ANALOG ASSERTION-BASED VERIFICATION ON PARTIAL STATE SPACE REPRESENTATIONS USING ASL

Sebastian Steinhorst

TUM CREATE
Singapore
sebastian.steinhorst@tum-create.edu.sg

Lars Hedrich

University of Frankfurt/Main
Germany
hedrich@em.cs.uni-frankfurt.de

ABSTRACT

In this contribution a novel approach to assertion-based analog property verification by considering state space property specification is introduced. In order to apply a formal property specification of complex analog circuit properties to transient simulation waveforms using the Analog Specification Language (ASL), a partial analog state space model is developed to which the simulation waveforms are transferred. In contrast to other approaches operating directly on transient waveforms, on the partial state space model, properties such as periodic behavior, startup times and other complex analog behavior can be systematically specified and automatically verified. Due to the different perspective of state space property specification compared to approaches considering only time signal properties, critical behavior that may be overlooked in time domain signals can be detected in the state space domain. A verification methodology is introduced and a case study on complex properties of a CMOS charge pump shows the feasibility and practicability of the approach for improving the automation of analog verification.

1. INTRODUCTION

Analog circuit verification is a bottleneck in the EDA design flow where automation and formalization is not yet achieved. While regression testing of system implementations with automatic evaluation against a specification using assertion-based verification approaches is common in the digital and software domain of embedded system design, the analog part of such systems is up to now still mainly verified by simple measurements or manually evaluating transient simulation waveforms. While there are some basic measures within common simulator frameworks that facilitate the evaluation of waveform characteristics to a certain degree, a structured verification that takes a formal analog property specification as input and automatically evaluates this specification on the simulation data is not available. For this reason, an approach to apply specifications of analog circuit properties to arbitrary analog transient simulation data by transferring them into a state space representation and applying verification algorithms is presented. Thereby, complex analog circuit properties can be automatically evaluated within a conventional transient simulation framework, reducing the manual interaction and allowing to have a modular automated assertion-based verification approach available for the analog domain. In contrast to evaluating assertions on transient waveforms, the presented approach defines an assertion-based verification methodology around an analog state space model. This new model allows to apply state space property specification approaches from the formal verification domain to industrial-sized circuits which cannot be processed by existing formal verification approaches due to being limited in manageable circuit complexity by state space explosion. By describing analog properties in a state space perspective, a more general and formal property specification and verification will be achieved that exceeds the possibilities of signal level specification.

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

This paper is structured as follows. A discussion on related work is given in Section 2. The structure of the analog state space model (ASSM) is presented in Section 3. The transfer of analog transient simulation waveforms to this ASSM is detailed in Section 4. Section 5 discusses the property specification and verification approach that is applied to the ASSM. The application of the new methodology in a case study is presented in Section 6 and conclusions are drawn in Section 7.

2. RELATED WORK

The first approaches to overcome the manual evaluation of analog simulation results were introduced in the area of automated circuit characterization [1, 2]. Thus, by developing reusable templates for circuit class specific properties, basic automated performance evaluations have been introduced into the analog design flow.

An emerging verification approach attempts to formalize the property specification and evaluation of conventional simulation results introducing assertions. Derived from the digital domain, assertion-based verification automates the evaluation of simulation results and hence shall enable regression testing. While a recent approach to include analog assertion-based verification on commercial platforms proposes property specification implemented either as an analog extension to SystemVerilog Assertions (SVA) or a library of analog assertion objects for the Open Verification Library (OVL) [3], complex analog properties cannot be identified using these assertions.

The tool AMT proposed in [4] uses STL/PSL in order to verify properties on transient simulation waveforms. However, due to the limited expressiveness of STL/PSL, only timing-oriented verification is possible. For higher model abstraction levels, a recent approach applies a Mixed Signal Assertion Language to heterogeneous SystemC-AMS models [5].

In the area of formal verification of analog systems, formal property specification approaches are available that originate from temporal logics [6] which have been modified for application to basic analog system properties [7]. In [8] a verification system applying an extended CTL derivative called AnaCTL to the transient response of analog circuits is proposed. This approach, however, is still limited in specification expressiveness and hence not suitable for complex analog properties. A designer-oriented approach to formal analog property specification that is capable of handling complex circuit properties such as oscillation, slew rate, startup time, etc. was proposed in [9], introducing the analog specification language ASL. Being the up to now most capable formalized specification approach for complex analog system properties, ASL will be adapted as specification approach in this contribution.

3. ANALOG STATE SPACE MODEL

In order to apply assertion-based verification algorithms with a state space property specification to the simulation data, the simulation waveforms have to be transferred into an analog state space model (ASSM). In this section, first the characteristics of the model are introduced formally, and the subsequent section will describe how transient simulation waveforms can be automatically transferred into an ASSM.

In the domain of discrete state system modeling for formal verification approaches, the Kripke structure [10] is a common model combining an automaton and a labeling of the states with atomic propositions for identifying sets of states where a certain proposition is true. In order to transfer analog transient simulation waveforms into a state space model for verification purposes, the Kripke structure has to be extended to an ASSM which incorporates additional information needed for carrying the analog system behavior. The states of the ASSM represent value combinations of the simulation variables of the analog system. Therefore, an extended labeling of the states has to assign this extended variable value vector to each state. As the structure of the ASSM will be determined by transient simulation steps, the transition times between states cannot be considered equal like in transition systems for synchronous clocked digital systems. Hence, the transitions of the ASSM have to be labeled with real valued transition times so that a transition sequence within the ASSM corresponds to the piecewise linear trajectory obtained from the transient simulation waveforms where time steps represent the sampled state variable values which are connected by piecewise linear transitions.

With these considerations, the ASSM can be defined as follows.

Definition 3.1 (Analog State Space Model (ASSM))
For a set of atomic state propositions AP and a set of atomic transition propositions TP, the ASSM M_{ASSM} over AP, TP is a five-tuple $M_{ASSM} = (\Sigma, R, L_A, L_V, T)$ where

- *Σ is a finite set of states of the system.*
- *$R \subseteq \Sigma \times \Sigma$ is a total transition relation, hence for every state $\sigma \in \Sigma$ there exists a state σ' such that $(\sigma, \sigma') \in R$.*
- *$L_A : \Sigma \rightarrow 2^{|AP|}$ is a labeling function that labels each state with the set of atomic propositions that are true in that state.*
- *$L_V : \Sigma \rightarrow \mathbb{R}^{n_d}$ is a labeling function that labels each state with the vector of n_d simulation variables containing the values in this state at this certain time point.*
- *$T : R \rightarrow \mathbb{R}_0^+$ is a labeling function that labels each transition from σ to σ' with a real valued positive or zero transition time that represents the time required for the trajectory in the state space between these states.*

Within the structure M_{ASSM}, a path π beginning at state σ is a sequence of states $\pi = \sigma_0, \sigma_1, \sigma_2, ..., \sigma_n$ with $\sigma_0 = \sigma$ and $(\sigma_i, \sigma_{i+1}) \in R$ for $0 \leq i < n$.

4. TRANSFERRING TRANSIENT SIMULATION WAVEFORMS TO AN ASSM

Transient simulation data consists of data tuples containing a sequence of signal values and their corresponding time points for every investigated node voltage or branch current of the circuit under verification. A transient waveform for such a single signal s_i is a sequence $s_i(t_0), s_i(t_1), ..., s_i(t_n)$.

Definition 4.1 (Path π_{tr} in State Space generated from Transient Simulation Waveforms)
From a set of transient simulation waveforms for different signals, the sequence of states is a path π_{tr} in the ASSM state

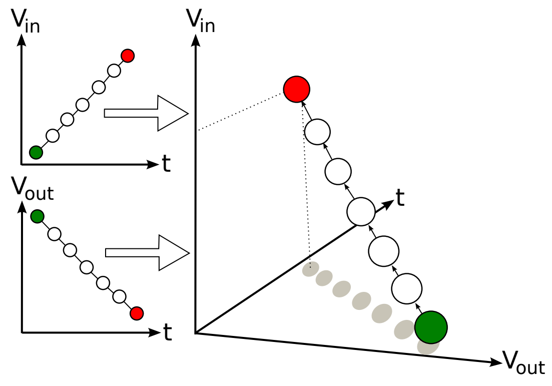


Fig. 1. ASSM graph structure obtained by transient simulation waveforms.

space model determined by the vector of the m signal values $s_i(t_j)$ with $1 \leq i \leq m$ for each time point t_j:

$$\pi_{tr} = \sigma_{t_0}, \sigma_{t_1}, ..., \sigma_{t_n} \tag{1}$$

with

$$L_V(\sigma_{t_j}) = \begin{bmatrix} s_1(t_j) \\ s_2(t_j) \\ \vdots \\ s_m(t_j) \end{bmatrix} \text{ for all } 0 \leq j \leq n \tag{2}$$

The transition relation is connecting consecutive states and the transition times are determined by the time steps of the transient simulation waveforms:

$$T(R(\sigma_{t_j}, \sigma_{t_{j+1}})) = t_{j+1} - t_j \tag{3}$$

The generation of a path π_{tr} in the ASSM is illustrated in Figure 1. Due to the transition times now being defined by the labels of the graph transitions, a time axis is obsolete but plotted for better understanding.

Figures 2(a) and 2(b) show two periodic signals that are combined to a state space representation in Figure 2(c). Due to the periodic behavior of the waveforms, a cycle is generated by mapping both signals to a plot over axis $s_1(t)$ and $s_2(t)$. The detection of periodic behavior is necessary for creating closed state transition cycles that contain important information for the state space representation of transient signals. For modeling state space path cycles, algorithmically, for each vertex σ_{t_r} it is checked whether its coordinate vector $\mathbf{p}^{(t_r)} = L_V(\sigma_{t_r})$ maps to the coordinate vector of another vertex $\mathbf{p}^{(t_s)} = L_V(\sigma_{t_s})$ within a defined tolerance interval ϵ:

$$\sigma_{t_r} \equiv \sigma_{t_s} \Leftrightarrow \forall\, 1 \leq i \leq m : |p_i^{(t_r)} - p_i^{(t_s)}| < \epsilon \tag{4}$$

In this case, the transitions from predecessors and to successors of σ_{t_s} are connected to σ_{t_r}:

$$R = R \cup \{(\sigma_{t_{s-1}}, \sigma_{t_r}), (\sigma_{t_r}, \sigma_{t_{s+1}})\} \tag{5}$$

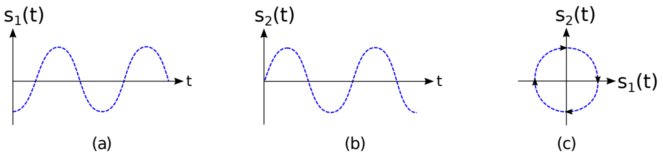


Fig. 2. Periodic transient signal waveforms $s_1(t)$ (a) and $s_2(t)$ (b). State space representation of periodic signals $s_1(t)$ and $s_2(t)$ (c).

Finally, the transitions connecting σ_{t_s} are removed as well as σ_{t_s}:

$$R = R \setminus \{(\sigma_{t_{s-1}}, \sigma_{t_s}), (\sigma_{t_s}, \sigma_{t_{s+1}})\} \quad \wedge \quad \Sigma = \Sigma \setminus \sigma_{t_s} \tag{6}$$

This mapping can be accomplished in $\mathcal{O}(n \log n)$.

5. PROPERTY SPECIFICATION AND VERIFICATION ON AN ASSM

In the previous section, the system representation for verification as an ASSM has been presented. Based on this introduced state space system representation, a property specification and verification approach will be introduced in the following. Starting with a definition of the three elementary concepts of property, performance and specification, an advanced approach for analog property specification for assertion-based verification on an ASSM is systematically developed.

5.1. Basic Definitions

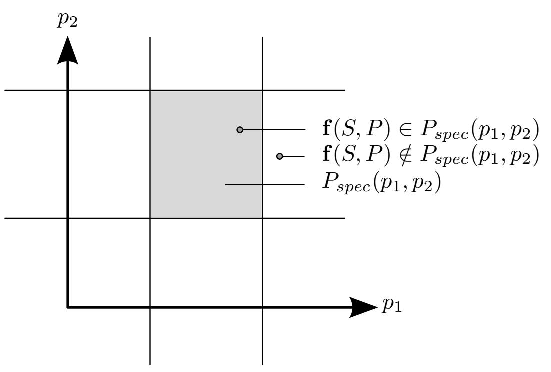
A set of properties can be defined for a system S that are relevant to evaluate the system behavior.

Definition 5.1 (Property)
A system's property can be any function that can be calculated on the system's variables. All properties of a system span the property space P.

Within the property space, the system exhibits a characteristic behavior that constrains the property space to nominal performances that the system can exhibit by system analysis.

Definition 5.2 (Performance)
A system performance $\mathbf{f}(S, P)$ is the result of an evaluation of system properties and hence represents a point in the property space.

Definition 5.3 (Specification)
A specification $P_{spec} \subset P$ defines a subspace of the property space by constraining it to required system performances. Figure 3 illustrates a property space with a system performance satisfying and a performance violating the specification.



p_2
$\mathbf{f}(S,P) \in P_{spec}(p_1,p_2)$
$\mathbf{f}(S,P) \notin P_{spec}(p_1,p_2)$
$P_{spec}(p_1,p_2)$
p_1

Fig. 3. Property space for properties p_1, p_2 with specification $P_{spec}(p_1, p_2)$ and performances satisfying and violating the specification.

5.2. Property Specification Approach

The property specification approach for this contribution is based on a modification of the Analog Specification Language (ASL) [9], which was originally developed for formal model checking of analog circuits based on a discrete model generated from the circuit equations using a state space sampling approach. While the model checking approach retains formal completeness of the verification, it can however not cope with circuits that are larger than simple block level. This is on the one hand due to a state space explosion in modeling, and on the other hand due to the worst case complexity of $\mathcal{O}(n^3)$ of the verification algorithms as some are based on an all pairs shortest path algorithm. In contrast, the ASSM state space modeling approach demonstrated in this contribution can be applied to any system's simulation data at any abstraction level, as the worst case complexity of the verification algorithms is $\mathcal{O}(n)$ (linear) in the number of states n of the ASSM. This is due to the generated state space model being non-branching with periodic cycles being the only exception. These cycles can, however, also be handled in $\mathcal{O}(n)$ by using a marking that is introduced during model generation.

While the approach presented in this contribution does not offer the completeness of formal verification approaches, state space property specification increases expressiveness over conventional signal-based approaches. In contrast to only identifying properties on a time-axis, ASL state space property specifications identify sets of states and transitions in the ASSM and can evaluate operations on their extended parameters including periodic steady states. In the following Section 5.3, the extended Backus Naur form (EBNF) of the language syntax grammar of ASL is presented and some remarks on the semantics are given in Section 5.4 for supporting the explanations and the property specifications discussed in Section 6.

5.3. EBNF Grammar of ASL

ASL_Specification :=
Spec_Sequence **QUIT**; | **QUIT**;

Spec_Sequence :=
Spec_Expression | Spec_Sequence Spec_Expression

Spec_Expression :=
SETVAR *Set_Variable*;
| **NUMVAR** *Number_Variable*; | *Set_Variable* = Set_Expression;
| *Number_Variable* = Number;
| **CALCULATION** *Calc_Name* (Calc_Expression);
| **FOR** Set_Expression **ASSERT** Set_Expression;
| **FOR** Number **ASSERT** Interval;
| *Number_Variable* = **FOR** Set_Expression **ASSERT** Set_Expression;
| *Number_Variable* = **FOR** Number **ASSERT** Interval;

Set_Expression :=
ON Base_Set Operation_Set | Operation_Set

Base_Set :=
Elementary_Set | *Set_Variable* | *State_Space_Variable* Interval
| (Base_Set) | **NOT** Base_Set | Base_Set **AND** Base_Set
| Base_Set **OR** Base_Set

Operation_Set :=
Elementary_Set | *Set_Variable* | *State_Space_Variable* Interval
| (Operation_Set) | **NOT** Operation_Set
| Operation_Set **AND** Operation_Set
| Operation_Set **OR** Operation_Set
| *Calc_Name* (calc_parameters) Interval
| **VALUE** (*State_Space_Variable*) Interval
| **ASSIGN** (*Number_Variable*, Assign_Type) Operation_Set
| **SELECT** Operation_Set
| **CYCLE** | Temporal_Logic_Expression Operation_Set
| **TRANSITION FROM** Operation_Set **TO** Operation_Set

Elementary_Set :=
ALL | **DCPOINTS**

Interval :=
[Number, Number] | [< Number] | [<= Number] | [> Number] | [>= Number] | ε

Temporal_Logic_Expression :=
Temporal_Logic_Operator Interval Direction Operation_Set
| **ALWAYS** Direction Operation_Set **UNTIL** Interval Operation_Set
| **A** Direction Operation_Set **U** Interval Operation_Set
| **EXISTS** Direction Operation_Set **UNTIL** Interval Operation_Set
| **E** Direction Operation_Set **U** Interval Operation_Set

Temporal_Logic_Operator :=
UNIVERSALLY | **AG** | **EVENTUALLY** | **AF**
| **STAY** | **EG** | **REACH** | **EF**

Direction :=
ε | **FROM** | **^ -1**

Calc_Expression :=
Number | calc_parameter N | (Calc_Expression)
| Calc_Expression Math_Operator Calc_Expression

Number :=
Floatingpoint_Constant | *Number_Variable*
| (Number) | Number Math_Operator Number | **ABS** (Number)

Assign_Type :=
MAX | **MIN** | **AVERAGE** | **RANGE**

Math_Operator :=
+ | **-** | ***** | **/**

5.4. ASL semantics for verification on ASSM

In the following, the relevant properties of ASL semantics for verification on the ASSM are discussed.

A set of states ϕ can be selected on the whole ASSM or on another set by a signal variable constrained to a specified interval. Algorithmically, for each state σ_i is decided whether it is included within the given interval by a comparison of the actual value $p_i^{(s)}$ of the entry in position s in the signal variable value vector $\mathbf{p}_i = L_V(\sigma_i)$ and the interval boundaries $\underline{r}$ and $\overline{r}$:

$$\sigma_i \in \phi \Leftrightarrow p_i^{(s)} \in [\underline{r}, \overline{r}] \qquad (7)$$

With the operation `assign`, the minimum and maximum of a signal value s_i in a set ϕ of states can be assigned to numeric variables. A set of reachable states ϕ from a state σ_0 in a set of starting states can be identified on a path π with the operation `reach`:

$$M_{ASSM}, \sigma_0 \models \text{ reach } \phi \Leftrightarrow \ \exists \pi = \sigma_0, ..., \sigma_i : \sigma_i \in \phi \qquad (8)$$

The transition time t_p of a connecting transition path π between two sets ϕ and ψ of states can be measured with the operation `transition`:

$$t_p = \sum T(R(\sigma_i, \sigma_j)) \text{ for all } (\sigma_i, \sigma_j) \ \in \pi \qquad (9)$$

Using the operation `assertion`, either the value of a numeric variable in a given interval or the inclusion of states in a given set of states can be evaluated. The results of all operations are printed to a verification report.

The ASSM-based verification flow for use with transient simulation waveforms transferred to a state space representation is illustrated in Figure 4.

6. CASE STUDY AND RESULTS

In this section a step-by-step case study to demonstrate the verification methodology on a CMOS charge pump circuit is performed. The verification algorithms and the ASL parser have been prototypically implemented in C++ and the Lex/Yacc parser generator.

The charge pump circuit shown in Figure 5 is a clocked step-up voltage converter allowing output voltages nearly twice the supply voltage. The parameters are $V_{DD} = 3$ V, $clk = 50$ kHz, $R_{load} = 1$ MΩ, $C_1 = C_{load} = 1$ nF.

In the following, an ASL specification for verifying the startup time is developed and applied to the ASSM that was generated from the transient simulation waveforms for $V_{C_{load}}$, V_{C_1}, and V_{clk}. In contrast to signal-based specification approaches, the state space specification methodology takes a different perspective. In the state space, properties of a circuit are determined by transitions between state space areas and their state space variable values. For a state space specification of a startup time property, we will determine the transition time from a start area to the the area where the output

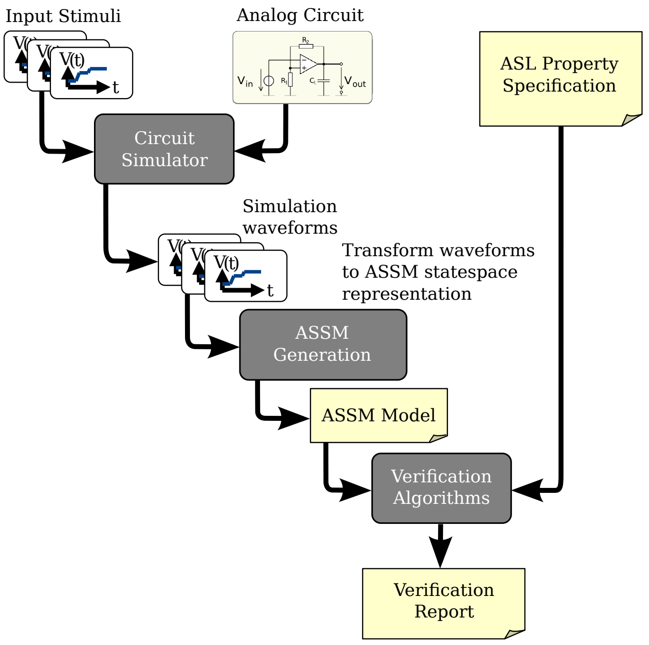


Fig. 4. ASL assertion-based verification flow for transient simulation results transferred to an ASSM state space representation.

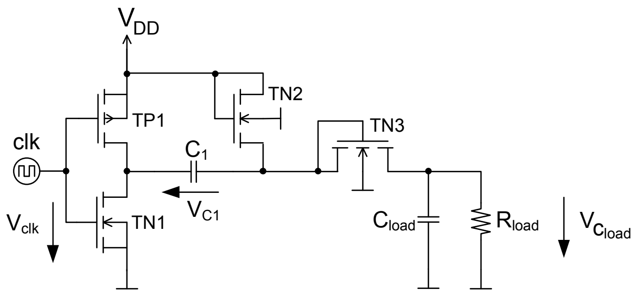


Fig. 5. Circuit schematic of the CMOS charge pump.

voltage is in the periodic steady state. The time to reach this state will be verified, as well as that it is within a specified range. The ripple of the periodic steady state is also verified not to exceed a specified range. An illustration of the subsequent specification steps in the state space representation is shown in Figure 6.

The start area for the startup time verification has to be selected:

```
startarea = (value(V_C_load)[<0.1])
  and (value(V_C_1)[>-0.1]);
```

The states reachable from this start area contain the dynamic startup behavior of the circuit on which the maximum output voltage can be measured and assigned to the number variable `%max_V_C_load`:

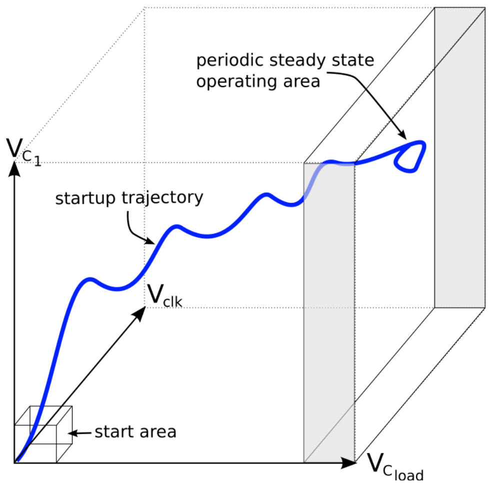


Fig. 6. State space representation of the chargepump startup behavior.

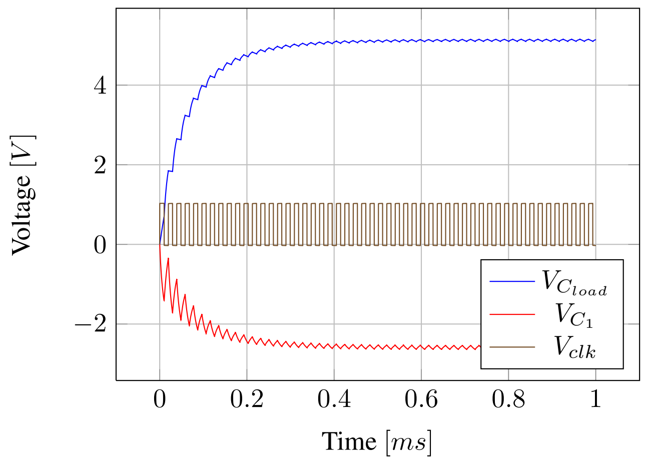


Fig. 7. Waveforms for $V_{C_{load}}$, V_{C_1} and V_{clk} in conventional representation over time axis.

```
reachable = reach from startarea;

on reachable assign(%max_V_C_load,max) value(V_C_load);
```

The value assigned to `%max_V_C_load` is 5.15885 V. Within the states reachable from the start area, the maximum startup time is measured using the transition operation, assigning the computed states of the startup path to the set `startup` where $V_{C_{load}}$ is greater than 90% of the maximum output voltage:

```
startup = assign(%startup_time,max) transition
  startarea to value(V_C_load)[>= 0.9 * %max_V_C_load];
```

A value of 169 μs is assigned to `%startup_time`. The periodic steady states where the circuit reaches its maximum output voltage over $V_{C_{load}}$ with a periodic signal ripple shall be analyzed in addition. Therefore, these periodic steady states, which form a cycle of transitions in the state space, are identified and the minimum and maximum values of $V_{C_{load}}$ are measured by the following ASL statements:

```
periodic_output_set = on reachable select cycle;

on periodic_output_set assign(%min_V_C_load_ripple,min)
  assign(%max_V_C_load_ripple,max) value(V_C_load);
```

The periodic output ripple is reported by the automatically generated verification report to be in the interval [5.1008 V, 5.1503 V]. In a final step, assertions for the required output voltage, the startup time and the voltage range of the periodic output ripple are verified:

```
for %max_voltage assert [>= %spec_parameter1];
for %startup_time assert [<= %spec_parameter2];
for %min_V_C_load_ripple assert [>= %spec_parameter3];
for %max_V_C_load_ripple assert [<= %spec_parameter4];
```

The transient simulation of the waveforms for ASSM generation contained 8424 time steps for a simulation time of 1 millisecond. The conventional waveform plot is shown in Figure 7. The transfer to an ASSM model consisting of 3528 vertices and 4539 transitions took 0.35 seconds on a Core 2 Quad with 2.8 GHz clock frequency and 8 GB of RAM. The lower number of states compared to the simulation time steps is due to the ripple behavior at the maximum output voltage mapping to a periodic cycle of states. Figure 8 illustrates a visualization of the transition steps in state space of the ASSM model over $V_{C_{load}}$, V_{C_1} and V_{clk}, generated from the transient simulation data. Note that there is no time axis for the ASSM visualization, as time is represented in the transition steps between the states.

7. CONCLUSIONS

In this contribution, a new approach to automated verification of analog system properties was presented by introducing a state space representation of analog transient simulation waveforms. The formalization of property specification by using the dedicated analog specification language ASL with its state space specification approach and verification algorithms delivers a completely automated assertion-based verification methodology for complex analog circuit behavior. With industrial embedded system design flows demanding more automation of the analog design part, this approach can contribute to a higher design efficiency by offering the desired decoupling of verification from design in the scope of future assertion-based analog verification. Moreover, the state space property specification perspective can help to improve verification coverage as it abstracts from purely timed behavior and therefore requires a more abstract and global view of the properties of the circuit under verification.

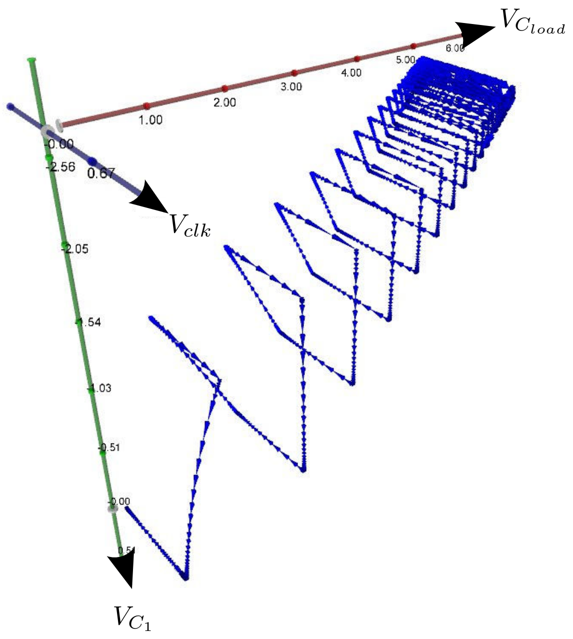


Fig. 8. Transient simulation waveforms of the charge pump for $V_{C_{load}}$, V_{C_1} and V_{clk} transferred into a ASSM representation for application of ASL verification algorithms.

8. REFERENCES

[1] S.A. Huss, M. Gerbershagen, and G. Traenkle. Automatic performance characterization of analog functional blocks. *Analog Integrated Circuits and Signal Processing*, 1(4):277–286, 1991.

[2] J. Eckmueller, M. Gropl, and H. Graeb. Hierarchical characterization of analog integrated cmos circuits. In *Proceedings of the conference on Design, automation and test in Europe*, pages 636–643, 1998.

[3] R. Mukhopadhyay, S. K. Panda, P. Dasgupta, and J. Gough. Instrumenting ams assertion verification on commercial platforms. *ACM Trans. Des. Autom. Electron. Syst.*, 14(2):1–47, 2009.

[4] D. Nickovic and O. Maler. Amt: A property-based monitoring tool for analog systems. In J.-F. Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 2007.

[5] S. Lämmermann, J. Ruf, T. Kropf, W. Rosenstiel, A. Viehl, A. Jesser, and L. Hedrich. Towards assertion-based verification of heterogeneous system designs. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1171–1176. European Design and Automation Association, 2010.

[6] E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, London, 1982.

[7] W. Hartong, L. Hedrich, and E. Barke. Model checking algorithms for analog verification. In *Proceedings of the 39th conference on Design automation (DAC '02)*, pages 542–547, 2002.

[8] T. R. Dastidar and P. P. Chakrabarti. A verification system for transient response of analog circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 12(3):1–39, 2007.

[9] S. Steinhorst and L. Hedrich. Model Checking of Analog Systems using an Analog Specification Language. In *Proc. of the Conference on Design, Automation and Test in Europe 2008 (DATE'08)*, pages 3247–329, 2008.

[10] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model checking*. Springer, 1999.