroelektronik, Mikrosystem- und Feinwerktechnik (GMM)

Tagungsleitung:
Christoph Grimm, Technische Universität Kaiserslautern

VDE VERLAG GMBH

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.dnb.de** abrufbar

ISBN 978-3-8007-4265-3
ISSN 0932-6022

Production: DMS – Dinsk Media Service, Berlin, Germany
Produced in Germany

Inhaltsverzeichnis

Keynotes

Verifikation von Robustheit

Synthese und Layoutgenerierung

Analoge Schaltungen in intelligenden Sensorsystemen

Schaltungen und Technologie

Verifikation von Mixed-Signal Systemen

Modellbasierte Entwicklung und Simulation von Systemen

Poster

Metrics for Formal Property Checking Against Undesired Circuit Behavior in Embedded Systems

Michael Rathmair and Florian Schupfer
TU Wien, Institute of Computer Technology, 1040 Vienna, Gusshausstrasse 27-29/384, Austria
Michael.Rathmair@tuwien.ac.at, Florian.Schupfer@tuwien.ac.at

Abstract

Modern embedded systems, including analog and digital circuits, strongly rely on the verification of the intended system functionality. Property checking, as a formal verification methodology may prove the correct behavior of design subparts. Due to scalability issues, a dedicated selection of characteristics to be checked and constrictive model complexity is required for keeping the verification effort reasonable. In this work we propose checking for undesired functionalities, whether they are intentionally (debug artifact), unintentionally (hardware Trojan) or due to reuse of functional modules present in the design. We define measures (abstracted costs) which may be used for effective verification planning. Characteristics are rated on a common knowledge base, revisioned over past design projects in combination with statistical runtime estimation. A resulting subset of cost efficient properties is finally handed over to an automatic checking tool.

1 Introduction

Today, the functional complexity of embedded systems increased to a level where segmentation of the full development task and reuse of design components became a necessity. This opens new vulnerabilities for interference behavior in hardware designs. Unspecified system functionalities may be caused by functional composition of modules and integration of new customer specific implementations [10]. Strictly speaking, sources for unwanted behavior may lead from undocumented lines of code in a behavioral description, over unused obfuscated circuit structures, debug and monitoring structures, to hardware Trojans which may enable attackers to compromise system functionalities. Finally, after refinement and synthesis of the model the represented behavior is implemented in silicon hardware structures. In this work predefined formal properties, describing unintentional characteristics as formulas are verified whether they are satisfied on a model. One of the major advantages of applied property checking algorithms is that verification requires no generation of specific test stimuli [3, 4]. As illustrated in Fig. 1 test based verification aims to check whether single functionalities satisfies their specification under strictly defined system states and input signals. Property checking, allows to describe abstract characteristics of objected behavior resulting in an area covering compositions of functionalities [11, 4].

Main investigation of this paper is not to increase the performance of analog formal verification itself, but to introduce a metric which proposes an approach to assess and estimate effort for verifying properties in a design (see Section 3). The proposed procedure may guide a verification engineer to decide which property formulas are checked on the model. Resulting counterexamples may act as start points for debugging and optimization and finally increase the confidence of the system. Section 4 gives a demonstration example where results are summarized in order to illustrate a formal verification planning strategy.

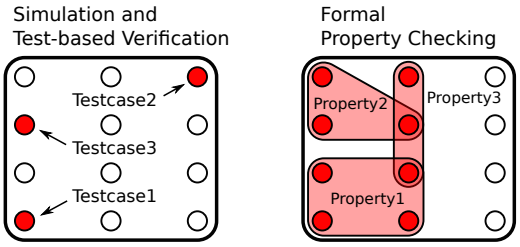


Figure 1 Abstract illustration of a specific design space. Test and simulation based verification is aimed to check a specific behavior under defined input stimuli. Formal property checking can show that a specific characteristic is satisfied (not satisfied) on a subpart of the full design.

2 Formal Property Checking of Analog Functionalities

Hardware property checking, is deduced from software verification purposes and adapted for verifying circuits. Therefore, the hardware behavior is represented in a formalized, mathematically precise and unambiguous manner [1, 11]. Applied algorithms systematically explore feasible states of the system and guarantee (mathematical proof) that defined characteristics hold for all possible input signals. For digital circuits, Finite State Machine (FSM) representations are unwinded and property formulations (p) are evaluated if they are satisfied on the resulting flattened model M, ($M \models p$) [1]. For analog functionalities a key task is the discretization of the continuous state space. Depending on the selected algorithm and resolution this causes an assessable discretization error [2]. An

ISBN 978-3-8007-4265-3

approach published in [3] is to represent the state space by a discrete graph structure. The reachable state space is partitioned into hypercubes of homogeneous behavior. Each hypercube is defined by derivations of system variables and represent vertices in the graph structure used for model checking [3].

Properties are defined in a formal language, which enables the description of logical and temporal characteristics. In this work, we make use of temporal logics such as Linear Temporal Logic or Linear-time Temporal Logic (LTL) and Computational Tree Logic (CTL) for property specification. For analog model checking a more specific Analog Specification Language (ASL) including enhanced formulations for offset, gain, slew rate, etc. characteristics may be used [3]. Model checking tools automatically prove the satisfaction of given formulas on the system model independent of input stimulus [4, 11, 1].

For this paper we follow a top-down refinement system development approach. Specified functional components are modeled using a highly abstracted behavioral description as SystemC-AMS, System Verilog, VHDL-AMS etc. These models represent an executable specification and allow verification, simulation and test processes in a very early design phase. This first coarse-grained model is refined by architectural- and circuit-level synthesis processes and finally implemented as a piece of hardware [8, 9].

As illustrated in Fig. 2, the full set of verification properties is divided into a limited set of desired properties P_d covered by the specification and an unbound set of undesired ones P_u.

Hence, an exhaustive verification of all feasible undesired characteristics is not possible. In the following we introduce an abstract cost level which indicates the effort invested in formal design verification. These costs can be equally viewed as a confidence value for the absence of unwanted characteristics during design phase and may act as measure for future designs.

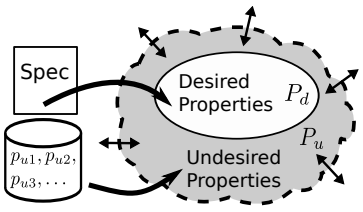


Figure 2 The set of desired properties, defined by the specification, is a limited area inside an infinitely expandable property space. Undesired characteristics are illustrated as a surrounding infinite cloud.

Single undesired properties $p_{u1}, p_{u2}, p_{u3}, \ldots p_{uK}$ are stored in a knowledge base, revisioned over several design projects. Each of the proposed property formulas describe a possibly included unspecified behavior caused by functional components itself or new composition of features for a new product release (e.g. unspecified signal reset, utilization of unused bits in communication frames, debug and monitoring structures used for the development of a component, etc.) Stored characteristics are reused, slightly adapted and applied on new designs.

3 Effort Estimation and Verification Planning

To estimate costs and efforts for formal property checking, we propose an abstract metric. Within this measure, developers and researchers may be able to specify a level of hardware verification confidence which correlates with effort spent on formal property checking. The proposed values will also rate the impact of undesired behavior for the overall goal of a functional correct design. Hence, the discussed values can be seen as abstracted without any direct influence to the design process itself. Virtually calculated costs C may be directly mapped to real development costs and may guide engineers to estimate cost of verification.

As a first step, complexity considerations advise partitioning the full design into functional blocks. Formal checking processes performed on a full design model results in vast checking costs. Thus, property checking, as used in this work, is applied on subsystems of the full design. We categorize a selected subsystem as a representative of a specific field of function or application. Specified undesired properties for these are collected into sets $P_{us1,2,3,\ldots}$ (see Fig. 3). Each specified undesired property is assigned to at least one segment $(p_{uk} \in P_{u1}) \vee (p_{uk} \in P_{us2}) \vee \ldots \vee (p_{uk} \in P_{usM})$, $\forall k = 1 \ldots N$. Total abstract checking costs for a system C are composed of subcosts for checking each segment $C = \sum_{x=1}^{M} C_{P_{usx}}$, where M is the number of defined segments.

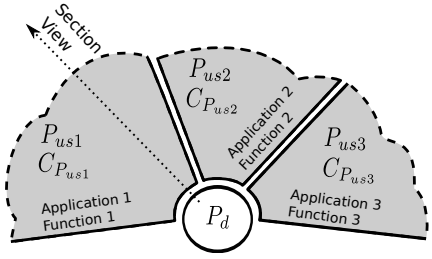


Figure 3 Segmented undesired property space of a design. For each segment representing a specific function or application a total cost value may be calculated for covering included undesired characteristics.

As a second step β_{pu} is defined which is an occurrence probability of a specific property formulation. β_{pu} rates the number of positive hits at previous verification runs to the number of totally applied checking runs. For the construction of a *section view*, as illustrated in Fig. 3, all properties in set P_{us1} are considered. For the computation

of a section view diagram, which reflects a segment cost characteristic, occurrence probabilities β_{pu} and checking costs C_{pu} of a property are plotted. Obtained functionalities have a maximum cost level of γ which is an initial static offset value when defining the undesired property costs.

Besides the calculation of β_{pu}, a further requirement for the construction of a section view diagram is to assess the verification costs C_{pu} of each property. In this work we restrict in checking CTL (computation tree logic) and LTL (linear temporal logic) property formulations. For CTL and LTL, complexity considerations are well described in literature [5, 6]. The size of the formal model $|S|$ has a main influence on checking time (number of states and transitions). The length of the property formula $length(\varphi)$, which is normally very small compared to the model size, has a minor impact on checking effort. Finally algorithmic complexities θ_{LTL} and θ_{CTL} are defined using the $\mathcal{O}$ notation [5, 6].

$$\theta_{CTL} = \mathcal{O}(|S| \cdot length(\varphi))$$
$$\theta_{LTL} = 2^{\mathcal{O}(length(\varphi))} \cdot \mathcal{O}(|S|)$$

This complexity value can be precalculated for each given model/property combination without execution of the specific verification tool.

For the estimation of checking duration at a dedicated calculated complexity value θ, an estimation function

$$f_\tau(\theta) = a \cdot \theta^b$$

is introduced. Coefficients a and b are evaluated by reading verification time consumption τ and complexity θ values from the described knowledge base. Quadratic residuals between prerecorded checking times and the estimation function values will be reduced under the constraint $min\left(\sum_{i=1}^{K} |\tau_i - f_\tau(\theta_i)|^2\right)$.
Thus, with the help of $f_\tau(\theta)$ we get an average time estimation for checking a selected property, verified on a new system respecting the complexity θ.
For the translation of this duration value into a cost quantity a conversion factor fc is introduced. fc is initially defined by engineering and computation costs for each single property. Due to model reuse, property order heuristics, etc. this factor is not constant and may be refined after each verification run. Finally, average abstract checking costs for a dedicated property may be estimated by $C_{pu} = f_\tau(\theta) \cdot fc$.
As shown in Fig. 4, for each property included in segment $C_{P_{us1}}$, a point is plotted in the section view. For a cost characteristic measure of a segment, an exponential function

$$fs(C_{p_u}) = e^{-k(C_{p_u} - \gamma)}$$

is constructed. Parameter k in this function representing the section metric is evaluated under the constraints that β_{pu} must be less or equal than fs for all properties included in the section, and fs has a static node at $(\gamma, 1)$:

$$\{fs(C_{p_u} \leq \gamma) = 1\} \wedge \{\beta_{p_{ui}} \leq fs(C_{p_{ui}})\}, \forall p_{ui} \in P_{usj}$$

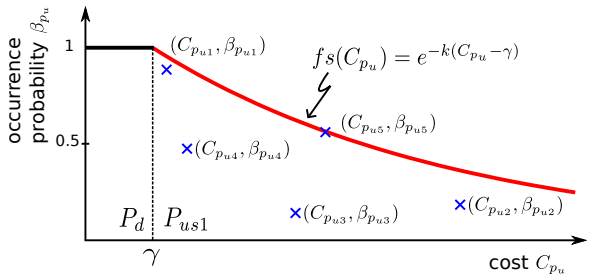


Figure 4 Section view as indicated in Fig. 3. Properties included in segment $C_{P_{us1}}$ are plotted in the diagram. For rating the segment, function $f_s(C_{p_u})$ is constructed where k represents a segment metric

As a result of defining occurrence probabilities, abstract costs and segmentation effort measures the following qualitative statements can be specified.

Refinement:
The characteristic measure k is mainly influenced by upper rightmost points in the segment view (Fig. 5-a). In other words, unintended properties which were often satisfied in previous projects cause a high checking costs characteristic for the full segment. As illustrated in Fig. 5-a, the property p_{u5} is refined and hence replaced by p_{u6} and p_{u7}. This may reduce occurrence probabilities but may increase total costs if checking of both refined properties is intended. After the refinement, p_{u2} determines the construction of the exponential characteristic function. Parameter k changes to a larger value $k' \geq k$. A faster decrease of the occurrence probability in respect to checking costs, rates the according segment as being cheaper. New refined properties have no prerecorded history of their usage. Thus, occurrence probabilities may be estimated by enhanced static model analysis.

Selection of a maximum cost level:
If a dedicated minimum occurrence probability $\beta_{p_{uMin}}$ is required, the segment measure function fs returns a maximum cost level C_{puMax}. This cost value is not exceeded by any single property where $\beta_{pu} \geq \beta_{p_{uMin}}$ is fulfilled.

Verification planning procedure:
More interesting for verification planning is the evaluation of total checking costs if a minimum occurrence probability β_{puMin} is required. Which properties are selected for a checking process if a limited verification budget is given? Summarized costs for checking properties of a segment are limited to $C_{P_{us}}$, and costs for each single property must not exceed C_{puMax}.

$$\left\{\sum_i (C_{pui} - \gamma) \leq C_{P_{us}}\right\} \wedge \{C_{pui} \leq C_{puMax}\}$$

To guarantee a desired minimum historical occurrence probability β_{puMin}, all properties in shaded area 1 (Fig. 5-b) have to be checked. It may happen that the checking budget for the segment $C_{P_{us}}$ is not fully consumed after considering all properties of area 1. The rest of the effort is used to select checking properties in shaded area 2, extending from

ISBN 978-3-8007-4265-3

low to high costs. A selection rule in this case is given by covering properties having lower cost first in order to increase the quantity.

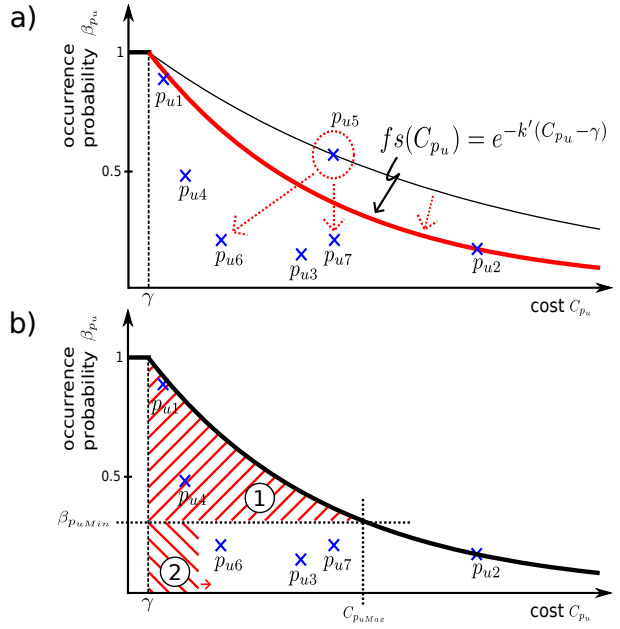


Figure 5 a) Refinement of unspecified property p_{u5} to 2 properties having a lower occurrence probability (refinement of the check). b) Planning procedure, for covering undesired properties below a defined verification budget.

4 Demonstration Example

The demonstration example selected for this work is a system generating pulse with modulated (PWM) signals for three phase motor control applications. As indicated in Fig. 6 control signals driving the gate inputs of the power amplifier stages are derived from continuous analog signals levels (SigIn1…3). These are compared with a periodic sawtooth signal. To avoid short circuit switching periods between the upper and the lower side of the power amplifier stages a signal delay circuit is inserted before the lower side transistor's gate inputs (M4, M5, M6).

The created behavioral model for property checking is highly abstracted and can be seen as an executable circuit specification. Analog voltages are discretized with an 10 bit resolution and represented by integer variables ranging from 0 to 1023. The sawtooth generator is represented by an up-counting value in combination with an appropriate overflow logic. Lower side gate delay elements are described by additionally derived sawtooth signal compare values. A silicon circuit implementation may result after various manual/automatic refinement and synthesis steps based on this high level description.
As described in the previous sections our planning methodology relies on a database where prior verification timing results are stored in. Thus, for this demonstration example we checked the same design properties also on three other circuit models: A single sawtooth generator

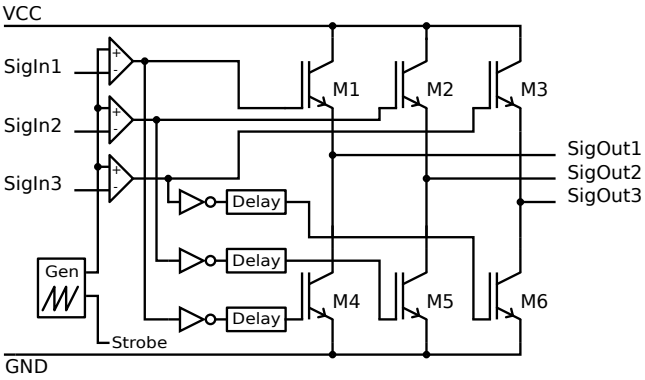


Figure 6 Demonstration example for this paper is a three phase motor control circuit. Signals driving the gate inputs of the power transistors are derived form analog voltages SigIn1…3.

just generates an upcounting value with reset as used in Fig. 6. A timer unit as used in a microcontroller including overflow and capture and compare functionalities. A RGB LED dimmer circuit which generates three PWM signal for continuous illumination and colour levels. These circuits are assumed as previous projects and will provide verification data (time consumption values for $f_\tau(\theta)$). Properties and corresponding verification timing results are stored in a knowledge base, realized as an XML file. A structural overview of the XML file is shown in Fig. 7. Each property holds a set of checking runs including additional detailed information as the specific property formula, runtime, system complexity, etc.

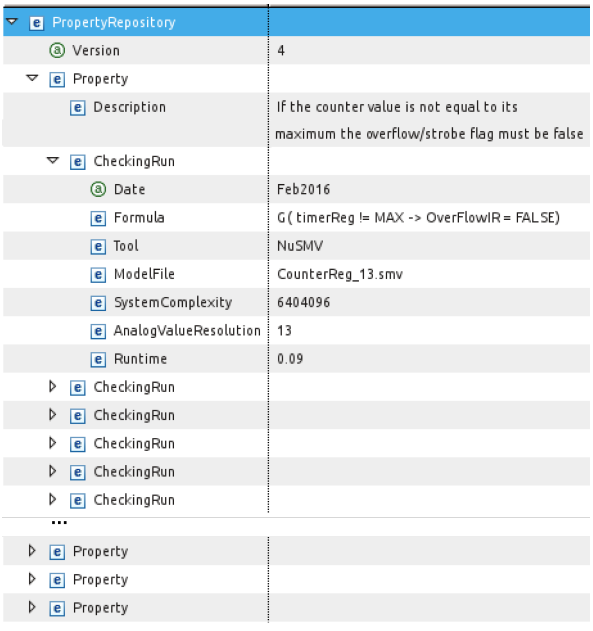


Figure 7 Structural illustration of the used XML repository including the properties and results of previous model checking runs.

As a set of undesired properties P_u, nine formulas $p_{u1} \ldots p_{u9}$ distributed into two segments $P_{us1} = \{p_{u1}, \ldots, p_{u4}\}$ and $P_{us2} = \{p_{u5}, \ldots, p_{u9}\}$, are

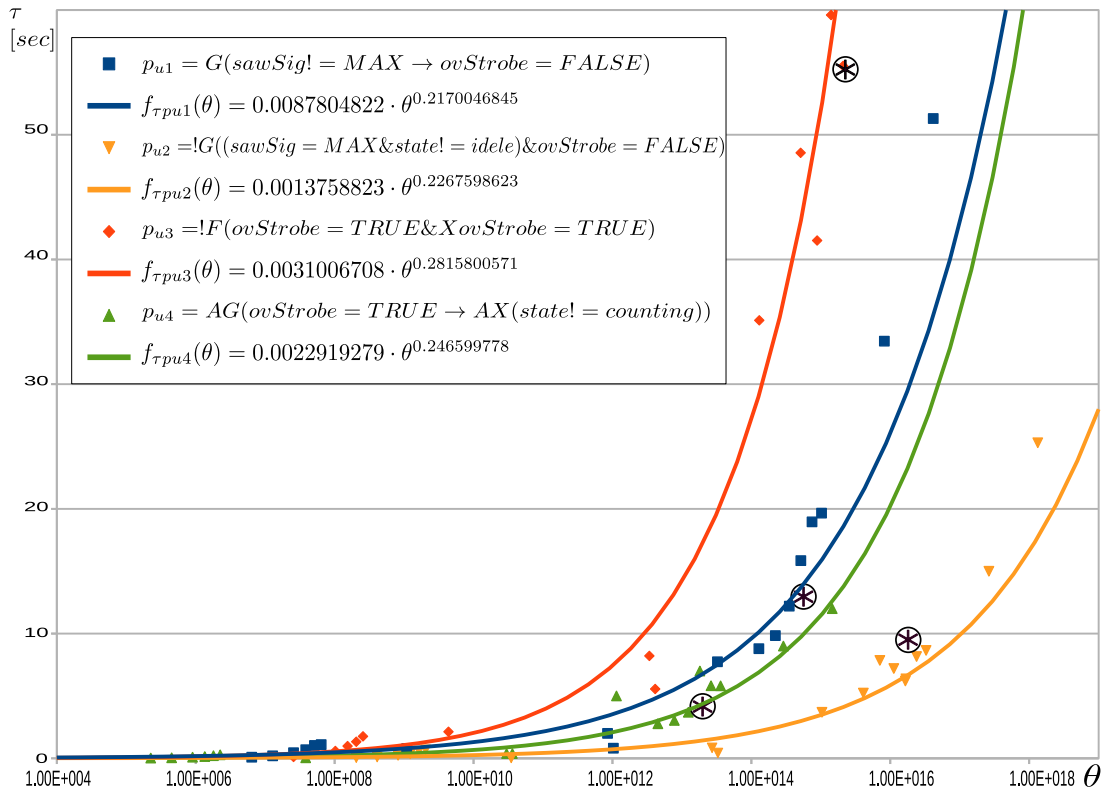


Figure 8 Time consumption for checking $p_{u1} \ldots p_{u4}$ applied on previous system models. Lines represent estimation functions $fs(C_{p_u})$, used for verification planning procedures. Data points highlighted by an asterisk mark real measured checking duration values for $p_{u1} \ldots p_{u4}$ on the new motor control design obtained from the used model checking tool.

expressed and handed over to the academic model checking tool NuSMV [7]. The checked properties represented as LTL and CTL formulas are stated in Fig. 8. They check whether unwanted functionalities of the strobe signal *ovStrobe*, which is set at the maximum of the sawtooth signal, is not satisfied on the model. The rest of this example concentrates on the proposed effort measures not detailing in the content of the checked formulas. Also other licensed commercial tools (e.g. Cadence) can be used for verification of the defined properties. Differences in verification software are covered within the results stored in the repository. The proposed methodology is independent of tool performance or implemented algorithms.

Fig. 8 illustrates time consumption values for checking properties $\{p_{u1}, \ldots, p_{u4}\}$ included in segment 1 on the systems assumed as previous projects (sawtooth signal generator, timer and LED dimmer applications). For extended creation of verification data filling the repository for past projects we additionally modified the resolution of analog signal discretizations (10 to 16 bit). Thus, we got a selection of 19 different system implementations. The verification tool is executed sequentially resulting in the verification execution time data τ at a specific system complexity θ for each property $\{p_{u1}, \ldots, p_{u4}\}$. Approximated functions plotted in the diagram show timing estimation functions $f_\tau(\theta)$. Coefficients a and b of $f_\tau(\theta)$ are evaluated under respect of the minimization constraint of residuals.

The automata model representing the discretized analog state space and the behavior of the new motor control system implements 1027 transitions and $2.2033 \cdot 10^{12}$ reachable system states. To validate the timing estimation functions $f_\tau(\theta)$ we checked the given properties on the new system (motor controller). The asterisk in Fig. 8 mark real measured timing results obtained from the verification tool. This shows, that under given system complexities θ the calculated timing approximation functions based on previous projects result in good verification time estimations. Assuming a cost rating factor $fc = 10$ the following abstract costs for checking $p_{u1} \ldots p_{u4}$ are calculated:

$$C_{p_{u1}} = 136.793$$
$$C_{p_{u2}} = 65.456$$
$$C_{p_{u3}} = 636.141$$
$$C_{p_{u4}} = 42.618$$

Resulting in total checking cost for the segment P_{us1}

$$C_{P_{us1}} = 881.008$$

Next, we calculate occurrence probabilities in order to create a section view for P_{us1} and evaluate the section effort function $fs(C_{p_u})$. As described previously in total we checked 19 systems assuming them as previous verification projects. Each design has been verified 100 times resulting in 1900 checking runs for each property formulation. Within this 1900 verification runs, we assume 3, 16, 1 and 12 property violations for $p_{u1} \ldots p_{u4}$. By calculating

ISBN 978-3-8007-4265-3

the dedicated occurrence probabilities $\beta_p u$, the according P_{us1} section view can be constructed (see Fig. 9). Property point p_{u3} in the diagram is defining an exponent value k of 0.011869477 to fulfill the previously described constraint for $fs(C_{p_u})$.

$$fs(C_{p_u1}) = e^{-0.011869477(C_{pu}-25)}$$

Maximum checking costs for desired functionalities γ are defined as 25. As a verification planning strategy, we evaluate checking cost of segment P_{us1} for a minimum occurrence probability of $\beta_{puMin} = 0.5\%$ (checked level in Fig. 9). Hence, property p_{u4} and p_{u2} has to be verified which results to a summarized abstract cost value of 108.074. This cost value represents a measure for planning and management procedures for this particular design and may be directly translated into real verification costs.

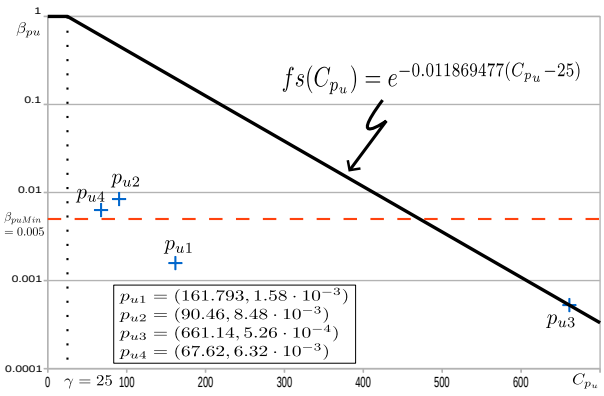


Figure 9 Section view for segment P_{us1}, indicating costs and occurrence probabilities for $p_{u1} \ldots p_{u4}$ (β_{pu} axis in logarithmic scale).

5 Conclusion and Future Work

In this paper we introduce effort and cost metrics for formal property verification. Methodologies are applied during the design phase and mainly objected to systems which behavior is given by a composition of previously implemented functional blocks. Thus, for verification planning we propose not just checking intended functionalities of a system but also verifying if undesired (maybe harmful) properties are not fulfilled. This work is not about improving the verification performance but rather in a target orientated selection of verification properties to optimize costs in projects. The selection of properties to be checked is based on a common repository, revisioned over several design projects. Finally, checking time and effort represented as an abstract cost value are evaluated by statistical methods and monitored execution characteristics of verification runs. Results, as effort metrics and planning strategies may be used for future design processes, but scalability improvements are depend on enhancements in model checking techniques and will directly influence the industrial applicability of the proposed methodology.

For future work, main investigation is to apply the proposed verification planning methodology at a small real industrial design project. Long term studies may show if the calculated metrics and proposed property selection methods result in an additional benefit.

6 References

[1] Christel Baier and Katoen Joost-Pieter, “Principles of Model Checking”, MIT Press, 2008.

[2] S. Steinhorst and L. Hedrich, “Trajectory-Directed discrete state space modeling for formal verification of nonlinear analog circuits,” 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, 2012, pp. 202-209.

[3] E. Barke and D. Grabowski, et.al. , “Formal Approaches to Analog Circuit Verification,” Design, Automation & Test in Europe (DATE), , 2009, pp.724-729.

[4] William K. Lam, “Hardware Design Verification: Simulation and Formal Method-Based Approaches” (Prentice Hall Modern Semiconductor Design Series), Prentice Hall PTR, 2005.

[5] E. M. Clarke and E. A. Emerson and A.P. Sistla, “Automatic verification of finite-state concurrent systems using temporal logic specifications”, ACM TOPLAS vol. 8(2), 1986.

[6] Ph. Schnoebelen, “The Complexity of Temporal Logic Model Checking”, Advances in Modal Logic, pp. 393-436, 2002.

[7] Roberto Cavada and Alessandro Cimatti and Charles Arthur Jochim and Gavin Keighren and Emanuele Olivetti and Marco Pistore and Marco Roveri and Andrei Tchaltsev, “NuSMV 2.5 User Manual”, FBK-irst, 2010.

[8] S. A. Huss, “Analog circuit synthesis: a search for the Holy Grail?,” 2006 IEEE International Symposium on Circuits and Systems, Island of Kos, 2006, pp. 4 pp.-1466.

[9] F. Paugnat, K. Morin-Allory and L. Fesquet, "A refinement process for top-down mixed-signal designs thanks to SystemC-AMS," New Circuits and Systems Conference (NEWCAS), 2011 IEEE 9th International, Bordeaux, 2011, pp. 378-381.

[10] V. Sklyarov and I. Skliarova, “Reuse Technique in Hardware Design”, 2007 IEEE International Conference on Information Reuse and Integration, Las Vegas, IL, 2007, pp. 36-41.

[11] Thomas Kropf, “Introduction to Formal Hardware Verification”, Springer-Verlag New York, Inc., 1999.

