Win with Technology Leadership. Leap Ahead

PDA for SI Analysis in LTI Systems

A VHDL-AMS Test Case

Arpad Muranyi
アルパづ ムラニ

Michael Mirmak
マイケル マ―マク

Intel Corporation
インテルコ―ポレ―ション

IBIS Summit
Tokyo, Japan
October 31, 2006

Legal Disclaimers

THIS DOCUMENT AND RELATED MATERIALS AND INFORMATION ARE PROVIDED "AS IS" WITH NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE. INTEL ASSUMES NO RESPONSIBILITY FOR ANY ERRORS CONTAINED IN THIS DOCUMENT AND HAS NO LIABILITIES OR OBLIGATIONS FOR ANY DAMAGES ARISING FROM OR IN CONNECTION WITH THE USE OF THIS DOCUMENT.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

Intel may make changes to specifications, product descriptions, dates and plans at any time, without notice.

*Other names and brands may be claimed as the property of others

Background

The IBIS Advanced Technology Modeling Group is now discussing adding channel analysis support to IBIS

Key points of the discussion include:

- Is a model API needed to support today's data processing designs?
- If so, for which language? *-AMS? C? Other?

http://www.vhdl.org/pub/ibis/summits/jul06/wang.pdf

This experiment attempts to show channel analysis using VHDL-AMS

- Channel analysis & LTI theory summarized by Bryan Casper of Intel

http://download.intel.com/education/highered/signal/ELCT865/Class2_15_16_Peak_Distortion_Analysis.ppt

Simulation setup

A simple circuit was used to generate a reasonable pulse response

The “channel” consists of

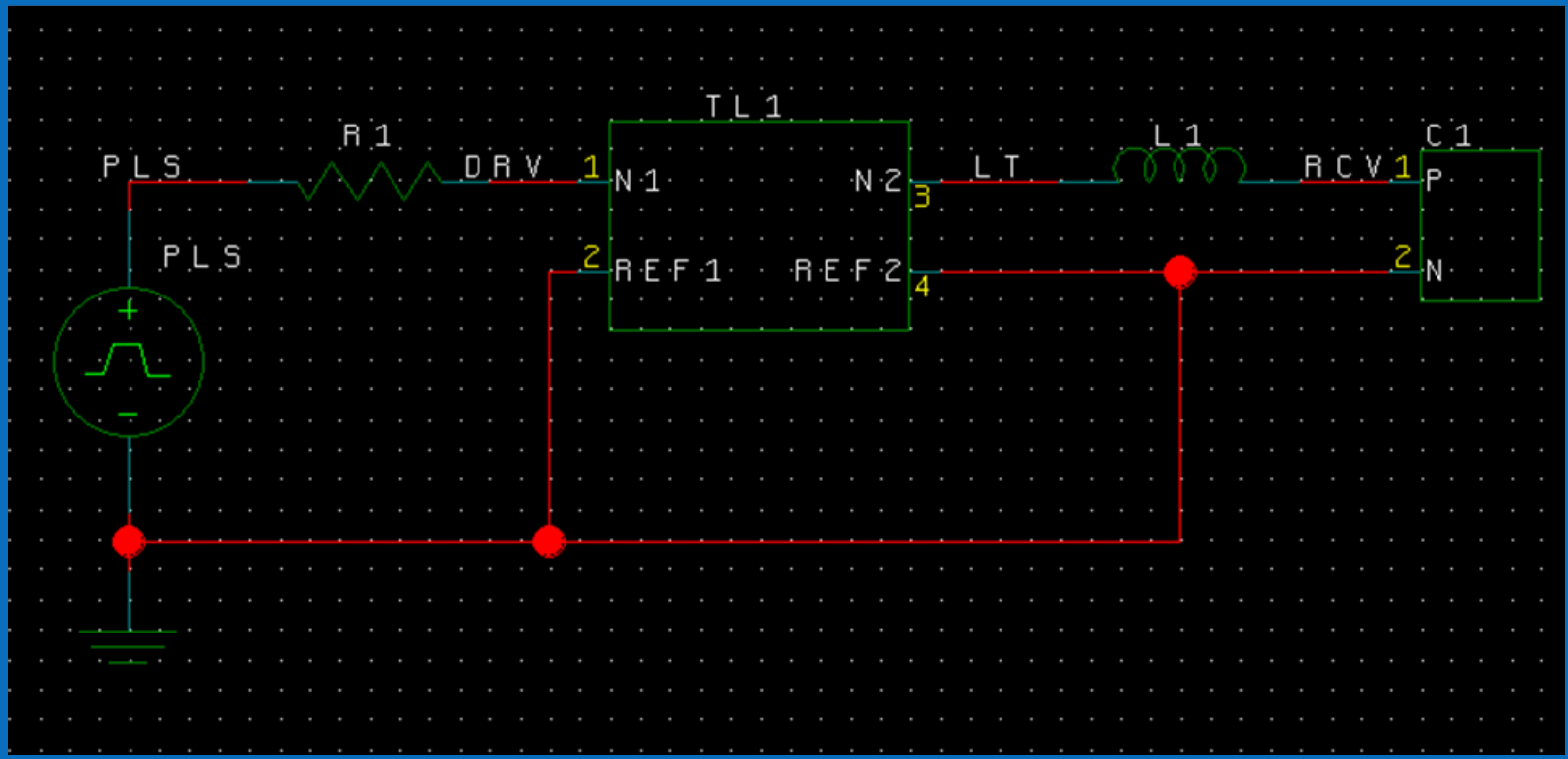
- A Thevenin driver using a pulse voltage source and a resistor,
- An ideal T-line
- An inductor and capacitor as the “receiver”

The channel was not meant to be realistic

The capacitor model has two “architectures”

- A normal capacitor
- A normal capacitor plus the PDA algorithm

Schematic of the simulated circuit



Pulse: 1 V, 200 ps wide, 1 ps edge
R1: 100 Ω
T-line: 100 Ω, 0.5 ns, lossless
L1: 10 nH
C1: 1 pF

Overview of the operation of the model

A normal time domain simulation is started

- Here, the first 3 ns of the simulation generate the pulse response
- The points of the pulse response waveform are stored in a “real_vector” (3000 points in this example)
- A 1 ps fixed time step is used for simplicity

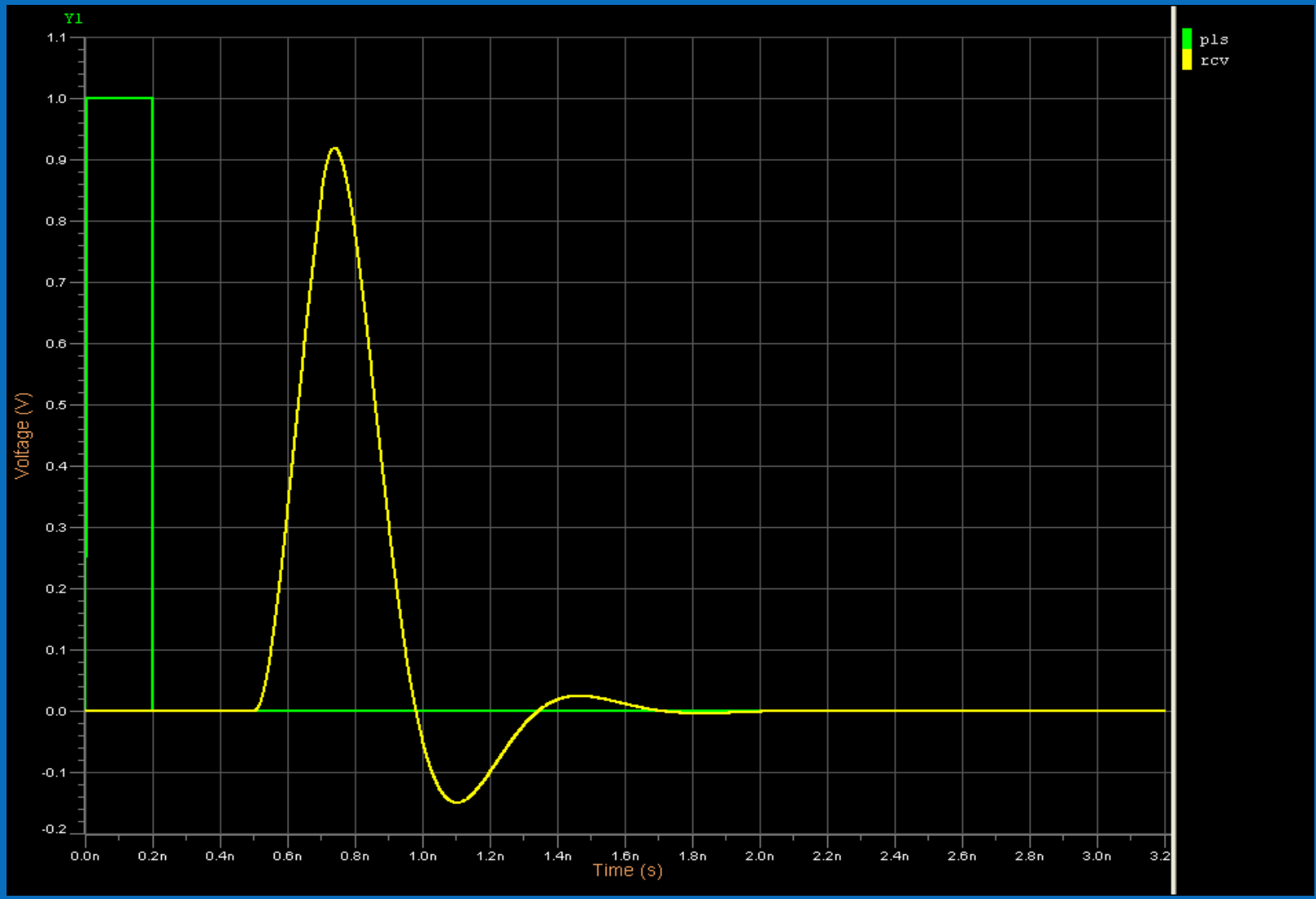
At the 3 ns time point of the simulation…

- A digital process begins (fast!)
- The process finds the peak of the pulse response, identifying the index of that point in the vector
- It finally generates an upper and lower eye contour using the methodology described in Bryan Casper’s presentation

The time domain simulation is continued for another 200 ps

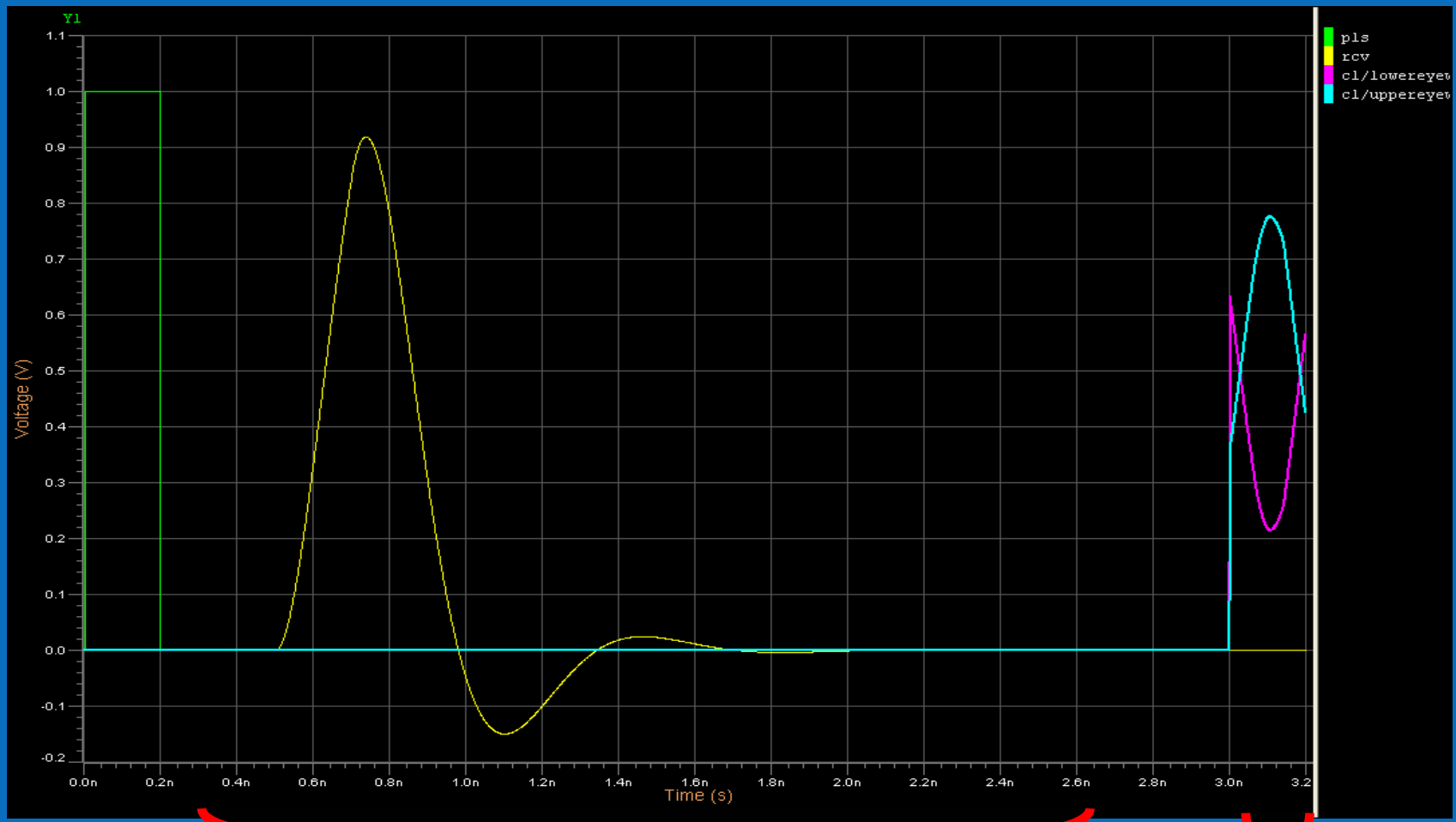
- The upper and lower eye contour vectors are plotted

Pulse response of the circuit



Win with Technology Leadership. Leap Ahead

Adding PDA post processing

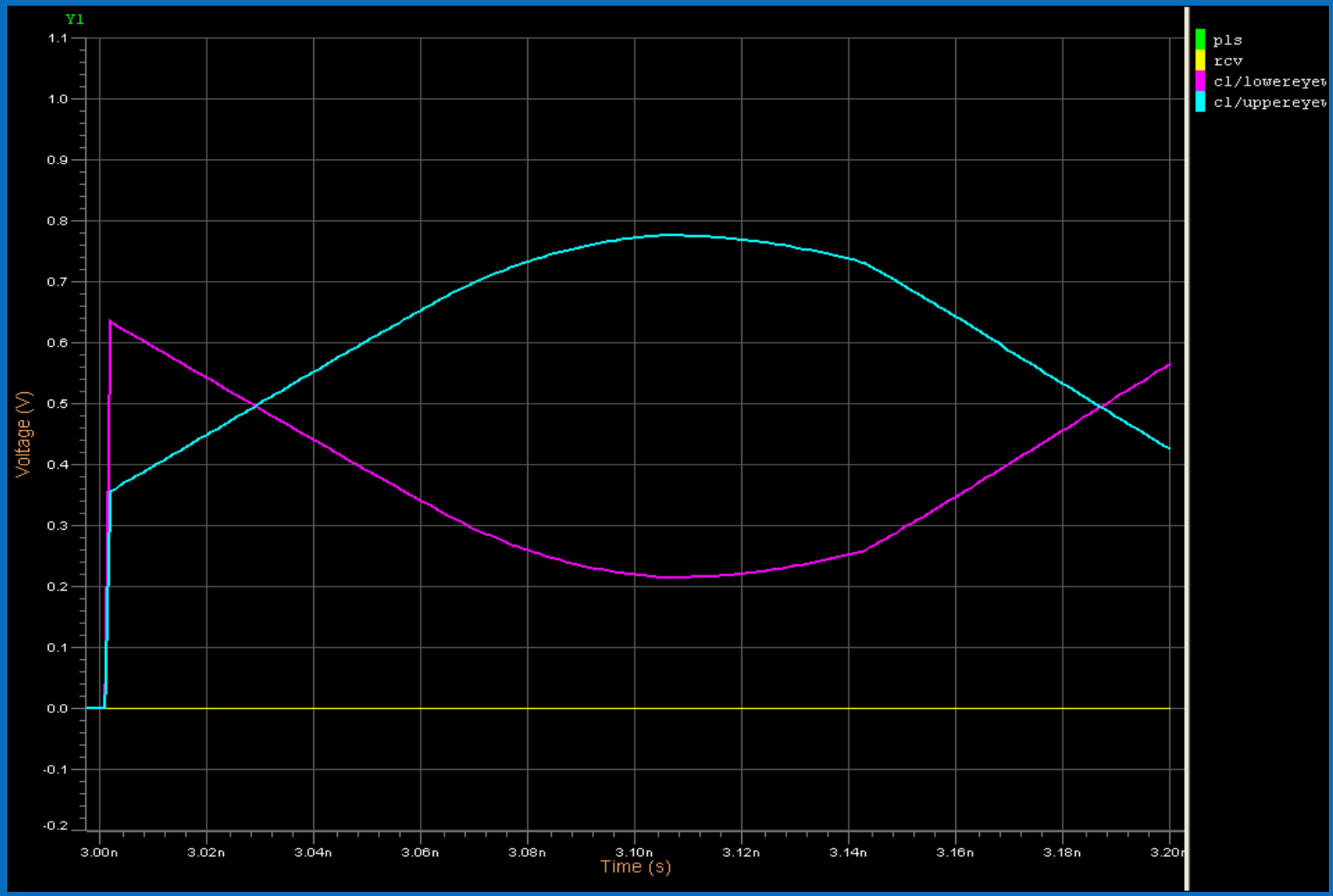


Pulse response TD simulation

PDA result "worst eye"

Win with Technology Leadership. Leap Ahead

Zoom: Worst eye plot



Win with Technology Leadership. Leap Ahead

Summary, benchmarks and future work

This experiment implemented the basic equations of PDA only

- No crosstalk
- No jitter
- No BER or more complex analyses included

Benchmarks with a 3000 point pulse response

- Approximately 230 ms CPU time without PDA algorithm
- Approximately 240 ms CPU time with PDA algorithm

Future work

- Show that more computationally intensive statistical analyses can also be implemented
- Implement same algorithm in Verilog-A(MS)

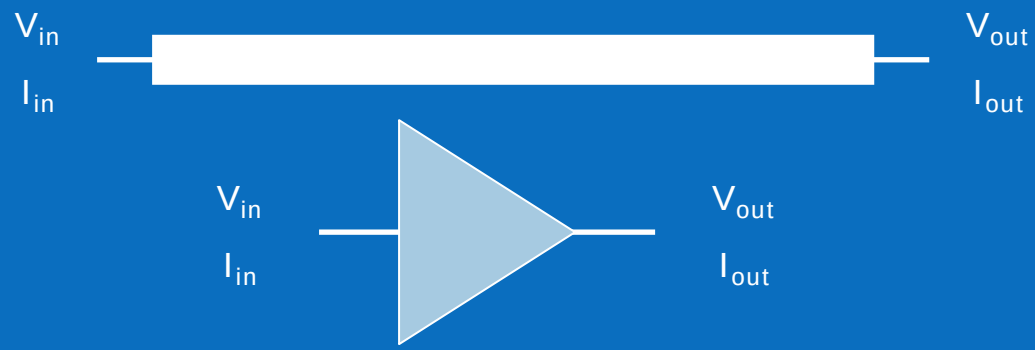
This experiment illustrates that a *-AMS language can be used for “signal processing” algorithms

Key questions remain…

PDA methods require “LTI” (linear time-invariant) systems.

How can we check buffer LTI?
Are common buffer designs truly LTI?

What is input and response?

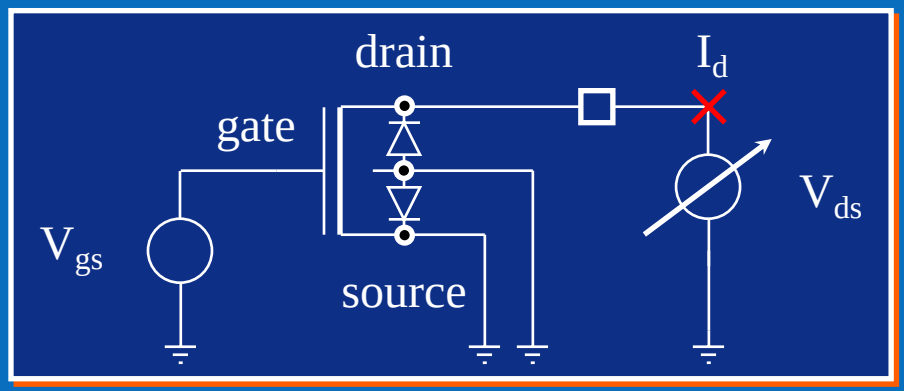


For S_{11} (Z_{11}, Y_{11}, etc…)

- if input is V_{in}, the response I_{in},
- if input is V_{out}, the response is I_{out},
- if input is I_{in}, the response is V_{in},
- if input is V_{out}, the response is I_{out}.

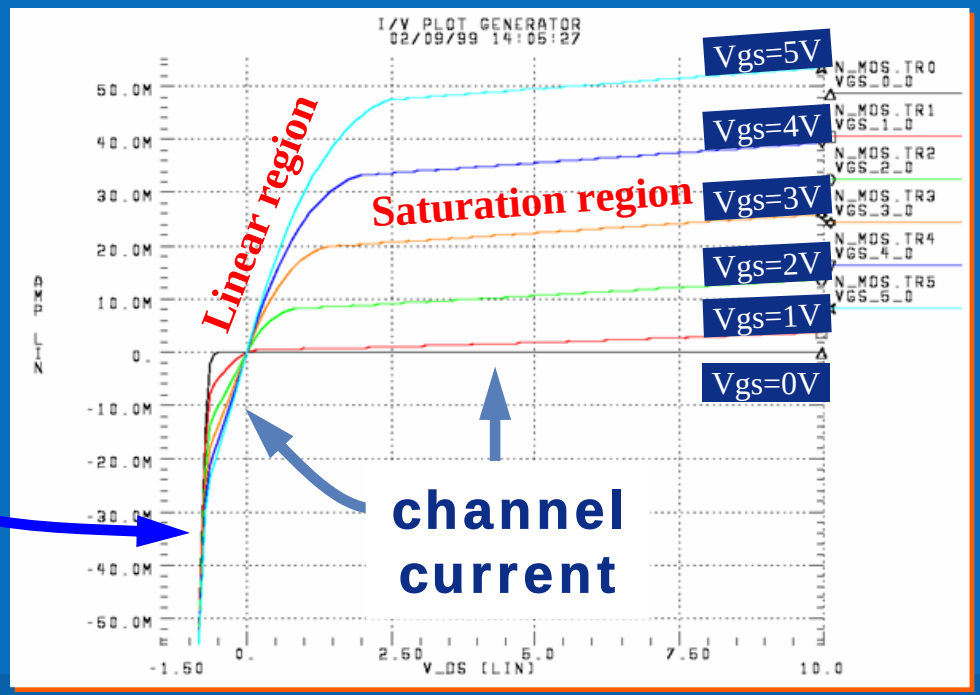
When we analyze a T-line with a driver and receiver for ISI effects, we must use the “11” parameters of the driver and receiver (V_{out}/I_{out}, I_{out}/V_{out}, or V_{in}/I_{in}, I_{in}/V_{in}, respectively) in response/input = transfer_function.

What is a non-linear driver (or receiver)?



If the input is V_{ds} and the response is I_d, we can see that the relationship is not linear.

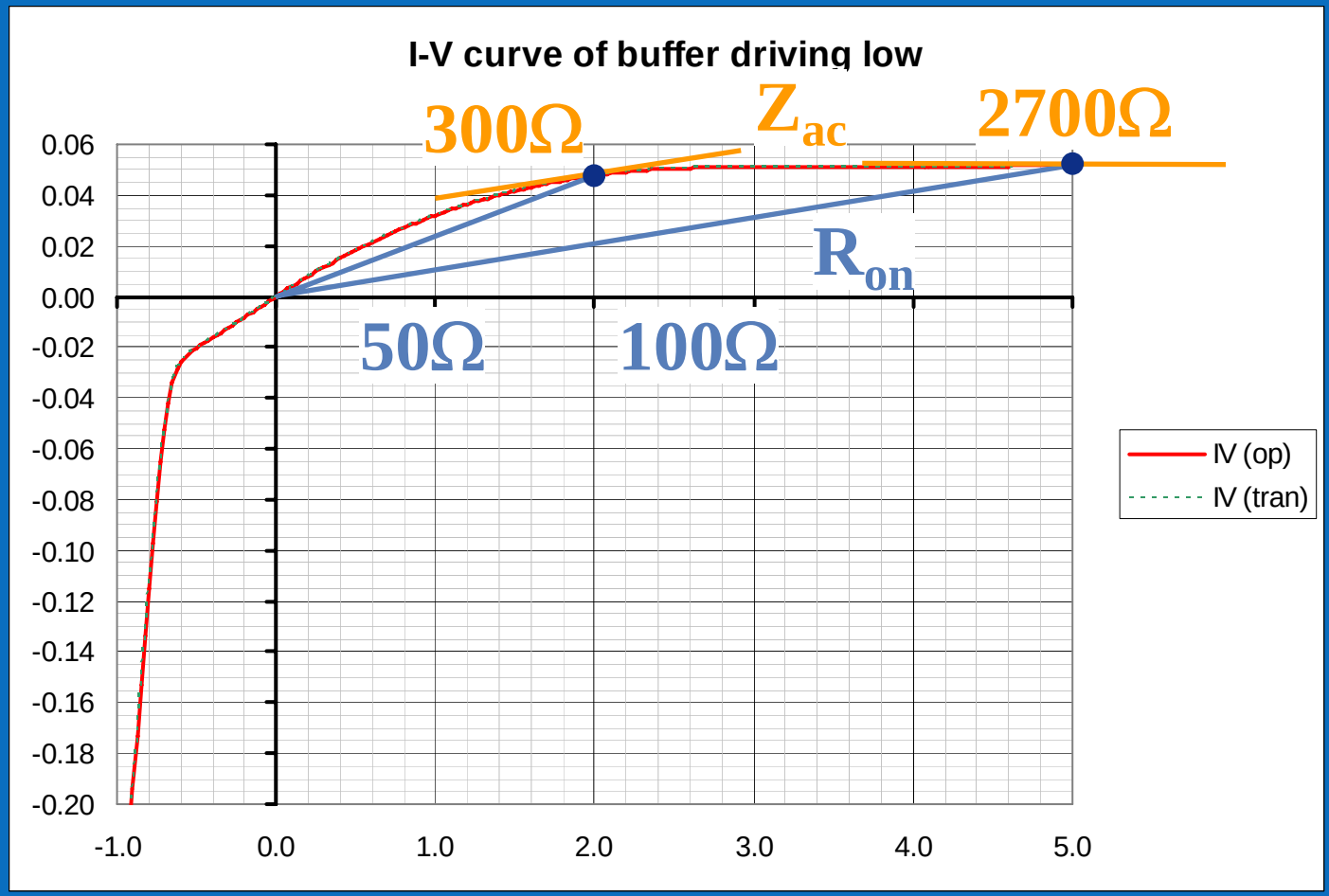
diode current



Even the so-called "linear region" is not perfectly linear.

Win with Technology Leadership. Leap Ahead

Reflections and FD analysis determined by Z_{ac} (not R_{on})

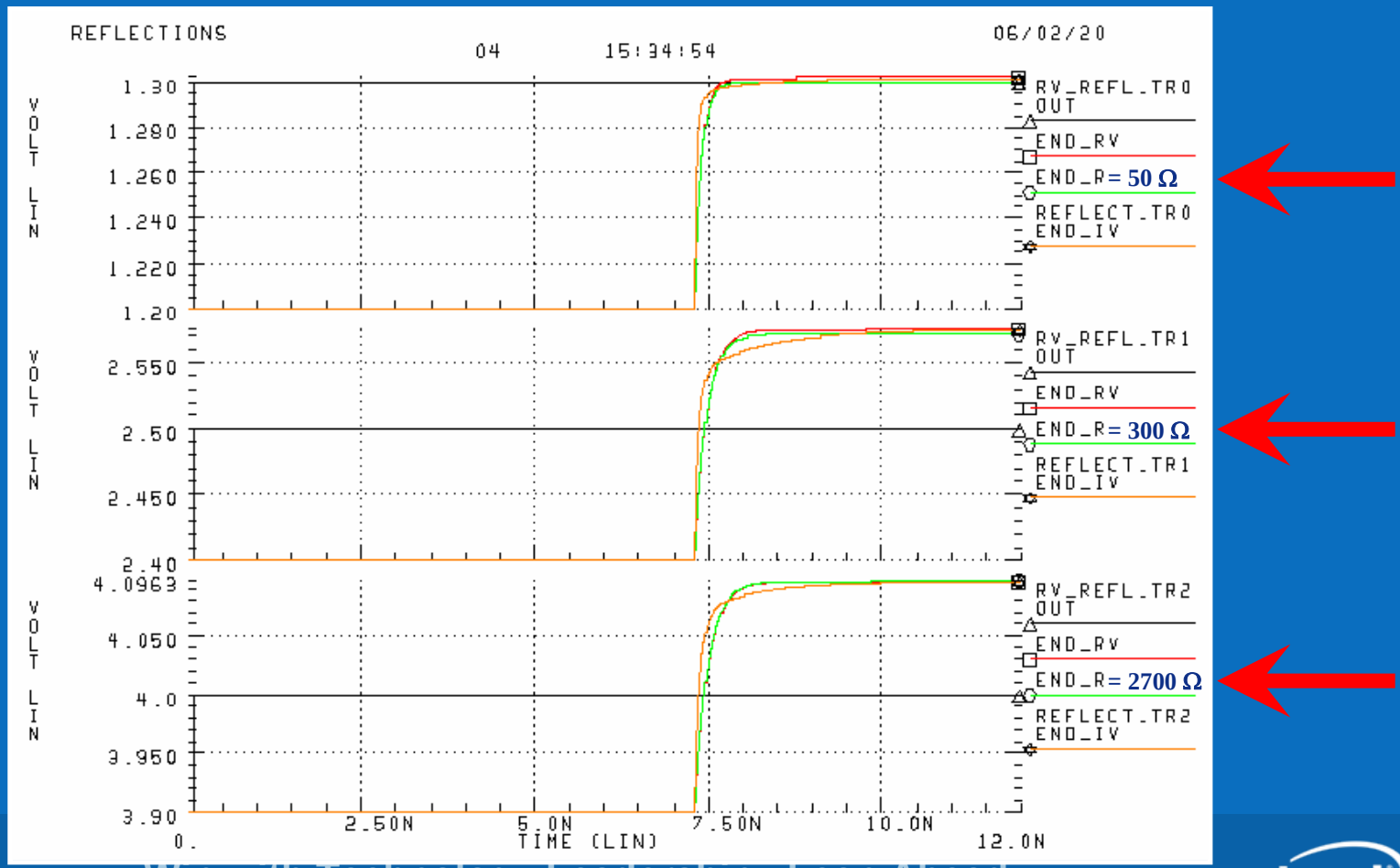


$dR_{on} \neq Z_{ac}$!!!

Win with Technology Leadership. Leap Ahead

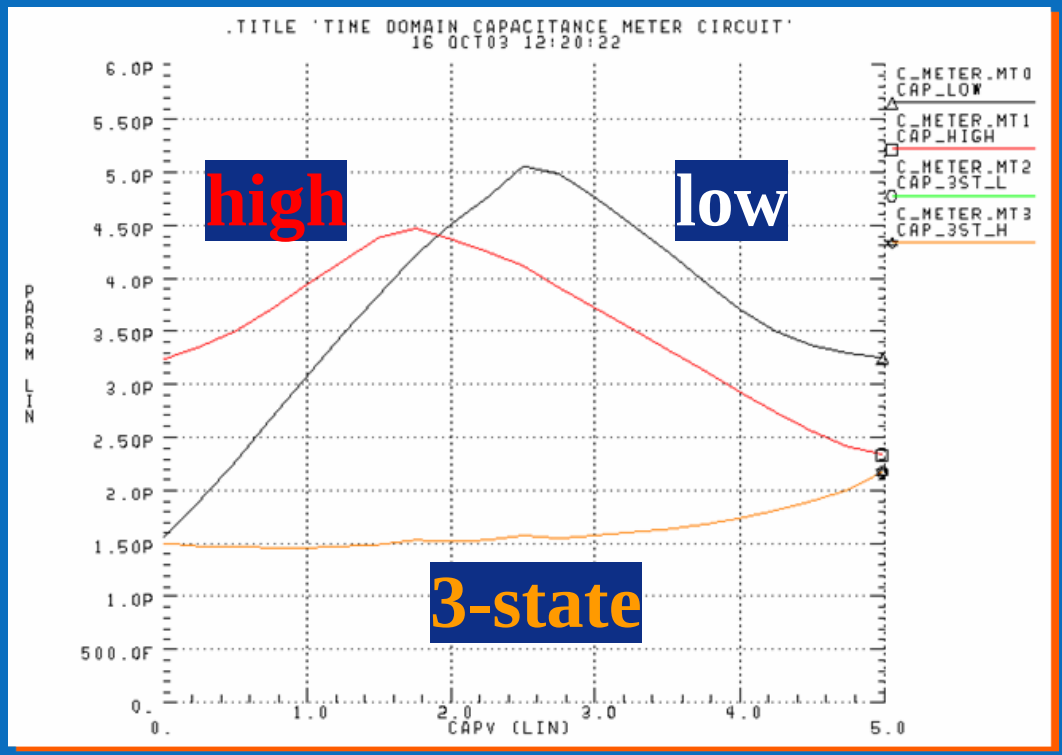
Proof with a time domain simulation

V_{in} = 100 mV step centered around 1.25, 2.45, 3.95 V_{dc}



Win with Technology Leadership. Leap Ahead

Capacitance can be non-linear and state dependent



C(V) means non-linear
C(state) means state dependent or time-variant

Win with Technology Leadership. Leap Ahead

Buffers and LTI Requirements

Our VHDL-AMS model driver was LTI:

- Resistive
- Same impedance across all times and states

Realistic drivers tend not to be LTI, without special design:

- Transistors usually have non linear IV curves
- High and low states may have different impedances
- Impedance may not be constant during transitions
- Impedance may change from bit to bit (data pattern dependence)

Therefore, toggling the driver to generate the pulse response may make the system non-LTI

Is it possible to generate a pulse (or step) response with a time domain simulation without violating the LTI requirement?

Backup

VHDL-AMS code

Code - processes (digital equations)

```
begin
    ------------------------------------------------------------------------
    GetCursorIndex : process is
    begin
      wait for 3.0e-9;
      CursorIndex <= FindCursorIndex(Wfm);
      wait;
    end process GetCursorIndex;

    GetEye : process is
    begin
      wait on CursorIndex;
      --report "Cursor index is: " & integer'image(CursorIndex);
      UpperEye <= EyeContour(Wfm, CursorIndex, BitWidthPts, "U");
      LowerEye <= EyeContour(Wfm, CursorIndex, BitWidthPts, "L");
    end process GetEye;

    Ticker : process is
    begin
      wait for 1.0e-12;
      if (Count < WfmPts) then
        Wfm(Count) <= Vout;
        Count      <= Count + 1;
      elsif (EyeIndex < BitWidthPts) then
        EyeIndex <= EyeIndex + 1;
      end if;
    end process Ticker;
    ------------------------------------------------------------------------
```

Code - analog equations

```
-------------------------------------------------------------------------
    break on Count, EyeIndex;

    if (domain = quiescent_domain) use
      Vout == V0;
    else
      Iout == Scale * Cval * Vout'dot;
    end use;

    if (now > 3.0e-9) use
      UpperEyeV == UpperEye(EyeIndex);
      LowerEyeV == LowerEye(EyeIndex);
    else
      UpperEyeV == 0.0;
      LowerEyeV == 0.0;
    end use;

  end architecture PDA_on;
-------------------------------------------------------------------------
```

Code - function FindCursorIndex

```
------------------------------------------------------------------------------
function FindCursorIndex (Wfm : real_vector) return integer is
------------------------------------------------------------------------------
  variable Index : integer := 0;
  variable Value : real    := 0.0;
------------------------------------------------------------------------------
begin
  for i in Wfm'range loop
    if i = 0 then
      Value := Wfm(i);
    else
      if Wfm(i) > Value then
        Value := Wfm(i);
        Index := i;
      end if;
    end if;
  end loop;
--  report "Index: " & integer'image(Index);

  return Index;
------------------------------------------------------------------------------
end function FindCursorIndex;
------------------------------------------------------------------------------
```

Code - (essentials of) function EyeContour

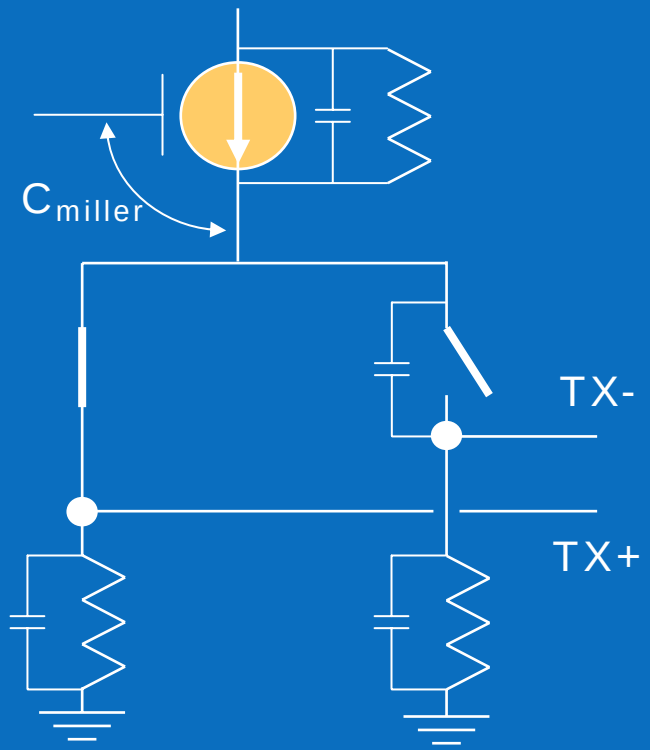
```
--------------------------------------------------------------------------------
function EyeContour (Wfm         : real_vector;
                     CursorIndex : integer;
                     BitWidth    : integer;
                     EyeSelector : string := "L") return real_vector is
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
begin
  while (CursorIndex - BitWidth/2 - i*BitWidth) > 0 loop
    i := i + 1;
  end loop;
  StartIndex := CursorIndex - BitWidth/2 - (i-1)*BitWidth;

  i := 0;
  while (StartIndex + (i+1)*BitWidth) <= Wfm'right loop
    if StartIndex + i*BitWidth + BitWidth/2 = CursorIndex then
      if (EyeSelector = "U") then
        for j in EyeContour'range loop
          EyeContour(j) := EyeContour(j) + Wfm(StartIndex + j-1 + i*BitWidth);
        end loop;
      end if;
    else
      if (EyeSelector = "U") then
        for j in EyeContour'range loop
          EyeContour(j) := EyeContour(j) + realmin(0.0, Wfm(StartIndex + j-1 + i*BitWidth));
        end loop;
      else
        for j in EyeContour'range loop
          EyeContour(j) := EyeContour(j) + realmax(0.0, Wfm(StartIndex + j-1 + i*BitWidth));
        end loop;
      end if;
    end if;
    i := i + 1;
  end loop;

  return EyeContour;
--------------------------------------------------------------------------------
end function EyeContour;
--------------------------------------------------------------------------------
```

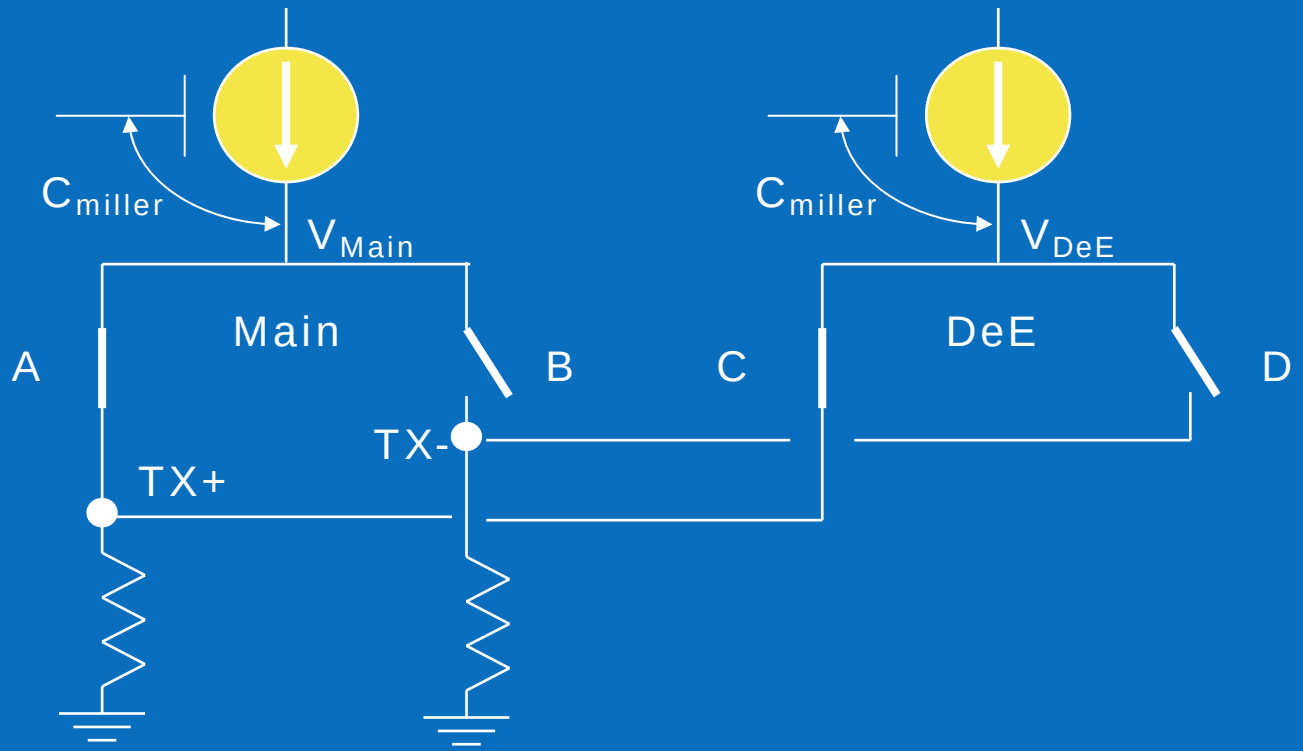
Buffer Variation Impact

State dependent driver impedance



The equivalent impedance looking into TX+ and TX- are not the same, i.e. a logic “1” has a different impedance from a logic “0”.

Buffer impedance may vary from bit-to-bit



Miller capacitance may modulate the bias circuit of the current source making it data pattern dependent (time-variant)

Win with Technology Leadership. Leap Ahead

*Other names and brands may be claimed as the property of others