Automated Mixed-Signal Verification using Monitors and Simulation Relations

Student Name : Debjit Pal
Roll Number  : 09CS7002
Supervisors  : Prof. Pallab Dasgupta
               (Computer Science and Engg. Dept.)
               Prof. Siddhartha Mukhopadhyay
               (Electrical Engg. Dept.)

Defense Seminar
Computer Science and Engineering Department
Indian Institute of Technology Kharagpur

7th January, 2013
Outline I

1. Introduction
2. Motivation
3. AMS Verification Library
4. Verification of Simulation Relations
5. Feature based Online Conformance Checking among AMS Circuits
Outline

1. Introduction

2. Motivation

3. AMS Verification Library

4. Verification of Simulation Relations

5. Feature based Online Conformance Checking among AMS Circuits
What is Analog-Mixed Signal (AMS) Verification?

Functional Diagram of BUCK Regulator

Different Types of AMS Verification

1. Automatic Monitoring of Behaviors.

2. Formal Conformance between Digital Controllers.


Functional Diagram of BUCK Regulator

Outline

1. Introduction

2. Motivation

3. AMS Verification Library

4. Verification of Simulation Relations

5. Feature based Online Conformance Checking among AMS Circuits
Automatic Monitoring of Behaviors

Automatic Monitoring Method

Property Languages

Digital Domain
1. SVA
2. PSL

AMS Domain
1. STL\(^1\)
2. AMS-SVA\(^2\)
3. AMS-LTL\(^3\)

Library based Approach

Digital Domain
1. OVL

AMS Domain
1. AMS-OVL\(^2\)

References

3. S. Mukherjee, P. Dasgupta et.al., Synchronizing AMS Assertions with AMS Simulation: From Theory to Practice, ACM Transaction on Design Automation and Electronic Systems, Accepted for Publication, 2012
Conformance of Digital Controllers

Digital Domain Conformance Checking Algorithm

- Kanellakis-Smolka’s Algorithm (KS).
- Paige-Tarjan Algorithm (PT).

Predicate Over Real Variables (PORVs)
A n-tuple $\langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle \in \mathbb{R}^n$ and a relational operator $\sim \in \{>, \geq\}$ over $V = \{x_1, x_2, \ldots, x_n\}$, i.e. $(\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n + c) \sim 0$ where $c$ is any constant and $c \in \mathbb{R}$.

- Digital controller (DC) an important component of Hybrid Systems
- DC has PORVs as inputs apart from propositions
Automatic Monitoring can be leveraged for Conformance checking of two AMS models / circuits.

Conformance Checking Scheme

Test Benches

Model M1

Model M2

Monitoring Network
AMS-VL + Auxiliary Func.

Report
Outline

1. Introduction
2. Motivation
3. AMS Verification Library
4. Verification of Simulation Relations
5. Feature based Online Conformance Checking among AMS Circuits
Simulation Based Verification Framework for Complex Properties

Figure: Circuit Simulation Environment with Complex Properties
## A Glimpse of the Modules of the AMS-VL

<table>
<thead>
<tr>
<th>Type of Module</th>
<th>Name of Module</th>
<th>Purpose of Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latch Modules</td>
<td>CaptureAndHold</td>
<td>captures and holds any input digital signal forever until simulation is over.</td>
</tr>
<tr>
<td></td>
<td>GenerateDelay</td>
<td>holds any input digital signal for a specified delay time.</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>ArithmeticOperator</td>
<td>generates sum or difference of voltage of analog input signals of the modules.</td>
</tr>
<tr>
<td>and</td>
<td>EventDetector</td>
<td>monitors analog cross events on an analog input signal with respect to a specified threshold parameter.</td>
</tr>
<tr>
<td>Boolean</td>
<td>EventDetector_Extended</td>
<td>monitors analog events based on the relative values of their input analog signals.</td>
</tr>
<tr>
<td>Operation</td>
<td>PredicateEvaluator</td>
<td>compares any analog signal with user specified threshold value.</td>
</tr>
<tr>
<td>Modules</td>
<td>PredicateEvaluator_Extended</td>
<td>compares two analog signals based on their relative values.</td>
</tr>
<tr>
<td></td>
<td>BoolOperator</td>
<td>performs standard Boolean operations.</td>
</tr>
<tr>
<td>Interval</td>
<td>GlobalOperator</td>
<td>checks the truth of an expression over a specified period of time.</td>
</tr>
<tr>
<td>Operation</td>
<td>EventuallyOperator</td>
<td>checks whether an expression ever becomes true within a specified time frame.</td>
</tr>
<tr>
<td>Modules</td>
<td>UntilOperator</td>
<td>checks whether an expression remains true over a time window until another event occurs</td>
</tr>
<tr>
<td></td>
<td>PredicateAssert</td>
<td>checks the truth of an expression when a particular condition is satisfied over a specified period of time.</td>
</tr>
</tbody>
</table>

**Table:** Broad Classification of AMS-VL Modules
Implementation Issue - 1 (Synchronization with Simulator)

Property

After $V_{in}$ exceeds 3.0V, $V_{out}$ should exceed 2.0V eventually within 2ms and 3ms.

We use time tolerance and value tolerance parameter of cross_event of Verilog-AMS to handle the issue.
Implementation Issue - II (Parallel Threading)

**Property**

After \( V_{in} \) crosses \( V_{th} \), \( V_{out} \) should cross \( V'_{th} \) sometime between 15\( \mu \)s and 25\( \mu \)s.

We use fork/join construct of Verilog-AMS to handle the issue.
Implementation Issue - II (Parallel Threading)

Input Signal

Analog Events

\[ V_{th} \]

\[ 10\mu s \quad 20\mu s \quad 30\mu s \quad 40\mu s \]

/\join construct of Verilog-AMS to handle time
### Simulation Results

<table>
<thead>
<tr>
<th>Cross Event Precisions</th>
<th>Number of Cross Events (approx)</th>
<th>Number Of Properties Verified</th>
<th>Sim Time (sec)</th>
<th>CPU Time (secs) Design</th>
<th>CPU Time (secs) Design + Aux Func</th>
<th>CPU Time (secs) Design + Aux Fun + Prop Checkers</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (sec)</td>
<td>value (V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-9</td>
<td>1e-6</td>
<td>248</td>
<td>28</td>
<td>700μs</td>
<td>83.8912</td>
<td>84.5162</td>
<td>97.1208</td>
</tr>
<tr>
<td>1e-6</td>
<td>1e-4</td>
<td>248</td>
<td>28</td>
<td>700μs</td>
<td>83.8912</td>
<td>84.5162</td>
<td>96.0134</td>
</tr>
<tr>
<td>1e-4</td>
<td>1e-3</td>
<td>248</td>
<td>28</td>
<td>700μs</td>
<td>83.8912</td>
<td>83.5162</td>
<td>95.4262</td>
</tr>
</tbody>
</table>

**Test Case I: LDO Circuit (Containing 6 LDOs)**

<table>
<thead>
<tr>
<th>Cross Event Precisions</th>
<th>Number of Cross Events (approx)</th>
<th>Number Of Properties Verified</th>
<th>Sim Time (sec)</th>
<th>CPU Time (secs) Design</th>
<th>CPU Time (secs) Design + Aux Func</th>
<th>CPU Time (secs) Design + Aux Fun + Prop Checkers</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (sec)</td>
<td>value (V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-9</td>
<td>1e-9</td>
<td>247</td>
<td>33</td>
<td>600μs</td>
<td>75.4402 K</td>
<td>75.8206 K</td>
<td>81.2415 K</td>
</tr>
<tr>
<td>1e-6</td>
<td>1e-4</td>
<td>247</td>
<td>33</td>
<td>600μs</td>
<td>75.4402 K</td>
<td>75.8206 K</td>
<td>81.2415 K</td>
</tr>
<tr>
<td>1e-4</td>
<td>1e-3</td>
<td>247</td>
<td>33</td>
<td>600μs</td>
<td>75.4402 K</td>
<td>75.8206 K</td>
<td>80.8945 K</td>
</tr>
</tbody>
</table>

**Test Circuit II: Buck Regulator (Containing 2 Buck Regulators)**

<table>
<thead>
<tr>
<th>Cross Event Precisions</th>
<th>Number of Cross Events (approx)</th>
<th>Number Of Properties Verified</th>
<th>Sim Time (sec)</th>
<th>CPU Time (secs) Design</th>
<th>CPU Time (secs) Design + Aux Func</th>
<th>CPU Time (secs) Design + Aux Fun + Prop Checkers</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (sec)</td>
<td>value (V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-9</td>
<td>1e-6</td>
<td>100</td>
<td>10</td>
<td>500μs</td>
<td>90.2509 K</td>
<td>92.3104 K</td>
<td>98.0576 K</td>
</tr>
<tr>
<td>1e-6</td>
<td>1e-4</td>
<td>100</td>
<td>10</td>
<td>500μs</td>
<td>90.2509 K</td>
<td>92.3104 K</td>
<td>97.7868 K</td>
</tr>
<tr>
<td>1e-4</td>
<td>1e-3</td>
<td>100</td>
<td>10</td>
<td>500μs</td>
<td>90.2509 K</td>
<td>92.3104 K</td>
<td>97.6695 K</td>
</tr>
</tbody>
</table>

**Test Case III: Integrated Circuit Netlist (Containing 4 LDOs and 1 Buck Regulator)**

<table>
<thead>
<tr>
<th>Cross Event Precisions</th>
<th>Number of Cross Events (approx)</th>
<th>Number Of Properties Verified</th>
<th>Sim Time (sec)</th>
<th>CPU Time (secs) Design</th>
<th>CPU Time (secs) Design + Aux Func</th>
<th>CPU Time (secs) Design + Aux Fun + Prop Checkers</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (sec)</td>
<td>value (V)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** CPU Time for Simulations of Circuits
Outline

1 Introduction

2 Motivation

3 AMS Verification Library

4 Verification of Simulation Relations

5 Feature based Online Conformance Checking among AMS Circuits
Controllers of Hybrid Systems

- *Labels can be Predicates over Real Variables (PORVs).*
- *Use of embedded control in hybrid system is omnipresent now a days.*

**Figure:** Typical Hybrid System Environment

**Specification of a Steam-Boiler Water Level Controller**

1. **P1**: *If the water level is below 10 units, then the controller should switch the pump to ON state.*

2. **P2**: *If the water level is above 85 units, then the controller should switch the pump to OFF state.*
Example of a Digital Controller (Water Level Controller)

(a) Specification

(b) Implementation 1

(c) Implementation 2

Figure: A Specification and Two Implementations
Why do we need Auxiliary Specification for Controller Automaton?

- To handle non-ideal environment conditions.

For specification controller we define the following term: \( \gamma : \mathcal{H} \rightarrow \{ W, S \} \)

where \( \mathcal{H} \) is the guard condition for state transition \( s_k \) to \( s_j \) i.e. \( s_k \xrightarrow[\mathcal{H}]{} s_j \), \( \mathcal{H} = \bigwedge \alpha_l \), where \( \alpha_l \in I^s \).

\[
\begin{align*}
\gamma : (w \geq 10) & \rightarrow S \\
\gamma : (w < 10) & \rightarrow W \\
\gamma : (w \leq 85) & \rightarrow S \\
\gamma : (w > 85) & \rightarrow W
\end{align*}
\]
A Few Definitions

**Sim(π^i, σ)**

A trace \(σ = σ_0, σ_1, σ_2, \ldots\) models a path \(π^i = q^i_0, e^i_0, q^i_1, e^i_1, \ldots\) of \(G^i\) if the following conditions hold good:

1. for all \(k\), the set of atomic propositions that are true in \(σ_k\) exactly match with the set of atomic propositions that are true in state \(q^i_k\).
2. for all \(k\), the valuation of real valued variables in each \(σ_k\), make the transition guard condition \(α\) of \(e^i_k\) true where \(q^i_k \xrightarrow{α} q^i_{k+1}\).

**Paths(\(G^i\))**: set of all paths of implementation \(G^i\)

**Traces(\(G^i\))**: set of all traces such that Sim(\(π^i, σ\)) is true.

**Paths(\(G^s\))**: set of all paths of specification \(G^s\).

**Sim(π^s, σ, γ)**

A trace \(σ = σ_0, σ_1, σ_2, \ldots\) models a path \(π^s = q^s_0, e^s_0, q^s_1, e^s_1, \ldots\) of \(G^s\) under a given refinement directives \(γ\) if the following conditions hold good:

1. for all \(k\), the set of atomic propositions that are true in \(σ_k\) exactly match with the set of atomic propositions that are true in state \(q^s_k\).
2. for all \(k\), the valuation of real valued variables in each \(σ_k\) is such that they satisfy the weakening or strengthening refinement directive on the transition guard condition \(β\) of \(e^s_k\) where \(q^s_k \xrightarrow{β} q^s_{k+1}\).
A Theorem

\[ G^i \preceq G^s \]

\( G^i \preceq G^s \) with respect to a given refinement directives \( \gamma \) iff for all traces \( \sigma \in \text{Traces}(G^i) \), \( \text{Sim}(\pi^i, \sigma) \) true implies there exists a path \( \pi^s \in \text{Paths}(G^s) \) such that \( \text{Sim}(\pi^s, \sigma, \gamma) \) is true. Precisely, \( \forall \sigma \in \text{Traces}(G^i), \text{Sim}(\pi^i, \sigma) \Rightarrow \exists \pi^s \in \text{Paths}(G^s), \text{Sim}(\pi^s, \sigma, \gamma) \) is true.

\[ \text{Sim}^A(\pi^i, \pi^s, \gamma) \]

A path \( \pi^s \in \text{Paths}(G^s) \) simulates a path \( \pi^i \in \text{Paths}(G^i) \) if the following conditions hold good:

1. for all \( k \), the set of atomic propositions that are true in \( q^i_k \) and the set of atomic propositions that are true in \( q^s_k \) exactly match i.e. \( L^i(q^i_k) = L^s(q^s_k) \).

2. for all \( k \), \( q^i_k \stackrel{\alpha}{\rightarrow} q^i_{k+1} \) and \( q^s_k \stackrel{\beta}{\rightarrow} q^s_{k+1} \), the refinement directives \( \gamma \) holds good between the guard conditions \( \alpha \) and \( \beta \).

\[ G^i \preceq^A G^s \]

\( G^i \preceq^A G^s \) iff \( \forall \pi^i \in \text{Paths}(G^i), \exists \pi^s \in \text{Paths}(G^s) \) such that \( \text{Sim}^A(\pi^i, \pi^s, \gamma) \) holds good.

Theorem (Simulation Relation)

\[ G^i \preceq G^s \iff G^i \preceq^A G^s. \]
Pre-Process: Step 1

Finding Valid State Pairs w.r.t Atomic Propositions

Transition Systems

(c) Specification

\[ P/\text{On} \xrightarrow{A:w<10} Q/\text{Off} \]

(d) Implementation 1

\[ R/\text{On} \xrightarrow{B:w<15} S/\text{Off} \]

(e) Implementation 2

\[ T/\text{On} \xrightarrow{C:w<5} U/\text{Off} \]

State Propositions

- Atomic Propositions: On and Off

Valid State Pairs

- \( P \& R \) is a valid state pair.
- \( P \& T \) is a valid state pair.
Pre-Process: Step 2

Finding Valid State Pairs w.r.t PORVs

Transition Systems

- **P/On** $\xrightarrow{A : w < 10} Q/Off$
- **R/On** $\xrightarrow{B : w < 15} S/Off$
- **T/On** $\xrightarrow{C : w < 5} U/Off$

(f) Specification

(g) Implementation 1

(h) Implementation 2

Minimal Unsat Cores

- **Involving** $A$ and $B : A \& \neg B$
- **Involving** $A$ and $C : \neg A \& C$

Valid State Pairs

- **P** & **R** is a valid state pair.
- **P** & **T** is not a valid state pair.
Example of Kanellakis-Smolka’s Algorithm: Initial Stage

Figure: Example of Kanellakis-Smolka’s Algorithm: Iteration 0

(i) State Machine 1

(j) State Machine 2

This example has been reproduced from *The Algorithmics of Bisimilarity* by L. Aceto, A. Ingolfsdottir and J. Srba published by Cambridge University Press
Example of Kanellakis-Smolka’s Algorithm : Final Stage

(a) State Machine 1

(b) State Machine 2

Figure: Example of Kanellakis-Smolka’s Algorithm : Iteration 1

---

1 This example has been reproduced from *The Algorithmics of Bisimilarity* by L. Aceto, A. Ingolfsdottir and J. Srba published by Cambridge University Press.
Outline

1. Introduction
2. Motivation
3. AMS Verification Library
4. Verification of Simulation Relations
5. Feature based Online Conformance Checking among AMS Circuits
In the PWM mode of operation, starting from beginning up to any time point, the maximum difference between the frequency of oscillation at the switching pin of specification and implementation should not exceed 500 Hz.

Figure: Monitoring Network for Example 1
Schematic of Example in Cadence Virtuoso

Figure: Cadence Schematic of Example 1
Tool Flow

Figure: Simulation Relation Finding Tool Flow
Conclusion

AMS Verification Library
- Extension of Digital OVL Library to AMS Domain.
- Re-entrant library modules with multi-match semantics.

Verification of Simulation Relation
- Concept of Refinement Directives.
- Definition of Sim($\pi^i, \sigma$), Sim($\pi^s, \sigma, \gamma$), $G^i \preceq G^s$, Sim$^A$($\pi^i, \pi^s, \gamma$), $G^i \preceq A G^s$.
- Proposed and proved $G^i \preceq G^s$ iff $G^i \preceq A G^s$.

Conformance Checking with AMSVL
- Concept of Feature based conformance checking of AMS circuits.
- Several topologies proposed for conformance checking.
Future Works

AMS Verification Library

- To investigate if more modules needed in AMSVL to capture richer set of properties.
- Porting AMSVL to Matlab to avail off richer set of auxiliary functions.

Verification of Simulation Relation

- Equivalence checking in terms of containment of Reach Sets of continuous variables of the plant dynamics.
- Extension of assume properties from digital domain verification to model more realistic constrained environment. Constraints can be specified as Unsafe set in state space.

Conformance Checking with AMSVL

- Synchronization issues between two AMS circuits during simulation needs further investigation.
There are more work done in analog formal verification. Need a more comprehensive description of the past work.

- More previous and recent works have been discussed in Chapter 2, pages 18-20.

One of the highlighted contributions is multi-match semantics. Do note that the digital OVL library have re-entrant modules with multi-match semantics. How is your work different than what already exists?

- From documentation, detail of multi-match semantics of OVL is not clear. We implemented a multi-match semantics which is suitable for verification of properties defined over dense real time.

Can you explain why you say OVL Library enabled graphical composition?

- Each AMSVL module has a graphical symbol in CDBA format along with the Verilog-AMS code. These symbols can be connected in a schematic editor like Cadence Virtuoso to build property verification modules. Hence, AMSVL modules enable graphical composition.
Any thoughts on how to extend the technique to support the Assume / Guarantee reasoning?

- Possibly we can convert the simulation relation finding problem to a standard model checking problem and can apply the Assume / Guarantee methods developed so far in literature. For that possibly we need to construct $G = G^i || G^s$, an LTS $A$ for the assumptions modeling the environment and $G || A$. There exists considerable amount of work in literature which uses AG for checking simulation conformance with propositional labeling. They need to be augmented with PORV labeling. Some of the works are as follows:

Can you cite an example to show that “AMS domain are not expressive enough?” Page 25 line 4-5

- We meant that the extension of assertion languages in AMS domain are not expressive enough to capture many interesting analog signal behaviors. We have shown a property in Example 2.7 (Page 16), where no existing assertion language can capture the exponential behavior of the analog signal.

“schematic like transistors”-why transistors? It is rather a block level schematic entry. Page 25 line 16.

- This sentence has been merged with the next sentence to correct the meaning. Please see Page 27, line 16.

Why do you believe that verification library has some definite advantages? Quantify the advantages. Page 25 line 20-22

- Three distinct advantages have been cited in the Page 27 (please see lines 19-26) followed by a behavioral response of a circuit which can be captured by the AMS Verification Library but cannot be captured by assertion languages.
In Section 3.2.1, it will be better if you can include some examples corresponding to the syntactical constructs after each of them. Also, “Notes” are not desirable in a thesis. It is better to put it as a continuous text.

- Suitable examples of syntactical constructs added and content of “Notes” added in main text. Please see pages 31-49.

Specify the complexities of “Algorithm 1”, “Algorithm 2”, “Algorithm 3” and “Algorithm 4”.

- These algorithms are essentially the high level working principle of the corresponding modules. Hence, the complexity of these Algorithms are not relevant in the context.

No explanation has been given for Fig 3.5, 3.6 and 3.7.

- Explanations have been added along with the respective Figures. Please see pages 43, 44 and 46.

In Fig 3.12 many of the arrows are not labeled.

- Corrected. Please see Page 54.
There is no qualitative / quantitative comparison with the contending strategies reported in the literature. Addition of such comparison will add value to the work.

- We introduced AMS verification library with re-entrant modules for the first time. To the best of our knowledge, we made first attempt to extend KS Algorithm to the domain equivalence checking of PORV labeled transition system. Hence in either of these two cases, we cannot provide a comparison right now.

Some typographical/grammatical errors in the thesis.

- Corrected as suggested.
Thank You...