10th January 2017

VLSID 2017

Generating AMS Behavioral Models with Formal Guarantees on Feature Accuracy

António Anastásio Bruto da Costa
Pallab Dasgupta

Formal Methods Laboratory,
Dept. of Computer Sci. & Engg

AMS Behavioral Modeling is highly significant today

Speed of Digital-Analog simulation is dominated by speed of analog simulation

- **Analog simulation remains way too expensive**
- **AMS Behavioral models are therefore widely used to accelerate simulation**

However, a significant fraction of the bugs found in chips today are attributed to incorrect behavior at the digital-analog interface

- **Which means that we are not doing the modeling correctly**
- **… Or not accurate enough for the right behaviors**

How should we build AMS behavioral models?

What are the primary concerns in behavioral modeling?

Abstraction versus accuracy

Accuracy

- Often accuracy requirements are demanded without specific reference to the end use of the model
- A practical strategy is to make the models as light as possible, while preserving the accuracy of those behaviors that are relevant for the end use of the model
- But how?

CHALLENGE

- Designing behavioral models that are accurate with respect to features of interest, where …
  - Features are functional properties of the component being modeled

Proving Feature Accuracy of AMS BMs

Proving feature accuracy is not easy

- AMS models are developed in various languages, most of which are not amenable for formal analysis
- AMS models are developed in various levels of abstraction. For low levels of abstraction adequate simulation is infeasible.

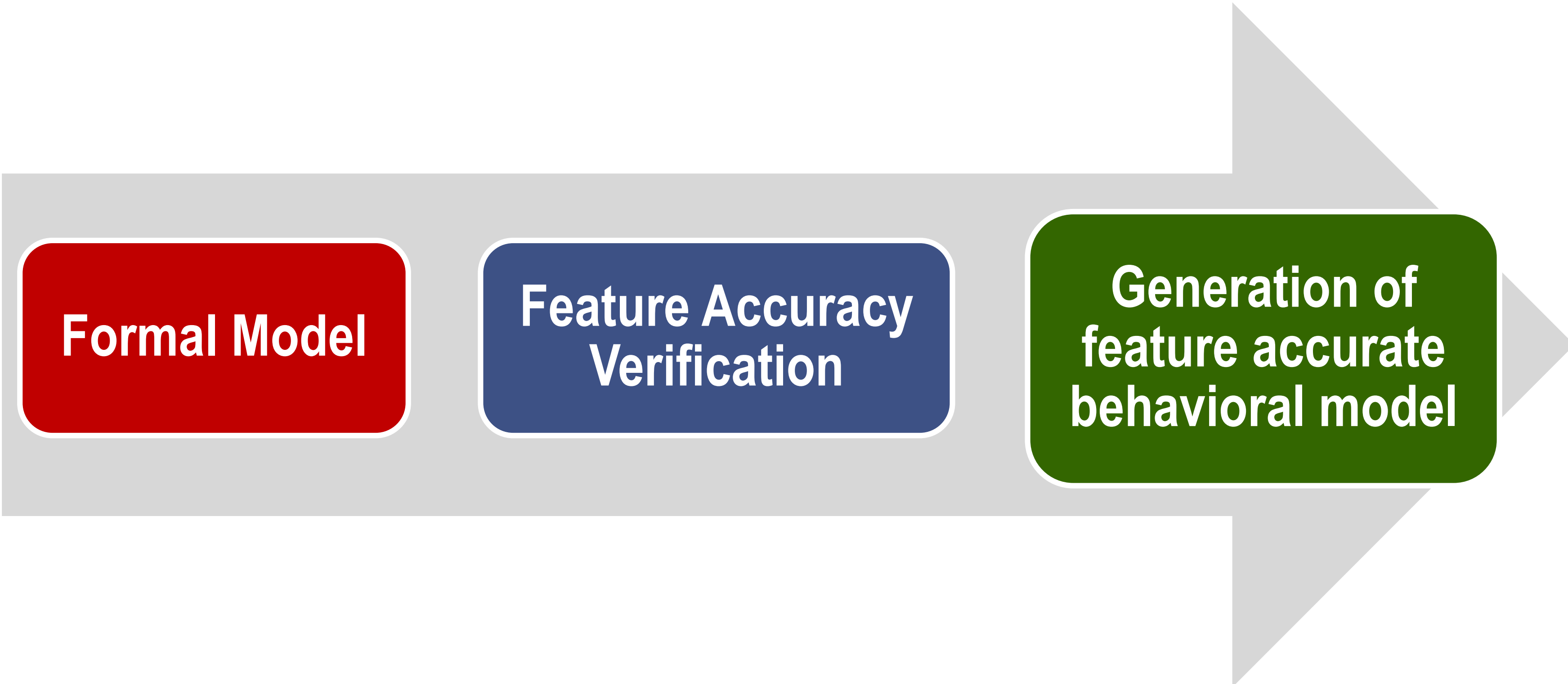
If we take a leaf out of existing practice for designing control systems ,,,

- The control law is designed and modeled using (say) a combination of MATLAB and Simulink/Stateflow
- The model is validated and proven to be correct
- The model is translated into the implementation that runs on the platform
    - There are significant challenges in this last step, but not unsurmountable ones

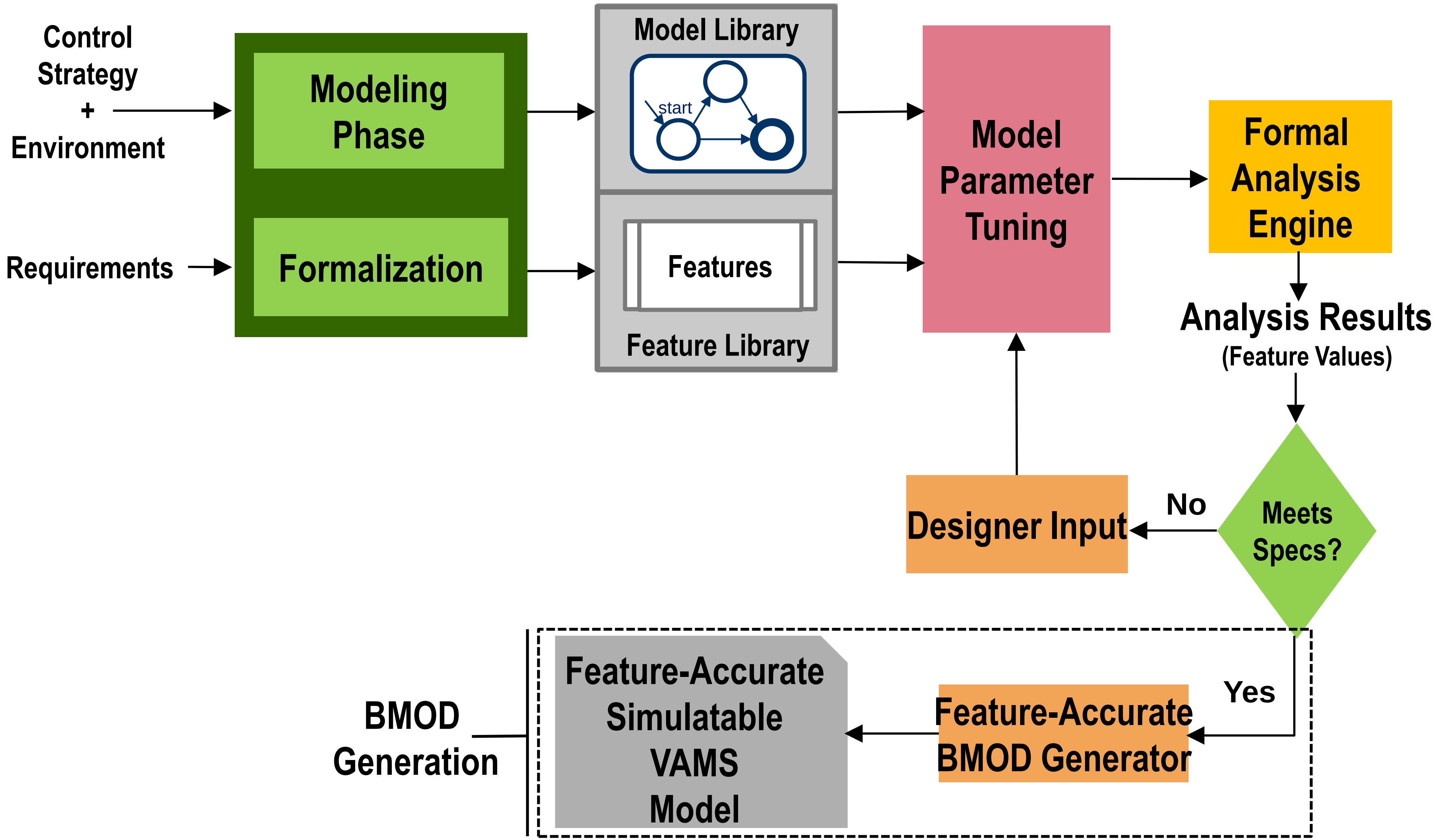
Formal Model driven AMS Behavioral Modeling

Our proposal:

- Formal modeling of design intent
- Formal definition of Features
- Formal methods for proving feature accuracy
- Automatic translation of feature accurate formal model into behavioral models of various types



Proposed Work Flow

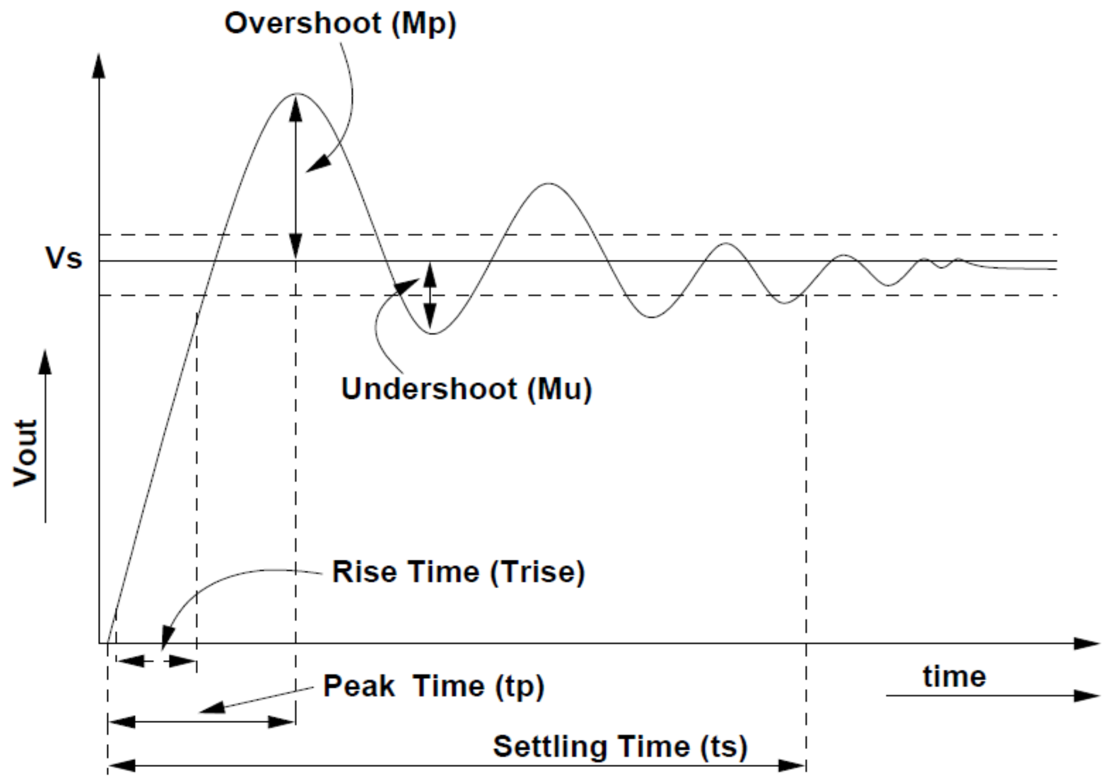


Features: *Real valued functions computed over assertion matches*

Quantitative measurement
over a **behaviour** of a system.

Assertion = Boolean (True/False)

Rise Time of a second order response of a signal is *the time taken for a signal (Vout) to rise from 10% to 90% of its rated value (Vs).*



The Assertion: ***Rise Time should be less than 10ms***

@⁺(M.Vout ≥ 0.1*Vs) ⟹ ##[0:10e-3] @⁺(M.Vout ≥0.9*Vs)

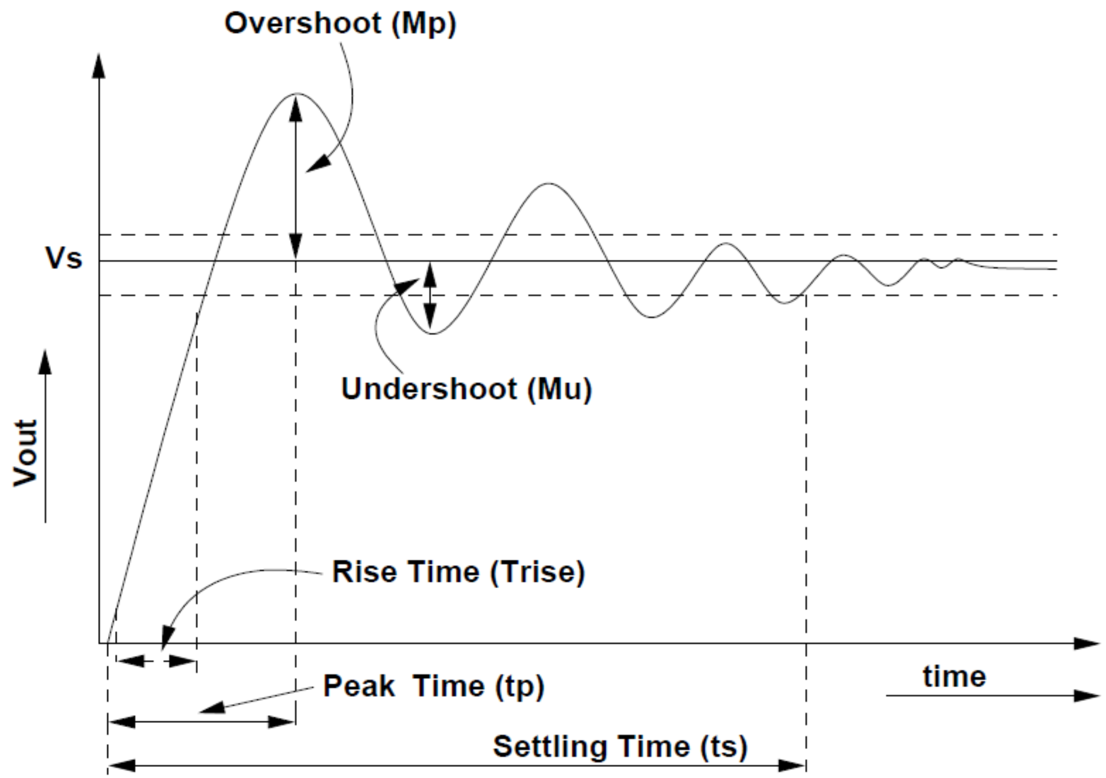
Features: *Real valued functions computed over assertion matches*

Quantitative measurement over a **behaviour** of a system.

Assertion ═ Boolean (True/False)

Feature ═ Real Valued Quantity

Rise Time of a second order response of a signal is *the time taken for a signal (Vout) to rise from 10% to 90% of its rated value (Vs).*



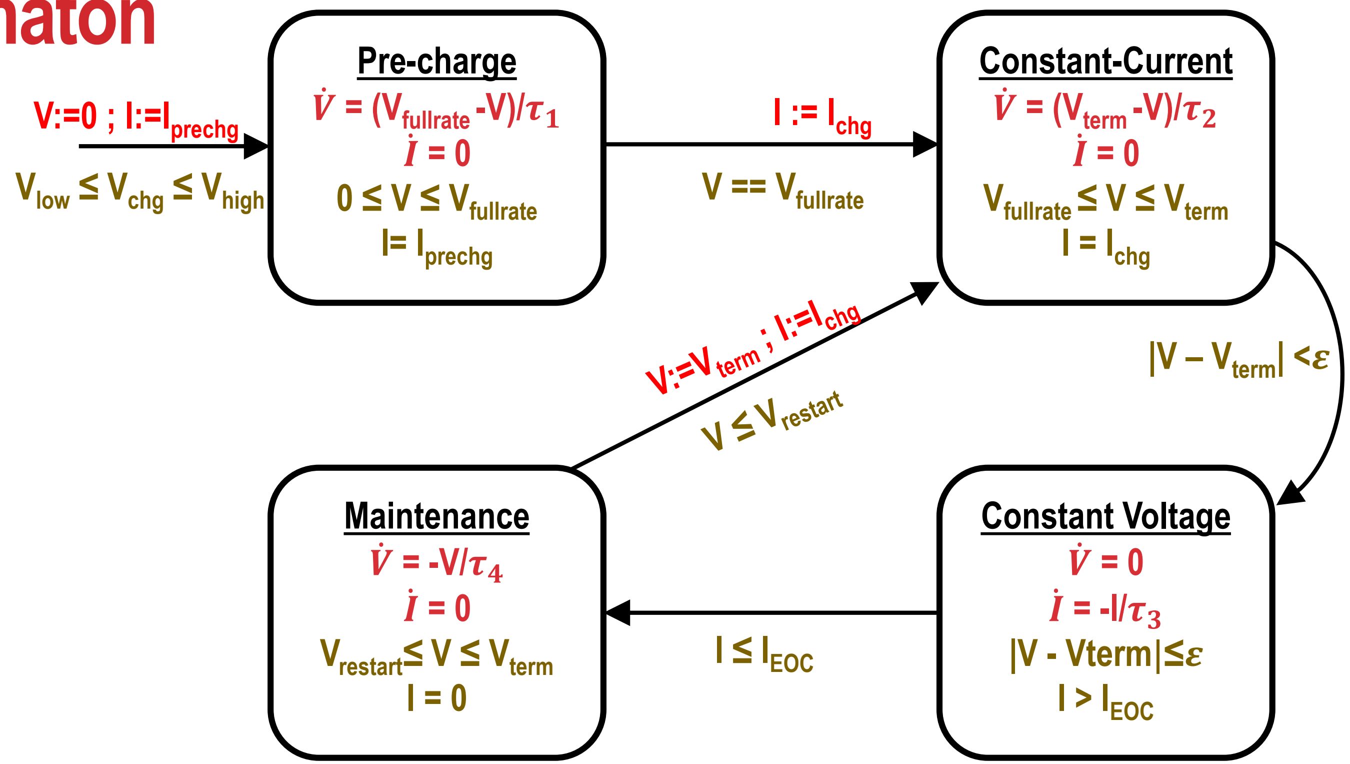
The Assertion: ***Rise Time should be less than 10ms***

The Feature: ***What is the Rise Time of the circuit?***

```
feature RiseTime(Vs);
begin
      var  t1, t2 ;
      @+(M.Vout ≥ 0.1*Vs) ,t1= $time   ##[0:$]     @+(M.Vout ≥0.9*Vs), t2= $time
            |->  RiseTime = t2 - t1;
end
```

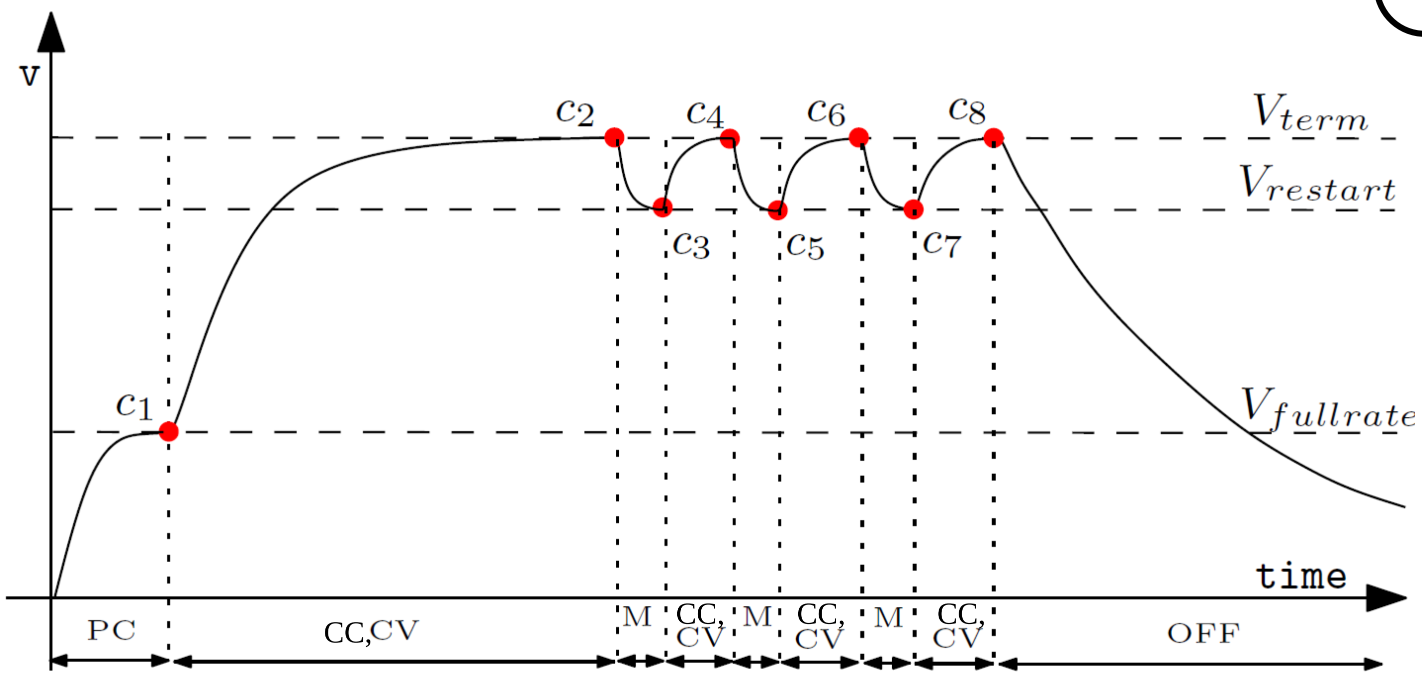
MinRise <= RiseTime <= MaxRise

Model Structure: The Hybrid Automaton



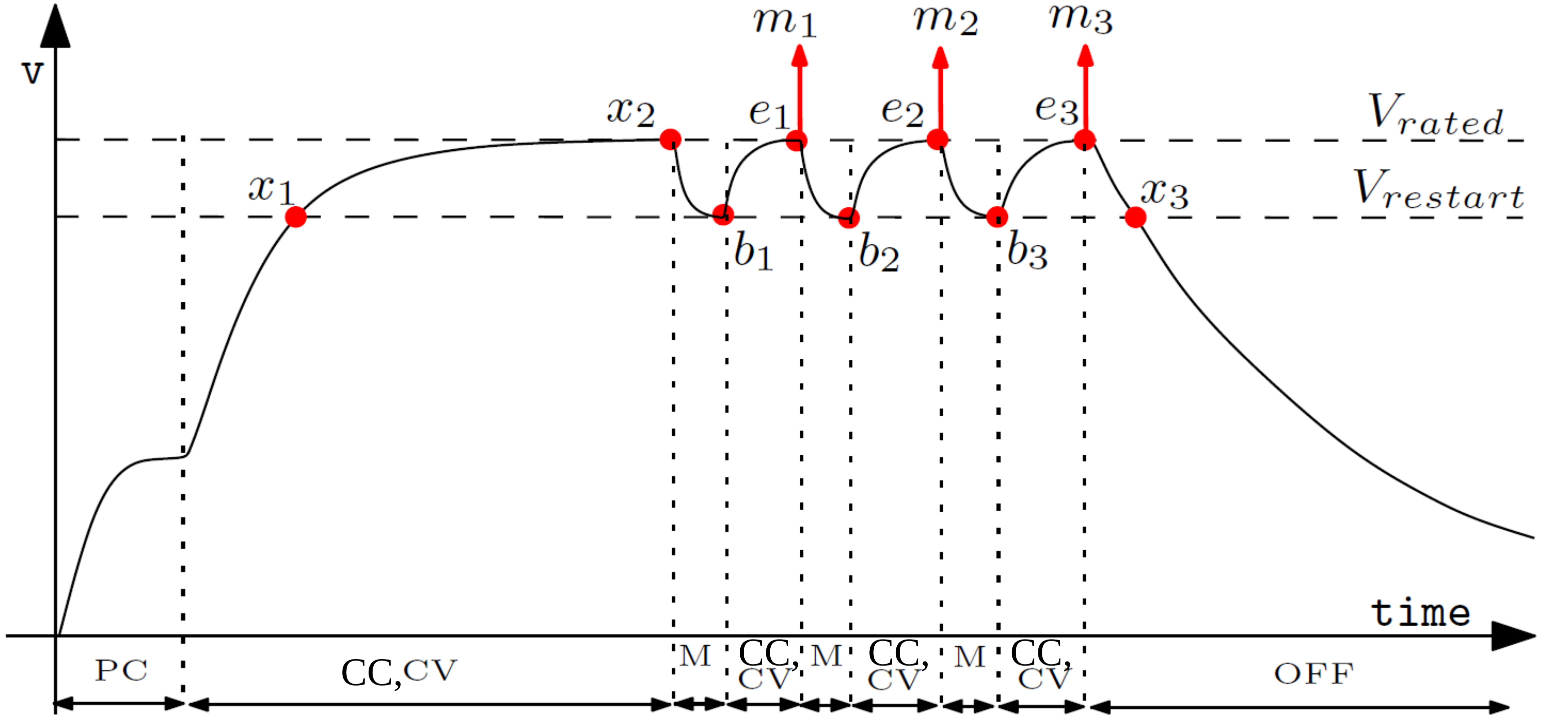
Hybrid Automaton
as a Simulatable model

c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8 represent points at which mode switching occurs.



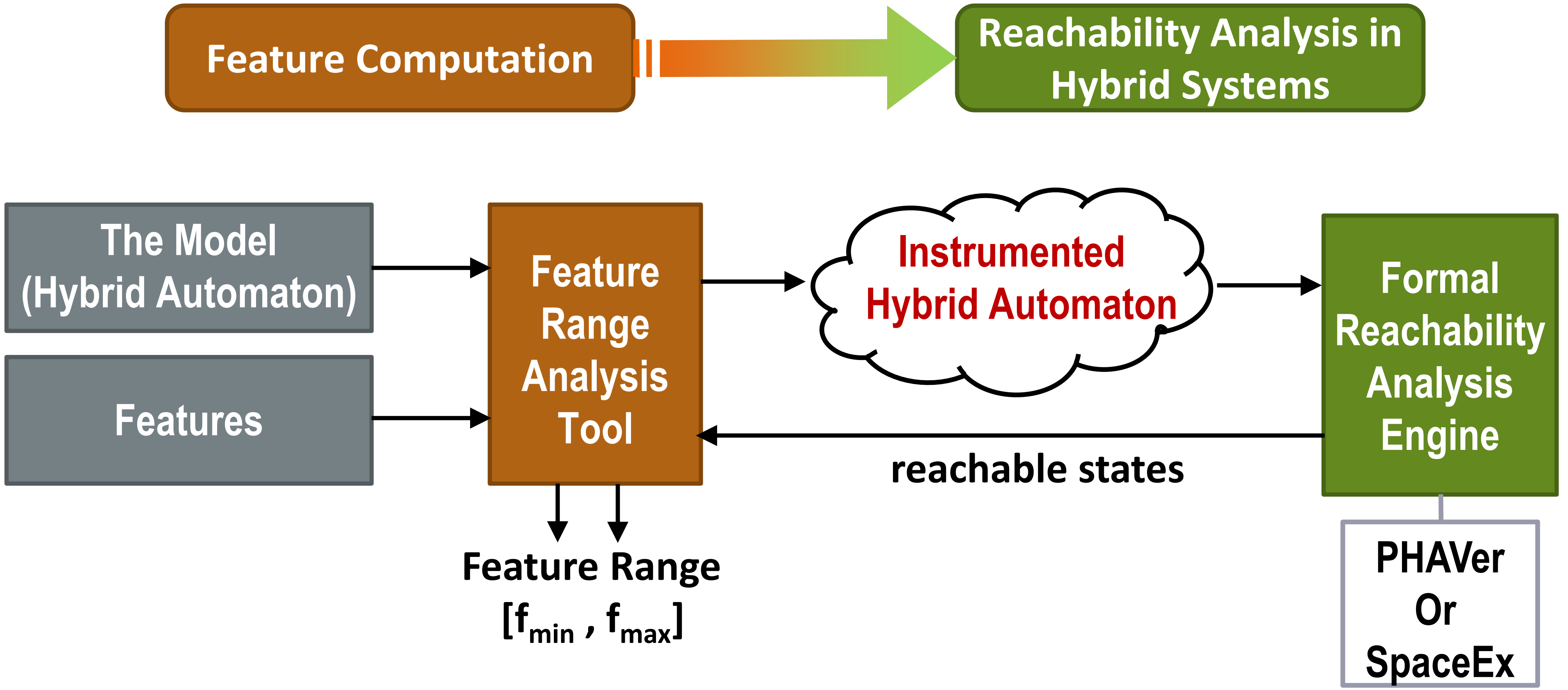
Feature Computation over Sequence Matches

Restoration time for a battery charger:
Time to restore charge in the maintenance mode.




```
feature restorationTime();
begin
    var t1,t2;
    state==M && v==Vrestart, t1 = $time ##[0:$] state == CV && v==Vterm, t2 = $time
        |→ restorationTime = t2-t1;
end
```

Formal Feature Measurement Strategy



GUARANTEED FEATURE RANGE CONTAINMENT

- All possible runs can be tested (No Test-benches prepared).
- Mathematical Proof that the feature range will always be contained within f_{min} and f_{max}.

❖ A. A. B. da Costa, P. Dasgupta and G. Frehse, **Formal feature analysis of hybrid automata**, 2016 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE), Kanpur, India, 2016, pp. 2-11.

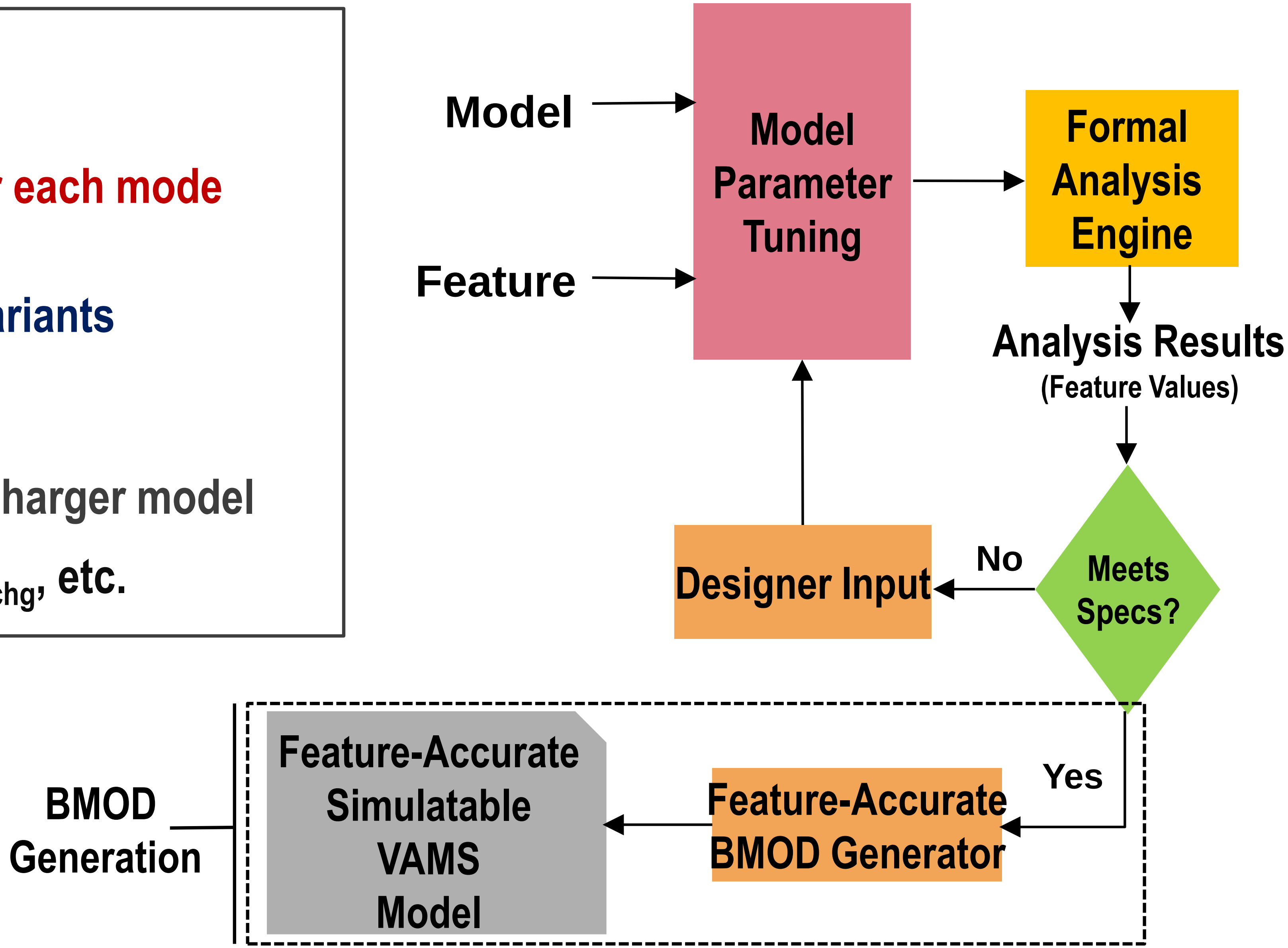
Feature-Accurate Behavioral Model Generation

Model Parameters

- Timing Parameters
  - Time constants for each mode
- Switching Parameters
- Parameters for Mode Invariants and Mode Dynamics

For example, for the battery charger model

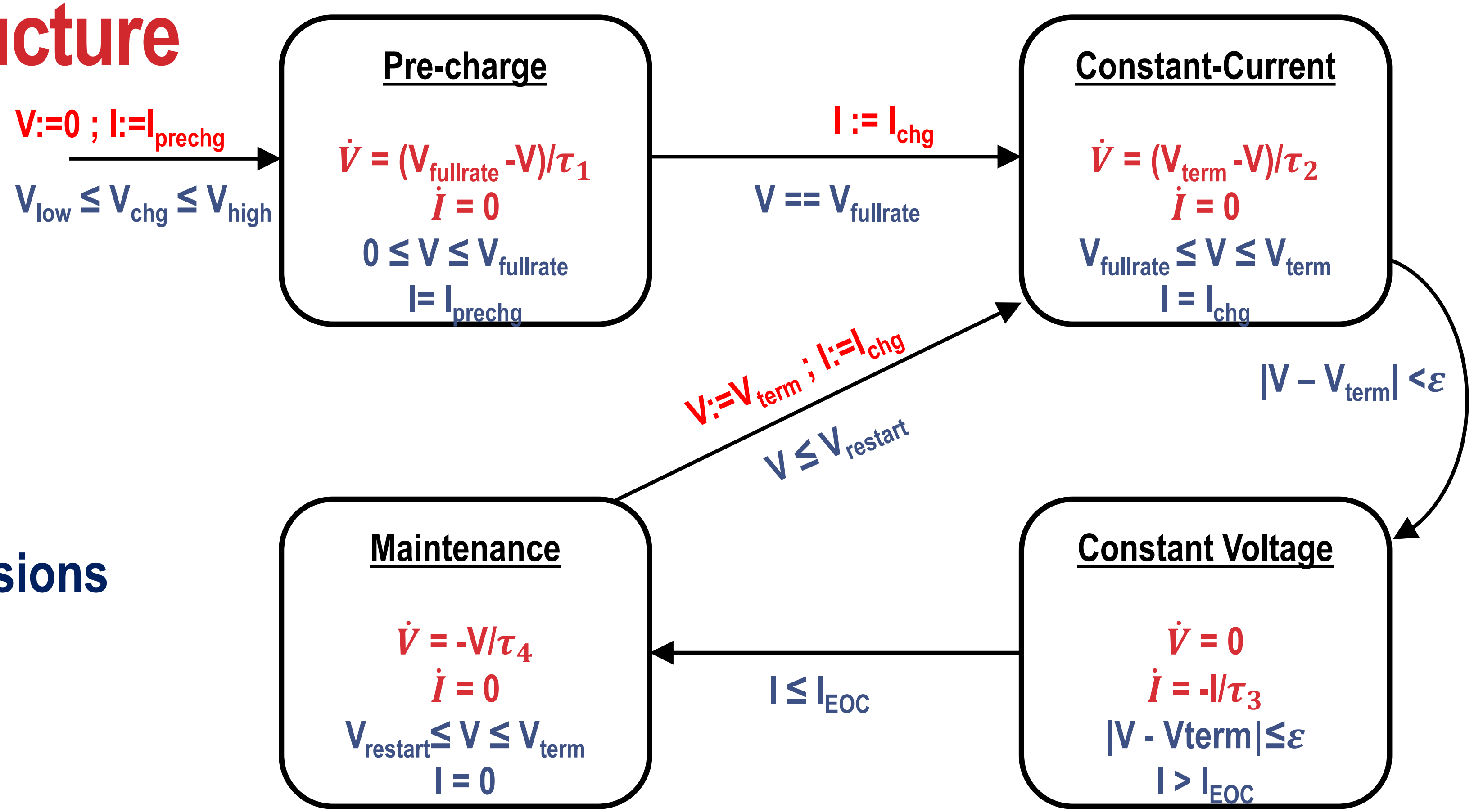
- $V_{restart}$, I_{EOC}, V_{term}, I_{chg}, I_{prechg}, etc.



Behavioral Model Structure

General outline:

- Model parameters are `parameter real`
- Continous parameters are `electrical` variables
- Guards are Boolean expressions over PORVs.



- <u>2 Switch Case Blocks</u>
  - Block 1: Change of control (mode switching)
  - Block 2: Specification of dynamics for each mode of operation.
- Maintaining pin-equivalence for simulation
  - Use of sense resistors to sense currents on input lines
  - Use of VAMS voltage and current constructs to specify outputs.

Code Snippets

INITIAL SET

```
@(initial_step) begin
        Vpre = 0;
        t_PC = $abstime;
        STATE = PRE_CHARGE;
        VTG = 0;
        ICURR = 0;
end
```

GUARD PROPOSITIONS

```
p2 = ( ( V(icurr) >= 0  ) && ( V(vtg) >= 0  )) ? 1 : 0;
p4 = ( ( V(vtg) >= Vfullrate  )) ? 1 : 0;
p6 = ( ( V(vtg) - Vterm <= epsilon  ) &&
     ( V(vtg) - Vterm >= -epsilon  )) ? 1 : 0;
p8 = ( ( V(icurr) <= IEOC  )) ? 1 : 0;
p10 = ( ( V(vtg) <= Vrestart  )) ? 1 : 0;
```

MODE SWITCHING CONTROL

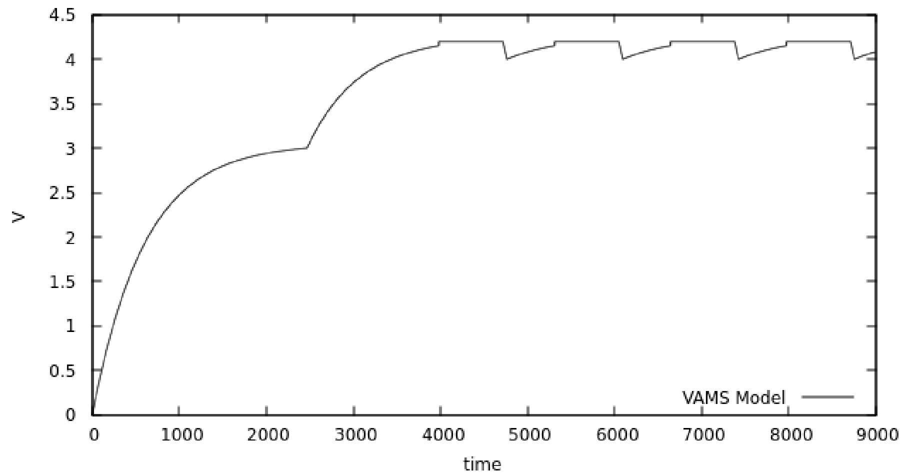
```
case(STATE)
 OFF: begin
  if(p2) begin
    $display("Guard(OFF -> PRE_CHARGE)");
    case(STATE)
      OFF: begin
        if(p2) begin
            STATE = PRE_CHARGE;
            t_PC=$abstime;
            Vpre = VTG;
         end
      end
      PRE_CHARGE: begin
        if(p4) begin
           t_CC=$abstime;
           Vchg = VTG;
           STATE = CC;
         end
      end
```

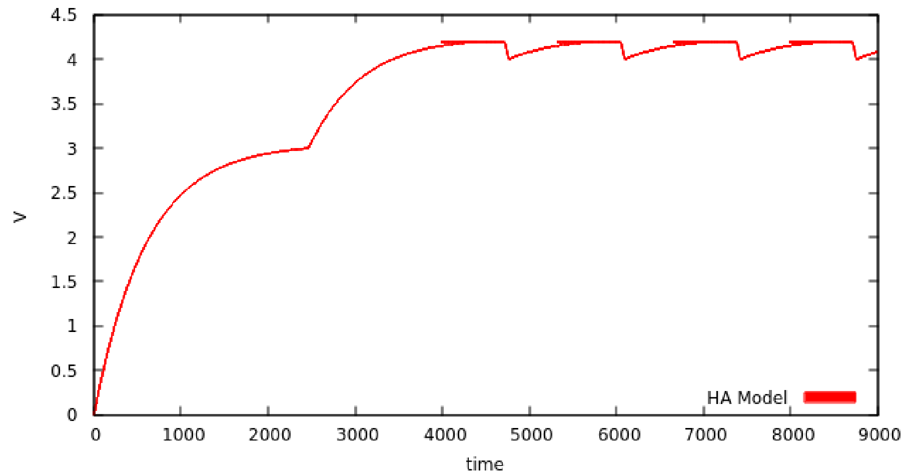
Code Snippets

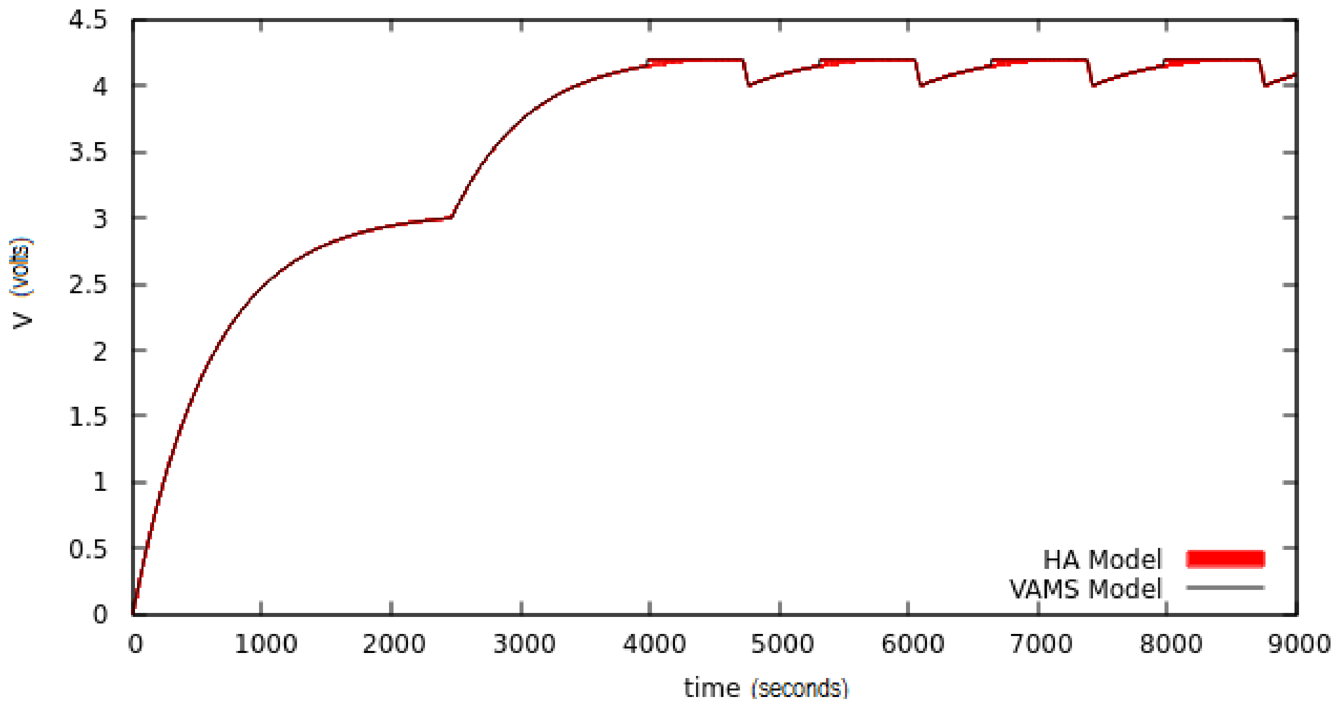
MODE DYNAMICS

```
case(STATE)

    ...

    PRE_CHARGE: begin
        VTG = Vpre + (Vfullrate - Vpre + epsilon)*(1-exp(-($abstime-t_PC)/T1));
        ICURR = Iprechg;
    end

    CC: begin
        VTG = Vchg + (Vterm - Vchg + epsilon)*(1-exp(-($abstime-t_CC)/T2));
        ICURR = Ichg
   end

    ...

endcase
```

Simulation Results







Summary of Work

Design validation necessitates the use of formal specification and evaluation of feature ranges.

- Formal verification of AMS designs as *unstructured* Behavioural Models still eludes the verification community.
- Formal Specification of a circuit as a skeletal model is possible using the construct of a Hybrid Automaton.
- The Feature Indented Assertion Language introduced by us in the past supports the specification of features for AMS designs.
- Formal Feature evaluation is germane to Analog Design Validation, and is a fairly recent development.

Feature-Accurate BMODs can be generated using Formal Methods.

- Formal Analysis techniques can be used to compute a conservative range for features of AMS designs (such as rise time, overshoot, etc.)
- A feature-accurate formal model (golden model) can be automatically translated into an equivalent simulatable model in VAMS for use in further verification tasks.

Key References

- A. A. B. da Costa, P. Dasgupta and G. Frehse, **Formal feature analysis of hybrid automata**, 2016 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE), Kanpur, India, 2016, pp. 2-11.
- A. Ain, A. A. B. da Costa, and P. Dasgupta**, Feature indented assertions for analog and mixed-signal validation,** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 11, pp. 1928–1941, Nov 2016.
- R. Alur, C.Courcoubetis,et al, **The algorithmic analysis of hybrid systems**. Theoretical Computer Science, 138:3-34, 1995.
- G. Frehse et al. **SpaceEx: Ecalable Verification of Hybrid Systems**. In Compuer Aided Verification (CAV), 2011.
- A. Ain, D. Pal, P. Dasgupta, S. Mukhopadhyay, R. Mukhopadhyay, and J. Gough**. Chassis: A platform for verifying PMU integration using autogenerated behavioral models**. ACM Transactions on Design Automation of Electronic Systems, 16(3):33:1-33:30, June 2011.
- S. Mukherjee, A. Ain, S. K. Panda, R. Mukhopadhyay, and P. Dasgupta, **A Formal Approach for Specification-driven AMS Behavioral Model Generation**, in Proc. of DATE, 2009.
- Analog Devices. **Designers guide to Charging Li-Ion Batteries**, http://seattlerobotics.org/encoder/200210/liion2.pdf, March 1998

Thank you for your attention