ATVA 2017

# ForFET: A Formal Feature Evaluation Tool for Hybrid Systems

António A. Bruto da Costa,
Dept. of Computer Sc & Engg

Pallab Dasgupta
Professor, Dept. of Computer Sc & Engg

**Acknowledging**

Semiconductor Research Corporation

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Assertions are commonplace in verification today. Standards include SVA and PSL.

AMS assertions have been studied. No standards yet.

A primary contribution of this research is in *formally* analyzing AMS features which look beyond assertions.

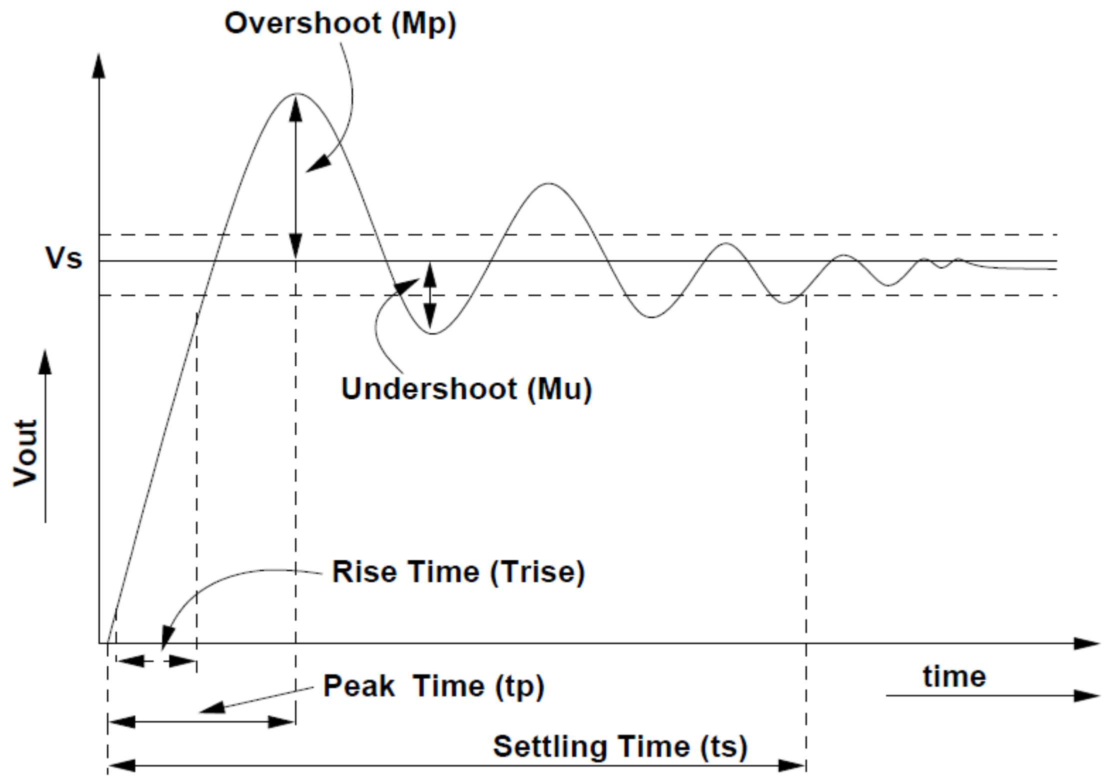
Features = *Real valued functions computed over assertion matches.*

# 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***

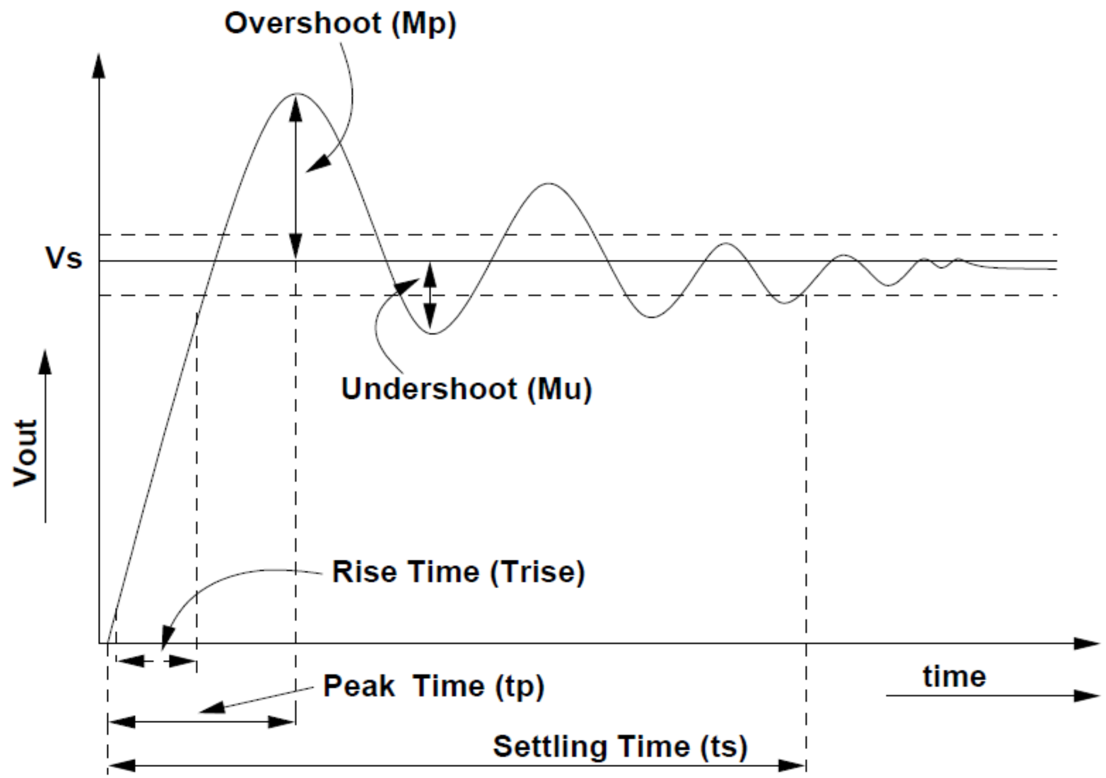
**@⁺(M.Vout ≥ 0.1*Vs) ⟹ ##[0:10e-3] @⁺(M.Vout ≥0.9*Vs)**

# 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).*




```
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
```

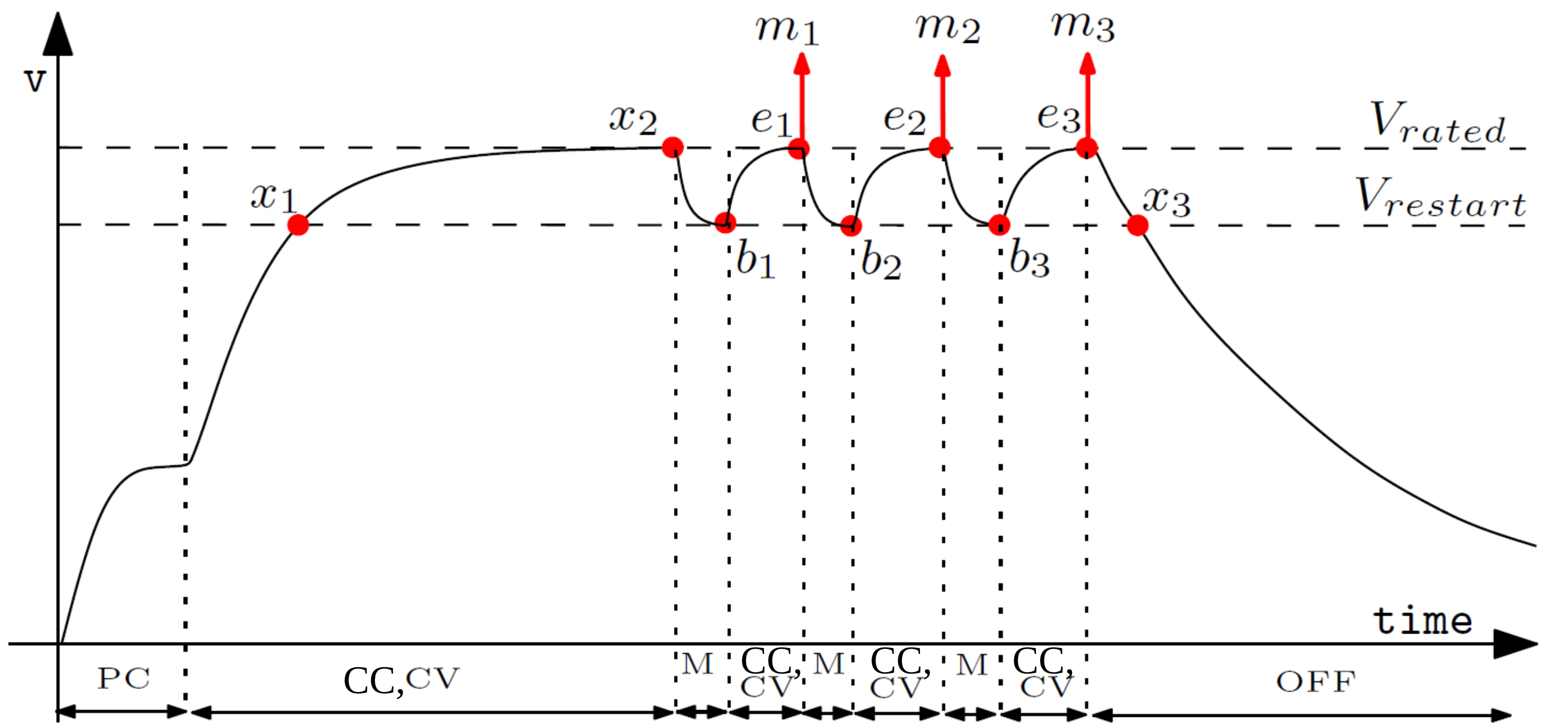
**The Assertion**: ***Rise Time should be less than 10ms***

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

**MinRise <= RiseTime <= MaxRise**

# Feature Computation over Sequence Matches

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



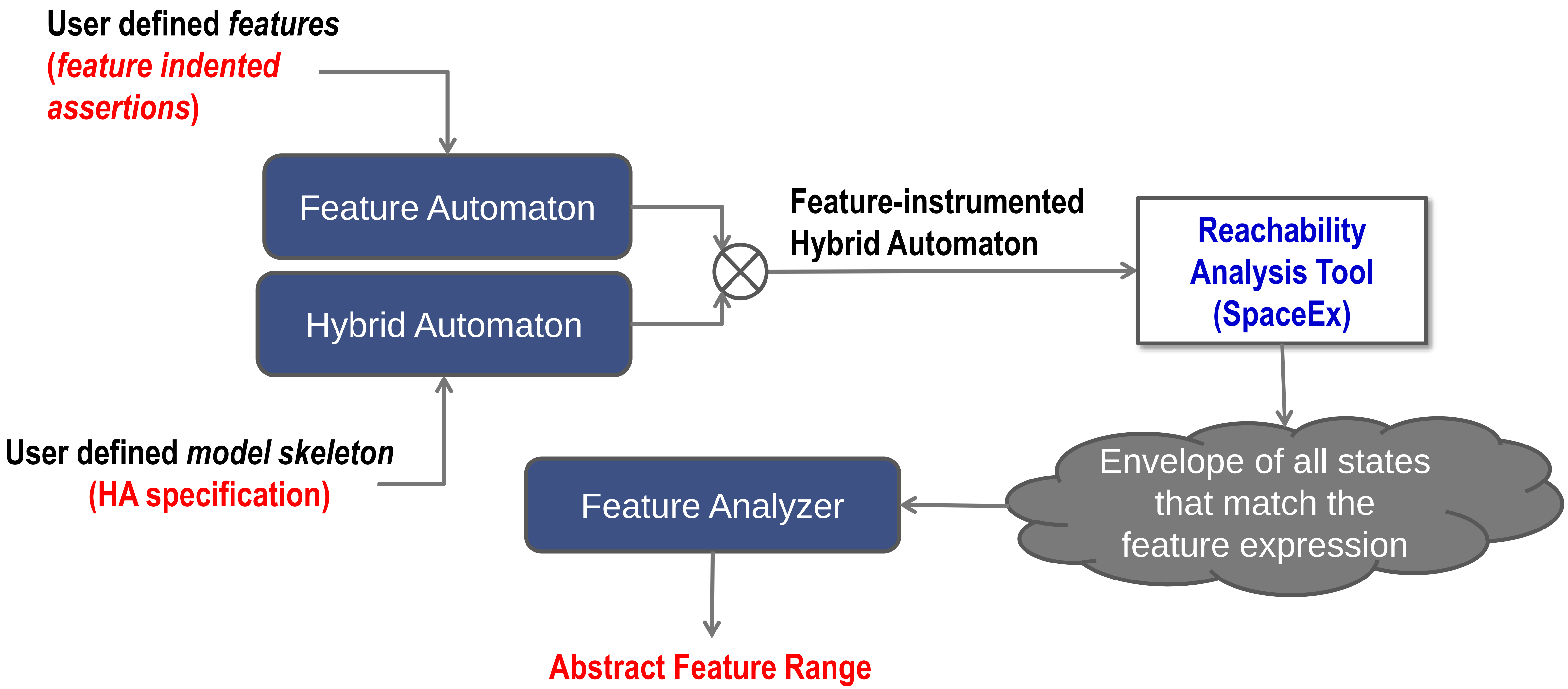

```
feature restorationTime();
begin
    var t1,t2;
    state==M && v==Vrestart , t1 = $time ##[0:$] state == CV && v==Vterm , t2 = $time
        |→ restorationTime = t2-t1;
end
```

# Our Contributions

- **The study presented here discusses:**
  - A Generalized Methodology for Constructing Feature Monitors
  - Using Feature Monitors for Analyzing Hybrid Automata
  - The ForFET Tool for Formal Feature Analysis

- **Our past work in this area:**
  - ➢ The Feature Indented Assertion (FIA) language for specifying features was introduced in, A. Ain, A. A. B. da Costa, and P. Dasgupta, "Feature Indented Assertions for Analog and Mixed-Signal Validation," IEEE TCAD, DOI:10.1109/TCAD.2016.2525798, 2016.
  - ➢ The notion of formally analyzing features over HA was introduced by us first in, A. A. B. da Costa and P. Dasgupta, "Formal interpretation of assertion based features on AMS designs," IEEE Design & Test, vol. 32 (1), pp. 9–17, 2015.

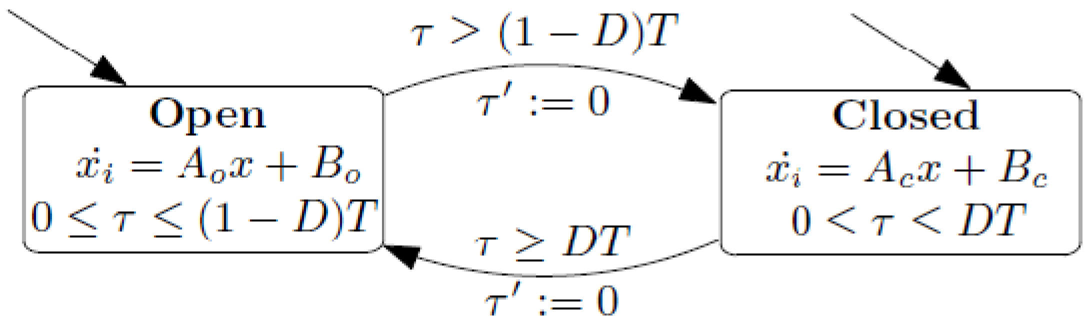
# Working of ForFET



# ForFET Methodology

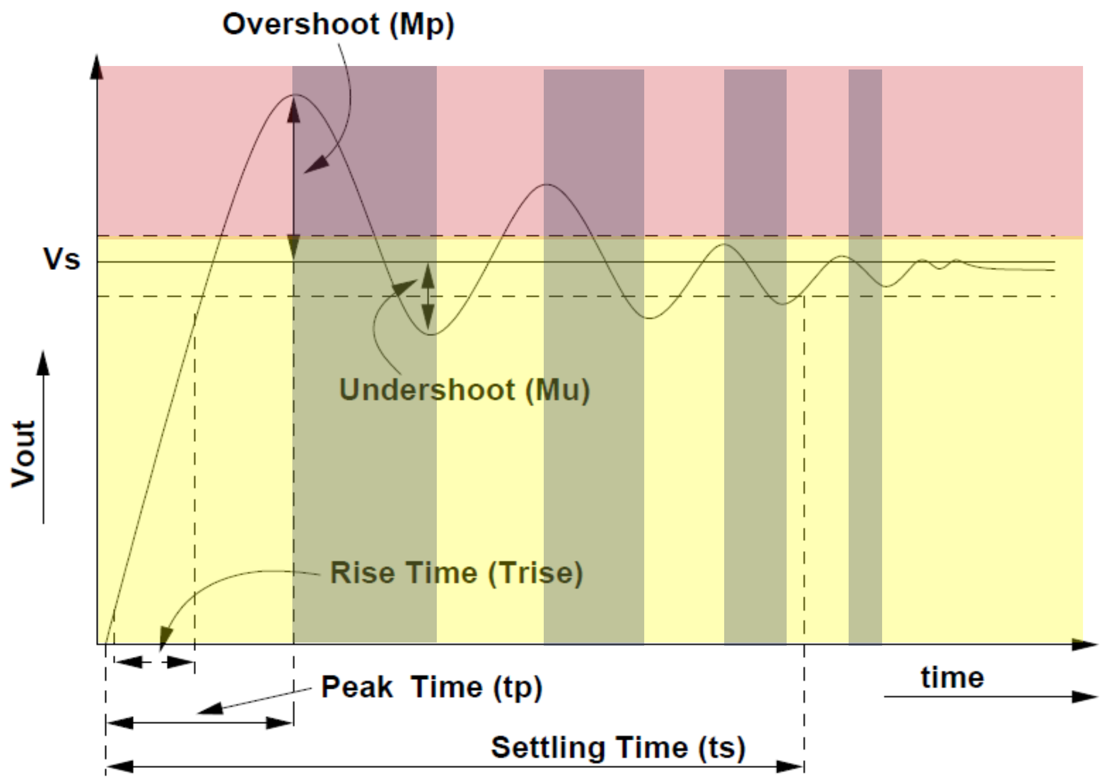
## Step 1: The Feature

**Settle Time:** *Time taken for the output voltage to settle to below Vr + E, where Vr is the rated voltage for the regulator, for two successive openings of the capacitor switch*

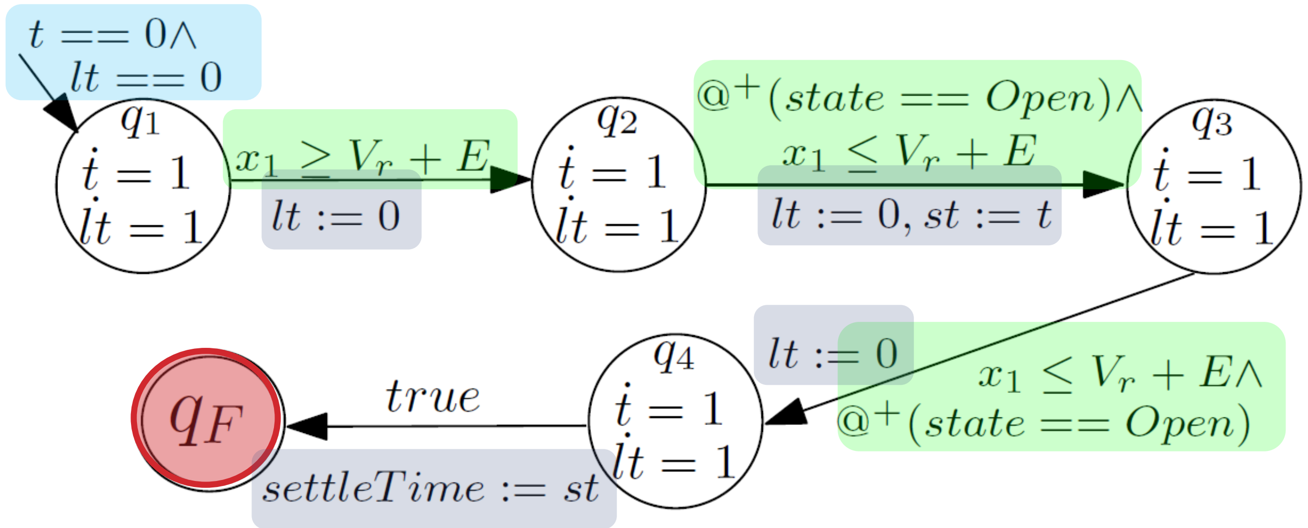



**Hybrid Automaton of a Buck Regulator**

```
feature settleTime(Vr,E);
begin
        var st;
        (x1>=Vr+E) ##[0:$]
                @+(state==Open) && (x1<=Vr+E), st=$time ##[0
                        @+(state==Open) && (x1<=Vr+E)
        |-> settleTime = st;
end
```



# ForFET Methodology
## Step 2: The Feature Automaton




**feature settleTime(Vr,E);**

**begin**

**var st;**

$q_1$ **(x1>=Vr+E) ##[0:$]** $q_2$

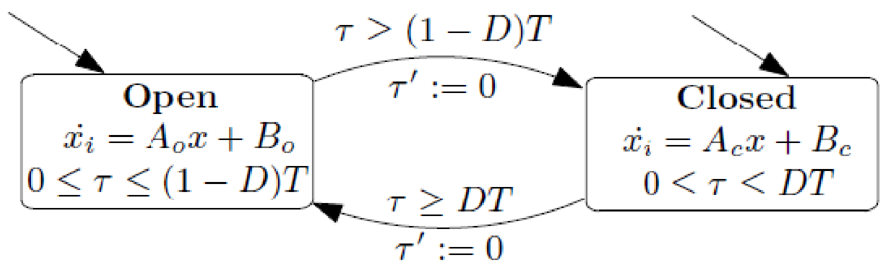
$q_2$ **@+(state==Open) && (x1<=Vr+E), st=$time ##[0:$]** $q_3$

$q_3$ **@+(state==Open) && (x1<=Vr+E)** $q_4$

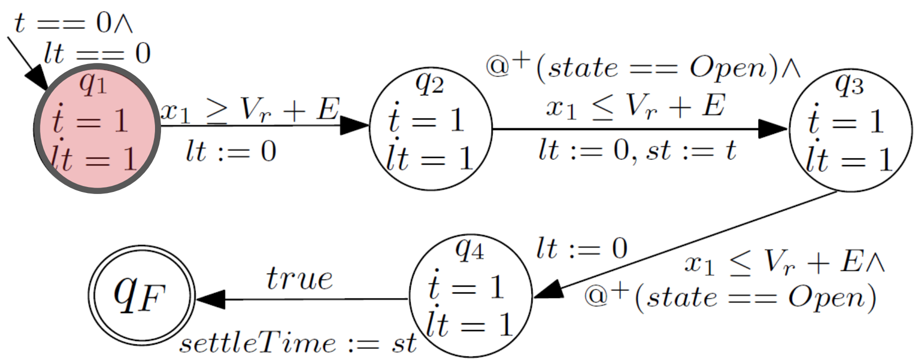
$q_4$ **|-> settleTime = st;** $q_F$

**end**

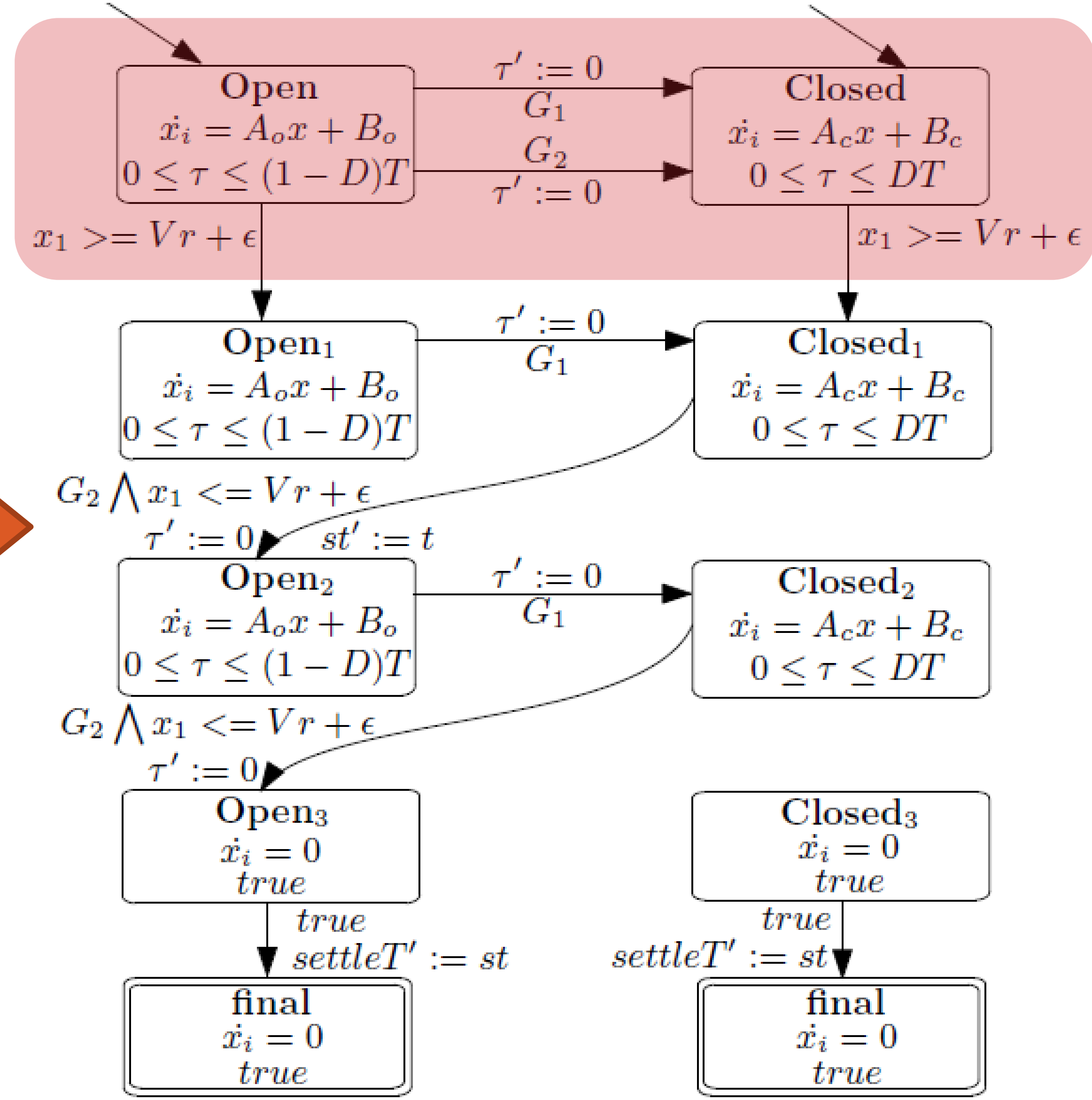
# ForFET Methodology
## Step 3: The Feature Tuned Hybrid Automaton



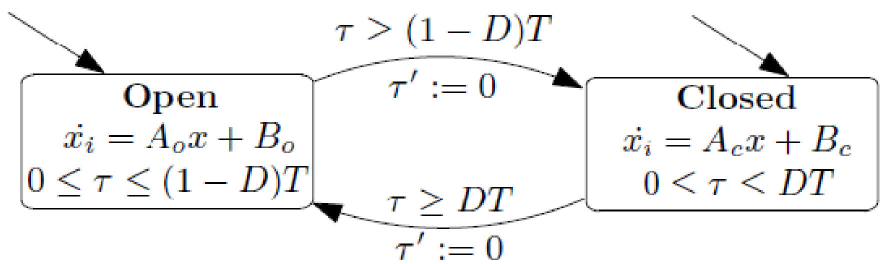

**Hybrid Automaton of a Buck Regulator**



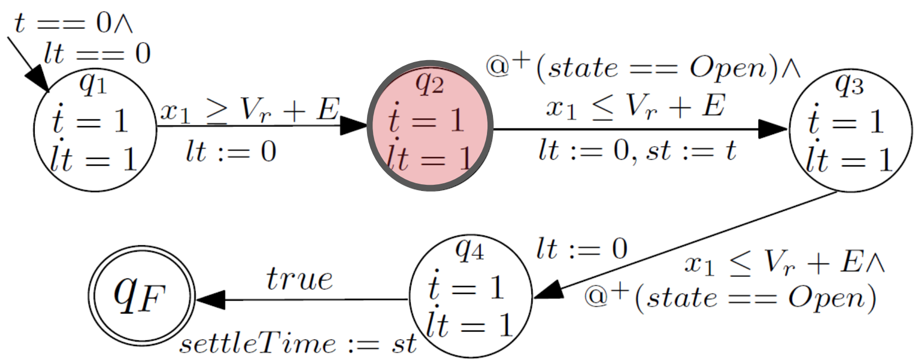

**Feature Automaton for settleTime**



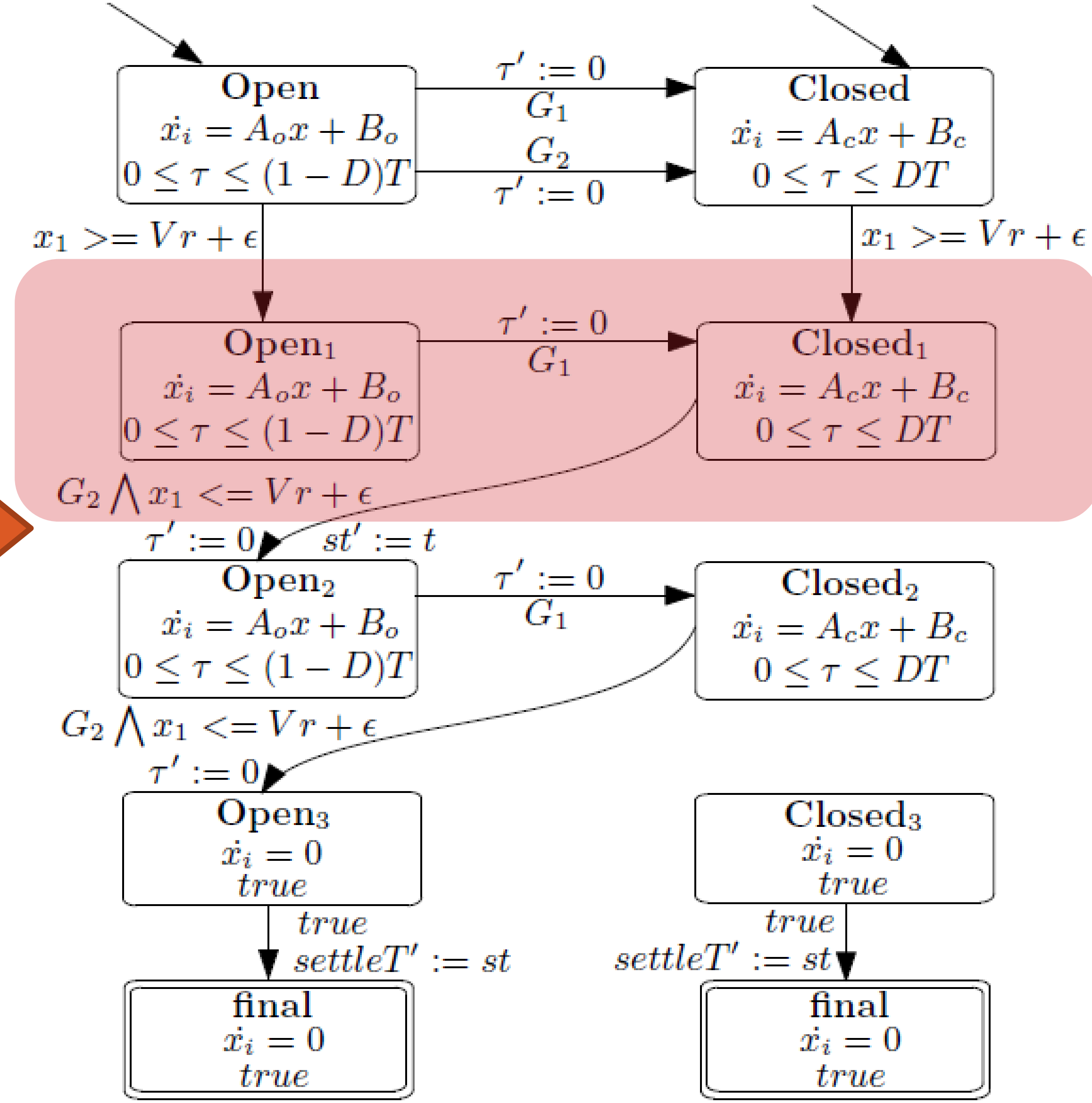
# ForFET Methodology
## Step 3: The Feature Tuned Hybrid Automaton



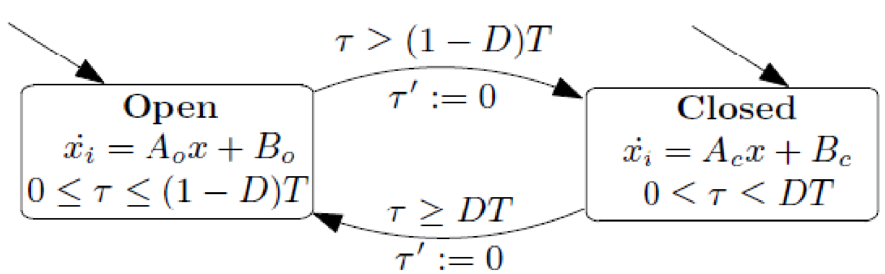

Hybrid Automaton of a Buck Regulator



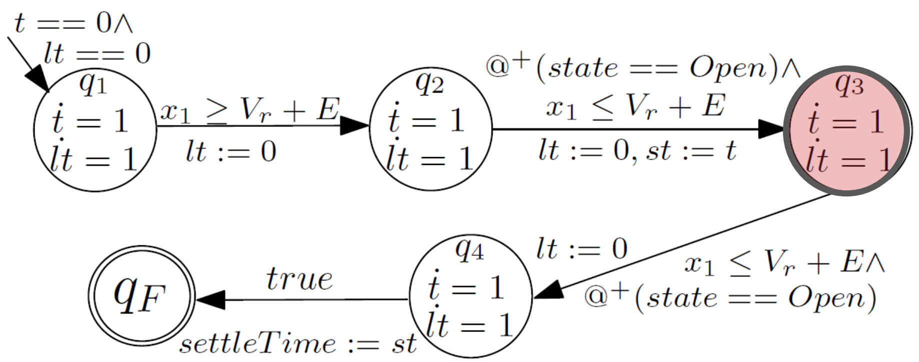

Feature Automaton for settleTime



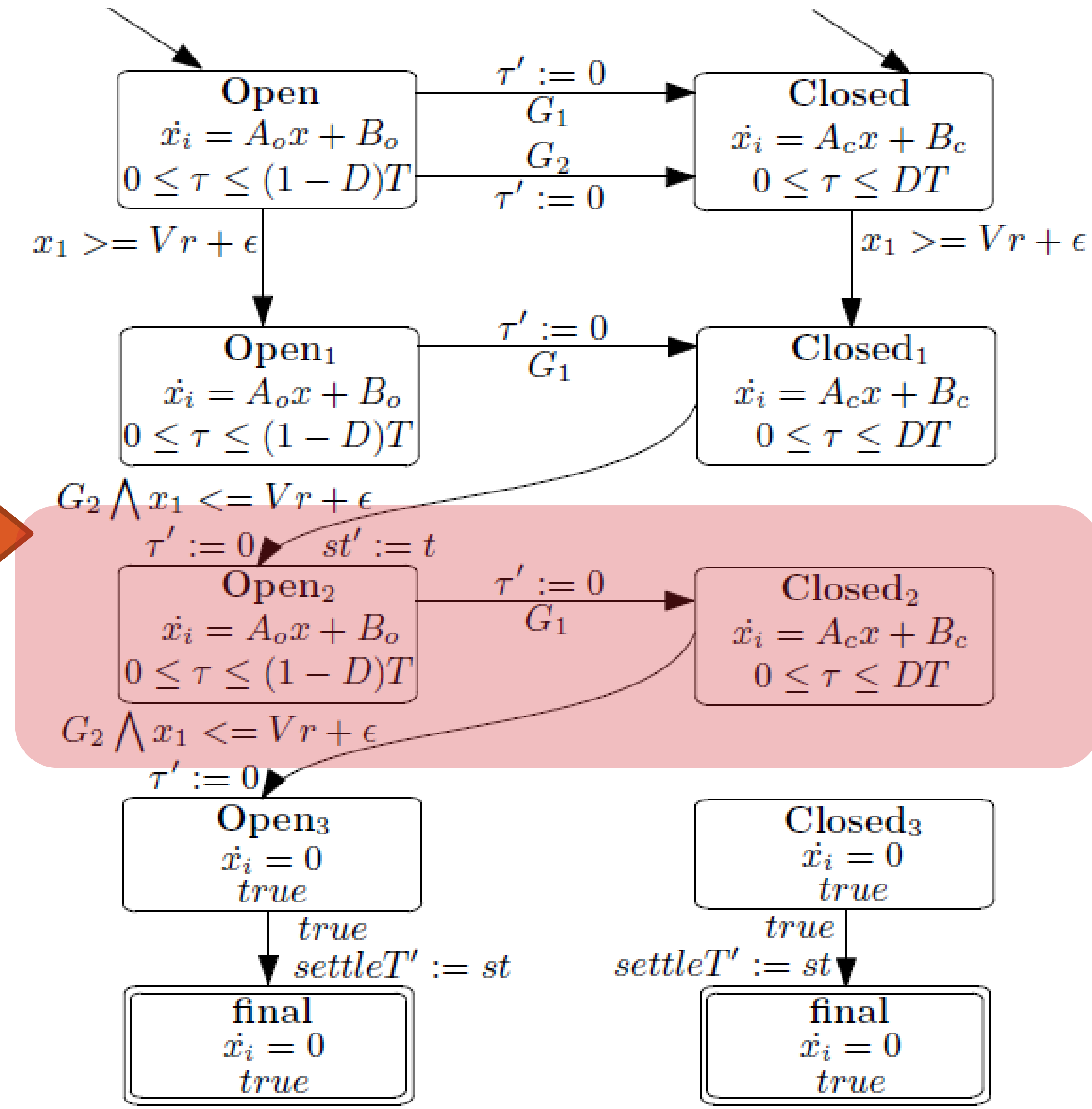
# ForFET Methodology
## Step 3: The Feature Tuned Hybrid Automaton



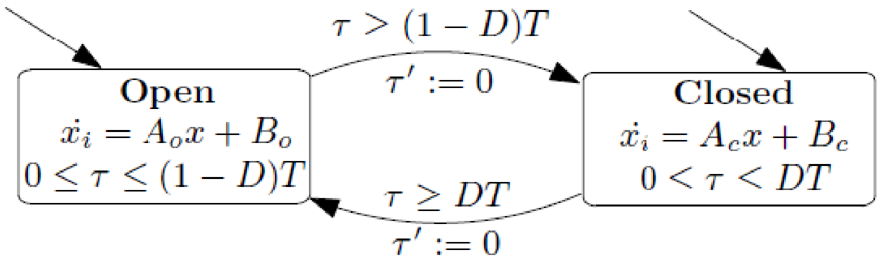

Hybrid Automaton of a Buck Regulator



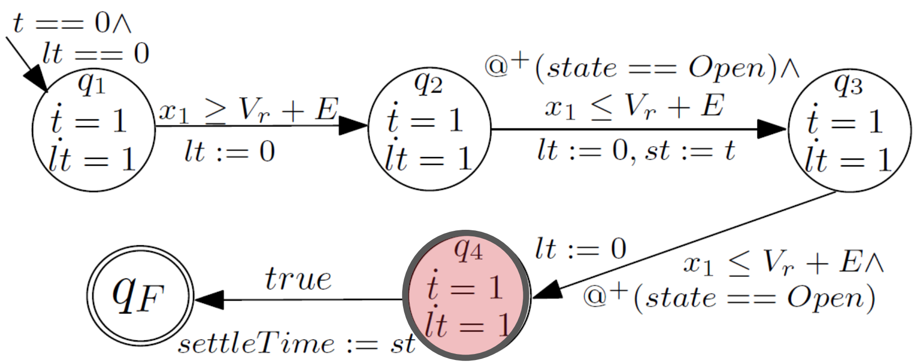

Feature Automaton for settleTime



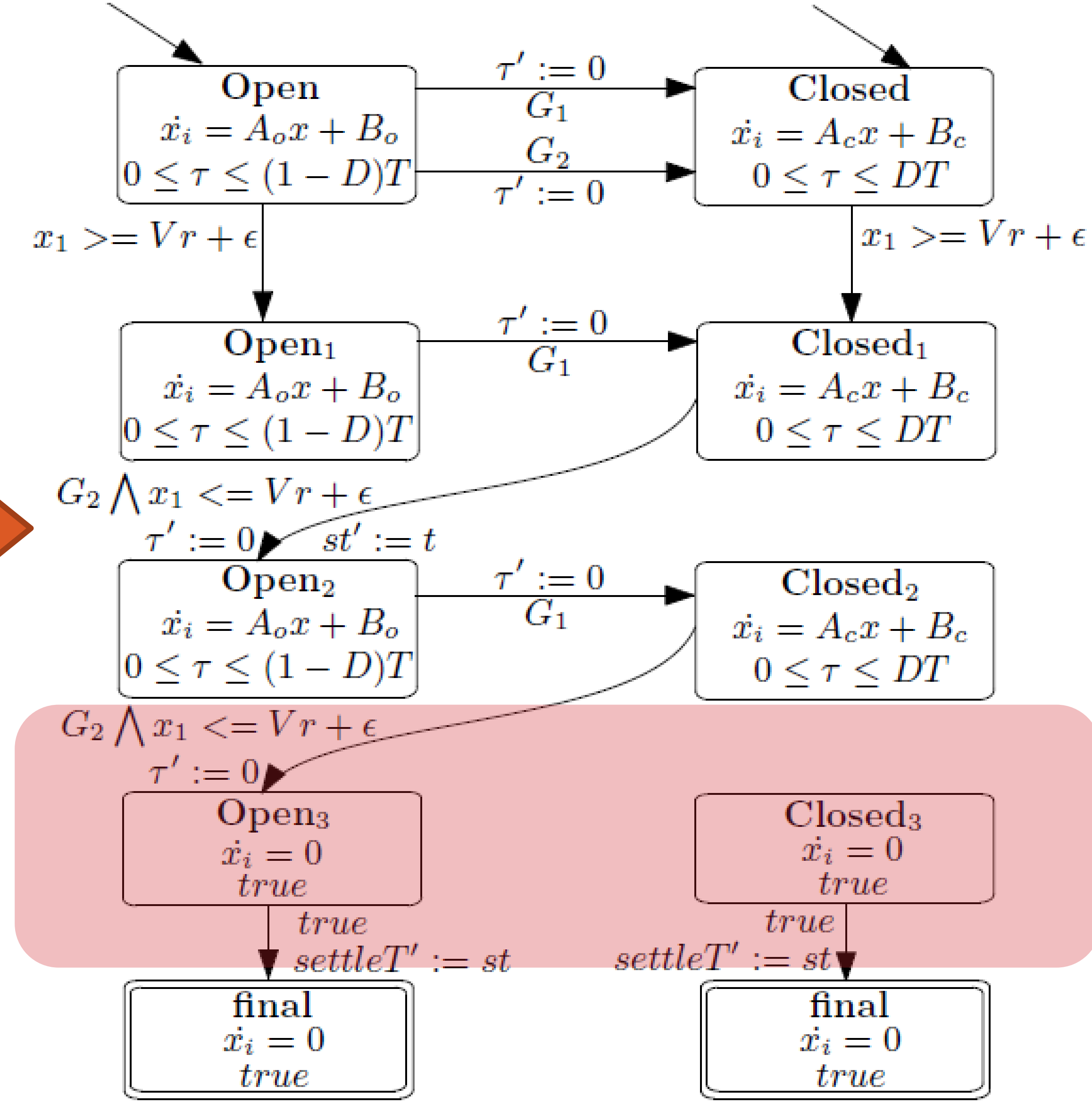
# ForFET Methodology
## Step 3: The Feature Tuned Hybrid Automaton



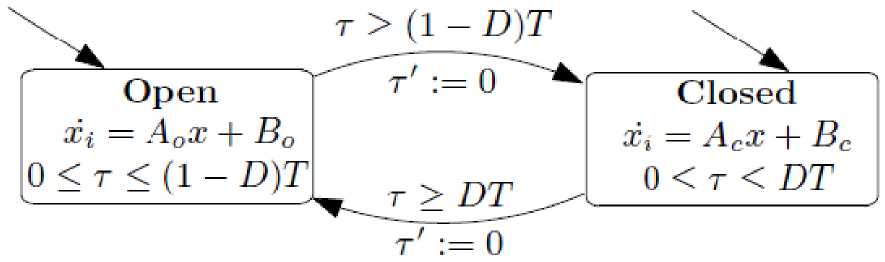

Hybrid Automaton of a Buck Regulator



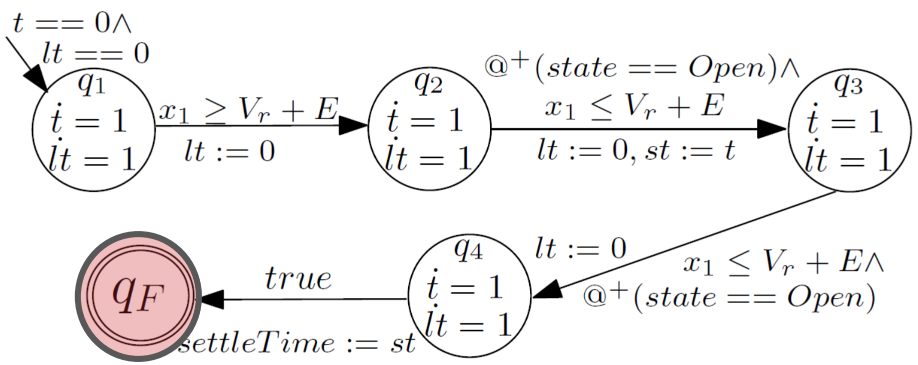

Feature Automaton for settleTime



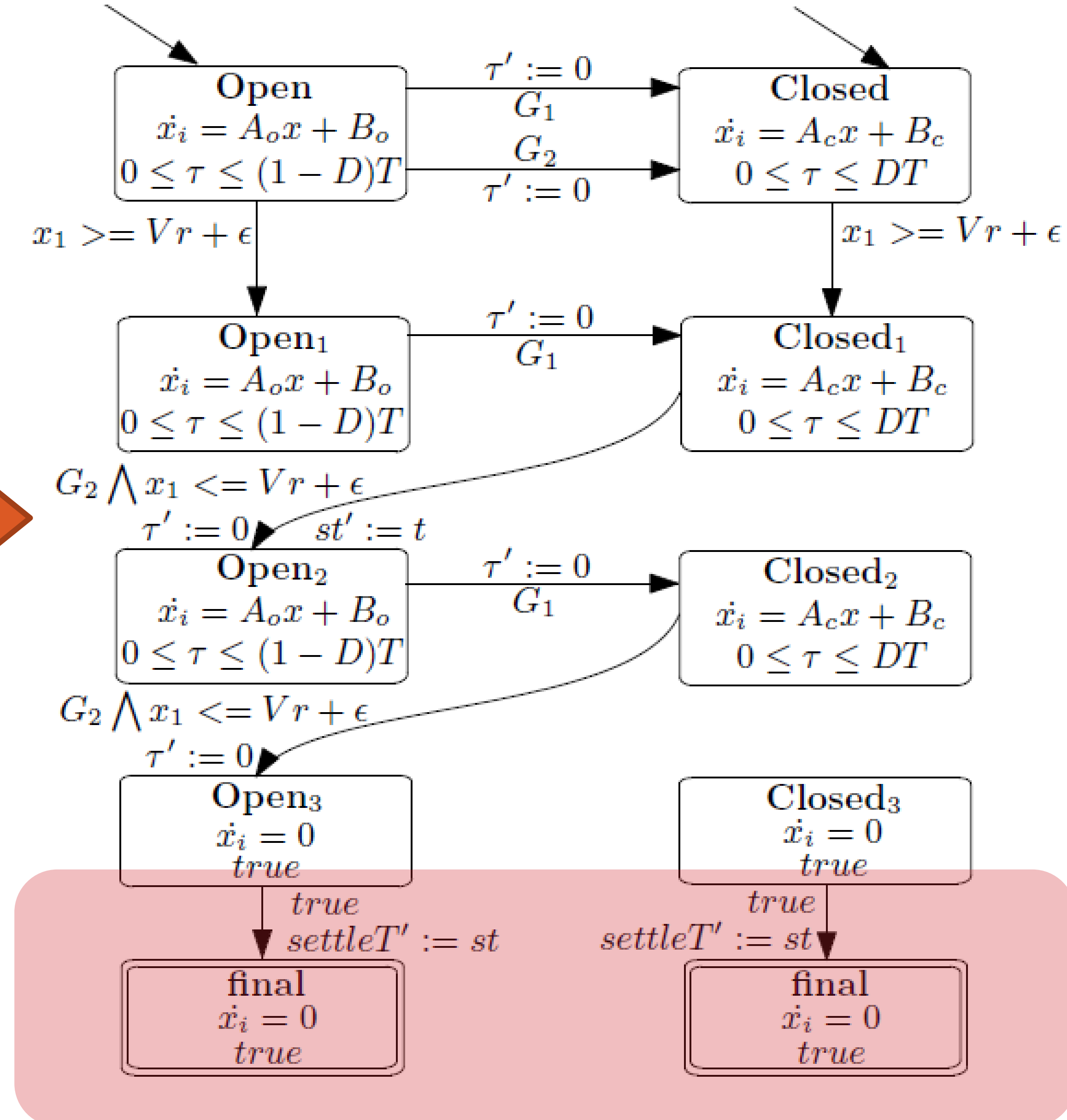
# ForFET Methodology
## Step 3: The Feature Tuned Hybrid Automaton




**Hybrid Automaton of a Buck Regulator**

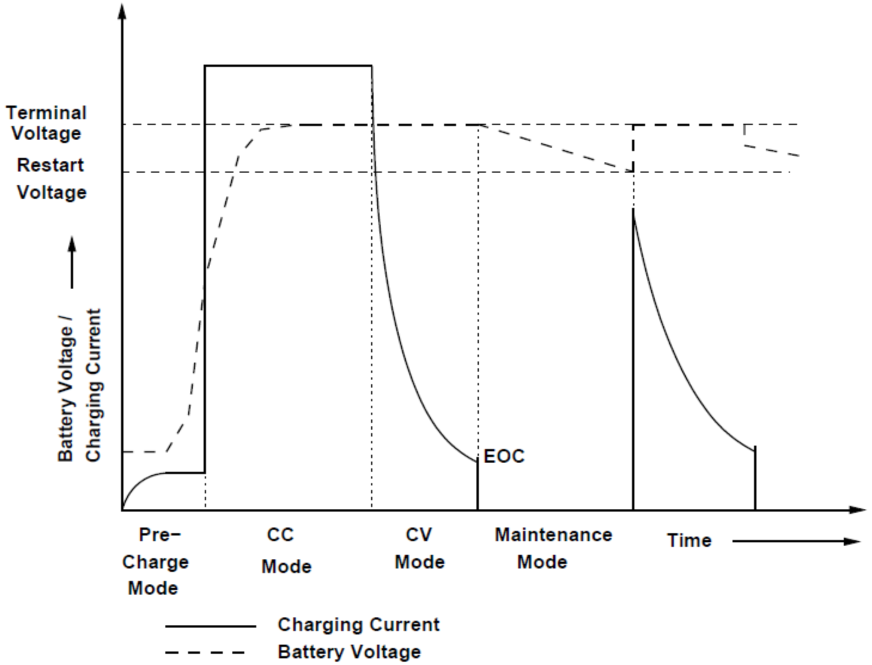



**Feature Automaton for settleTime**

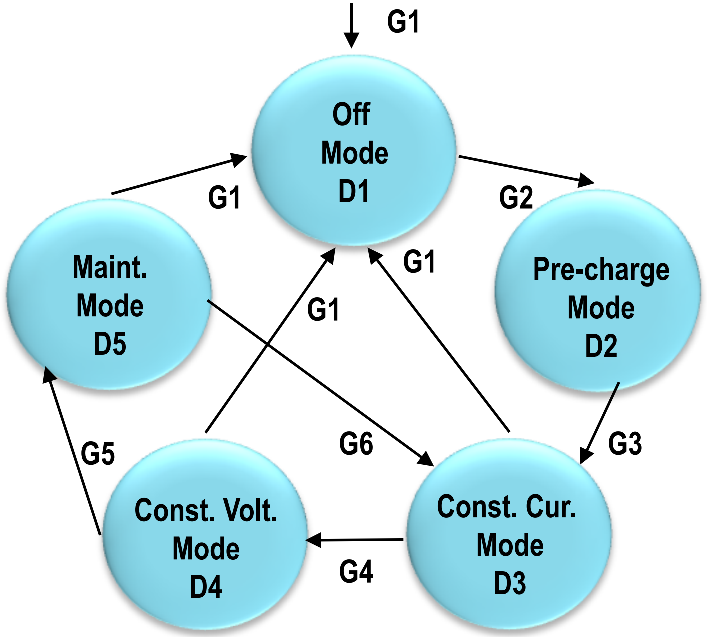


# Case Studies and Results

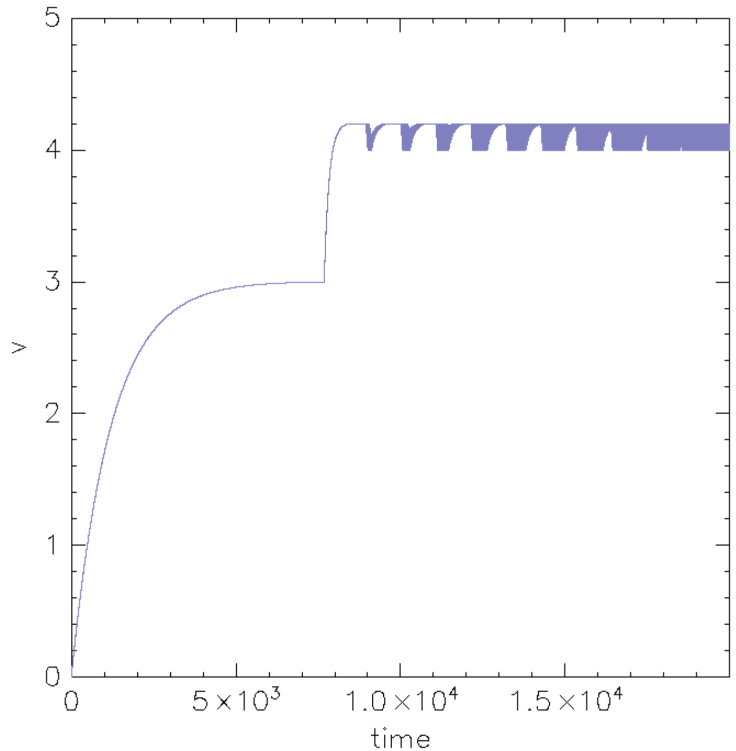
- **Battery Charger Behavioural (Functional) Model:**




**Output Voltage and Charging Current of a Typical Battery Charger in Different Charging Modes**




**State Transition Diagram of an Battery Charger**




**FlowPipe (Battery Voltage vs. Time) for the Battery Charger**

## Some of the time domain features of a battery charger are:

- Pre-charge current
- Time constant of the charging current
- Constant charge current
- Restoration time
- Time constant of the voltage response

# Features: Battery Charger

**Charge Time**

Time taken by the battery to rise from 10% in the precharge mode to the fully charged state.

```
feature chargeTime;
begin
   var t1,t2;
   (batt.state == PreCharge)  &&  ( @-(batt.V >= 0.1*Vterm) , t1 = $time )  ##[0:$]
   (batt.state == CV) && ( @+(batt.V ,Vterm) , t2 = $time )
       |-> chargeTime = t2 - t1;
end
```

---

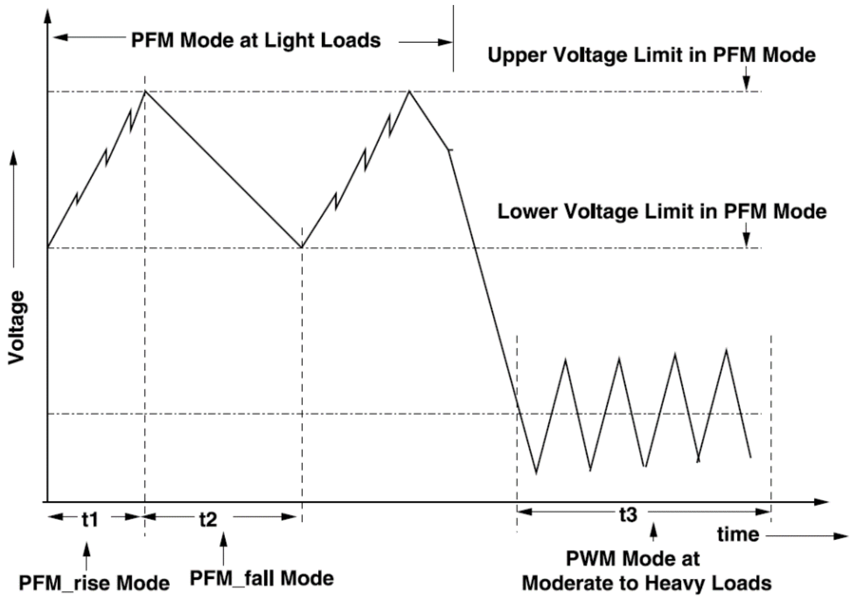
**Restoration Time**

Time taken by battery to restore back to constant voltage (CV) mode from maintenance mode.

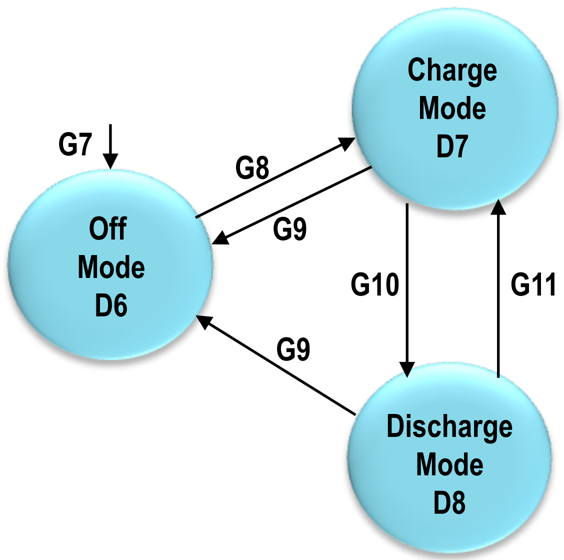
```
feature RestorationTime;
begin
    var t1,t2;
   (batt.state == Maintenance)  && ( @-(batt.V,Vrestart), t1 = $time ) ##[0:$]
   (batt.state==CV) && ( @+(batt.V ,Vterm),t2 = $time )
       |-> RestorationTime = t2 - t1;
end
```

# Case Studies and Results

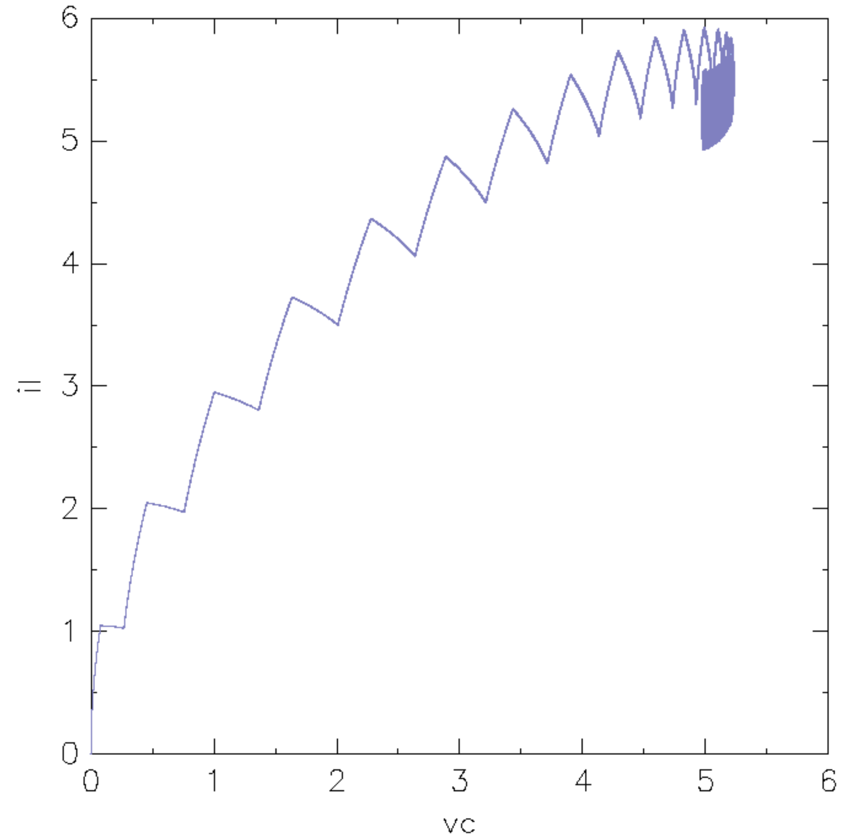
- **Buck Regulator Behavioural (Functional) Model:**




**Output Voltage of a Typical DC/DC Buck Regulator in Different Charging Modes**




**State Transition Diagram of PFM Operation of a Typical DC/DC Buck Regulator**




**FlowPipe (Current vs. Voltage) for the Buck Regulator**

**Some of the time domain features of a buck regulator are:**

- Peak Overshoot Voltage
- Settle Time
- Peak to Peak Output Voltage
- Switching Duty Cycle

# Features: Buck Regulator

**Settle Time**

Time taken for the output voltage to settle below E of the rated voltage, Vr, for two successive openings of the capacitor switch.

```
feature settleTime(Vr,E);
begin
    var t;
    (buck.v >= Vr+E)  ##[0:$]
        @+(buck.state == Open) && ( buck.v <=Vr+E), t = $time ##[0:$]
            @+(buck.state == Open) && ( buck.v <=Vr+E)
    |-> settleTime = t;
end
```

---

**Restoration Time**

The peak value of the voltage response curve measured from the desired response of the system.

```
feature overshoot(Vr);
begin
    var v1;
   (buck.state == Discharge)  && ( buck.v >= Vr), v1 = v
        |-> overshoot = v1;
end
```

# The numbers that count

| Model Name | Number of | | Analysis | |
|---|---|---|---|---|
| | Locations | Variables | Accuracy | CPU-Time |
| Buck Regulator | 2 | 4 | $10^{-6}$ | 1min 2sec |
| Battery Charger | 5 | 3 | 0.1 | 2.3 sec |
| Cruise Control | 6 | 4 | 0.1 | 0.8 sec |

## Results of Formal Feature Analysis

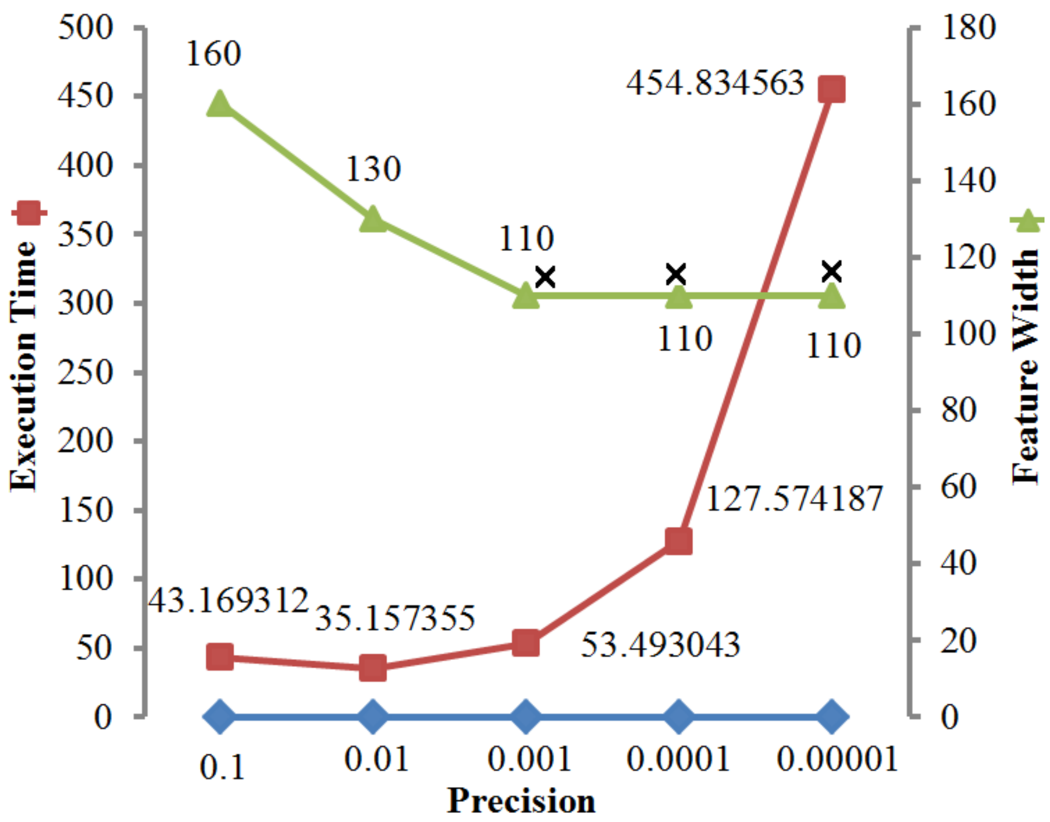
| Feature Name | Size of Set | | Step Size | CPU-Time (mins : secs) | Feature Range | |
|---|---|---|---|---|---|---|
| | $Q_F$ | $X_F$ | | | Min | Max |
| **Test Case: Buck Regulator** | | | | | | |
| Settle Time | 4 | 7 | $10^{-6}$ | 21m : 38s | 125.166$\mu s$ | 225.166$\mu s$ |
| | | | $10^{-3}$ | 0m : 40s | 125.166$\mu s$ | 225.166 $\mu s$ |
| Overshoot | 4 | 7 | $10^{-6}$ | 13m : 22s | 5V | 5.21V |
| | | | $10^{-3}$ | 0m : 7s | 5V | 5.21V |
| **Test Case: Battery Charger** | | | | | | |
| Charge Time | 7 | 7 | 1 | 0m : 30s | 1hr 24min | 4hr 34min |
| | | | 0.1 | 0m : 50s | 2hr 4min | 4hr 27min |
| Restoration Time | 7 | 7 | 1 | 1m : 26s | 5min 51sec | 12min 3sec |
| | | | 0.1 | 4m : 33s | 7min 35sec | 10min 2sec |
| Bandwidth | 7 | 7 | 1 | 0m :25s | 16.8$\mu$Hz | 202.5$\mu$Hz |
| | | | 0.1 | 0m : 56 s | 32.87$\mu$Hz | 65.85$\mu$Hz |
| **Test Case: Cruise Control Model** | | | | | | |
| Speed Capture Precise k=40 | 8 | 8 | 1 | 0m : 0.831 s | 37sec | 49sec |
| | | | 0.1 | 0m : 11.68s | 41sec | 44.8sec |
| | | | 0.01 | 3m : 51.13s | 41.44sec | 44.26sec |
| Speed Capture Range, k1=20, k2=40 | 8 | 8 | 1 | 0 m : 5.20s | 33sec | 49sec |
| | | | 0.1 | 3m : 4.28s | 35.3sec | 45.9sec |
| | | | 0.01 | 31m : 31s | 35.45sec | 45.41sec |

**A few key observations…**

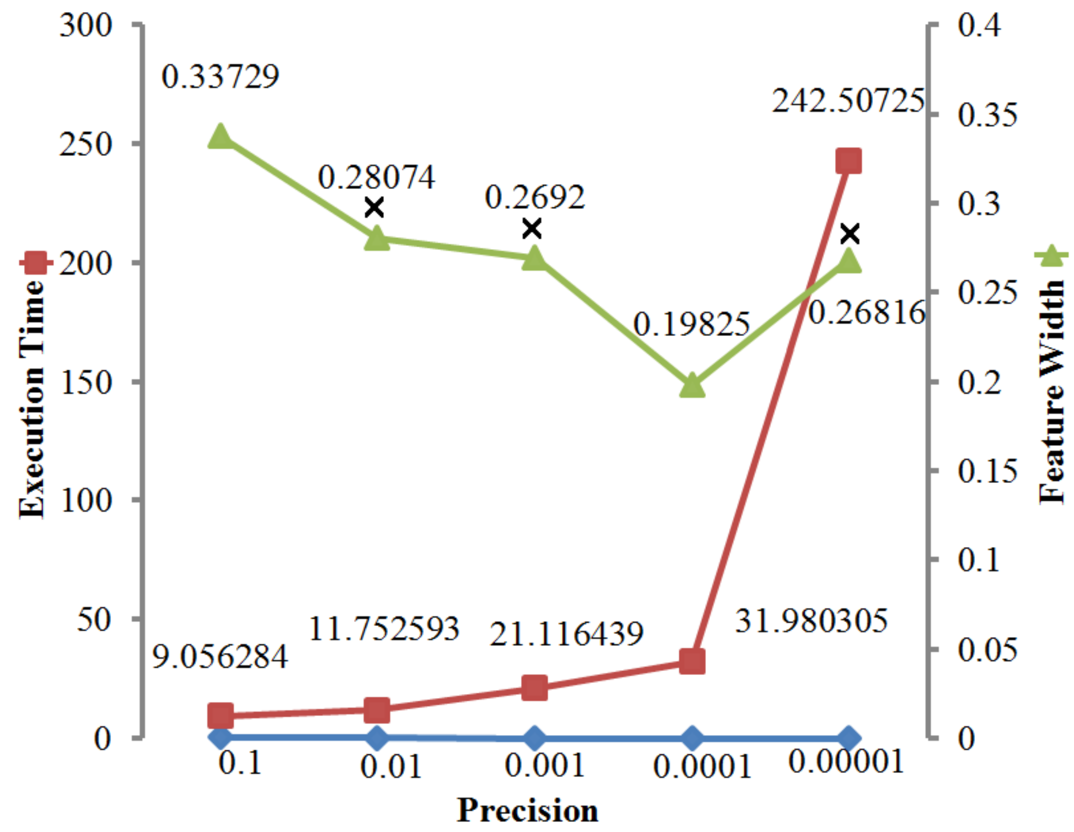
- The method of analysis scales well for various types of features.
- Computational accuracy beyond a point leads to insignificant improvements in the feature range computed.
- For quick analysis an appropriate *Step Size* may be decided upon.
- Unsatisfactory feature ranges require re-evaluation of models parameters, and fine-tuning of the design strategy.

# Observations

- Accuracy Saturation marked as x
- Precision α *(Feature Width)*$^{-1}$ α Execution Time



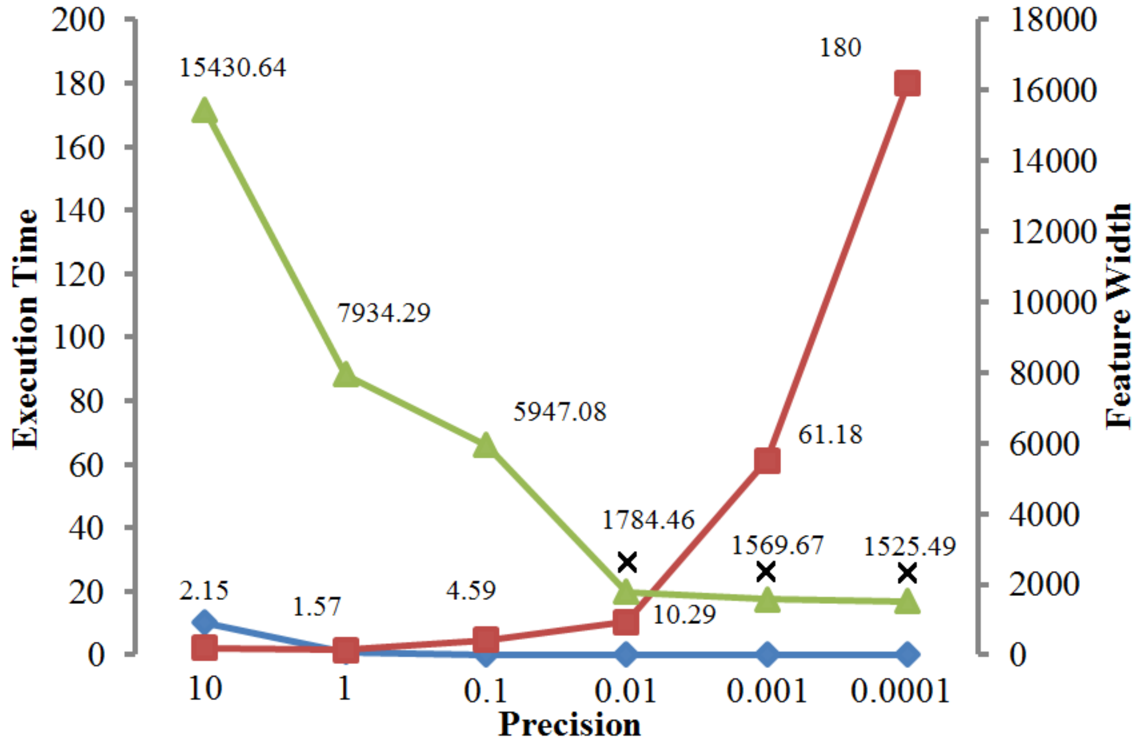

Buck Regulator: **settleTime**
([135 : 245]*μs*)



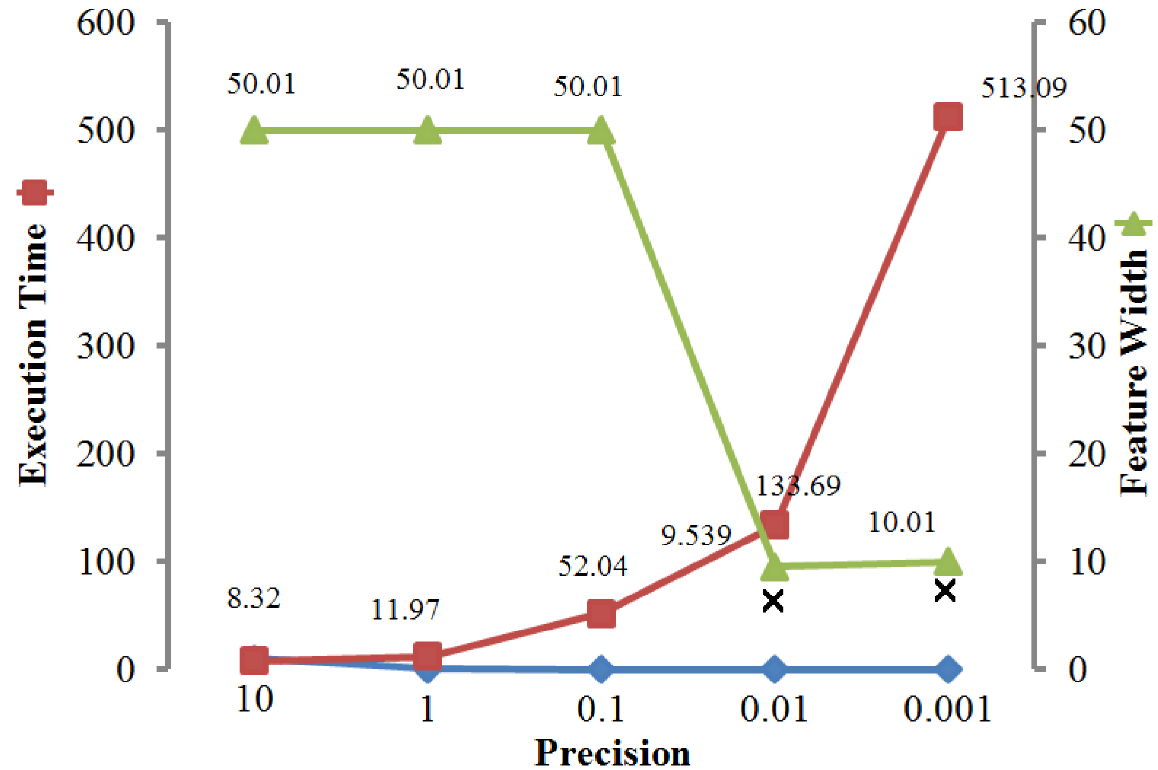

Buck Regulator: **overshoot**
([6.26 : 6.46]*V* )

# Observations Contd.

- Accuracy Saturation marked as x
- Precision α *(Feature Width)*$^{-1}$ α Execution Time




Battery Charger: **chargeTime** ([7562.75 : 9088.24]*s*)




Nuclear Reactor: **unsafe** ([590 : 600])

# The Road Forward…

- Building a cool GUI for ForFET.
  - Hybrid Automaton model browser and editor
  - Browser, Editor and Syntax Highlighter for ***features***
  - One click feature analysis

- Feature Range [ $F_{min}$ , $F_{max}$ ] extreme value analysis
  - Can we tell which action choices and timings lead to $F_{min}$ , $F_{max}$
  - Using SMT solvers dReach/dReal for δ-satisfiability to generate traces for $F_{min}$ , $F_{max}$

- What about simulatable models? Can we learn features from simulation traces?
  - Learning AMS assertions from simulation traces.

- A specialized Feature Analysis Engine

# Thanks for listening!

## Any Questions?