International Journal of Control and Automation
Vol. 9, No. 4 (2016), pp. 223-232
http://dx.doi.org/10.14257/ijca.2016.9.4.22

SFV: A Scalable Approach to Formal Verification for AMS SoC

Jingbo Shao[1, 3], Yongqing Fu[1] and Shikai Wang[2*]

[1]*Postdoctoral Research Station of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China*
[2]*College of Mathematical Sciences, Harbin Normal University, Harbin 150025, China*
[3]*College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China*
*zro_bo@163.com, *wsk_1250@163.com*

Abstract

This paper proposes a new scalable approach to formal verification (SFV) for AMS SoC designs. Induction rules defined in computer symbolic algebra system Maple are followed to extract representation of property of AMS SoC, constraints solving is performed to formally verify the correctness of the system with respect to its given property. With an AMS description and a set of properties, SMT based model checker is applied to validate if the given system has a certain properties. The proposed methodology is applied on a tunnel diode oscillator. Experimental results for a benchmark and a tunnel diode oscillator demonstrate the effectiveness of the approach. The proposed method is scalable to other AMS system using language VHDL-AMS as description tool.

Keywords*: AMS SoC, formal verification, Maple*

1. Introduction

Nowadays with the ever-increasing complexity of *System-on-Chips* (SoC), analog and Mixed-Signal (AMS) designs are becoming more and more complicated in electronic devices. Analog devices are small but may cause a lot of problems, it is reported that 20% - 80% fault comes from analog circuits. Moreover our ability to understand complex systems grows slower than our ability to assemble them; verification of analog circuits is becoming a bottleneck for AMS design and plays an important role in today's industrial design flow.

The verification of AMS designs concerns the correct functionality in addition to checking whether an AMS design is robust with respect to different kinds of inaccuracies like noises, nonlinearities, etc. However, due to the fact that time-to-market is shrinking, re-spins during post silicon phase incur a huge cost in terms of time and money, additionally formal verification meets with a couple of issues, *i.e.*, interaction between digital and analog blocks are extremely complicated and simulation for AMS design can be in weeks, it is late when bugs are found. It is significant and time-urgent for verification researchers and engineers to provide high-quality designs to reduce test costs and enhance development efficiency.

Verification of digital circuits has changed tremendously in the past ten years while AMS circuit verification remains mostly unchanged. Researchers and engineers for verification are responsible for developing novel and economical methods to shorten verification cycle and reduce related costs. Solutions to formal verification for AMS designs include unifying digital verification flows and analog verification flows, reusing

* Corresponding Author

ISSN: 2005-4297 IJCA

design and verification environment.

CAD tools for AMS designs have been proposed to overcome the difficulties occurred during the process of development, *e.g.*, to designing and improving the quality of more complicated integrated circuit to meet time-to-market and improve productivity. Thus CAD tool and concept helps people to understand behavior and characteristics of integrated circuit deeply, learn to use CAD tool to handle these crucial behaviors and model for AMS design correctly and efficiently.

Recently, breakthrough has been achieved for different aspects of CAD process, especially for the development of hardware description language suitable for various AMS behaviors. Typical tool as VHDL-AMS has been applied for application. Different from digital design, verification runs through different phase of the process of the whole design. Moreover the property and requirements for AMS design differs from types to type. The work for verification involves of illustrating equivalent behavior between devices and proving property conformance, *i.e.*, requirements for area and power dissipation should be considered and handled for the correctness of the design.

Techniques of simulation and test for formal verification of AMS designs have been tapped in [1]. Traditionally implementation of simulation is usually done manually, and the exploration of state space is not complete. However such method lack the rigor to ensure the correctness of design, and it does not ensure the correctness of communication implementation and the model on the subsequent design level or the two models on the same level either.

Theorem proving is also referred to as proof based methodology [2]. The designers utilize axioms and reasoning rules to construct the mathematical proof which meets the description of the models or structure. Therefore theorem proof is a powerful technology for verification; in that it provides a uniform framework for verification tasks on different levels. Whereas tasks of complex theorem proof require the efforts and originality of the users together with the aid of experts.

Exploration of state space mainly includes equivalence test and model checking [3]. Equivalence test compares the output signals of two different models of a certain design for a given input conditions. In comparison with theorem proof, it does not need the research of mathematical proving. But the correctness of such method relies on the exploration of reachable state space and the comparison. For model checking, the model for the design to be verified is a transitional system to describe all its possible behaviors. Algorithms for state exploration are used to check if the model meets the given properties. The counter-example can be generated once the property is violated. Then the design can be corrected and re-verify the design. The advantages of state space exploration are its automation; however it suffers from state space explosion, which limits the application of such approach. The efficiency for state exploration and model checking relies heavily on the size of reachable state space. The bigger the state space, the more time and memory it requires for system verification.

In the past thirty years, formal verification has been applied for digital hardware system and software system. As a technology to overcome the limitation of traditional simulation techniques, formal verification has been suitable for and utilized verification for AMS system [4]. Hybrid semi-formal techniques, which combine simulation and formal methods, have been exploited, where the logic model is used to analyze simulation structure.

Linear transfer equation is proposed in [5] to describe the two designs for analog system for model checking. The proposed method bases on bi-linear transfer and transfer function is discrete to z domain. Therefore the design can be expressed as discrete-time component, and be encoded as finite state machine like binary decision tree.

The author in [6] focuses on the consistency checking for parameter variation. Some researchers utilize equivalence checking for linear analogous circuit to demonstrate that, for a given frequency interval, a real circuit performs description for all the parameter

variations. Transfer function, extracted from netlist using symbol analysis, is applied to describe linear analogous circuit, and forms parameterized description of circuit behavior. Over-approximation for transfer function of the implementation is picked to ensure the correctness of description, and under-approximation for description of transfer function.

Equivalence checking for VHDL-AMS design is presented in [7]. Equivalence checking, rewriting system and simulation are combined into verification environment. Firstly codes for description and implementation are partitioned into digital part, analogous part and data conversion part. Classical equivalence verification methods are adopted for digital part, and rewriting rules and pattern matching simplify the description and implementation of simulation. The approach can be realized for simple designs, while it ignores the coupling between analogous part and digital part.

The authors in [8] use technology for hybrid system to verify AMS design. d/dt is adopted for over-approximation analysis for a system described by differential equations. To overcome memory explosion problem of discrete-time AMS design, optimized control is presented to find out the reachability boundary.

Authors in [9] propose more accurate approximation for reachable states for AMS design, where exploration algorithm for state space is handled using Talor approximation over interval. Such modeling method allows for computing for continuous variable and avoids the irrationality of numeric Talor approximation. Safety and liveness for bounded model checking is analyzed using these techniques and symbol treatment. This method is applicable to discrete-time and continuous-time AMS design.

Automata based logic model for hybrid system is proposed to facilitate analysis safety problem and automatic verification. These kinds of models are characterized by taking the hybrid system as automata in macro-perspective and each state of the automata is a set of differential equations in micro-perspective. When the automata transits to a certain state, the corresponding differential equations work. The whole hybrid system transits among different states of the automata. These models all deal with continuous part of AMS system as linear differential equations. However such models suffer from lacking controllability, observability and stability.

This paper proposes a scalable method for verification of AMS SoC. First, abstractions of circuit behaviors using exhaustive algorithm is formulated for reachability analysis. For one running process with initial status and input signals given, model checking searches the whole state space to demonstrate that all the reachable states meet predefined specifications.

2. The Proposed Algorithm SFV the Proposed Algorithm SFV

2.1. Outline of the Proposed Approach: SFV

Figure 1 shows the flow of formal verification for AMS SoC. Model for the original system is established, and then polyhedral partition and reachable set approximation are performed. The original system is transformed into finite state transitional one. Actions based temporal model is adopted for satisfactory modular theory (SMT) based model checking to judge if the system meets the specifications. The property of AMS design is written as Φ in programming languages VHDL-AMS, and the AMS system is formally represented as M. A model checking algorithm determines whether a mathematical model of a system meets a specification. Or formally, given a model *M* of an AMS SoC and a property *P*, check$M \models P$, *i.e.*, check if *P* holds in the model *M*.

The proposed method searches for a counter-example for a given property verified against the design model *M* for bounded number of verification steps. If a concrete counter-example is found, then refinement is performed to repeat the verification, otherwise, it needs to increase the number of steps until property validation is carried out. Generally speaking, the verification is not complete because of time and memory

limitations. To overcome this problem, we observed that under certain conditions and for some classes of specification properties, the verification can be complete if we complement the BMC with other methods such as abstraction and constraint based verification methods. To test and validate the proposed approaches, we developed a prototype implementation in Maple and we targeted AMS design like tunnel diode oscillator for the tests of the proposed approach.

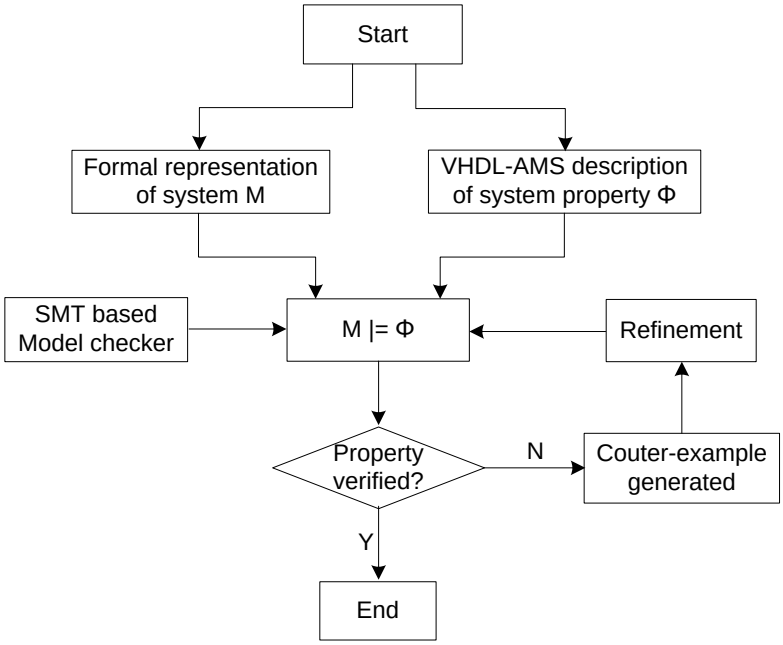

Figure 1. Overview of Formal Verification for AMS Soc

Language VHDL-AMS is capable of modeling AMS SoC, and models described in VHDL-AMS need a mixed-signal simulator which has a continuous-time kernel and a discrete-event kernel. For system-level verification where performance of the simulations is critical, it is advisable to have models which are very efficient if they have sufficient accuracy.

The SMT based model checker accepts a verification property described in VHDL-AMS. The models produced by the model generation tool have these properties embedded in them so that a model checker can verify them formally. The properties in VHDL-AMS are specified using the failure transitions and the models produced by the model generation tool have these properties embedded in the form of assertions.

Tunnel diode oscillator shown as Figure 2 is taken for research target. For some range of voltages, the current of tunnel diode oscillator decreases with increasing voltage as shown in Figure 3. More specifically, when a small forward-bias voltage is applied across a tunnel diode oscillator, it begins to conduct current. As the voltage increases, the current increases and reaches a peak value. If the voltage increases a little bit, the current virtually starts to decrease until it reaches valley value.

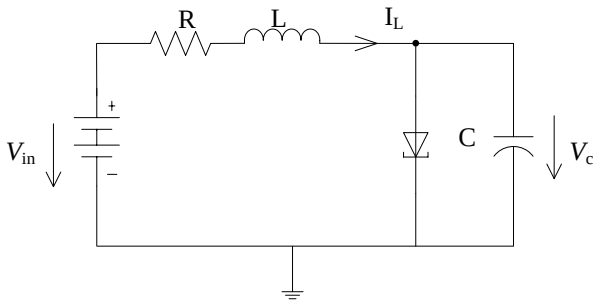

Figure 2. Tunnel Diode Oscillator

Two sets of parameters for the tunnel diode oscillator are chosen as follows:

1) $C = 1000e^{-12}, L = 1e^{-6}, R = 5000e^{-3}, V_{in} = 0.3,$

2) $C = 1000e^{-12}, L = 1e^{-6}, R = 3000e^{-3}, V_{in} = 0.3,$

The initial values of voltage [0.8 V, 0.9 V], with the analysis region being $-1V < V_c < 1V$, $mA\,0.01 < I_L < 0.9mA$. The first set of parameters is chosen to validate the stable oscillatory behavior of the circuits, while the other is to testify when the oscillation dies out.

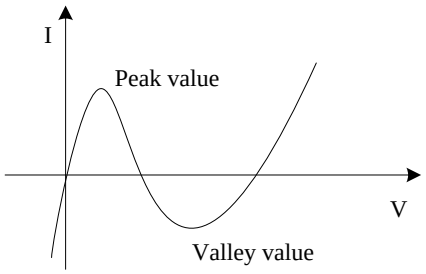

Figure 3. Characteristic Curve for Tunnel Diode Oscillator

2. 2. Algorithm for the Proposed Method

Algorithm SFV

Input: n=1,

T_{n-1}= T_0,

$x[n-1]=j+a$,

$R_{n-1}=x[n-1]$,

$delta=delta_0$,

T_f=Length($G<T_f\Phi$),

G-flag=1

Output: Failure/Success

do while $T_n < T_f$ && *Flag*==0

if verify ($X(n)$, *delta*, T_{n-1})==1

R^n=TM-reach ($X(n)$, T_{n-1}+*delta*, *delta*, T_{n-1})

```
        n++;
        t_n=Step(T_n-1, delta_0);
    else G_flag==0
    endif
     if G_flag==1
         return SUCCESS
    else return FAILURE
    endif
endif.
```

For an AMS SoC, the proposed verification method is based on bounded model checking to demonstrate a correctness proof for the system. Given an AMS SoC and a set of properties, Satisfiability Modular Theory (SMT) based model checking is adopted. Under the set of all input conditions, interval analysis within a certain time steps is utilized to keep the system to go from the given initial states to the final states satisfying the concerned property. If the property of interest is satisfied, then the verification is complete, otherwise counter-example for the non-proved property is provided.

An initial set of states are encoded as intervals at first. Then the possible reachable subsequent states are evaluated. If there exists a path of which the property is evaluated as false, then a concrete counter-example is found. Otherwise, if all paths are true, then the set of current states are transformed to constraints and it is proved if the property holds for all future states. If a proof is obtained, then the property is verified. Otherwise, if the proof fails, then the SMT based step is increased; the next set of interval states is computed and the above operations are re-executed.

By using the above algorithm, the properties are satisfied for the first set of parameters described above, it means that the tunnel diode oscillator is oscillatory under the selected initial conditions, within the given time interval.

By applying the same process for the system with the second set of parameters, but with the same selected initial conditions, the tunnel diode oscillator is found to be non-oscillatory.

3. Counter-Example Generation

The algorithm for counter-example generation is as follows:

Algorithm counter-example generation

```
Input: X[n]={x[n]|n∈N && n<k}
  for (m= |Q|, m>0, m--)
    for (n=0, n<k, n++)
    if judge_cirm(x|n|)==0 && judge_cirm(x|n|)==X
       Q=Q/a;
       break;
      endif
   if a∈Q
      if verify (initial conditions)==1
      return CE=a
    endif
    endif
    if Q≠0
      invoke check_prop(related conditions)
```


endif

Owning to the over-approximation incurred by interval analysis, divergence may lead to false negative. To alleviate this problem, unbounded verification can be obtained using the rules of induction over the recurrence equations. A positive proof by induction ensures that the property of concern is always met; otherwise a counter-example can be generated.

Starting from an initial set of states the validity of the trace is checked. If some bad states are not reachable, then the verification ends. Otherwise, a counter-example is generated. If the counter-example is valid, then verification terminates; otherwise, a refinement process is launched, and the verification is re-executed.

Given the reachable states which belong to the bad ones, the corresponding initial interval states are indentified. Next, it is evaluated whether those initial states would actually lead to bad states or not. If so, then it concludes that the initial condition will lead to an unsatisfied property. A concrete counter-example is found. Otherwise a reachable trace starting from selected initial states is generated.

4. Experimental Results

For the tunnel diode oscillator, it needs 32 separate regions to model the oscillatory and non-oscillatory behaviors. The property is verified when the current I is between 0.45 and 0.55 mA and voltage V_c is between 0.4 and 0.47V. As expected, the property is verified with R = 200 in 13.01 s after finding 16512 state sets, and the property is not verified when R = 242 in 0.57 s after finding 1895 state sets. The verification using the HyTech tool [10] is realized. HyTech can complete analysis with less precision, at the expense of no longer producing oscillating behavior. The proposed approach completes the verification work on the tunnel diode oscillator in 10.39s, and outperforms model checker HyTech which consumes 68.4 s. This indicates that our method can provide drastic performance improvement over two other methods with no loss in verification accuracy.

Table 1 shows the verification results for HyTech [10], LHPN [11] and the proposed method. The same parameters as **[**12] are used for the experiment. The initial current and voltage are $I_l \in$ [0.4, 0.5] mA, $V_c \in$ [0.4,0.5]v.

Table 1. Verification Results Comparisons on Tunnel Diode Oscillator

Approach	HyTech[10]		LHPN[11]		Proposed	
Instance	Time(s)	Verified?	Time(s)	Verified?	Time(s)	Verified?
tunnel diode oscillator (oscillatory)	68. 4	N/A	13.01（R=200Ω）	Yes	10.39	Yes
tunnel diode oscillator (Non-oscillatory)	n.r.	N/A	0.57（R=242Ω）	No	0.30	Yes

The proposed verification method mainly deals with modeling for oscillating behavior and non-oscillating behaviors of tunnel diode oscillator using 32-bit discrete regions. When R=200Ω, LHPN model uses 13.01 s for property verification. 16512 state sets are found in 13.01 s when R=242Ω, 1895 state sets are found in 0.57 s when R=242Ω. We failed to use tool HyTech for verification because of overflow errors. HyTech model checking accomplishes oscillating property checking for tunnel diode oscillator within 68.4 s, however the proposed method only takes 11.57 s, hence obtains higher efficiency than that in [11].

Table 2 shows the verification results comparisons among the proposed method, PHAVer[13] and LEMA[10]. $|\Phi|$ represents the total number of states. The proposed method consumes 13.14 seconds for formal verification of tunnel diode oscillator, while

LEMA uses 18.06 seconds. The proposed method verifies the oscillating behavior and non-oscillating behavior for tunnel diode oscillator successfully. LEMA verifies the oscillating state for tunnel diode oscillator and fails to verify the non-oscillating behavior with less CPU time of 1.9 s. The experimental results show that the proposed approach to formal verification for tunnel diode oscillator is promising.

Table 2. Verification Results Comparisons

Approach	PHAVer[13]			LEMA[10]			Proposed method		
Circuit	|Φ|	Time(s)	Verified?	|Φ|	Time(s)	Verified?	|Φ|	Time(s)	Verified?
tunnel diode oscillator (oscillatory)	17703	72.8	N/A	18524	18.06	Yes	17623	13.14	Yes
tunnel diode oscillator (Non-oscillatory)	1826	n.r.	N/A	1885	1.9	No	1804	0.48	Yes

Table 3 shows the verification results on tunnel diode oscillator under three sets of parameters for the proposed method. It can be seen that for the first two sets of parameters, the property are proved to be true, while for the third parameters, the property are proved to be false, and the corresponding counter-example can be found. The proposed algorithm described above is adopted for counter-example generation.

Table 3. Verification Results under Different Parameters

Parameter	Constraints	Verified?
P_1	$0 \le x \le 0.01$, $-0.01 \le y \le 0$	Yes
	$0 \le x \le 0.02$, $-0.03 \le y \le -0.01$	Yes
P_2	$0 \le x \le 0.01$, $-0.01 \le y \le 0$	Yes
	$0.012 \le x \le 0.013$, $0.01 \le y \le 0.02$	Yes
P_3	$0 \le x \le 0.02$, $-0.03 \le y \le -0.01$	No
	$0.012 \le x \le 0.013$, $0.01 \le y \le 0.02$	No

5. Conclusion

This paper presents a scalable approach to formal verification of dynamic property for AMS designs: SFV. The proposed method is applied on tunnel diode oscillator for formal verification of AMS SoC. An AMS SoC and a set of properties are expressed in VHDL-AMS, then SMT based model checking is adopted to validate if the properties hold in the given model for the design, if the properties are satisfied then the algorithm terminates otherwise the counter-examples are acquired. Experimental results demonstrate the effectiveness of the proposed method. The proof for stability and oscillatory state of tunnel diode oscillator has been conducted respectively under two sets of parameters.

However this paper is just first step for formal verification of AMS designs. It needs a great amount of research work for us to make the proposed method applicable to actual applications. Scalability of the proposed method is one merit; experimental results show that the proposed approach is promising in tackling formal verification of AMS SoC. And it is more advisable to extend the proposed method from hardware design to software one.

Acknowledgments

The author Jingbo Shao would like to thank her supervisor for the support and the help from her university. The research work is supported by Scientific Research Fund of Heilongjiang provincial Education Department (under grant No. 12541237).

References

[1] R. A. Rutenbar, G. G. Gielen and B. A. Antao, "Computer-Aided Design of Analog Integrated Circuits and Systems", IEEE Press, New York, **(2002)**.
[2] T. Kropf, "Introduction to Formal Hardware Verification", Springer, Berlin, **(2000)**.
[3] E. M. Clarke, O. Grumberg and D. A. Peled, "Model Checking", MIT Press, Cambridge, MA, **(2000)**.
[4] W. Hartong, R. Klausen and L. Hedrich, "Formal Verification for Nonlinear Analog Systems", Approaches to Model and Equivalence Checking, Advanced Formal Verification, Kluwer, Dordrecht, **(2004)**, pp. 205-245.
[5] T. R. Dastidar and P. P. Chakrabarti, "A verification system for transient response of analog circuits", ACM Transactions on Design Automation of Electronic Systems, vol. 12, no. 3, **(2007)**, pp. 1-39.
[6] R. Narayanan, B. Akpour, M. H. Zaki, S. Tahar and L. C. Paulson, "Formal verification of analog circuits in the presence of noise and process variation", Design, Automation & Test in Europe Conference & Exhibition, **(2010)**, pp. 1309-1312.
[7] A. Salem, "Semi-formal verification of VHDL-AMS descriptions", IEEE International Symposium on Circuits and Systems, **(2002)**, pp. 333-336.
[8] E. Asarin, T. Dang and O. Maler, "The d/dt tool for verification of hybrid systems, in: Computer Aided Verification", Lecture Notes in Computer Science, vol. 2404, Springer, Berlin, **(2002)**, pp. 365-370.
[9] M. Zaki, .AlSammane, S. Tahar and G. Bois, "Combining symbolic simulation and interval arithmetic for the verification of AMS designs", IEEE International Conference on Formal Methods in Computer-Aided Design, **(2007)**, pp. 207-215.
[10] T. A. Henzinger, P. H. Ho and W. T. Howard, "HYTECH: a model checker for hybrid systems", International Journal on Software Tools for Technology Transfer, vol. 1, no. 1, **(1997)**, pp. 110-122.
[11] K. Lata, S. K. Roy and H. S. Jamadagni, "Towards formal verification of analog mixed signal designs using SPICE circuit simulation traces", 1st Asia Symposium on Quality Electronic Design, CEDT, IISc Bangalore, India Bangalore, **(2009)**, pp. 162-172.
[12] W. Denman, "Towards the Automated Modelling and Formal Verification of Analog Designs", Thesis, Concordia University, **(2009)**, pp. 32-56.
[13] G. Frehse, B. H. Krogh and R. A. Rutenbar, "Verifying analog oscillator circuits using forward/backward refinement", Proc. Design, Automation and Test in Europe. IEEE Computer Society Press, **(2006)**, pp. 257-262.

Author

Jingbo Shao, received the M. E. degree and Ph.D. degrees from Harbin Engineering University, Harbin, China, in 2007, and 2008, respectively. She is currently an associate professor with the Department computer science and information engineering, Harbin Normal University, Harbin, China. Her current research interests include computer aided design, VLSI design automation.