Dynamic Time Step Control Algorithm Enhancements

James C. (Jim) Bach

Sr. Project Engineer
Electrical Analysis & Simulation Group
Delphi Delco Electronics Systems
Kokomo, IN
James.C.Bach@DelphiAuto.com

DELPHI
Automotive Systems

Introduction

- Analog simulators typically utilize a scheme known as “Dynamic Time Step” control (aka DTS)
  - Provide adequately spaced time steps during *active* portions of simulation (signal edges, oscillations)
    - Tightens-up time steps when matrix solver has difficulty converging on a solution (i.e. lots of NR iterations to converge)
  - Reduce the amount of calculations performed during *inactive* portions of simulation (stable signals)
    - Loosens-up time steps when matrix-solver finds it easy to converge on a solution (i.e. few NR iterations to converge)
- In general, DTS does a good job, *most* of the time
- The techniques presented here can help overcome some troublesome situations
  - These are only a FEW of the “helpers” I’ve created

Introduction (cont.)

- Am I “picking-on” Saber?
  - No
    - I am attempting to point-out to the developers (and users) of AHDL (aka AMS) simulators that this has been an issue in the past, and needs to be addressed in anything they create for the future
    - Saber’s MAST language is the predecessor of VHDL-AMS
      - I am afraid that these deficiencies will become a part of the Next-Generation (aka AMS) simulators
    - I want to offer solutions to AHDL (AMS) users to problems they might encounter

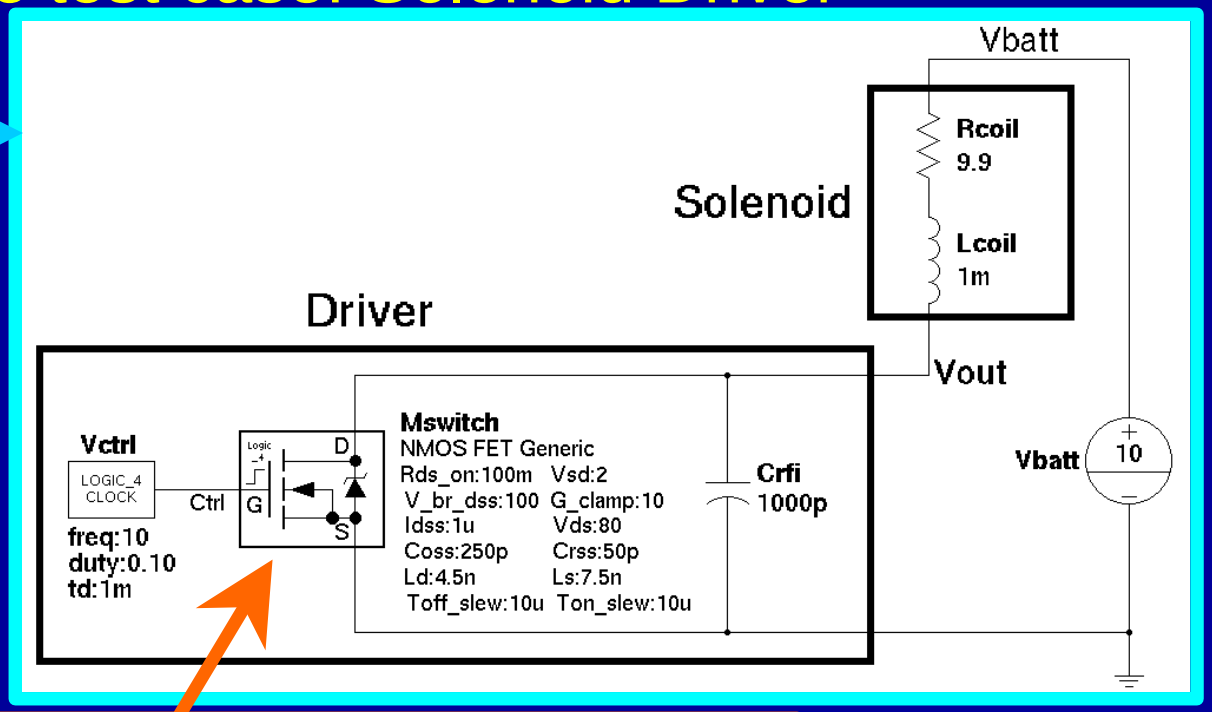
The Problem

- DTS algorithms can perform inadequately
  - Loosen-up time steps too quickly
    - Waveforms appear “jagged” or “steppy”
    - Oscillations decay-away too abruptly
  - Generate insufficient time steps to produce “smooth” waveforms
    - Oscillations appear “segmented”
- An *inaccurate* answer, obtained quickly, is often less useful (or even more harmful) than *no* answer at all
  - Can validate false assumptions
  - Can lead to invalid conclusions
  - Can cause more “design turns” (wasted time)
  - Can give false optimism that circuit/system works OK
  - Can give false pessimism that circuit/system won’t work

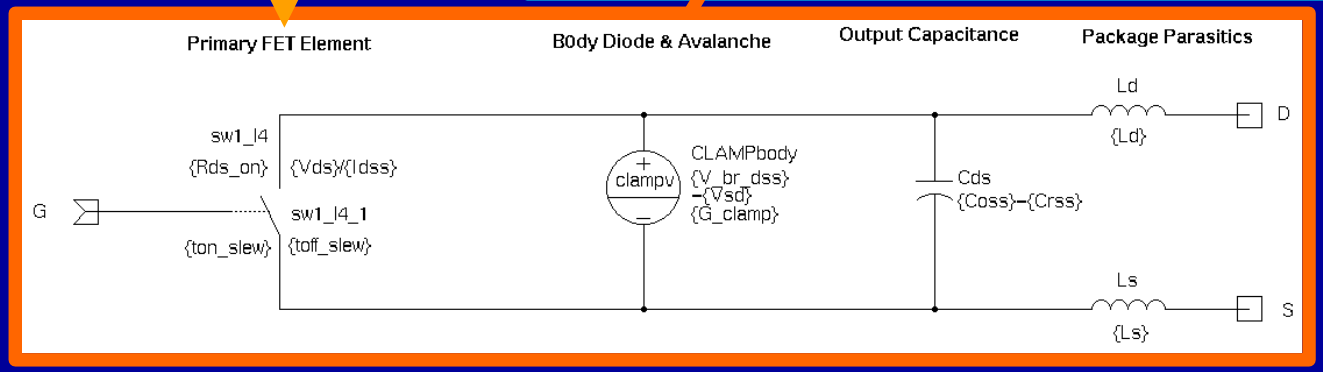
The Problem (cont.)

- Given a simple test case: Solenoid Driver

Top-Level Circuit



Simple FET Macro Model



DELPHI Automotive Systems

The Problem (cont.)

- The circuit should have the following characteristics:
  - Natural Resonance Frequency (NRF)

$$F_{Res} = \frac{1}{2p\sqrt{L_{Coil}(C_{RFI} + C_{DS})}} \approx 153.2\text{kHz}$$

  - Decay (or Damping) Time-Constant

$$t_{Decay} = 2\frac{L_{Coil}}{R_{Coil}} \approx 200\text{ms}$$

  - Time to decay to ½ of peak amplitude

$$T_{\frac{1}{2}\text{Amplitude}} = 0.69315 * t_{Decay} \approx 138.6\text{ms}$$

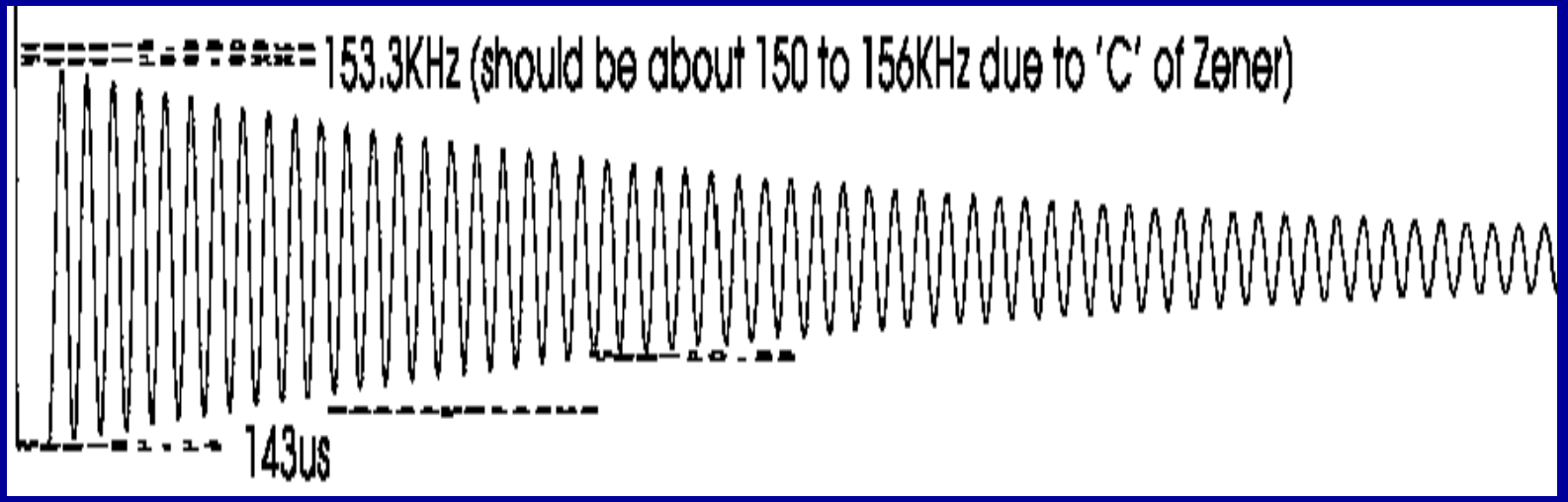
  - Apparent Resonance Frequency

$$F_{Osc} = \sqrt{F_{Res}{}^2 - \left(\frac{1}{2p * t_{Decay}}\right)^2} \approx \sqrt{153.2\text{kHz}^2 - 795.8\text{Hz}^2}$$

$$\approx 153.197\text{kHz}$$

The Problem (cont.)

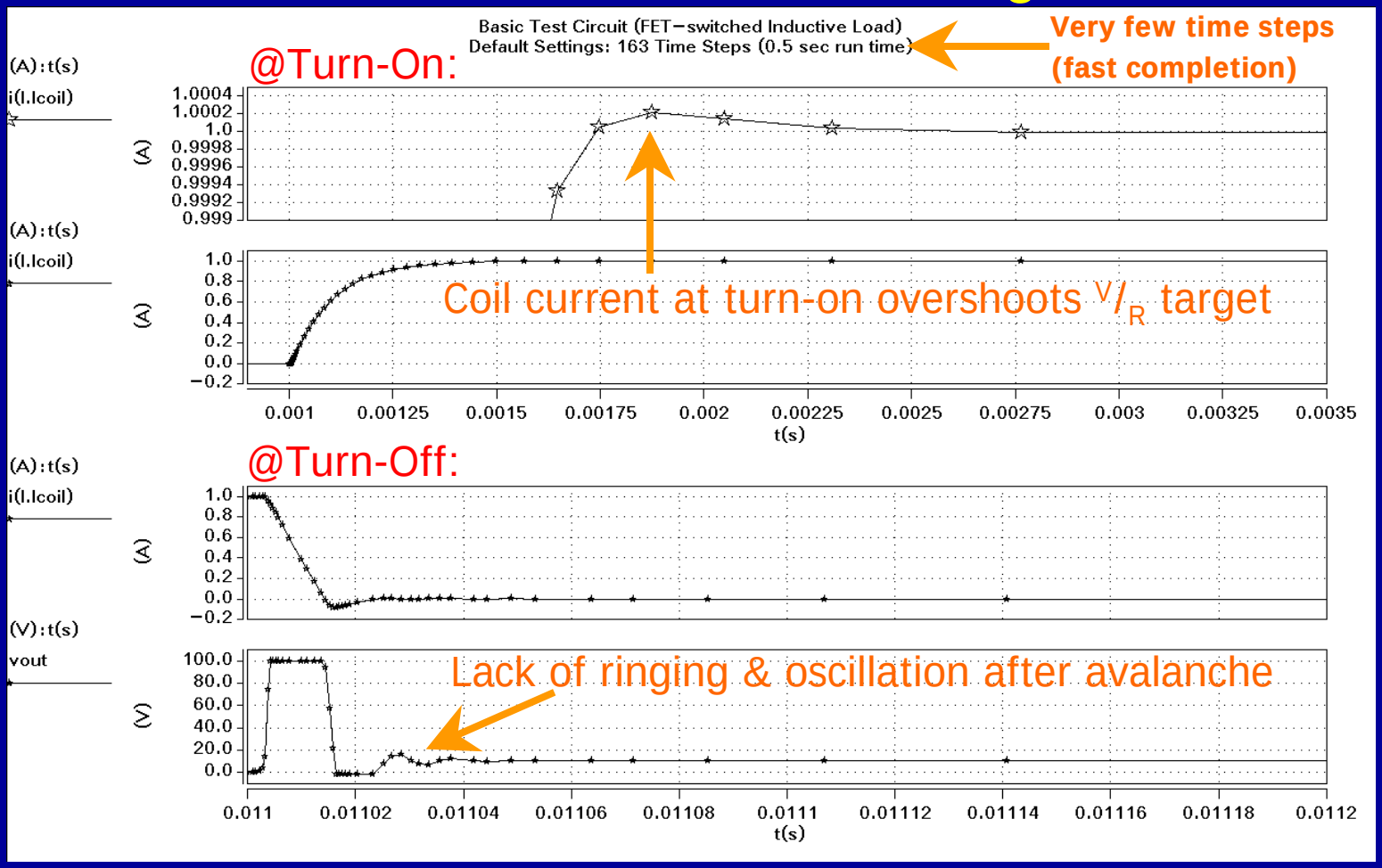
- The text book answer:
  - Post-avalanche ringing



  - as produced using PSPICE (without effort)
  - Saber requires considerable “adjusting” to obtain

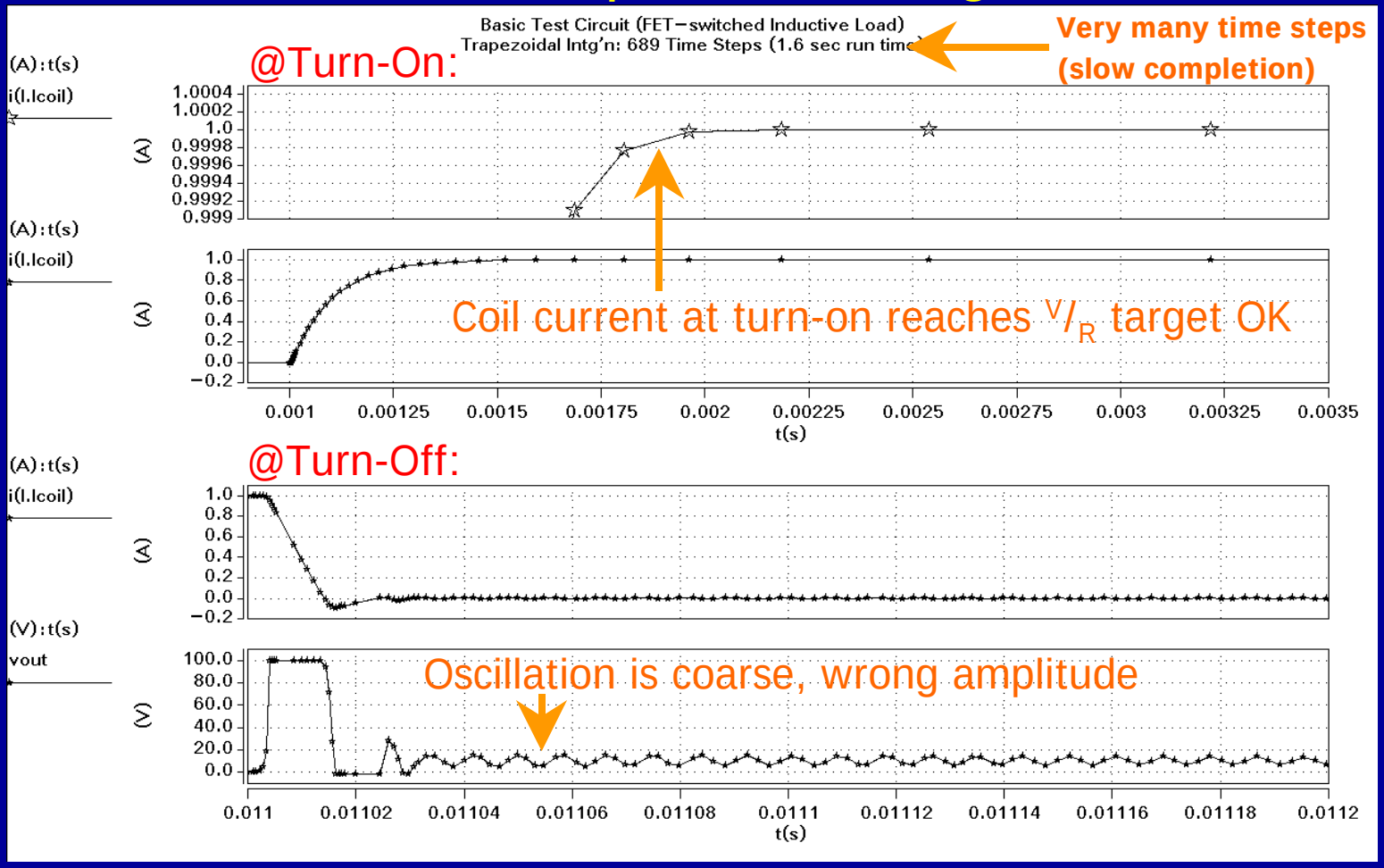
The Problem (cont.)

- Saber results with default DTS settings:



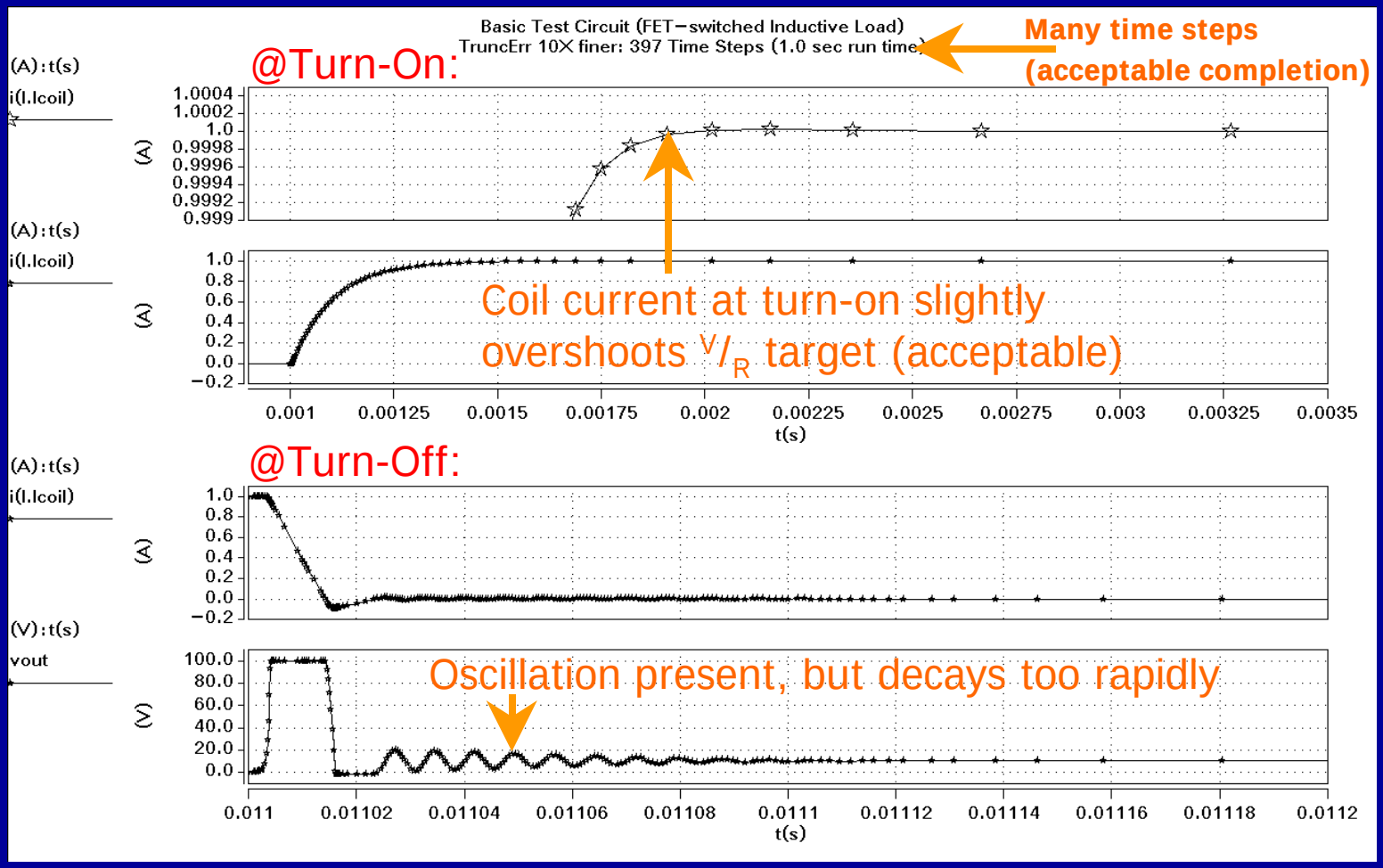
The Problem (cont.)

- Saber results with Trapezoidal Integration:



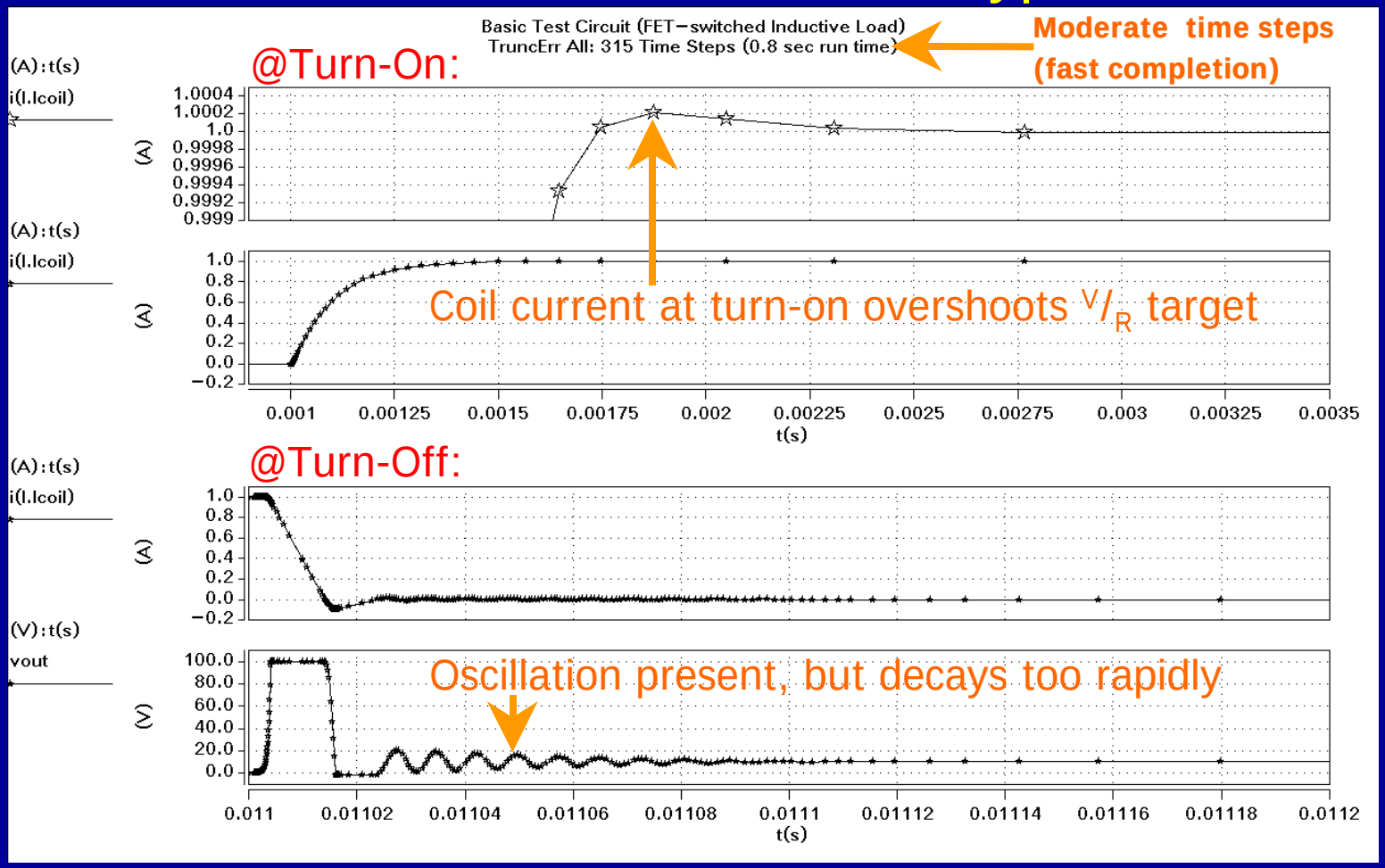
The Problem (cont.)

- Saber results with 10X finer Truncation Error:



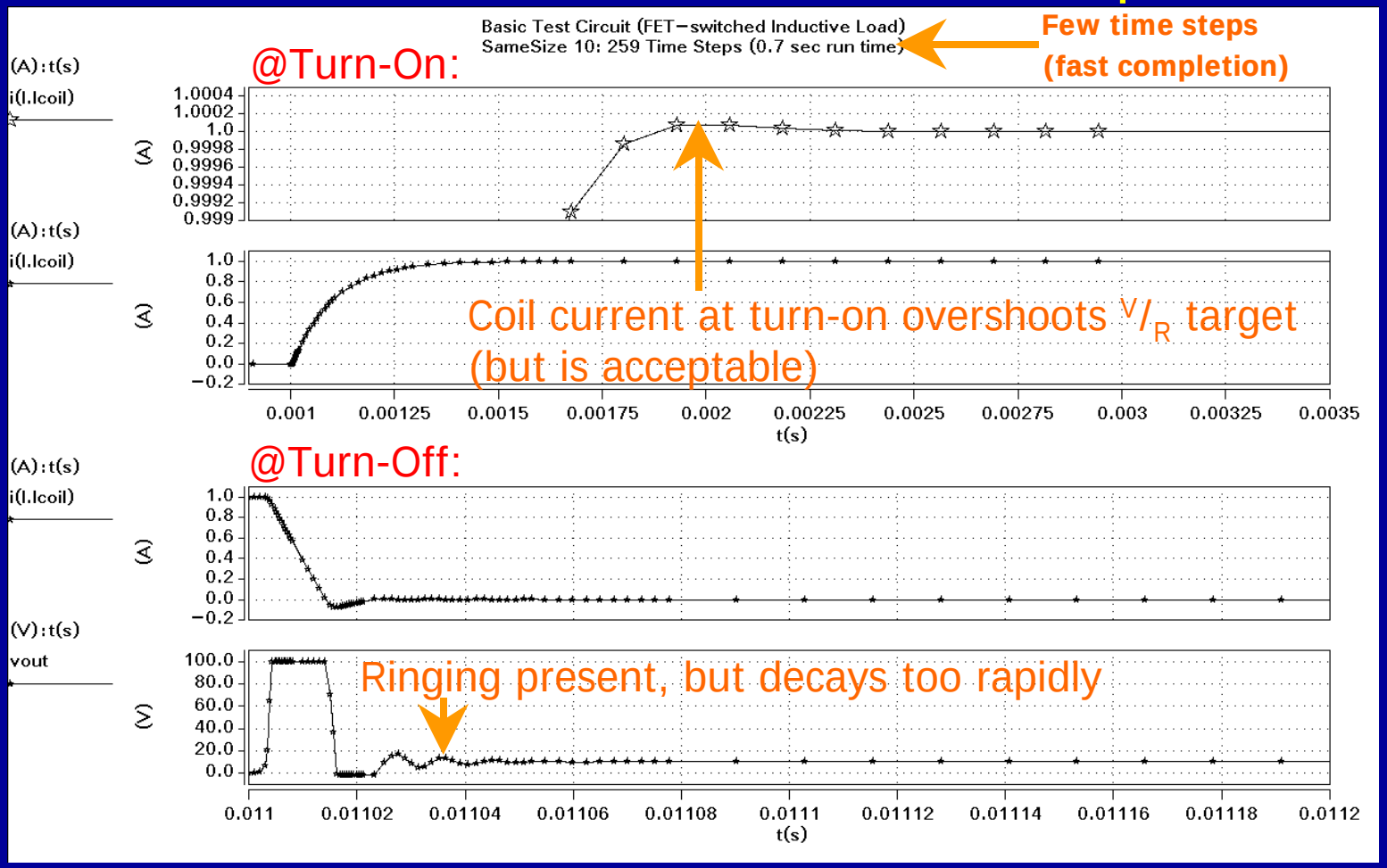
The Problem (cont.)

- Saber results with “All” Truncation Type:



The Problem (cont.)

- Saber results with 10 same-size time steps:



The Problem (cont.)

- Considering four major and two minor control parameters:

Table 1 – Saber DTS Control Settings

Parameter	# of Settings	Settings
Truncation Error	3	Default ± 1 decade (0.05, 0.005, 0.0005)
Truncation Type	3	Method to use (Dynamic, Static, All)
Truncation Normalization	6	Formula to use (1, 2, 3, 4, 5, 6)
Integration Method	2	Method to use (Gear, Trapezoid)
Number of Same Points	3	Default + 2 decades (1, 10, 100)
Sample Point Density	3	Default + 2 decades (1, 10, 100)
	972	**VERY many combinations**

Major Controls

Minor Controls

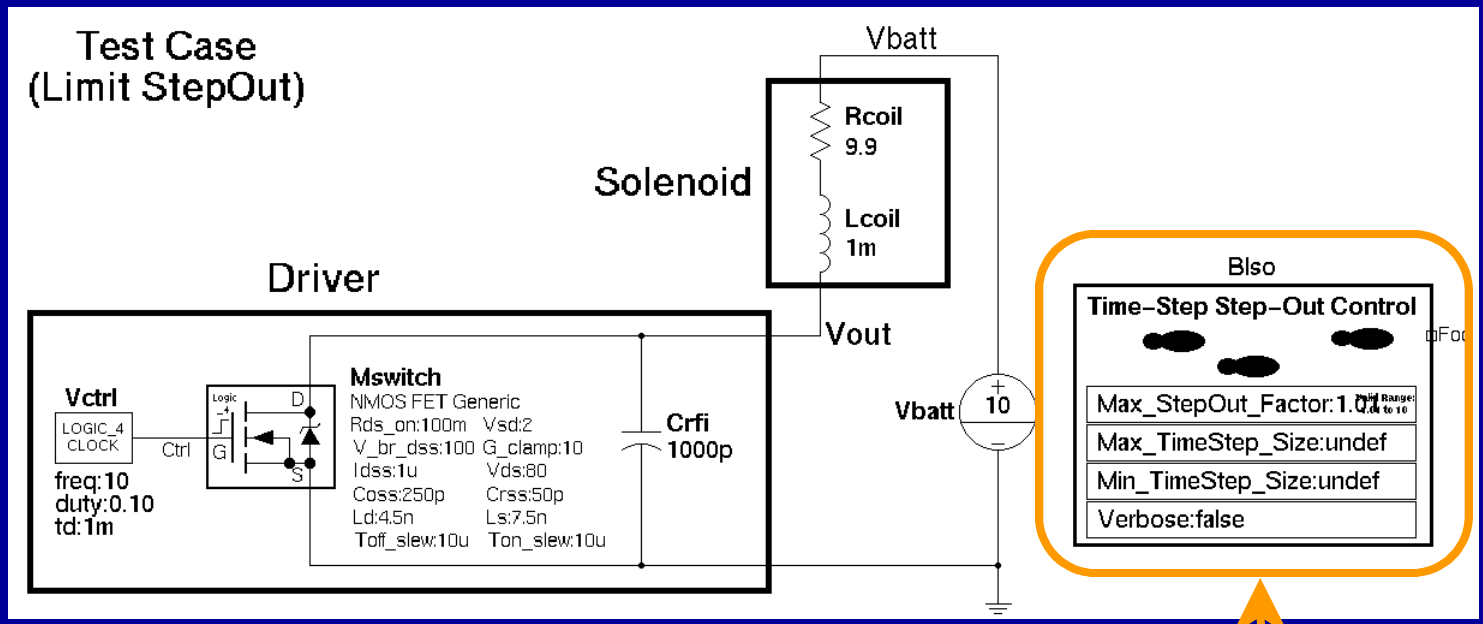
- "Calibrating the simulator" can be a very daunting task: there are TOO MANY combinations to try!

Solution #1: Limit Step-Out

- Limit Step-Out (LSO) performs these tasks:
  - At the end of each time step, the template calculates the size of the current step (Time_Step) by subtracting the current time (*time*) from the time it was when the last time step was finished (Last_Time).
  - The current time step size is multiplied by the user-specified relaxation rate (StepOut_Factor) in order to determine how large the next time step can be (Desired_Step_Size).
  - The DTS is constrained (via *step_size* system variable) as to how large it can make the next time step (Desired_Step_Size).

Solution #1: Limit Step-Out (cont.)

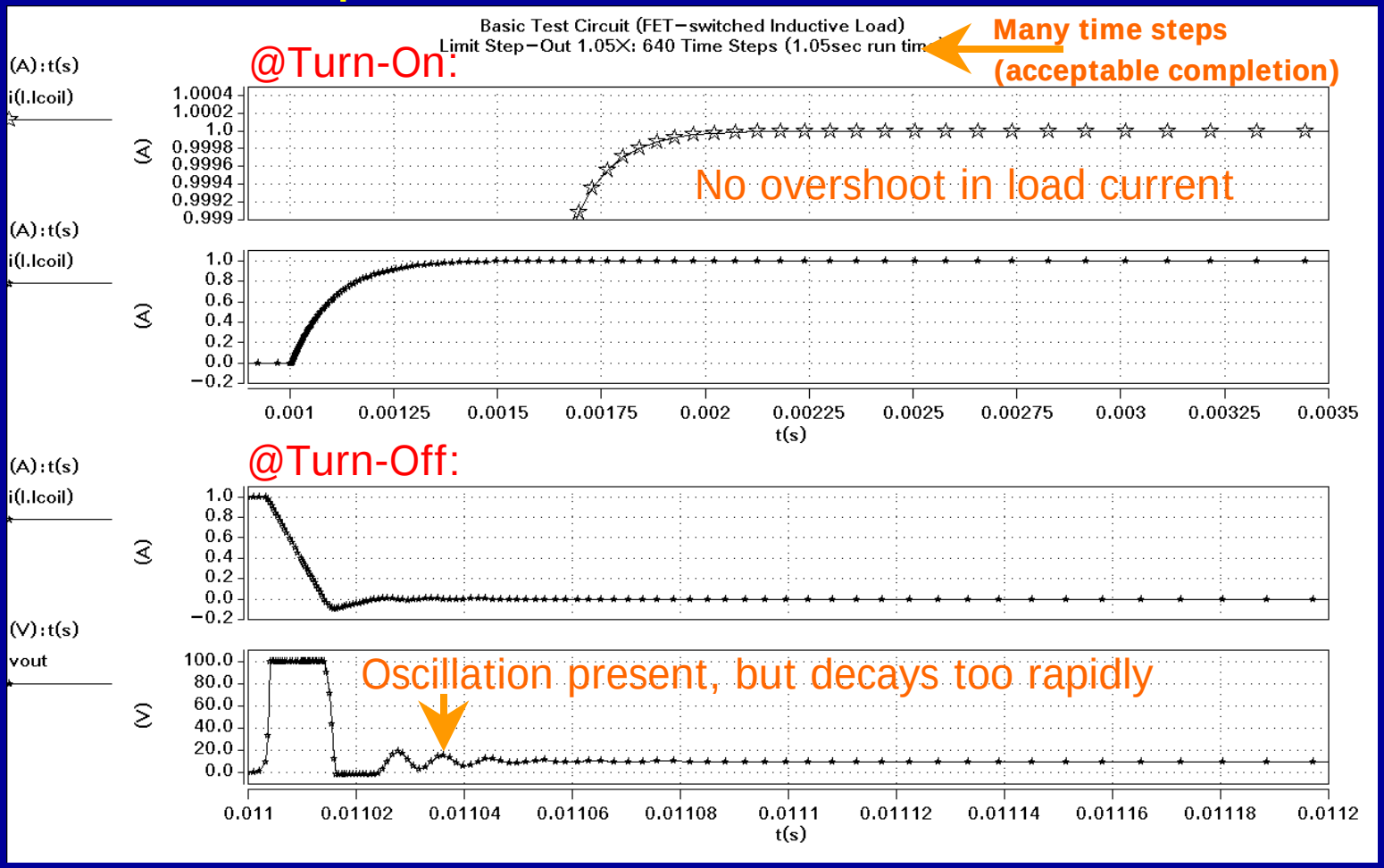
- LSO added to the original circuit:



- Simply place on top-level schematic
  - Needs no “connection” to circuit
  - Provides three user-adjustable (optional) parameters

Solution #1: Limit Step-Out (cont.)

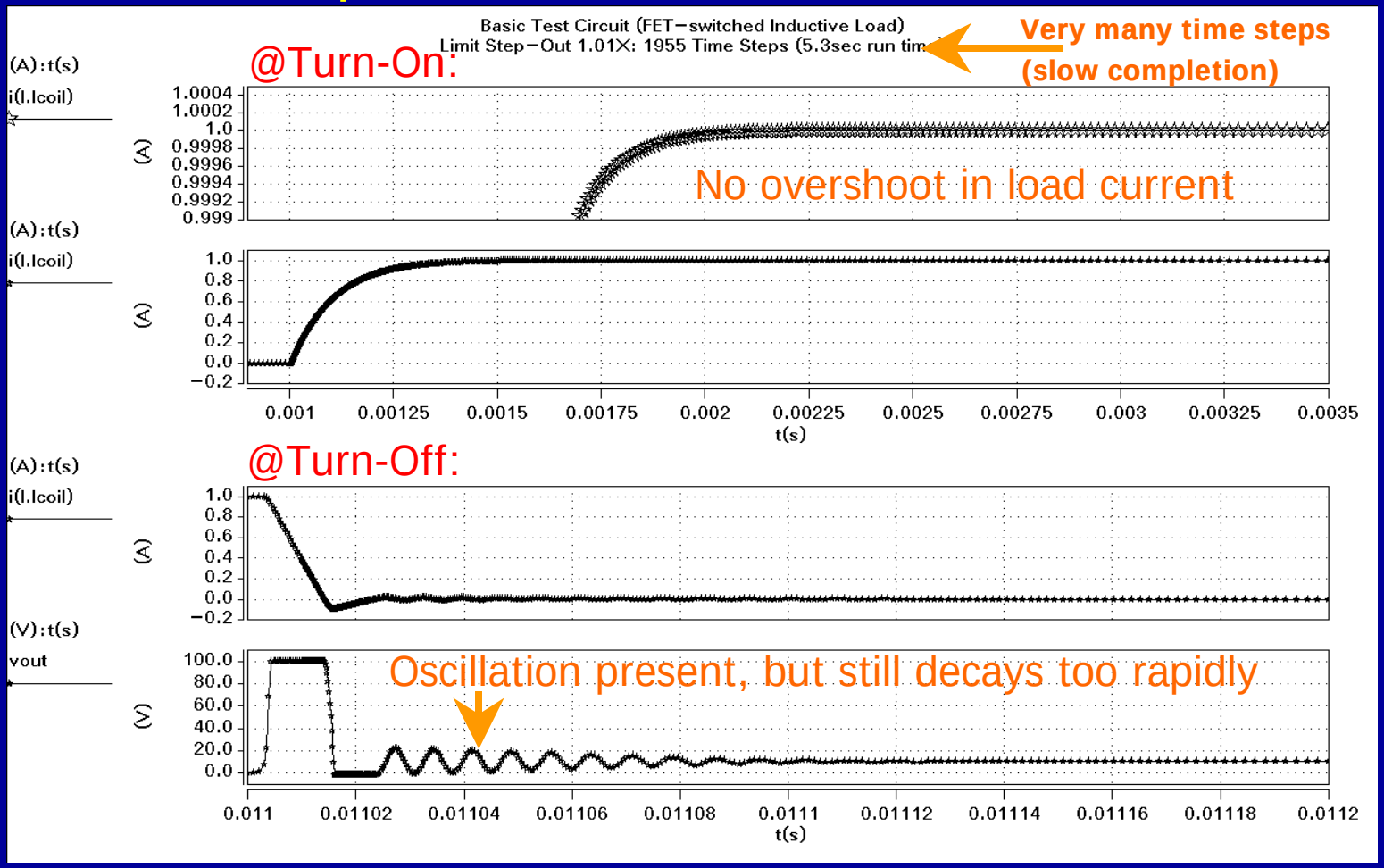
- Results: Step-Out Factor=1.05



- All other DTS parameters set to “Defaults”

Solution #1: Limit Step-Out (cont.)

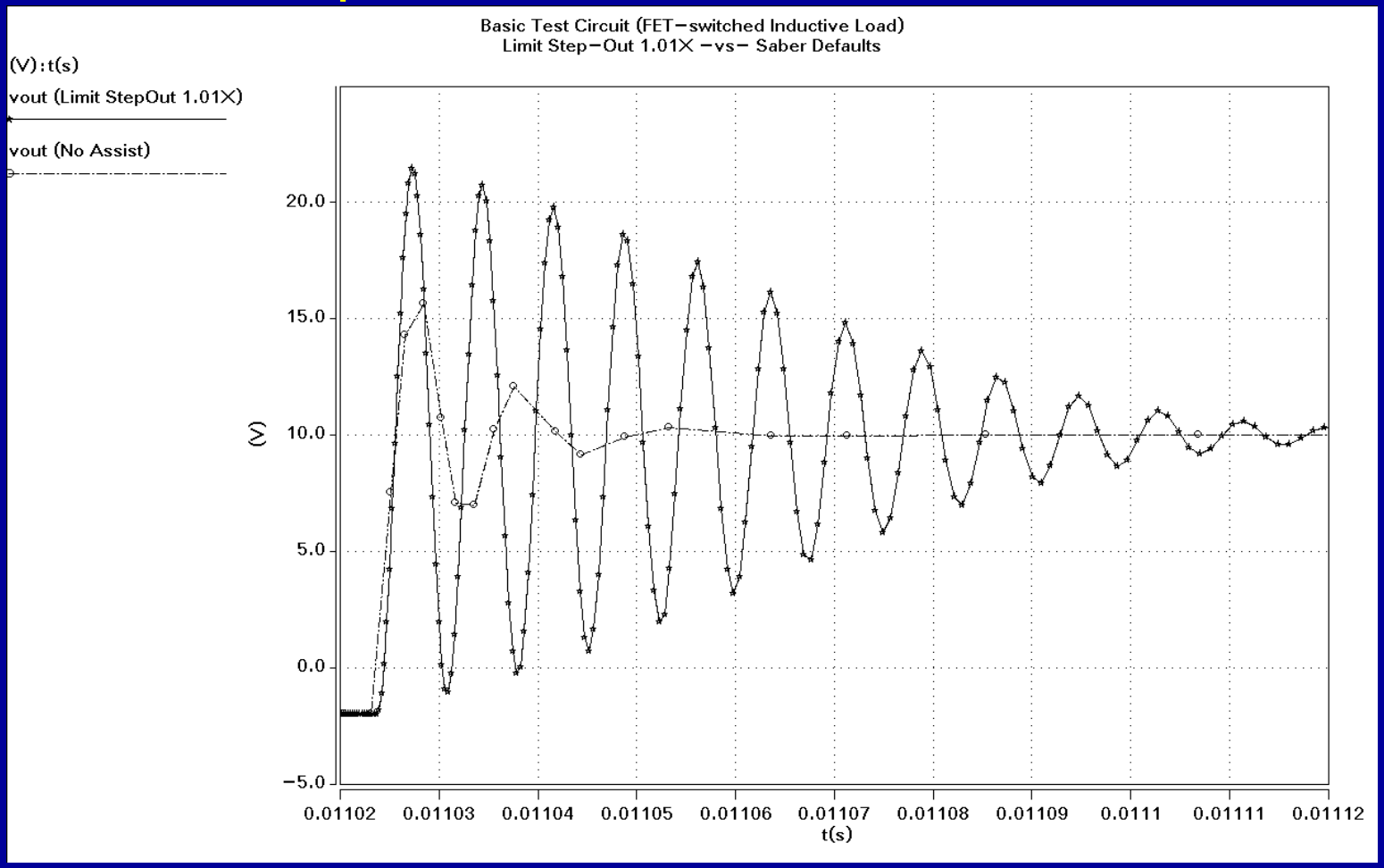
- Results: Step-Out Factor=1.01



- All other DTS parameters set to “Defaults”

Solution #1: Limit Step-Out (cont.)

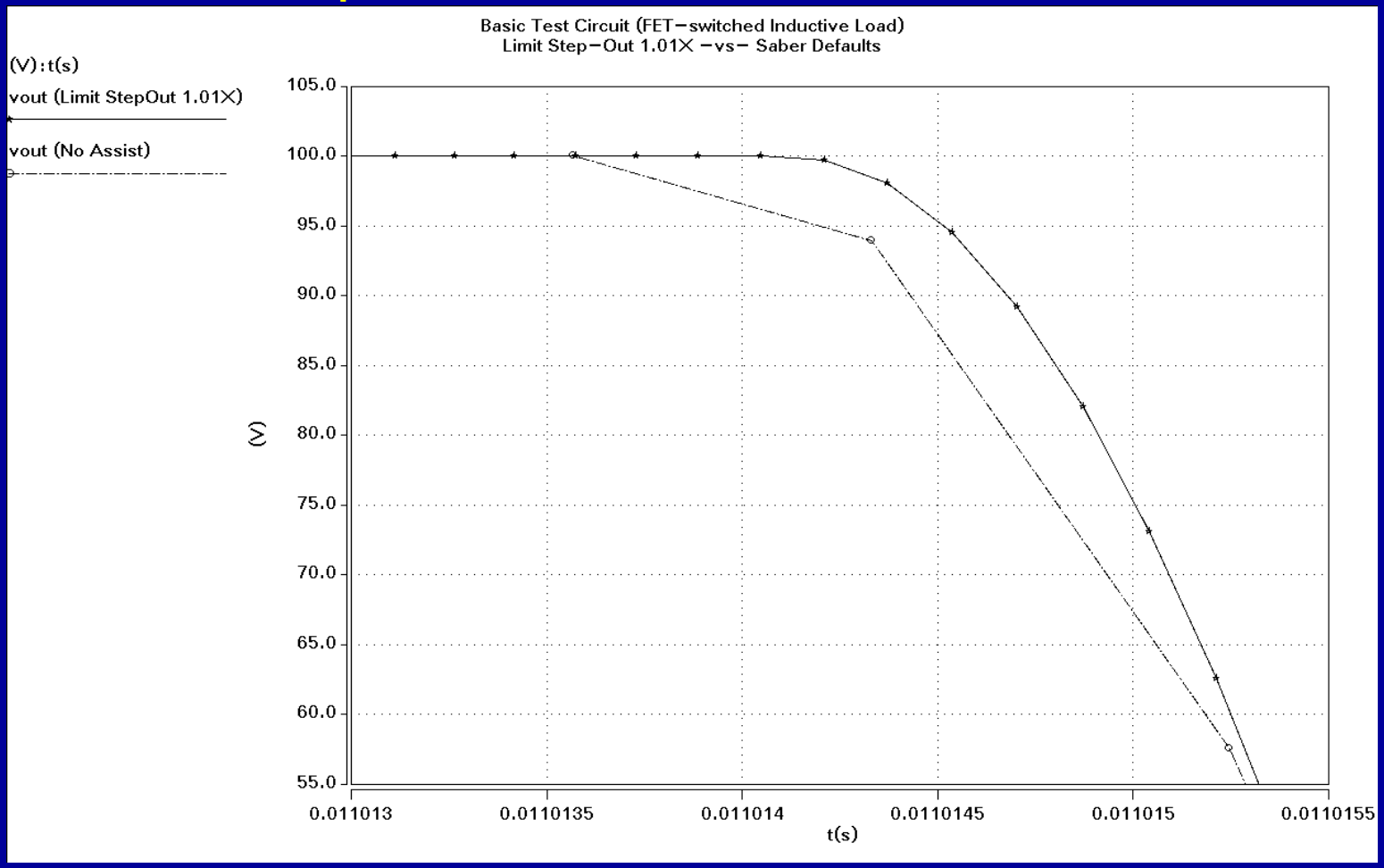
- Results: Step-Out Factor=1.01 *-vs-* No Assistance



- Post avalanche ringing

Solution #1: Limit Step-Out (cont.)

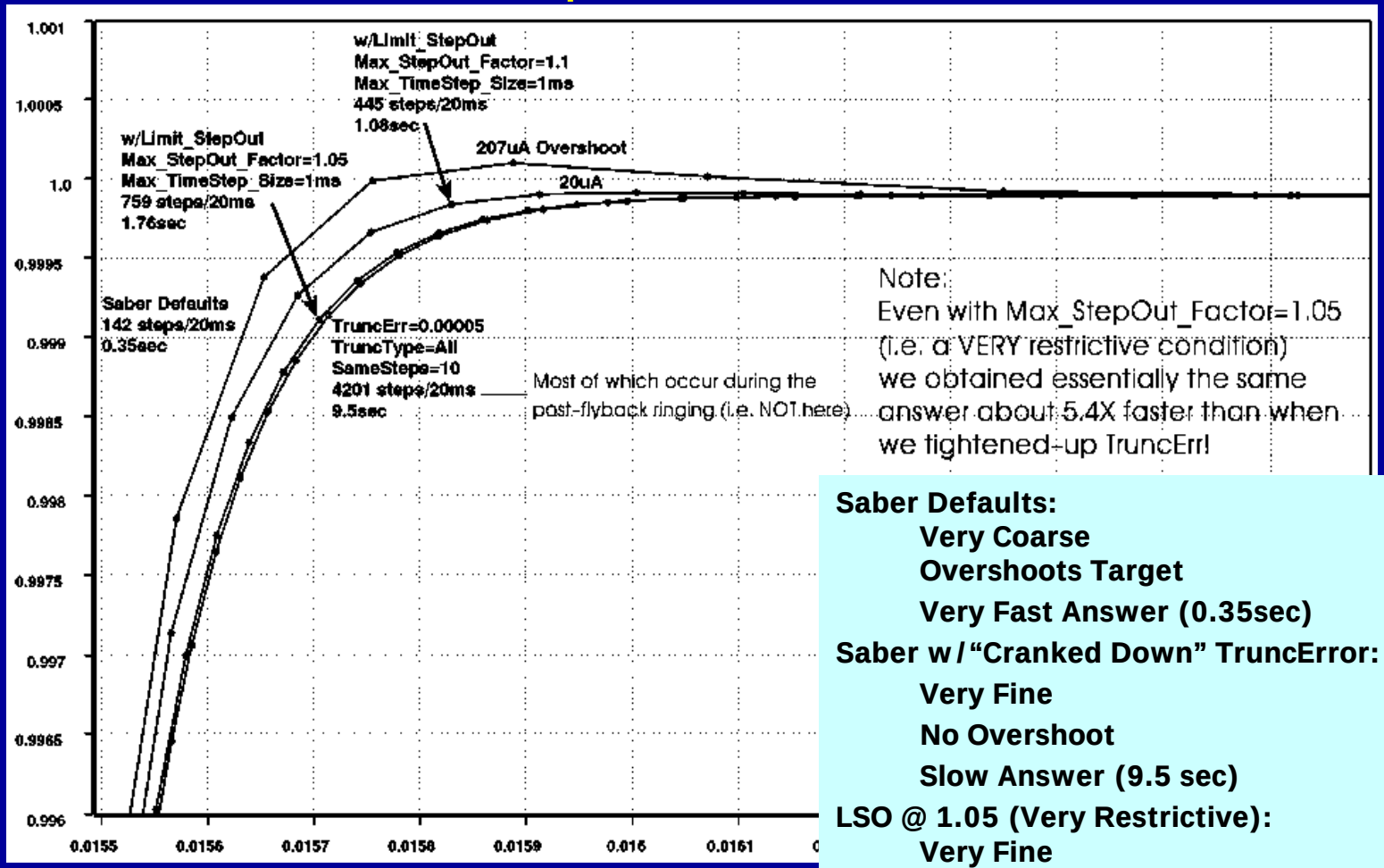
- Results: Step-Out Factor=1.01 -*vs*- No Assistance



- End of avalanche interval

Solution #1: Limit Step-Out (cont.)

- Results: Various Step-Out Factors



- Current build-up at turn-on

Saber Defaults:
- **Very Coarse**
- **Overshoots Target**
- **Very Fast Answer (0.35sec)**

Saber w/"Cranked Down" TruncError:
- **Very Fine**
- **No Overshoot**
- **Slow Answer (9.5 sec)**

LSO @ 1.05 (Very Restrictive):
- **Very Fine**
- **No Overshoot**
- **Fast Answer (1.76sec)**

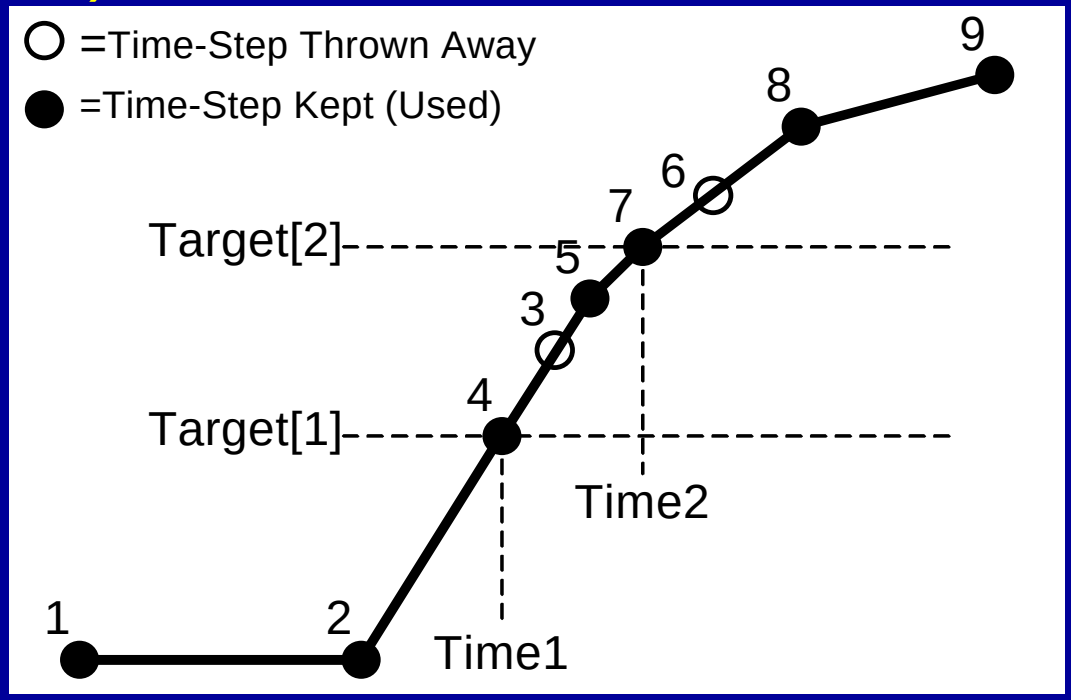
DELPHI Automotive Systems

Solution #2: Target Crossing

- Target Crossing (TC) operates as follows:
  - The user specifies up to ten voltage levels at which time steps should occur (Targets).
  - The voltage between the inputs (P and M) are monitored (using *when(threshold())* construct).
  - When any of the ten thresholds are crossed, an analog time step is forced (using the *schedule_next_time* construct) at the point in time (*time*) which the built-in linear interpolation routine estimated the threshold was actually crossed.
  - The analog simulator throws away the data for the just-calculated time step and goes back to the specified (interpolated) point in time.

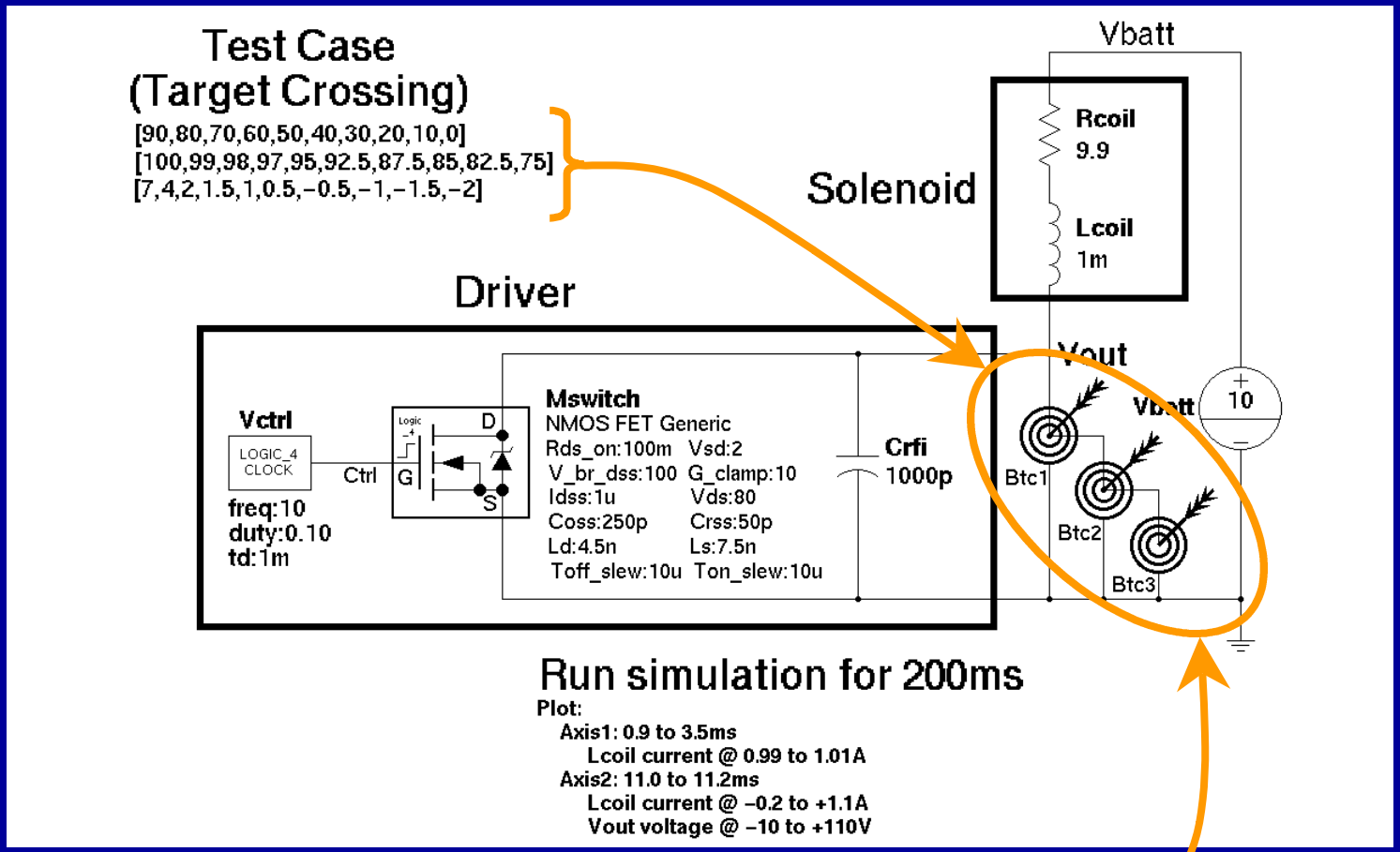
Solution #2: Target Crossing (cont.)

- Target Crossing (TC) illustrated:
  - Time step #3 is calculated, but signal crossed Target[1] level.
  - Estimated time of crossing is Time1, so DTS forced back for time step #4.
  - Time step #5 OK.
  - Time step #6 causes signal to pass through Target[2], so cross time is estimated to be Time2, and time step #7 is forced to occur there.
  - Time step #8 OK, so simulation continues.



Solution #2: Target Crossing (cont.)

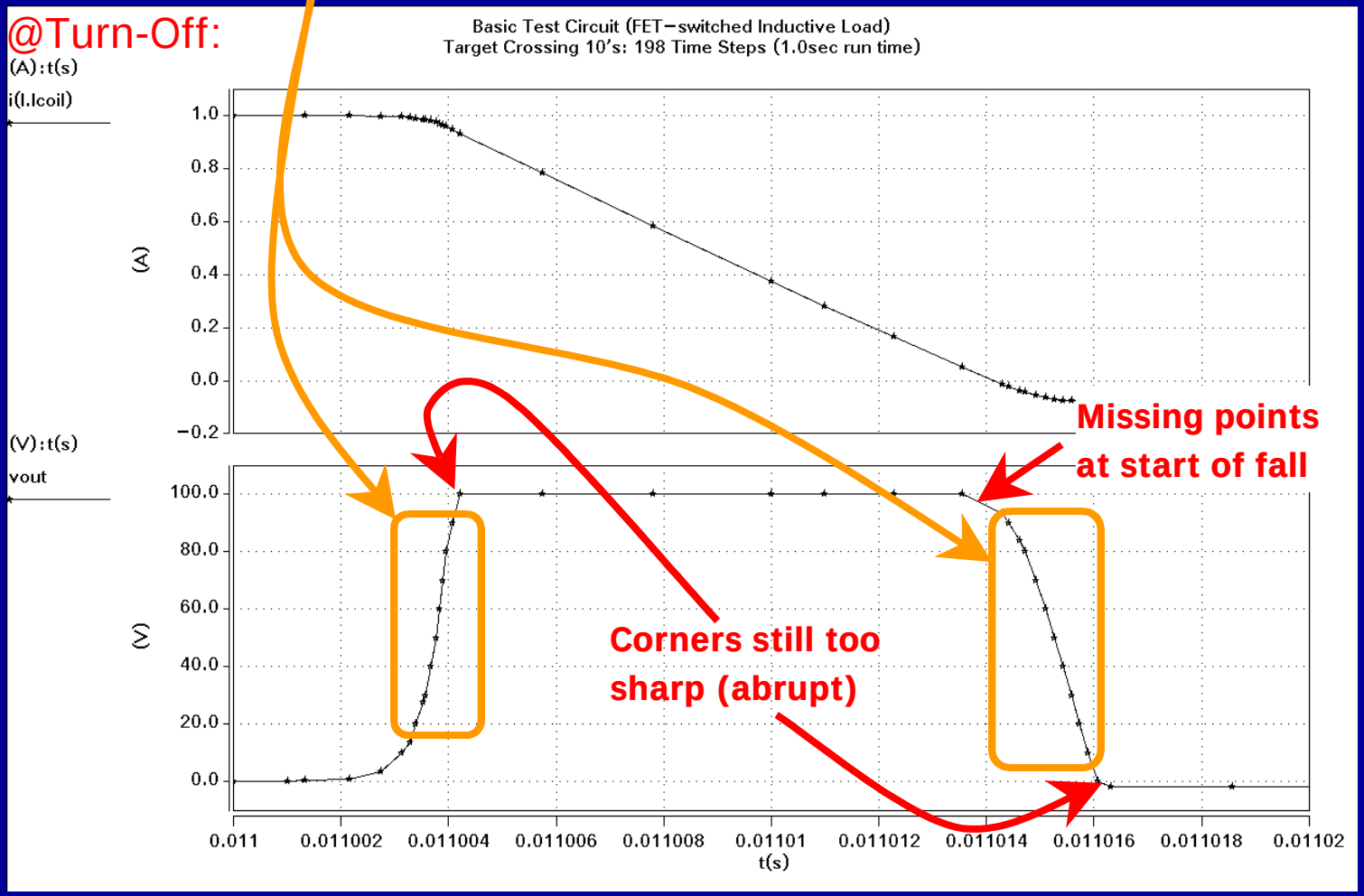
- Target Crossings (TCs) added to schematic:



3 TC blocks added: 30 target voltages

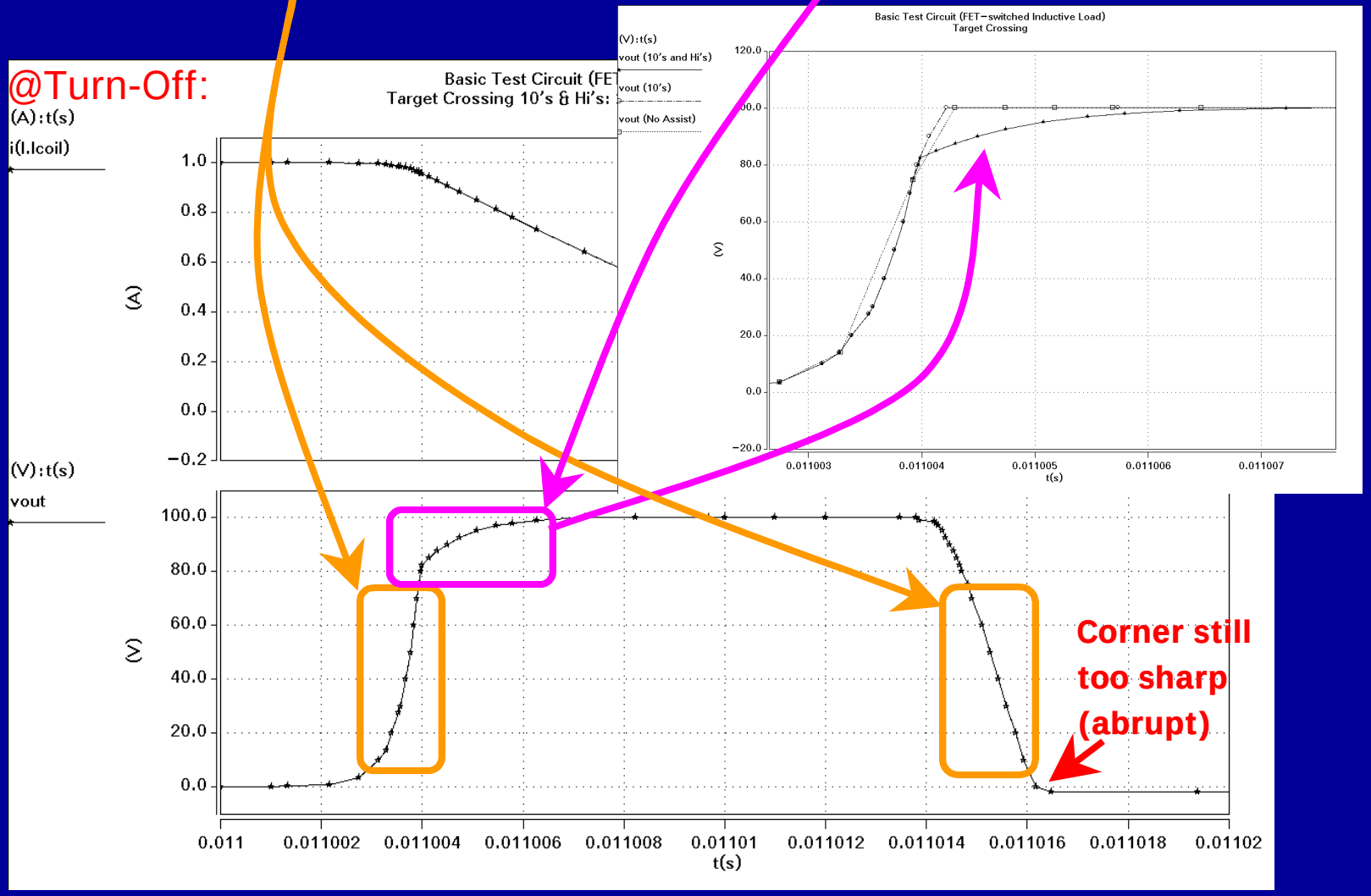
Solution #2: Target Crossing (cont.)

- TCs at 10V intervals (during avalanche):



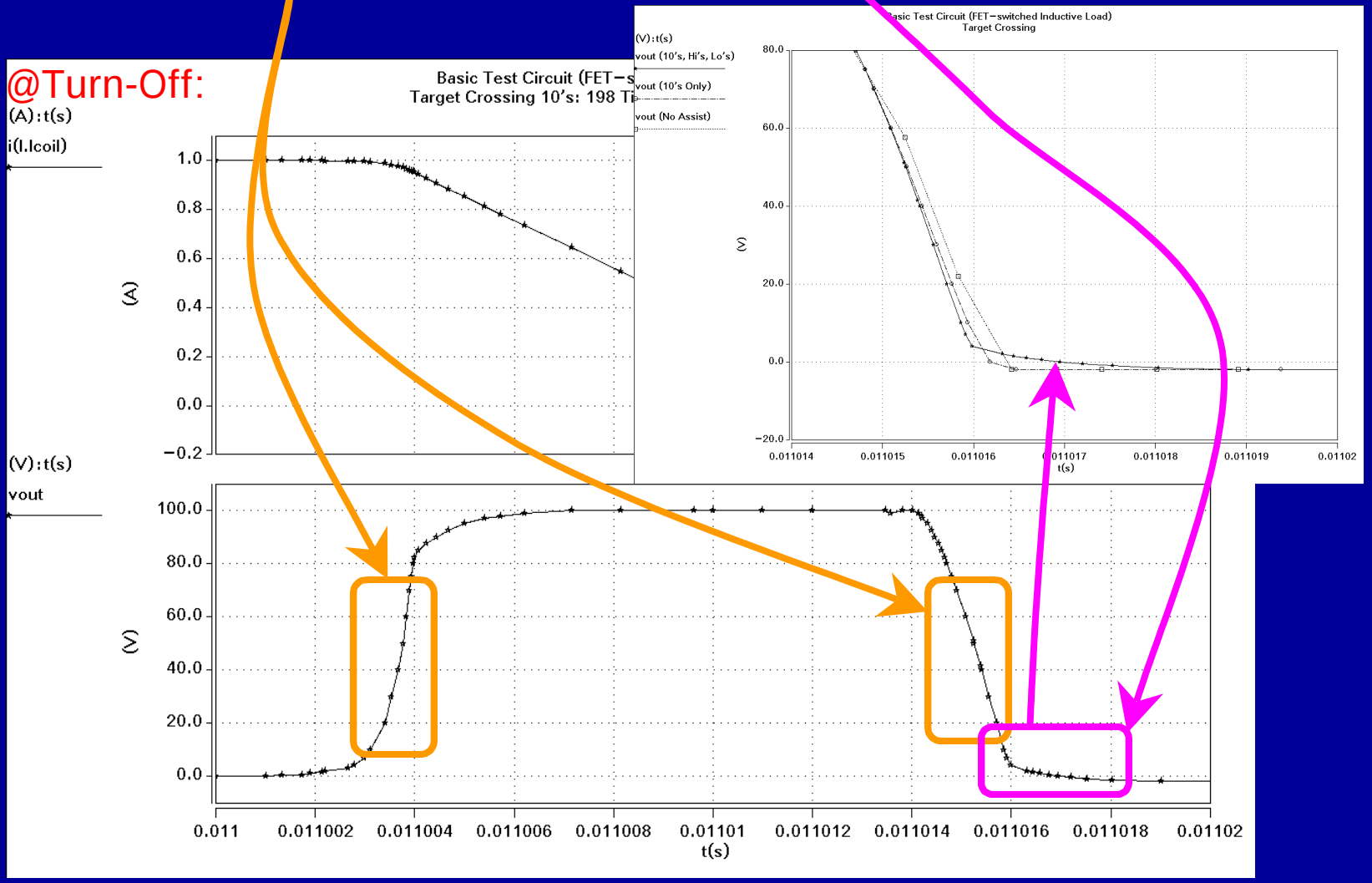
Solution #2: Target Crossing (cont.)

- TCs at 10V intervals overall & finer at upper corners:



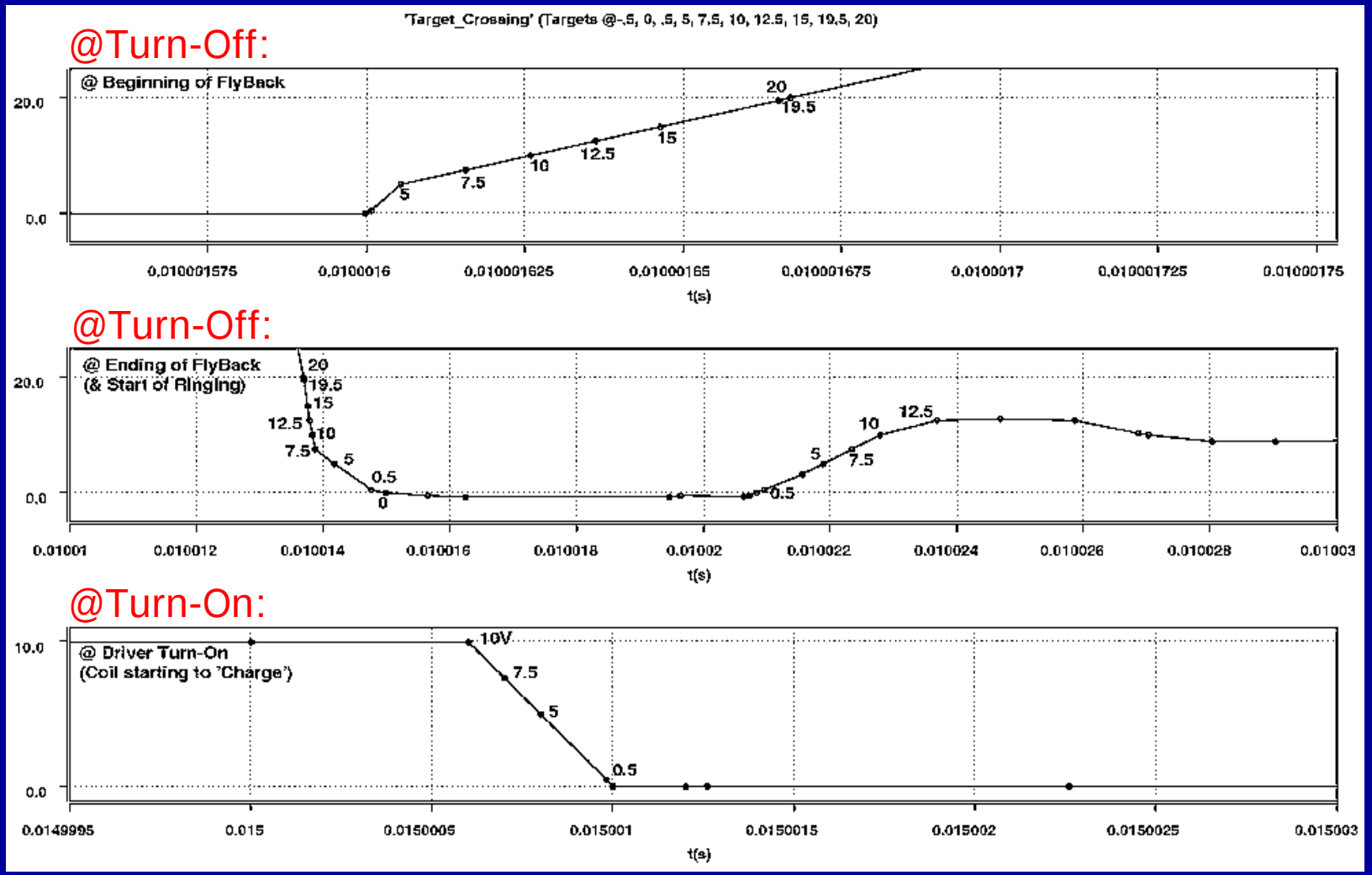
Solution #2: Target Crossing (cont.)

- TCs at 10V intervals overall & 1V at bottom corners:



Solution #2: Target Crossing (cont.)

- TCs at key levels from GND to V_{SUPPLY}:



Conclusion

- Using AHDL coding, it is possible to enhance the generation/control of time steps so that more calculations are performed during active intervals.
- More calculations in active intervals leads to more accurate rendition of waveforms
  - Oscillations become apparent
  - Rising/Falling edges don't erroneously overshoot target
  - Rising/Falling edges approach target smoothly
- All AHDL (AMS) languages should provide mechanisms for allowing the model-writer to manipulate time steps from within device models (as Saber's MAST does).